எளிய தமிழில் BIG DATA

து. நித்யா

எளிய தமிழில் Big Data

து.நித்யா

‘Data is the new Oil’ என்பது புதுமொழி. இணைய தளங்கள், கைபேசி செயலிகள் யாவும் தம் பயனரின் அனைத்து செயல்களையும் தகவல்களையும் சேமித்து வருகின்றன. இவ்வாறு சேமிப்பதும், அவற்றில் இருந்து பயனுள்ள தகவல்களை தேடி எடுப்பதும், சில ஆண்டுகளுக்கு

முன் சாத்தியமே இல்லை. குறைந்து வரும் வன்பொருள் விலையும், சிறந்த கட்டற்ற மென்பொருட்களும் இணைந்து, பல்லாயிரம் சாத்தியங்களுக்கும், சாதனைகளுக்கும் வழிவகுத்துள்ளன.

Big Data — பெருந்தரவு. இதை **Mainframe, Super Computer** போன்ற எந்த சிறப்பு கட்டமைப்புகளும் இன்றி, நமது கணினிகள், மடிக்கணினிகள் கொண்டே, **Cluster** உருவாக்கி, **Elasticsearch, Hadoop, Spark**

போன்ற கட்டற்ற மென்பொருட்களை நிறுவி, கற்கவும், செயல்படுத்தவும் முடியும்.
இவற்றை, இந்த நூல் எளிமையாக அறிமுகம் செய்கிறது.

தமிழில் கட்டற்ற மென்பொருட்கள் பற்றிய தகவல்களை "கணியம்" மின் மாத இதழ், **2012** முதல் வெளியிட்டு வருகிறது. இதில் வெளியான **Bigdata** பற்றிய கட்டுரைககளை இணைத்து ஒரு முழு புத்தகமாக வெளியிடுவதில் பெரு மகிழ்ச்சி கொள்கிறோம்.

உங்கள் கருத்துகளையும், பிழை திருத்தங்களையும் editor@kaniyam.com க்கு மின்னஞ்சல் அனுப்பலாம்.

http://kaniyam.com/learn-bigdata-in-tamil-ebook என்ற முகவரியில் இருந்து இந்த நூலை பதிவிறக்கம் செய்யலாம். உங்கள் கருத்துகளையும் இங்கே பகிரலாம். படித்து பயன் பெறவும், பிறருடன் பகிர்ந்து மகிழவும் வேண்டுகிறோம்.

கணியம் இதழை தொடர்ந்து வளர்க்கும் அனைத்து அன்பர்களுக்கும் எமது நன்றிகள்.

த.சீனிவாசன்
tshrinivasan@gmail.com

ஆசிரியர்
கணியம்
editor@kaniyam.com

எளிய தமிழில் Big Data

முதல் பதிப்பு ஜூன் 2018

ஆசிரியர் - து.நித்யா - nithyadurai87@gmail.com

பிழை திருத்தம்: த.சீனிவாசன் - tshrinivasan@gmail.com

அட்டைப்படம், வடிவமைப்பு: த.சீனிவாசன்

- திருத்தி எழுதி வெளியிடலாம்.
- வணிக ரீதியிலும்யன்படுத்தலாம்.

ஆனால், மூலப் புத்தகம், ஆசிரியர் மற்றும் **www.kaniyam.com** பற்றிய விவரங்களை சேர்த்து தர வேண்டும். இதே உரிமைகளை யாவருக்கும் தர வேண்டும். கிரியேடிவ் காமன்ஸ் என்ற உரிமையில் வெளியிட வேண்டும்.

நூல் மூலம் :

http://static.kaniyam.com/ebooks/learn-bigdata-in-tamil/learn-bigdata-in-tamil.odt

பொருளடக்கம்

முன்னுரை

பெரும்தரவு (Bit Data) என்பது தற்போது அதிகமாக வளர்ந்து வரும் தொழில்நுட்பம். . ஆனால் இத்துறையில் அனுபவம் கொண்டவர்கள் மிகவும் குறைவே.! எனவே இத்துறையில் உள்ள சில பல முக்கியக் கருவிகளைப் பற்றி அடிப்படை அறிவு வளர்த்துக் கொண்டால் போதும். இத்துறையில் நுழைந்து நாம் வல்லுனராகிவிடலாம். இதனை மனதில் கொண்டு இப்புத்தகத்தில் நாம் ELK Stack, Hadoop மற்றும் Spark ஆகிய கருவிகளைப் பற்றிக் கற்கப் போகிறோம். பெரும் தரவுகளில் "நிகழ்காலத் தரவுகளைக்

கையாளுவது" மற்றும் "வரலாற்றுத் தரவுகளைக் கையாளுவது" என்று இரண்டு விதங்கள் உள்ளன. உதாரணத்துக்கு twitter, facebook போன்ற நிறுவனங்களுக்கு ஒவ்வொரு நிமிடமும் தரவுகள் வந்த வண்ணம் இருந்து கொண்டே இருக்கும். எனவே இத்தகைய நிறுவனங்களுக்கு நிகழ்காலத் தரவுகளைக் கையாளுவதற்கான தேவை அதிகமாக இருக்கும். இதையே 'Real Time Analysis' என்போம். வேறு சில நிறுவனங்கள் பழைய தரவுகளை எடுத்து அதனை ஆய்வு செய்து அதனடிப்படையில் தேர்ந்த முடிவுகளை எடுக்கும். இதையே 'Historical Analysis' என்று கூறுவர். நிகழ்காலத் தரவுக்கு ELK Stack-ஐயும் வரலாற்றுத் தரவுக்கு hadoop-ஐயும் பார்க்கப் போகிறோம். மேலும் spark எவ்வகையில் hadoop-ஐ விட மேன்மையான

செயல்பாடுகள் கொண்டு விளங்குகிறது என்பதையும் இதில் காணலாம்.

ஆசிரியர் உரை

சமீப காலத்தில் நான் ஒருசில காரணங்களால் இருக்கும் வேலையை விட்டுவிட்டு புதிய வேலை தேடிக் கொண்டிருந்தேன். இது உண்மையிலேயே மிகவும் கடினமான காலகட்டமாக எனக்கு இருந்தது. 10 வருடமாக ‘Datawarehouse Testing' துறையில் வேலை பார்த்துக் கொண்டிருந்த நான், இப்போது வேலை தேடும்போது என்னுடைய இத்தனை வருட அனுபவத்திற்கான வாய்ப்பு மிகவும் குறைந்து விட்டதை உணர்ந்தேன். பெரும் நிறுவனங்களில் இருந்து எனக்கு நேர்காணல்

அழைப்புகள் வந்து கொண்டே இருந்தாலும், என்னவோ தெரியவில்லை. அனைத்து சுற்றுகளிலும் நான் தேர்வாகினாலும், கடைசி சுற்றில் நிராகரிக்கப்பட்டுவிடுவேன். எனது 10 வருட அனுபவம் பிரச்சனையா, அல்லது குறைந்த அனுபவம் கொண்டவர்களையே அதிகம் தேர்வுசெய்கிறார்களா, அல்லது என்னிடம் இன்னும் நிறைய திறமைகளை எதிர்பார்க்கிறார்களா என்று எனக்குத் தெரியவில்லை. ஆனால் ஒன்று மட்டும் நன்றாக விளங்கியது. என்னுடைய Manual மற்றும் ETL Testing அனுபவத்தை மட்டும் வைத்துக் கொண்டு எனக்கான ஒரு நல்ல வேலையைத் தேடுவது என்பது எட்டாக் கனியாகத் தோன்றியது. என்னுடைய திறமைகளை வளர்த்துக் கொள்வததைத் தவிர எனக்கு வேறு வழியே தோன்றவில்லை. இந்த சமயத்தில்தான் ஏதாவதொரு நிறுவனத்தில் சேர்ந்து Bigdata

பற்றிய படிப்பினைப் படித்து முடித்துவிட்டு மீண்டும் வேலை தேடலாம் என்று யோசித்தேன். ஆனால் அதற்கான பயிற்சித் தொகையோ மிகவும் அதிகமாக இருந்தது. கிட்டத்தட்ட ஒரு லட்சம் ரூபாய் கேட்டார்கள்.. இவ்வளவு ரூபாய் கட்டிப் படித்தாலும் வேலைக்கு உத்தரவாதம் இல்லை. எனவே இதிலும் மனம் செல்லவில்லை.

இச்சூழ்நிலையில் தான் எந்த ஒரு பயிற்சி நிறுவனமும் போகமால் Big data-பற்றிய விஷயங்களை என் சொந்த முயற்சியில் நானே படிக்க ஆரம்பித்தேன்.. இத்தொழில்நுட்பத்தில் உள்ள ஒருசில முக்கியக் கருவிகளைக் கற்றேன்.. இதுவே எனக்கு நல்ல நல்ல வாய்ப்புகள் கிடைக்க வழிவகை செய்தது. கடும் பயிற்சிக்குப் பிறகு, எனது சுய விவர

அறிக்கையில் Elasticsearch, hadoop-ஐ சேர்த்தேன். நேர்காணல்களில் Hadoop-ஐப் பற்றிப் பேச ஆரம்பித்தேன். மேலும் பல பயிற்சிகள், கற்றல்களள், நேர்முகத் தேர்வுகளுக்குப் பின், இப்போது ஒரு நிறுவனத்தில் IOT எனும் துறையில் தேர்வாகியுள்ளேன்.. இதுவரை நான் கற்றவற்றை இப்புத்தகத்தில் எழுதியுள்ளேன். இத்தனை இப்போது உங்கள் அனைவரிடமும் பகிர்ந்து கொள்ள விரும்புகிறேன்.

நான் BigData கற்ற போது எழுதிய குறிப்புகளைக் கொண்டு கணியம் இதழில் தொடராக எழுதினேன். அதே

தொடர் இப்போது மின்னூலாக வெளியிடப்படுவது மிக்க மகிழ்ச்சி.

தமிழில் கணிணி நுட்பங்களைப் பகிர, ஒரு களமாக உள்ள 'கணியம்' தளத்தில், இதுவரை வெளியான எனது மின்னூல்களுக்கு வாசகர்கள் தரும் ஆதரவு பெருமகிழ்ச்சி அளிக்கிறது.

“தேமதுரத் தமிழோசை உலகெல்லாம் பரவும் வகை செய்தல் வேண்டும்”

“பிற நாட்டு நல்லறிஞர் சாத்திரங்கள் தமிழ் மொழியிற் பெயர்த்தல் வேண்டும்”

என்ற பாரதியின் விருப்பங்களை நிறைவேற்றுவதில், என் பங்களிப்பும் உள்ளது என்பதே, மிகவும் மகிழ்ச்சி.

தொடர்ந்து ஊக்கம் அளிக்கும் என் குடும்பத்தினருக்கும், கணியம் குழுவினருக்கும், FreeTamilEbooks.com குழுவினருக்கும், வாசகர்களுக்கும் நன்றிகள்.

து. நித்யா

சென்னை

28 ஜூன் 2018

மின்னஞ்சல்:
nithyadurai87@gmail.com

வலை பதிவு:
http://nithyashrinivasan.wordpress.com

1 Big Data - ஓர் அறிமுகம்

source - https://commons.wikimedia.org/wiki/File:BigData_2267x1146_white.png

நமது ஊரில் உள்ள பழக்கப்பட்ட மளிகைக் கடைக்குச் சென்று பொருட்கள் வாங்கும்போது, அந்தக் கடைக்காரருக்கு நம்மைப் பற்றிய விவரம் முழுவதும் தெரிந்திருக்கும். மேலும் அவர் நம்முடன் கொண்ட பழக்கத்தினால் நமக்கு எது பிடிக்கும் எது பிடிக்காது என்பதை சற்று கணித்து வைத்திருப்பார். எனவே நமது ரசனைக்கேற்ப அவரிடம் ஏதேனும் புது சரக்குகள் வந்து இறங்கியிருப்பின், அதனை நம்மிடம் காட்டி 'இது உங்களுக்கு மிகவும் பிடிக்கும். பயன்படுத்தித்தான் பாருங்களேன்"

என்பார். நாமும் “சரி! வாங்கித்தான் பார்ப்போமே!” என்று வாங்கிவிடுவோம். இதனால் அவரது வியாபாரமும் பெருகுகிறது, நமக்கும் நமக்கு பிடித்த பொருட்களை வாங்குவது எளிதாகிறது.

இது போன்ற விஷயங்களெல்லாம் இப்போது கணினி வழியே நடைபெறுகின்றன. இப்போதெல்லாம் நாம் கடைக்குச் சென்று பொருட்களை வாங்காமல் கணினி வழியே வாங்குகின்றோம். எனவே கடைக்காரருக்கோ கணினி வழியேயே நமது விவரங்களையும், நமது ரசனைகளையும் தெரிந்து கொள்ள வேண்டியது அவசியமாகிறது. உதாரணத்துக்கு Flipkart, Amazon போன்ற

வலைத்தளங்களில் நாம் ஒரு கைக்கடிகாரத்தை வாங்குகிறோம் எனில், பின்னர் நாம் அடுத்த முறை அந்த வளைத்தளத்திற்குச் செல்லும்போது, புதிய அழகழகான கைக்கடிகாரங்களெல்லாம் நமக்கு விளம்பரங்களாக வரும். நாம் எந்த பொருளின்மீது அதிக விருப்பம் காட்டுகிறோமோ, அந்த பொருளும் நம் கண்ணில் அடிக்கடி தென்படுமாறு வந்துபோகும். அதாவது அந்த கடைக்காரர் கணினி வழியேயே நமது ரசனையைத் தெரிந்துகொண்டார். இதுவே ‘Machine Learning’ என்று அழைக்கப்படும். இந்த Machine Learning-ன் அடிப்படையில் அமைவதே ‘Artificial Intelligence’ ஆகும். அதாவது நமது விருப்பு வெறுப்புகளைப் பற்றிய அறிவினை கணினி செயற்கை முறையில் பெற்றுவிடுகிறது. இவையெல்லாம் எப்படி

சாத்தியப்பட்டது என்று சிந்தித்துப் பார்த்தால் அவை அனைத்தும் Big Data-வின் அம்சங்களே !

ஆரம்ப காலகட்டத்தில் நமது தொழிலை திறம்பட நடத்துவதற்காக ஒருசில முக்கியமான விஷயங்களை எல்லாம் நாம் நினைவில் வைத்துக்கொண்டோம். பின்னர் நினைவில் வைத்துக்கொள்ள முடியாத அளவுக்கு விஷயங்கள் சேர்ந்து விடும்போது, அதனை ஒரு நோட்டில் எழுதி வைக்கத் தொடங்கினோம். பின்னர் நோட்டுகளும் பத்தவில்லையென்று கணினியில் விவரங்களை சேமித்து வைத்தோம். பின்னர் ஒரு கணினியால் சேமித்து வைக்கக் கூடிய எல்லையையும் தாண் டினோம். உதாரணத்துக்கு ஒரு நோட்டில் 2000 வரிகள் தான் எழுத முடியும் என்பது போல ஒரு கணினியிலும் 1TB வரைதான் data-வை

சேமிக்க முடியும் என்று இருக்கலாம். நாம் சேமிக்க வேண்டிய தகவலின் அளவு 1TB -ஐத் தாண்டும்போது, நமது இரண்டாவது கணினியில் சென்று தகவலைச் சேமிக்க ஆரம்பிப்போம். இதுவே ‘Clustered systems’ / ‘Distributed systems’ எனப்படும். நம்மிடம் 10 கணினிகள்தான் இருக்கிறதெனில் 10TB-வரைதான் நம்மால் data-வை சேமிக்க முடியும். அதுவே நாம் சேமிக்க வேண்டிய தகவலின் அளவு 10TB-ஐத் தாண்டும்போது, நாம் புதுப்புது கணினிகளை வாங்கி சேர்த்துக்கொண்டே போகாமல், சேமிப்பிற்கான இடத்தை மட்டும் ஒருசில நிறுவனங்களிடமிருந்து பெற்றுக்கொண்டு பயன்படுத்தினோம். இதுவே ‘Cloud Storage’ எனப்படும். அதாவது இத்தகைய cloud computers-ஐ நம்மால் பார்க்க

முடியாது (virtual). ஆனால் நமது கணினியைப் போன்றே அதிலும் அனைத்து வேலைகளையும் செய்யலாம். Amazon, Google, Cloudera போன்றவை இத்தகைய services-ஐ வழங்குகின்றன.

ஒரு நோட்டில் எழுதக் கூடிய அளவுக்கு அனைத்து தகவல்களும் இருந்த மட்டும், நமக்கு வேண்டிய ஒருசில தகவல்களை தேடி எடுப்பது, அதனை ஆராய்வது, அதனடிப்படையில் முடிவுகளை எடுப்பது என்பது போன்ற விஷயங்களெல்லாம் நம்மால் சாதாரணமாக செய்ய முடிந்தது. ஆனால் தற்போதோ பல்வேறு முறையில் இணைக்கப்பட்ட கணினிகளிலிருந்து கோடிக்கணக்கான தகவல்களை அலசி ஆராய்ந்து அதனடிப்படையில் நமது வளர்ச்சிக்குத் தேவையான முடிவுகளை எடுப்பதற்கு

உதவுவதே Big Data ஆகும். இது பெரும்பாலும் மின்வர்த்தகம் மற்றும் சமூக ஊடகத் துறைகளில் பெரும்பான்மையான பொறுப்புகளை ஏற்கிறது.

மேலும் வானியல், பொருளாதாரம், வேதியியல், போக்குவரத்து, ஆராய்ச்சி போன்ற பலதரப்பட்ட துறைகளிலும் இப்போது Big Data என்பது தலையெடுக்கத் தொடங்கியுள்ளது. ஒவ்வொரு துறையும் அதனதன் வளர்ச்சிக்காகவும், திறம்படச் செயல்புரிவதற்காகவும் மிக மிக நுண்ணிய தகவல்களையெல்லாம் இப்போது சேமித்து வைக்கத் துவங்கியுள்ளது. இத்தகைய எண்ணிலடங்கா தகவல்களையெல்லாம் எங்கு சேமித்து வைப்பது (Data storage), அதனை எவ்வாறு தரம் பிரிப்பது (Data Mapping),

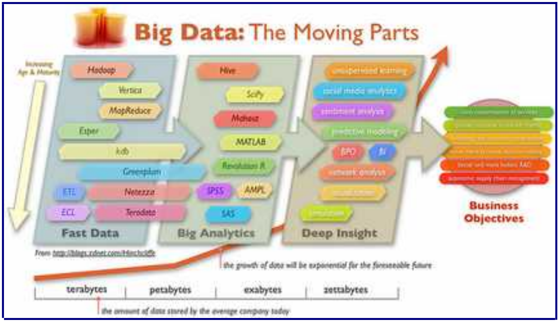
மிக முக்கிய தகவல்களை மட்டும் எவ்வாறு மொத்த தகவல்களிலிருந்து பிரித்தெடுப்பது (Data Mining), பிரித்தெடுத்த தகவல்களை எவ்வாறு ஒன்றோடொன்று இணைத்து (Data Pipeline)அர்த்தமுள்ள தரவுகளாக மாற்றுவது, அதனை எவ்வாறு பயனருக்கு சமர்ப்பிப்பது (Visualization of Reports) போன்ற அனைத்து வேலைகளையும் “பெரிய தரவு” எனும் “Big Data” புரிகிறது.

இந்த பெரிய தரவானது ஒழுங்கான வடிவத்தில் உள்ள விவரங்களை சேமிப்பத்தோடு மட்டுமல்லாமல், ஒழுங்கற்ற வடிவத்திலிருக்கும் விவரங்களையும் சேமிக்கும் வல்லமை கொண்டது. அதாவது ஒருவரின் பெயர், வயது, ஊர், மின்னஞ்சல் முகவரி போன்றவை எல்லாம் ஒழுங்கான வடிவத்தில் உள்ள தரவுகள் (Structured data). ஆனால்

facebook போன்ற வலைத்தளத்தில் ஒரு பயனரின் அடிப்படை விவரங்களோடு சேர்த்து அவரின் மனநிலை , விருப்பு வெறுப்புகள், செயல்பாடுகள், விமர்சனங்கள் போன்ற அனைத்து விதமான தகவல்களையும் சேமிக்க வேண்டும். இவையெல்லாம் எந்த வடிவத்தில் வேண்டுமானாலும் இருக்கலாம். ஒருவர் தனது மனநிலையை வார்த்தைகளாகவும் வெளிப்படுத்தலாம், படங்களாகவும் வெளிப்படுத்தலாம். எனவே இவை எந்த வடிவத்தில் இருக்கும் என்பதை நம்மால் கணிக்க முடியாது. இவையே ஒழுங்கற்ற வடிவத்தில் இருக்கும் தகவல்களாக (Unstructured data) சேமிக்கப்படுகின்றன.

பல்வேறு நிறுவனங்கள் பெரிய தரவில் கூறப்பட்டுள்ள ஒவ்வொரு விஷயத்தையும்

புரிவதற்காக பல்வேறு தொழில்நுட்பங்களைப் பயன்படுத்தி கருவிகளை உருவாக்குகின்றன. இறுதியில் அக்கருவிகளை ஒன்றாக இணைத்து ஒரு package-ஆக அந்நிறுவனம் வழங்குகின்ற பெரிய தரவுக்கான ஒரு கருவியாக வெளியிடுகின்றன. Hadoop, Spark, Druid, ELK ஆகியவை தற்போது சந்தையில் புகழ்பெற்று விளங்குகின்ற பெரிய தரவுக்கான திறந்த மூல மென்பொருள் கருவிகளாகும். இவற்றை Apache, Cloudera, Amazon போன்ற நிறுவனங்கள் வழங்குகின்றன.



source: https://www.flickr.com/photos/dionh/7550578346

YARN என்பது பல்வேறு கணினிகளில் சேமிக்கப்பட்ட விவரங்களை கையாளுவதற்கு (cluster management) உதவும் ஒரு மென்பொருள் பயன்பாடு ஆகும். DFS (distributed file system), MongoDB போன்றவை பல்வேறு விதங்களில் வருகின்ற

தகவல்களை சேமிக்க உதவும் சேமிப்புக் கிடங்குகள் ஆகும். Mapper என்பது முதல் நிலை தரவுகளை எடுத்து அவற்றை சுருக்கி (key, value) pairs-ஆக சேமிக்கும். Reducer என்பது Mapper-வெளிப்படுத்தும் இணைகளை மீண்டும் சுருக்கி ஒரு அர்த்தமுள்ள விதத்தில் சேமித்து வைத்துக்கொள்ளும். Mapper மற்றும் Reducer இவை இரண்டும் சேர்ந்து ‘MapReducer’ என்று அழைக்கப்படும். இது Java-வைப் பயன்படுத்தி எழுதப்பட்ட ஒரு library ஆகும். Pig & Hive ஆகியவை Map Reducer library- ஐப் பயன்படுத்துவதற்கு உதவும் மொழிகளாகும். இவற்றையெல்லாம் பயன்படுத்தி உருவாக்கப்பட்டதே Hadoop எனப்படும் ஒரு கட்டமைப்பு. Datameer, Tableau என்பது கடைசியாக மேல்மட்ட

அதிகாரிகள் அவர்கள் விரும்பும் விதத்தில் அறிக்கை எடுக்க உதவும் கருவிகள் ஆகும்.

அடுத்ததாக ELK என்பது Elastic search, Logstash & Kibana எனும் 3-ஐயும் இணைத்து உருவாக்கப்பட்ட கட்டமைப்பு ஆகும். இதில் Elastic search என்பது முதல் நிலை தகவல்களை சேமிக்க உதவும் ஒரு Engine ஆகும். Logstash என்பது கோப்பு வடிவத்திலோ அல்லது வலைத்தளத்திலோ இருக்கும் தகவல்களை Engine-க்குள் செலுத்த உதவும் கருவி ஆகும். Kibana என்பது Engine-ல் இருந்து தகவல்களை அறிக்கைக்கு தேவையான விதத்தில் தேடி எடுத்து வெளிப்படுத்த உதவும் கருவி ஆகும்.

இவ்வாறே Spark, Druid போன்றவை அதற்கென்று ஒவ்வொரு கட்டமைப்பைப் பெற்றுத் திகழுகின்றன. இனிவரும் பகுதிகளில் மேற்கூறியவற்றில் ஏதேனும் ஒரு கட்டமைப்பைப் பற்றி விரிவாகக் காணலாம்.

2 ELK Stack - ஓர் அறிமுகம்

ELK Stack என்பது Logstash, Elastic Search, Kibana எனும் 3 தனித்தனி திறந்த மூல மென்பொருள் கருவிகளின் கூட்டமைப்பு ஆகும். இவை முறையே 2009 , 2010, 2011 ஆகிய ஆண்டுகளில் தனித்தனி நபர்களால் உருவாக்கப்பட்டு தனித்தனி திறந்தமூலக் கருவிகளாக வெளிவந்து கொண்டிருந்தன. பின்னர் 2012-ஆம் ஆண்டு "Elastic Search" எனும் நிறுவனம் உருவாக்கப்பட்ட பின்னர், ஒவ்வொருவராக

அந்நிறுவனத்தில் இணைய, அவர்கள் உருவாக்கிய கருவிகளும் இணைக்கப்பட்டு "ELK Stack" எனும் பெயரில் பெரிய தரவுக்கான ஒரு கருவியாக வெளிவந்தன. இதில் Elastic search என்பது முதல் நிலை தகவல்களை சேமிக்க உதவும் ஒரு Engine ஆகும். Logstash என்பது கோப்பு வடிவத்திலோ அல்லது வலைத்தளத்திலோ இருக்கும் தகவல்களை Engine-க்குள் செலுத்த உதவும் கருவி ஆகும். Kibana என்பது Engine-ல் இருந்து தகவல்களை அறிக்கைக்கு தேவையான விதத்தில் தேடி எடுத்து வெளிப்படுத்த உதவும் கருவி ஆகும். இப்போது இவற்றில் உள்ள ஒவ்வொரு கருவிகளின் செயல்பாடுகளைப் பற்றியும் கீழே விளக்கமாகக் காணலாம்.

2.1 ELK-ஐ Ubuntu கணினியில் நிறுவுதல்

ELK இயங்குவதற்கு Java தேவை. பின்வரும் 3 கட்டளைகள் Java-வை install செய்யும்.

```
sudo add-apt-repository -y
ppa:webupd8team/java
sudo apt-get update
sudo apt-get -y install oracle-
java8-installer
```

இப்போது பின்வருமாறு கொடுத்து java முறையாக நிறுவப்பட்டுவிட்டதா என்று பார்க்கவும்.

```
java -version
```

வெளியீடு:

```
java version "1.8.0_131"
Java(TM) SE Runtime Environment
(build 1.8.0_131-b11)
```

```
Java HotSpot(TM) 64-Bit Server VM
(build 25.131-b11, mixed mode)
```

அடுத்ததாக கீழ்க்கண்ட 4 கட்டளைகளையும் ஒன்றன்பின் ஒன்றாக இயக்கவும்.

```
wget -qO -
https://artifacts.elastic.co
/GPG-KEY-elasticsearch |
sudo apt-key add -
```

```
sudo apt-get install apt-transport-https

echo "deb https://artifacts.elastic.co/packages/5.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-5.x.list


sudo apt-get update && sudo apt-get install elasticsearch logstash kibana
```

இப்போது ELK வெற்றிகரமாக நிறுவப்பட்டுவிட்டது. இவற்றைப் பயன்படுத்தி எந்த ஒரு வேலையையும் செய்வதற்கு முன்னர் இவைகளின் செயல்பாடுகளைத் துவக்க வேண்டும். அதற்கான கட்டளை பின்வருமாறு அமையும்.

```
sudo service elasticsearch start
sudo service logstash start
sudo service kibana start
```

பின்னர் இவற்றின் தற்போதைய நிலைகளை அறிந்து கொள்ள பின்வரும் கட்டளையைப் பயன்படுத்தலாம். இது Active / Inactive எனும் மதிப்புகளைப் பெற்றிருக்கும்.

```
sudo service elasticsearch status
sudo service logstash status
sudo service Kibana status
```

அவ்வாறே இவைகளின் செயல்பாடுகளை நிறுத்துதல், மீண்டும் துவக்குதல் போன்ற செயல்களைச் செய்வதற்குப் பின்வரும் கட்டளைகள் பயன்படும்.

```
sudo service elasticsearch stop
sudo service logstash stop
sudo service kibana stop
```

```
sudo service elasticsearch restart
sudo service logstash restart
sudo service kibana restart
```

3 Elastic Search

ElasticSearch என்பது பல கோடிக்கணக்கான தரவுகளை சேமித்து வைத்துக்கொண்டு, நாம் கேட்கும் நேரங்களில் கேட்கும் தகவல்களை துரிதமாக வெளிப்படுத்த உதவும் ஒரு சேமிப்புக் கிடங்கு மற்றும் தேடு இயந்திரம் (Storage area & Search engine) ஆகும். தேடலிலும் நமக்கு உதவும் வகையில் இது வடிவமைக்கப்பட்டுள்ளது. GitHub, Google, StackOverflow, Wikipedia போன்றவை இதனைப் பயன்படுத்தி அதிக அளவு தகவல்களை

சேமிப்பத்தோடு மட்டுமல்லாமல், பயனர்கள் தங்கள் விருப்பம்போல் தகவல்களை தேடி எடுப்பதற்கு ஏற்ற வகையில் தேடும் திறனையும், வேகத்தினையும் துரிதமாக்குகின்றன. எடுத்துக் காட்டுக்கு, Google-ல் நாம் எதையாவது கொடுத்து தேடும்போது, நாம் கொடுக்கின்ற வார்த்தைகளில் முழுவதுமாகப் பொருந்தக் கூடிய இணைப்புகளை முதலாவதாகவும், சற்று ஓரளவு தொடர்புடைய வார்த்தைகளைப் பெற்றுள்ள இணைப்புகளை அடுத்தடுத்தும் வெளிப்படுத்தும். அவ்வாறே நாம் தேடுவதற்காக வார்த்தைகளை அடிக்கும்போதே 'suggessions' என்ற பெயரில் நம் தேடலுக்கு ஏற்ற வார்த்தைகளை அதுவே நமக்கு பரிந்துரைக்கும். இவையெல்லாம் Elastic Search-ன் அம்சங்களே.

வாக்கியங்கள்/வார்த்தைகளின் அடிப்படையில் துரிதமாகத் தேடல்களை நிகழ்த்துவதற்கு ஏற்ற வகையில் முதன்முதலில் வடிவமைக்கப்பட்டதே Apache's Lucene எனும் இயந்திரம் ஆகும். இது Java மொழியைக் கொண்டு எழுதப்பட்ட cross-platform திறன் கொண்ட ஒரு இயந்திரம். இதனை அடித்தளமாக வைத்துத்தான் இன்றைய 'ElasticSearch' உருவாக்கப்பட்டுள்ளது. Lucene-ல் காணப்பட்ட ஒருசில சிரமங்களைக் களைவதற்காக, இது வலிமைமிகு REST API-ஐப் பயன்படுத்துகிறது. இதன் மூலம் தரவுகள் அனைத்தும் index முறையில் கையாளப்படுகின்றன. எனவே தரவுகளை தேடுவதும், எடுப்பதும், சேமிப்பதும் சுலபமாகிறது. மேலும் இந்த Rest API-ஆனது Curl எனும் கட்டளையைப் பயன்படுத்தி

நேரடியாக elastic search-வுடன் பேசுகிறது. இவ்வாறு பேசும்போது இது செலுத்தும் தகவல்களும், பெற்றுக்கொள்ளும் தகவல்களும் JSON வடிவத்தில் அமையும்.

ElasticSearch-ஆனது Lucene-ன் செயல்பாடுகளை இன்னும் அதிக அளவில் விரிவாக்கியுள்ளது. அதாவது சமீபத்திய தகவல்களின் அடிப்படையில் உடனுக்குடன் ஆய்வறிக்கை எடுக்கும் அளவிற்கு (Real-time Analysis with real-time data) இது மிகவும் துரிதமானது. எடுத்துக்காட்டுக்கு டுவிட்டரில் எத்தனை பேர் ‘#கதிராமங்கலம்’ என்று பதிவிட்டுள்ளனர் என்பதனை அறிக்கையாக எடுத்துப்பார்த்தால், அதில் ஒரு சில நிமிடத்திற்கு முன்னர் பதிவிட்டவர் கூட இடம் பெற்றிருப்பார். அந்த அளவுக்கு இதன்

வேகம் மிகவும் துரிதமானது. ElasticSearch-ல் வடிவான வடிவற்ற தகவல்கள் அனைத்தும் ஒரு ஒழுங்குமுறைப்படுத்தப்பட்ட நிலையில், index-ன் அடிப்படையில் சேமிக்கப்படுகின்றன. எனவே பயனர்கள் கேட்கும் தகவல்களை இது நொடியில் எடுத்து வெளிப்படுத்தும். அதே போல இதனுள் செலுத்தப்படும் தகவல்களும் உடனுக்குடன் உள் சென்றுவிடும்.

3.1 ஒருசில அடிப்படைப் பதங்கள்

அடுத்தபடியாக ElasticSearch-ல் பயன்படுத்தும் ஒருசில அடிப்படையான சொற்களைப் பற்றி அறிந்து கொள்வோம்.

3.1.1 Node

இதுதான் நமது கணினியில் elasticsearch-ஐ இயக்கிக்கொண்டிருக்கின்ற இடம் ஆகும். Cluster முறையில் பல்வேறு கணினிகள் இணைக்கப்பட்டு அவை அனைத்தும் தங்களுக்கென்று உள்ள node-ல் elasticsearch-ஐ இயக்கிக்கொண்டிருக்கும். ஒரு node-ல் ஏதோ கோளாறு ஏற்பட்டு, அது தன் செயல்பாட்டை நிறுத்திவிட்டால்கூட, மற்றொரு கணினியில் இயங்கிக்கொண்டிருக்கும் node-ஆனது செயல்பட்டு நமது வேலைக்குத் தடைகள் ஏதும் நிகழாமல் பார்த்துக்கொள்ளும். இவ்வகையாக பல்வேறு கணினிகளில் இயங்கிக்கொண்டிருக்கும் nodes-ஐ, அது செய்யும் வேலையைப் பொறுத்து கீழ்க்கண்ட 3

விதங்களில் பிரிக்கலாம். ஒவ்வொரு கணினியில் உள்ள node-ன் configurations-ம் ஒவ்வொரு வகையில் அமையும்.

Master Node:- இது cluster முறையில் இணைக்கப்பட்டுள்ள பல்வேறு கணினிகளில் உள்ள அனைத்து nodes-ஐயும் மேலாண்மை செய்கின்ற வேலையை மட்டும் செய்யும். தரவுகளை சேமித்தல், வேண்டிய தரவுகளைத் தேடி எடுத்துக் கொடுத்தல் போன்ற எந்த ஒரு வேலையையும் செய்யாது. அதாவது தான் பெற்றுக்கொள்கின்ற ஒரு வேலையை மற்ற data nodes-க்கு அனுப்பி செய்யச் சொல்லும். அவ்வளவுதான். ஒரு node-ஐ இவ்வகையில் செயல்பட வைக்க விரும்பினால் elasticsearch.yml எனும் கோப்பில் சென்று node.master = true , node.data=false என்று கொடுக்க வேண்டும்.

Data Node:- இது தரவுகளை சேமித்தல், வேண்டிய தரவுகளைத் தேடி எடுத்துக் கொடுத்தல் போன்ற அடிப்படை வேலைகளைச் செய்கிறது. ஒரு node-ஐ இவ்வகையில் செயல்பட வைக்க விரும்பினால் அதே கோப்பில் சென்று node.master = true , node.data=false என்று கொடுக்க வேண்டும்.

Routing node or load balancer node:- Master Node-தான் தன்னிடம் உள்ள வேலைகளை data node-க்கு அனுப்பும் என்று ஏற்கனவே பார்த்தோம். ஆனால் இப்படிப் பிரித்துக் கொடுப்பதும் ஒரு

வேலைதானே. வேலைப்பளு அதிகமான நேரங்களில் இந்த வேலையைச் செய்வதற்குக் கூட Master-ஆனது Routing Node என்பதனைப் பயன்படுத்துகிறது. இந்த routing node-தான் வேலைப்பளுவை மொத்தமாகப் பெற்றுக்கொண்டு அதனைப் பிரித்து, data node-க்கு அனுப்பும் வேலையைச் செய்கிறது. ஒரு node-ஐ இவ்வகையில் செயல்பட வைக்க node.master = false , node.data=false என்று கொடுக்க வேண்டும்.

3.1.2 Shards

ஒவ்வொரு Node-லும் ஒன்று அல்லது அதற்கு மேற்பட்ட ‘shard’ வசிக்கின்றன. இத்தகைய shards-தான் தரவுகளைத் தாங்கியிருக்கும் பற்பல பிரிவுகள் ஆகும். ஒவ்வொரு shard-ம் பற்பல segments-களில் தரவுகளை சேமித்துக்கொண்டே வரும். நாம் கொடுக்கும் தகவல்கள் அப்படியே சென்று இதில் சேமிக்கப்படாது. வாக்கியங்கள் வார்த்தைகளாகப் பிரிக்கப்பட்டு, அந்த வார்த்தைகளில் ஒரே மாதிரியானவை நீக்கப்பட்டு தனித்தனி வார்த்தைகளாக சேமிக்கப்படும். இதே முறையில்தான் எண்கள், குறியீடுகள், வடிவங்கள் போன்றவற்றையும் கையாளும். அதாவது “I Love Chennai” என்பது சேமிக்கப்பட்ட பின்னர், மீண்டும் “I

Love London” என்றொரு வார்த்தை வருகிறதெனில், I Love என்பது தவிர்க்கப்பட்டு London என்பது மட்டுமே சேமிக்கப்படும். இவ்வகையான சேமிப்பு முறைக்கு ‘Inverted Index’ என்று பெயர். இதுதான் ஒரு வார்த்தை எந்தெந்த வார்த்தைகளுடன் தொடர்புடையது(term relevance), அது எத்தனை முறை இடம் பெற்றுள்ளது(term frequencies), எந்தெந்த இடங்களில் இடம்பெற்றுள்ளது(word proximity) என்பது போன்ற விவரங்களைப் பாதுகாத்து ஒரு கையேடு போன்று செயல்படும். இவ்வகையான ‘Inverted Index’ நிலையானது. சேமிக்கப்பட்டுள்ள தரவுகளில் எவ்வித மாற்றமும் செய்ய இயலாது. ஏனெனில் ஒரு இடத்தில் மாற்றினால், அது தொடர்புடைய

பல்வேறு இடங்களில் சேமிக்கப்பட்டுள்ள வார்த்தைகளும் சேர்ந்து பாதிப்படையும்.

ஒரு node-ல் பல்வேறு shards உள்ளன என்று நாம் பார்த்தோம். இந்த node-ல் ஏதோ கோளாறு ஏற்பட்டு, அது தன் செயல்பட்டை நிறுத்திவிட்டால், இதன் அடிப்படையில் நிகழ்த்து கொண்டிருக்கும் அனைத்து விதமான செயல்பாடுகளும் நின்று விடும் அபாயம் உள்ளது. இதனைத் தவிர்ப்பதற்காக உள்ளதே Replica shards ஆகும். அதாவது Cluster முறையில் இணைக்கப்பட்ட பல்வேறு கணினிகளில் ஒன்றில் முதன்மையான Primary Shards உருவாக்கப்படும். பின்னர் அதன் வடிவமைப்பிலேயே மற்றொரு இயந்திரத்தில் ஒரு பிரதி எடுத்துவைக்கப்படும். இதுவே Replica Shards எனப்படும். எனவே

முதன்மை இயந்திரத்தில் ஏதேனும் கோளாறு ஏற்பட்டால், மற்றொரு இயந்திரத்தில் உள்ள replica shards இயங்க ஆரம்பிக்கும். அவ்வாறே ஒவ்வொரு இயந்திரத்தினுடைய பிரதிகளும் அடுத்தடுத்த இயந்திரங்களில் பாதுகாக்கப்படும். ஏதேனும் ஒன்று தன் செயல்பாட்டை நிறுத்தினால் கூட மற்றொன்று இயங்கி நமக்கு கைகொடுக்கும். இதுவே ‘Failover Mechanism’ என்று அழைக்கப்படும்.

3.1.3 Index

எந்த shard-ல் எந்தெந்த விவரங்கள் பொதிக்கப்பட்டுள்ளன, அவை ஒன்றுடன் ஒன்று எவ்வாறு இணைக்கப்பட்டுள்ளன என்பது போன்ற விவரங்களை அளிப்பதற்கு

index ஒரு மாபெரும் தளம் போன்று செயல்படும். ஒன்றுக்கும் மேற்பட்ட index செயல்பட்டு இத்தகைய தரவுகளைப் பராமரிக்கின்றன. பொதுவாக ஒரேமாதிரியான குணநலன்களைக் கொண்ட தரவுகள் அனைத்தும் ஒரு index-ன் கீழ் அமையும். மேலும் இவை பல்வேறு வகைகளில்(types) பிரிக்கப்பட்டு, ஒவ்வொரு வகையின் கீழும் விவரங்களை அதனதன் பண்புகளின்(characteristics) அடிப்படையில் சேமிக்கும்.

