

UNIVERSITY PARTNER



BIG DATA (6CS030)

Portfolio Assessment

Student Id : 1928449|1928579
Student Name : Shristi Shakya | Shreejan Shrestha
Group : C3G1
Supervisor : Mr. Rupak Koirala | Jnaneshwar Bohara
Cohort : 3
Submitted on : 2020/05/30
Word Count :1500

Table of Contents

1	Introduction to Big data	1
2	Introduction to Datasets:	1
2.1	Justification of datasets:	1
2.2	CSV Datasets:	2
2.3	Json Datasets:	2
3	Import/ Cleaning of data	2
3.1.1	Issues with the datasets	2
3.2	Discuss which of the 3 databases were appropriate for each dataset and why.	4
4	Analysis of the data and visualizations	5
4.1	Discuss what techniques can be used to query the data. Show the results of the investigation.	5
4.1.1	Oracle	5
4.1.2	Mongo Db	11
4.1.3	Hadoop	17
5	Comparison table	22
5.1	Advantages	22
5.1.1	Oracle:	22
5.1.2	Mango DB:	22
5.1.3	Hadoop:	22
5.2	Disadvantages	23
5.2.1	Oracle:	23
5.2.2	Mango DB:	23
5.2.3	Hadoop:	23
6	Conclusions and recommendations:	24
7	Appendix	25
7.1	Cleaning the data	25
7.1.1	Document thoroughly what changes were made to the data and why they are carried out 25	
7.2	Analysis of the data	29
7.2.1	Oracle	29
7.2.2	Mongo DB	35
7.2.3	Hadoop Mapreduce:	42
7.3	Visualizations:	47

7.3.1	Scatter plot analysis of csv data	50
8	References	51

Table of Figures

Figure 1	Original and Modified Column name.....	2
Figure 2:	Showing the null value in each column	3
Figure 3:	Showing the Sig value less than 0.5.	3
Figure 4	Removing missing value and rechecking the values.....	4
Figure 5:	Total cases and deaths in 3 years	5
Figure 6:	Total cases and deaths Comparison	5
Figure 7	Confirmed_cases and Death is each country in 3 years	6
Figure 8:	Using Roll-up	6
Figure 9:	Cases and deaths with respect to countries.....	7
Figure 10	Cases and death in 2014	7
Figure 11	Cases and death in 2015	8
Figure 12	Cases and death in 2016	8
Figure 13:	Cases and deaths between 2014-16.....	8
Figure 14	Cases and death in 2014	9
Figure 15	Cases and death in 2015	9
Figure 16	Cases and death in 2016	10
Figure 17:	Visual Analysis of the confirmed case and death	10
Figure 18:	Pie-chart showing most affected gender	11
Figure 19:	Violin plot visualizing settlement area suffered by Ebola	12
Figure 20:	Bar graph of most affected age group.....	13
Figure 21:	Visualization of adult group suffering from Ebola.....	14
Figure 22:	Visualization of male and female with respect to age-group	14
Figure 23:	Visualization of Average age on basis of gender	15
Figure 24:	Over all visual representation of data used in MongoDB	16
Figure 25:	Final MapReduce command Result	17
Figure 26:	Visualization of cases and deaths with respect to countries	18
Figure 27:	Over all visual representation of CSV data set	19
Figure 28	Importing csv dataset into pySpark	20
Figure 29	Death count in Rural and Urban area	21
Figure 30	Death on the basis of gender.....	21
Figure 31	Death on basis of age-group.....	21
Figure 32	Analysis of missing value using IBM SPSS software.....	27
Figure 33 :	Showing Sig value less than 0.5	27
Figure 34 :	Dropping the missing data.....	28
Figure 35	Saving the cleaned dataset	28

Figure 36 Final MapReduce command	42
Figure 37 Output of mapreduce in putty.....	42
Figure 38 Error in Json file	45
Figure 39 Death on the basic of settlement area	46
Figure 40 death count on the basis of gender	46
Figure 41 death on the basis of age-group	46
Figure 42 :Pivot table option from Insert is selected	47
Figure 43:Ok option selected Figure 44:Selected Preferences.....	47
Figure 45:From Chart, Pie chart option selected.....	48
Figure 46: Imported python visualization libraries.....	48
Figure 47:Relplot example	49
Figure 48:Scatter plot analysis of confirmed cases and deaths	50

Peer review

Shristi Shakya 1928449	Shreejan Shrestha 1928579
Analyzing of Mongo DB	Analyzing of Oracle and Hadoop
Introduction to database and datasets	Cleaning of data
Visualizations of analysis	Analysis of data
Conclusion and recommendation	Comparision table
Appendix	

1 Introduction to Big data

Big data can be described as huge collection of data which grows exponentially with the rate of time (Beal, 2020). The term Big Data was initially coined by **Roger Mougalias** from O'Reilly Media in 2005. Though the term Big Data was introduced in 2005, the main concept was introduced back from 7,000 years in Mesopotamia for accounting. In the same year i.e. 2005, Yahoo created Hadoop that is built on the top of Google's MapReduce (Rijmenam, 2013). The rapid use of internet in today's digital world and development of computational science has enabled in the generation of the massive data sets leading to form Big Data. This data is generated and are used in many fields to develop predictive model, recommendation system and also utilized as the aid in many fields such as education, communication and is in use for prediction of diseases and epidemic. (Lee & Yoon, 2017).

2 Introduction to Datasets:

The datasets that have been implemented and analyzed in this project are related to **“Ebola (EVD)”** an infectious and deadly disease that was first appeared in 1976 but it spread like a wildfire in 2014-2016 in West African countries such Guinea, Sierra Leone, Spain, Nigeria, United States, etc. There are two datasets: CSV and json which are both taken from Kaggle. Because of less active of media and sites back then, compare to Covid-19, Ebola and its consequences was not highlighted enough. The aim for taking this dataset is to analyze the affect, cases and human casualties of Ebola.

2.1 Justification of datasets:

From the given datasets we will be analyzing various components. This report will contain the analysis of confirmed cases and confirmed death due to Ebola in various countries in between the time span of 2014-2016. We will also be analyzing gender and settlement wise cases in Nigeria in 2016. The aim for taking this dataset is to analysis the total number of cases and human casualty caused by Ebola in the time spam of 3 years.

2.2 CSV Datasets:

We have taken the csv datasets from Kaggle which consists of 1649 records. This dataset consists of 10 attributes such as country, date, confirmed_cases, etc. As the dataset consist of cases and deaths occurred in different countries at different time period, we will analyze total number of confirmed_cases and confirmed_deaths on the basis of year and country.

2.3 Json Datasets:

The json file has also been taken from Kaggle. The JSON file contains the cases of Ebola of 2016 in Nigeria. The file contains 2797 data with 8 attributes such as gender, settlement, age. From this dataset, we have analyzed the total number of populations affected on the basis of gender, settlement area and age group.

3 Import/ Cleaning of data

3.1.1 Issues with the datasets

Modification of each column_name is made by renaming and eliminating spaces and various characters in between the columns name like “!”, “#”, “-” etc.

Original

Attribute:

	Country	Date	No. of suspected cases	No. of probable cases	No. of confirmed cases	No. of confirmed, probable and suspected cases	No. of suspected deaths	No. of probable deaths	No. of confirmed deaths	No. of confirmed, probable and suspected deaths
0	Guinea	8/29/2014	25.0	141.0	482.0	648.0	2.0	141.0	287.0	430
1	Nigeria	8/29/2014	3.0	1.0	15.0	19.0	0.0	1.0	6.0	7

Modified Attribute:

```
In [4]: header=['Country','Date','Suspected_cases','Probable_cases','Confirmed_cases','Total_cases','Suspected_deaths','Probable_deaths','Confirmed_deaths','Total_deaths']
        pd.read_csv('ebola_2014_2016.csv',header=0,names=new_header,skiprows=1)
        .head(10)
```

Out[4]:

	Country	Date	Suspected_cases	Probable_cases	Confirmed_cases	Total_cases	Suspected_deaths	Probable_deaths	Confirmed_deaths	Total_deaths
0	Nigeria	8/29/2014	3.0	1.0	15.0	19.0	0.0	1.0	6.0	7
1	Sierra Leone	8/29/2014	54.0	37.0	935.0	1026.0	8.0	34.0	380.0	422
2	Liberia	8/29/2014	382.0	874.0	322.0	1378.0	188.0	301.0	225.0	694

Figure 1 Original and Modified Column name

Missing values in the dataset were checked using `.isnull().sum()` function.

```
In [9]: data.isnull().sum()
Out[9]: Country          0
Date          0
Suspected_cases    119
Probable_cases     49
Confirmed_cases     1
Total_cases        8
Suspected_deaths   1177
Probable_deaths    959
Confirmed_deaths    837
Total_deaths       0
dtype: int64
```

Figure 2: Showing the null value in each column

To treat the missing value, first, the dataset was checked whether it is Missing Completely at Random (MCAR) or Not Missing at Random (NMAR). To know this, IBM SPSS Statistics software's expectation-maximization (EM) technique has been used.

EM Estimated Statistics

EM Means^a

Confirmed_cases	Confirmed_deaths
1437.00	670.08

a. Little's MCAR test: Chi-Square = 54.049, DF = 1, Sig. = .000

Figure 3: Showing the Sig value less than 0.5.