அதாவது ElasticSearch-ல் பல்வேறு வகையான indices (plural of index) காணப்படும். ஒவ்வொரு index-ம் பல்வேறு types-ஐக் கொண்டிருக்கும். ஒவ்வொரு type-ம் documents-ஐ பல்வேறு வகையான

Properties-ன் அடிப்படையில் சேமிக்கும். மற்றொரு வகையில் சொல்ல வேண்டுமானால் Elastic Search-ஐ Mysql-வுடன் ஒப்பிட்டு மேற்கண்ட அனைத்தையும் பின்வருமாறு புரிந்து கொள்ளலாம்.

Elastic Search = Mysql
Index = Database
Types = Tables
Properties = Columns
Documents = Rows

3.2 Curl கட்டளையின் மாதிரி வடிவம்

நாம் ஏற்கனவே கண்டது போல ElasticSearch-வுடன் பேசி அதனுள்

விவரங்களைப் போடுவதற்கும், எடுப்பதற்கும், தேடுவதற்கும் RESTful API என்பது பயன்படுகிறது. இது command line-ல் curl எனும் கட்டளையைப் பயன்படுத்தி index-வுடன் நேரடியாகப் பேசுகிறது. இதன் மாதிரி வடிவம் பின்வருமாறு.

```
curl -X<verb> '<protocol>://<host>:<port>/<path>/<operation_name>?<query_string>' -d '<body>'
```

verb : என்ன செய்ய வேண்டும் என்பதனைக் குறிக்கும் ஒரு வினைச்சொல்லை இது பெற்றிருக்கும். GET, POST, PUT, DELETE, HEAD ஆகிய மதிப்புகளில் ஏதேனும் ஒரு மதிப்பு இங்கு காணப்படும்.

protocol : இது http அல்லது https ஆகிய மதிப்புகளில் ஒன்றாக இருக்கும்.

host : இது 'localhost' என்றோ அல்லது cluster-ல் இயங்கிக்கொண்டிருக்கின்ற node-ன் IP முகவரியாகவோ இருக்கும்.

port : தற்போது elasticsearch இயங்கிக்கொண்டிருக்கின்ற port-ன் உடைய எண் ஆகும். பொதுவாக 9200 என்று இருக்கும்.

path : இது index,type போன்ற மதிப்பினைக் கொண்டிருக்கும்.

operation_name : என்ன வகையான செயல் செய்யப்போகிறோம் என்பது இங்கு காணப்படும். _search , _count போன்ற மதிப்புகளைக் கொண்டிருக்கும்.

query_string : இது கட்டாய மதிப்பு கிடையாது. விருப்பம் இருந்தால் ?pretty எனக் கொடுக்கலாம். இது தெளிவான வடிவில் தகவல்களை வெளிப்படுத்த உதவும்.

body : data என்பதைக் குறிக்கும் -d ஐத் தொடர்ந்து இப்பகுதி காணப்படும். நமது விருப்பத்திற்கு ஏற்றார் போன்ற தகவல்களைப் பெறுவதற்கு இப்பகுதியில் தான் கூடுதல் விவரங்கள் அளிக்கப்படும்.

3.2.1 எடுத்துக்காட்டு 1

பின்வரும் கட்டளை localhost-னுடைய port எண் 9200-ல் எது இயங்கிக்கொண்டிருக்கிறதோ அதனுடைய விவரங்களைப் பெற்று json வடிவில் வெளிப்படுத்தும். பொதுவாக 9200-ல் ElasticSearch இயங்கிக்கொண்டிருப்பதால் அதனுடைய மதிப்பு இங்கு வெளிப்பட்டுள்ளது.

```
curl -XGET
'http://localhost:9200/?pretty'
```

வெளியீடு:

```
{
  "name" : "GfZBwlm",
  "cluster_name" :
"elasticsearch",
  "cluster_uuid" :
"Ux_Fc9t6Tzin5hh06NhbYA",
  "version" : {
    "number" : "5.5.1",
    "build_hash" : "19c13d0",
    "build_date" : "2017-07-
18T20:44:24.823Z",
    "build_snapshot" : false,
    "lucene_version" : "6.6.0"
  },
  "tagline" : "You Know, for
Search"
}
```

3.2.2 எடுத்துக்காட்டு 2

இது கொடுக்கப்பட்டுள்ள port-ல் இயங்கிக்கொண்டிருக்கும் elasticsearch அமைந்துள்ள cluster பற்றிய விவரங்களை அளிக்கிறது. Cluster-ன் பெயர், அதன் நிலை, அதிலுள்ள nodes, shards என்று அனைத்து விவரங்களையும் வெளிப்படுத்தும்.

Cluster முறையில் இணைக்கப்பட்ட கணினிகளில் உள்ள nodes, shards ஆகியவை தயார் நிலையில் உள்ளதா இல்லையா என்பதைத் தெரிந்து கொள்வதற்கு health உதவும். இது தன்னுடைய நிலையினை பின்வரும் 3 மதிப்புகளில் வெளிப்படுத்தும்.

- Red என வெளிப்படுத்தினால் "ஒருசில முதன்மை shards தயாராக இல்லை" என்று அர்த்தம்.
- Yellow என வெளிப்படுத்தினால் "முதன்மை shards அனைத்தும் தயாராக உள்ளது, ஒருசில replica shards-தான் தயாராக இல்லை" என்று அர்த்தம். பொதுவாக ஒரே ஒரு கணினியில் இயங்கிக்கொண்டிருக்கும் elasticsearch இந்நிலையினை வெளிப்படுத்தும். ஏனெனில் அதற்கு replica என்ற ஒன்று கிடையாது.
- Green என வெளிப்படுத்தினால் "முதன்மை Replica ஆகிய அனைத்தும் தயாராக உள்ளது" என்று அர்த்தம்.

```
curl -XGET
'http://localhost:9200/_cluster/h
ealth?pretty=true'
```

வெளியீடு:

```
{
  "cluster_name" :
"elasticsearch",
  "status" : "yellow",
  "timed_out" : false,
  "number_of_nodes" : 1,
  "number_of_data_nodes" : 1,
  "active_primary_shards" : 6,
  "active_shards" : 6,
```

```
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 6,
  "delayed_unassigned_shards" :
0,
  "number_of_pending_tasks" : 0,
  "number_of_in_flight_fetch" :
0,

"task_max_waiting_in_queue_millis
" : 0,

"active_shards_percent_as_number"
: 50.0
}
```

Cluster-ன் நிலையினைத் தெரிந்து கொண்டதுபோல shards, index ஆகிய அனைத்தின் நிலையினையும் health மூலம் தெரிந்து கொள்ளலாம். அதற்கான கட்டளைகள் பின்வருமாறு.

```
curl -XGET
'http://localhost:9200/_cluster/health?level=cluster
curl -XGET
'http://localhost:9200/_cluster/health?level=shards
curl -XGET
'http://localhost:9200/_cluster/health?level=indices
```

3.2.3 எடுத்துக்காட்டு 3

இது கொடுக்கப்பட்டுள்ள port-ல் இயங்கிக்கொண்டிருக்கும் அனைத்து index-ஐயும் பட்டியலிடும். ?v என்பது தலைப்புடன் சேர்த்து வெளியிடும். ?v தரவில்லையெனில் தலைப்புகள் வெளிவராது.

```
curl
'localhost:9200/_cat/indices?v'
```

வெளியீடு:

```
health status index    uuid
pri rep docs.count docs.deleted
store.size pri.store.size
yellow open  interviews  EU-
BIHTrSgmoScvHsNGXzA  5  1     4
0   25.7kb     25.7kb
yellow open  bookdb_index
TeMyXrZeT8iMsQjoVxBqwQ  1  1
4      0   27.5kb     27.5kb
```

3.2.4 எடுத்துக்காட்டு 4

இப்போது ஒரு புது index-ஐ உருவாக்கி அதனுள் எவ்வாறு தகவல்களை உட்செலுத்துவது என்று பார்க்கலாம்.பின்வரும் கட்டளை

elasticsearch இயங்கிக்கொண்டிருக்கும் இடத்தில் 'interviews' எனும் பெயர் கொண்ட index-ஐ உருவாக்கி அதில் 'candidates'எனும் வகையின்கீழ் கொடுக்கப்பட்டுள்ள விவரங்களை உட்செலுத்துகிறது.

-d ஐத் தொடர்ந்து single quotes-க்குள் {பண்பு:தரவு} (property:data) எனும் முறையில் விவரங்களை அமைக்க வேண்டும் .

```
curl -XPOST
'http://localhost:9200/interviews
/candidates' -d '
{
```

```
"Name" : "Nandhini",
"DOB" : "1988-10-04",
"Experience": 7,
"Organizations_worked" :
["Polaris", "Virtusa","Infosys"],
"Skillset": "Ruby, Python, Pearl,
ETL Testing",
"Feedback":"Strong in Testing,
Expert in SQL, Beginner in
Python"
}'
```

இதன் வெளியீடு பின்வருமாறு அமைந்தால் தரவுகள் வெற்றிகரமாக உட்சென்றுவிட்டன என்று அர்த்தம்.

```
{"_index":"interviews","_type":"c
andidates","_id":"AV2n8sn4DfKljeJ
YHE0Y","_version":1,"result":
"created","_shards":
{"total":2,"successful":1,"failed
":0},"created":true}
```

மேற்கூறிய அதே முறையில் பின்வரும் தரவுகளையும் ஒவ்வொன்றாக உட்செலுத்தி விடவும்.

```
"Name" : "Ruby Jackson",
"DOB" : "1983-05-23",
"Experience": 10,
"Organizations_worked" : ["Virtusa",
"TVS Lucas","Tech Mahindra"],
```

```
"Skillset": "DW Testing, Expert
systems, Java","Feedback":"Beginner
in Selenium, Knows Java and ETL"

"Name" : "Narayani",
"DOB" : "1995-11-14",
"Experience": 2,
"Organizations_worked" : ["Aspire
Systems"],
"Skillset": "Big Data, ELK Stack",
"Feedback":"Expertise in Big Data,
Beginner in ELK"

"Name" : "N.T.Madhuri",
"DOB" : "1990-05-30",
"Experience": 4,
"Organizations_worked" :
```

```
["Thoughtworks", "Virtusa"],
"Skillset": "Unix, Linux",
"Feedback":"Strong knowledge in
Unix & Linux"
```

இப்போது நந்தினி, ரூபி ஜாக்சன், நாராயாணி, N.T.மாதுரி ஆகியோரின் பெயர், பிறந்த தேதி, எத்தனை வருட அனுபவமுடையவர்கள், அவர்கள் வேலை செய்த நிறுவனங்களின் விவரம், எந்தெந்த தொழில்நுட்பங்களில் திறன் மிக்கவர்கள் மற்றும் அவர்களை நேர்காணல் செய்தவர்கள் கொடுத்த பின்னொட்டம் ஆகியவை elasticsearch-க்குள் செலுத்தப்பட்டுவிட்டன.

இப்போது இவற்றை வைத்து எந்தெந்த வகையில் நமக்கு வேண்டிய தரவுகளை மட்டும்

தேடுவது, எடுப்பது, அதற்குக் கட்டளைகளை எவ்வாறெல்லாம் அமைப்பது போன்றவை பற்றி இனி பார்க்கப்போகிறோம்.

3.2.5 எடுத்துக்காட்டு 5

நாம் உருவாக்கிய index-ல் உள்ள அனைத்து தரவுகளையும் தேடி எடுப்பதற்கான கட்டளை பின்வருமாறு அமையும். மேற்கண்டது போன்றே கட்டளையை அமைத்து கடைசியாக _search எனக் கொடுத்தால், அது கொடுக்கப்பட்ட index/type-ல் உள்ள அனைத்துத் தரவுகளையும் எடுத்து வெளிப்படுத்தும்.

```
curl -XGET
'http://localhost:9200/interviews
/candidates/_search?pretty'
```

வெளியீடு:

```
{
  "took" : 5,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
```

```
    "failed" : 0
  },

  "hits" : {
 "total" : 4,
 "max_score" : 1.0,
 "hits" : [
 {
 "_index" : "interviews",
 "_type" : "candidates",
 "_id" : "AV2nkI0pDfKljeJYHE0U",
 "_score" : 1.0,
 "_source" : {
 "Name" : "Narayani",
 "DOB" : "1995-11-14",
 "Experience" : 2,
 "Organizations_worked" : [
```

```
 "Aspire Systems"
 ],
 "Skillset" : "Big Data, ELK Stack",
 "Feedback" : "Expertise in Big Data, Beginner in ELK"
 }
 },
 {
 "_index" : "interviews",
 "_type" : "candidates",
 "_id" : "AV2neRCiDfKljeJYHE0O",
 "_score" : 1.0,
 "_source" : {
 "Name" : "N.T.Madhuri",
 "DOB" : "1990-05-30",
 "Experience" : 4,
```

```
"Organizations_worked" : [
            "Thoughtworks",
            "Virtusa"
          ],
          "Skillset" : "Unix,
Linux",
          "Feedback" : "Strong
knowledge in Unix & Linux"
        }
      },
      {
        "_index" : "interviews",
        "_type" : "candidates",
        "_id" :
"AV2nkG8DDfKljeJYHE0T",
        "_score" : 1.0,
        "_source" : {
          "Name" : "Ruby
```

```
Jackson",
          "DOB" : "1983-05-23",
          "Experience" : 10,

"Organizations_worked" : [
            "Virtusa",
            "TVS Lucas",
            "Tech Mahindra"
          ],
          "Skillset" : "DW
Testing, Expert systems, Java",
          "Feedback" : "Beginner
in Selenium, Knows Java and ETL"
        }
      },
      {
        "_index" : "interviews",
        "_type" : "candidates",
```

```
        "_id" :
"AV2n8sn4DfKljeJYHE0Y",
        "_score" : 1.0,
        "_source" : {
          "Name" : "Nandhini",
          "DOB" : "1988-10-04",
          "Experience" : 7,

"Organizations_worked" : [
            "Polaris",
            "Virtusa",
            "Infosys"
          ],


          "Skillset" : "Ruby,
Python, Pearl, ETL Testing",
          "Feedback" : "Strong in
```

```
Testing, Expert in SQL, Beginner
in Python"
        }
      }
    ]
  }
}
```

இங்கு தரவுகளுக்கான ஒவ்வொரு தொகுப்பும் தனித்தனி id-யுடன் விளங்குவதைக் காணவும். இந்த id-தான் பின்னாளில் இத்தரவுகளை அழிக்கவோ மாற்றம் செய்யவோ பயன்படும்.

3.2.6 எடுத்துக்காட்டு 6

ஒரு குறிப்பிட்ட id-ஐக் கொண்ட தரவுகளை ஆழிப்பதற்கான கட்டளை பின்வருமாறு அமையும். இதில் 'AV2nkFPQDfKljeJYHE0S' என்பதுதான் id ஆகும்.

```
curl -XDELETE
'http://localhost:9200/interviews
/candidates/AV2nkFPQDfKljeJYHE0S'
```

இதன் வெளியீடு பின்வருமாறு அமைந்தால் தரவுகள் வெற்றிகரமாக நீக்கப்பட்டுவிட்டன என்று அர்த்தம்.

```
{"found":true,"_index":"interview
s","_type":"candidates","_id":"AV
2nkFPQDfKljeJYHE0S","_version":2,
"result":"deleted","_shards":
{"total":2,"successful":1,"failed
":0}}
```

3.2.7 எடுத்துக்காட்டு 7

இப்போது ஒருவர் நாம் மேலே உட்செலுத்திய விவரங்களின் அடிப்படையில் தனக்கு வேண்டிய தேடலைத் தொடங்குகிறார் என்று வைத்துக்கொள்வோம். அவர் Ruby *எனும் மொழி தெரிந்தவர்களின் தொகுப்புகளைப் பெறுவதற்காக* _search *ஐத் தொடர்ந்து* q=Ruby *எனப் பின்வருமாறு கொடுத்துள்ளார்.*

```
curl -XGET
'http://localhost:9200/interviews
/candidates/_search?
q=Ruby&pretty'
```

ஆனால் இதுவோ Ruby மொழி தெரிந்த நந்தினியின் தரவுகளை வெளியிடுவதோடு நிறுத்தாமல், ரூபி எனும் பெயர் கொண்டவரின் தகவலையும் சேர்த்து வெளியிட்டுள்ளது.

வெளியீடு:

```
{
```

```
  "took" : 37,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },

  "hits" : {
    "total" : 2,
    "max_score" : 0.26228312,
  "hits" : [
      {
        "_index" : "interviews",
        "_type" : "candidates",
        "_id" :
"AV2nkG8DDfKljeJYHE0T",
```

```
        "_score" : 0.26228312,
        "_source" : {
          "Name" : "Ruby Jackson",
          "DOB" : "1983-05-23",
          "Experience" : 10,
          "Organizations_worked" :
[
            "Virtusa",
            "TVS Lucas",
            "Tech Mahindra"
          ],
          "Skillset" : "DW
Testing, Expert systems, Java",
          "Feedback" : "Beginner
in Selenium, Knows Java and ETL"
        }
      },
      {
```

```
        "_index" : "interviews",
        "_type" : "candidates",
        "_id" :
"AV2n8sn4DfKljeJYHE0Y",
        "_score" : 0.2568969,
        "_source" : {
          "Name" : "Nandhini",
          "DOB" : "1988-10-04",
          "Experience" : 7,
          "Organizations_worked" :
[
            "Polaris",
            "Virtusa",
            "Infosys"
          ],
          "Skillset" : "Ruby,
Python, Pearl, ETL Testing",
          "Feedback" : "Strong in
```

```
Testing, Expert in SQL, Beginner
in Python"
        }
      }
    ]
  }
}
```

இதுபோன்ற குழப்பங்களைத் தவிர்க்க நாம் கொடுக்கின்ற வார்த்தையை எந்தப் பண்பின்கீழ் தேட வேண்டும் எனவும் குறிப்பிட்டுக் கூற வேண்டும்.

3.2.8 எடுத்துக்காட்டு 8

Ruby எனும் வார்த்தையை skillset எனும் பண்பின்கீழ் தேடி அதில் பொருந்துபவர்களை மட்டும் வெளியிடுக என்று கூறுவதற்கான கட்டளை பின்வருமாறு அமையும்.

```
curl -XGET
'http://localhost:9200/interviews
/candidates/_search?
q=Skillset:Ruby&pretty'
```

3.2.9 எடுத்துக்காட்டு 9

மேற்கண்ட அதே கட்டளையை DSL (Domain Specific Language) மொழி கொண்டு பின்வருமாறு எழுதலாம்.

```
curl -XPOST
'http://localhost:9200/interviews
/candidates/_search?pretty' -d '


{
 "query" : {
 "match" : {
 "Skillset" : "Ruby"
```

```
            }
        }
    }'
```

அதாவது ஒன்று அல்லது அதற்கு மேற்பட்ட விதிமுறைகளை இணைத்து நமது தேடுதலைக் குறுக்குவதற்கு இம்மொழி மிகவும் உதவுகிறது. இங்கு நாம் கண்டது ஒரே ஒரு பண்பில் மட்டும் வார்த்தையைத் தேடுகின்ற அடிப்படை 'match query' ஆகும். இனிவரும் கட்டளைகளில் இதன் பல்வேறு அம்சங்களைக் காணலாம்.

3.2.10 எடுத்துக்காட்டு 10

ஒன்றுக்கும் மேற்பட்ட பண்புகளில் வார்த்தையைத் தேடுகின்ற 'multi_match query' பின்வருமாறு அமையும். இங்கு Expert எனும் வார்த்தை "Skillset"-ல் காணப்பட்டாலோ அல்லது "Feedback"-ல் காணப்பட்டாலோ அத்தரவுத் தொகுப்பு வெளிப்படும்.

```
curl -XPOST
'http://localhost:9200/interviews
/candidates/_search?pretty' -d '
```

```
{
     "query" : {
         "multi_match" : {
             "query" : "Expert",
             "fields":
["Skillset","Feedback"]
         }
     }
 }'
```

3.2.11 எடுத்துக்காட்டு 11

இங்கு multi_match query-ல் உள்ள fuzziness எனும் பண்பினைப் பற்றிக் காணலாம். இது நமது தேடு வார்த்தையில் உள்ள எழுத்துப் பிழைகளை நீக்க உதவுகிறது.

தவறுதலாக Expertis என்று கொடுத்தாலும் சரியான Expertise எனும் வார்த்தையை எடுத்துக்கொண்டு தேடுகிறது.

```
curl -XPOST
'http://localhost:9200/interviews
/candidates/_search?pretty' -d '


{
     "query" : {
         "multi_match" : {
             "query" :
"Expertis",
             "fields":
```

```
  ["Skillset","Feedback"],
                "fuzziness": "AUTO"
            }
       }
   }'
```

3.2.12 எடுத்துக்காட்டு 12

must, should, must_not மூலம் விதிமுறைகளை இணைத்து நமது தேடுதலை இன்னும் சிறப்பாகக் குறுக்குவதற்கு 'bool query' பயன்படுகிறது. இவை முறையே

AND, OR, NOT போன்றவற்றிற்கு மாற்றாகப் பயன்படுத்தப்படுகின்றன.

பின்வரும் bool query-ல் உட்புறமும் வெளிப்புறமுமாக இரண்டு bools பயன்படுத்தப்பட்டுள்ளன. உட்புற bool-ல் ஒருவருடைய skillset-ஆனது 'ETL Testing' அல்லது 'DW Testing'-ஆக இருக்க வேண்டும் என்பதற்காக அவை should-க்குள் கொடுக்கப்பட்டுள்ளன. இந்தப் பொருத்தத்தின் மூலம் தேர்ந்தெடுக்கப்படுபவர் virtusa-நிறுவனத்தில் பணிபுரிந்தவராகவும் polaris-ல் பணிபுரியாதவராகவும் இருக்க வேண்டும் என்பதற்காக அவைகள் முறையே must மற்றும் must_not பயன்படுத்தி அமைக்கப்பட்டுள்ளன. இவை அனைத்திலும் பொருந்துகின்ற ஒருவரைத் தேர்வு

செய்வதற்காக வெளிப்புற bool உருவாக்கப்பட்டு அனைத்து விதிமுறைகளையும் இணைக்கிறது.

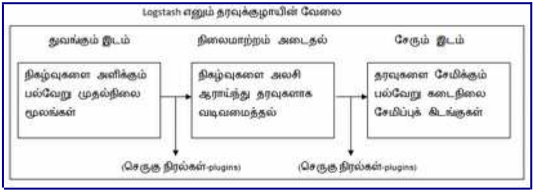
```
curl -XPOST
'http://localhost:9200/interviews
/candidates/_search?pretty' -d '


{
 "query": {
 "bool": {


            "must": {
                "bool" :
{ "should": [
                      { "match":
```

```
{ "Skillset": "ETL Testing" }},
                      { "match":
{ "Skillset": "DW Testing" }} ] }
            },
            "must": { "match":
{ "Organizations_worked":
"Virtusa" }},
            "must_not":
{ "match":
{"Organizations_worked":
"Polaris" }}
        }
    }
}'
```

4 Logstash

Logstash என்பது நிகழ்வுகளைப் பெற உதவும் ஒரு தரவுக் குழாய் (data pipeline) ஆகும். இது ரூபி மொழியில் எழுதப்பட்ட பல்வேறு வகையான செருகு நிரல்களை (plugins) வைத்து இயங்குகிறது. எனவே தான் இது "Plugin based events processing data pipeline" என்று அழைக்கப்படுகிறது. இந்த தரவுக் குழாய் 3 வகையான நிலைகளில் தரவுகளைக் கையாள்கிறது. இது பின்வருமாறு:Logstash என்பது நிகழ்வுகளைப் பெற உதவும் ஒரு தரவுக்

குழாய் (data pipeline) ஆகும். இது ரூபி மொழியில் எழுதப்பட்ட பல்வேறு வகையான செருகு நிரல்களை(plugins) வைத்து இயங்குகிறது. எனவே தான் இது “Plugin based events processing data pipeline” என்று அழைக்கப்படுகிறது. இந்த தரவுக் குழாய் 3 வகையான நிலைகளில் தரவுகளைக் கையாள்கிறது. இது பின்வருமாறு:



முதலில் பல்வேறுபட்ட மூலங்களிலிருந்து (different sources) வரும்

நிகழ்ச்சிகளை(events) Logstash என்பது பெற்றுக்கொள்ளும். இதன் பெயர் வேண்டுமானால் LogStash என்று இருக்கலாம். அதற்காக இது logs-ஐ மட்டும் பெற்றுக்கொள்ளும் கருவி என்று கிடையாது. Files, sockets, script outputs போன்ற பல்வேறுபட்ட வடிவங்களில் வரும் அனைத்து நிலைத் தரவுகளையும் பெற்றுக்கொள்ளும். இவ்வாறு வேறுபட்ட மூலங்களிலிருந்து வரும் நிகழ்வுகளை எடுப்பதற்கு வெவ்வேறு வகையான செருகு நிரல்கள் logstash - வுடன் இணைந்து செயல்படுகின்றன. இவை உள்ளீட்டுக்கான செருகு நிரல்கள் (Input plugins) என்று அழைக்கப்படுகின்றன. எடுத்துக்காட்டுக்கு file, irc, jdbc, kafka, github exec, eventlog, http, imap போன்றவை files, servers, webhooks, message queues, shell command outputs போன்ற பல்வேறுபட்ட மூலங்களிலிருந்து நிகழ்ச்சிகளை எடுக்க உதவுகின்றன. நிகழ்ச்சி என்று நான் குறிப்பிடுவது அதிலுள்ள தகவல்களை மட்டுமே! இதுவே முதல் நிலை ஆகும். அடுத்ததாக இவ்வாறு பெற்றுக்கொண்ட தகவல்களை நமக்குத் தேவைப்படுகின்ற விதங்களில் மாற்றி (data Processing) வடிவமைக்கின்றன. இந்நிலையில்தான் எதன் அடிப்படையில் தகவல்கள் பிரிக்கப்பட வேண்டும், எவை மாற்றப்பட வேண்டும், எவை நீக்கப்பட வேண்டும் போன்ற தகவல்களெல்லாம் கொடுக்கப்படுகின்றன. இதுவே இரண்டாம் நிலை.கடைசியாக வடிவமைக்கப்பட்ட தரவுகளை பல்வேறுபட்ட சேமிப்புக் கிடங்குகளில் கொண்டு சேர்க்கின்றன(destination). அவ்வாறு

கொண்டு சேர்ப்பதற்கு பல்வேறு செருகு நிரல்கள் உள்ளன. இவை வெளியீட்டுக்கான செருகு நிரல்கள் (Output plugins) என்று அழைக்கப்படுகின்றன. எடுத்துக்காட்டுக்கு csv, datalog, email, irc, jira, exec, kafka, elasticsearch போன்றவை files, servers, message queues, storage engine, databases போன்ற பல்வேறுபட்ட கடைநிலை சேருமிடங்களில் கொண்டு சேர்க்கின்றன. இதுவே மூன்றாம் நிலை.இப்போது இவை அனைத்தையும் எவ்வாறு செயல்வடிவில் செய்வது என்று காணலாம்.

4.1 ஒரு கோப்பிலிருந்து தரவுகளை உள்ளெடுத்தல்

earthquakes எனும் csv கோப்பில் இருக்கும் அனைத்துத் தகவல்களையும் எவ்வாறு logstash-க்குள் செலுத்துவது, பின்னர் அதனை எவ்வாறு வடிவமைப்பது, கடைசியாக elasticsearch எனும் search engine-க்குள் எவ்வாறு கொண்டு சேர்ப்பது என்று பார்க்கப்போகிறோம். இவை அனைத்தையும் செய்வதற்கு நாம் ஏதேனும் ஒரு பெயரைத் தொடர்ந்து .conf எனும் கோப்பினை உருவாக்கி அதில் பின்வருமாறு நிரல்களை எழுத வேண்டும்.

4.1.1 Config File

இங்கு earthquakes.conf எனும் பெயரில் நான் கோப்பினை உருவாக்கியுள்ளேன்.

```
input {
 file {
 path =>
["/home/ubuntu/earthquakes.txt"]
 type => "eqs"
 start_position => "beginning"
 }
}

filter {
```

```
csv {
 separator => ","
 columns =>
["DateTime","Latitude","Longitude
","Depth","Magnitude","MagType","
NbStations","Gap","Distance","RMS
","Source","EventID"]
 }
 mutate {convert => ["Latitude",
"float"]}
 mutate {convert => ["Longitude",
"float"]}
 mutate {convert => ["Depth",
"float"]}
 mutate {convert => ["Magnitude",
"float"]}
 mutate {convert => ["MagType",
```

```
"float"]}
}

output {

    elasticsearch {
        action => "index"
        hosts =>
["localhost:9200"]
        index => "earthquakes"
        workers => 1
    }

    stdout {}

}
```

இங்கு input{}, filter{}, outout{} எனும் functions, logstash-ல் உள்ள 3 நிலைகளில் நடத்தப்பட வேண்டிய வேலையைச் செய்கின்றன.

Input {}

இங்கு logstash-ஆனது csv கோப்பிலிருந்து தகவல்களைப் பெறவிருப்பதால் file எனும் உள்ளீட்டுக்கான செருகு நிரலைப் பயன்படுத்தியுள்ளது. அதன் path, type, start_position எனும் 3 parameters இதன் வேலையைச் சுலபமாக்குகின்றன.

- path என்பது கோப்பு இருக்கின்ற இடத்தின் பாதையை குறிக்கிறது.

- type என்பது நாம் வழங்கியுள்ள eqs என்பதனை தன் மதிப்பாகப் பெற்றுள்ளது. இதுவே நாம் பின்னர் தரவுகளை தேடுவதற்கும் எடுப்பதற்கும் உதவும் ஒரு அடிப்படை மதிப்பாக அமையும்.
- start_position என்பது beginning / end எனும் இரண்டு மதிப்புகளில் ஒன்றினைப் பெற்று விளங்கும். சமீபத்திய தகவல்களிலிருந்து துவங்க விரும்பினால் end எனவும், பழமையான தகவல்களிலிருந்து துவங்க விரும்பினால் begining எனவும் கொடுக்கலாம்.
- இவற்றின் அடிப்படையில் logstash-ஆனது தரவுகளை உள் எடுக்க ஆரம்பிக்கும்.

Filter {}

இது நமக்கு வேண்டிய வடிவங்களில் உள்ளெடுக்கப்பட்ட தரவுகளைப்பொருள்படும்படிப் பிரித்து நேர்த்தியாக வடிவமைக்கப்படுகிறது. எந்தவகையான மூலத்திலிருந்து தரவுகளைப் பெறுகின்றோமோ அதற்கேற்றார் போன்ற filter plugin பயன்படுத்தப்படும். இங்கு நாம் csv கோப்பிலிருந்துதகவல்களைப் பெற்றுள்ளதால் csv { } filter-ஐப் பயன்படுத்தியுள்ளோம். இதேபோன்று பிற வடிவிலான தரவுகளுக்காக json { }, xml { } போன்ற வகையான filters-ம் உள்ளன.

- csv-ன் separator எனும் பண்பானது, கோப்பில் உள்ள comma-வின்

அடிப்படையில் தரவுகள் பிரிக்கப்பட வேண்டும் என்பதைக்குறிக்கின்றது.

- columns எனும் பண்பின் மதிப்புகளாக இடம்பெற்றுள்ள fields மட்டுமே வெளியீட்டினுள் செலுத்தப்படும். அதாவது இதில்பட்டியலிடப்பட்டுள்ள fields மட்டும்தான் ES-க்குள் செல்லும்.
- convert மூலம் இவை அனைத்தும் float எனும் தரவகைக்குமாற்றப்படுகின்றன. ஏனெனில் Kibana வரைபடம் வரைவதற்கு இந்தத் தரவு வகைதான் பொருத்தமானதாக அமையும். இதுபோன்ற மாற்றங்கள் அனைத்தும் mutate { } எனும் filter-க்குள் கொடுக்கப்படவேண்டும்.

- mutate { }- ஆனது convert, copy, rename, replace, update, lowercase போன்ற வகையான மாற்றங்களை நிகழ்த்த உதவும் filter ஆகும்.

Output {}

கடைசியாக தகவல்களை ES-க்குள் செலுத்த வேண்டும் என்பதற்காக அதற்கான elasticsearch { } எனும் plugin-ஐப் பயன்படுத்தியுள்ளோம். அதன் action, hosts, index, workers எனும் 4 parameters இதற்கு உதவியுள்ளது.

- action என்பது எப்போதும் 'index' எனும் மற்றொரு parameter-ஐ அதன் மதிப்பாகப் பெற்றிருக்கும். இதற்கு delete எனும் மற்றொரு மதிப்பிணையும் வழங்கலாம். index என்பது ஒரு database-ஐ உருவாக்குவதற்கும், delete என்பது ஏற்கனவே இருக்கும் ஒன்றை ஆழிப்பதற்கும் பயன்படும்.
- hosts என்பது எங்கு உருவாக்க வேண்டும் என்பதனைக் குறிக்கிறது. இது elasticsearch இயங்கிக்கொண்டிருக்கும் hostname அல்லது IP முகவரியைக் கொண்டிருக்கும்.
- index என்பது earthquakes எனும் பெயர் கொண்ட ஒரு database-ஐ

உருவாக்கியுள்ளது. ஒவ்வொரு நாளும் இப்பெயரின் தொடர்ச்சியில் அந்நாளின் தேதியிட்டு தரவுகளை செலுத்திக்கொண்டே செல்லும். "earthquakes" எனும் பெயர் கொண்ட index என்பது ஒரு database போன்று செயல்படும். Input plugin-ல் உள்ள eqs எனும் பெயர் கொண்ட type என்பது அதிலுள்ள tables போன்று செயல்படும். ஒரு index-ல் எத்தனை type வேண்டுமானாலும் செலுத்தலாம். தற்போதைக்கு earthquakes-ல் உள்ள ஒரே type, 'eqs' ஆகும்.

- கடைசியாக stdout{ } எனும் plugin-ஐப் பயன்படுத்தி console-

லிலும் அவற்றை வெளியிடுமாறு கொடுத்துள்ளோம்.

4.1.2 கட்டளை

```
sudo
/usr/share/logstash/bin/logstash
-f earthquakes.conf
```

எனும் கட்டளை earthquakes கோப்பில் உள்ள அனைத்து நிகழ்வுகளையும் வேண்டிய மாற்றம் செய்து ElasticSearch Engine-க்குள் செலுத்தி விடும்.

4.2 Twitter போன்ற நிகழ்கால தரவுகளை உள்ளெடுத்தல்

இதன் Configuration file பின்வருமாறு. இதை இப்போது தங்களாலேயே புரிந்து கொள்ள முடியும். "#Ganesh Chathurthi" என்று tweet செய்தவர்களை எடுத்து logstash வழியே Elastic Search-க்குள் செலுத்தியுள்ளேன். twitter { } -க்கான input plugin இங்கு பயன்படுத்தப்பட்டு வேண்டிய தகவல்கள் உள்ளெடுக்கப்பட்டுள்ளன. அவை எவ்வித மாற்றமும் செய்யப்படாமல், 'twitter_elastic_example' எனும் index பெயரில் elastic search-க்குள் செலுத்தப்பட்டுள்ளன.

```
Input {
 twitter {
 consumer_key =>
"Mn4O9nBwKIwVfdsgNf8gqs546"
 consumer_secret =>
"Bgm7io78g0Ks7n1WAbF4oPAKXaLWAw3A
hj4ft47k6ooTNsRIIJ"
 oauth_token => "44962404-
v7EXtrfc8ZTqWosyPhoPDM5w5qBAefSQf
HOklLQeL"
 oauth_token_secret =>
"zosREB0kdInNbE03RMurjWkdyejsTmqt
POXlF2YHxzVqV"
 keywords => ["Ganesh
Chathurthi"]
          full_tweet => true
     }
    }
```

```
filter { }

output {
stdout {
    codec => rubydebug
}

  elasticsearch {
      hosts => "localhost:9200"
      index         =>
"twitter_elastic_example"
      document_type => "tweets"
      template      =>
"/etc/logstash/conf.d/twitter-
template.json"
      template_name =>
"twitter_elastic_example"
```

```
      template_overwrite => true
  }
}
```

5 Kibana

Kibana என்பதுElasticSearch-ல் இருக்கும் தரவுகளை வரைபடங்களாக மாற்றி வெளிப்படுத்தஉதவும் ஒரு Visual Interface ஆகும். ElasticSearch-ல் இருக்கும் தரவுகளை வைத்து ஒருசில முக்கிய முடிவுகளை எடுப்பதற்கு Kibana-வின் வரைபடங்கள் உதவுகின்றன. இதனை அறிக்கைக்கான கருவி (ReportingTool) என்றும் கூறலாம். அதாவது வெறும் எண்ணிக்கையினாலான தகவல்களை மட்டும்வைத்துக்கொண்டு ஒருசில முக்கிய முடிவுகளை எடுப்பது என்பது சற்று கடினமானவிஷயம். எனவேதான் Kibana-

வானது அவற்றை அழகிய வரைபடங்களாக மாற்றி, அதனைப் பார்க்கும் போதே தரவுகளின் சாராம்சங்களைப் புரிந்து கொள்ளக் கூடியஅளவுக்கு சுலபமான வகையில் வெளிப்படுத்துகிறது. இத்தகைய வரைபடங்களைவைத்துக்கொண்டு நாம் எளிதில் அனைத்தையும் புரிந்து கொள்ள முடியும்.Microsoft Excel-ல் உள்ள graphs மற்றும் Reporting-காகவே வடிவமைக்கப்பட்டஇன்னபிற கருவிகள்கூட இவ்வேலையைத்தான் செய்யும் என்றாலும், அவைகளால் அதிகபட்சம் ஒரு மில்லியன் வரைதான் தரவுகளைத் தாங்க முடியும். அதற்கும் மேலாகதரவுகள் வரும்போது அவை படுத்துவிடும். ஆனால் Kibana-வோ சிறப்பாகசெயல்படும். அதிக எண்ணிக்கையிலான தரவுகளையும்

வரைபடங்களாக மாற்றுவதற்குKibana-வே சிறப்பாக அமைகிறது.

இது உலாவியில் இயங்கக்கூடிய கருவி ஆகும். பொதுவாக 5601 எனும் Port எண்ணில் இயங்கிக் கொண்டிருக்கும். இதில் Management, Discover, Visualize, Dashboard ஆகிய முக்கியப் பகுதிகள் உள்ளன. இவை ஒவ்வொன்றின் செயல்பாடுகளைப் பற்றியும் பின்வருமாறு காணலாம்.

5.1 Management

Kibana உலாவியில் இயங்கத் தொடங்கியவுடன் இப்பகுதிதான் default-ஆக வந்து அமையும். இதுவே ES-ல் உள்ள index-ஐ kibana-க்குள் ஏற்றம் செய்தல், ஏற்றம் செய்த index-ஐ kibana-விலிருந்து நீக்குதல், வரைபடங்களாக சேமிக்கப்பட்டவற்றைப் பட்டியலிடுதல் மற்றும் அவைகளில் தேவையில்லாதவற்றை நீக்குதல் போன்ற அனைத்து விதமான “ஆக்கல், அழித்தல், நிர்வகித்தல்” போன்ற வேலைகளைச் செய்கிறது. இதில் உள்ள ‘Index Pattern’ என்பதன் மீது சொடுக்கினால் ‘Configure an Index Pattern’ எனும் பக்கம் வெளிப்படும்.

ES– க்குள் தற்போது interviews, earthquakes, twitter_elastic_example ஆகிய 3 indices உள்ளன. இவை அனைத்தையும் kibana-க்குள் ஏற்றம் செய்வதற்கான படிகள் பின்வருமாறு அமையும்.

1. Index name or pattern எனக் கேட்கும் பெட்டியில் interviews -எனக் கொடுக்கவும். இதையடுத்து தானாகவே இதில் உள்ள DOB

எனும் field கீழிறக்கப் பெட்டியிலிருந்து தேர்வு செய்யப்படுவதைக் காணலாம்.

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

Index name or pattern

Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

interviews

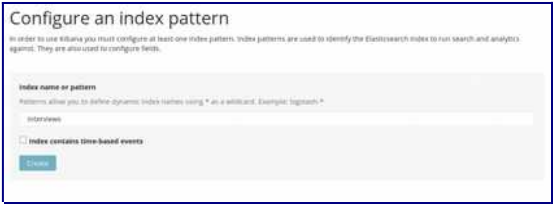
☑ Index contains time-based events

Time-field name: DOB

☐ Use event times to create index names [DEPRECATED]

2. பொதுவாக kibana-க்குள் ஏற்றம் செய்யப்படும் நிகழ்வுகள் அனைத்தும் காலநேர அடிப்படையில்தான் காட்சிப்படுத்தப்படுகின்றன. எனவே நமது index-ல் காலநேர மதிப்பினைப் பெற்று விளங்கக் கூடிய ஒரு field இயல்பாகவே

தேர்வு செய்யப்பட்டு வெளிப்படுகிறது. இதை நாம் நீக்க விரும்பினால் ‘Index contains time-based events’ எனும் தேர்வினை நீக்கி விடவும். இது பின்வருமாறு அமையும்.



Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

Index name or pattern

Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

interviews

☐ **Index contains time-based events**

3. இப்போது Create எனும் பொத்தானின் மீது சொடுக்கவும். இது interviews-ல் உள்ள நிகழ்வுகள் அனைத்தையும் பின்வருமாறு ஏற்றம் செய்து வெளிப்படுத்தும். இந்த interviews-ன் இடப்புறத்தில் ஒரு நட்சத்திரக்குறி இருப்பதை

கவனிக்கவும். இது default index என்பதைக் குறிக்கிறது. அதாவது Discover, Visualize போன்ற kibana- வின் மற்ற பகுதிகளுக்குச் செல்லும்போது இதன் தரவுகள் தான் default- ஆக வெளிப்படும்.

4. இதே முறையில் earthquakes, twitter_elastic_example

ஆகியவற்றையும் ஏற்றம் செய்து விடவும். ஆனால் இவைகளை காலநேர அடிப்படையிலான நிகழ்வுகளாக ஏற்றம் செய்யவும். அதாவது இயல்பாக இருக்கும் ‘Index contains time-based events’ எனும் தேர்வினை நீக்கி விடாமல், தானாகவே வெளிப்படும் @timestamp எனும் மதிப்பினையும் மாற்றிவிடாமல், create-ன் மீது சொடுக்கவும்.

5. இப்போது Management-பக்கத்தின் இடப்புறத்தில் நாம் ஏற்றம் செய்த அனைத்து index-ன் பட்டியலையும் காணலாம். பொதுவாக முதலில் ஏற்றம் செய்யப்படுவதுதான் default index-ஆக அமையும். இதை நாம் மாற்ற விரும்பினால் இடப்புறப் பட்டியலிலிருந்து நமக்கு வேண்டிய index-ஐ தேர்வு செய்து வலது கோடியில் உள்ள நட்சத்திரக்குறி மீது சொடுக்கவும். இங்கு twitter_elastic_example என்பது default-ஆக மாற்றப்பட்டுள்ளது.

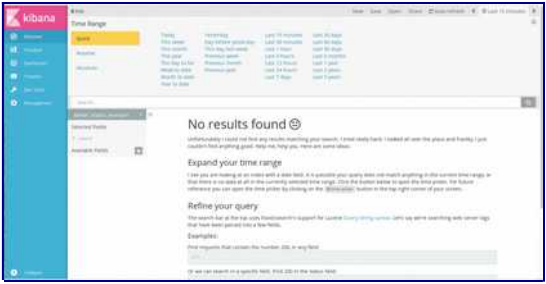
II. Discover

இப்பகுதியில் Kibana-க்குள் ஏற்றம் செய்யப்பட்ட அனைத்து indices மற்றும் அவைகளின் முழுமையான தரவுகள் அனைத்தும் காணப்படும். இதிலிருந்து நமக்கு வேண்டிய தரவினை மட்டும் தேடி எடுத்து அத்தரவுத் தொகுப்புக்கு ஒரு பெயர் வைத்து

சேமித்து வைத்துக்கொள்ளலாம். பின்னர் தேவைப்படும் நேரங்களில் அப்பெயரின் மீது சொடுக்கினால் அத்தரவுத் தொகுப்புகள் நமக்கு வெளிப்படுவதைக் காணலாம். இதனை நாம் பிறருடன் பகிர்ந்து கொள்ளவும் செய்யலாம். இவற்றில் உள்ள முக்கியப் படிகள் பின்வருமாறு.

1. இப்பகுதியில் default-ஆக தேர்வு செய்யப்பட்டுள்ள twitter_elastic_example எனும் index-ன் தரவுகள் காணப்பட வேண்டும். ஆனால் இங்கே ‘No results found’ என்பது வெளிப்பட்டுள்ளது. என்ன காரணம் என்று பார்த்தால் New, Save, Open, Share, Auto-refresh ஆகியவை அமைந்துள்ள பொத்தான்களின் வரிசையில் கடைசியாக ‘Last

15 minutes’ எனும் கால-நேர வகை தேர்வு செய்யப்பட்டுள்ளது. இது Kibana-வின் இயல்பான தேர்வு ஆகும். அதாவது கடைசி 15 நிமிடத்திற்குள் ES-ல் செலுத்தப்பட்ட தரவுகள் மட்டுமே இங்கு வெளிப்பட வேண்டும் என்று அர்த்தம். ஆனால் இத்தரவுகள் ES-ல் செலுத்தப்பட்டு கிட்டத்தட்ட 1 மாதம் ஆகியிருக்கும். எனவே இதற்கேற்றார் போன்ற ஒரு கால-நேர வகையைத் தேர்வு செய்ய வேண்டும். பின்வரும் 3 விதங்களில் இதனை நாம் தேர்வு செய்யலாம்.



No results found ☹

Expand your time range

Refine your query

Examples:

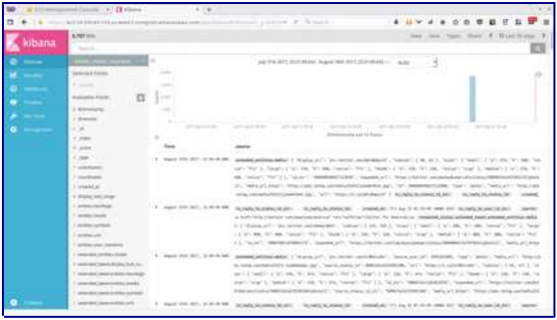
1.1 Quick: சுருக்காக ஒரு குறிப்பிட்ட கால இடைவெளியில் அமையும் தரவுகளை வெளிப்படுத்துவதற்கு இவ்வகை உதவும். இதில் தென்படும் கால-நேர மதிப்புகளை மேற்கண்ட படத்தில் காணலாம்.

1.2 Relative: இதில் உள்ள மதிப்புகள் எதிலிருந்து எதுவரை என்று நம்மிடம் கேட்பது

போல் பின்வருமாறு அமையும்.

1.3 நமக்கு வேண்டிய தரவுகளை தேதி, நேர மற்றும் நிமிட சுத்தமாகப் பெறுவதற்கு இவ்வகை உதவும். இது பின்வருமாறு.

Quick-லிருந்து ‘Last 30 days’ என்று தேர்வு செய்யும்போது தரவுகள் வெளிப்படுவதைக் காணலாம்.



2. இதில் தரவுகள் அனைத்தும் Time , Source எனும் 2 பிரிவுகளின் கீழ் வெளிப்பட்டுள்ளத்தைக் காணலாம். ஏனெனில் நாம் இதனை ஏற்றம் செய்யும்போதே Time based events-ஆக அமைத்துள்ளோம்.

இங்கு வெளிப்பட்டுள்ள வரைபடமும் இதன் அடிப்படையிலேயே அமைந்துள்ளது. ஆனால் இந்த வரைபடத்துக்கும் தரவுகளுக்கும் உள்ள ஒரே ஒரு வித்தியாசம் என்னவெனில், அனைத்துத் தரவுகளும் Aug 25-ம் தேதியில் ES-க்குள் செலுத்தப்பட்டிருப்பதால் அந்தத் தேதியில் மட்டும் bar வரையப்பட்டுள்ளத்தை கவனிக்கவும். ஆனால் வெளிப்பட்டுள்ள தரவகளோ அந்த ஒரே தேதியில் அமையாமல் வினாடி நேர வேறுபாட்டின் அடிப்படையில் அமைந்துள்ளது. இதேபோன்று வரைப்படத்தையும் மாற்ற அதன் இயல்பான தேர்வான Auto என்பதனை Millisecond என்று மாற்றி விடவும். பின்னர் அந்த ஒரே ஒரு bar வரையப்பட்டுள்ள பகுதியின் மீது cursor மூலம் drag செய்து விட்டால், அப்பகுதியில் உள்ள தரவுகள் அனைத்தும் வினாடி நேர வித்தியாசத்தில் பிரிக்கப்படுவதைக் காணலாம்.

இது பின்வருமாறு.

3. அடுத்ததாக ஒருசில முக்கியத் தரவுகளை மட்டும் எவ்வாறு எடுத்து வைத்துக்கொள்வது என்று பார்க்கலாம்.

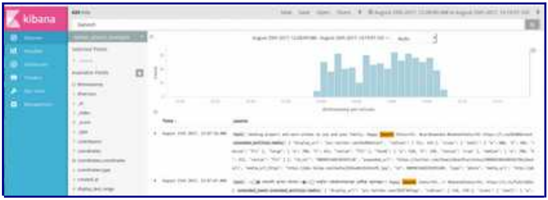
twitter_elastic_example எனும் index-ல் "விநாயகர் சதுர்த்தி" என்று tweet செய்தவர்களின் தரவுகள் அனைத்தும் இடம்பெற்றுள்ளது. இதில், □ எத்தனைபேர் #Vinayakar என்று பதிவிட்டுள்ளனர்□ எத்தனை பேர் k-க்குப் பதிலாக g-ஐப் பயன்படுத்தி #Vinayagar என்று பதிவிட்டுள்ளனர் □ எத்தனை பேர் #Ganesh

எனும் வார்த்தையைப் பயன்படுத்தி பதிவிட்டுள்ளனர் என்று தேடி அவற்றை தனித்தனியாக சேமிக்கப் போகிறோம். தேடலுக்கான பெட்டியில் Vinayakar எனக் கொடுத்து தேடவும். 2 பேர் இவ்வாறு பதிவிட்டுள்ளனர் என்று பின்வருமாறு வெளிப்படுத்தும்.

இதனை Save-ன் மீது சொடுக்கி அதற்கான ஒரு பெயரை அளித்து (No.of.Vinayakar)

சேமித்து வைத்துக்கொள்ளவும்.

4. பின்னர் மீண்டும் New-ன் மீது சொடுக்கி Vinayagar , Ganesh என்றெல்லாம் தேடி அவற்றையும் சேமித்து வைத்துக் கொள்ளவும்.



இப்போது,

- No.of.Vinayakar எனும் பெயரில் 2 தரவுகளும்
- No.of.Vinayagar எனும் பெயரில் 13 தரவுகளும்
- No.of.Ganesh எனும் பெயரில் 420 தரவுகளும் சேமிக்கப்பட்டுள்ளன

இவ்வாறாக நமக்கு வேண்டிய தரவுகளை மட்டும் தேடி அதனை ஒரு பெயர் கொடுத்து சேமித்து வைத்துக் கொள்ளலாம். Open-ன் மீது சொடுக்கி நமக்கு வேண்டிய தரவுத் தொகுப்பினை எடுத்துக் கொள்ளலாம்.

5. Share என்பது நாம் சேமித்துள்ளவற்றை பிறருடனோ அல்லது பிற இடங்களிலோ பகிந்து கொள்ள உதவும். இதில்,

- Share saved search எனும் வகையில் உள்ள இணைப்பைப் பயன்படுத்திப் பகிர்ந்தால், நாம் பகிரக்கூடிய நபரால் அதில் மாற்றம் செய்ய இயலும்.
- Share Snapshot எனும் வகையில் பகிர்ந்தால், எவ்வித மாற்றமும் செய்ய இயலாது. அது ஒரு image-ஆக மட்டுமே பகிரப்படும்.

6. அடுத்ததாக interviews எனும் index-ஐ தேர்வு செய்தால் அது பின்வருமாறு வெளிப்படுவதைக் காணலாம்.

இதில் காணப்படும் தரவுகள் கால-நேர அடிப்படையில் பிரிக்கப்படவில்லை. எனவே அதற்கான வரைபடமும் இல்லை. ஏனெனில் இவை Kibana-க்குள் ஏற்றம்

செய்யப்படும்போதே ‘இது time-based events அல்ல’ என்று கூறுவதுபோல் அத்தேர்வினை நீக்கிவிட்டே ஏற்றம் செய்தோம். ஆகவே வெறும் தரவுகள் மட்டுமே வெளிப்பட்டுள்ளது. இதேபோல் Discover-ல் அடுத்தடுத்த index-ஐ தேர்வு செய்து மேற்கூறிய அனைத்தையும் வெவ்வேறு தரவுகளுக்கு செய்து பார்க்கவும்.

5.2 III. Visualize

Visualize என்பது தரவுகளுக்கு ஏற்றார் போன்று வரைபடங்களை வரைந்து காட்டுகின்ற ஒரு பகுதி ஆகும். இதில் உள்ள படிகளைப் பின்வருமாறு காணலாம்.

1. முதலில் ‘Create a visualization’ என்பதின்மீது சொடுக்கவும்.

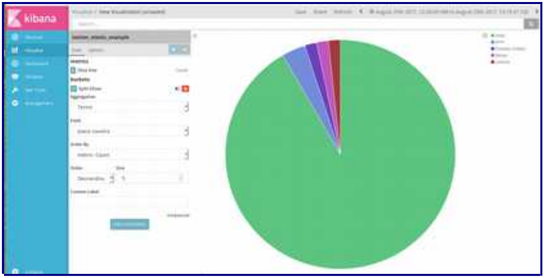
இது “எந்த வடிவில் உங்களுக்கு வரைபடங்கள் வேண்டும்” என்று கேட்பது போல் Select visualization type எனும் பக்கத்தினை வெளிப்படுத்தும். இதில் பல்வேறு விதமான வரைபட வகைகள் காணப்படும். இதில் ஒன்றை நாம் தேர்வு செய்ய வேண்டும். Pie என்பதனைத் தேர்வு செய்து கொள்ளவும்.

2. பின்னர் நாம் தேர்வு செய்திருக்கின்ற pie - க்கான மூலத் தரவுகளை நாம் புதிதாகத் தேடி எடுக்கப் போகின்றோமா அல்லது ஏற்கனவே தேடி எடுத்து வைத்த தரவுகளுக்கு வரைபடங்களை உருவாக்கப் போகின்றோமா என்று கேட்பது போல, From a New Search or From a Saved Search என்று 2 பகுதிகள் வெளிப்படும்.

- From a New Search: Kibana- ல் உள்ள அனைத்து index-ன் பட்டியலும் இங்கு காணப்படும்.
- From a Saved Search: நாம் சேமித்து வைத்த அனைத்துத் தரவுகளின் பட்டியலும் இங்கு காணப்படும்.

நாம் ‘From a New Search’ என்பதிலிருந்து twitter_elastic_example-என்பதனைத் தேர்வு செய்து கொள்ளலாம்.

3. பின்னர் எந்தெந்த நாடுகளிலிருந்து எவ்வளவு பேர் பதிவிட்டுள்ளனர் என்பதற்கான pie வரைபடம் பின்வருமாறு அமையும்.



4. இதனை 'Country_wise_split' என்று ஒரு பெயர் கொடுத்து சேமித்து வைத்துக் கொள்ளவும்.

5. இப்போது Visualize-ல் நாம் சேமித்து வைத்த வரைபடங்களின் பட்டியல் காணப்படும்.

இதே போன்று பல்வேறு வகையான வரைபடங்களை பல்வேறுபட்ட தரவுகளுக்கு வரைந்து பார்க்கவும்.

5.3 IV. Dashboard

Dashboard என்பது பல்வேறு வரைபடங்களை ஒன்றாக இணைத்தோ அல்லது புதிதாக வரைபடங்களை வரைந்து அவற்றை ஒன்றுடன் ஒன்று இணைத்தோ காட்சிப்படுத்த

உதவும் ஒரு செயல்பாடு ஆகும். பல்வேறு விதமான தரவுகளின் அடிப்படையில் உருவாக்கப்பட்ட வரைபடங்களின் தொகுப்பிற்கு Dashboard என்று பெயர். எடுத்துக்காட்டுக்கு twitter_elastic_example எனும் index-ல்,

- மொத்தம் எத்தனை tweets செய்யப்பட்டுள்ளன
- அவை எந்தெந்த நாடுகளிலிருந்து செய்யப்பட்டுள்ளன
- அவற்றுள் Ganesh, Vinayakar, Vinayagar போன்ற வெவ்வேறு வார்த்தையைப் பயன்படுத்தி tweet செய்தவர்களின் மொத்த எண்ணிக்கை

ஆகிய 3 தனித்தனி விவரங்களை, 3 தனித்தனி வரைபடங்களாக வரைந்து அவற்றை ‘Twitter Dashboard’ எனும் பெயரில் தொகுத்து உருவாக்குவது எவ்வாறு என்று பின்வருமாறு காணலாம்.

1. முதலில் ‘Create a dashboard’ என்பதின்மீது சொடுக்கவும்.

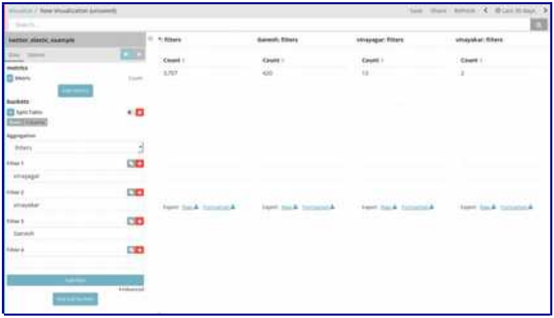
2. Add Panels - என்பதில் கீழ்க்கண்ட இரு பிரிவுகள் காணப்படும்.

- Visualization: இதில் வரைபடமாக மாற்றப்பட்ட தரவுகளின் பட்டியல் காணப்படும்.
- Saved Search: இதில் discover பகுதியில் தேடி எடுத்து சேமித்து வைத்த தரவுகளின் பட்டியல் காணப்படும்.

இவற்றில் Visualization-ன் கீழ் அமைந்துள்ள Country_wise_split ன் மீது சொடுக்கினால் அது dashboard panel-ல் இணைக்கப்பட்டுவிடும். பின்னர் Add new Visualization-ன் மீது சொடுக்கி ஒரு புது வரைப்படத்தை உருவாக்கவும்.

3. இது எத்தனைபேர் Ganesh, Vinayakar, Vinayagar போன்ற வெவ்வேறு வார்த்தையைப் பயன்படுத்தி tweet செய்துள்ளனர் என்பதற்கான 'Data Table' வகையச் சேர்ந்த வரைபடம் ஆகும். இதனை

Table எனும் பெயரில் சேமித்துக் கொள்ளவும்.



4. அவ்வாறே மொத்தம் எத்தனை tweets செய்யப்பட்டுள்ளன என்பதை 'Total Tweets' எனும் பெயரில் Metric வகை வரைபடமாக சேமித்துக் கொள்ளவும்.

5. கடைசியாக Add Panel-ல் சென்று drag & drop மூலம் மேற்குறிப்பிட்ட 3 வரைப்படத்தையும் இணைத்து பின்வருமாறு

ஒரு dashboard-ஐ உருவாக்கவும்.

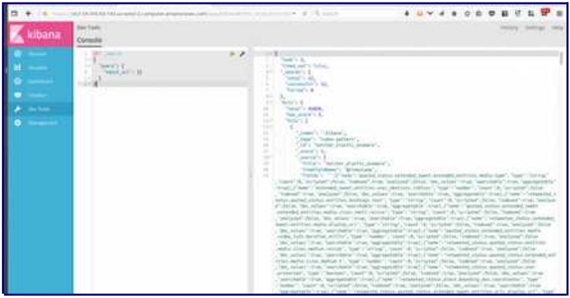
இதே முறையில் வெவ்வேறு index-க்கு வெவ்வேறு dashboards-ஐ உருவாக்கிப் பார்க்கவும்.

5.4 V. Dev Tools

இப்பகுதி கட்டளையை நேரடியாக execute செய்து அதற்கான விடையை சரிபார்க்கின்ற ஒரு இடம் ஆகும். இது Console-ஐப்

பிரதிபலிக்கும். Kibana- வின் மற்ற பகுதிகளில் நேரடியாக சிக்கலான query-ஐ எழுதி தரவுகளை எடுக்காமல், இப்பகுதியில் ஒருமுறை இயக்கி அதனை சரி

பார்த்துக்கொள்ளலாம்.



6 HADOOP

6.1 வரலாறு

Hadoop என்பது Apache நிறுவனம் வழங்குகின்ற திறந்த மூல மென்பொருள் கருவி ஆகும். இதனை Doug Cutting என்பவர் உருவாக்கினார். இது பெரிய தரவில் கூறப்படுகின்ற பல்வேறு வேலைகளையும் குறைந்த செலவில் திறம்பட செய்வதற்காக உருவாக்கப்பட்ட பல்வேறு

மென்பொருள்களின் கூட்டமைப்பு ஆகும். Hadoop உருவாக்கத்திற்கு முன்னர் Doug Cutting என்பவர் ‘Apache Lucene’ எனும் கருவியை உருவாக்கியிருந்தார். இக்கருவியைப் பற்றி நாம் ELK Stack-ல் ஏற்கனவே பார்த்துள்ளோம். வாக்கியங்கள்/வார்த்தைகளின் அடிப்படையில் துரிதமாகத் தேடல்களை நிகழ்த்துவதற்கு ஏற்ற வகையில் முதன்முதலில் வடிவமைக்கப்பட்டதே Apache’s Lucene எனும் இயந்திரம் ஆகும். அதற்கும் முன்னர் Apache Nutch எனும் ஒரு சிறிய அளவிலான கருவி இவ்வேலைக்காகப் பயன்பட்டது. அதாவது 1990 – 2000 ஆண்டுகளில் வலைத்தலங்களில் காணப்படும் தரவுகளின் எண்ணிக்கை மில்லியன், பில்லியன் என உயர ஆரம்பித்தபோது, நாம் தேடும் விவரங்களை விரைவாகப் பெறுவது என்பது சற்று

கடினமாகவும், நேர விரயம் தரக்கூடியதாகவும் இருந்தது. இப்பிரச்சனைகளை சமாளிப்பதற்காக Doug Cutting மற்றும் Mike Cafarella ஆகியோர் இணைந்து Nutch எனப்படும் ஒரு web crawler-ஐ உருவாக்கினர். இது வலைத்தளப்பக்கங்களை பதிவிறக்கம் செய்து தரவுகளை சேமித்துக்கொண்டே வரும் வல்லமை மிக்கது. பின்னர் இத்தகைய அதிக அளவிலான தரவுகளை முறையாகப் பிரித்து, சேமித்து, துரிதமாக எடுத்து வெளிப்படுத்துவதற்காக முதன்முதலில் உருவாக்கப்பட்ட ஒரு தேடுபொறிதான் Apache Lucene ஆகும். ஆனால் இத்தகைய Nutch, Lucene ஆகியவற்றின் செயல் திறன்களும் ஒரு குறிப்பிட்ட அளவுக்கே இருந்தன. இவற்றால் அதிக அளவிலான தகவல்களைக் கையாள முடியவில்லை. இந்நிலையில்தான் 2004-ஆம்

ஆண்டு Google நிறுவனம் ‘Google Distributed Filesystem (GFS)’ எனும் கட்டமைப்பினை உருவாக்கி அது பற்றிய ஆய்வு அறிக்கைகளை வெளியிட்டது. அதனைத் தொடர்ந்து 2005-ம் ஆண்டு Mapreduce- ஐ உருவாக்கி அதன் ஆய்வு அறிக்கைகளை வெளியிட்டது. இவ்விரண்டும் மிக அதிக அளவிலான (terabyte கணக்கில்) தரவுகளை சேமித்தும், அவற்றை ஒருசில நொடிகளில் பகுத்தாய்வு செய்து வெளிப்படுத்தியும் திறம்பட செயல்பட்டு வந்தன.

இவைகளைப் பின்பற்றியே அதற்கு அடுத்தடுத்த ஆண்டுகளில் NDFS (Nutch Distributed File System) மற்றும் Nutch- க்கான ‘Mapreduce’ ஆகியவை google வெளியிட்ட ஆய்வு அறிக்கைகளின்

அடிப்படையில் உருவாக்கப்பட்டன. 2006-ம் ஆண்டு ஆண்டில் தான் Doug Cutting, yahoo-நிறுவனத்தில் இணைந்து Nutch திட்டத்தை விரிவுபடுத்தும் வேலையைச் செய்தார். இதன்படி வலைத்தளப்பக்கங்களை அப்படியே பதிவிறக்கம் செய்து சேமித்துக்கொண்டே செல்லும் web crawler-க்கு Nutch என்றும். அவற்றை முறையாகப் பகுத்து பிரித்தாய்ந்து சேமிக்கும் முறைக்கு Hadoop என்றும் பெயரிடப்பட்டன . அதாவது Apache Lucene-ல் காணப்பட்ட ஒருசில சிரமங்களைக் களைந்து அதன்செயல்பாட்டுத் திறன்களையும் விரிவுபடுத்தி உருவாக்கப்பட்ட ஒன்றே Hadoop ஆகும். இதற்கு அவர் Hadoop-எனப் பெயரிட்டதற்கும், யானை பொம்மையை சின்னமாக வைத்ததற்கும் ஒரு தற்செயலான நிகழ்வே காரணமாக அமைந்தது. அவரது

அன்புக்குரிய 2 வயது மகன் ஆசையோடு எடுத்து விளையாடும் மஞ்சள் நிற யானை பொம்மையின் பெயர்தான் இந்த Hadoop. இதைப்பற்றி அவர் கூறுகையில் "இப்பெயருக்கு எந்தஒரு அர்த்தமும் இல்லை. இது சொல்வதற்கு சுலபமாகவும், கவர்ச்சிகரமாகவும், வேறெங்கும் பயன்படுத்தாத வகையிலும் இருப்பதால் இப்பெயரையே வைத்துவிட்டேன். குழந்தைகள் எப்போதுமே இதுபோன்ற விஷயங்களை உருவாக்குவதில் சிறந்தவர்கள். Googol என்பதும் ஒரு குழந்தையின் உருவாக்கம்தானே" என்று குறிப்பிட்டுள்ளார்.

Hartonworks எனும் நிறுவனம் 2011-ம் ஆண்டு Rob Bearden என்பவரால் உருவாக்கப்பட்டது. இவர் இந்நிறுவனத்தை

Yahoo-வுடன் இணைந்து, Hadoop உருவாக்கத்திற்கு உதவிய அதே மென்பொருளாளர் குழுவை வைத்து உருவாக்கினார். இந்நிறுவனம் பெரிய தரவு மேளாண்மைக்காக Apache-ன் hadoop கருவிகளில் ஒருசில முக்கியக் கருவிகளை இணைத்து ஒரு கட்டமைப்பாக நமக்கு வழங்குகிறது. இதன்மூலம் பயணர்கள் அவர்களுக்கு வேண்டிய கருவிகளை தனித்தனியாக சிரமப்பட்டு நிறுவத் தேவையில்லை. இந்தக் கட்டமைப்புக்குள் சென்று வேண்டிய கருவிகளைத் தேர்வு செய்தால் போதுமானது. அவை அனைத்தும் சுலபமாக எவ்விதப் பிரச்சனையும் இல்லாமல் நமது கணினியில் நிறுவப்பட்டுவிடும். இதுவே HDP (Hartonworks Data Platform) என்று அழைக்கப்படுகிறது. Ambari என்பது HDP-ஐ நிறுவி மேளாண்மை செய்யும்

வேலையை பயணர்களுக்கு சுலபமாக்குவதற்காக உலாவி வழியே 8080-எனும் port-ல் இயங்குகின்ற ஒரு கருவி ஆகும்.

6.2 HADOOP கட்டமைப்பு

“Hadoop ஒரு தனிக்கருவி அல்ல; பல்வேறு சிறுசிறு கருவிகளின் கூட்டமைப்பே !” என்று ஏற்கனவே பார்த்தோம். இதன் மிக முக்கிய ஆங்கங்களாக HDFS மற்றும் MapReducer ஆகியவை விளங்குகின்றன. இவை முறையே தரவுகளை சேமித்தல் மற்றும் பகுத்தாய்தல் ஆகிய முக்கியப் பணிகளைச் செய்கின்றன. இவற்றுடன் சேர்த்து Pig, Hive, HBase, Phoenix, Spark, ZooKeeper, Flume, Sqoop, Oozie, மற்றும் Storm ஆகிய கருவிகளையும் Apache நிறுவனம்

வழங்குகிறது. இவற்றை நமது தேவைக்கேற்ப Hadoop கட்டமைப்பில் இணைத்தோ /நீக்கியோ பயன்படுத்திக்கொள்ளலாம். இந்தக் கூட்டமைப்பில் இடம் பெற்றுள்ள கருவிகளையும், அவை எந்தெந்த இடங்களில் பங்களிக்கின்றன என்பதையும் பின்வருமாறு காணலாம்.

1. தரவுகளை சேமித்தல் (Storage) : HDFS, HBase
2. சேமித்தவற்றை நெறிப்படுத்துதல் (Processing) : MapReduce, YARN
3. நெறிமுறைப்படுத்தப்பட்டவற்றை தேவைக்கேற்றபடி பெறுதல் (Access) : Pig, Hive, Mahout, Avro, Sqoop
4. தரவுகளின் மேளாண்மை (Management) : Oozie, Chukwa, Flume, Zoo Keeper

Hadoop கட்டமைப்பின் முதல் வெளியீட்டில் (v1) தரவுகளை சேமிப்பதற்கு hdfs-ம் அவற்றிலிருந்து தரவுகளை நெறிமுறைப்படுத்தி அடுத்த நிலைக்கு பக்குவப்படுத்துவதற்கு mapreduce-ம் பயன்படுத்தப்பட்டன. அவ்வாறே அதன் இரண்டாவது வெளியீட்டில் (v2) சேமிக்கும் வேலைக்கு hbase-ம் பதப்படுத்தும் வேலைக்கு yarn-ம் வெளிவந்தன. மற்ற கருவிகள் அனைத்திற்கும் பொதுவானவை. இவை ஒவ்வொன்றைப்பற்றியும் பின்வருமாறு காணலாம்.

6.2.1 HDFS

Hadoop Distributed File System என்பதுதான் அனைத்திற்கும் அடித்தளம். இது

அதிக அளவிலான கோப்புகளை அணுகி சேமிக்க வல்லது. இதன் ஆற்றல் ஒரே ஒரு கணினியில் சேமிப்பதோடு நின்றுவிடாமல், கோப்புகளின் எண்ணிக்கை பெருகப் பெருக பல்வேறு கணினியில் அவற்றைப் பிரித்து சேமிக்க வல்லது. இவ்வாறு ஒன்று போன்ற தகவல்களைப் பிரித்து சேமித்துள்ள அனைத்துக் கணினிகளும் cluster முறையில் இணைக்கப்பட்டு அவற்றில் ஒரு கணினி server-ஆகவும் மற்றவை clients-ஆகவும் செயல்புரியத் துவங்கும். server-ல் உள்ளது master node எனவும், clients-ல் உள்ளது data node எனவும் அழைக்கப்படும். இதில்தான் தரவுகள் சென்று சேரும்.

6.2.2 Hbase

Hadoop database என்பதுதான் Hbase என்றழைக்கப்படுகிறது. இது NoSQL-ஐ அடிப்படையாக வைத்து உருவாக்கப்பட்டது. எனவே வடிவற்ற தகவல்களையும் சேமிக்கும் திறன் பெற்று விளங்குகிறது. ஆனால் கோப்பு முறையில் தரவுகள் சேமிக்கப்படுவதிலிருந்து இது வித்தியாசப்பட்டு விளங்குகிறது. ஒரு database-ல் சேமிப்பது போன்று இதில் தரவுகள் சேமிக்கப்படுகின்றன. எனவே திடீரென இடையில் தகவல்களைப் போடுவதோ எடுப்பதோ இங்கு சுலபமாகிறது.

6.2.3 MapReducer

சேமிக்கப்பட்ட தகவல்களை பதப்படுத்தி, பக்குவப்படுத்தி, அடுத்த நிலைக்கு வேண்டியவாறு நெறிமுறைப்படுத்தும் வேலையை Mapreducer செய்கிறது. இது Mapper மற்றும் Reducer எனும் இருவேறு algorithmns ஆகும். இது cluster முறையில் இணைக்கப்பட்டுள்ள அனைத்துக் கருவிகளிலிருந்தும் தகவல்களை அணுக வல்லது. இதனுடைய சிறப்பே அதன் வேகம்தான். அதாவது இத்தகைய வேலைகளைச் செய்வதற்கு rdbms 20 மணி நேரம் எடுத்துக்கொள்கின்றது என்றால், இக்கருவி அதனை 3 மூன்றே நிமிடங்களில் முடித்துவிடும்.

6.2.4 YARN

இது ஒன்றோடொன்று இணைந்து செயல்படுகின்ற அனைத்து கணினிகளையும் மேளாண்மை செய்வதற்கும், அவற்றுக்குள் வேலைகளை ஒதுக்கீடு செய்து தொடர்ச்சியாக அவற்றைக் கண்காணிப்பதற்கும் பயன்படுகின்ற cluster மேளாண்மைக்கான ஒரு மென்பொருள் கருவியாகும். Yet another resource negotiator என்பது இதன் பொருள். இதுவும் mapreducer-ன் பணிகளையே செய்கிறது என்றாலும், புதுப்புது அணுகுமுறைகளில் தரவுகளைக் கையாண்டு அதன் பணிகளை இன்னும் சுலபமக்குகிறது. எனவே இதனை mapreducer-ன் இரண்டாம் தலைமுறை என்று கூறலாம்.

6.2.5 Pig

இது தரவுகளை எவ்விதம் மாற்ற வேண்டும், எவ்வாறு மாற்ற வேண்டும் (data transformation logic) என்பதை கணினிக்கு கூற உதவும் ஒரு நிரலாக்க மொழி ஆகும். SQL போன்று செயல்படக் கூடிய இந்த நிரலாக்க மொழி PigLatin என்றும், இதை வைத்து உருவாக்கப்படுபவை PigPrograms என்றும் அழைக்கப்படுகின்றன. இத்தகைய programs-தான் Mapreducer algorithmn-களில் எழுதப்பட்டு அவை hadoop cluster-ல் இயக்கப்படுகின்றன. Pig-ல் பயனர்கள் தாங்களே ஒரு function-ஐ வரையறுப்பதன் (User Defined Functions) மூலம் Jruby, Jython, Java போன்ற மற்ற நிரலாக்க மொழிகளுடனும்

இணைந்து செயல்பட முடியும். இதன் விளைவாக நிகழ்காலப் பிரச்சனைகளை அலசுவதற்கு உதவும் விதமாக உருவாக்கப்பட்ட ஒருசில பயன்பாட்டியல்களில் Pig ஒரு கருவியாக சுலபமாக இணைந்து செயல்பட வல்லது .

6.2.6 Hive

இதுவும் SQL போன்று செயல்படக் கூடிய ஒரு மொழிதான். ஆனால் Pig என்பது ஒரு scripting மொழியெனில், Hive என்பது ஒரு query மொழி. Transformation Logic-ஐ எழுதுவதற்கு நிரலாக்க மொழியில்

பழக்கமுள்ளவர்கள் Pig-ஐத் தேர்வு செய்வார்கள். ETL, Data Warehousing போன்ற துறைகளில் வேலைப்பார்த்து SQL-ல் அதிக பழக்கமுள்ளவர்கள் Hive-ஐத் தேர்வு செய்வார்கள் . மற்றபடி இரண்டும் கிட்டத்தட்ட ஒரே வேலையைச் செய்வதற்குத்தான் பயன்படுகின்றன. தரவுகளை அணுகும் முறைகளில்தான் இவை வேறுபட்டு விளங்குகின்றன.

6.2.7 Mahout

இது இயந்திரவழிக் கற்றலுக்கு(machine learning) உதவும் algorithmns-ஐக் கொண்ட ஒரு நூலகம் ஆகும். இதுவே செயற்கை அறிவுத்திறனை (Artificial

Intelligence) வளர்ப்பதற்கு அடித்தளமாக விளங்குகிறது. இப்பிரிவில் இயந்திரங்கள் என்ன செய்ய வேண்டும் என்பதனை வெளிப்படையாக நிரல்கள் மூலம் சொல்லத் தேவையில்லை. hdfs-ல் உள்ள அதிகப்படியான தரவுகளிலிருந்து தாமாகவே ஒரு pattern-ஐக் கண்டுபிடித்து, அடுத்த நிலைக்குச் செல்லும் திறன் பெற்றதாக இயந்திரங்களை மாற்றுவதற்கு இத்தகைய Mahout algorithmns உதவுகின்றன. இவற்றை நாம் Mapreducer-ல் பயன்படுத்தி hdfs-ல் உள்ள தரவுகளைக் கையாளச் செய்யலாம்.

6.2.8 Avro

Data Serialization-க்கு உதவுகின்ற ஒரு கோப்பு வகைதான் இந்த Avro. இதனையும் Doug Cutting என்பவர்தான் கண்டுபிடித்தார். ஒரு மாபெரும் தரவுத் தளத்திலுள்ள தரவுகளை எடுத்து json வடிவிலோ, binary வடிவிலோ கொடுப்பதற்கு Data Serialization என்று பெயர். உதாரணத்துக்கு 1TB அளவுகொண்ட தரவுகளை text வடிவில் மாற்றினால் அது 2TB அளவுடையதாகவும், அதையே binary வடிவான avro வடிவில் மாற்றினால் அது 500MB அளவுடையதாகவும் மாறிவிடும். இவ்வாறு அதிக அளவு கொண்டவற்றை குறைந்த அளவில் மாற்றுவதற்குத் தான் zip, tar ஆகியவை ஏற்கனவே உள்ளனவே என்று

நீங்கள் கேட்கலாம். ஆனால் இவையெல்லாம் ஒரே கணினியில் வேண்டுமானால் சுலபமாக unzip, untar செய்து பயன்படுத்தப்படலாம். இதுவே ஒரு cluster என்று வரும்போது இத்தகைய வடிவங்கள் உதவாது. அதற்கென சிறப்பாக உருவாக்கப்பட்ட ஒரு கோப்புவடிவம் தான் இந்த Avro.

Avro வடிவில் சுருக்கப்பட்ட தரவுகளைப் பிரித்து பல்வேறு கணினிகளுக்கு அனுப்பும்போது, எந்த ஒரு தகவல் இழப்பும் ஏற்படாமல், அவைகள் பிரிக்கப்பட்ட அளவில் தனித்தனியே process செய்யப்படும். இதற்கென Compressible மற்றும் splittable எனும் இரு வடிவங்களில் binary கோப்புகள் உருவாக்கப்படுகின்றன. இதுவே இதன் சிறப்பம்சம் ஆகும். மேலும் இத்தகைய

கோப்புகளை பல்வேறு வகை நிரலாக்க மொழிகளாலும் எடுத்துப் படிக்க முடியும். Avro schema-வானது json வடிவில் வரையறுக்கப்பட்டு முழுமையாக அதைச் சார்ந்தே செயல்படுகிறது. எனவே இது மொழி சார்பற்றதாகவும் schema-based ஆகவும் விளங்குகிறது.

6.2.9 Sqoop

தரவுத்தளம் / தரவுக்கிடங்கு (database / datastores) ஆகியவற்றிலிருந்து hadoop-க்கு மொத்தமாக (bulk-data processing)தரவுகளைப் பரிமாற்றம் செய்வதற்கு sqoop (SQL + Hadoop) பயன்படுகிறது. என்பதே sqoop ஆகும். ETL-

க்குத் துணைபுரியும் வகையில் தரவுகள் அனைத்தையும் தரவுக்கிடங்கிலிருந்து(EDW) hadoop-க்கு மாற்றியும், அவ்வாறே hadoop-லிருந்து கிடங்கிற்கு மாற்றியும் குறைந்த செலவில் திறம்பட செயலாற்றுவதற்கு இது உதவுகிறது. மேலும் Teradata, Mysql, Oracle, Postgre போன்ற அனைத்து வகையான தரவுத்தளத்துடனும் இது இணைந்து செயலாற்ற வல்லது.

6.2.10 Oozie

இது ஒரு workflow schedular ஆகும். ஒன்றுடன் ஒன்று சார்பு கொண்ட தொடர்ச்சியான வேலைகளை அடுத்தடுத்து இயக்கப் பயன்படும். அதாவது தனிச்சையாக இயங்கக்கூடிய வேலைகளைக் கையாள்வது

சுலபம். ஆனால் ஒரு வேலையின் வெளியீட்டை வைத்துத்தான் மற்றொன்று தன் செயல்பாட்டைத் துவங்கவேண்டுமெனில், அதற்கு இந்த Oozie பயன்படுகிறது. இதில் DAG (Directed Acyclical Graphs) எனும் வரைப்படத்தை உருவாக்கி, அதன் மூலம் பல்வேறு வேலைகளின் தொடர்புகளையும், அவை இயக்கப்பட வேண்டிய வரிசைமுறைகளையும் தெளிவாக விளக்கலாம். இதில் ஒரு node செயலுக்காகவும், மற்றொன்று செயல்களை தீர்மானிக்கவும் பயன்படுகிறது.

Action node என்பது செயலுக்கானது. Hdfs-ல் கோப்புகளை கொண்டு சேர்த்தல், mapreducer-ஐ இயக்குதல், pig/hive-ன் இயக்கங்களைத் துவக்குதல், பல்வேறு மூலங்களிலிருந்து தரவுகளை இறக்குதல்

ஆகியன இதில் அடங்கும். Control node என்பது செயல்களை தீர்மானிக்கக் கூடியது. அதாவது ஒரு வேலையின் முடிவினை வைத்து அடுத்து எங்கு செல்ல வேண்டும், எந்த விதமான முடிவுக்கு எங்கிருந்து அடுத்தடுத்த வேலைகளைத் துவங்க வேண்டும் போன்ற அனைத்தையும் தீர்மானிக்கிறது. Start, End, Error ஆகிய nodes அனைத்தும் இந்த செயல்களைத் தீர்மானிக்கும் node வகையின் கீழ் வருகின்றன.

6.2.11 Chukwa

பொதுவாக hadoop என்பது hdfs-க்குள் ஏற்றப்பட்ட நிலையான தகவல்களை எடுத்து தனது வேலைகளைத் துவங்கும். ஆனால் பல்வேறு கணிணிகளிடமிருந்து தொடர்ச்சியாக

வருகின்ற logs அனைத்தையும் hdfs-க்குள் செலுத்தி அதனை ஆய்வு செய்வதற்கு இத்தகைய chukwa, flume போன்ற கருவிகள் பயன்படுகின்றன. இதில் agent மற்றும் collector என்று இரண்டு அமைப்புகள் உள்ளன. Agent என்பது தரவுகளை எடுத்து அனுப்புவதற்காக ஒவ்வொரு கணினியிலும் செயல்படுவது. Collector என்பது agent அனுப்பிய தரவுகளைப் பெற்று hdfs-க்குள் அனுப்புவது.

Chukwa எனும் இக்கருவி collection, processing போன்ற பல்வேறு நிலைகளைக் கொண்ட ஒரு குழாய் போன்ற செயல்பாட்டமைப்பினைப் பெற்றிருக்கும். இதன் ஒவ்வொரு நிலையும் பல்வேறு இடைமுகப்புகளால் இணைக்கப்பட்டிருக்கும்.

எனவே பிற்காலத்தில் புதுப்புது மாறுதல்கள் வந்தாலும்கூட, அதன் தற்போதைய நிலையில் எந்தஒரு குறைப்பாடும் இருக்காது. HICC (Hadoop Infrastructure Care Center) என்பதும் தகவல்களை வெளிப்படுத்துவதற்கு உதவுகின்ற வலைப்பக்கத்திற்கான ஒரு இடைமுகப்பு ஆகும். இதன்மூலம் தகவல்களை திறம்பட வெளிப்படுத்தி, கண்காணித்து அதனடிப்படையில் தேர்ந்த முடிவுகளை எடுக்கலாம்.

6.2.12 Flume

இதுவும் chukwa போன்றே batch processing-க்கு உதவுகின்ற, ஆனால் அதைவிட மிகக் குறைந்த நேரத்தில் logs-ஐ எடுத்து அனுப்பும் ஒரு கருவியாகும். Chukwa

என்பது 5 நிமிடத்திற்கு ஒருமுறை batch முறைப்படி logs-ஐ எடுத்து அனுப்பினால், Flume-ஆனது சமகாலத்தில் வந்து கொண்டிருக்கின்ற logs-ஐ எடுத்து உடனுக்குடன் அனுப்புகிறது. இது logs விழ விழ அதனை எடுத்து உடனுக்குடன் Continuous Streaming முறைப்படி அனுப்பிக்கொண்டே இருக்கும்.

6.2.13 Zoo Keeper

இக்கருவி பிரிந்து இயங்குகின்ற பல்வேறு பயன்பாட்டியல்களை ஒருங்கிணைக்கப் பயன்படுகிறது. உதாரணத்துக்கு cluster-ல் இயங்கிக்கொண்டிருக்கும் 50-க்கும் மேற்பட்ட கணிணிகளில் ஏதேனும் ஒன்றில் கோளாறு ஏற்பட்டால் கூட அது எந்த node, எங்கிருந்து

வருகிறது , மாற்றாக என்ன செய்வது போன்ற அனைத்து விதமான கோணங்களிலிருந்தும் யோசித்து , மாற்று செயல்பாடுகளுக்கான வேலையைத் துவங்கி அவை அனைத்தையும் பராமரிக்கும் வேலையை Zoo Keeper செய்கிறது.

7 HDFS, Mapreducer

ஒரே ஒரு கணினியில் hadoop-ஐ நிறுவினால் அது single node cluster-எனவும், பல்வேறு server-களை இணைத்து நிறுவினால் அது multi-node cluster எனவும் அழைக்கப்படும். இங்கு Ubuntu 16.04 எனும் கணினியில் நிறுவுவது பற்றி பார்க்கலாம்.

1. Hadoop எனும் கட்டமைப்பு Java-ல் எழுதப்பட்டிருப்பதால், முதலில் நமது

கணினியில் Java நிறுவப்பட்டுள்ளதா என்பதை $ java -version எனக் கொடுத்து சோதிக்கவும். இது பின்வருமாறு ஒரு வெளியீட்டைக் கொடுத்தால் java நிறுவப்பட்டுள்ளது என்று அர்த்தம். இல்லையெனில் பின்வருமாறு கொடுத்து அதனை நிறுவவும்.

```
$ sudo apt-get install default-jdk
```

```
nithya@nithya-Lenovo-ideapad:~$ java -version
openjdk version "1.8.0_151"
OpenJDK Runtime Environment (build 1.8.0_151-8u151-b12-0ubuntu0.17.10.2-b12)
OpenJDK 64-Bit Server VM (build 25.151-b12, mixed mode)
```

2. பின்னர் hadoop-க்கான முகவரியிலிருந்து அதனை பதிவிறக்கம் செய்யவும். அதனைப் பிரித்து தரவிறக்கப்பட்ட இடத்திலிருந்து நமக்கு வேண்டிய இடத்திற்கு மாற்றிக் கொள்ளவும். மாற்றப்பட்ட directory-க்கான முழு அனுமதிகளையும் பயனருக்கு வழங்க வேண்டும்.

```
$ wget http://redrockdigimark.com/apachemirror/hadoop/common/hadoop-3.0.0/hadoop-3.0.0.tar.gz
$ tar -xzvf hadoop-3.0.0.tar.gz
$ sudo mv ~/hadoop-3.0.0 /usr/local/hadoop
```

```
$ sudo chown -R nithya:nithya
/usr/local/hadoop
```

```
nithya@nithya-Lenovo-ideapad:~$ wget http://redrockdigimark.com/apachemirror/had
oop/common/hadoop-3.0.0/hadoop-3.0.0.tar.gz
--2018-03-19 15:12:34--  http://redrockdigimark.com/apachemirror/hadoop/common/h
adoop-3.0.0/hadoop-3.0.0.tar.gz
Resolving redrockdigimark.com (redrockdigimark.com)... 119.18.61.94
Connecting to redrockdigimark.com (redrockdigimark.com)|119.18.61.94|:80... conn
ected.
HTTP request sent, awaiting response... 200 OK
Length: 306392917 (292M) [application/x-gzip]
Saving to: ‘hadoop-3.0.0.tar.gz’

hadoop-3.0.0.tar.gz 100%[====================>] 292.20M  2.13MB/s    in 2m 5s

2018-03-19 15:14:39 (2.33 MB/s) - ‘hadoop-3.0.0.tar.gz’ saved [306392917/3063929
17]
```

```
nithya@nithya-Lenovo-ideapad:~$ sudo mkdir /usr/local/hadoop
nithya@nithya-Lenovo-ideapad:~$ sudo mv ~/hadoop-3.0.0 /usr/local/hadoop
nithya@nithya-Lenovo-ideapad:~$ sudo chown -R nithya:nithya /usr/local/hadoop
```

3. Hadoop என்பது கணினிகளுடன் பேசுவதற்கு Secured Shell (SSH) -ஐப் பயன்படுத்துகிறது. அப்போது நாம்

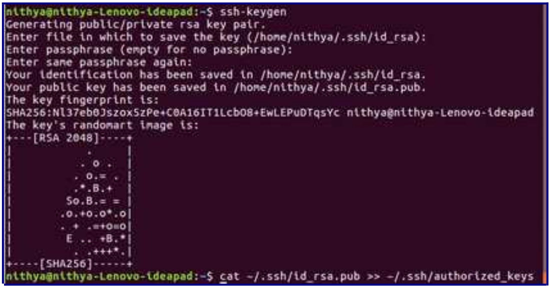
ஒவ்வொருமுறையும் சரியான கடவுச்சொல்லை அளித்தால் மட்டுமே அதனுடன் பேச முடியும். இதனைத் தவிர்க்க SSH-ன் சான்றிதழை எடுத்து அதனை அங்கீகரிக்கப்பட்ட திறவுகோல்களின் பட்டியலில் இணைத்தால் போதுமானது. ஒவ்வொரு கணினியும் கடவுச்சொல் இல்லாமலேயே தங்களுடன் ஒன்றோடொன்று பேசிக்கொள்ள முடியும்.

```
$ ssh-keygen
$ cat ~/.ssh/id_rsa.pub >>
~/.ssh/authorized_keys
```

கோப்பின் பெயர் மற்றும் passphrase ஆகியவற்றைக் கேட்டு இக்கட்டளை நிற்கும் போதெல்லாம் ஒன்றும் அளிக்காமல் enter-ஐ

மட்டும் கொடுக்கவும்.

இது நமது home directory-ல் .ssh எனும் folder-ஐ உருவாக்கி, அதற்குள் பின்வருமாறு கோப்புகளை உருவாக்கும்.



```
nithya@nithya-Lenovo-ideapad:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/nithya/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/nithya/.ssh/id_rsa.
Your public key has been saved in /home/nithya/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:Nl37eb0Jszox5zPe+C0A16IT1LcbO8+EwLEPuDTqsYc nithya@nithya-Lenovo-ideapad
The key's randomart image is:
+---[RSA 2048]----+
|            .    |
|          . o .  |
|         . o.= . |
|         .*.B.+  |
|        So.B.= = |
|       .o.+o.o*.o|
|       . + .=+o=o|
|        E .. +B.*|
|         . .+++*.|
+----[SHA256]-----+
nithya@nithya-Lenovo-ideapad:~$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

- id_rsa (தனிப்பட்ட திறவுகோல்)
- id_rsa.pub (பொதுவான திறவுகோல்)
- authorised_keys (அங்கீகரிக்கப்பட்ட திறவுகோல்கள்)

இதில் id_rsa.pub என்பதுதான் SSH-க்கான சான்றிதழ். இதனை நாம் authorised_keys என்பதற்குள் போட வேண்டும். இதேபோல் ஒன்றோடொன்று இணைக்கப்பட்டுள்ள ஒவ்வொரு கணினியிலிருந்தும் அதனதன் id_rsa.pub கோப்பினை SCP செய்து அவற்றையும் இந்த authorised_keys-க்குள் போட வேண்டும். இவ்வாறு செய்தால் ஒவ்வொரு கணினியும் கடவுச்சொல் இல்லாமலேயே தங்களுடன் பேசிக் கொள்ள முடியும்.

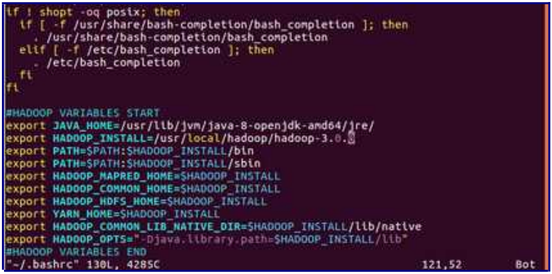
4. அடுத்ததாக நமது home directory-ல் உள்ள .bashrc எனும் கோப்பிற்குள் கடைசியில் சென்று பின்வரும் விவரங்களை இணைக்க வேண்டும். ஒருசில மதிப்புகள் தெரியவில்லையெனில் echo

$Variablename எனக் கொடுத்து அதன் மதிப்பினைப் பெறலாம். உதாரணத்துக்கு echo $JAVA_HOME எனும் கட்டளை நமது கணினியில் java நிறுவப்பட்டுள்ள இடத்தின் பாதையைக் காட்டும். அதை எடுத்து இங்கு பயன்படுத்தலாம்.

```
$ vim ~/.bashrc
#HADOOP VARIABLES START
export
JAVA_HOME=/usr/lib/jvm/java-8-
openjdk-amd64/jre/
export HADOOP_INSTALL=/usr/local/
hadoop/hadoop-3.0.0
export PATH=$PATH:
$HADOOP_INSTALL/bin
```

```
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
```

```
#HADOOP VARIABLES END
$ source ~/.bashrc
```



```
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre/
export HADOOP_INSTALL=/usr/local/hadoop/hadoop-3.0.
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
#HADOOP VARIABLES END
"~/.bashrc" 130L, 4285C                                        121,52        Bot
```

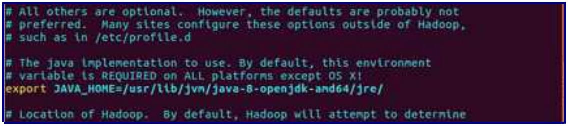
```
nithya@nithya-Lenovo-ideapad:~$ vim ~/.bashrc
nithya@nithya-Lenovo-ideapad:~$ source ~/.bashrc
nithya@nithya-Lenovo-ideapad:~$
```

5. பின்னர் hadoop folder-க்குள் உள்ள configuration files-ல் செய்ய வேண்டிய மாற்றங்களைப் பின்வருமாறு காணலாம். Hadoop ஒவ்வொரு முறை செயல்படத் துவங்கும்போதும் இதில் காணப்படுகின்ற மதிப்புகளை வைத்தே தனது செயல்பாட்டினைத் துவக்கும் .

5.1. hadoop-env.sh என்பதற்குள் பின்வருமாறு சேர்க்கவும் அல்லது ஏற்கனவே comment செய்யப்பட்டு இருப்பின் uncomment செய்துவிடவும்.

```
$ sudo vim
/usr/local/hadoop/hadoop-3.0.0/et
```

```
c/hadoop/hadoop-env.sh
JAVA_HOME=/usr/lib/jvm/java-8-
openjdk-amd64/jre/
```



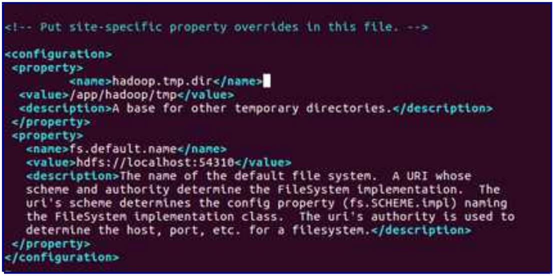
```
# All others are optional.  However, the defaults are probably not
# preferred.  Many sites configure these options outside of Hadoop,
# such as in /etc/profile.d

# The java implementation to use. By default, this environment
# variable is REQUIRED on ALL platforms except OS X!
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre/

# Location of Hadoop.  By default, Hadoop will attempt to determine
```

5.2. core-site.xml என்பதற்குள் config. tags-க்கிடையில் பின்வருமாறு சேர்க்கவும். பின்னர் இதில் குறிப்பிட்டுள்ளது போலவே ஒரு directory-ஐ உருவாக்கி அதற்கு முழு அனுமதிகளையும் வழங்க வேண்டும்.

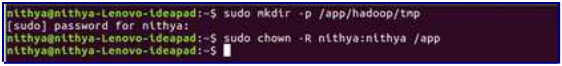
```
$ sudo vim
/usr/local/hadoop/hadoop-3.0.0/et
c/hadoop/core-site.xml
<configuration>
<property>
<name>hadoop.tmp.dir</name>
<value>/app/hadoop/tmp</value>
<description>A base for other
temporary
directories.</description>
</property>

<property>
<name>fs.default.name</name>
<value>hdfs://localhost:54310</va
lue>
<description>The name of the
```

```
default file system. A URI whose
scheme and authority determine
the FileSystem implementation.
The
uri's scheme determines the
config property (fs.SCHEME.impl)
naming
the FileSystem implementation
class. The uri's authority is
used to
determine the host, port, etc.
for a filesystem.</description>
</property>
</configuration>
```

```
$ sudo mkdir -p /app/hadoop/tmp
$ sudo chown -R nithya:nithya
/app
```



```
<!-- Put site-specific property overrides in this file. -->

<configuration>
 <property>
        <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
  <description>A base for other temporary directories.</description>
 </property>
 <property>
   <name>fs.default.name</name>
   <value>hdfs://localhost:54310</value>
   <description>The name of the default file system.  A URI whose
   scheme and authority determine the FileSystem implementation.  The
   uri's scheme determines the config property (fs.SCHEME.impl) naming
   the FileSystem implementation class.  The uri's authority is used to
   determine the host, port, etc. for a filesystem.</description>
 </property>
</configuration>
```

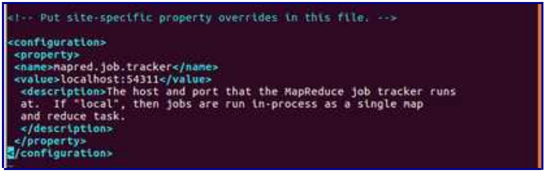


```
nithya@nithya-Lenovo-ideapad:~$ sudo mkdir -p /app/hadoop/tmp
[sudo] password for nithya:
nithya@nithya-Lenovo-ideapad:~$ sudo chown -R nithya:nithya /app
nithya@nithya-Lenovo-ideapad:~$
```

5.3. mapred-site.xml.template என்று இருந்தால் mapred-site.xml என பெயர்மாற்றம் செய்து விடவும். பின்னர் அதன் config. tags-க்கிடையில் பின்வருமாறு சேர்க்கவும்.

```
$ sudo mv
/usr/local/hadoop/hadoop-3.0.0/et
c/hadoop/mapred-site.xml.template
/usr/local/hadoop/hadoop-3.0.0/et
c/hadoop/mapred-site.xml
```

```
$ sudo vim
/usr/local/hadoop/hadoop-3.0.0/et
c/hadoop/mapred-site.xml
<configuration>
<property>
<name>mapred.job.tracker</name>
<value>localhost:54311</value>
<description>The host and port
that the MapReduce job tracker
runs
at. If "local", then jobs are run
in-process as a single map
and reduce task.
</description>
</property>
<configuration>
```



```
<!-- Put site-specific property overrides in this file. -->

<configuration>
 <property>
 <name>mapred.job.tracker</name>
 <value>localhost:54311</value>
  <description>The host and port that the MapReduce job tracker runs
  at.  If "local", then jobs are run in-process as a single map
  and reduce task.
  </description>
 </property>
</configuration>
```

5.4. hdfs-site.xml என்பதற்குள் config. tags-க்கிடையில் பின்வருமாறு சேர்க்கவும். பின்னர் இதில் குறிப்பிட்டுள்ளது போலவே directories-களை உருவாக்கி அதற்கு முழு அனுமதிகளையும் வழங்கவும்.

```
$ sudo vim
/usr/local/hadoop/hadoop-3.0.0/et
c/hadoop/hdfs-site.xml
```

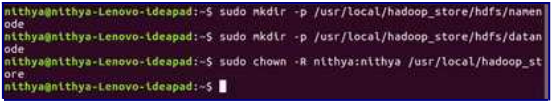
```
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
<description>Default block
replication.
The actual number of replications
can be specified when the file is
created.
The default is used if
replication is not specified in
create time.
</description>
</property>
<property>
<name>dfs.namenode.name.dir</name
>
<value>file:/usr/local/hadoop_sto
```

```
re/hdfs/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name
>
<value>file:/usr/local/hadoop_sto
re/hdfs/datanode</value>
</property>
<configuration>
```

```
$ sudo mkdir -p
/usr/local/hadoop_store/hdfs/name
node
```

```
$ sudo mkdir -p
/usr/local/hadoop_store/hdfs/data
node
$ sudo chown -R nithya:nithya
/usr/local/hadoop_store
```

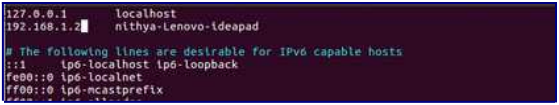
```
<!-- Put site-specific property overrides in this file. -->

<configuration>
 <property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is created.
  The default is used if replication is not specified in create time.
  </description>
 </property>
 <property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
 </property>
 <property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
 </property>
</configuration>
~
```



```
nithya@nithya-Lenovo-ideapad:~$ sudo mkdir -p /usr/local/hadoop_store/hdfs/namen
ode
nithya@nithya-Lenovo-ideapad:~$ sudo mkdir -p /usr/local/hadoop_store/hdfs/datan
ode
nithya@nithya-Lenovo-ideapad:~$ sudo chown -R nithya:nithya /usr/local/hadoop_st
ore
nithya@nithya-Lenovo-ideapad:~$ 
```

6. பின்னர் /etc/hosts கோப்பிற்குள் அனைத்துக் கணினிகளின் IP முகவரி மற்றும் hostname-ஐச் சேர்க்க வேண்டும்.

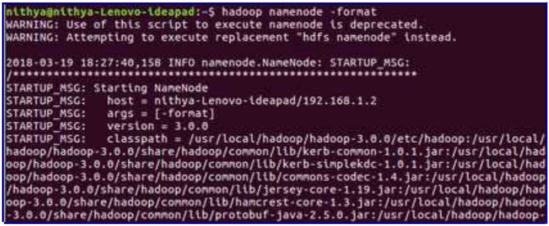
```
$ sudo vim /etc/hosts
192.168.1.2 nithya-Lenovo-ideapad
```



```
127.0.0.1       localhost
192.168.1.2     nithya-Lenovo-ideapad

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
```

7. கடைசியாக hfs-ஐ ஒருமுறை format செய்து பயன்படுத்தத் தொடங்கவும். துவங்கிய பின் இடையில் எப்போதும் இக்கட்டளையை அளிக்கக் கூடாது . அவ்வாறு செய்தால் hfs-ல் சேமிக்கப்பட்டுள்ள அனைத்துத் தகவல்களும் அழிக்கப்பட்டு datanode-ல் பிரச்சனை ஏற்பட்டுவிடும்.

```
$ hadoop namenode -format
```



```
nithya@nithya-Lenovo-ideapad:~$ hadoop namenode -format
WARNING: Use of this script to execute namenode is deprecated.
WARNING: Attempting to execute replacement "hdfs namenode" instead.

2018-03-19 18:27:40,158 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = nithya-Lenovo-ideapad/192.168.1.2
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 3.0.0
STARTUP_MSG:   classpath = /usr/local/hadoop/hadoop-3.0.0/etc/hadoop:/usr/local/
hadoop/hadoop-3.0.0/share/hadoop/common/lib/kerb-common-1.0.1.jar:/usr/local/had
oop/hadoop-3.0.0/share/hadoop/common/lib/kerb-simplekdc-1.0.1.jar:/usr/local/had
oop/hadoop-3.0.0/share/hadoop/common/lib/commons-codec-1.4.jar:/usr/local/hadoop
/hadoop-3.0.0/share/hadoop/common/lib/jersey-core-1.19.jar:/usr/local/hadoop/had
oop-3.0.0/share/hadoop/common/lib/hamcrest-core-1.3.jar:/usr/local/hadoop/hadoop
-3.0.0/share/hadoop/common/lib/protobuf-java-2.5.0.jar:/usr/local/hadoop/hadoop-
```

8. sbin எனுமிடத்தில் ஒவ்வொரு services-ஐயும் தனித்தனியாகத் துவக்குவதற்குத் தேவையான shell scripts அனைத்தும் காணப்படும். start-dfs.sh எனும் கட்டளை

namenode, datanode, secondary namenode ஆகியவற்றையும், start-yarn.sh எனும் கட்டளை resourcemanager,nodemanagers ஆகியவற்றையும் துவக்கும். தனித்தனியாகத் துவக்குவதற்குப் பதிலாக start-all.sh எனக் கொடுத்து அனைத்தையும் ஒரே நேரத்தில் துவக்கலாம். jps எனக் கொடுத்து services வெளிப்படுகின்றனவா என உறுதி செய்து கொள்ளவும். அவ்வாறே stop-all.sh என்பது நிறுத்த உதவும்.

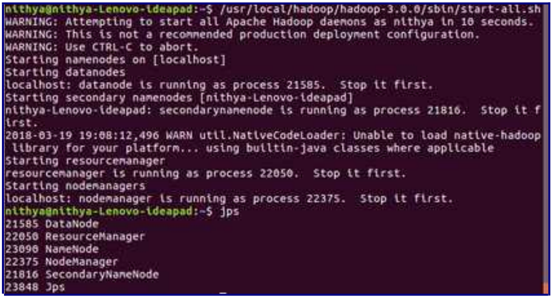
```
$ /usr/local/hadoop/hadoop-3.0.0/
sbin/start-all.sh
```

```
$ jps
$ stop-all.sh
```

```
nithya@nithya-Lenovo-ideapad:~$ /usr/local/hadoop/hadoop-3.0.0/sbin/start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as nithya in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [nithya-Lenovo-ideapad]
2018-03-19 18:59:02,260 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
Starting resourcemanager
Starting nodemanagers
nithya@nithya-Lenovo-ideapad:~$ jps
22560 Jps
21585 DataNode
22050 ResourceManager
22375 NodeManager
21816 SecondaryNameNode
```

இதில் Namenode வெளிப்படவில்லை. எனவே namenode, datanode ஆகியவற்றில் உள்ள கோப்புகளை எல்லாம் நீக்கிவிட்டு மீண்டும் ஒருமுறை format செய்து துவக்கவும் .

```
$ sudo rm -rf
/usr/local/hadoop_store/hdfs/name
node/*
$ sudo rm -rf
/usr/local/hadoop_store/hdfs/data
node/*
$ hadoop namenode -format
```



```
nithya@nithya-Lenovo-ideapad:~$ /usr/local/hadoop/hadoop-3.0.0/sbin/start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as nithya in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
localhost: datanode is running as process 21585.  Stop it first.
Starting secondary namenodes [nithya-Lenovo-ideapad]
nithya-Lenovo-ideapad: secondarynamenode is running as process 21816.  Stop it f
irst.
2018-03-19 19:08:12,496 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
Starting resourcemanager
resourcemanager is running as process 22050.  Stop it first.
Starting nodemanagers
localhost: nodemanager is running as process 22375.  Stop it first.
nithya@nithya-Lenovo-ideapad:~$ jps
21585 DataNode
22050 ResourceManager
23090 NameNode
22375 NodeManager
21816 SecondaryNameNode
23848 Jps
```

இப்போது NameNode வெளிப்படுவதைக் காணலாம்.

9. பின்வரும் கட்டளை hadoop-ன் இயக்கத்தை உறுதி செய்யும்.

```
$ /usr/local/hadoop/hadoop-3.0.0/
bin/hadoop
```

```
nithya@nithya-Lenovo-ideapad:~$ /usr/local/hadoop/hadoop-3.0.0/bin/hadoop
Usage: hadoop [OPTIONS] SUBCOMMAND [SUBCOMMAND OPTIONS]
 or    hadoop [OPTIONS] CLASSNAME [CLASSNAME OPTIONS]
  where CLASSNAME is a user-provided Java class

  OPTIONS is none or any of:

buildpaths                       attempt to add class files from build tree
--config dir                     Hadoop config directory
--debug                          turn on shell script debug mode
--help                           usage information
hostnames list[,of,host,names]   hosts to use in slave mode
hosts filename                   list of hosts to use in slave mode
loglevel level                   set the log4j level for this command
workers                          turn on worker mode

  SUBCOMMAND is one of:
```

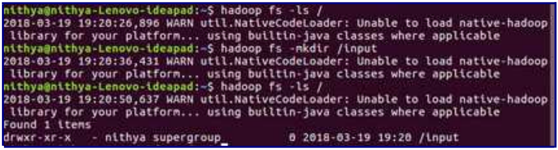
10. hdfs-க்குள் தகவல்களைக் கையாள்வதற்கு உதவும் கட்டளைகளைப் பற்றிப் பார்க்கலாம். இது பெரும்பாலும் linux கட்டளைகளை

ஒத்திருக்கும். linux கட்டளைகளுக்கு முன்னர் hadoop fs - என்பது இடம் பெற்றிருக்கும்.

```
$ hdfs fs -ls / (hdfs-ன் root
directory-ல் உள்ளவற்றைப்
பட்டியலிடும்)
$ hadoop fs -mkdir /input (root-
ல் input எனும் folder-ஐ உருவாக்கும்)
$ hadoop fs -put ~/Kanchi.txt
/input (இது ஓரிடத்திலுள்ள
கோப்பினை hdfs-க்குள் செலுத்தும் i.e
upload)
$ hadoop fs -copyFromLocal
~/Kanchi.txt /input (இதுவும் put
போன்றதே. ஆனால் நமது கணினியில்
இருந்து மட்டும்தான் பிரதி செய்யும்)
$ hadoop fs -get
```

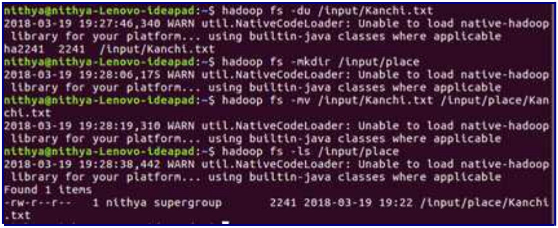
```
/input/Kanchi.txt ~/ (இது hdfs-ல்
உள்ள கோப்பினை ஓரிடத்தில் செலுத்தும்
i.e download)
$ hadoop fs - copyToLocal /input/
Kanchi.txt ~/ (இதுவும் get
போன்றதே. ஆனால் hdfs-ல் உள்ள
கோப்பினை நமது கணினியில்
மட்டும்தான் செலுத்தும்)
$ hadoop fs -cat
/input/Kanchi.txt (கோப்பிலுள்ள
தகவல்களை வெளிக்காட்டும்)
$ hadoop fs -tail
/input/Kanchi.txt (கோப்பிலிருந்து
கடைசி ஒருசில வரிகளை
வெளிப்படுத்தும்)
$ hadoop fs -du /input/Kanchi.txt
(கோப்பின் நீளத்தை காண உதவும்)
$ hadoop fs -mv /input/Kanchi.txt
```

```
/input/place/Kanchi.txt
(கோப்பினை இடமாற்றம் செய்யும்)
$ hadoop fs -rm
/input/place/Kanchi.txt
(கோப்பினை நீக்கும்)
$ hadoop fs -rmr /input
(கோப்புகளுடன் சேர்த்து directory-ஐ
நீக்கும்)
```



```
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -ls /
2018-03-19 19:20:26,896 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -mkdir /input
2018-03-19 19:20:36,431 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -ls /
2018-03-19 19:20:50,637 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
Found 1 items
drwxr-xr-x   - nithya supergroup_         0 2018-03-19 19:20 /input
```

```
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -rm /input/place/Kanchi.txt
2018-03-19 19:31:29,401 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
Deleted /input/place/Kanchi.txt
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -rmdir /input
2018-03-19 19:31:44,339 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
rmdir: `/input': Directory is not empty
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -rmr /input
rmr: DEPRECATED: Please use '-rm -r' instead.
2018-03-19 19:31:59,739 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
Deleted /input
```

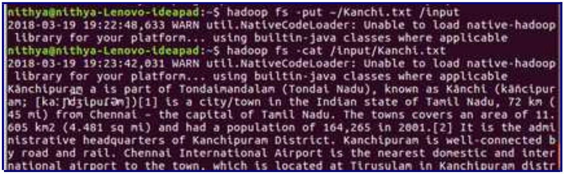


```
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -du /input/Kanchi.txt
2018-03-19 19:27:46,340 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
ha2241  2241  /input/Kanchi.txt
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -mkdir /input/place
2018-03-19 19:28:06,175 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -mv /input/Kanchi.txt /input/place/Kan
chi.txt
2018-03-19 19:28:19,310 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -ls /input/place
2018-03-19 19:28:38,442 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
Found 1 items
-rw-r--r--   1 nithya supergroup       2241 2018-03-19 19:22 /input/place/Kanchi
.txt
```

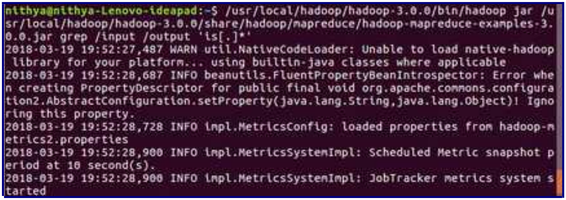
11. ஏற்கனவே உள்ள mapreducer program-ஐ வைத்து ஒரு கோப்பிற்குள் ஒரு குறிப்பிட்ட வார்த்தை எத்தனை முறை

இடம்பெற்றுள்ளது என்பதைக் கண்டுபிடிக்கலாம். இதற்கான கட்டளை பின்வருமாறு.

```
$ /usr/local/hadoop/hadoop-3.0.0/
bin/hadoop jar /usr/local/hadoop/
hadoop-3.0.0/share/hadoop/mapredu
ce/hadoop-mapreduce-examples-
3.0.0.jar grep /input /output
'is[.]*'
```



```
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -put ~/Kanchi.txt /input
2018-03-19 19:22:48,633 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -cat /input/Kanchi.txt
2018-03-19 19:23:42,031 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
Kānchipuram a is part of Tondaimandalam (Tondai Nadu), known as Kānchi (kāñcipur
am; [kaːɲdʒipuɾəm])[1] is a city/town in the Indian state of Tamil Nadu, 72 km (
45 mi) from Chennai - the capital of Tamil Nadu. The towns covers an area of 11.
605 km2 (4.481 sq mi) and had a population of 164,265 in 2001.[2] It is the admi
nistrative headquarters of Kanchipuram District. Kanchipuram is well-connected b
y road and rail. Chennai International Airport is the nearest domestic and inter
national airport to the town, which is located at Tirusulam in Kanchipuram distr
```



```
nithya@nithya-Lenovo-ideapad:~$ /usr/local/hadoop/hadoop-3.0.0/bin/hadoop jar /u
sr/local/hadoop/hadoop-3.0.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.
0.0.jar grep /input /output 'is[.]*'
2018-03-19 19:52:27,487 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
2018-03-19 19:52:28,687 INFO beanutils.FluentPropertyBeanIntrospector: Error whe
n creating PropertyDescriptor for public final void org.apache.commons.configura
tion2.AbstractConfiguration.setProperty(java.lang.String,java.lang.Object)! Igno
ring this property.
2018-03-19 19:52:28,728 INFO impl.MetricsConfig: loaded properties from hadoop-m
etrics2.properties
2018-03-19 19:52:28,900 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot p
eriod at 10 second(s).
2018-03-19 19:52:28,900 INFO impl.MetricsSystemImpl: JobTracker metrics system s
tarted
```

இதற்கான விளக்கம்:

- முதலில் நமது கணினியில் உள்ள கோப்பினை hdfs-ன் ரூட் (/) க்குள் செலுத்த வேண்டும்.

```
$ hadoop fs -put ~/Kanchi.txt /input
```

- அடுத்து hdfs-ல் சேமிக்கப்பட்டுள்ள கோப்பினை உள்ளீடாக எடுத்துக்கொண்டு அதிலுள்ள ஒவ்வொரு வார்த்தையும் எத்தனை முறை திரும்பத் திரும்ப வந்துள்ளது என்பதற்கான mapreducer program ஒரு jar file-ஆக ஏற்கனவே எழுதப்பட்டுள்ளது (hadoop-mapreduce-examples-3.0.0.jar) . இதனை hadoop directory-க்குள் share/hadoop/mapreduce எனுமிடத்தில் காணலாம். இக்கட்டளை grep மூலம் /input எனுமிடத்தில் உள்ள கோப்பினை உள்ளீடாக எடுத்துக்கொண்டு,

அதில் is என்பது எத்தனை முறை இடம்பெற்றுள்ளது என்பதைக் கண்டறிந்து /output எனுமிடத்தில் வெளிப்படுத்தியுள்ளது..

```
/usr/local/hadoop/share/hadoop/
mapreduce/hadoop-mapreduce-
examples-3.0.0.jar grep /input
/output 'is[.]*'
```

- கடைசியாக /usr/local/hadoop/bin/hadoop மூலம் இக்கட்டளை இயக்கப்படுகிறது. இது jar

எனும் parameter-ஐப் பயன்படுத்தி இந்த jar program-ஐ இயக்குகிறது. கடைசியாக /output எனுமிடத்தில் சென்று பார்த்தால் இதன் வெளியீடாக _SUCCESS , part-r-00000 எனுமிரண்டு கோப்புகள் இடம் பெற்றிருக்கும்.

- part-r-00000 எனும் கோப்பினைத் திறந்து பார்த்தால் is எனும் வார்த்தை 28 முறை இடம்பெற்றுள்ளது எனும் விவரம் காணப்படும்..

```
$ hadoop fs -ls /output
$ hadoop fs -cat /output/part-r-
00000
```

```
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -ls /output
2018-03-19 19:56:41,245 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--   1 nithya supergroup          0 2018-03-19 19:52 /output/_SUCCESS
-rw-r--r--   1 nithya supergroup          6 2018-03-19 19:52 /output/part-r-0000
0
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -cat /output/part-r-00000
2018-03-19 19:56:53,649 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
28      is
```

12. அடுத்ததாக பைதான் மொழியில் mapreducer program-ஐ நாமே உருவாக்கி, நமது கோப்பிற்குள் உள்ள ஒவ்வொரு வார்த்தையும் எத்தனை முறை இடம்பெற்றுள்ளது என்பதைக் கண்டுபிடிக்கலாம்.

Mapper - முதலில் நமது கோப்பில் உள்ள ஒவ்வொரு வார்த்தையின் பக்கத்திலும் 1 எனும் எண்ணைச் சேர்ப்போம். இதற்கான நிரல்கள்

எழுதப்பட்டுள்ள கோப்பினை mapper.py எனும் பெயரில் சேமிப்போம். ஒரே வார்த்தை திரும்பத் திரும்ப வந்தாலும் அவ்வார்த்தையின் பக்கத்தில் 1 என இடுவோம்.

mapper.py

```
#!/usr/bin/env python
import sys


for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for i in words:
        print '%s\t%s' % (i, 1)
```

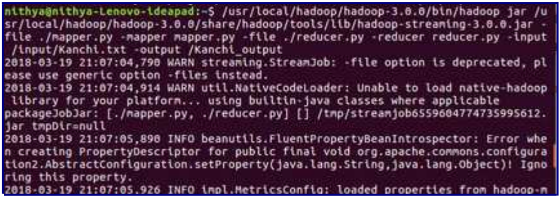
Reducer - ஒரே வார்த்தை திரும்பத் திரும்ப வந்தால் (mapper-ன் வெளியீட்டில்) அதன் பக்கத்தில் உள்ள எண்ணிக்கையை அடுத்தடுத்து அதிகரிப்பதற்கான நிரல் எழுதப்பட்டுள்ளது. இதற்காக ஒரு காலி dictionary உருவாக்கப்பட்டு இடப்புறத்தில் வார்த்தைகளும் வலப்புறத்தில் அவ்வார்த்தைகளின் எண்ணிக்கையும் சேமிக்கப்படுகிறது. பின்னர் அந்த dictionary-ன் மதிப்புகளை for loop-வைத்து வெளிப்படுத்துவதன் மூலம் ஒவ்வொரு வார்த்தையும் எத்தனை முறை இடம்பெற்றுள்ளது என்பதைக் கண்டறியலாம்..

reducer.py

```
#!/usr/bin/env python
import sys

wordcount={}

for line in sys.stdin:
    line = line.strip()
    word= line.split('\t')[0]
    if word not in wordcount:
        wordcount[word] = 1
    else:
        wordcount[word] += 1

for i in wordcount:
    print i, wordcount[i]
```

அதாவது mapper என்பது கொடுக்கப்பட்டுள்ள தகவல்களை key, value pairs-ஆக மாற்றுகிறது. Reducer என்பது இத்தகைய இணைகளை உள்ளீடாக எடுத்துக்கொண்டு வேலைகளைச் செய்கிறது. இதற்கான கட்டளை அமைப்பு பின்வருமாறு அமையும். -file எனும் parameter நாம் எழுதியுள்ள நிரல்களை hadoop-streaming வுடன் இணைக்கிறது. அவ்வாறே -input, -output ஆகிய parameters மூலம் உள்ளீட்டினை எங்கிருந்து எடுக்க வேண்டும், வெளியீட்டினை எங்கு போட வேண்டும் ஆகிய விவரங்களை அளிக்கிறது.

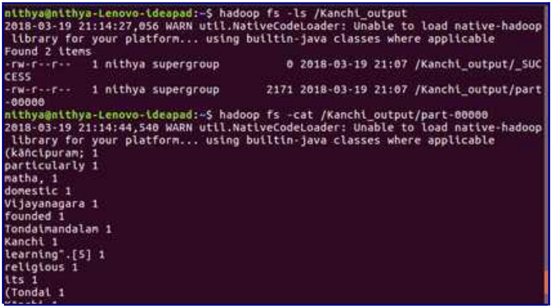
```
$ /usr/local/hadoop/hadoop-3.0.0/
bin/hadoop jar
/usr/local/hadoop/hadoop-3.0.0/sh
are/hadoop/tools/lib/hadoop-
streaming-3.0.0.jar
-file ./mapper.py -mapper
mapper.py
-file ./reducer.py -reducer
reducer.py
-input /input/Kanchi.txt
-output /Kanchi_output
```



```
nithya@nithya-Lenovo-ideapad:~$ /usr/local/hadoop/hadoop-3.0.0/bin/hadoop jar /u
sr/local/hadoop/hadoop-3.0.0/share/hadoop/tools/lib/hadoop-streaming-3.0.0.jar -
file ./mapper.py -mapper mapper.py -file ./reducer.py -reducer reducer.py -input
 /input/Kanchi.txt -output /Kanchi_output
2018-03-19 21:07:04,790 WARN streaming.StreamJob: -file option is deprecated, pl
ease use generic option -files instead.
2018-03-19 21:07:04,914 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
packageJobJar: [./mapper.py, ./reducer.py] [] /tmp/streamjob6559604774735995612.
jar tmpDir=null
2018-03-19 21:07:05,890 INFO beanutils.FluentPropertyBeanIntrospector: Error whe
n creating PropertyDescriptor for public final void org.apache.commons.configura
tion2.AbstractConfiguration.setProperty(java.lang.String,java.lang.Object)! Igno
ring this property.
2018-03-19 21:07:05,926 INFO impl.MetricsConfig: loaded properties from hadoop-m
```

இது இயங்கி முடித்த பின்னர் வெளியீட்டுக்கான இடத்தில் _SUCCESS , part-00000 ஆகிய இரு கோப்புகள் செலுத்தப்பட்டிருப்பதைக் காணலாம். இதில் இரண்டாவது கோப்பில் ஒவ்வொரு வார்த்தையும் எத்தனை முறை இடம்பெற்றுள்ளது என்பது காணப்படுகிறது.