After analyzing, Sig value is less than 0.5 in EM technique, it was found the data is MCAR, that is why, missing values were removed from the dataset. Pandas' dropna function has been used to drop the missing value.

```
data.dropna(subset=['Confirmed_cases','Confirmed_deaths'],how='any', inplace=True)

data.Confirmed_cases.isnull().sum()

0

data.Confirmed_deaths.isnull().sum()

0
```

Figure 4 Removing missing value and rechecking the values

Detail explanation of cleaning of data on Appendix 7.1

3.2 Discuss which of the 3 databases were appropriate for each dataset and why.

1. Oracle is relational database management system which stores the data in tables/ scheme and uses structured query language (SQL) to access the data. As our csv dataset is in structured and predefined tabular format, oracle is the best DBMS to handle the csv dataset.
2. MongoDB promotes No SQL. In MongoDB, we do not need predefined structure and has the ability to handle semi-structured data. As json is in semi-structured format with no particular format or schema, we used MongoDB for json data set.
3. As in oracle, data representation must be in tabular format, using json which has no predefined structure would face difficulties. As to import data in oracle, all data must have same column size and in json file format, some data can have absence of detail of an attribute. Because of the reason, we did not use Json in oracle.
4. We have also used Hadoop for csv and json data but for now, we found oracle better than Hadoop, as we have small dataset i.e. 1648, importing data and using SQL queries became easier than using separate java file for defining different MapReduce function. Hadoop would be best if we had large data size as Hadoop have the best ability to handle data in large format.
5. Major disadvantage of MongoDB was to visualize the analysis

4 Analysis of the data and visualizations

4.1 Discuss what techniques can be used to query the data. Show the results of the investigation.

4.1.1 Oracle




1. Total of 2812955 people got infected and Total of 1288179 people lost their life because of Ebola in 3 years.

Worksheet

Query Builder

```
SELECT SUM(confirmed_cases), SUM(confirmed_deaths)
from Ebola;
```

Query... x

   SQL | All Rows Fetched: 1 in 0.218 seconds

	SUM(CONFIRMED_CASES)	SUM(CONFIRMED_DEATHS)
1	2812955	1288179

Figure 5: Total cases and deaths in 3 years

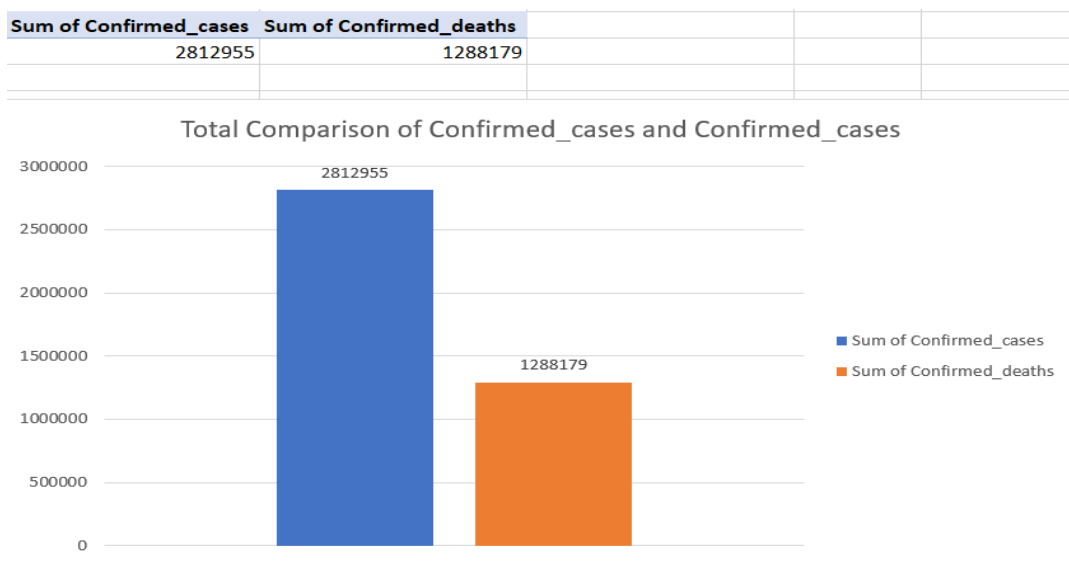


Figure 6: Total cases and deaths Comparison

Note: Scatter plot analysis on Appendix [7.3.1]

- Using SUM () function and GROUPBY clause, we find out, country Sierra Leone was affected the most most with total confirmed cases of 1921048 and death of 768620 whereas Italy and UK were least affected with no casualty in 3 years.

```
SELECT SUM(confirmed_cases), SUM(confirmed_deaths),country
from ebola
GROUP BY country;
```

	SUM(CONFIRMED_CASES)	SUM(CONFIRMED_DEATHS)	COUNTRY
1	713850	441036	Guinea
2	41085	29050	Liberia
3	35288	12946	Spain
4	1	0	Italy
5	1921048	768620	Sierra Leone
6	14194	6875	Mali
7	32517	13071	United States of America
8	2	0	United Kingdom
9	33440	9621	Senegal
10	21530	6960	Nigeria

Figure 7 Confirmed_cases and Death is each country in 3 years

```
SELECT SUM(confirmed_cases), SUM(confirmed_deaths),country,edate
from ebola
GROUP BY ROLLUP (country,edate);
```

	SUM(CONFIRMED_CASES)	SUM(CONFIRMED_DEATHS)	COUNTRY	EDATE
1	3001	940	Mali	17-OCT-14
2	5501	2945	Mali	22-OCT-14
3	5692	2990	Mali	25-OCT-14
4	14194	6875	Mali	(null)
5	1	0	Italy	15-MAY-15
6	1	0	Italy	(null)

Figure 8:Using Roll-up

Results are same, if we compare Mali and Italy.

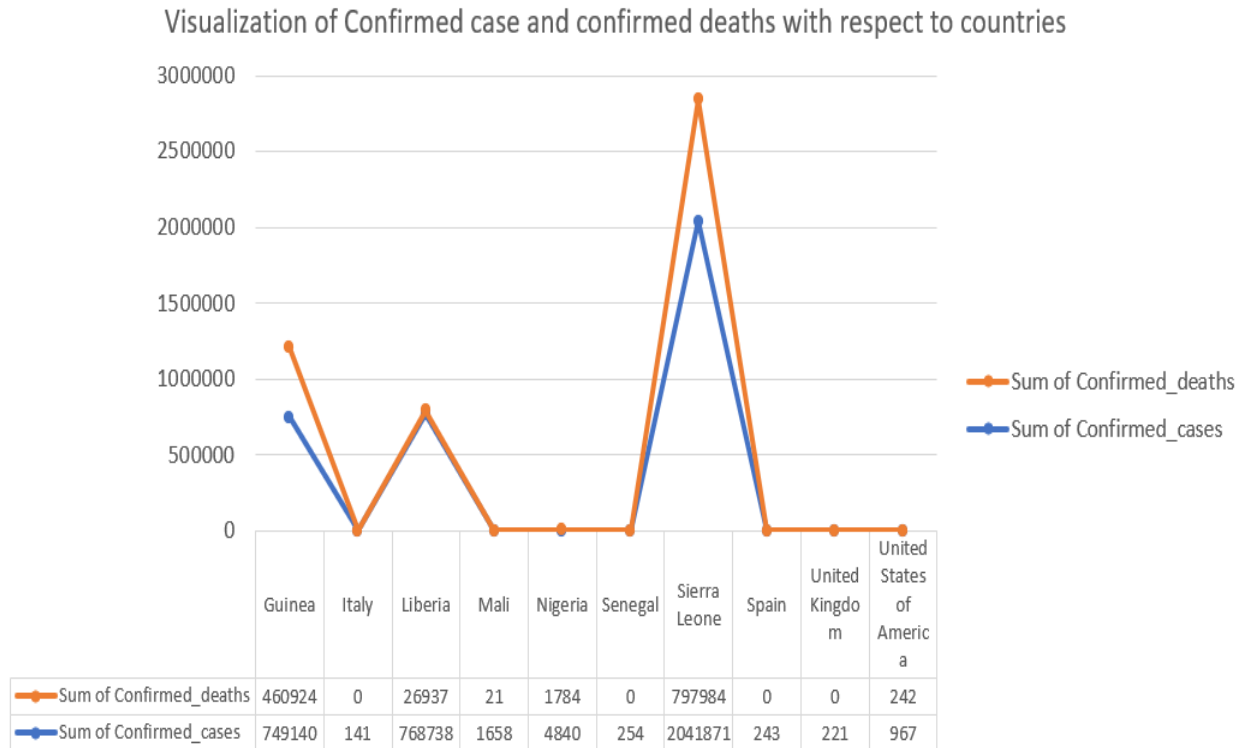


Figure 9: Cases and deaths with respect to countries

- Using SUM () function, WildCards (), LIKE Function and Where clause, we analyzed in which year cases and death count were high and find out year 2015 had the most cases and death rate in all 3 years.