```
$ hadoop fs -ls /Kanchi_output
$ hadoop fs -cat
/Kanchi_output/part-00000
```



```
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -ls /Kanchi_output
2018-03-19 21:14:27,056 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--   1 nithya supergroup          0 2018-03-19 21:07 /Kanchi_output/_SUC
CESS
-rw-r--r--   1 nithya supergroup       2171 2018-03-19 21:07 /Kanchi_output/part
-00000
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -cat /Kanchi_output/part-00000
2018-03-19 21:14:44,540 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
(kāñcipuram; 1
particularly 1
matha, 1
domestic 1
Vijayanagara 1
founded 1
Tondaimandalam 1
Kanchi 1
learning".[5] 1
religious 1
its 1
(Tondai 1
```

Mapper மற்றும் Reducer இரண்டையும் ஒரே program-ல் எழுதி விடலாமே என்று நீங்கள் கேட்கலாம். இது இரண்டுக்குமான வித்தியாசத்தைப் பற்றிக் கூற வேண்டுமானால், நாம் சிறு வயதில் பல வண்ண நிறங்களில் சீரக மிட்டாய் வாங்கி இருப்போம் அல்லவா ! அப்போது அதை உண்பதற்கு முன்னர் ஒவ்வொரு நிறத்திலும் எத்தனை சீரக மிட்டாய் உள்ளது என்பதை ஒரு விளையாட்டாகக் கணக்கெடுப்போம் . அதற்கு என்ன செய்வோம் என்று நினைத்துப் பாருங்கள் . முதலில் ஒரே நிறத்தில் உள்ள மிட்டாய்களை எல்லாம் ஒன்றாகச் சேர்த்து அவற்றைத் தனித்தனிக் குழுக்களாகப் பிரித்து வைப்போம். பின்னர் ஒவ்வொரு நிறக் குழுவிலும் சென்று எத்தனை மிட்டாய் உள்ளது என்பதை சுலபமாகக் கணக்கெடுப்போம். இதையே தற்போது Mapreducer செய்கிறது. தகவல்களை

அடையாளம் கண்டு பிரித்து வைக்கும் வேலையை mapper-ம் அதனடிப்படையில் கணக்கெடுக்கும் வேலையை reducer-ம் செய்கிறது.

8 PIG

2006-ஆம் ஆண்டு Yahoo நிறுவனத்தின் ஒரு ஆய்வுத் திட்டமாக Pig என்பது உருவாக்கப்பட்டது. இது குறிப்பாக mapreduce வேலைகளைச் செய்வதற்காகப் பயன்படுத்தப்பட்டது. பின்னர் Apache நிறுவனம் 2008-ல் இதனை திறந்த மூல மென்பொருள் கருவியாக அறிவித்து வெளியிட்டது. Pig என்பது java, python போன்ற நிரலாக்க மொழிகளின் துணையில்லாமல், வெறும் SQL-ஐ வைத்து hadoop-ல் உள்ள தரவுகளை அணுக உதவும்

கருவி ஆகும். Hadoop பற்றிய அடிப்படை அறிவு பெற்றிருந்தால் போதும். இதனைச் சுலபமாகக் கற்றுக் கொள்ளலாம். தரவுத்தளமானது ஒரு database-ஆக இருக்கும்போது அதனை SQL வைத்துக் கையாளலாம். ஆனால் அதுவே கோப்பு வடிவில் (files) இருக்கும்போது அவற்றையும் கையாள்வதற்கு pig பயன்படுகிறது.

PigLatin எனும் ஒரு நிரலாக்க மொழியை இது வழங்குகின்றது. இதனை வைத்து நிரலாளர்கள் செய்யும் அனைத்து வேலைகளையும் நாம் செய்யலாம். பொதுவாக java தெரியாதவர்கள் hadoop-வுடன் வேலை செய்வதற்கு சற்று தடுமாறுவார்கள் (mapreduce போன்ற இடங்களில்). அவர்களின் சிரமத்தைக்

கலைவதற்காக வந்ததே PigLatin ஆகும். இதில் பயன்படுத்தப்படும் பெரும்பாலான operators, built-in functions, data types ஆகியவை SQL-ன் செயல்பாடுகளை ஒத்திருக்கும். இதை வைத்து எழுதப்படுவதே pig scripts ஆகும்.

இத்தகைய pig scripts-ஐ இயக்குவதற்கு நாம் grunt shell, user defined functions, embedded போன்ற ஏதாவதொன்றை நாம் தேர்ந்தெடுக்கலாம் (இப்பகுதியில் நாம் grunt shell-ஐப் பயன்படுத்தியுள்ளோம்). இயக்கத்தின் போது இவை தேவையான மாறுதல்களை அடைந்து கடைசியாக நமக்கு வேண்டிய வெளியீட்டினைக் கொடுக்கும். Pig-ன் கட்டமைப்பில் இத்தகைய இயக்கங்களை நிகழ்த்தி வெளியீட்டினைக் கொடுப்பதற்கு

உதவும் மிக முக்கிய அங்கங்கள் பற்றியும் அவற்றின் செயல்பாடுகள் பற்றியும் பின்வருமாறு காணலாம்.

i) Parser : ஆரம்பத்தில் pig scripts-ன் வடிவம், தரவு வகை மற்றும் இன்ன பிற விஷயங்களை சோதித்து (syntax, type & other checking), DAG (directed acyclic graph) எனும் வரைபடத்தை உருவாக்குகிறது. இவ்வரைபடம் nodes & edges-ஐக் கொண்டிருக்கும். nodes என்பது scripts-ல் பயன்படுத்தப்பட்டுள்ள logical operators-ஐயும், edges என்பது piglatin-வாக்கியங்கள் மூலம் நடைபெறும் data flow-ஐயும் குறிக்கும்.

ii) Optimizer : இது DAG வரைபடத்தைப் பெற்றுக்கொண்டு அதிலிருந்து projection/pushdown ஆகிய வேலைகளைச் செய்கிறது.
iii) Compiler : Optimize செய்யப்பட்ட திட்டத்தினை தொடர்ச்சியான mapreduce jobs-ஆக compile செய்கிறது.
iv) Execution Engine : Compiler உருவாக்கிய jobs-ஐ hadoop-ல் இயக்கி நமக்கு வேண்டிய வெளியீட்டினைக் கொடுக்கிறது.

8.1 Pig-ன் சிறப்பியல்புகள்

i) Pig Latin மொழியைக் கொண்டு நிரலாளர்கள் Java பற்றிய அறிவு இல்லாமலேயே சிறப்பாக map reduce வேலைகளைச் செய்ய முடியும்.

ii) Multi query approach முறையைப் பயன்படுத்தி அதிகப்படியான நிரல்கள் எழுதப்படுவதைத் தவிர்க்கலாம். அதாவது java-ல் எழுதப்படுகின்ற 200 வரிகள் இம்முறையில் எழுதப்படும் 10 வரிகளுக்குச் சமமாகும்.

iii) SQL கொஞ்சம் தெரிந்தால் போதும். இதனைச் சுலபமாகக் கற்றுக்கொள்ள முடியும்.

iv) MapReduce -ல் காணப்படாத tuples, bags, maps போன்ற சிறப்புத் தரவு வகைகள் இதில் காணப்படுகின்றன. இவை nested data types என்று அழைக்கப்படுகின்றன.

v) அனைத்து வகைத் தரவுகளையும் (வடிவான மற்றும் வடிவற்ற) இது கையாள்கிறது.

8.2 Pig-ல் பயன்படுத்தப்படும் முக்கியப் பதங்கள்

i) Atom : ஒவ்வொரு தனி மதிப்பும் atom என்று குறிக்கப்படும். இது எந்த தரவு வகையைச் சேர்ந்ததாகவும் இருக்கலாம். எடுத்துக்காட்டு: ‘Nithya’ or ‘30’

ii) Tuple : ஒரு ordered set-க்குள் உள்ள அனைத்து மதிப்புகளும் சேர்ந்து tuple என்று அழைக்கப்படும். இது table-ல் உள்ள row-ஐப் போன்றது. எடுத்துக்காட்டு: (Nithya,30)

iii) Bag : ஒரு unordered set-க்குள் உள்ள tuples அனைத்தும் சேர்ந்து Bag என்று

அழைக்கப்படும். இது table-ஐப் போன்றது. எடுத்துக்காட்டு: {(Nithya,30),(Shrinivasan,35)}

ஆனால் அதற்காக ஒவ்வொரு row-ம் ஒரே எண்ணிக்கையினாலான தரவுத் தொகுப்பினைக் கொண்டிருக்க வேண்டுமென்றோ, அவை ஒரே வகைத் தரவாக இருக்க வேண்டும் என்றோ அவசியமில்லை. எடுத்துக்காட்டு: {(Nithya,30),(35,Shrinivasan,shrinivasan@gmail.com)}

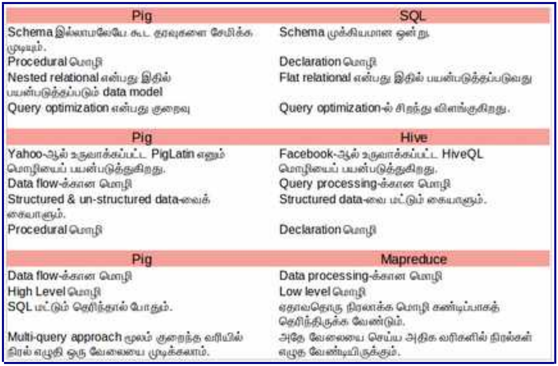
iv) Inner bag : ஒரு bag-க்குள் மற்றொரு bag காணப்படின் அது inner bag என்று அழைக்கப்படும். எடுத்துக்காட்டு: {Nithya,30,{35,Shrinivasan,shrinivasan@gmail.com}}

v) Map : அடைப்புக்குறிக்குள் key மற்றும் value இணைகளாக காணப்படுபவை maps என்று அழைக்கப்படும். key என்பது chararray தரவு வகையைச் சேர்ந்ததாகவும், value என்பது எந்த வகைத் தரவாக வேண்டுமானாலும் இருக்கலாம். எடுத்துக்காட்டு: [Name#Nithya,age#30]

vi) Relations : ஒன்றுக்கும் மேற்பட்ட tuples-ஐ உள்ளடக்கிய bags-தான் relations என்றழைக்கப்படும். இதில் உள்ள தரவுகள் எந்த வரிசையில் வேண்டுமானாலும் process செய்யப்படலாம். வரிசைப்படுத்தப்பட்ட அதே முறையில் தான் process செய்யப்படும் என்று கிடையாது.

8.3 மற்றவைகளோடு Pig-ன் ஒப்பீடு

Pig-ஆனது SQL, Hive போன்றவற்றுடன் எந்தெந்த விதங்களில் வேறுபடுகிறது, மேலும் மற்ற நிரலாக்க மொழிகளை வைத்து எழுதப்படுகின்ற mapreduce-க்கும் pig-க்கும் என்னென்ன வேறுபாடு ஆகியவற்றைப் பின்வருமாறு காணலாம்.



Pig	SQL
Schema இல்லாமலேயே கூட தரவுகளை சேமிக்க முடியும்.	Schema முக்கியமான ஒன்று.
Procedural மொழி	Declaration மொழி
Nested relational என்பது இதில் பயன்படுத்தப்படும் data model	Flat relational என்பது இதில் பயன்படுத்தப்படுவது
Query optimization என்பது குறைவு	Query optimization-ல் சிறந்து விளங்குகிறது.

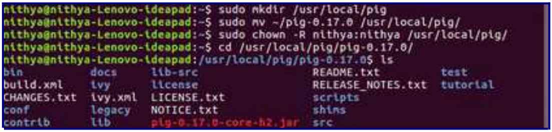
Pig	Hive
Yahoo-ஆல் உருவாக்கப்பட்ட PigLatin எனும் மொழியைப் பயன்படுத்துகிறது.	Facebook-ஆல் உருவாக்கப்பட்ட HiveQL மொழியைப் பயன்படுத்துகிறது.
Data flow-க்கான மொழி	Query processing-க்கான மொழி
Structured & un-structured data-வைக் கையாளும்.	Structured data-வை மட்டும் கையாளும்.
Procedural மொழி	Declaration மொழி

Pig	Mapreduce
Data flow-க்கான மொழி	Data processing-க்கான மொழி
High Level மொழி	Low level மொழி
SQL மட்டும் தெரிந்தால் போதும்.	ஏதாவதொரு நிரலாக்க மொழி கண்டிப்பாகத் தெரிந்திருக்க வேண்டும்.
Multi-query approach மூலம் குறைந்த வரியில் நிரல் எழுதி ஒரு வேலையை முடிக்கலாம்.	அதே வேலையை செய்ய அதிக வரிகளில் நிரல்கள் எழுத வேண்டியிருக்கும்.

1. Pig-ஐ பதிவிறக்கம் செய்து, பிரித்து, நமக்கு வேண்டிய இடத்திற்கு மாற்றி, மாற்றப்பட்ட directory-க்கான முழு அனுமதிகளையும் பயனருக்கு வழங்கவும்.

```
$ wget
http://redrockdigimark.com/apache
mirror/pig/pig-0.17.0/pig-
0.17.0.tar.gz
$ tar -xzvf pig-0.17.0.tar.gz
$ sudo mkdir /usr/local/pig
$ sudo mv ~/pig-0.17.0
/usr/local/pig/
$ sudo chown -R nithya:nithya
/usr/local/pig/
```

```
nithya@nithya-Lenovo-ideapad:~$ wget http://redrockdigimark.com/apachemirror/pig
/pig-0.17.0/pig-0.17.0.tar.gz
--2018-03-19 22:07:42--  http://redrockdigimark.com/apachemirror/pig/pig-0.17.0/
pig-0.17.0.tar.gz
Resolving redrockdigimark.com (redrockdigimark.com)... 119.18.61.94
Connecting to redrockdigimark.com (redrockdigimark.com)|119.18.61.94|:80... conn
ected.
HTTP request sent, awaiting response... 200 OK
Length: 230606579 (220M) [application/x-gzip]
Saving to: ‘pig-0.17.0.tar.gz’

pig-0.17.0.tar.gz   100%[======================>] 219.92M   483KB/s    in 5m 17s

2018-03-19 22:12:59 (711 KB/s) - ‘pig-0.17.0.tar.gz’ saved [230606579/230606579]
```



```
nithya@nithya-Lenovo-ideapad:~$ sudo mkdir /usr/local/pig
nithya@nithya-Lenovo-ideapad:~$ sudo mv ~/pig-0.17.0 /usr/local/pig/
nithya@nithya-Lenovo-ideapad:~$ sudo chown -R nithya:nithya /usr/local/pig/
nithya@nithya-Lenovo-ideapad:~$ cd /usr/local/pig/pig-0.17.0/
nithya@nithya-Lenovo-ideapad:/usr/local/pig/pig-0.17.0$ ls
bin          docs     lib-src                 README.txt         test
build.xml    ivy      license                 RELEASE_NOTES.txt  tutorial
CHANGES.txt  ivy.xml  LICENSE.txt             scripts
conf         legacy   NOTICE.txt              shims
contrib      lib      pig-0.17.0-core-h2.jar  src
```

2. நமது home directory-ல் உள்ள .bashrc எனும் கோப்பிற்குள் பின்வரும் விவரங்களை கடைசியில் இணைக்கவும்.

```
$ vim ~/.bashrc
export PIG_HOME=/usr/local/pig/pig-0.17.0/
export PATH=$PATH:/usr/local/pig/pig-0.17.0/bin
export PIG_CLASSPATH=$PIG_HOME/conf
$ source ~/.bashrc
```

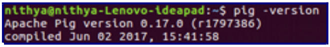
```
export PIG_HOME=/usr/local/pig/pig-0.17.0/
export PATH=$PATH:/usr/local/pig/pig-0.17.0/bin
export PIG_CLASSPATH=$PIG_HOME/conf
"~/.bashrc" 134L, 4413C                                   134,35        Bot
```

.

```
nithya@nithya-Lenovo-ideapad:~$ vim ~/.bashrc
nithya@nithya-Lenovo-ideapad:~$ source ~/.bashrc
```

3. pig நிறுவப்பட்டு விட்டதா என்பதை சோதிக்க பின்வரும் கட்டளையை அளித்துப் பார்க்கவும்.

```
$ pig -version
```



```
nithya@nithya-Lenovo-ideapad:~$ pig -version
Apache Pig version 0.17.0 (r1797386)
compiled Jun 02 2017, 15:41:58
```

4. Pig-ஐ local mode (நமது கணினியில் உள்ள கோப்புகளை அணுகும்) மற்றும் mapreduce mode (hdfs-ல் உள்ள கோப்புகளை அணுகும்) எனும் இரண்டு விதங்களில் இயக்கலாம். இயல்பாக $ pig

எனக் கொடுத்தால் இது mapreduce mode-ல் தான் செயல்படத் தொடங்கும். local-க்கு மாற்றுவதற்கு பின்வருமாறு கொடுக்கவும்.

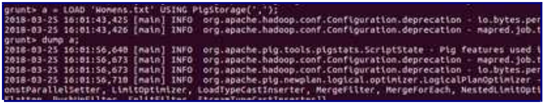
```
$ pig -x local
```

இதன் வெளியீடாக grunt> shell உருவாகும். இதுவே Interactive mode-ஆக செயல்படும். இங்கு நாம் PigLatin வாக்கியங்களைக் கொடுத்து அதன்

வெளியீட்டினை சோதித்துப் பார்க்கலாம் (dump operator மூலம்).

கீழ்க்கண்டவற்றில் = க்கு வலப்புறம் உள்ளது PigLatin வாக்கியம் ஆகும். இடப்புறம் உள்ளது relation என்றழைக்கப்படும் . இது LOAD எனும் operator மூலம் womens.txt எனும் கோப்பிற்குள் உள்ளவற்றை a எனும் relation-ல் கொண்டு சேர்க்கிறது. பின்னர் dump operator மூலம் relation-ல் ஏற்றப்பட்டுள்ள விவரங்களை வெளிப்படுத்தியுள்ளது .

```
grunt> a = LOAD 'Womens.txt'
USING PigStorage(',');
grunt> dump a;
```



```
grunt> a = LOAD 'Womens.txt' USING PigStorage(',');
2018-03-25 16:01:43,425 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per
2018-03-25 16:01:43,426 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.t
grunt> dump a;
2018-03-25 16:01:56,640 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used
2018-03-25 16:01:56,673 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.t
2018-03-25 16:01:56,673 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per
2018-03-25 16:01:56,710 [main] INFO  org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer -
onstParallelSetter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, MergeForEach, NestedLimitOpti
```

ஒவ்வொரு PigLatin வாக்கியமும் பின்வரும் வரையறைகளைப் பெற்று விளங்கும் .

- இறுதியில் ; (semi-colon) காணப்படும்.
- relation-வுடன் எப்போதும் சேர்ந்தே இருக்கும். இதுவே expression மற்றும் schema-வை உள்ளடக்கி இருக்கும்.
- load மற்றும் store தவிர்த்து, இன்ன பிற operators அனைத்தும் ஒரு relation-ஐயே உள்ளீடாக எடுத்துக்கொண்டு புதியதொரு

relation-ஐ உருவாக்கக் கூடியவை. (Eg: join, cogroup).

- இதில் காணப்படும் பல்வேறு வகையான operators-தான் நாம் விரும்பும் அனைத்து வேலைகளையும் செய்ய உதவும்.

அடுத்து PigLatin வாக்கியங்களை உள்ளடக்கி smp.pig எனும் script file-ஐ உருவாக்கியுள்ளோம். இதனை இயக்குவதற்கு -x என்பதைப் பயன்படுத்தலாம்.

```
$ cat smp.pig
$ pig -x local smp.pig
```

```
nithya@nithya-Lenovo-ideapad:~$ cat smp.pig
a = LOAD 'Womens.txt' USING
   PigStorage(',') as (id:int,fname:chararray,lname:chararray,mobile:int,city:chararray);
Dump a;
nithya@nithya-Lenovo-ideapad:~$ pig -x local smp.pig
2018-03-25 15:29:17,393 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using bu
2018-03-25 15:29:17,464 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
2018-03-25 15:29:17,465 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
2018-03-25 15:29:17,490 [main] INFO  org.apache.pig.Main - Apache Pig version 0.17.0 (r1797386) compiled Jun 02 2017,
2018-03-25 15:29:17,490 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/nithya/pig_1521971957489.l
```

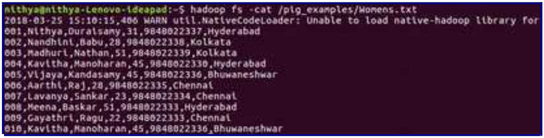
5. Pig-ஐ mapreduce mode-ல் இயக்குவதற்கு பின்வரும் இரண்டு கட்டளைகளில் ஒன்றைத் தேர்ந்தெடுக்கலாம். இரண்டும் ஒன்றே ஆகும்.

```
$ pig
$ Pig –x mapreduce
```

இதன் வெளியீடாக grunt> உருவாவதைக் காணலாம். இதில் ஒருசில PigLatin வாக்கியங்களைக் கொடுத்து சோதிப்பதற்கு முன்னர், hdfs-ன் / -க்குள் நமக்கு வேண்டிய கோப்புகளைச் சேர்த்து விடவும். ஏனெனில் இது hdfs-ல் சென்றுதான் கோப்புகளைத் தேட ஆரம்பிக்கும் .

பின்வரும் கட்டளை மூலம் hdfs-ன் /pig_examples எனுமிடத்தில் Womens.txt எனும் கோப்பு உள்ளதை உறுதி செய்து கொள்ளலாம் .

```
$ hadoop fs -cat
/pig_examples/Womens.txt
```



```
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -cat /pig_examples/Womens.txt
2018-03-25 15:10:15,406 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
001,Nithya,Duraisamy,31,9848022337,Hyderabad
002,Nandhini,Babu,28,9848022338,Kolkata
003,Madhuri,Nathan,51,9848022339,Kolkata
004,Kavitha,Manoharan,45,9848022330,Hyderabad
005,Vijaya,Kandasamy,45,9848022336,Bhuwaneshwar
006,Aarthi,Raj,28,9848022335,Chennai
007,Lavanya,Sankar,23,9848022334,Chennai
008,Meena,Baskar,51,9848022333,Hyderabad
009,Gayathri,Ragu,22,9848022333,Chennai
010,Kavitha,Manoharan,45,9848022336,Bhuwaneshwar
```

பின்னர் grunt> -க்குள் சென்று PigLatin வாக்கியங்களைக் கொடுக்கவும்.

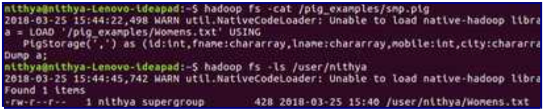
```
grunt> a = LOAD
'hdfs://localhost:54310/pig_examp
les/Womens.txt' USING
PigStorage(',');
grunt> dump a;
```

```
grunt> a = LOAD '/pig_examples/Womens.txt' USING PigStorage(',');
grunt> dump a;
2018-03-25 15:13:10,983 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: UNKNOWN
2018-03-25 15:13:11,017 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not gen
2018-03-25 15:13:11,066 [main] INFO  org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES_ENABLED=[AddForEach
onstParallelSetter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, MergeForEach, NestedLimitOptimizer, PartitionFilterOptim
Flatten, PushUpFilter, SplitFilter, StreamTypeCastInserter]}
2018-03-25 15:13:11,117 [main] INFO  org.apache.pig.impl.util.SpillableMemoryManager - Selected heap (PS Old Gen) of size 69940
0128, usageThreshold = 489100128
2018-03-25 15:13:11,161 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenati
2018-03-25 15:13:11,188 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan
2018-03-25 15:13:11,188 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan
```

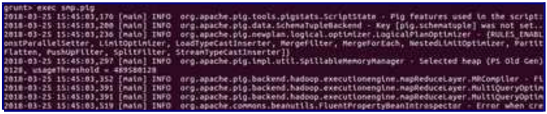
அடுத்ததாக மேலே செய்து பார்த்தது போன்றே, smp.pig -க்குள் ஒருசில PigLatin வாக்கியங்களை அளித்து அதனை hdfs-ன்

/pig_examples எனுமிடத்தில் சேர்க்கவும். இதனை exec அல்லது run ஆகிய கட்டளைகளைப் பயன்படுத்தி இயக்கலாம் .

```
grunt> exec smp.pig (or) grunt>
run smp.pig
```



```
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -cat /pig_examples/smp.pig
2018-03-25 15:44:22,498 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
a = LOAD '/pig_examples/Womens.txt' USING
   PigStorage(',') as (id:int,fname:chararray,lname:chararray,mobile:int,city:chararra
Dump a;
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -ls /user/nithya
2018-03-25 15:44:45,742 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
Found 1 items
-rw-r--r--   1 nithya supergroup        428 2018-03-25 15:40 /user/nithya/Womens.txt
```



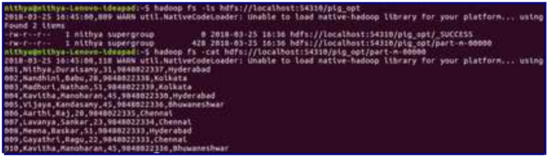
```
grunt> exec smp.pig
2018-03-25 15:45:03,170 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script:
2018-03-25 15:45:03,200 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set..
2018-03-25 15:45:03,236 [main] INFO  org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES_ENABL
onstParallelSetter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, MergeForEach, NestedLimitOptimizer, Partit
Flatten, PushUpFilter, SplitFilter, StreamTypeCastInserter]}
2018-03-25 15:45:03,297 [main] INFO  org.apache.pig.impl.util.SpillableMemoryManager - Selected heap (PS Old Gen)
0128, usageThreshold = 489580128
2018-03-25 15:45:03,352 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - Fi
2018-03-25 15:45:03,391 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptim
2018-03-25 15:45:03,391 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptim
2018-03-25 15:45:03,519 [main] INFO  org.apache.commons.beanutils.FluentPropertyBeanIntrospector - Error when cre
```

6. STORE எனும் operator ஒரு relation-ல் சேமிக்கப்பட்டுள்ள விவரங்களை hdfs-ன் ஏதேனும் ஒரு இடத்தில் சேமிக்கப் பயன்படுகிறது. எடுத்துக்காட்டாக a எனும் relation-ல் சேமிக்கப்பட்டுள்ள விவரங்களை pig_opt எனுமிடத்தில் சேமிப்பதற்கான கட்டளை பின்வருமாறு அமையும்.

```
grunt> STORE a INTO
'hdfs://localhost:54310/pig_opt'
USING PigStorage(',');
```

pig_opt1 எனுமிடத்தில் சென்று பார்த்தால் இரண்டு கோப்புகள் உருவாகியிருப்பதைக் காணலாம். அதில் part-m-ooooo எனும் கோப்பில் தான் a -ல் சேமிக்கப்பட்டுள்ள விவரங்கள் காணப்படும்.

```
$ hadoop fs -cat
hdfs://localhost:54310/pig_opt/pa
rt-m-ooooo
```



```
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -ls hdfs://localhost:54310/pig_opt
2018-03-25 16:45:00,009 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using
Found 2 items
-rw-r--r--   1 nithya supergroup          0 2018-03-25 16:36 hdfs://localhost:54310/pig_opt/_SUCCESS
-rw-r--r--   1 nithya supergroup        428 2018-03-25 16:36 hdfs://localhost:54310/pig_opt/part-m-00000
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -cat hdfs://localhost:54310/pig_opt/part-m-00000
2018-03-25 16:45:08,118 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using
001,Nithya,Duraisamy,31,9848022337,Hyderabad
002,Nandhini,Babu,28,9848022338,Kolkata
003,Madhuri,Nathan,51,9848022339,Kolkata
004,Kavitha,Manoharan,45,9848022330,Hyderabad
005,Vijaya,Kandasamy,45,9848022336,Bhuwaneshwar
006,Aarthi,Raj,28,9848022335,Chennai
007,Lavanya,Sankar,23,9848022334,Chennai
008,Meena,Baskar,51,9848022333,Hyderabad
009,Gayathri,Ragu,22,9848022333,Chennai
010,Kavitha,Manoharan,45,9848022336,Bhuwaneshwar
```

7. Pig-ல் கீழ்க்கண்ட 4 விதமான operators உள்ளன. இவையே Illustration operators எனப்படும்.

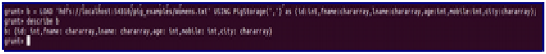
Dump : ஒரு relation-ல் உள்ளவற்றை திரையில் வெளிப்படுத்தும்.
Describe : அதன் schema-வை வெளிப்படுத்தும்.
Explain : அதன் பல்வேறு இயக்கத்

திட்டங்களை வெளிப்படுத்தும்.
Illustrate : படிப்படியான இயக்க நிகழ்வுகளை வெளிப்படுத்தும்.

இதில் dump-க்கான எடுத்துக்காட்டினை ஏற்கனவே பார்த்துள்ளோம் . மற்ற மூன்றுக்குமான எடுத்துக்காட்டு பின்வருமாறு.

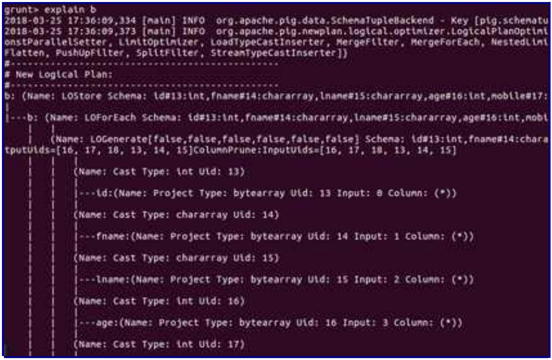
```
grunt> b = LOAD 'hdfs://localhost:54310/pig_examples/Womens.txt' USING PigStorage(',') as (id:int,fname:chararray,lname:chararray,age:int,mobile:int,city:chararray);
```

```
grunt> describe b
```



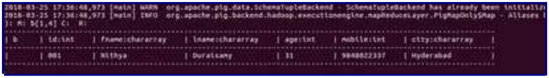
```
grunt> b = LOAD 'hdfs://localhost:54310/pig_examples/mobens.txt' USING PigStorage(',') as (id:int,fname:chararray,lname:chararray,age:int,mobile:int,city:chararray);
grunt> describe b
b: {id: int,fname: chararray,lname: chararray,age: int,mobile: int,city: chararray}
grunt>
```

```
grunt> explain b
```



```
grunt> explain b
2018-03-25 17:36:09,334 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematu
2018-03-25 17:36:09,373 [main] INFO  org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimi
onstParallelSetter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, MergeForEach, NestedLimi
Flatten, PushUpFilter, SplitFilter, StreamTypeCastInserter])
#-----------------------------------------------
# New Logical Plan:
#-----------------------------------------------
b: (Name: LOStore Schema: id#13:int,fname#14:chararray,lname#15:chararray,age#16:int,mobile#17:
|
|---b: (Name: LOForEach Schema: id#13:int,fname#14:chararray,lname#15:chararray,age#16:int,mobi
    |   |
    |   (Name: LOGenerate[false,false,false,false,false,false] Schema: id#13:int,fname#14:chara
tputUids=[16, 17, 18, 13, 14, 15]ColumnPrune:InputUids=[16, 17, 18, 13, 14, 15]
    |   |   |
    |   |   (Name: Cast Type: int Uid: 13)
    |   |   |
    |   |   |---id:(Name: Project Type: bytearray Uid: 13 Input: 0 Column: (*))
    |   |   |
    |   |   (Name: Cast Type: chararray Uid: 14)
    |   |   |
    |   |   |---fname:(Name: Project Type: bytearray Uid: 14 Input: 1 Column: (*))
    |   |   |
    |   |   (Name: Cast Type: chararray Uid: 15)
    |   |   |
    |   |   |---lname:(Name: Project Type: bytearray Uid: 15 Input: 2 Column: (*))
    |   |   |
    |   |   (Name: Cast Type: int Uid: 16)
    |   |   |
    |   |   |---age:(Name: Project Type: bytearray Uid: 16 Input: 3 Column: (*))
    |   |   |
    |   |   (Name: Cast Type: int Uid: 17)
    |   |   |
```

```
grunt> illustrate b
```



```
2018-03-25 17:38:48,973 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initializ
2018-03-25 17:38:48,973 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.PigMapOnly$Map - Aliases 
): M: b[1,4] C:  R:
----------------------------------------------------------------------------------------------------------------
| b      | id:int     | fname:chararray     | lname:chararray     | age:int     | mobile:int     | city:chararray     |
----------------------------------------------------------------------------------------------------------------
|        | 001        | Nithya              | Duraisamy           | 31          | 9848022337     | Hyderabad          |
----------------------------------------------------------------------------------------------------------------
```

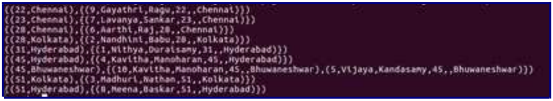
8. GROUP எனும் operator ஒரு குறிப்பிட்ட மதிப்பின் அடிப்படையில், relation-ல் உள்ள தரவுகளை குழுக்களாக இணைத்து வெளிப்படுத்தும். பின்வரும் எடுத்துக்காட்டில் 28, 45 வயதில் உள்ள ஒன்றுக்கும் மேற்பட்ட நபர்கள் ஒரே குழுக்களாக இணைந்து வெளிப்படுத்தப் பட்டிருப்பதைக் காணலாம்.

```
grunt> i = GROUP b by age;
grunt> dump i;
```

```
(22,{(9,Gayathri,Ragu,22,,Chennai)})
(23,{(7,Lavanya,Sankar,23,,Chennai)})
(28,{(6,Aarthi,Raj,28,,Chennai),(2,Nandhini,Babu,28,,Kolkata)})
(31,{(1,Nithya,Duraisamy,31,,Hyderabad)})
(45,{(10,Kavitha,Manoharan,45,,Bhuwaneshwar),(5,Vijaya,Kandasamy,45,,Bhuwaneshwar),(4,Kavitha,Manoharan,45,,Hyderabad)})
(55,{(8,Meena,Baskar,55,,Hyderabad),(3,Madhuri,Nathan,55,,Kolkata)})
```

அவ்வாறே கீழ்க்கண்ட எடுத்துக்காட்டில் (age, city) எனக் கொடுத்திருப்பதால், ஒரே வயதில் இருந்தாலும், அவர்கள் ஒரே ஊரைச் சேர்ந்தவர்களாக இருந்தால் மட்டுமே குழுக்களாக இணைக்கப்பட்டிருப்பதைக் காணலாம்.

```
grunt> j = GROUP b by (age,
city);
grunt> dump j;
```

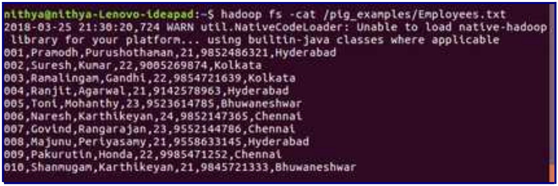


```
((22,Chennai),{(9,Gayathri,Ragu,22,,Chennai)})
((23,Chennai),{(7,Lavanya,Sankar,23,,Chennai)})
((28,Chennai),{(6,Aarthi,Raj,28,,Chennai)})
((28,Kolkata),{(2,Nandhini,Babu,28,,Kolkata)})
((31,Hyderabad),{(1,Nithya,Duraisamy,31,,Hyderabad)})
((45,Hyderabad),{(4,Kavitha,Manoharan,45,,Hyderabad)})
((45,Bhuwaneshwar),{(10,Kavitha,Manoharan,45,,Bhuwaneshwar),(5,Vijaya,Kandasamy,45,,Bhuwaneshwar)})
((51,Kolkata),{(3,Madhuri,Nathan,51,,Kolkata)})
((51,Hyderabad),{(8,Meena,Baskar,51,,Hyderabad)})
```

மேலும் All எனக் கொடுத்தால் அனைத்தும் ஒரே குழுவாக இணைக்கப்படும் .

```
grunt> k = GROUP b All;
grunt> dump k;
```

```
(all,{(10,Kavitha,Manoharan,45,,Bhuwaneshwar),(9,Gayathri,Ragu,22,,Chennai),(8,Meena,Baskar,51,,Hyderabad),(7,Lavanya,Sankar,23
,,Chennai),(6,Aarthi,Raj,28,,Chennai),(5,Vijaya,Kandasamy,45,,Bhuwaneshwar),(4,Kavitha,Manoharan,45,,Hyderabad),(3,Madhuri,Nath
an,51,,Kolkata),(2,Nandhini,Babu,28,,Kolkata),(1,Nithya,Duraisamy,31,,Hyderabad)})
```



```
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -cat /pig_examples/Employees.txt
2018-03-25 21:30:20,724 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
001,Pramodh,Purushothaman,21,9852486321,Hyderabad
002,Suresh,Kumar,22,9005269874,Kolkata
003,Ramalingam,Gandhi,22,9854721639,Kolkata
004,Ranjit,Agarwal,21,9142578963,Hyderabad
005,Toni,Mohanthy,23,9523614785,Bhuwaneshwar
006,Naresh,Karthikeyan,24,9852147365,Chennai
007,Govind,Rangarajan,23,9552144786,Chennai
008,Majunu,Periyasamy,21,9558633145,Hyderabad
009,Pakurutin,Honda,22,9985471252,Chennai
010,Shanmugam,Karthikeyan,21,9845721333,Bhuwaneshwar
```

9. COGROUP எனும் operator ஒன்றுக்கும் மேற்பட்ட relations-ஐ எடுத்துக்கொண்டு அதனை கொடுக்கப்பட்ட மதிப்புகளுக்கு ஏற்றவாறு குழுக்களாக இணைத்து வெளிப்படுத்தும் .

```
grunt> c = LOAD
'hdfs://localhost:54310/pig_examp
```

```
les/Employees.txt' USING
PigStorage(',') as
(id:int,fname:chararray,lname:cha
rarray,age:int,mobile:int,city:ch
ararray);


grunt> d = COGROUP b by age, c by
age;
grunt> dump d;
```

```
(21,{},{(10,Shanmugam,Karthikeyan,21,,Bhuwaneshwar),(8,Rajunu,Periyasamy,21,,Hyderabad),(4,Ranjit,Agarwal,21,,Hyderabad),(1,Pramodh,
(22,{(9,Gayathri,Ragu,22,,Chennai)},{(9,Pakurutla,Honda,22,,Chennai),(3,Ramalingam,Gandhi,22,,Kolkata),(2,Suresh,Kumar,22,,Kolkata)}
(23,{(7,Lavanya,Sankar,23,,Chennai)},{(7,Govind,Rangarajan,23,,Chennai),(5,Toni,Mohanthy,23,,Bhuwaneshwar)})
(24,{},{(6,Naresh,Karthikeyan,24,,Chennai)})
(28,{(6,Aarthi,Raj,28,,Chennai),(2,Nandhini,Babu,28,,Kolkata)},{})
(31,{(1,Nithya,Duraisamy,31,,Hyderabad)},{})
(45,{(10,Kavitha,Manoharan,45,,Bhuwaneshwar),(3,Vijaya,Kandasamy,45,,Bhuwaneshwar),(4,Kavitha,Manoharan,45,,Hyderabad)},{})
(51,{(8,Meena,Baskar,51,,Hyderabad),(3,Madhuri,Nathan,51,,Kolkata)},{})
(,{},{(,,,,,)})
```

10. JOIN எனும் operator ஒன்றுக்கும் மேற்பட்ட relations-ஐ இணைத்து, கொடுக்கப்பட்ட condition-ல் பொருந்தும் தகவல்களை மட்டும் எடுத்து வெளிப்படுத்தும் Inner Join போன்று செயல்படும்.

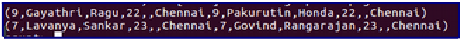
கீழ்க்கண்டவற்றில் age-ஐ வைத்துப் பொருத்தி இருவேறு relation-ல் இருக்கும் தகவல்களை வெளிப்படுத்தியுள்ளது .