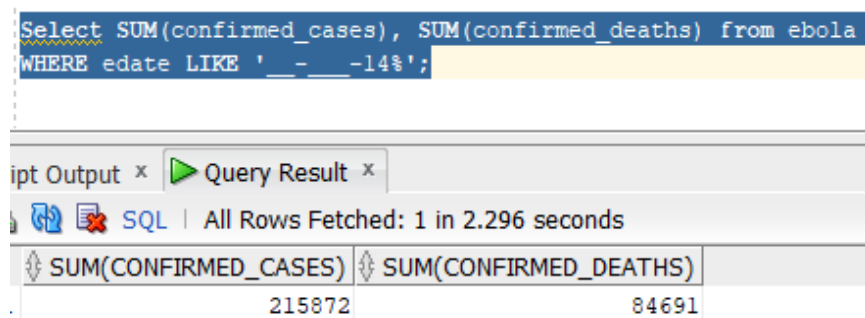


Figure 10 Cases and death in 2014

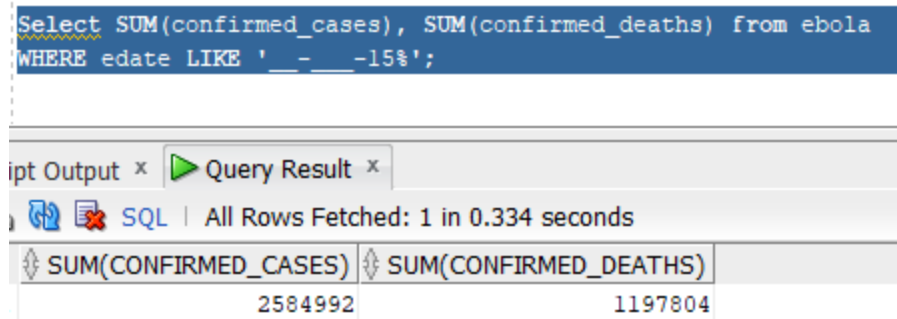


Figure 11 Cases and death in 2015

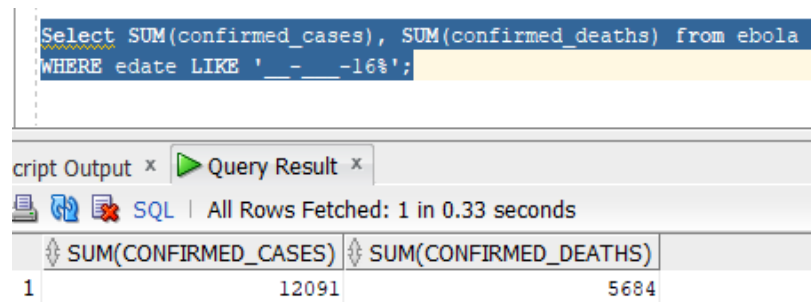


Figure 12 Cases and death in 2016

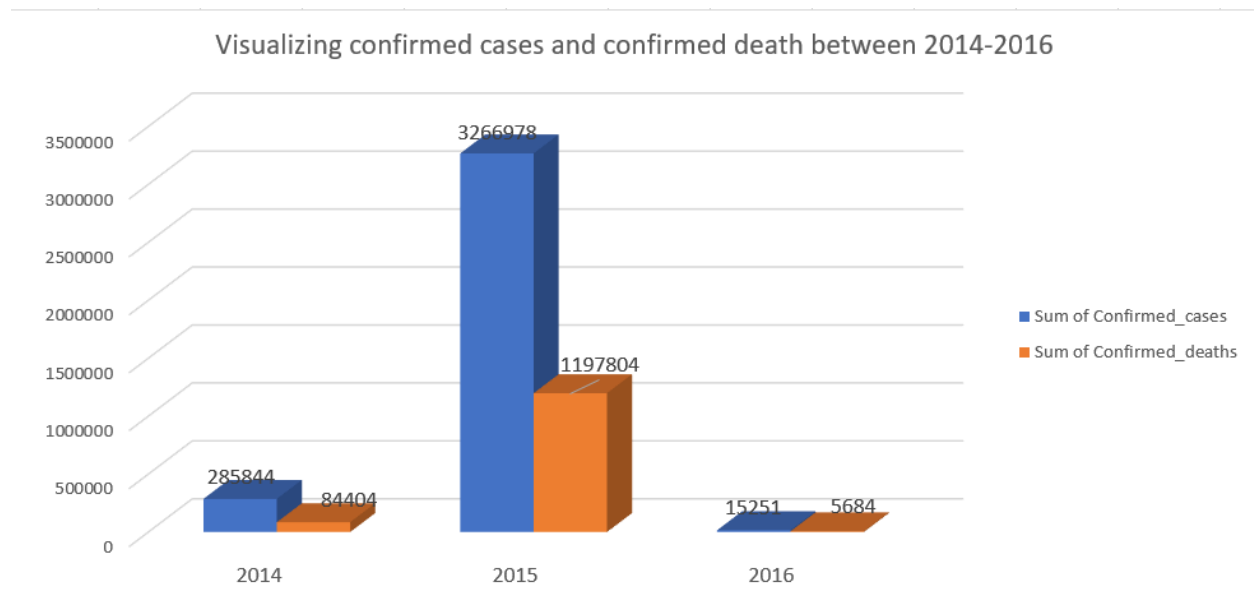


Figure 13: Cases and deaths between 2014-16

- Using SUM(), Wildcards, LIKE Function, GROUPBY clause, we analyzed cases and death count of each country in each year.

```
SELECT SUM(confirmed_cases), SUM(confirmed_deaths),country from ebola
where edate LIKE '__-__-14%'
GROUP BY country;
```

SUM(CONFIRMED_CASES)	SUM(CONFIRMED_DEATHS)	COUNTRY
20523	10839	Guinea
27841	13498	Liberia
35071	12946	Spain
14194	6875	Mali
35964	12617	Sierra Leone
31649	12854	United States of America
17407	5441	Nigeria
33223	9621	Senegal

Figure 14 Cases and death in 2014

In 2014, Spain have the highest number of infected people but total death number of Liberia is highest. Nigeria has the least number of death cases.

```
SELECT SUM(confirmed_cases), SUM(confirmed_deaths),country from ebola
where edate LIKE '__-__-15%'
GROUP BY country;
```

SUM(CONFIRMED_CASES)	SUM(CONFIRMED_DEATHS)	COUNTRY
689976	428114	Guinea
13233	15548	Liberia
216	0	Spain
1	0	Italy
1876380	752414	Sierra Leone
864	216	United States of America
2	0	United Kingdom
216	0	Senegal
4104	1512	Nigeria

Figure 15 Cases and death in 2015

In 2015, infected population and total death number of Sierra Leone is at highest and Italy have only 1 case with no human casualty.

```
SELECT SUM(confirmed_cases), SUM(confirmed_deaths),country from ebola
where edate LIKE '___-__-16%'
GROUP BY country;
```

	SUM(CONFIRMED_CASES)	SUM(CONFIRMED_DEATHS)	COUNTRY
1	3351	2083	Guinea
2	11	4	Liberia
3	1	0	Spain
4	8704	3589	Sierra Leone
5	4	1	United States of America
6	1	0	Senegal
7	19	7	Nigeria

Figure 16 Cases and death in 2016

In 2016, cases were reduced drastically. Sierra Leone had high cases and death count followed by Guinea whereas other nations were recovering well.

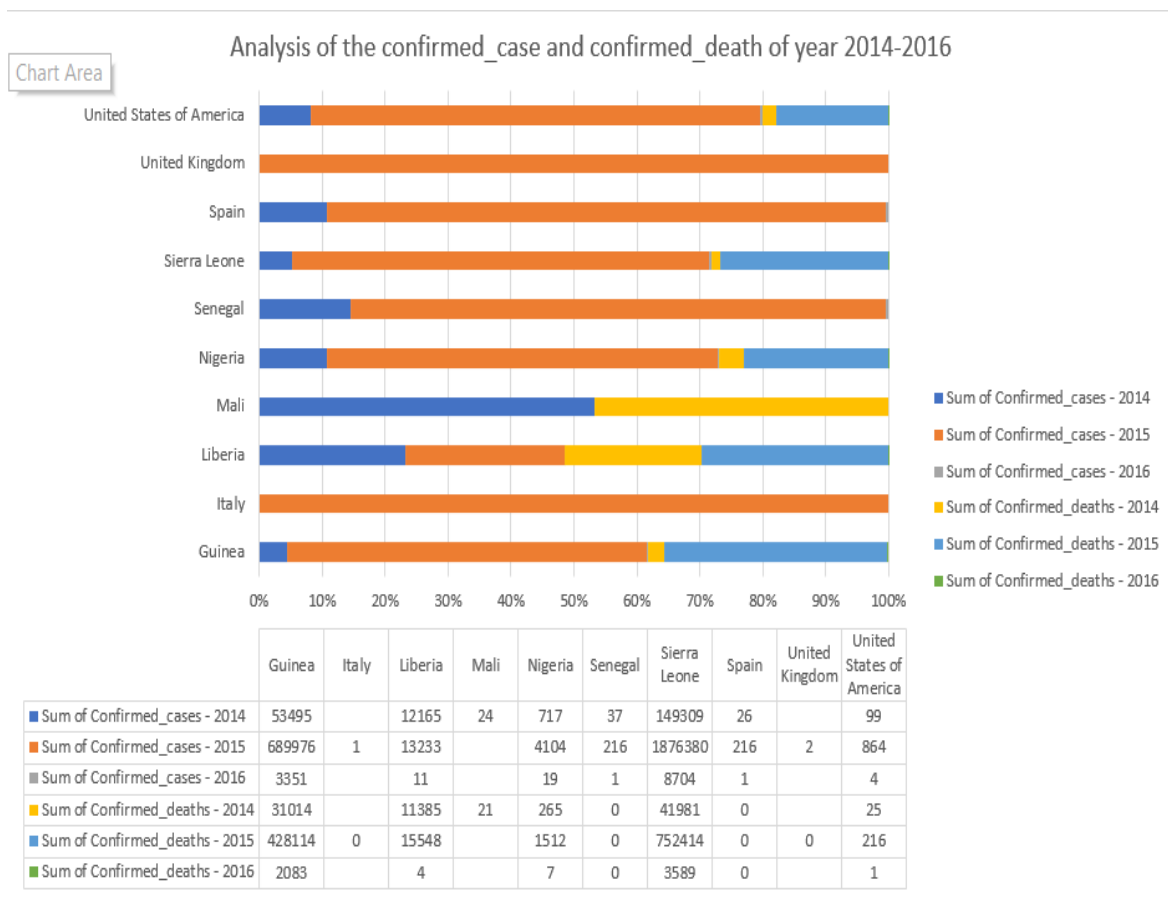


Figure 17: Visual Analysis of the confirmed case and death

Note: Detail explanation on Appendix 7.2.1

4.1.2 Mongo Db

1. Analyzing infected population on the basis of gender.

Note: over all code description in [**Appendix7.2.2**]

```
> db.bigcw.find().count()
2796
> db.bigcw.count({gender:"Male"})
1321
> db.bigcw.count({gender:"Female"})
1475
>
```

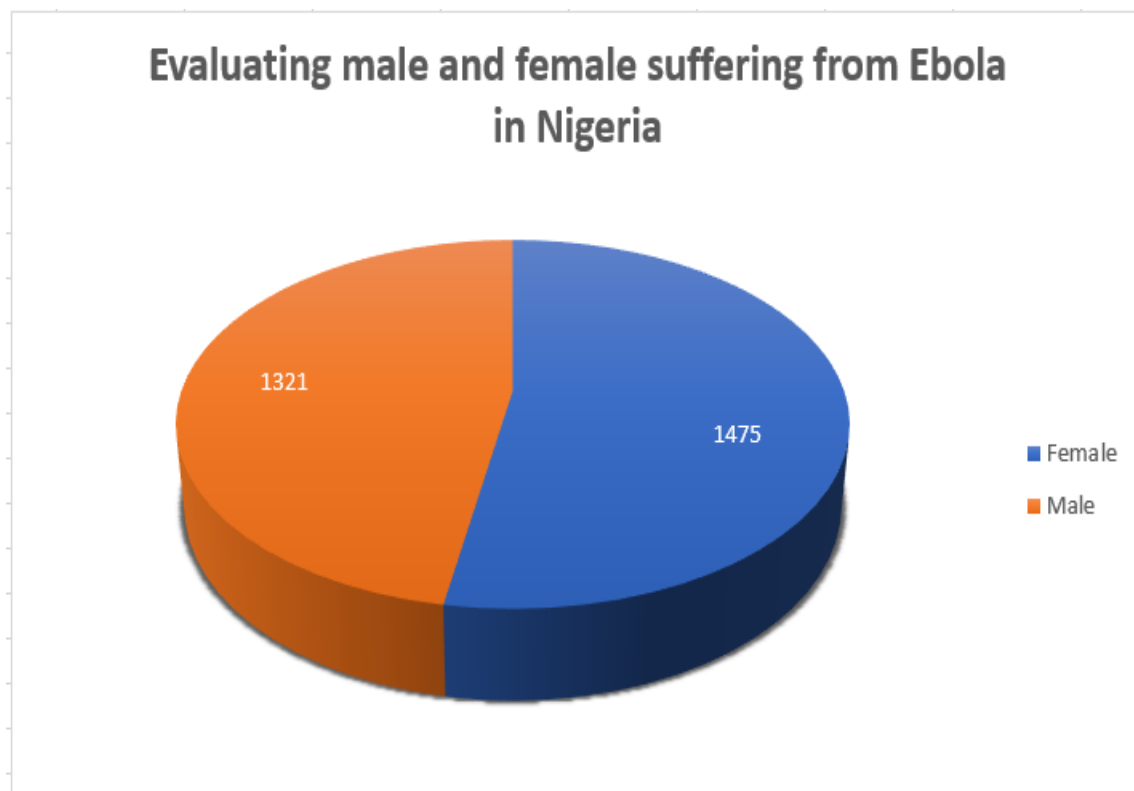


Figure 18: Pie-chart showing most affected gender

2. Analyzing infected population on the basis of settlement area:

```
> db.bigcw.count({settlement:"Rural"})
1446
> db.bigcw.count({settlement:"Urban"})
1350
>
```