```
grunt> e = JOIN b by age, c by
age;
grunt> dump e;
```

```
(9,Gayathri,Ragu,22,,Chennai,9,Pakurutin,Honda,22,,Chennai)
(9,Gayathri,Ragu,22,,Chennai,3,Ramalingam,Gandhi,22,,Kolkata)
(9,Gayathri,Ragu,22,,Chennai,2,Suresh,Kumar,22,,Kolkata)
(7,Lavanya,Sankar,23,,Chennai,7,Govind,Rangarajan,23,,Chennai)
(7,Lavanya,Sankar,23,,Chennai,5,Toni,Mohanthy,23,,Bhuwaneshwar)
```

அடுத்து age,city-ஐ வைத்துப் பொருத்தி தகவல்களை வெளிப்படுத்தியுள்ளது .

```
grunt> i = JOIN b BY (age,city),
c BY (age,city);
grunt> dump i;
```



```
(9,Gayathri,Ragu,22,,Chennai,9,Pakurutin,Honda,22,,Chennai)
(7,Lavanya,Sankar,23,,Chennai,7,Govind,Rangarajan,23,,Chennai)
```

பின்வருவது left outer join போன்று செயல்படும். அதாவது இடப்புற relation-ல் இருக்கும் அனைத்து தகவல்களையும் வெளிப்படுத்தி, வலப்புற relation-ல் இருந்து condition-ல் பொருந்தும் தகவல்களை மட்டும் எடுத்து வெளிப்படுத்தும். இல்லையெனில் Null மதிப்பினை வெளிப்படுத்தும் .

```
grunt> f = JOIN b by age LEFT
OUTER, c by age;
grunt> dump f;
```

```
(9,Gayathri,Ragu,22,,Chennai,9,Pakurutin,Honda,22,,Chennai)
(9,Gayathri,Ragu,22,,Chennai,3,Ramalingam,Gandhi,22,,Kolkata)
(9,Gayathri,Ragu,22,,Chennai,2,Suresh,Kumar,22,,Kolkata)
(7,Lavanya,Sankar,23,,Chennai,7,Govind,Rangarajan,23,,Chennai)
(7,Lavanya,Sankar,23,,Chennai,5,Toni,Mohanthy,23,,Bhuwaneshwar)
(6,Aarthi,Raj,28,,Chennai,,,,,,)
(2,Nandhini,Babu,28,,Kolkata,,,,,,)
(1,Nithya,Duraisamy,31,,Hyderabad,,,,,,)
(10,Kavitha,Manoharan,45,,Bhuwaneshwar,,,,,,)
(5,Vijaya,Kandasamy,45,,Bhuwaneshwar,,,,,,)
(4,Kavitha,Manoharan,45,,Hyderabad,,,,,,)
(8,Meena,Baskar,51,,Hyderabad,,,,,,)
(3,Madhuri,Nathan,51,,Kolkata,,,,,,)
```

பின்வருவது right outer join ஆகும். இது வலப்புறத்தில் இருக்கும் அனைத்துத் தகவல்களையும் வெளிப்படுத்தி, இடப்புறத்தில் இருந்து பொருந்தும் தகவல்களையும், இல்லையெனில் Null மதிப்பினையும் வெளிப்படுத்தும் .

```
grunt> g = JOIN b by age RIGHT, c
by age;
grunt> dump g;
```

```
(,,,,,,10,Shanmugam,Karthikeyan,21,,Bhuwaneshwar)
(,,,,,,8,Majunu,Periyasamy,21,,Hyderabad)
(,,,,,,4,Ranjit,Agarwal,21,,Hyderabad)
(,,,,,,1,Pramodh,Purushothaman,21,,Hyderabad)
(9,Gayathri,Ragu,22,,Chennai,9,Pakurutin,Honda,22,,Chennai)
(9,Gayathri,Ragu,22,,Chennai,3,Ramalingam,Gandhi,22,,Kolkata)
(9,Gayathri,Ragu,22,,Chennai,2,Suresh,Kumar,22,,Kolkata)
(7,Lavanya,Sankar,23,,Chennai,7,Govind,Rangarajan,23,,Chennai)
(7,Lavanya,Sankar,23,,Chennai,5,Toni,Mohanthy,23,,Bhuwaneshwar)
(,,,,,,6,Naresh,Karthikeyan,24,,Chennai)
(,,,,,,,,,,,)
```

பின்வருவது full outer join ஆகும்.

```
grunt> h = JOIN b by age FULL, c
by age;
grunt> dump h;
```

```
(,,,,,,10,Shanmugam,Karthikeyan,21,,Bhuwaneshwar)
(,,,,,,8,Majunu,Periyasamy,21,,Hyderabad)
(,,,,,,4,Ranjit,Agarwal,21,,Hyderabad)
(,,,,,,1,Pramodh,Purushothaman,21,,Hyderabad)
(9,Gayathri,Ragu,22,,Chennai,9,Pakurutin,Honda,22,,Chennai)
(9,Gayathri,Ragu,22,,Chennai,3,Ramalingam,Gandhi,22,,Kolkata)
(9,Gayathri,Ragu,22,,Chennai,2,Suresh,Kumar,22,,Kolkata)
(7,Lavanya,Sankar,23,,Chennai,7,Govind,Rangarajan,23,,Chennai)
(7,Lavanya,Sankar,23,,Chennai,5,Toni,Mohanthy,23,,Bhuwaneshwar)
(,,,,,,6,Naresh,Karthikeyan,24,,Chennai)
(6,Aarthi,Raj,28,,Chennai,,,,,,)
(2,Nandhini,Babu,28,,Kolkata,,,,,,)
(1,Nithya,Duraisamy,31,,Hyderabad,,,,,,)
(10,Kavitha,Manoharan,45,,Bhuwaneshwar,,,,,,)
(5,Vijaya,Kandasamy,45,,Bhuwaneshwar,,,,,,)
(4,Kavitha,Manoharan,45,,Hyderabad,,,,,,)
(8,Meena,Baskar,51,,Hyderabad,,,,,,)
(3,Madhuri,Nathan,51,,Kolkata,,,,,,)
(,,,,,,,,,,,)
```

11. CROSS என்பது Cartesian Join போன்று செயல்படும். அதாவது இடப்புற relation-ல் இருக்கும் முதல் record, வலப்புறத்தில் உள்ள அனைத்து records-வுடனும் இணைத்து வெளிப்படுத்தப்படும். பின்னர், இதே முறையில் அனைத்து records-ம் வெளிப்படும் .

```
grunt> j = CROSS b, c;
grunt> dump j;
```

```
(10,Kavitha,Manoharan,45,,Bhuwaneshwar,,,,,,)
(10,Kavitha,Manoharan,45,,Bhuwaneshwar,10,Shanmugam,Karthikeyan,21,,Bhuwaneshwar)
(10,Kavitha,Manoharan,45,,Bhuwaneshwar,9,Pakurutin,Honda,22,,Chennai)
(10,Kavitha,Manoharan,45,,Bhuwaneshwar,8,Majunu,Periyasamy,21,,Hyderabad)
(10,Kavitha,Manoharan,45,,Bhuwaneshwar,7,Govind,Rangarajan,23,,Chennai)
(10,Kavitha,Manoharan,45,,Bhuwaneshwar,6,Naresh,Karthikeyan,24,,Chennai)
(10,Kavitha,Manoharan,45,,Bhuwaneshwar,5,Toni,Mohanthy,23,,Bhuwaneshwar)
(10,Kavitha,Manoharan,45,,Bhuwaneshwar,4,Ranjit,Agarwal,21,,Hyderabad)
(10,Kavitha,Manoharan,45,,Bhuwaneshwar,3,Ramalingam,Gandhi,22,,Kolkata)
(10,Kavitha,Manoharan,45,,Bhuwaneshwar,2,Suresh,Kumar,22,,Kolkata)
(10,Kavitha,Manoharan,45,,Bhuwaneshwar,1,Pramodh,Purushothaman,21,,Hyderabad)
(9,Gayathri,Ragu,22,,Chennai,,,,,,)
(9,Gayathri,Ragu,22,,Chennai,10,Shanmugam,Karthikeyan,21,,Bhuwaneshwar)
(9,Gayathri,Ragu,22,,Chennai,9,Pakurutin,Honda,22,,Chennai)
(9,Gayathri,Ragu,22,,Chennai,8,Majunu,Periyasamy,21,,Hyderabad)
(9,Gayathri,Ragu,22,,Chennai,7,Govind,Rangarajan,23,,Chennai)
(9,Gayathri,Ragu,22,,Chennai,6,Naresh,Karthikeyan,24,,Chennai)
```

12. UNION எனும் operator இருவேறு relation-ல் இருக்கும் தரவுகளை இணைத்து வெளிப்படுத்தும் .

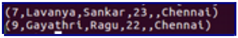
```
grunt> k = UNION b, c;
grunt> dump k;
```

```
(1,Pramodh,Purushothaman,21,,Hyderabad)
(2,Suresh,Kumar,22,,Kolkata)
(3,Ramalingam,Gandhi,22,,Kolkata)
(4,Ranjit,Agarwal,21,,Hyderabad)
(5,Toni,Mohanthy,23,,Bhuwaneshwar)
(6,Naresh,Karthikeyan,24,,Chennai)
(7,Govind,Rangarajan,23,,Chennai)
(8,Majunu,Periyasamy,21,,Hyderabad)
(9,Pakurutin,Honda,22,,Chennai)
(10,Shanmugam,Karthikeyan,21,,Bhuwaneshwar)
(,,,,,)
(1,Nithya,Duraisamy,31,,Hyderabad)
(2,Nandhini,Babu,28,,Kolkata)
(3,Madhuri,Nathan,51,,Kolkata)
(4,Kavitha,Manoharan,45,,Hyderabad)
(5,Vijaya,Kandasamy,45,,Bhuwaneshwar)
(6,Aarthi,Raj,28,,Chennai)
(7,Lavanya,Sankar,23,,Chennai)
(8,Meena,Baskar,51,,Hyderabad)
(9,Gayathri,Ragu,22,,Chennai)
(10,Kavitha,Manoharan,45,,Bhuwaneshwar)
```

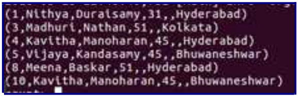
13. SPLIT என்பது ஒரு relation-ல் இருக்கும் தகவல்களைக் குறிப்பிட்ட condition-ஐக் கொடுத்துப் பிரித்து, அதன் மதிப்புகளை இருவேறு relation-ல் சேமிக்க உதவும். கீழ்க்கண்ட எடுத்துக்காட்டில் b-ல் உள்ள

தகவல்களில், 28 வயதுக்குக் கீழ் உள்ளவர்களை x-லும், அதற்கு மேல் உள்ளவர்களை y-லும் சேமித்துள்ளது.

```
grunt> SPLIT b into x if age<28,
y if age>28;
grunt> dump x;
grunt> dump y;
```



```
(7,Lavanya,Sankar,23,,Chennai)
(9,Gayathri,Ragu,22,,Chennai)
```



```
(1,Nithya,Duraisamy,31,,Hyderabad)
(3,Madhuri,Nathan,51,,Kolkata)
(4,Kavitha,Manoharan,45,,Hyderabad)
(5,Vijaya,Kandasamy,45,,Bhuwaneshwar)
(8,Meena,Baskar,51,,Hyderabad)
(10,Kavitha,Manoharan,45,,Bhuwaneshwar)
```

.

14. FILTER என்பது condition-ல் பொருந்தும் தகவல்களை மட்டும் தேர்ந்தெடுத்து வெளிப்படுத்தும் .

```
grunt> l = FILTER b BY city ==
'Chennai';
grunt> dump l;
```

```
(6,Aarthi,Raj,28,,Chennai)
(7,Lavanya,Sankar,23,,Chennai)
(9,Gayathri,Ragu,22,,Chennai)
```

15. DISTINCT என்பது ஒரு record-ஐ ஒருமுறை மட்டுமே வெளிப்படுத்தும் .

```
grunt> m = DISTINCT b by city;
grunt> dump m;
```

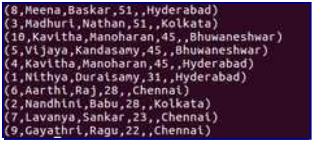
16. FOREACH எனும் operator

```
grunt> n = FOREACH b GENERATE
fname,age,city;
grunt> dump n;
```

```
(Nithya,31,Hyderabad)
(Nandhini,28,Kolkata)
(Madhuri,51,Kolkata)
(Kavitha,45,Hyderabad)
(Vijaya,45,Bhuwaneshwar)
(Aarthi,28,Chennai)
(Lavanya,23,Chennai)
(Meena,51,Hyderabad)
(Gayathri,22,Chennai)
(Kavitha,45,Bhuwaneshwar)
```

17. Order By என்பது தரவுகளை ஏறுவரிசையிலோ, இறங்கு வரிசையிலோ முறைப்படுத்தி வெளிப்படுத்தும்.

```
grunt> o = ORDER b by age DESC;
grunt> dump o;
```



```
(8,Meena,Baskar,51,,Hyderabad)
(3,Madhuri,Nathan,51,,Kolkata)
(10,Kavitha,Manoharan,45,,Bhuwaneshwar)
(5,Vijaya,Kandasamy,45,,Bhuwaneshwar)
(4,Kavitha,Manoharan,45,,Hyderabad)
(1,Nithya,Duraisamy,31,,Hyderabad)
(6,Aarthi,Raj,28,,Chennai)
(2,Nandhini,Babu,28,,Kolkata)
(7,Lavanya,Sankar,23,,Chennai)
(9,Gayathri,Ragu,22,,Chennai)
```

18. LIMIT என்பது relation-ல் உள்ள தரவுகளை குறிப்பிட்ட அளவில் நிறுத்த உதவும்
.

```
grunt> p = LIMIT b 4;
grunt> dump p;
```

```
(1,Nithya,Duraisamy,31,,Hyderabad)
(2,Nandhini,Babu,28,,Kolkata)
(3,Madhuri,Nathan,51,,Kolkata)
(4,Kavitha,Manoharan,45,,Hyderabad)
```

மேற்கண்டவைகளோடு சேர்த்து pig-ல் பல்வேறு வகையான built-in-operators மற்றும் user-defined functions காணப்படுகின்றன.

9 Hive

Facebook நிறுவனம் hadoop-ஐ பயன்படுத்தத் துவங்கிய காலங்கள் முதல், அதனிடம் வந்து சேரும் தரவுகளின் அளவு 1GB, 1TB, 15TB என உயர்ந்து கொண்டே சென்றது. அப்போது அவற்றினை அலசி தரவுச் சுருக்கங்களைத் தேர்ந்தெடுப்பதற்கு oracle database-ஐயும் பைதான் மொழியையும் பயன்படுத்தியது. ஆனால் வருகின்ற மூலத் தரவுகளின் அளவும், வடிவங்களும் அதிகரிக்க அதிகரிக்க data analysis தேவைக்கென ஒரு புதிய முறை கண்டுபிடித்தாக வேண்டி இருந்தது

. அப்போதுதான் facebook நிறுவனம் இத்தகைய datawarehouse தேவைகளைப் பூர்த்தி செய்வதற்கென்றே Hive எனும் ஒரு புதிய கருவியைக் கண்டுபிடித்தது. இக்கருவி rdbms-ன் கோட்பாடுகளை உள்ளடக்கிய, SQL-ஐப் போன்றே செயல்படக் கூடிய , ஆனால் முழுக்க முழுக்க analysis தேவைக்கென உருவாக்கப்பட்ட ஒரு data warehouse framework ஆகும். இதனை transactional database-ஆகப் பயன்படுத்த முடியாது. பின்னாளில் இது Apache நிறுவனத்தால் திறந்த மூல மென்பொருள் கருவியாக மாற்றப்பட்டு மக்கள் பயன்பாட்டிற்கு வழங்கப்பட்டது .

Pig மற்றும் hive ஆகிய இரண்டும் sql-ஐப் போன்ற மொழியைப் பயன்படுத்தி

mapreduce jobs-ஐ எழுதி hadoop-ல் உள்ள தரவுகளைக் கையாளும். Pig பயன்படுத்துவது piglatin என்றும், Hive பயன்படுத்துவது HQL என்றும் அழைக்கப்படும். இவை இரண்டின் செயல்பாடுகளும் ஒத்திருந்தாலும், Hive என்பது analysis தேவைக்கேன்றே உருவாக்கப்பட்ட பிரத்தியேகக் கருவியாக விளங்குவதால், data warehouse மற்றும் BI Reporting துறையில் இருப்பவர்கள் hive-ஐயும், ஆராயிச்சித் துறையில் இருப்பவர்கள் மற்றும் நிரலாளர்கள் pig-ஐயும் தேர்ந்தெடுப்பார்கள். மேலும் Hive என்பது வடிவான தகவல்களை மட்டுமே கையாளும். அதாவது hdfs-ல் கோப்பு வடிவில் சேமிக்கப்பட்டுள்ள தரவுகளும் hive-ல் அட்டவணை வடிவில் இருக்கும்.

9.1 Hive-ன் சிறப்பம்சங்கள்

- HQL என்பது Hive Query Language எனப்படும். இதன் வடிவம் sql-ஐப் போன்றதே. Hive> எனப்படும் command line interface மூலம் இதனை நாம் இயக்கலாம்.
- HQL-ன் இயக்கமானது தொடர்ச்சியாக இயங்கக் கூடிய பல்வேறு mapreduce jobs-ஆக அமையும். hive கட்டமைப்பில் இதன் இயக்கத்தைப் பற்றித் தெளிவாகப் பார்க்கலாம்.
- இதில் database மற்றும் tables தனியாக உருவாக்கப்படுகின்றன. அதன் பின்னர்தான் தகவல்கள் உட்செலுத்தப்படுகின்றன.
- இது structured data-வுக்கென்றே சிறப்பாக வடிவமைக்கப்பட்ட கருவியாதலால், query optimization-க்குத் தேவையான

பல்வேறு காரணிகள்(udf, filtering etc) இதில் காணப்படுகின்றன. Query optimization என்பது query-ன் இயக்கத்தை இன்னும் துரிதமாக்க உதவுவதாகும்.

- Hive-ல் உள்ள அட்டவணைகளின் schema பற்றிய தகவல்களை சேமிப்பதற்காக Metastore-ஐயும், அந்த அட்டவணைகளில் process-செய்யப்பட்ட அல்லது செய்யப்படப்போகும் தரவுகளை சேமிப்பதற்காக hdfs/hbase-ஐயும் hive பயன்படுத்துகிறது.
- mysql, derby, mongodb போன்ற ஏதாவதொரு relational database-ஐ நாம் metastore-ன் தேவைக்காகப் பயன்படுத்தலாம். jdbc, web gui போன்றவற்றைக் கொண்டு அதனுடன் இணைப்பை ஏற்படுத்தலாம்.

- Derby-ஆனது தனியொரு பயனரை மட்டும் அதரிக்கிறது. ஆனால் Mysql பல்வேறு பயணர்களை அதரிப்பதால் இதனையே பெரும்பாலோர் தேர்ந்தெடுக்கின்றனர். single metadata எனும் போது derby ஐயும், shared metadata எனும்போது mysql ஐயும் பயன்படுத்தலாம்.
- hdfs-க்கும் hive-க்கும் இடையில் தரவுகளை பரிமாறிக் கொள்வதற்கு web ui, command line, hd insight ஆகிய 3 வகையான interfaces உள்ளன. இப்பகுதியில் நாம் பயன்படுத்தியுள்ள அனைத்து உதாரணங்களும் hive command line மூலம் காட்டப்பட்டுள்ளன.
- Mapreduce-க்கு மாற்றாக metadata-ன் மீது இயக்கப்படுகின்ற query வடிவமே HQL என்று அழைக்கப்படுகிறது. இந்த query

வடிவத்தை இயக்குகின்ற இயந்திரம் தான் HiveQL Process Engine ஆகும்.

- பின்வரும் 4 விதமான கோப்பு வடிவங்களை hive ஆதரிக்கிறது : text, sequence, orc & rc files.

9.2 Hive-ஐ நிறுவுதல்

1. Hive-ஐ பதிவிறக்கம் செய்து, பிரித்து, நமக்கு வேண்டிய இடத்தில் மாற்றி வைக்கவும்.

```
$ wget http://www-us.apache.org/dist/hive/hive-2.2.0/apache-hive-2.2.0-bin.tar.gz
```

```
$ tar -xzvf apache-hive-2.2.0-bin.tar.gz
$ sudo mv ~/apache-hive-2.2.0-bin /usr/local
```

2. hive-ன் 2.2.0-வுடன் ஒத்துப்போகும் hadoop version-ஆன 2.6.5-வையும் பதிவிறக்கம் செய்து அதே இடத்தில் மாற்றி வைத்துக் கொள்ளவும்.

```
$ wget http://redrockdigimark.com/apache
```

```
mirror/hadoop/common/hadoop-
2.6.5/hadoop-2.6.5.tar.gz
$ tar -xzvf hadoop-2.6.5.tar.gz
$ sudo mv ~/hadoop-2.6.5
/usr/local
```

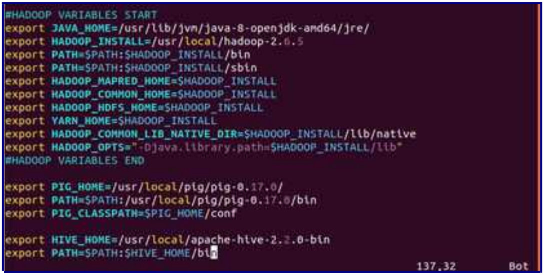
/usr/local எனுமிடத்தில் hive மற்றும் hadoop சேமிக்கப்பட்டுள்ளதைக் காணலாம்.

```
nithya@nithya-Lenovo-ideapad:~$ ls /usr/local
apache-hive-2.2.0-bin  bin  etc  games  hadoop  hadoop-2.6.5  hadoop_store  incl
ude  lib  man  pig  sbin  share  src
```

இந்த hadoop-ன் புதிய version-ல் அது செயல்படுவதற்குத் தேவையான அனைத்து விதமான மாற்றங்களையும் hadoop-env.sh,core-site.xml,mapred-site.xml,hdfs-site.xml முதலிய கோப்புகளிலும் செய்துவிடவும். இவை அனைத்தும் /usr/local/hadoop-2.6.5/etc/hadoop எனுமிடத்தின் கீழ் காணப்படும். இதை எவ்வாறு செய்வது என்பது பற்றி நாம் முன்னரே பார்த்துள்ளோம்.

2. பின்னர் நமது home directory-ல் உள்ள .bashrc எனும் கோப்பிற்குள் பின்வருமாறு விவரங்களை இணைக்கவும்.

```
$ vim ~/.bashrc
export HIVE_HOME=/usr/local/apache-hive-2.2.0-bin
export PATH=$PATH:$HIVE_HOME/bin
export HADOOP_INSTALL=/usr/local/hadoop-2.6.5
export PATH=$PATH:$HADOOP_INSTALL/bin
$ source ~/.bashrc
```



```
#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre/
export HADOOP_INSTALL=/usr/local/hadoop-2.6.5
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
#HADOOP VARIABLES END

export PIG_HOME=/usr/local/pig/pig-0.17.0/
export PATH=$PATH:/usr/local/pig/pig-0.17.0/bin
export PIG_CLASSPATH=$PIG_HOME/conf

export HIVE_HOME=/usr/local/apache-hive-2.2.0-bin
export PATH=$PATH:$HIVE_HOME/bin
                                                            137,32          Bot
```

3. hive-config.sh-க்குள் பின்வருமாறு சேர்க்கவும்

```
$ vim /usr/local/apache-hive-2.2.0-bin/bin/hive-config.sh
export HADOOP_INSTALL=/usr/local/hadoop-2.6.5
```

```
# Default to use 256MB
export HADOOP_HEAPSIZE=${HADOOP_HEAPSIZE:-256}
export HADOOP_INSTALL=/usr/local/hadoop-2.6.5
<ache-hive-2.2.0-bin/bin/hive-config.sh" 71L, 1946C
```

4. பொதுவாக hive-ஆனது அதன் அட்டவணைகளின் schema பற்றிய தகவல்களை சேமித்து வைப்பதற்காக derby-ன் metastore-ஐத்தான் பயன்படுத்தும். ஆனால் derby-ஐ விட mysql சிறந்தது. ஏனெனில் mysql-ஐ backend-ல் பயன்படுத்தும்போது ஒரே நேரத்தில் பல்வேறு பயனர்கள் hive-ஐத் தொடர்பு கொள்ள முடியும். இவ்வசதி derby-ல் கிடையாது. எனவே இப்போது mysql-ஐ நிறுவுவதற்கான கட்டளைகள் பின்வருமாறு அமையும்.

4.1. mysql server-ஐ நமது கணினியில் பதிவிறக்கம் செய்து நிறுவவும்.

```
$ sudo apt-get install mysql-server
```

4.2. அதற்கான java connector-ஐயும் நிறுவவும். இதுவே hive-வுடன் இணைப்பதற்கு உதவும்.

```
$ sudo apt-get install libmysql-java
```

4.3. அந்த java connector-க்கு hive-ன் lib-க்குள் சென்று ஒரு இணைப்பை (soft link) உருவாக்கவும்.

```
$ ln -s /usr/share/java/mysql-connector-java.jar /usr/local/apache-hive-2.2.0-bin/lib/mysql-connector-java.jar
```

4.4. பின்னர் hive-ன் conf-க்குள் உள்ள hive-default.xml.template எனும் கோப்பினை hive-site.xml என மாற்றி அதற்குள் உள்ள மதிப்புகளைக் கீழே குறிப்பிட்டுள்ளவாறு மாற்றி/சேர்த்து அமைக்கவும்.

```
$ cp /usr/local/apache-hive-
2.2.0-bin/conf/hive-
default.xml.template
/usr/local/apache-hive-2.2.0-bin/
conf/hive-site.xml
$ vim /usr/local/apache-hive-
2.2.0-bin/conf/hive-site.xml
```

```
system:java.io.tmpdir
/tmp/hive/java
system:user.name
${user.name}
javax.jdo.option.ConnectionURL
jdbc:mysql://localhost/metastore?
createDatabaseIfNotExist=true&use
SSL=false
javax.jdo.option.ConnectionDriver
Name com.mysql.jdbc.Driver
javax.jdo.option.ConnectionUserNa
me
nithya

javax.jdo.option.ConnectionPasswo
rd
nithya
```

இதில் கொடுக்கப்பட்டுள்ள nithya , nithya என்பது mysql-ஐ அணுகுவதற்கு முழுமையான அனுமதிகள் பெற்ற பயனர்பெயர் மற்றும் கடவுச்சொல் ஆகும்.

4.5. hive-site.xml -க்குள் system:java.io.tmpdir -ன் மதிப்பாக நாம் கொடுத்துள்ளது போலவே ஒரு directory-ஐ உருவாக்கிவிடவும்.

```
$ mkdir -p /tmp/hive/java
```

4.6. hadoop-ஐ இயக்கி அதன் root (/) -க்குள் user/hive எனும் directory-ஐ உருவாக்கவும். hive இங்கிருந்துதான் தரவுகளை எடுத்து process செய்யும். மேலும் hive-ல் உருவாக்கப்படும் schema, tables, partitions, buckets அனைத்தும் இந்த இடத்தில் warehouse எனும் directory-ஐ உருவாக்கி அதன் கீழ் சேமிக்கப்படும்.

```
$ /usr/local/hadoop-2.6.5/sbin/start-all.sh
$ hadoop fs -mkdir /user
$ hadoop fs -mkdir /user/hive
$ hadoop fs -chmod g+w /user/hive
```

4.6. இது hive-க்கும் mysql-க்கும் முதன்முதலில் தொடர்பை ஏற்படுத்த உதவும். இதன் பின்னர் இரண்டும் sync-ல் செயல்பட்டுக் கொண்டிருக்கும்.

```
$ /usr/local/apache-hive-2.2.0-
bin/bin/schematool -initSchema -
dbType mysql
```

5. இது hive shell-ஐ உருவாக்கும்.

```
$ hive
```

```
nithya@nithya-Lenovo-ideapad:~$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/apache-hive-2.2.0-bin/lib/log4j-slf
4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.6.5/share/hadoop/common/li
b/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/usr/local/apache-hive-2.2.0
-bin/lib/hive-common-2.2.0.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versio
ns. Consider using a different execution engine (i.e. spark, tez) or using Hive
1.X releases.
hive> 
```

9.3 Hive-ன் செயல்பாடுகள்

1. Hive-ல் பயன்படுத்தப்படும் மிக அடிப்படையான கட்டளைகள் கீழே

கொடுக்கப்பட்டுள்ளன. அதற்கான விளக்கங்களை வலத்துப்புறத்தில் காணலாம்.

```
> create database exercises; (ஒரு db-ஐ உருவாக்கும்)
> show databases; (அனைத்து db-ஐயும் பட்டியலிடும் )
> alter table states_of_india rename to indian_states;(ஒரு table-ஐ பெயர்மாற்றம் செய்யும்)
> show tables; (அனைத்து db-ஐயும் பட்டியலிடும்)
> drop table indian_states; (அனைத்து tables-ஐயும் பட்டியலிடும்)
> drop view aaaa; (ஒரு view-ஐ நீக்க உதவும்)
```

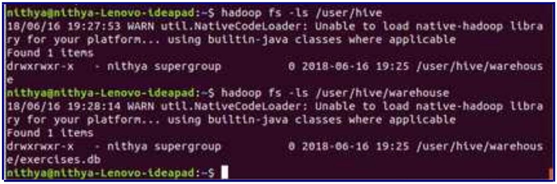
```
> drop database sample; (ஒரு db-ஐ
நீக்க உதவும்)
```

கீழ்க்கண்ட உதாரணத்தில் exercises எனும் பெயர் கொண்ட db-ஐ உருவாக்கியுள்ளோம்.

```
hive> show databases;
OK
default
Time taken: 2.968 seconds, Fetched: 1 row(s)
hive> create database exercises;
OK
Time taken: 3.788 seconds
hive> exit;
```

உடனே hadoop-ல் /user/hive எனுமிடத்தில் சென்று பார்த்தீர்களானால்

warehouse எனும் directory உருவாக்கப்பட்டிருக்கும். அதற்குள் சென்று பார்த்தால், நாம் உருவாக்கிய exercises.db என்பது காணப்படும். இனிமேல் இந்த db-க்குள் தான் நாம் உருவாக்கப்போகும் அனைத்தும் காணப்படும்.



```
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -ls /user/hive
18/06/16 19:27:53 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Found 1 items
drwxrwxr-x   - nithya supergroup          0 2018-06-16 19:25 /user/hive/warehous
e
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -ls /user/hive/warehouse
18/06/16 19:28:14 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Found 1 items
drwxrwxr-x   - nithya supergroup          0 2018-06-16 19:25 /user/hive/warehous
e/exercises.db
nithya@nithya-Lenovo-ideapad:~$ 
```

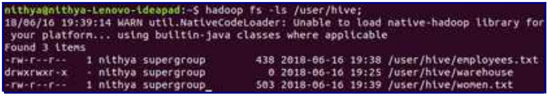
2. RDBMS-ஐப் போலவே இதிலும் அட்டவணைகளை உருவாக்கி பல்வேறு operations-ஐச் செய்யலாம். ஆனால் இதில்

உருவாக்கப்படும் அட்டவணைகளை internal, external என்று இரண்டு விதமாகப் பிரிக்கலாம். இவைகள் முறையே data on schema, schema on data என்று அழைக்கப்படுகின்றன. இவ்விரண்டுக்கும் உள்ள வித்தியாசம் பற்றியும், எவ்வகையில் இவ்விதம் அழைக்கப்படுகின்றன என்பது பற்றியும் கீழே காணலாம்.

2.1 நமது கணினியில் employees.txt , women.txt ஆகிய கோப்பிற்குள் ஒருசில விவரங்கள் உள்ளன. அவற்றை நாம் hive-ல் அட்டவணைகளாக உருவாக்கப் போகிறோம். employees எனும் கோப்பிற்குள் உள்ளவற்றை internal table-ஆகவும், women எனும் கோப்பிற்குள் உள்ளவற்றை

external table-ஆகவும் உருவாக்கப்போகிறோம். முதலில் இவ்விரண்டு கோப்புகளையும் /user/hive எனுமிடத்திற்கு மாற்றவும். அப்போதுதான் hive எடுத்து process செய்யும்.

```
$ hadoop fs -put
~/employees.txt /user/hive;
$ hadoop fs -put ~/women.txt
/user/hive;
```

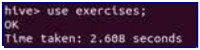


```
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -ls /user/hive;
18/06/16 19:39:14 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
 your platform... using builtin-java classes where applicable
Found 3 items
-rw-r--r--   1 nithya supergroup        438 2018-06-16 19:38 /user/hive/employees.txt
drwxrwxr-x   - nithya supergroup          0 2018-06-16 19:25 /user/hive/warehouse
-rw-r--r--   1 nithya supergroup_       503 2018-06-16 19:39 /user/hive/women.txt
```

2.2 employees எனும் பெயர்கொண்ட ஒரு internal அட்டவணையை உருவாக்குவதற்கான கட்டளைகள் பின்வருமாறு.

hive-க்குள் சென்று exercises db-க்குள் செல்லவும்.

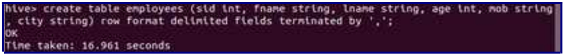
```
> use exercises;
```



```
hive> use exercises;
OK
Time taken: 2.688 seconds
```

முதலில் employees எனும் அட்டவணையை மட்டும் உருவாக்குவதற்கான கட்டளை பின்வருமாறு.

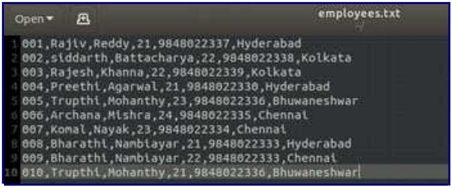
```
> create table employees (sid
int, fname string, lname string,
age int, mob string, city string)
row format delimited fields
terminated by ',';
```



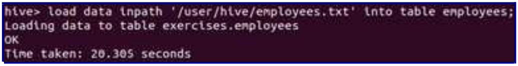
```
hive> create table employees (sid int, fname string, lname string, age int, mob string
, city string) row format delimited fields terminated by ',';
OK
Time taken: 16.961 seconds
```

இப்போது கோப்பிற்குள் உள்ளவற்றை அட்டவணைக்குள் மாற்றுவதற்கான கட்டளை பின்வருமாறு.

```
>load data inpath
'/user/hive/employees.txt' into
table employees;
```



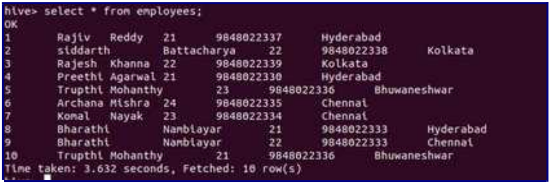
```
001,Rajiv,Reddy,21,9848022337,Hyderabad
002,siddarth,Battacharya,22,9848022338,Kolkata
003,Rajesh,Khanna,22,9848022339,Kolkata
004,Preethi,Agarwal,21,9848022330,Hyderabad
005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar
006,Archana,Mishra,24,9848022335,Chennai
007,Komal,Nayak,23,9848022334,Chennai
008,Bharathi,Nambiayar,21,9848022333,Hyderabad
009,Bharathi,Nambiayar,22,9848022333,Chennai
010,Trupthi,Mohanthy,21,9848022336,Bhuwaneshwar
```



```
hive> load data inpath '/user/hive/employees.txt' into table employees;
Loading data to table exercises.employees
OK
Time taken: 20.305 seconds
```

அடுத்து அட்டவணைக்குள் சென்று எல்லாம் சரியாக ஏற்றப்பட்டுவிட்டதா என சரிபார்க்கவும்.

```
>select * from employees;
```



```
hive> select * from employees;
OK
1       Rajiv   Reddy   21      9848022337      Hyderabad
2       siddarth        Battacharya     22      9848022338      Kolkata
3       Rajesh  Khanna  22      9848022339      Kolkata
4       Preethi Agarwal 21      9848022330      Hyderabad
5       Trupthi Mohanthy        23      9848022336      Bhuwaneshwar
6       Archana Mishra  24      9848022335      Chennai
7       Komal   Nayak   23      9848022334      Chennai
8       Bharathi        Nambiayar       21      9848022333      Hyderabad
9       Bharathi        Nambiayar       22      9848022333      Chennai
10      Trupthi Mohanthy        21      9848022336      Bhuwaneshwar
Time taken: 3.632 seconds, Fetched: 10 row(s)
```

கடைசியாக hadoop-க்குள் சென்று பார்த்தால் நாம் உருவாக்கிய அட்டவணை exercise db-க்குள் காணப்படும். அதற்குள் சென்று பார்த்தால் அந்த அட்டவணைக்கான மூலக்கோப்பு காணப்படும். இம்முறையில் data-ஆனது schema-க்குள் சென்று சேருவதால் இது 'data on schema'

என்றழைக்கப்படுகிறது.

```
>hadoop fs -ls
/user/hive/warehouse/exercises.db
/employees
```

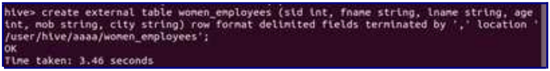
```
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -ls /user/hive/warehouse/exercises.db/employ
ees
18/06/16 19:52:10 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
 your platform... using builtin-java classes where applicable
Found 1 items
-rwxrwxr-x   1 nithya supergroup        438 2018-06-16 19:38 /user/hive/warehouse/exer
cises.db/employees/employees.txt
```

2.3 women_employees எனும் பெயர்கொண்ட ஒரு external அட்டவணையை உருவாக்குவதற்கான கட்டளைகள் பின்வருமாறு.

women_employees எனும் பெயர் கொண்ட அட்டவணையை external-ஆக உருவாக்குவதற்கு create external table எனும் வார்த்தை வெளிப்படையாகக் கொடுக்கப்பட்டுள்ளது . மேலும் hadoop-ன் எந்த இடத்தில் உருவாக்க வேண்டும் என்பதை location மூலம் குறிப்பிட்டுள்ளது.

```
> create external table
women_employees (sid int, fname
string, lname string, age int,
mob string, city string) row
```

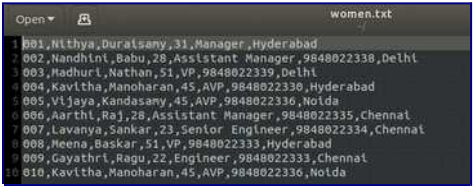
```
format delimited fields
terminated by ',' location
'/user/hive/aaaa/women_employees'
;
```



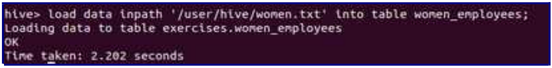
```
hive> create external table women_employees (sid int, fname string, lname string, age
int, mob string, city string) row format delimited fields terminated by ',' location '
/user/hive/aaaa/women_employees';
OK
Time taken: 3.46 seconds
```

கோப்பிற்குள் உள்ளவற்றை அட்டவணைக்குள் மாற்றுவதற்கான கட்டளை பின்வருமாறு.

```
> load data inpath
'/user/hive/women.txt' into table
women_employees;
```



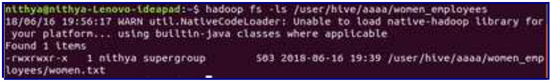
```
001,Nithya,Duraisamy,31,Manager,Hyderabad
002,Nandhini,Babu,28,Assistant Manager,9848022338,Delhi
003,Madhuri,Nathan,51,VP,9848022339,Delhi
004,Kavitha,Manoharan,45,AVP,9848022330,Hyderabad
005,Vijaya,Kandasamy,45,AVP,9848022336,Noida
006,Aarthi,Raj,28,Assistant Manager,9848022335,Chennai
007,Lavanya,Sankar,23,Senior Engineer,9848022334,Chennai
008,Meena,Baskar,51,VP,9848022333,Hyderabad
009,Gayathri,Ragu,22,Engineer,9848022333,Chennai
010,Kavitha,Manoharan,45,AVP,9848022336,Noida
```



```
hive> load data inpath '/user/hive/women.txt' into table women_employees;
Loading data to table exercises.women_employees
OK
Time taken: 2.202 seconds
```

இப்போது hadoop-ல் சென்று பார்த்தால் நாம் உருவாக்கிய அட்டவணை exercises db-ன் கீழ் காணப்படாது. நாம் குறிப்பிட்டுள்ள location-ல் தான் உருவாக்கப்பட்டிருக்கும். அந்த location, hadoop-ல் இல்லையென்றாலும், அதனை உருவாக்கி அதற்குள் கோப்புகளை சேமிக்கும். எனவே இது "Schema on data” என்றழைக்கப்படுகிறது .