Note: Further Description of Visualization in **Appendix 7.3**

```
import seaborn as sns
plt.title('Evaluating affected settlement area')
sns.violinplot(ebola2['age'], ebola2['settlement']) #Variable Plot
sns.despine()
```

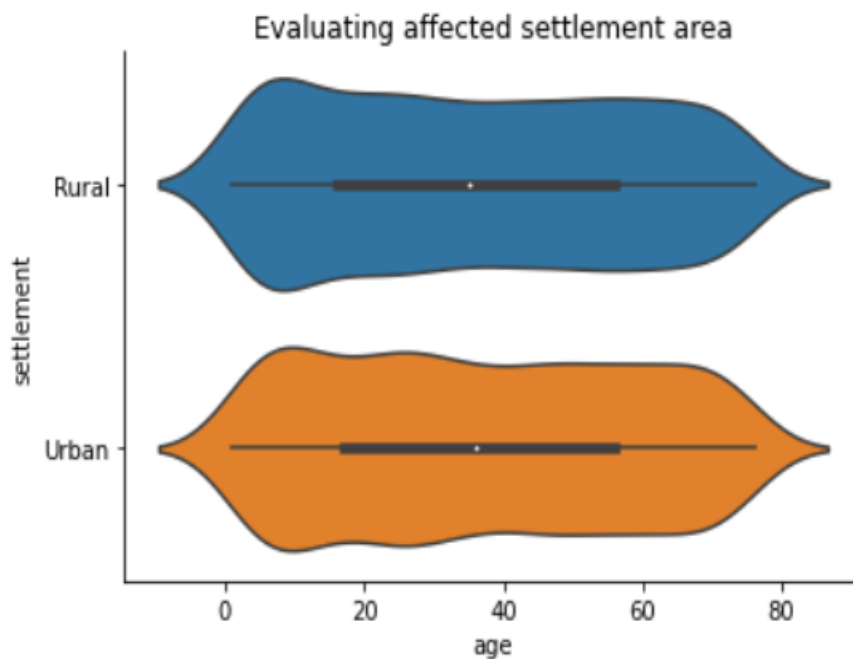


Figure 19: Violin plot visualizing settlement area suffered by Ebola

3. It has found that adults were affected the most than child. Total of 741 children were affected, 2055 adults were affected by Ebola. In both age group female seems to get more infected than male.

```
> db.bigcw.aggregate(
... [
... {$match:{}},
... {$group:{_id:"$gender",total:{$sum:"$adult_group"} } }
... ]
... )
{ "_id" : "Male", "total" : 969 }
{ "_id" : "Female", "total" : 1086 }
>
```

```
> db.bigcw.aggregate(
... [
... {$match:{}},
... {$group:{_id:"$gender",total:{$sum:"$child_group"} } }
... ]
... )
{ "_id" : "Male", "total" : 352 }
{ "_id" : "Female", "total" : 389 }
>
```

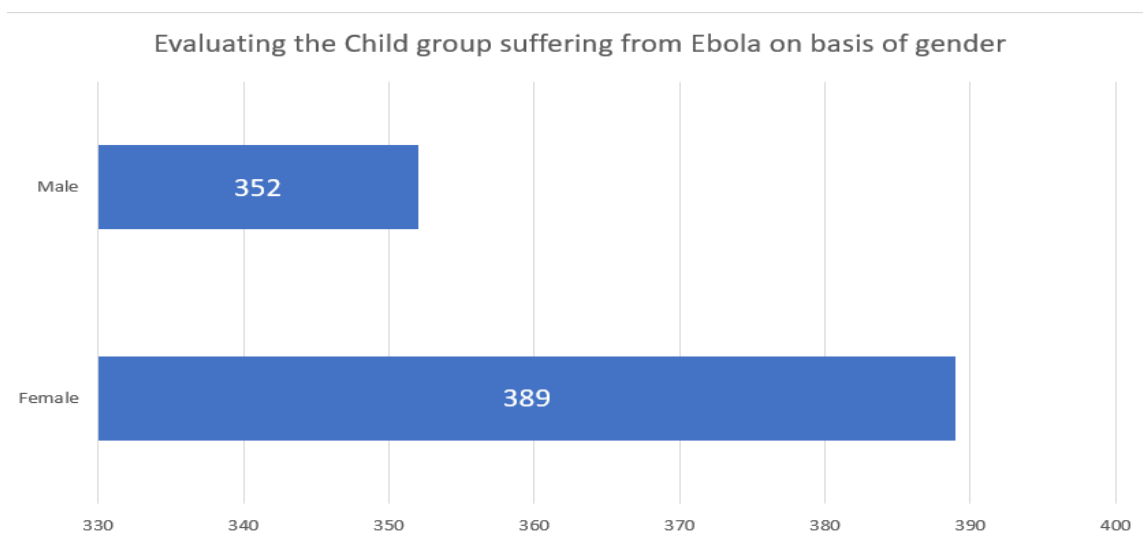


Figure 20: Bar graph of most affected age group

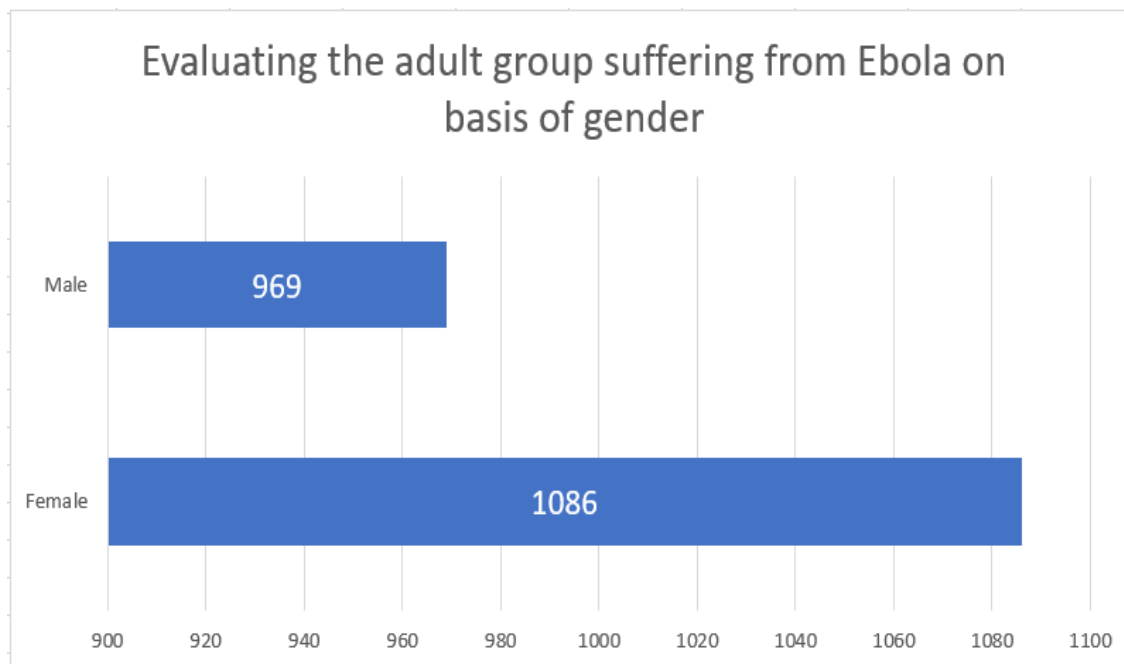


Figure 21: Visualization of adult group suffering from Ebola

```
sns.relplot(x='gender', y='age', hue='age', size='child_group', col='groups', data=ebola3)
```

<seaborn.axisgrid.FacetGrid at 0x22f9f9154c0>

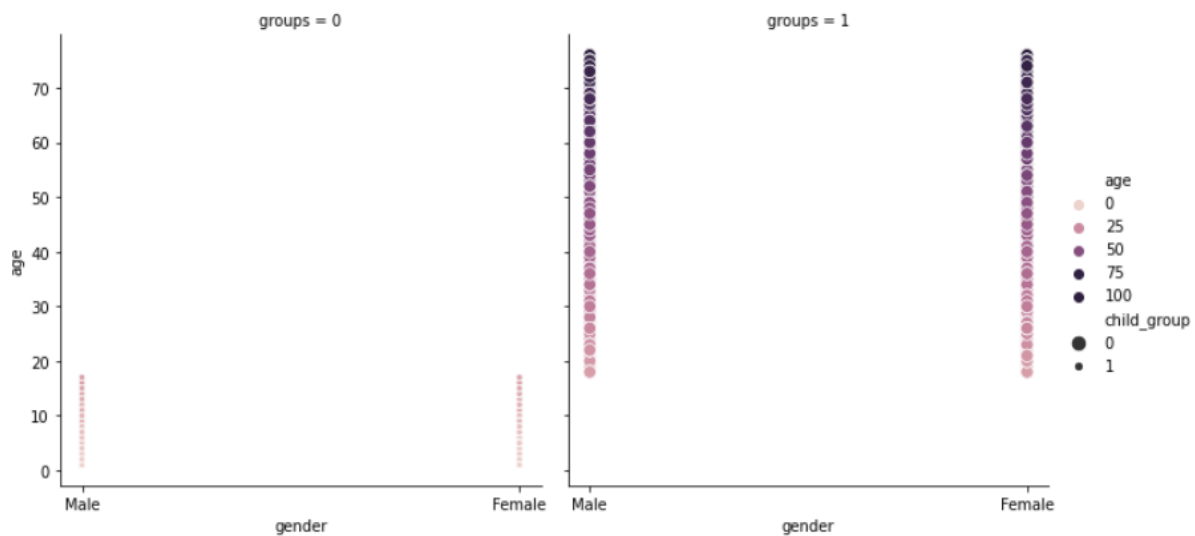


Figure 22: Visualization of male and female with respect to age-group

4. Average infected age of both male and female are 36.

```
> db.bigcw.aggregate(  
... [  
... {$group:{_id:"$gender",total:{$avg:"$age"} } }  
... ]  
... )  
{ "_id" : "Male", "total" : 36.25662376987131 }  
{ "_id" : "Female", "total" : 36.53423728813559 }  
>
```

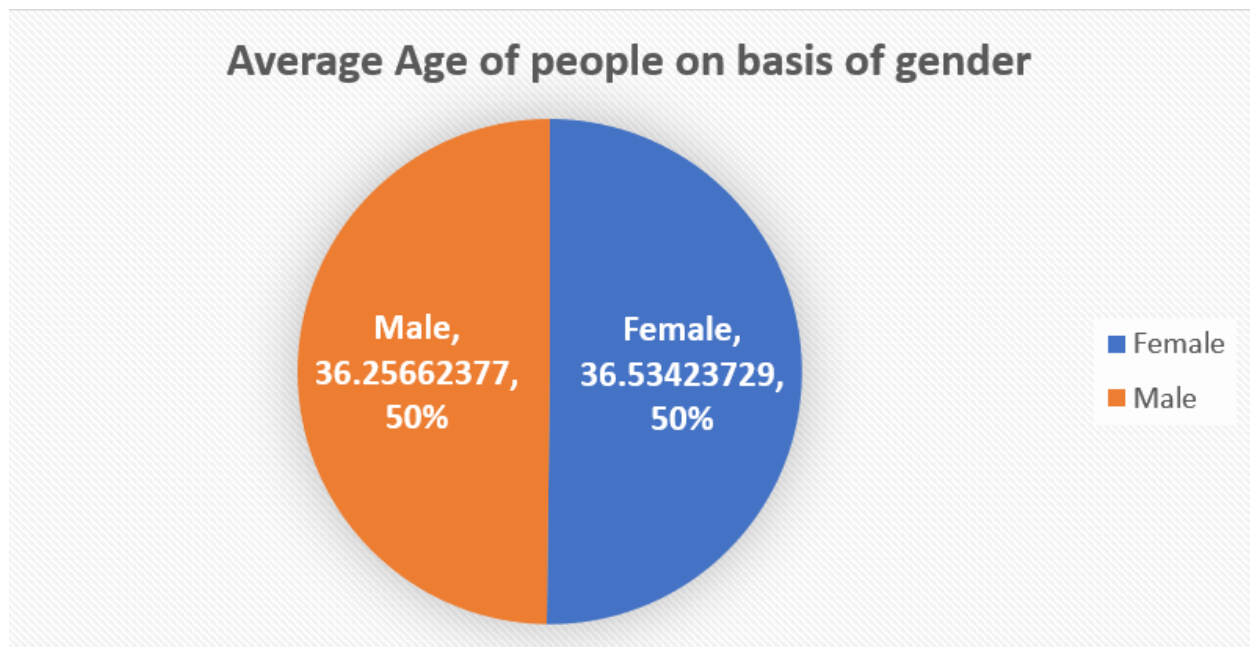


Figure 23: Visualization of Average age on basis of gender

```
sns.pairplot(ebola2, hue='adult_group')
```

<seaborn.axisgrid.PairGrid at 0x1d090059760>

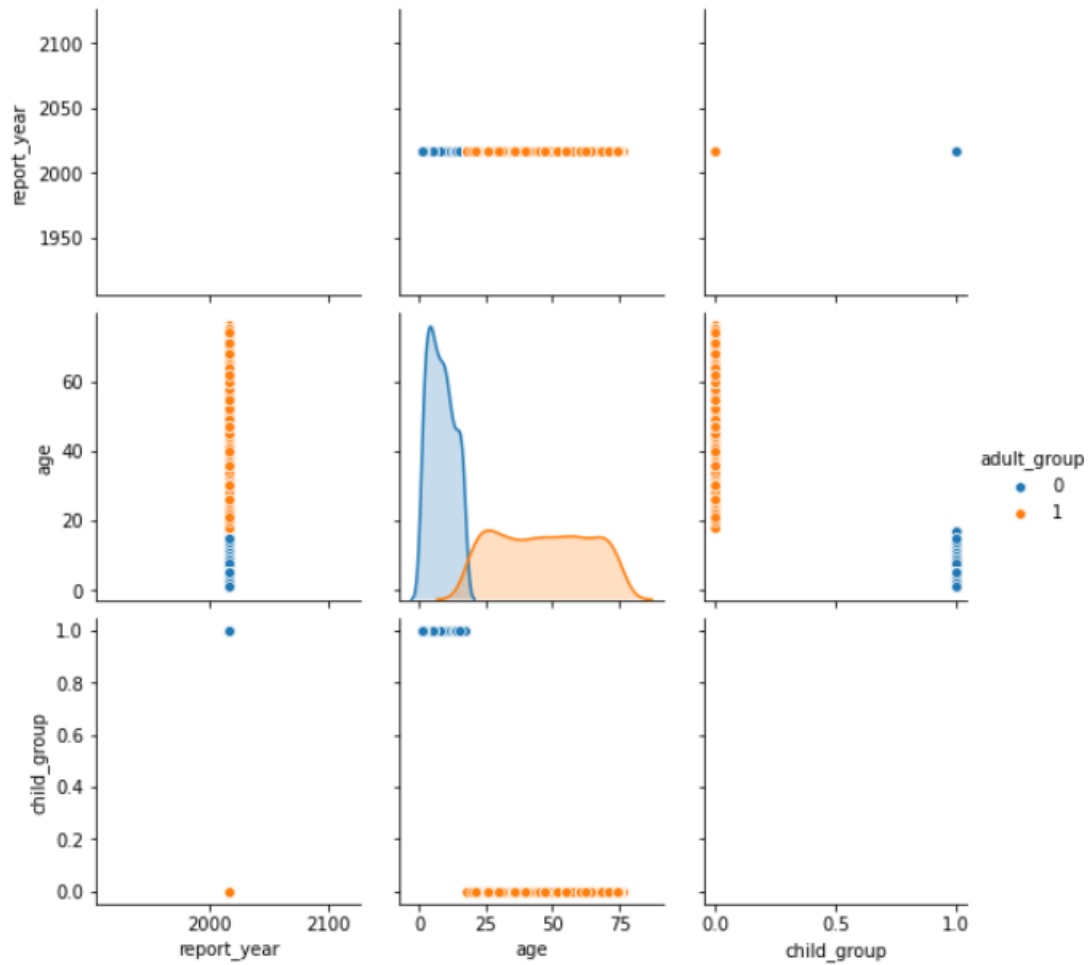


Figure 24: Over all visual representation of data used in MongoDB

Detail explanation on Appendix **7.2.2**

4.1.3 Hadoop

Hadoop MapReduce program for total confirmed cases and confirmed death in each country in time spam of 3 years

For MapReduce java code, appendix: **7.2.3.1**

```
1928579@hpd-srv:~/bigdatacwfinal$ hadoop jar EbolaHadoop.jar EbolaHadoop input_e/data_case.csv input_e/Ebola_death.csv output_e
```

Figure 25: Final MapReduce command Result

```
1928579@hpd-srv:~/bigdatacwfinal$ hdfs dfs -cat output_e/part-r-00000
Guinea 713850,441036
Italy 1,0
Liberia 41085,29050
Mali 14194,6875
Nigeria 21530,6960
Senegal 33440,9621
Sierra Leone 1921048,768620
Spain 35288,12946
United Kingdom 2,0
United States of America 32517,13071
```

After export in csv:

Guinea	713850	441036
Italy	1	0
Liberia	41085	29050
Mali	14194	6875
Nigeria	21530	6960
Senegal	33440	9621
Sierra Leone	1921048	768620
Spain	35288	12946
United Kingdom	2	0
United States of America	32517	13071

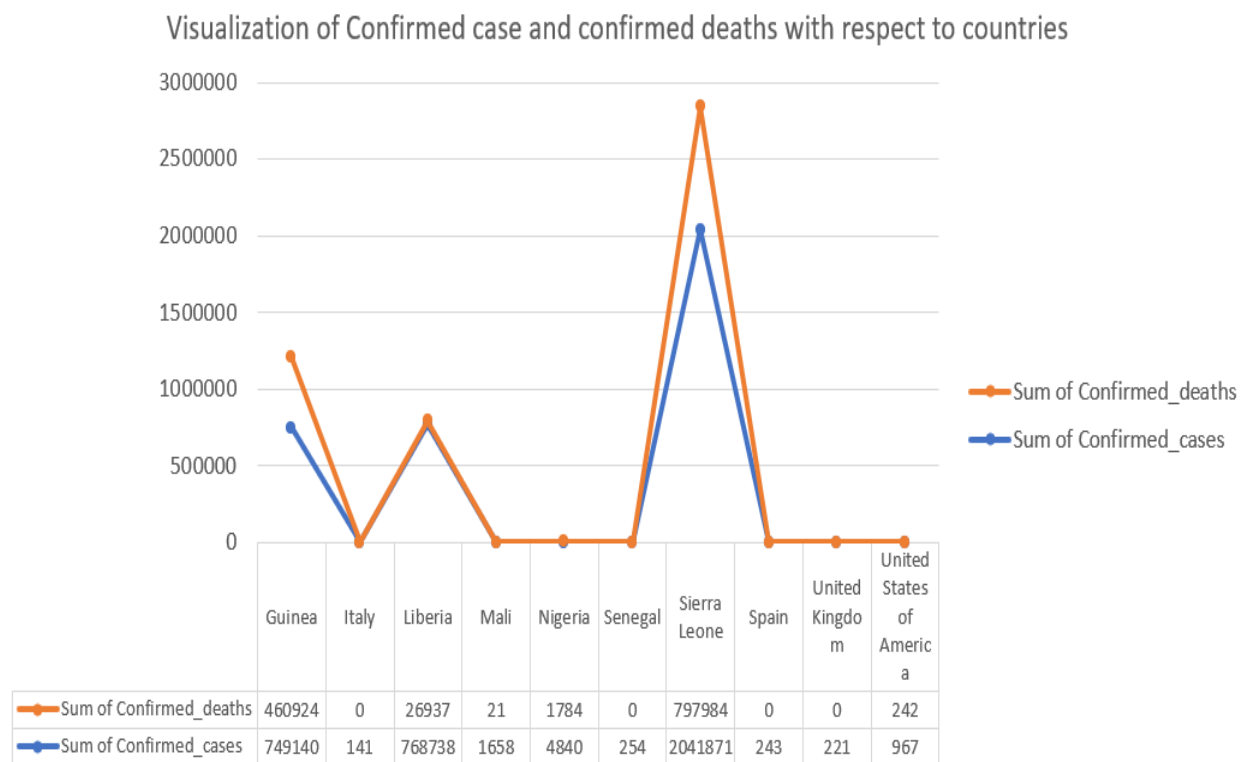


Figure 26: Visualization of cases and deaths with respect to countries

```
sns.pairplot(ebola)
```

```
<seaborn.axisgrid.PairGrid at 0x23700b08130>
```