```
> hadoop fs -ls
/user/hive/aaaa/women_employees;
```



```
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -ls /user/hive/aaaa/women_employees
18/06/16 19:56:17 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
 your platform... using builtin-java classes where applicable
Found 1 items
-rwxrwxr-x   1 nithya supergroup        503 2018-06-16 19:39 /user/hive/aaaa/women_emp
loyees/women.txt
```

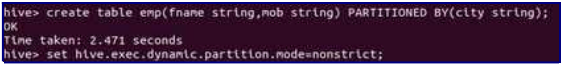
3. Partitions என்பது ஒரு அட்டவணையை பல்வேறு பகுதிகளாகப் பிரிப்பதே ஆகும். ஒரு பெரிய அட்டவணையிலிருந்து நமக்குத் தேவைப்படும் columns-ஐ மட்டும் எடுத்து, அவற்றை நமக்கேற்றவாறு பிரித்து, ஒரு சிறிய அட்டவணையாக சேமிப்பதே partitions எனப்படும். அவ்வாறு பிரித்து எடுப்பதற்கு நாம் பயன்படுத்துவது partition key என்றழைக்கப்படும்.

மேலே நாம் உருவாக்கிய employee அட்டவணையிலிருந்து fname, mob columns-ல் இருக்கும் தரவுகளை மட்டும்

எடுத்து emp எனும் ஒரு சிறிய அட்டவணையை நாம் உருவாக்கப் போகிறோம். அடுத்து அதிலுள்ளவற்றை city column-ல் இருக்கும் மதிப்புகளின் அடிப்படையில் பிரித்து சேமிக்கப்போகிறோம். இதிலுள்ள படிகள் பின்வருமாறு.

பின்வரும் கட்டளை emp எனும் ஒரு அட்டவணையை உருவாக்கி, அதில் சேமிக்கப்படும் தரவுகளை city-ன் அடிப்படையில் பல்வேறு பகுதிகளாகப் பிரிக்கும். அடுத்த கட்டளை அட்டவணை dynamic-ஆகப் பிரிக்கப்படுவதை அனுமதிக்கும் ஒரு பண்பின் மதிப்பினை set செய்கிறது.

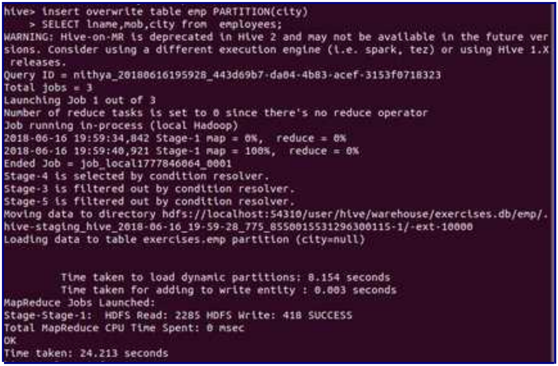
```
> create table emp(fname
string,mob string) PARTITIONED
BY(city string);
> set
hive.exec.dynamic.partition.mode=
nonstrict;
```



```
hive> create table emp(fname string,mob string) PARTITIONED BY(city string);
OK
Time taken: 2.471 seconds
hive> set hive.exec.dynamic.partition.mode=nonstrict;
```

partition அட்டவணைக்குள் தரவுகளை ஏற்றுவதற்கான கட்டளை பின்வருமாறு.

```
>insert overwrite table emp
PARTITION(city)
SELECT lname,mob,city from
employees;
```



```
hive> insert overwrite table emp PARTITION(city)
    > SELECT lname,mob,city from  employees;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future ver
sions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X
 releases.
Query ID = nithya_20180616195928_443d69b7-da04-4b83-acef-3153f0718323
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Job running in-process (local Hadoop)
2018-06-16 19:59:34,842 Stage-1 map = 0%,  reduce = 0%
2018-06-16 19:59:40,921 Stage-1 map = 100%,  reduce = 0%
Ended Job = job_local1777846064_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to directory hdfs://localhost:54310/user/hive/warehouse/exercises.db/emp/.
hive-staging_hive_2018-06-16_19-59-28_775_8550015531296300115-1/-ext-10000
Loading data to table exercises.emp partition (city=null)


         Time taken to load dynamic partitions: 0.154 seconds
         Time taken for adding to write entity : 0.003 seconds
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 2285 HDFS Write: 418 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 24.213 seconds
```

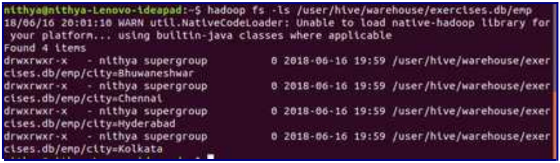
இப்போது அந்த அட்டவணையில் சென்று பார்த்தால் தரவுகள் அனைத்தும் city-ன் அடிப்படையில் பிரிக்கப்பட்டிருப்பதைக் காணலாம்.

```
> select * from emp;
```

```
hive> select * from emp;
OK
Mohanthy        9848022336      Bhuwaneshwar
Mohanthy        9848022336      Bhuwaneshwar
Mishra  9848022335      Chennai
Nayak   9848022334      Chennai
Nambiayar       9848022333      Chennai
Reddy   9848022337      Hyderabad
Agarwal 9848022330      Hyderabad
Nambiayar       9848022333      Hyderabad
Battacharya     9848022338      Kolkata
Khanna  9848022339      Kolkata
Time taken: 0.738 seconds, Fetched: 10 row(s)
```

hadoop-ல் சென்று பார்த்தால் city-ன் மதிப்புகளைப் பொறுத்து பல்வேறு பகுதிகள் பிரிக்கப்பட்டிருப்பதைக் காணலாம்.

```
> hadoop fs -ls
/user/hive/warehouse/exercises.db
/emp
```



```
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -ls /user/hive/warehouse/exercises.db/emp
18/06/16 20:01:10 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
 your platform... using builtin-java classes where applicable
Found 4 items
drwxrwxr-x   - nithya supergroup          0 2018-06-16 19:59 /user/hive/warehouse/exer
cises.db/emp/city=Bhuwaneshwar
drwxrwxr-x   - nithya supergroup          0 2018-06-16 19:59 /user/hive/warehouse/exer
cises.db/emp/city=Chennai
drwxrwxr-x   - nithya supergroup          0 2018-06-16 19:59 /user/hive/warehouse/exer
cises.db/emp/city=Hyderabad
drwxrwxr-x   - nithya supergroup          0 2018-06-16 19:59 /user/hive/warehouse/exer
cises.db/emp/city=Kolkata
```

4. Buckets என்பது Partitions-ன் தரவுகளை இன்னும் சிறு பகுதிகளாகப் பிரித்து துரிதமாக இயங்குவதற்கு ஏற்ற வகையில் சேமிக்கும்.

பின்புலத்தில் இயங்கும் hashing algorithm இதற்காகப் பயன்படுகிறது. emp எனும் பெயர் கொண்ட partition-ல் இருந்து, ep எனும் பெயர் கொண்ட bucket-ஐ உருவாக்குவதில் உள்ள படிகள் பின்வருமாறு.

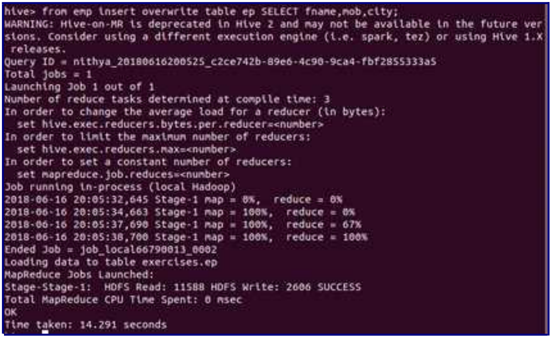
எதனடிப்படையில் எத்தனை buckets உருவாக்க வேண்டும் எனப் பின்வருமாறு குறிப்பிடலாம்.

```
> create table ep(fname
string,mob string,city string)
CLUSTERED BY(city) into 3 buckets
row format delimited fields
terminated by ',';
```

```
hive> create table ep(fname string,mob string,city string) CLUSTERED BY(city) into 3 b
uckets row format delimited fields terminated by ',';
OK
Time taken: 2.494 seconds
```

partition-ல் இருந்து bucket-க்குள் தரவுகளை ஏற்றுவதற்கான கட்டளை பின்வருமாறு.

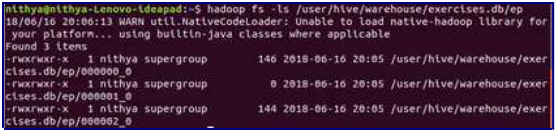
```
> from emp insert overwrite table
ep SELECT fname,mob,city;
```



```
hive> from emp insert overwrite table ep SELECT fname,mob,city;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future ver
sions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X
 releases.
Query ID = nithya_20180616200525_c2ce742b-89e6-4c90-9ca4-fbf2855333a5
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 3
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2018-06-16 20:05:32,645 Stage-1 map = 0%,  reduce = 0%
2018-06-16 20:05:34,663 Stage-1 map = 100%,  reduce = 0%
2018-06-16 20:05:37,690 Stage-1 map = 100%,  reduce = 67%
2018-06-16 20:05:38,700 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local66790013_0002
Loading data to table exercises.ep
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 11588 HDFS Write: 2606 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 14.291 seconds
```

hadoop-ல் சென்று பார்த்தால் ep அட்டவணையின் கீழ் 000000, 000001, 000002 எனும் பெயர் கொண்ட 3 buckets உருவாக்கப்பட்டிருப்பதைக் காணலாம்.

```
> hadoop fs -ls
/user/hive/warehouse/exercises.db
/ep
```



```
nithya@nithya-Lenovo-ideapad:~$ hadoop fs -ls /user/hive/warehouse/exercises.db/ep
18/06/16 20:06:13 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
 your platform... using builtin-java classes where applicable
Found 3 items
-rwxrwxr-x   1 nithya supergroup        146 2018-06-16 20:05 /user/hive/warehouse/exer
cises.db/ep/000000_0
-rwxrwxr-x   1 nithya supergroup          0 2018-06-16 20:05 /user/hive/warehouse/exer
cises.db/ep/000001_0
-rwxrwxr-x   1 nithya supergroup        144 2018-06-16 20:05 /user/hive/warehouse/exer
cises.db/ep/000002_0
```

5. Views என்ற ஒன்றை hive-லும் உருவாக்கலாம். ஒரு query result-ஐ தற்காலிகமாக ஒரு புதிய பெயரில் சேமித்துக்கொள்வதை view என்போம். இங்கு

age எனும் பெயர் கொண்ட view-ஐ உருவாக்குவதற்கான கட்டளை கீழே கொடுக்கப்பட்டுள்ளது.

```
> create view age as select *
from employees where age>22;
```

```
hive> create view age as select * from employees where age>22;
OK
Time taken: 3.447 seconds
```

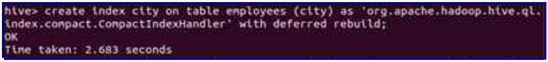
6. Index என்பது அட்டவணையிலுள்ள அனைத்து தகவல்களையும் அணுகுவதற்கான ஒரு reference போன்று செயல்படும். ஒரு அட்டவணையிலுள்ள மிக முக்கிய columns-ன் மீது index-ஐ உருவாக்கி அதனடிப்படையில் நாம் மற்ற columns-ஐ அணுகலாம். Hive-ல் compact, bitmap எனும் 2 வகையான indexes உள்ளன. அதாவது cluster முறையில் பிரித்து வைக்கப்பட்டுள்ள தரவுகளை அதன் index கொண்டு தேடும்போது,

- compact index என்பது நாம் தேடுகின்ற மதிப்பு மற்றும் அது சேமிக்கப்பட்டுள்ள தொகுதியினை (value, block-id) எனும் வடிவில் key-value இணைகளாக சேமித்து வைத்திருக்கும்.
- bitmap index என்பது நாம் தேடுகின்ற மதிப்பு சேமிக்கப்பட்டுள்ள இடங்களின்

பட்டியலை பட வடிவில் சேமித்திருக்கும். அதாவது (value , list of rows as a bitmap) எனும் வடிவில் key-value இணைகளாக இருக்கும்.

6.1 employees அட்டவணையிலுள்ள city column-ன் மீது compact index-ஐ உருவாக்குவதற்கான கட்டளை பின்வருமாறு.

```
> create index city on table
employees (city) as
'org.apache.hadoop.hive.ql.index.
compact.CompactIndexHandler' with
deferred rebuild;
```



```
hive> create index city on table employees (city) as 'org.apache.hadoop.hive.ql.
index.compact.CompactIndexHandler' with deferred rebuild;
OK
Time taken: 2.683 seconds
```

6.2 அதே அட்டவணையில் உள்ள mob column-ன் மீது bitmap index-ஐ உருவாக்குவதற்கான கட்டளை பின்வருமாறு.

```
> create index mobile on table
employees (mob) as 'bitmap' with
deferred rebuild;
```

```
hive> create index mobile on table employees (mob) as 'bitmap' with deferred reb
uild;
OK
Time taken: 1.404 seconds
```

6.3 பின்வரும் கட்டளை ஒரு அட்டவணையிலுள்ள அனைத்து index-ஐயும் வெளிப்படுத்தும்.

```
> show formatted index on
employees;
```

```
hive> show formatted index on employees;
OK
idx_name             tab_name             col_names            idx_tab_name              idx_type             comment


city                 employees            city                 exercises__employees_city__   compact
mobile               employees            mob                  exercises__employees_mobile__ bitmap
Time taken: 0.138 seconds, Fetched: 5 row(s)
```

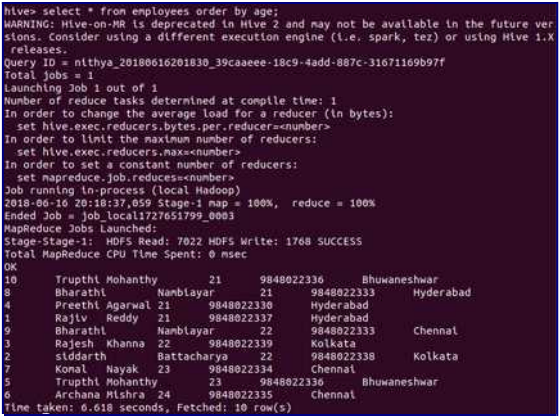
6.4 ஒரு index-ஐ நீக்குவதற்கான கட்டளை பின்வருமாறு.

```
> drop index mobile on employees;
```

```
hive> drop index mobile on employees;
OK
Time taken: 5.587 seconds
```

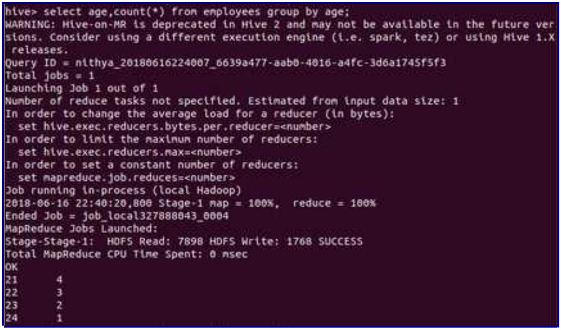
7. sql-ல் பயன்படுத்தப்படும் order by, group by, sort by, inner join, left outer join, right outer join, full outer joins & sub queries ஆகிய அனைத்தும் hive-லும் உள்ளன. இவைகளுக்கான queries-ஐப் பின்வருமாறு காணலாம்.

```
> select * from employees order
by age;
```



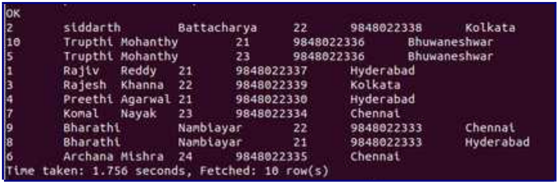
```
hive> select * from employees order by age;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future ver
sions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X
 releases.
Query ID = nithya_20180616201830_39caaeee-18c9-4add-887c-31671169b97f
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2018-06-16 20:18:37,059 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1727651799_0003
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 7022 HDFS Write: 1768 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
10      Trupthi Mohanthy        21      9848022336      Bhuwaneshwar
8       Bharathi        Nambiayar       21      9848022333      Hyderabad
4       Preethi Agarwal 21      9848022330      Hyderabad
1       Rajiv   Reddy   21      9848022337      Hyderabad
9       Bharathi        Nambiayar       22      9848022333      Chennai
3       Rajesh  Khanna  22      9848022339      Kolkata
2       siddarth        Battacharya     22      9848022338      Kolkata
7       Komal   Nayak   23      9848022334      Chennai
5       Trupthi Mohanthy        23      9848022336      Bhuwaneshwar
6       Archana Mishra  24      9848022335      Chennai
Time taken: 6.618 seconds, Fetched: 10 row(s)
```

```
> select age,count(*) from
employees group by age;
```



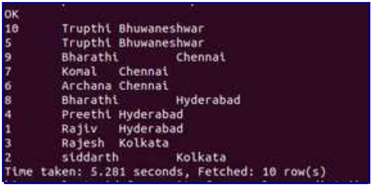
```
hive> select age,count(*) from employees group by age;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future ver
sions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X
 releases.
Query ID = nithya_20180616224007_6639a477-aab0-4016-a4fc-3d6a1745f5f3
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2018-06-16 22:40:20,800 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local327888043_0004
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 7898 HDFS Write: 1768 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
21      4
22      3
23      2
24      1
```

```
> select * from employees sort by
fname desc;
```



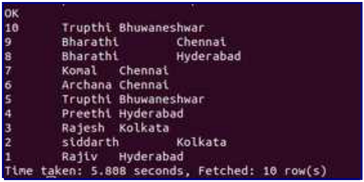
```
OK
2       siddarth        Battacharya     22      9848022338      Kolkata
10      Trupthi Mohanthy        21      9848022336      Bhuwaneshwar
5       Trupthi Mohanthy        23      9848022336      Bhuwaneshwar
1       Rajiv   Reddy   21      9848022337      Hyderabad
3       Rajesh  Khanna  22      9848022339      Kolkata
4       Preethi Agarwal 21      9848022330      Hyderabad
7       Komal   Nayak   23      9848022334      Chennai
9       Bharathi        Nambiayar       22      9848022333      Chennai
8       Bharathi        Nambiayar       21      9848022333      Hyderabad
6       Archana Mishra  24      9848022335      Chennai
Time taken: 1.756 seconds, Fetched: 10 row(s)
```

```
> select sid,fname,city from
employees cluster by city;
```



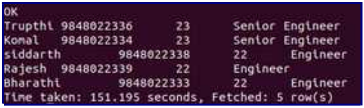
```
OK
10      Trupthi Bhuwaneshwar
5       Trupthi Bhuwaneshwar
9       Bharathi        Chennai
7       Komal   Chennai
6       Archana Chennai
8       Bharathi        Hyderabad
4       Preethi Hyderabad
1       Rajiv   Hyderabad
3       Rajesh  Kolkata
2       siddarth        Kolkata
Time taken: 5.281 seconds, Fetched: 10 row(s)
```

```
> select sid,fname,city from
employees distribute by city;
```



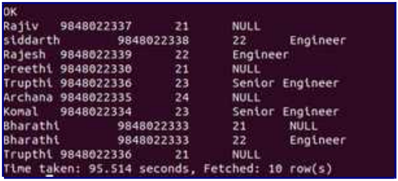
```
OK
10      Trupthi Bhuwaneshwar
9       Bharathi        Chennai
8       Bharathi        Hyderabad
7       Komal   Chennai
6       Archana Chennai
5       Trupthi Bhuwaneshwar
4       Preethi Hyderabad
3       Rajesh  Kolkata
2       siddarth        Kolkata
1       Rajiv   Hyderabad
Time taken: 5.888 seconds, Fetched: 10 row(s)
```

```
> select e.fname, e.mob, e.age,
w.desig FROM employees e JOIN
women w on(e.age=w.age);
```



```
OK
Trupthi 9848022336      23      Senior Engineer
Komal   9848022334      23      Senior Engineer
siddarth        9848022338      22      Engineer
Rajesh  9848022339      22      Engineer
Bharathi        9848022333      22      Engineer
Time taken: 151.195 seconds, Fetched: 5 row(s)
```

```
> select e.fname, e.mob, e.age,
w.desig FROM employees e LEFT
OUTER JOIN women w
on(e.age=w.age);
```



```
OK
Rajiv   9848022337      21      NULL
siddarth        9848022338      22      Engineer
Rajesh  9848022339      22      Engineer
Preethi 9848022330      21      NULL
Trupthi 9848022336      23      Senior Engineer
Archana 9848022335      24      NULL
Komal   9848022334      23      Senior Engineer
Bharathi        9848022333      21      NULL
Bharathi        9848022333      22      Engineer
Trupthi 9848022336      21      NULL
Time taken: 95.514 seconds, Fetched: 10 row(s)
```

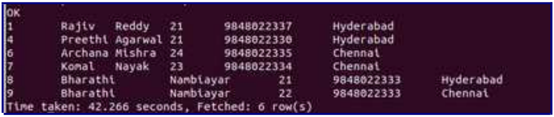
```
> select e.fname, e.mob, e.age,
w.desig FROM employees e RIGHT
OUTER JOIN women w
on(e.age=w.age);
```

```
OK
NULL    NULL    NULL    Manager
NULL    NULL    NULL    Assistant Manager
NULL    NULL    NULL    VP
NULL    NULL    NULL    AVP
NULL    NULL    NULL    AVP
NULL    NULL    NULL    Assistant Manager
Trupthi 9848022336      23      Senior Engineer
Komal   9848022334      23      Senior Engineer
NULL    NULL    NULL    VP
siddarth        9848022338      22      Engineer
Rajesh  9848022339      22      Engineer
Bharathi        9848022333      22      Engineer
NULL    NULL    NULL    AVP
Time taken: 56.104 seconds, Fetched: 13 row(s)
```

```
> select e.fname, e.mob, e.age,
w.desig FROM employees e FULL
OUTER JOIN women w
on(e.age=w.age);
```

```
OK
Trupthi 9848022336      21      NULL
Bharathi        9848022333      21      NULL
Preethi 9848022330      21      NULL
Rajiv   9848022337      21      NULL
Bharathi        9848022333      22      Engineer
Rajesh  9848022339      22      Engineer
siddarth        9848022338      22      Engineer
Komal   9848022334      23      Senior Engineer
Trupthi 9848022336      23      Senior Engineer
Archana 9848022335      24      NULL
NULL    NULL    NULL    Assistant Manager
NULL    NULL    NULL    Assistant Manager
NULL    NULL    NULL    Manager
NULL    NULL    NULL    AVP
NULL    NULL    NULL    AVP
NULL    NULL    NULL    AVP
NULL    NULL    NULL    VP
NULL    NULL    NULL    VP
Time taken: 3.766 seconds, Fetched: 18 row(s)
```

```
> select * from employees where
city in (select city from women);
```



```
OK
1       Rajiv   Reddy   21      9848022337      Hyderabad
4       Preethi Agarwal 21      9848022330      Hyderabad
6       Archana Mishra  24      9848022335      Chennai
7       Komal   Nayak   23      9848022334      Chennai
8       Bharathi        Nambiayar       21      9848022333      Hyderabad
9       Bharathi        Nambiayar       22      9848022333      Chennai
Time taken: 42.266 seconds, Fetched: 6 row(s)
```

10 Spark

Spark என்பது hadoop-ன் துணைத்திட்டமாக 2009-ம் ஆண்டு உருவாக்கப்பட்டது. பின்னர் 2010-ல் திறந்த மூல மென்பொருள் கருவியாக BSD உரிமத்தின் கீழ் வெளியிடப்பட்டது . 2013-ம் ஆண்டு இது அறக்கட்டளையுடன் இணைந்தது முதல் சிறப்பாக செயல்பட்டு வருகிறது. இதிலும் தரவுகளை சேமிக்க hdfs-தான் பயன்படுகிறது. ஆனால் சேமிக்கப்பட்டுள்ள தரவுகளை அணுகுவதற்கு வெறும் mapreduce-யோடு நின்று விடாமல் spark sql, spark streaming,graphx, MLlib (Machine Learning Library) போன்ற பல்வேறு அம்சங்களை வழங்குகிறது. மேலும் java, scala, python போன்ற பல்வேறு மொழிகளை ஆதரிக்கிறது. எனவே Analytics-க்குத் தேவைப்படும் applications-ஐ இவைகளுக்கான API-ஐப் பயன்படுத்தி எந்த மொழியில் வேண்டுமானாலும் எழுதலாம்.

Spark என்பது தரவுகளின் மீது அதிவேகமாக இயங்கி கணக்கீடுகளை நிகழ்த்த உதவும் ஒரு கட்டமைப்பு ஆகும். Scalable, flexible, fault-taularant, cost-effective (இவைகளுக்கான விளக்கத்தைக் கீழே காணலாம்) போன்ற அனைத்து விஷயங்களிலும் Hadoop &

Spark இரண்டும் சிறந்து விளங்கினாலும், வேகம் (speed) எனும் ஒரு விஷயத்தில் மட்டும் hadoop-ஐ விட spark சிறந்து விளங்குகிறது. ஏனெனில் 'in-memory cluster computing' என்பது இதில் ஒரு சிறப்பு அம்சமாக விளங்குகிறது. அதாவது hadoop போன்று ஒவ்வொரு முறையும் hard disk-ல் சென்று தரவுகளைத் தேடாமல் அதிகமாக அணுகப்படும் தரவுகளை RAM-ல் சேமித்து வைத்துக் கொண்டு வேண்டிய நேரத்தில் துரிதமாக எடுத்து வெளிப்படுத்துவதே 'in-memory cluster computing' எனப்படும். அதாவது computing-க்குத் தேவையானவற்றை in-memory யிலும், referencing dataset-ஐ external storage-லும் சேமிக்கிறது. இத்தகைய 'in-memory' functions மூலம் disk-க்குச்

செல்லும் read/write operations-ன் அளவு குறைவதால் இதன் வேகம் அதிகரிக்கிறது.

- scalable : நமது தேவைக்கேற்ப கணினிகளை இணைத்தோ/நீக்கியோ பயன்படுத்தும் வசதி. தரவுகளின் எண்ணிக்கை பெருகப் பெருக கணினிகளை சேர்த்துக்கொண்டே செல்வது scale-up என்றும், தேவையில்லாத தருணங்களில் இணைப்பிலிருந்து நீக்கிவிடுவது scale-down என்றும் அழைக்கப்படும்.
- flexible : நமது தேவைக்கேற்ப ஒன்றை முழுமையாக மாற்றி அமைக்கும் வசதி. உதாரணத்துக்கு Photoshop என்பது flexible அல்ல. ஏனெனில் இதன் config files-ஐ மாற்றி அமைக்கும் உரிமை நமக்குக் கிடையாது.

ஆனால் hadoop & spark போன்றவற்றின் config files அனைத்தும் நமது கட்டுப்பாட்டுக்குள் உள்ளதால் அதனை நாம் எவ்வாறு வேண்டுமானாலும் மாற்றி அமைத்துக் கொள்ளலாம்.

- fault-tolerant : இதில் அவ்வளவு எந்தஒரு தகவல் இழப்போ, தவறுளோ சுலபமாக நிகழ்ந்து விடாத நிலை. இதில் தரவுகள் அனைத்தும் பிரித்து சேமிக்கப்படுகின்றன மற்றும் backup எடுத்து வைக்கப்படுகின்றன. எனவே ஏதேனும் ஒன்றில் தவறு நிகழ்ந்தால் கூட, மற்றொன்று செயல்பட்டு அதனை சரி செய்துவிடும்.
- cost-effective : பெருந்தொகை செலுத்தத் தேவையில்லை. திறந்த மூல நிரலாகக் கிடைக்கிறது.

முன்பெல்லாம் hadoop-ஆல் செய்யப்படும் batch-processing முறையே போதுமானதாக இருந்தது. ஒரு குறிப்பிட்ட விஷயத்தின் அடிப்படையில் தரவுகளைப் பல்வேறு தொகுதிகளாகப் பிரித்து, அவைகளைக் கொண்டு கணக்கீடு செய்வதே batch processing எனப்படும். இது பொதுவாக historical data-வை process செய்ய உதவும். உதாரணத்துக்கு மக்கள் தொகை கணக்கெடுப்பு எனும் நிகழ்வானது ஒட்டு மொத்தமாக கி.பி.1-லிருந்து கி.பி.2018-வரை என்று நிகழாமல், காலநேர அடிப்படையில் ....,1990-2000, 2001-2010,... எனும் பத்து பத்து வருடங்களாகப் பிரிக்கப்பட்டு நிகழ்த்தப்படுவதைச் சொல்லலாம். ஆனால் அண்மையில்தான், நிகழ்காலத்தில் வந்துகொண்டே இருக்கக்கூடிய தரவுகளைக்

கையாள்வதற்கான தேவை ஏற்பட்டது. உதாரணத்துக்கு credit/debit அட்டைகள் வழங்கும் நிறுவனங்களில் ஒரு நிமிடத்திற்கு பல்லாயிரக்கணக்கானோர் அந்த அட்டைகளைத் தேய்த்து பணப் பரிமாற்றம் செய்யும் விவரங்கள் வந்த வண்ணம் இருக்கும். இதுவே continuous streaming எனப்படும். இம்முறையில் வரும் தரவுகள் 'real-time data' அல்லது 'fast-data' எனப்படும். hadoop-ஆல் continuous streaming-ல் வரக்கூடியவற்றைக் கையாள முடியாது. ஆனால் இவற்றைக் கையாள்வதற்கான யுக்திகள் spark-ல் காணப்படுகின்றன. எனவே batch applications, Iterative algorithmns, Interactive queries, continuous steaming போன்ற அதிக அளவு

வேலைப்பலு கொண்ட துறைகளில் Spark பயன்படுத்தப்படுகின்றன.

10.1 Spark-ன் கட்டமைப்புக் கூறுகள்

Spark-core என்பது இதன் அனைத்து விதமான செயல்பாடுகளுக்கும் அடித்தளமாக அமைகின்ற ஒரு விஷயம் ஆகும். இதன் சிறப்பம்சங்களான sql queries, streaming data, machine learning, graph algorithmns ஆகிய அனைத்தும் இதனை மையமாக வைத்தே செயல்படுகின்றன.

- Spark sql : structured மற்றும் semi-structured data-வைக் கையாள்கிறது.

core-ன் மீது அமைக்கப்படும் ஒரு சிறு கருவி ஆகும். schema RDD எனப்படும் ஒரு புதிய முறையினை இது வழங்குகிறது. இதனைப் பற்றிக் கீழே விளக்கமாகக் காணலாம். இதன் மூலம் continuous streaming முறையில் வரும் தரவுகளை சிறுசிறு batches-ஆகப் பிரித்து அதனை RDD-ஆக மாற்றி analytics-ஐ நிகழ்த்துகிறது.

- Streaming data : நிகழ்காலத்தில் தரவுகள் வந்து கொண்டே இருப்பதே streaming data எனப்படும். Cricket scoring, facebook, twitter, banking, whether forecasting போன்றவை இதற்கு உதாரணங்களாக அமையும்.
- MLlib : spark-ஆனது disk-ல் மட்டும்

அல்லாமல் ram-யிலும் தரவுகளை சேமிப்பதால் இத்தகைய distributed memory based architecture-க்கு ஏற்ற வகையில் distributed machine learning framework-ஆக இது சிறந்து விளங்குகிறது. Hadoop கட்டமைப்பில் இருக்கும் mahout -ஐ விட spark-ன் MLlib-ஆனது 9 மடங்கு வேகத்துடன் சிறந்து விளங்குகிறது.

- GraphX : distributed graph processing-க்கான ஒரு framework ஆகும். உதாரணத்துக்கு facebook, twitter போன்ற வலைத்ததளங்களில் ஒருவர் எத்தனை பேருடன் இணைந்துள்ளார், அவர்களுடன் எத்தனை பேர் இணைந்துள்ளனர், எத்தனை பேர் பின் தொடர்கின்றனர் என்பது போன்ற விவரங்களெல்லாம் ஒரு அச்சுப்பட வடிவில் இணைக்கப்பட்டிருக்கும். இத்தகைய

அச்சுப்படங்களை வைத்து தரவுகளைக் கணக்கீடு செய்ய உதவும் ஒரு API-ஆக graphx விளங்குகிறது. Pregal abstraction என்பது அந்த API பெயர் ஆகும்.

10.2 Spark-ஐ நிறுவுதல்

இதனை பின்வரும் 3 விதங்களில் நிறுவலாம்.

- standalone : இதில் spark என்பது hdfs-ன் மீது இயங்குகிறது. அதற்கென்று தனியாக இட ஒதுக்கீடு அளிக்கிறது. இதில் spark-ம் mapreduce-ம் ஒத்தவாறு ஓடிக்கொண்டிருக்கும்.
- Hadoop yarn : cluster management என்பது spark-ன் இயல்பான செயல்பாடுகளில்

ஒன்றாக அமைகிறது. அதற்கென்று தனியாக yarn-என்ற ஒன்றோ, mesos-என்ற ஒன்றோ தேவையில்லை. ஆனால் ஏற்கனவே hadoop-ன் மீது yarn செயல்பட்டுக் கொண்டிருப்பின் அதனை நீக்கிவிட்டு spark-ஐ நிறுவத் தேவையில்லை. அதனுடன் சேர்த்தே spark-ஐ நிறுவலாம்.

- Spark in Map Reduce - hadoop மற்றும் mareduce மட்டும் இயங்கிக்கொண்டிருக்கும் ஒரு கட்டமைப்பில், நமது தேவைக்கேற்ப spark-ஐயும் சேர்த்துக்கொள்ளலாம். இதற்கென்று எந்தஒரு சிறப்பு அனுமதிகளும் தேவையில்லை.

Spark-ஐ standalone mode-ல் நிறுவுவதற்கான படிகள் பின்வருமாறு.

1.Spark-ஐ பதிவிறக்கம் செய்து /usr/local -ல் மாற்றி வைக்கவும்.

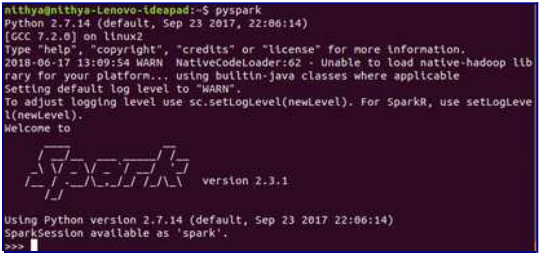
```
$ wget http://redrockdigimark.com/apachemirror/spark/spark-2.3.1/spark-2.3.1-bin-hadoop2.7.tgz
$ tar -xzvf spark-2.3.1-bin-hadoop2.7.tgz
$ sudo mv ~/spark-2.3.1-bin-hadoop2.7 /usr/local
```

2. நமது home directory-ல் உள்ள .bashrc -க்குள் spark-ன் விவரங்களை இணைக்கவும்.

```
$ vim ~/.bashrc
export PATH=$PATH:/usr/local/spark-2.3.1-bin-hadoop2.7/bin
$ source ~/.bashrc
```

3. பின்னர் command line-ல் சென்று pyspark என அடித்தால் spark shell வெளிப்படுவதைக் காணலாம்.

```
$ pyspark
```



```
nithya@nithya-Lenovo-ideapad:~$ pyspark
Python 2.7.14 (default, Sep 23 2017, 22:06:14)
[GCC 7.2.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
2018-06-17 13:09:54 WARN  NativeCodeLoader:62 - Unable to load native-hadoop lib
rary for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve
l(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.3.1
      /_/

Using Python version 2.7.14 (default, Sep 23 2017 22:06:14)
SparkSession available as 'spark'.
>>> 
```

10.3 Resilient Distributed Datasets

இதுவே spark-ன் அடிப்படை. இம்முறையில்தான் spark-ல் தரவுகள் சேமிக்கப்படுகின்றன. ஆய்வாளர்கள் mapreduce-ன் பண்புகளான replication, serialization மற்றும் disk IO போன்றவற்றால் அதன் வேகம் குறைவதைக் கண்டறிந்தனர். எனவே வேகத்தை அதிகரிக்கும் பொருட்டு ‘in-memory computing’ என்ற ஒன்றைக் கண்டறிந்தனர். RDD (Resilient Distributed Datasets) என்பது இதனை implement செய்யும் ஒரு framework-ஆகும்.

அதாவது RDD-ல் சேமிக்கப்படும் தரவுகள் காரிய முறைப்படி (logically) பல்வேறு தொகுதிகளாகப் பிரிக்கப்படும். அவை cluster-ல் இணைக்கப்பட்டுள்ள பல்வேறு nodes-ல் செயல்பட்டு அதனதன் தரவு ஆய்வுக்கான வேலையைத் துவங்குகின்றன. இவையே jobs எனப்படும். இத்தகைய rdd முறையில் தரவுகளை சேமிக்கும்போது அத்தரவுகளுக்கான memory-ஆனது ஒரு object-ஆக சேமிக்கப்பட்டு அவை அதனுடைய jobs-க்கிடையே பகிரப்படுகின்றன. இதுவே immutable distributed collection of objects என்றழைக்கப்படுகிறதுன. RDD என்பதனை python, java, scala போன்ற எந்தஒரு மொழியில் வேண்டுமானாலும் எழுதலாம்.

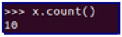
spark shell-ல் எவ்வாறு ஒரு rdd-ஐ உருவாக்கி அதற்குள் விவரங்களை செலுத்துவது என்று பார்க்கலாம். கீழ்க்கண்ட எடுத்துக்காட்டில் x எனும் பெயர் கொண்ட rdd-ஐ உருவாக்கி அதற்குள் employees எனும் கோப்பிற்குள் உள்ள விவரங்கள் உட்செலுத்தப்பட்டுள்ளன. பின்னர் அதன் மீது செயல்படும் collect() எனும் function-ஆனது rdd-ல் உள்ள விவரங்களை எடுத்து வெளிக்காட்டும்.

```
>>> x = sc.textFile
("file:///home/nithya/employees.t
```

```
xt")
>>> x.collect()
```

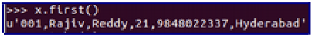
```
>>> x = sc.textFile ("file:///home/nithya/employees.txt")
>>> x.collect()
[u'001,Rajiv,Reddy,21,9848022337,Hyderabad', u'002,siddarth,Battacharya,22,98480
22338,Kolkata', u'003,Rajesh,Khanna,22,9848022339,Kolkata', u'004,Preethi,Agarwa
l,21,9848022338,Hyderabad', u'005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar',
u'006,Archana,Mishra,24,9848022335,Chennai', u'007,Komal,Nayak,23,9848022334,Che
nnai', u'008,Bharathi,Nambiayar,21,9848022333,Hyderabad', u'009,Bharathi,Nambiay
ar,22,9848022333,Chennai', u'010,Trupthi,Mohanthy,21,9848022336,Bhuwaneshwar']
```

```
>>> x.count() (தரவுகளின்
எண்ணிக்கையை வெளிக்காட்டும்)
```



```
>>> x.count()
10
```

>>> x.first() (முதல் dataset-ஐ வெளிக்காட்டும்)



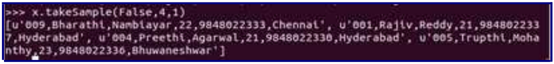
```
>>> x.first()
u'001,Rajiv,Reddy,21,9848022337,Hyderabad'
```

>>> x.take(3) (parameter-ன் எண்ணிக்கையைப் பொறுத்து dataset-ஐ எடுத்து வெளிக்காட்டும்)

```
>>> x.take(3)
[u'001,Rajiv,Reddy,21,9848022337,Hyderabad', u'002,siddarth,Battacharya,22,98480
22338,Kolkata', u'003,Rajesh,Khanna,22,9848022339,Kolkata']
```

```
>>> x.takeSample(False,4,1)
```

(Random-ஆக எடுத்து dataset-ஐ வெளிக்காட்டும்)



```
>>> x.takeSample(False,4,1)
[u'009,Bharathi,Nambiayar,22,9848022333,Chennai', u'001,Rajiv,Reddy,21,984802233
7,Hyderabad', u'004,Preethi,Agarwal,21,9848022330,Hyderabad', u'005,Trupthi,Moha
nthy,23,9848022336,Bhuwaneshwar']
```

10.4 Spark – Mysql இணைப்பை ஏற்படுத்துதல்

Spark மற்றும் Mysql இரண்டையும் இணைத்து எவ்வாறு ஒரு database-ல் உள்ள தரவுகளை spark-க்குள் செலுத்துவது என்று பார்க்கலாம். இதில் உள்ள படிகள் பின்வருமாறு.

1. முதலில் ஒரு சிறிய அட்டவணையை database-ல் உருவாக்கவும்.

```
$ mysql -u nithya -p
> create database sample;
```

```
> use sample;
> create table info(id int,name
varchar(20),age int);
> insert into info
values(1,"Aarthi",25),
(2,"Sathya",18);
> Select * from info;
```

```
mysql> create database sample;
Query OK, 1 row affected (0.06 sec)

mysql> use sample;
Database changed
mysql> create table info(id int,name varchar(20),age int);
Query OK, 0 rows affected (0.49 sec)

mysql> insert into info values(1,"Aarthi",25),(2,"Sathya",18);
Query OK, 2 rows affected (0.11 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> Select * from info;
+------+--------+------+
| id   | name   | age  |
+------+--------+------+
|    1 | Aarthi |   25 |
|    2 | Sathya |   18 |
+------+--------+------+
2 rows in set (0.00 sec)
```

2. mysql.jar எனும் கோப்பினை கீழ்க்கண்ட முகவரியிலிருந்து பதிவிறக்கம் செய்யவும். இதுவே database-வுடன் இணைப்பை ஏற்படுத்துவதற்கு உதவும்.