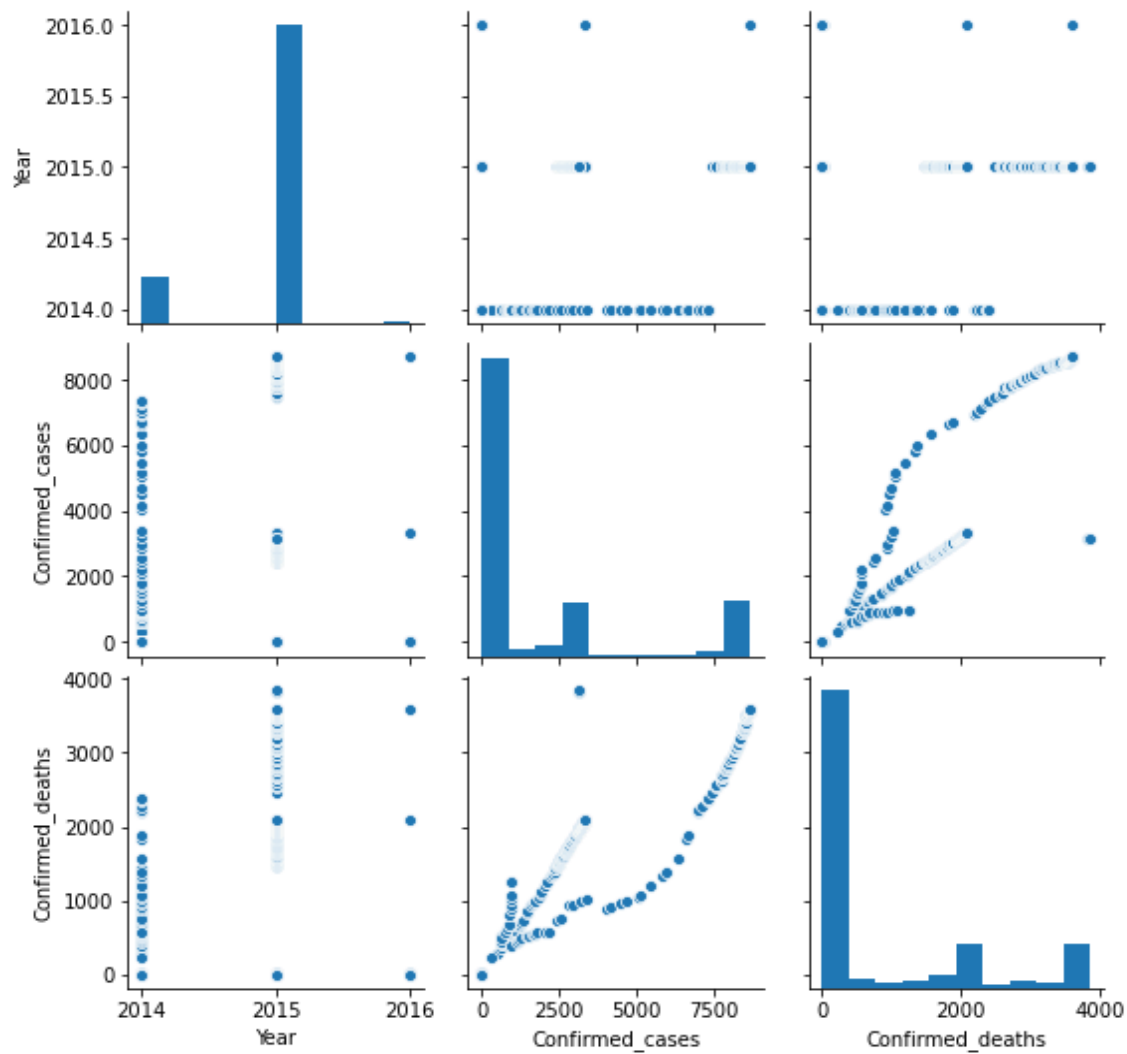


Figure 27: Over all visual representation of CSV data set

Explanation and justification for using .csv on Appendix 7.2.3.2

Figure 28 Importing csv dataset into pySpark


```
>>> sqlDF=spark.sql("SELECT COUNT(settlement)from Ebola Where settlement='Rural' ").show()
+-----+
|count(settlement)|
+-----+
|          1446|
+-----+

>>> sqlDF=spark.sql("SELECT COUNT(settlement)from Ebola Where settlement='Urban' ").show()
+-----+
|count(settlement)|
+-----+
|          1350|
+-----+
```

Figure 29 Death count in Rural and Urban area

```
>>> sqlDF=spark.sql("SELECT COUNT(gender)from Ebola Where gender='Female' ").show(20,False)
+-----+
|count(gender)|
+-----+
|1475         |
+-----+
```

```
>>> sqlDF=spark.sql("SELECT COUNT(gender)from Ebola Where gender='Male' ").show(20,False)
+-----+
|count(gender)|
+-----+
|1321         |
+-----+
```

Figure 30 Death on the basis of gender

```
>>> sqlDF=spark.sql("SELECT SUM(child_group)from Ebola").show()
+-----+
|sum(CAST(child_group AS DOUBLE))|
+-----+
|              741.0|
+-----+

>>> sqlDF=spark.sql("SELECT SUM(adult_group)from Ebola").show()
+-----+
|sum(CAST(adult_group AS DOUBLE))|
+-----+
|              2055.0|
+-----+
```

Figure 31 Death on basis of age-group

5 Comparison table

5.1 Advantages

5.1.1 Oracle:

1. larger community support for every step from installation to every queries.
2. Handles structured data well.
3. The certain types of errors in data was easy to trace while importing the data in Oracle
4. Case insensitive
5. Table handling was easy to flexible. We could add and delete the column easily.

5.1.2 Mango DB:

1. Major advantage of using Mongo DB is, we do not have to set primary key. Unique _id gets auto generated after creating documents.
2. We do not have to define schema.
3. Handles the unstructured data well.
4. Installation of Mongo DB is easy
5. Similar to python's dictionary, Mongo DB stores data in key-value pairs, which makes it more readable.

5.1.3 Hadoop:

1. Replication of data makes quick response and higher availability of data
2. Sharding emphasizes the capacity of database
3. Has high fault tolerance.
4. Works with multiple file systems.
5. Can handle large amount of data well.

5.2 Disadvantages

5.2.1 Oracle:

1. Handling unstructured data was difficult and tedious process in oracle.
2. Unlike Mongo DB, we had to define the schema in advance which makes Mango DB easy to access.
3. Less support for unstructured dataset.

5.2.2 Mango DB:

1. Visualization of error in Mongo DB was difficult.
2. Map-reduce is slower in Mongo DB compare to Hadoop.
3. Mongo Db is case sensitive.
4. Square brackets [], curly brackets { } are used every time, the brackets must be lined up correctly. If we miss one bracket or do not line up in incorrect format, it will throw error message.

5.2.3 Hadoop:

1. As our data set for coursework was small, it was tedious to work in Hadoop.
2. Tedious for small dataset as file has to be transferred for small work also example final coursework.
3. In MapReduce, we had to wait for entire process to finish for any output.
4. We must ensure if the updates are broadcasted to all copies of data.
5. Writing MapReduce program needed person with programming experience.

6 Conclusions and recommendations:

To sum up, we took two data sets, CSV and JSON which contained the information about Ebola in 2014-2016. We analyzed dataset with 3 different DBMS i.e. Oracle, MongoDB and Hadoop. We visualized dataset using excel and Jupyter Notebook. We learned the pros and cons of all three DBMS and their compatible data format. i.e. Oracle for .CSV, MongoDB for. Json and Hadoop is flexible for both formats. Using this dataset, we analyzed the cases and death count caused by Ebola virus and came to a conclusion that year 2015 was the most affected year. Rural areas were affected the most than urban areas and female were affected the most in compare to male.

7 Appendix

7.1 Cleaning the data

7.1.1 Document thoroughly what changes were made to the data and why they are carried out

Originally, the csv dataset contained 2484 records with 10 attributes but after cleaning the dataset, the dataset has been reduced to 1647. We cleaned our data using python's Pandas library. Pandas is a python's data manipulation tool which is used to deal with data analysis and manipulation (learnpython.org, 2020). To clean the data, we imported our csv dataset in Jupyter Notebook by using pandas function ". read_csv". We modified each column_name by making it short and eliminating spaces and various characters and symbols in between the columns name like "!", "#", "-" etc.

```
In [1]: import pandas as pd
```

```
In [2]: data= pd.read_csv('ebola_2014_2016.csv')
data.head(2)
```

Out[2]:

	Country	Date	No. of suspected cases	No. of probable cases	No. of confirmed cases	No. of confirmed, probable and suspected cases	No. of suspected deaths	No. of probable deaths	No. of confirmed deaths	No. of confirmed, probable and suspected deaths
0	Guinea	8/29/2014	25.0	141.0	482.0	648.0	2.0	141.0	287.0	430
1	Nigeria	8/29/2014	3.0	1.0	15.0	19.0	0.0	1.0	6.0	7

```
In [3]: for heading in enumerate(data.columns):
print(heading)
```

```
(0, 'Country')
(1, 'Date')
(2, 'No. of suspected cases')
(3, 'No. of probable cases')
(4, 'No. of confirmed cases')
(5, 'No. of confirmed, probable and suspected cases')
(6, 'No. of suspected deaths')
(7, 'No. of probable deaths')
(8, 'No. of confirmed deaths')
(9, 'No. of confirmed, probable and suspected deaths')
```

```
In [4]: new_header=['Country','Date','Suspected_cases','Probable_cases','Confirmed_cases','Total_cases','Suspected_deaths','Probable_dea',
'Confirmed_deaths','Total_deaths']
data= pd.read_csv('ebola_2014_2016.csv',header=0,names=new_header,skiprows=1)
data.head(2)
```

Out[4]:

	Country	Date	Suspected_cases	Probable_cases	Confirmed_cases	Total_cases	Suspected_deaths	Probable_deaths	Confirmed_deaths	Total_deaths
0	Nigeria	8/29/2014	3.0	1.0	15.0	19.0	0.0	1.0	6.0	7
1	Sierra Leone	8/29/2014	54.0	37.0	935.0	1026.0	8.0	34.0	380.0	422

After modifying the column name, we check if there are any missing value in the data set. For this we used pandas's `isnull().sum()` function.

```
data.isnull().sum()
```

```
Country      0
Date         0
Suspected_cases    119
Probable_cases    49
Confirmed_cases     1
Total_cases       8
Suspected_deaths   1177
Probable_deaths    959
Confirmed_deaths   837
Total_deaths      0
dtype: int64
```

For Knowledge Discovery in Databases (KDD), if the percentage of missing value of particular attribute is less than 1% then, it will not affect the procedure of data analysis but if it is more than 15%, we must follow 'Missing data treatment methods' (Peng & Lei, 2004). As we are analyzing all the confirmed cases and death cases of each countries in each year, we targeted Confirmed_cases and Confirmed_deaths. As there is only 1 missing value in Confirmed_cases, which is less than 1 % of total record, we did not make any changes in that column.

```
total=data.Confirmed_cases.shape[0]
num=data[data.Confirmed_cases.isnull()].shape[0]
```

```
percentage=(num/total)*100
percentage
```

```
0.040257648953301126
```

but in Confirmed_deaths, the rate of missing value is 33.6%, if not taken care of, will affect the analysis of data, because of that, we dropped all the records of missing value rows.

```
deaths_num= data[data.Confirmed_deaths.isnull()].shape[0]
```

```
death_percentage=(deaths_num/total)*100
death_percentage
```

```
33.69565217391305
```

Why we dropped the records?

First, we checked whether the dataset is Missing Completely at Random (MCAR) or Not Missing at Random (NMAR). To know this, we used IBM SPSS Statistics software's expectation-maximization (EM) technique.