```
$ wget
http://www.java2s.com/Code/JarDownload/mysql/mysql.jar.zip
```

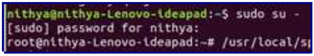
```
nithya@nithya-Lenovo-ideapad:~$ wget http://www.java2s.com/Code/JarDownload/mysq
l/mysql.jar.zip
--2018-06-17 13:33:09--  http://www.java2s.com/Code/JarDownload/mysql/mysql.jar.
zip
Resolving www.java2s.com (www.java2s.com)... 45.40.136.91
Connecting to www.java2s.com (www.java2s.com)|45.40.136.91|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 239337 (234K) [application/zip]
Saving to: 'mysql.jar.zip'

mysql.jar.zip       100%[===================>] 233.73K   115KB/s    in 2.0s

2018-06-17 13:33:12 (115 KB/s) - 'mysql.jar.zip' saved [239337/239337]
```

```
$ unzip mysql.jar.zip
```

```
nithya@nithya-Lenovo-ideapad:~$ unzip mysql.jar.zip
Archive:  mysql.jar.zip
  inflating: mysql.jar
```

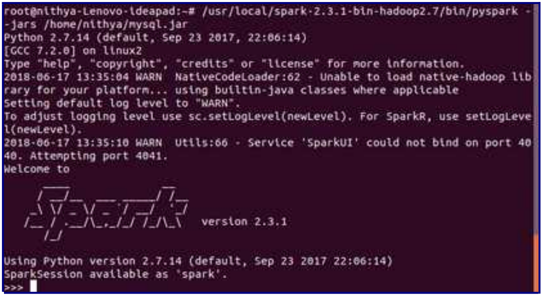
3. அடுத்ததாக pyspark கட்டளையை இந்த jar file-வுடன் இணைத்து பின்வருமாறு இயக்கவும். அதற்கு முன்னர் நீங்கள் இக்கட்டளையை இயக்குவதற்கு root-ஆக login செய்ய வேண்டும். இவை பின்வருமாறு.

```
$ sudo su -
```



```
nithya@nithya-Lenovo-ideapad:~$ sudo su -
[sudo] password for nithya:
root@nithya-Lenovo-ideapad:~# /usr/local/s
```

```
$ /usr/local/spark-2.3.1-bin-
hadoop2.7/bin/pyspark --jars
/home/nithya/mysql.jar
```



```
root@nithya-Lenovo-ideapad:~# /usr/local/spark-2.3.1-bin-hadoop2.7/bin/pyspark -
-jars /home/nithya/mysql.jar
Python 2.7.14 (default, Sep 23 2017, 22:06:14)
[GCC 7.2.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
2018-06-17 13:35:04 WARN  NativeCodeLoader:62 - Unable to load native-hadoop lib
rary for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve
l(newLevel).
2018-06-17 13:35:10 WARN  Utils:66 - Service 'SparkUI' could not bind on port 40
40. Attempting port 4041.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.3.1
      /_/

Using Python version 2.7.14 (default, Sep 23 2017 22:06:14)
SparkSession available as 'spark'.
>>> 
```

இக்கட்டளை இயங்கி முடிந்த பின்னர் spark shell வெளிப்படுவதைக் காணலாம். இவ்வாறு pyspark-வுடன் mysql.jar ஐ இணைத்து உருவாக்கப்படும் spark shell-ல் நாம் சுலபமாக database-ஐ அணுகி அதற்குள் உள்ள அட்டவணைகளில் இருக்கும் தகவல்களை rdd-ஆக மாற்ற முடியும்.

4. இப்போது mysql-க்குள் சற்று முன் நாம் உருவாக்கிய info அட்டவணையிலுள்ள தரவுகளை spark-க்குள் செலுத்துவதற்கு sqlContext என்பது பயன்படுகிறது. இது ஒரு dataframe-ஐ உருவாக்கி அதற்குள் அட்டவணையில் இருக்கும் தரவுகளை spark-க்குள் செலுத்துகிறது. கீழ்க்கண்ட எடுத்துக்காட்டில் df எனும் பெயர்கொண்ட dataframe உருவாக்கப்பட்டுள்ளது.

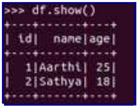
```
>>> df =
sqlContext.read.format("jdbc").op
tions(
url="jdbc:mysql://localhost:3306/
sample",driver =
```

```
"com.mysql.jdbc.Driver",dbtable =
"info",user="nithya",
password="nithya").load()
```

```
>>> df = sqlContext.read.format("jdbc").options( url="jdbc:mysql://localhost:330
6/sample",driver = "com.mysql.jdbc.Driver",dbtable = "info",user="nithya", passw
ord="nithya").load()
2018-06-17 14:18:06 WARN  ObjectStore:6666 - Version information not found in me
tastore. hive.metastore.schema.verification is not enabled so recording the sche
ma version 1.2.0
2018-06-17 14:18:07 WARN  ObjectStore:568 - Failed to get database default, retu
rning NoSuchObjectException
2018-06-17 14:18:12 WARN  ObjectStore:568 - Failed to get database global_temp,
returning NoSuchObjectException
```

show() எனும் function தரவுகளை வெளிக்காட்டுவதற்குப் பயன்படுகிறது.

```
>>> df.show()
```



```
>>> df.show()
+---+------+---+
| id|  name|age|
+---+------+---+
|  1|Aarthi| 25|
|  2|Sathya| 18|
+---+------+---+
```

Dataframe என்பது rdd-ன் பின்புலமாக இயங்கி கணக்கீடுகளை நிகழ்த்த உதவும் ஒன்றாகும். pandas என்பதனை dataframe பயன்படுத்தி இக்கணக்கீடுகளை நிகழ்த்துகிறது. உதாரணத்துக்கு Excel application-ன் பின்புறத்தில் கணக்கீடுகளை நிகழ்த்த உதவும்

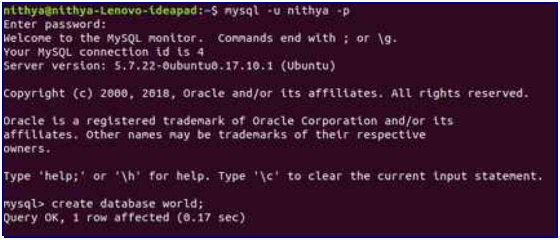
matrix operations போல இவை இரண்டும் spark-ன் பின்புலமாக செயல்படுகின்றன.

10.5 Dataframes-ன் செயல்பாடுகள்

country, countrylanguage எனும் 2 அட்டவணைகளை உள்ளடக்கிய ஒரு sql கோப்பானது ஒரு வலைத்தள முகவரியிலிருந்து கிடைக்கிறது. அதனை பதிவிறக்கம் செய்து இவ்விரண்டு அட்டவணைகளையும் இணைத்து எவ்வாறு நமக்கு வேண்டிய dataframe-ஐ உருவாக்குவது என்று இப்பகுதியில் பார்க்கப் போகிறோம். இதில் உள்ள படிகள் பின்வருமாறு.

1. முதலில் mysql-க்குள் சென்று 'world' எனும் பெயர் கொண்ட ஒரு database-ஐ உருவாக்கவும். இதில்தான் sql கோப்பினை பதிவேற்றம் செய்யப்போகிறோம்.

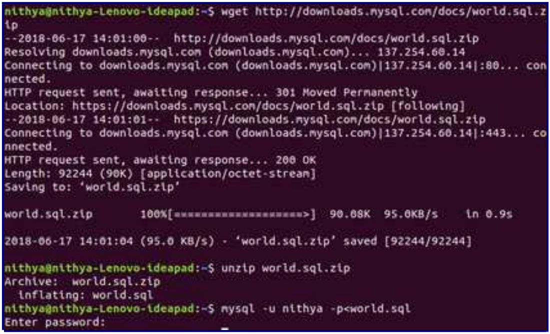
```
$ mysql -u nithya -p
> create database world;
```



```
nithya@nithya-Lenovo-ideapad:~$ mysql -u nithya -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.22-0ubuntu0.17.10.1 (Ubuntu)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database world;
Query OK, 1 row affected (0.17 sec)
```

2. கீழ்க்கண்ட முகவரியிலிருந்து அந்த sql கோப்பினை பதிவிறக்கம் செய்யவும். பின்னர் நாம் உருவாக்கியுள்ள world db-க்குள் கோப்பினை import செய்வதற்கான கட்டளை கொடுக்கப்பட்டுள்ளது.

```
$ wget
http://downloads.mysql.com/docs/w
orld.sql.zip
$ unzip world.sql.zip
$ mysql -u nithya -p<world.sql
```



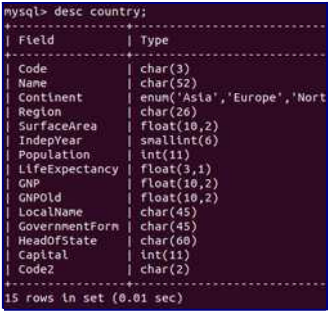
```
nithya@nithya-Lenovo-ideapad:~$ wget http://downloads.mysql.com/docs/world.sql.z
ip
--2018-06-17 14:01:00--  http://downloads.mysql.com/docs/world.sql.zip
Resolving downloads.mysql.com (downloads.mysql.com)... 137.254.60.14
Connecting to downloads.mysql.com (downloads.mysql.com)|137.254.60.14|:80... con
nected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://downloads.mysql.com/docs/world.sql.zip [following]
--2018-06-17 14:01:01--  https://downloads.mysql.com/docs/world.sql.zip
Connecting to downloads.mysql.com (downloads.mysql.com)|137.254.60.14|:443... co
nnected.
HTTP request sent, awaiting response... 200 OK
Length: 92244 (90K) [application/octet-stream]
Saving to: ‘world.sql.zip’

world.sql.zip       100%[===================>]  90.08K  95.0KB/s    in 0.9s

2018-06-17 14:01:04 (95.0 KB/s) - ‘world.sql.zip’ saved [92244/92244]

nithya@nithya-Lenovo-ideapad:~$ unzip world.sql.zip
Archive:  world.sql.zip
  inflating: world.sql
nithya@nithya-Lenovo-ideapad:~$ mysql -u nithya -p<world.sql
Enter password:
```

3. இப்போது mysql-ல் சென்று பார்த்தால் city, country, countrylanguage எனும் 3 அட்டவணைகளும் world db-க்குள்

செலுத்தப்பட்டிருப்பதைக் காணலாம்.

```
mysql> use world;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----------------+
| Tables_in_world |
+-----------------+
| city            |
| country         |
| countrylanguage |
+-----------------+
3 rows in set (0.00 sec)
```

4. country, countrylanguage எனும் அட்டவணைகளில் உள்ள columns-ஐப் பின்வருமாறு காணலாம்.



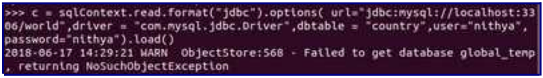
```
mysql> desc country;
+----------------+-------------------------
| Field          | Type
+----------------+-------------------------
| Code           | char(3)
| Name           | char(52)
| Continent      | enum('Asia','Europe','Nort
| Region         | char(26)
| SurfaceArea    | float(10,2)
| IndepYear      | smallint(6)
| Population     | int(11)
| LifeExpectancy | float(3,1)
| GNP            | float(10,2)
| GNPOld         | float(10,2)
| LocalName      | char(45)
| GovernmentForm | char(45)
| HeadOfState    | char(60)
| Capital        | int(11)
| Code2          | char(2)
+----------------+-------------------------
15 rows in set (0.01 sec)
```

```
mysql> desc countrylanguage;
+-------------+---------------+------+-----+---------+-------+
| Field       | Type          | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+-------+
| CountryCode | char(3)       | NO   | PRI |         |       |
| Language    | char(30)      | NO   | PRI |         |       |
| IsOfficial  | enum('T','F') | NO   |     | F       |       |
| Percentage  | float(4,1)    | NO   |     | 0.0     |       |
+-------------+---------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

5. இப்போது c dataframe-க்குள் country-ல் உள்ளவற்றையும், l dataframe-க்குள்

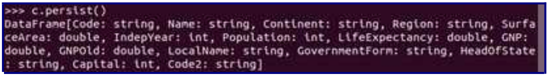
countrylanguage-ல் உள்ளவற்றையும் உட்செலுத்தியுள்ளோம்.

```
>>> c =
sqlContext.read.format("jdbc").op
tions(
url="jdbc:mysql://localhost:3306/
world",driver =
"com.mysql.jdbc.Driver",dbtable =
"country",user="nithya",
password="nithya").load()
```



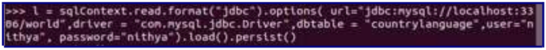
```
>>> c = sqlContext.read.format("jdbc").options( url="jdbc:mysql://localhost:33
06/world",driver = "com.mysql.jdbc.Driver",dbtable = "country",user="nithya",
password="nithya").load()
2018-06-17 14:29:21 WARN  ObjectStore:568 - Failed to get database global_temp
, returning NoSuchObjectException
```

>>> c.persist() (ஒவ்வொரு column-ஐயும் அதனதன் datatype-வுடன் வெளிப்படுத்துகிறது)



```
>>> c.persist()
DataFrame[Code: string, Name: string, Continent: string, Region: string, Surfa
ceArea: double, IndepYear: int, Population: int, LifeExpectancy: double, GNP:
double, GNPOld: double, LocalName: string, GovernmentForm: string, HeadOfState
: string, Capital: int, Code2: string]
```

```
>>> l =
sqlContext.read.format("jdbc").op
tions(
url="jdbc:mysql://localhost:3306/
world",driver =
"com.mysql.jdbc.Driver",dbtable =
"countrylanguage",user="nithya",
password="nithya").load().persist
()
```

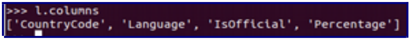


```
>>> l = sqlContext.read.format("jdbc").options( url="jdbc:mysql://localhost:33
06/world",driver = "com.mysql.jdbc.Driver",dbtable = "countrylanguage",user="n
ithya", password="nithya").load().persist()
```

```
>>> c.columns (ஒரு dataframe-ல்
உள்ள columns-ஐ வெளிப்படுத்துகிறது)
```

```
>>> c.columns
['Code', 'Name', 'Continent', 'Region', 'SurfaceArea', 'IndepYear', 'Populatio
n', 'LifeExpectancy', 'GNP', 'GNPOld', 'LocalName', 'GovernmentForm', 'HeadOfS
tate', 'Capital', 'Code2']
```

```
>>> l.columns
```



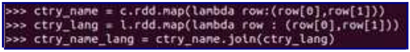
```
>>> l.columns
['CountryCode', 'Language', 'IsOfficial', 'Percentage']
```

6. இப்போது ஒரு நாட்டினுடைய அடையாள எண், அந்நாட்டின் பெயர், அங்கு பேசப்படுகின்ற மொழி இம்மூன்றையும் எடுக்க, மேற்கூறிய 2 dataframe-ஐயும் countrycode-ஆல் இணைக்கலாம். இதிலுள்ள படிகள் பின்வருமாறு.

c frame-ல் உள்ள முதல் இரண்டு columns-ஆன code, name ஆகியவை ctry_name எனும் புதிய dataframe-ல் சேமிக்கப்படுகின்றன. அவ்வாறே l frame-ல் உள்ள முதல் இரண்டு columns-ஆன countrycode, language ஆகியவை

ctry_lang எனும் frame-ல் சேமிக்கப்படுகின்றன. பின்னர் இவ்விரண்டு frame-ஐயும் join() மூலம் இணைத்து ctry_name_lang எனும் ஒரு புதிய frame உருவாக்கப்படுகிறது.

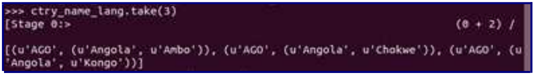
```
>>> ctry_name = c.rdd.map(lambda
row:(row[0],row[1]))
>>> ctry_lang = l.rdd.map(lambda
row : (row[0],row[1]))
>>> ctry_name_lang =
ctry_name.join(ctry_lang)
```



```
>>> ctry_name = c.rdd.map(lambda row:(row[0],row[1]))
>>> ctry_lang = l.rdd.map(lambda row : (row[0],row[1]))
>>> ctry_name_lang = ctry_name.join(ctry_lang)
```

அவ்வாறு உருவான புதிய frame-ன் மீது .take(3) எனக் கொடுத்து முதல் 3 records-ஐ எடுத்து நமக்கு வேண்டியவாறு உள்ளதா என சரிபார்க்கலாம்.

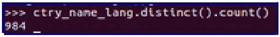
```
>>> ctry_name_lang.take(3)
```



```
>>> ctry_name_lang.take(3)
[Stage 0:>                                                          (0 + 2) /

[(u'AGO', (u'Angola', u'Ambo')), (u'AGO', (u'Angola', u'Chokwe')), (u'AGO', (u
'Angola', u'Kongo'))]
```

அதன் மீதே distinct().count() எனக் கொடுத்து பல்வேறு நாடுகளில் பேசப்படுகின்ற தனித்தனி மொழிகளின் எண்ணிக்கையை எடுக்கலாம்.

```
>>>
ctry_name_lang.distinct().count()
```



```
>>> ctry_name_lang.distinct().count()
984 _
```

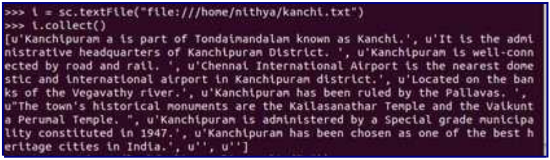
10.6 Text file-ஐ process செய்தல்

ஒரு text file-ல் உள்ளவற்றை rdd-ஆக மாற்றி அதன் மீது செயல்படும் பல்வேறு functions-ஐக் கீழே காணலாம். கீழ்க்கண்ட எடுத்துக்காட்டில் kanchi எனும் கோப்பிற்குள் உள்ளவை i எனும் பெயர் கொண்ட rdd-ல் சேமிக்கப்படுகிறது.

```
>>> i = sc.textFile("file:///home/nithya/kanchi.txt")
```

1. i-ன் மீது செயல்படும் collect() -ஆனது rdd-ல் எவ்வாறு அனைத்து வரிகளும் சேமிக்கப்பட்டுள்ளது என்பதைக் காட்டுகிறது. பொதுவாக rdd என்பது அதனுள் உள்ளவற்றை வரிகளாகப் பிரித்து list வடிவில் வைத்துக்கொள்ளும். அதனுள் உள்ள ஒவ்வொரு வரியையும் list-ன் item-ஆகப் பிரிக்கும். கீழ்க்கண்ட எடுத்துக்காட்டில் ஒவ்வொரு வரியின் துவக்கத்திலும் u’ எனக் காணப்படுவது இதை உணர்த்துகிறது.

```
>>> i.collect()
```

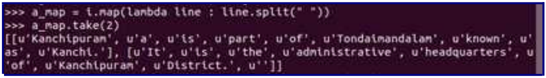


```
>>> i = sc.textFile("file:///home/nithya/kanchi.txt")
>>> i.collect()
[u'Kanchipuram a is part of Tondaimandalam known as Kanchi.', u'It is the admi
nistrative headquarters of Kanchipuram District. ', u'Kanchipuram is well-conn
ected by road and rail. ', u'Chennai International Airport is the nearest dome
stic and international airport in Kanchipuram district.', u'Located on the ban
ks of the Vegavathy river.', u'Kanchipuram has been ruled by the Pallavas. ',
u"The town's historical monuments are the Kailasanathar Temple and the Vaikunt
a Perumal Temple. ", u'Kanchipuram is administered by a Special grade municipa
lity constituted in 1947.', u'Kanchipuram has been chosen as one of the best h
eritage cities in India.', u'', u'']
```

2. i-ன் மீது செயல்படும் map() -ஆனது rdd-ல் உள்ள ஒவ்வொரு வரியையும் அடைப்புக் குறிக்குள் [] பிரித்து, பின்னர் அதிலுள்ள ஒவ்வொரு வார்த்தையையும் இடைவெளிகளை வைத்துப் பிரிக்கிறது. எனவே இதன் வடிவம் [[.,.,.,.],[.,.,.,.],[.,.,.,.]] இவ்வாறு அமைகிறது. கீழ்க்கண்ட எடுத்துக்காட்டில் ஒவ்வொரு வார்த்தையின் துவக்கத்திலும் u' எனக் காணப்பட்டாலும் ஒவ்வொரு வாக்கியமும் தனித்தனி அடைப்புக் குறிக்குள் [ ] அடங்கி, பிறகு மொத்தமாக ஒரு அடைப்புக்

குறிக்குள் அடங்குவதைக் காணலாம். ஆகையால் இது 'iterable of iterables' என்று அழைக்கப்படுகிறது.

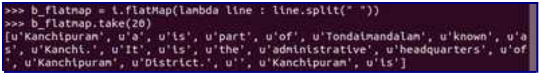
```
>>> a_map = i.map(lambda line :
line.split(" "))
>>> a_map.take(2)
```



```
>>> a_map = i.map(lambda line : line.split(" "))
>>> a_map.take(2)
[[u'Kanchipuram', u'a', u'is', u'part', u'of', u'Tondaimandalam', u'known', u'
as', u'Kanchi.'], [u'It', u'is', u'the', u'administrative', u'headquarters', u
'of', u'Kanchipuram', u'District.', u'']]
```

3. i-ன் மீது செயல்படும் flatmap() -ஆனது rdd-ல் உள்ள ஒவ்வொரு வார்த்தையையும் இடைவெளிகளை வைத்துப் பிரித்து அவற்றை list-ன் item-ஆக சேமிக்கிறது. கீழ்க்கண்ட எடுத்துக்காட்டில் ஒவ்வொரு வார்த்தையின் துவக்கத்திலும் u’ எனக் காணப்படுவது இதை உணர்த்துகிறது. இதன் வடிவம் [.,..,..,..,..,] என்று அமைகிறது. இது 'iterable of strings’ என்று அழைக்கப்படுகிறது.

```
>>> b_flatmap = i.flatMap(lambda
line : line.split(" "))
>>> b_flatmap.take(20)
```

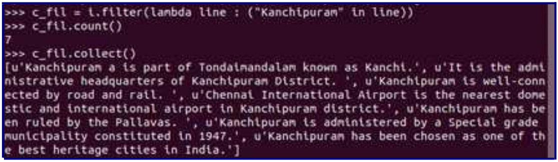


```
>>> b_flatmap = i.flatMap(lambda line : line.split(" "))
>>> b_flatmap.take(20)
[u'Kanchipuram', u'a', u'is', u'part', u'of', u'Tondaimandalam', u'known', u'a
s', u'Kanchi.', u'It', u'is', u'the', u'administrative', u'headquarters', u'of
', u'Kanchipuram', u'District.', u'', u'Kanchipuram', u'is']
```

4. i-ன் மீது செயல்படும் filter() -ஆனது கொடுக்கப்பட்டுள்ள வார்த்தை இடம்பெற்றுள்ள வரிகளை எடுத்து தனியாக வேறொரு rdd-ல் சேமித்துக்கொள்கிறது. அதன் மீது செயல்படும் count() அவ்வார்த்தை எத்தனை முறை இடம்பெற்றுள்ளது என்பதையும், collect() அவ்வார்த்தை இடம்பெற்றுள்ள வரிகளையும் வெளிப்படுத்துகிறது.

```
>>> c_fil = i.filter(lambda
line : ("Kanchipuram" in line))
```

```
>>> c_fil.count()
>>> c_fil.collect()
```

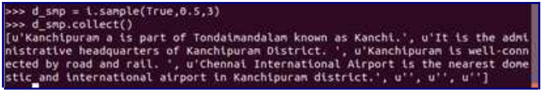


```
>>> c_fil = l.filter(lambda line : ("Kanchipuram" in line))
>>> c_fil.count()
7
>>> c_fil.collect()
[u'Kanchipuram a is part of Tondaimandalam known as Kanchi.', u'It is the admi
nistrative headquarters of Kanchipuram District. ', u'Kanchipuram is well-conn
ected by road and rail. ', u'Chennai International Airport is the nearest dome
stic and international airport in Kanchipuram district.', u'Kanchipuram has be
en ruled by the Pallavas. ', u'Kanchipuram is administered by a Special grade
municipality constituted in 1947.', u'Kanchipuram has been chosen as one of th
e best heritage cities in India.']
```

5. தரவுகளின் எண்ணிக்கை அதிகமாக இருக்கும்போது மாதிரிக்கு ஒருசில வரிகளை எடுக்க விரும்பினால் sample-ஐப்

பயன்படுத்தலாம். இது random-ஆக வரிகளை எடுத்து வெளிப்படுத்தும்.

```
>>> d_smp = i.sample(True,0.5,3)
>>> d_smp.collect()
```



```
>>> d_smp = i.sample(True,0.5,3)
>>> d_smp.collect()
[u'Kanchipuram a is part of Tondaimandalam known as Kanchi.', u'It is the admi
nistrative headquarters of Kanchipuram District. ', u'Kanchipuram is well-conn
ected by road and rail. ', u'Chennai International Airport is the nearest dome
stic and international airport in Kanchipuram district.', u'', u'', u'']
```

6. ஒரு rdd-ன் மீது செயல்படும் getNumPartitions() என்பது rdd எத்தனை பகுதிகளாகப் பிரிக்கப்பட்டுள்ளது என்பதை

வெளிப்படுத்தும். இங்கு rdd-ஐ உருவாக்கும்போதே அது எத்தனை பகுதிகளாகப் பிரிக்க வேண்டும் எனும் எண்ணிக்கை கொடுக்கப்பட்டுள்ளது.

```
>>> j =
sc.textFile("file:///home/nithya/
Kanchi.txt",4)
>>> j.getNumPartitions()
```

```
>>> j = sc.textFile("file:///home/nithya/Kanchi.txt",4)
>>> j.getNumPartitions()
4
```

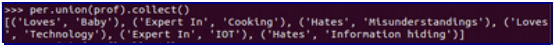
10.7 Union, Join, Intersection

1. ஒரே வடிவம் கொண்ட rdd-ன் மீது செயல்படும் union() function-ஆனது அவை இரண்டையும் மொத்தமாக இணைத்து பின்வருமாறு வெளிப்படுத்தும். இதற்கான கட்டளைகளின் வடிவம் பின்வருமாறு.

```
>>> personal = [("Loves","Baby"),
("Expert In","Cooking"),
("Hates","Misunderstandings")]
>>> professional =
[("Loves","Technology"),("Expert
In","IOT"),("Hates","Information
hiding")]
>>> per =
sc.parallelize(personal)
```

```
>>> prof =
sc.parallelize(professional)
>>> per.union(prof).collect()
```

```
>>> personal = [("Loves","Baby"),("Expert In","Cooking"),("Hates","Misunderstandings")]
>>> professional = [("Loves","Technology"),("Expert In","IOT"),("Hates","Information hiding")]
>>> per = sc.parallelize(personal)
>>> prof = sc.parallelize(professional)
```

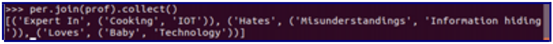


```
>>> per.union(prof).collect()
[('Loves', 'Baby'), ('Expert In', 'Cooking'), ('Hates', 'Misunderstandings'), ('Loves
', 'Technology'), ('Expert In', 'IOT'), ('Hates', 'Information hiding')]
```

2. அதுவே இவை இரண்டும் join() மூலம் இணைக்கப்பட்டால், அவை ஒரு பொதுவான மதிப்பால் இணைக்கப்பட்டு அதனடிப்படையில் இரண்டிலிருந்தும்

தரவுகளை எடுத்து வெளிப்படுத்தும். இது பின்வருமாறு.

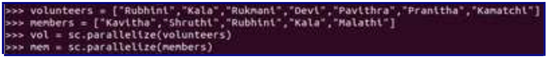
```
>>> per.join(prof).collect()
```



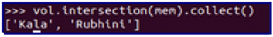
```
>>> per.join(prof).collect()
[('Expert In', ('Cooking', 'IOT')), ('Hates', ('Misunderstandings', 'Information hiding
')), ('Loves', ('Baby', 'Technology'))]
```

3. intersect() மூலம் இரண்டு rdd-ஐ இணைக்கும்போது அவற்றின் வடிவம் வெவ்வேறாக இருந்தாலும், அதிலிருக்கும் பொதுவான தகவல்களை மட்டும் எடுத்து வெளிப்படுத்தும். இது பின்வருமாறு.

```
>>> volunteers =
["Rubhini","Kala","Rukmani","Devi
","Pavithra","Pranitha","Kamatchi
"]
>>> members =
["Kavitha","Shruthi","Rubhini","K
ala","Malathi"]
>>> vol =
sc.parallelize(volunteers)
>>> mem = sc.parallelize(members)
>>>
vol.intersection(mem).collect()
```



```
>>> volunteers = ["Rubhini","Kala","Rukmani","Devi","Pavithra","Pranitha","Kamatchi"]
>>> members = ["Kavitha","Shruthi","Rubhini","Kala","Malathi"]
>>> vol = sc.parallelize(volunteers)
>>> mem = sc.parallelize(members)
```



```
>>> vol.intersection(mem).collect()
['Kala', 'Rubhini']
```

10.8 User defined functions

நாமே நமக்கேற்றவாறு functions-ஐ உருவாக்கி எவ்வாறு ஒரு தனித்தியங்கும் spark application-ஐ உருவாக்குவது என்று இப்பகுதியில் பார்க்கலாம். girls.txt எனும் கோப்பிற்குள் பெண்களின் பெயர்கள், அவர்களின் வயது, அவர்கள் வகிக்கும் பதவி என்பது போன்ற விவரங்களெல்லாம் கொடுக்கப்பட்டுள்ளன. இதை வைத்து மொத்தம் எத்தனை பெண்கள் உள்ளனர், ஒவ்வொரு பதவியிலும் எத்தனை பெண்கள் உள்ளனர், அதில் 30 வயதுக்கு

உட்பட்டவர்களும், 50 வயதுக்கு மேற்பட்டவர்களும் பதவி உயர்வுக்கு தகுதி இல்லாதவர்களாகக் கருதப்பட்டு அதில் எத்தனை பேர் வருகின்றனர் என்பது போன்ற விஷயங்களெல்லாம் user defined functions-ஆக எழுதப்பட்டுள்ளன. இதனை age_grouping.py எனும் கோப்பிற்குள் காணலாம்.

```
001,Nithya,Duraisamy,31,Manager,9587412536,Hyderabad
002,Nandhini,Babu,28,Assistant Manager,9848022338,Delhi
003,Madhuri,Nathan,51,VP,9848022339,Delhi
004,Kavitha,Manoharan,45,AVP,9848022330,Hyderabad
005,Vijaya,Kandasamy,40,AVP,9848022336,Noida
006,Aarthi,Raj,22,Assistant Manager,9848022335,Chennai
007,Lavanya,Sankar,23,Senior Engineer,9848022334,Chennai
008,Meena,Baskar,56,VP,9848022333,Hyderabad
009,Gayathri,Ragu,36,Engineer,9848022333,Chennai
010,Kavitha,Manoharan,49,AVP,9848022336,Noida
```

age_grouping.py

```
from pyspark import SparkContext,
SparkConf

conf =
SparkConf().setAppName('MyFirstSt
andaloneApp')
sc = SparkContext(conf=conf)
fle =
sc.textFile("file:///home/nithya/
girls.txt")

def age_calc(data):

sno,fname,lname,age,desig,mob,loc
ation = data.split(",")
        return
```

```
fname,lname,age_group(int(age)),d
esig,location,int(age)


def  age_group(age):
        if age < 30 :
           return '0-30'
        elif age < 40:
           return '30-40'
        elif age < 50:
           return '40-50'
        elif age < 60:
           return '50-60'
        else :
           return '60+'


age_map = fle.map(age_calc)
freq = age_map.map(lambda line :
```

```
line[3]).countByValue()
print "Total no.of
ladies",fle.count()
print "Total no.of ladies on each
designation",dict(freq)




Under_age = sc.accumulator(0)
Over_age = sc.accumulator(0)

def promotion_check(data):
    global Over_age, Under_age
    age_grp = data[2]
    if(age_grp == "50-60"):
        Over_age +=1
```

```
    if(age_grp == "0-30"):
        Under_age +=1
    return data


df =
age_map.map(promotion_check).coll
ect()
print "Not qualified for
promotions due to under
age",Under_age
print "Not qualified for
promotions due to over
age",Over_age
```

இந்த age_grouping.py கோப்பினை இயக்குவதற்கு முன்னர் spark-ன் conf-க்குள்

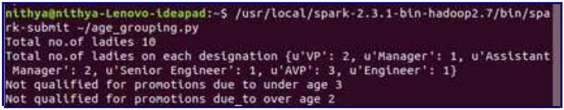
சென்று log4j.properties.template என இருக்கும் கோப்பினை log4j.properties எனப் பெயர் மாற்றம் செய்து, அதில் log4j.logger.org=OFF எனும் பண்பினை சேர்க்கவும். இது பின்வருமாறு.

```
$ sudo cp /usr/local/spark-2.3.1-bin-hadoop2.7/conf/log4j.properties.template /usr/local/spark-2.3.1-bin-hadoop2.7/conf/log4j.properties
$ sudo vim /usr/local/spark-2.3.1-bin-hadoop2.7/conf/log4j.properties
log4j.logger.org=OFF
```

```
# SPARK-9183: Settings to avoid annoying messages when looking up nonexisten
t UDFs in SparkSQL with Hive support
log4j.logger.org.apache.hadoop.hive.metastore.RetryingHMSHandler=FATAL
log4j.logger.org.apache.hadoop.hive.ql.exec.FunctionRegistry=ERROR

log4j.logger.org=OFF
<in-hadoop2.7/conf/log4j.properties" 42L, 2047C          42,1          Bot
```

இப்போது age_grouping.py கோப்பினை இயக்குவதற்கான கட்டளை பின்வருமாறு.

```
$ /usr/local/spark-2.3.1-bin-
hadoop2.7/bin/spark-submit
~/age_grouping.py
```



```
nithya@nithya-Lenovo-ideapad:~$ /usr/local/spark-2.3.1-bin-hadoop2.7/bin/spa
rk-submit ~/age_grouping.py
Total no.of ladies 10
Total no.of ladies on each designation {u'VP': 2, u'Manager': 1, u'Assistant
 Manager': 2, u'Senior Engineer': 1, u'AVP': 3, u'Engineer': 1}
Not qualified for promotions due to under age 3
Not qualified for promotions due_to over age 2
```

இதன் வெளியீடு நாம் எதிர்பார்த்த வண்ணம் உள்ளதைக் காணலாம்.

11 முடிவுரை

இந்த நூலில் Elasticsearch, Logstash, Kibana, Hadoop, Spark அகியவற்றின் அடிப்படைகளை மட்டுமே பார்த்துள்ளோம்.

இன்னும் இந்த நூலில் எழுதப் படாதவை பல. அவற்றை வாசகர்கள் இணையத்தில் தேடி, அறிந்து கொள்ள இந்த நூல் ஆர்வத்தைத் தூண்டும் என நம்புகிறேன்.

பின்வரும் இணைப்புகள் மிகவும் பயனுள்ளதாக இருக்கும்.

https://www.elastic.co/start

https://logz.io/learn/complete-guide-elk-stack/

https://www.digitalocean.com/community/tutorials/an-introduction-to-hadoop

https://www.coursera.org/learn/hadoop

12 ஆசிரியர் பற்றி

து. நித்யா

கணிணி நுட்பங்களை தமிழில் எழுதி வருகிறேன். Tata Consultancy Services நிறுவனத்தில், Internet Of Things துறையில் பணிபுரிய உள்ளேன்.

“தேமதுரத் தமிழோசை உலகெல்லாம்
பரவும் வகை செய்தல் வேண்டும்”
“பிற நாட்டு நல்லறிஞர் சாத்திரங்கள் தமிழ்
மொழியிற் பெயர்த்தல் வேண்டும்”

என்ற பாரதியின் விருப்பங்களை நிறைவேற்றுவதில், என் பங்களிப்பும் உள்ளது என்பதே, மிகவும் மகிழ்ச்சி.

மின்னஞ்சல் -

nithyadurai87@gmai.com

வலைப்பதிவு -

http://nithyashrinivasan.wordpress.com

13 ஆசிரியரின் பிற மின்னூல்கள்

---

http://freetamilebooks.com/authors/nithyaduraisamy/

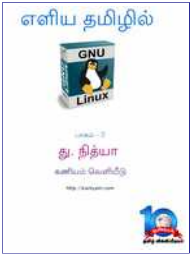
எளிய தமிழில்

GNU Linux

பாகம் - 1

து. நித்யா

கணியம் வெளியீடு



எளிய தமிழில்

GNU Linux

பாகம் - 2

து. நித்யா

கணியம் வெளியீடு

எளிய தமிழில்

MySQL

● து.நித்யா

2


எளிய தமிழில்
MySQL
து. நித்யா
கணியம் வெளியீடு


எளிய தமிழில்
HTML
< து நித்யா >

14 கணியம் பற்றி

இலக்குகள்

- கட்டற்ற கணிநுட்பத்தின் எளிய விஷயங்கள் தொடங்கி அதிநுட்பமான அம்சங்கள் வரை அறிந்திட விழையும் எவருக்கும் தேவையான தகவல்களை தொடர்ச்சியாகத் தரும் தளமாய் உருபெறுவது.

- *உரை, ஒலி, ஒளி என பல்லூடக வகைகளிலும் விவரங்களை தருவது.*
- *இத்துறையின் நிகழ்வுகளை எடுத்துரைப்பது.*
- *எவரும் பங்களிக்க ஏதுவாய் யாவருக்குமான நெறியில் விவரங்களை வழங்குவது.*
- *அச்சு வடிவிலும், புத்தகங்களாகவும், வட்டுக்களாகவும் விவரங்களை வெளியிடுவது.*

பங்களிக்க

- *விருப்பமுள்ள எவரும் பங்களிக்கலாம்.*

- *கட்டற்ற கணிநுட்பம் சார்ந்த விஷயமாக இருத்தல் வேண்டும்.*

- *பங்களிக்கத் தொடங்கும் முன்னர் கணியத்திற்கு உங்களுடைய பதிப்புரிமத்தை அளிக்க எதிர்பார்க்கப்படுகிறீர்கள்.*

- **editor@kaniyam.com**

  முகவரிக்கு கீழ்க்கண்ட

  விவரங்களடங்கிய மடலொன்றை

உறுதிமொழியாய் அளித்துவிட்டு

யாரும் பங்களிக்கத் தொடங்கலாம்.

- **மடலின் பொருள்:**

  பதிப்புரிமம் அளிப்பு

- **மடல் உள்ளடக்கம்**

  - *என்னால் கணியத்திற்காக அனுப்பப்படும் படைப்புகள் அனைத்தும் கணியத்திற்காக முதன்முதலாய்*

படைக்கப்பட்டதாக உறுதியளிக்கிறேன்.

- இதன்பொருட்டு எனக்கிருக்கக்கூடிய பதிப்புரிமத்தினை கணியத்திற்கு வழங்குகிறேன்.

- உங்களுடைய முழுப்பெயர், தேதி.

- தாங்கள் பங்களிக்க விரும்பும் ஒரு பகுதியில் வேறொருவர் ஏற்கனவே பங்களித்து வருகிறார் எனின் அவருடன் இணைந்து பணியாற்ற முனையவும்.

- கட்டுரைகள் மொழிபெயர்ப்புகளாகவும், விஷயமறிந்த ஒருவர் சொல்லக் கேட்டு கற்று இயற்றப்பட்டவையாகவும் இருக்கலாம்.

- படைப்புகள் தொடர்களாகவும் இருக்கலாம்.

- தொழில் நுட்பம், கொள்கை விளக்கம், பிரச்சாரம், கதை, கேலிச்சித்திரம், நையாண்டி எனப் பலசுவைகளிலும் இத்துறைக்கு பொருந்தும்படியான ஆக்கங்களாக இருக்கலாம்.

- தங்களுக்கு இயல்பான எந்தவொரு நடையிலும் எழுதலாம்.

- *தங்களது படைப்புகளை எளியதொரு உரை ஆவணமாக* editor@kaniyam.com *முகவரிக்குஅனுப்பிவைக்கவும்*.

- *தள பராமரிப்பு, ஆதரவளித்தல் உள்ளிட்ட ஏனைய விதங்களிலும் பங்களிக்கலாம்*.

- *ஐயங்களிருப்பின்* editor@kaniyam.com *மடலியற்றவும்*.

விண்ணப்பங்கள்

- *கணித் தொழில்நுட்பத்தை அறிய விழையும் மக்களுக்காக*

மேற்கொள்ளப்படும் முயற்சியாகும் இது.

- இதில் பங்களிக்க தாங்கள் அதிநுட்ப ஆற்றல் வாய்ந்தவராக இருக்க வேண்டும் என்ற கட்டாயமில்லை.

- தங்களுக்கு தெரிந்த விஷயத்தை இயன்ற எளிய முறையில் எடுத்துரைக்க ஆர்வம் இருந்தால் போதும்.

- இதன் வளர்ச்சி நம் ஒவ்வொருவரின் கையிலுமே உள்ளது.

- குறைகளிலிருப்பின் முறையாக தெரியப்படுத்தி முன்னேற்றத்திற்கு வழி வகுக்கவும்.

வெளியீட்டு விவரம்

ஆசிரியர்: த. சீனிவாசன் – editor@kaniyam.com +91 98417 95468

15 நன்கொடை

Creative Commons உரிமையில், யாவரும் இலவசமாகப் பகிரும் வகையில் தமது நூல்களை வெளியிடும் எழுத்தாளரை உங்கள் நன்கொடைகள் ஊக்குவிக்கும்.

வங்கி விவரங்கள்.

Name - Nithya Duraisamy

CITI Bank A/C Number -

5409095448

Branch - Chennai

IFSC code - CITI0000003