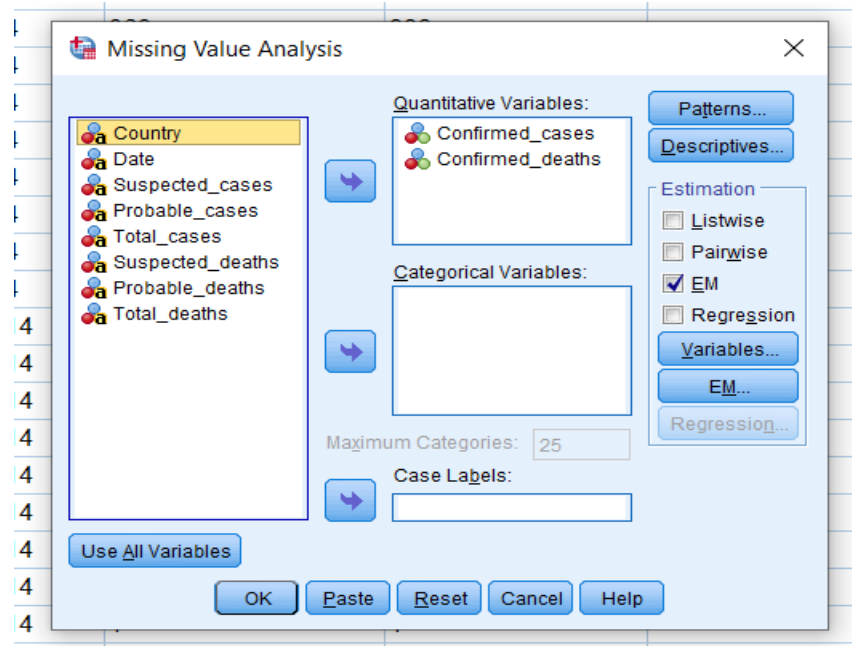


Figure 32 Analysis of missing value using IBM SPSS software

In EM technique, as the value of Sig is less than 0.5, we can say that dataset is MCAR. As our dataset is MCAR, we have deleted the missing value instead of using mean or mode.

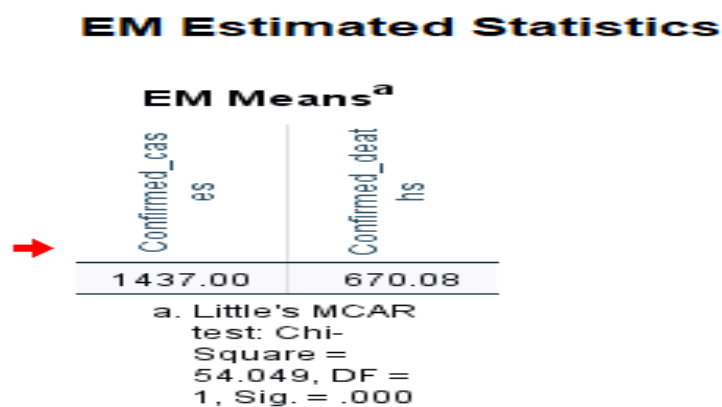


Figure 33 : Showing Sig value less than 0.5

To drop all the missing values, .dropna() function has been used. Below is the screenshot of the codes which shows dataset before and after the values and size of dataset.

```
data[data.Confirmed_deaths.isnull()].head()
```

	Country	Date	Suspected_cases	Probable_cases	Confirmed_cases	Total_cases	Suspected_deaths	Probable_deaths	Confirmed_deaths	Total_deaths
70	Liberia	10/15/2014	1376.0	1923.0	950.0	4249.0	NaN	NaN	NaN	2458
79	Liberia	10/17/2014	NaN	NaN	NaN	4262.0	NaN	NaN	NaN	2484
96	Liberia	10/29/2014	2480.0	1540.0	2515.0	6535.0	NaN	NaN	NaN	2413
101	Guinea	10/29/2014	316.0	199.0	1391.0	1906.0	NaN	NaN	NaN	997
103	Sierra Leone	10/29/2014	1213.0	322.0	3700.0	5235.0	NaN	NaN	NaN	1500

```
data.shape
```

```
(2484, 10)
```

```
data.dropna(subset=['Confirmed_cases','Confirmed_deaths'],how='any', inplace=True)
```

```
data.Confirmed_cases.isnull().sum()
```

```
0
```

```
data.Confirmed_deaths.isnull().sum()
```

```
0
```

```
data.shape
```

```
(1647, 10)
```

```
data.head()
```

	Country	Date	Suspected_cases	Probable_cases	Confirmed_cases	Total_cases	Suspected_deaths	Probable_deaths	Confirmed_deaths	Total_deaths
0	Nigeria	8/29/2014	3.0	1.0	15.0	19.0	0.0	1.0	6.0	7
1	Sierra Leone	8/29/2014	54.0	37.0	935.0	1026.0	8.0	34.0	380.0	422
2	Liberia	8/29/2014	382.0	674.0	322.0	1378.0	168.0	301.0	225.0	694
3	Sierra Leone	9/5/2014	78.0	37.0	1146.0	1261.0	11.0	37.0	443.0	491
4	Nigeria	9/5/2014	3.0	1.0	18.0	22.0	0.0	1.0	7.0	8

Figure 34 :Dropping the missing data

After Modifying all the column_name and addressing the missing value, the whole dataset was cleaned and was ready to process the csv dataset in Oracle and Hadoop. Pandas to_csv () function has been used to save the cleaned the dataset.

```
data.to_csv('Ebola_data_second.csv',index=False)
```

Figure 35 Saving the cleaned dataset

7.2 Analysis of the data

- to include examples of querying the data, e.g., SQL or queries using MongoDB or Hadoop

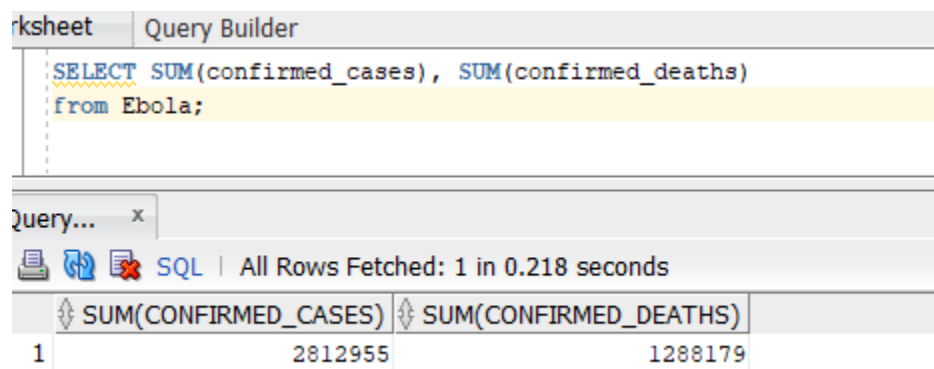
7.2.1 Oracle

1. First, Total confirmed cases and death cases caused Ebola in 3 years were analyzed. To fetch the related data, SUM() function has been used. After applying the query, it was found that total of 2812955 people got infected and Total of 1288179 people lost their life.

Query used:

```
SELECT SUM(confirmed_cases), SUM(confirmed_deaths)
from Ebola;
```

Result of the query:



The screenshot shows the Oracle SQL Developer interface. At the top, the 'Query Builder' tab is active, displaying the SQL query: `SELECT SUM(confirmed_cases), SUM(confirmed_deaths) from Ebola;`. Below the query editor, a 'Query...' window shows the execution status: 'All Rows Fetched: 1 in 0.218 seconds'. At the bottom, a table displays the results of the query.

	SUM(CONFIRMED_CASES)	SUM(CONFIRMED_DEATHS)
1	2812955	1288179

2. Second, out of the total confirmed cases 2812955 and deaths of 1288179, cases and deaths in each country in 3 years was analyzed using SUM () function and GROUPBY clause. After analyzing the data, it was found that country Sierra Leone was affected the most most with total confirmed cases of 1921048 and death of 768620 whereas Italy and UK were least affected with no casualty in 3 years.

Query Used:

```
SELECT SUM(confirmed_cases), SUM(confirmed_deaths),country
from ebola
GROUP BY country;
```

Result of the query:

```
SELECT SUM(confirmed_cases), SUM(confirmed_deaths),country
from ebola
GROUP BY country;
```

	SUM(CONFIRMED_CASES)	SUM(CONFIRMED_DEATHS)	COUNTRY
1	713850	441036	Guinea
2	41085	29050	Liberia
3	35288	12946	Spain
4	1	0	Italy
5	1921048	768620	Sierra Leone
6	14194	6875	Mali
7	32517	13071	United States of America
8	2	0	United Kingdom
9	33440	9621	Senegal
10	21530	6960	Nigeria

As Rollup was taught during the academic period of oracle, Rollup command was also used to fetched the same data.

Query Used:

```
SELECT SUM(confirmed_cases), SUM(confirmed_deaths),country,edate
from ebola
GROUP BY ROLLUP (country,edate);
```

Result of the query:

```
SELECT SUM(confirmed_cases), SUM(confirmed_deaths),country,edate
from ebola
GROUP BY ROLLUP (country,edate);
```

	SUM(CONFIRMED_CASES)	SUM(CONFIRMED_DEATHS)	COUNTRY	EDATE
1	3001	940	Mali	17-OCT-14
2	5501	2945	Mali	22-OCT-14
3	5692	2990	Mali	25-OCT-14
4	14194	6875	Mali	(null)
5	1	0	Italy	15-MAY-15
6	1	0	Italy	(null)

As Rollup sum-up all the total at last of each head, using Roll-up, same output has been gained. for assurance, confirmed_cases and confirmed_death from both queries are same for Mali and Italy.

3. Third, as the Ebola was active for 3 continuous year from 2014 to 2016. Analysis of, in which year Ebola virus was spread the most was done. Using SUM () function, WildCards (), LIKE Function and Where clause, it can be seen that, out of three years, year 2015 had the most cases and death rate in all 3 years with total of 2584992 confirmed_cases and total of 1197804 death cases.

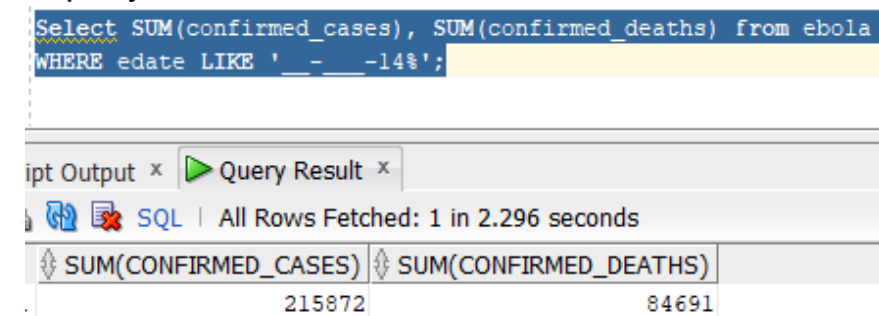
Query Used:

```
Select SUM(confirmed_cases), SUM(confirmed_deaths) from ebola
WHERE edate LIKE '__-__-14%';
```

```
Select SUM(confirmed_cases), SUM(confirmed_deaths) from ebola
WHERE edate LIKE '__-__-15%';
```

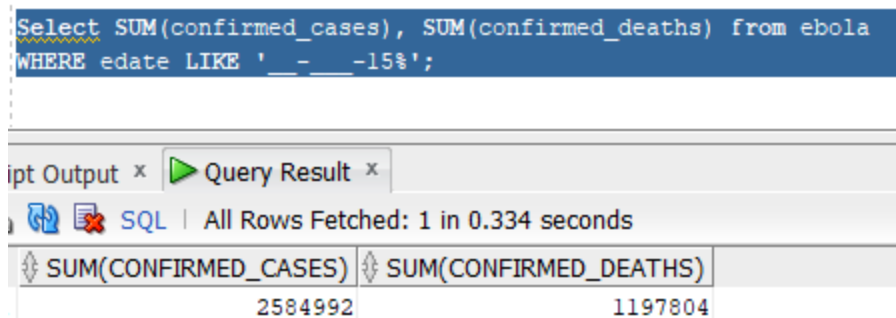
```
Select SUM(confirmed_cases), SUM(confirmed_deaths) from ebola
WHERE edate LIKE '__-__-16%';
```

Result of the query:



The screenshot shows a SQL query editor with the query: `Select SUM(confirmed_cases), SUM(confirmed_deaths) from ebola WHERE edate LIKE '__-__-14%';`. Below the query, the results are displayed in a table with two columns: `SUM(CONFIRMED_CASES)` and `SUM(CONFIRMED_DEATHS)`. The values are 215872 and 84691 respectively. The interface also shows 'All Rows Fetched: 1 in 2.296 seconds'.

SUM(CONFIRMED_CASES)	SUM(CONFIRMED_DEATHS)
215872	84691



The screenshot shows a SQL query editor with the query: `Select SUM(confirmed_cases), SUM(confirmed_deaths) from ebola WHERE edate LIKE '__-__-15%';`. Below the query, the results are displayed in a table with two columns: `SUM(CONFIRMED_CASES)` and `SUM(CONFIRMED_DEATHS)`. The values are 2584992 and 1197804 respectively. The interface also shows 'All Rows Fetched: 1 in 0.334 seconds'.

SUM(CONFIRMED_CASES)	SUM(CONFIRMED_DEATHS)
2584992	1197804

<pre>Select SUM(confirmed_cases), SUM(confirmed_deaths) from ebola WHERE edate LIKE '___-___-16%';</pre>		
Script Output x Query Result x		
SQL All Rows Fetched: 1 in 0.33 seconds		
	SUM(CONFIRMED_CASES)	SUM(CONFIRMED_DEATHS)
1	12091	5684

4. Fourth, after it was cleared that in 2015, the cases and casualties were highest, analysis of confirmed_cases and confirmed_death of each country in each year from 2014-16 were analyzed. For this SUM(), Wildcards, LIKE Function, GROUPBY clause were used. After applying the queries for 2014, In 2014, Spain had the highest number of infected people with 35071 but total death number of Liberia was the highest with dead count of 13498. Nigeria had the least number of death cases with 5441 in 2014.

Query Used:

```
SELECT SUM(confirmed_cases), SUM(confirmed_deaths), country
from Ebola
where edate LIKE '___-___-14%'
GROUP BY country;
```

Result of the query:

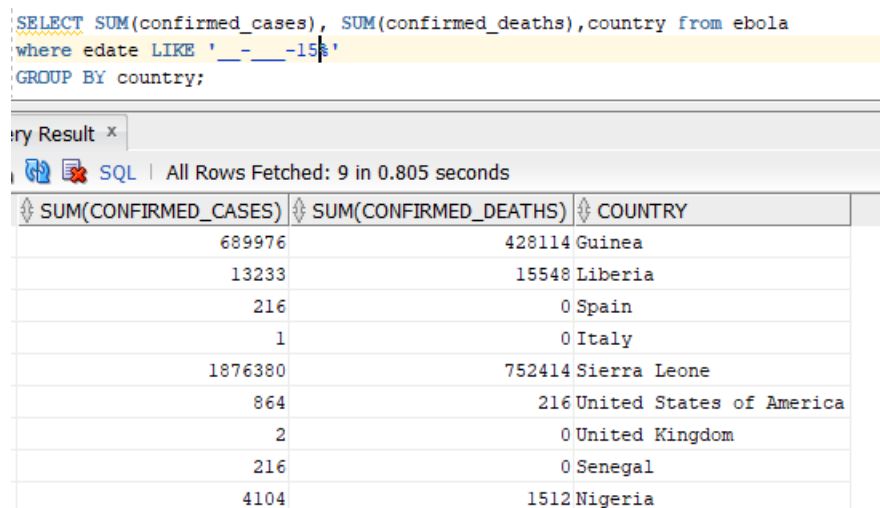
<pre>SELECT SUM(confirmed_cases), SUM(confirmed_deaths),country from ebola where edate LIKE '___-___-14%' GROUP BY country;</pre>		
Query Result x		
SQL All Rows Fetched: 8 in 2.389 seconds		
	SUM(CONFIRMED_CASES)	SUM(CONFIRMED_DEATHS) COUNTRY
	20523	10839 Guinea
	27841	13498 Liberia
	35071	12946 Spain
	14194	6875 Mali
	35964	12617 Sierra Leone
	31649	12854 United States of America
	17407	5441 Nigeria
	33223	9621 Senegal

In 2015, infected population and total death number of Sierra Leone is at highest with 1876380 cases and death count reached 752414 whereas Italy had only 1 case with no human casualty.

Query Used:

```
SELECT SUM(confirmed_cases), SUM(confirmed_deaths), country
from Ebola
where edate LIKE '__-__-15%'
GROUP BY country;
```

Result of the query:



SUM(CONFIRMED_CASES)	SUM(CONFIRMED_DEATHS)	COUNTRY
689976	428114	Guinea
13233	15548	Liberia
216	0	Spain
1	0	Italy
1876380	752414	Sierra Leone
864	216	United States of America
2	0	United Kingdom
216	0	Senegal
4104	1512	Nigeria

In 2016, cases were reduced drastically. Sierra Leone had high cases and death count with 8404 infected cases and 3589 death cases followed by Guinea which had 3351 confirmed_cases and 2083 death cases whereas other nations were recovering well with just couple of death cases and few active infected cases.

Query Used:

```
SELECT SUM(confirmed_cases), SUM(confirmed_deaths), country
from Ebola
where edate LIKE '__-__-16%'
GROUP BY country;
```

Result of the query:



	SUM(CONFIRMED_CASES)	SUM(CONFIRMED_DEATHS)	COUNTRY
1	3351	2083	Guinea
2	11	4	Liberia
3	1	0	Spain
4	8704	3589	Sierra Leone
5	4	1	United States of America
6	1	0	Senegal
7	19	7	Nigeria

7.2.2 Mongo DB

1) Connecting to mongo db server

```
Microsoft Windows [Version 10.0.18363.815]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Program Files\MongoDB\Server\4.2\bin>mongod
2020-05-10T15:27:38.181+0545 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2020-05-10T15:27:38.510+0545 I CONTROL [initandlisten] MongoDB starting : pid=6040 port=27017 dbpath=C:\data\db\ 64-bit host=DESKTOP-J68221J
2020-05-10T15:27:38.510+0545 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2020-05-10T15:27:38.510+0545 I CONTROL [initandlisten] db version v4.2.3
2020-05-10T15:27:38.510+0545 I CONTROL [initandlisten] git version: 6874650b362138df74be53d366bbefc321ea32d4
2020-05-10T15:27:38.510+0545 I CONTROL [initandlisten] allocator: tcmalloc
2020-05-10T15:27:38.510+0545 I CONTROL [initandlisten] modules: none
2020-05-10T15:27:38.510+0545 I CONTROL [initandlisten] build environment:
2020-05-10T15:27:38.511+0545 I CONTROL [initandlisten] distmod: 2012plus
2020-05-10T15:27:38.511+0545 I CONTROL [initandlisten] distarch: x86_64
2020-05-10T15:27:38.511+0545 I CONTROL [initandlisten] target_arch: x86_64
2020-05-10T15:27:38.511+0545 I CONTROL [initandlisten] options: {}
2020-05-10T15:27:38.516+0545 I STORAGE [initandlisten] Detected data files in C:\data\db\ created by the 'wiredTiger' storage engine, so setting the active storage engine to 'wiredTiger'
2020-05-10T15:27:38.517+0545 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=7602M,cache_overflow=(file_max=0M),session_max=33000,eviction=(threads_min=4,threads_max=4),statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000,close_scan_interval=10,close_handle_minimum=250),statistics_log=(open_interval=60,quiet=false)
2020-05-10T15:27:38.990+0545 I SHARDING [initandlisten] Marking collection admin.system.version as collection version: <unsharded>
2020-05-10T15:27:39.003+0545 I SHARDING [initandlisten] Marking collection local.startup_log as collection version: <unsharded>
2020-05-10T15:27:39.150+0545 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory 'C:/data/db/diagnostic.data'
2020-05-10T15:27:39.153+0545 I SHARDING [LogicalSessionCacheRefresh] Marking collection config.system.sessions as collection version: <unsharded>
2020-05-10T15:27:39.153+0545 I SHARDING [LogicalSessionCacheReap] Marking collection config.transactions as collection version: <unsharded>
2020-05-10T15:27:39.155+0545 I NETWORK [listener] Listening on 127.0.0.1
2020-05-10T15:27:39.155+0545 I NETWORK [listener] waiting for connections on port 27017
2020-05-10T15:27:40.004+0545 I SHARDING [ftdc] Marking collection local.oplog.rs as collection version: <unsharded>
```

2) Importing the json file:

```
> show dbs
admin          0.000GB
big            0.000GB
bigdatacw     0.000GB
config        0.000GB
la            0.000GB
local         0.000GB
worksheetdb   0.001GB
ws5           0.000GB
```

```
mongoimport --db bigdatacw --collection bigcw --file
C:\Users\msi\Downloads\data.json --jsonArray
```

```
C:\Program Files\MongoDB\Server\4.2\bin>mongoimport --db bigdatacw --collection bigcw --file C:\Users\msi\Downloads\data.json --jsonArray
2020-05-10T15:29:53.242+0545 connected to: mongodb://localhost/
2020-05-10T15:29:53.361+0545 2796 document(s) imported successfully. 0 document(s) failed to import.
```

```
C:\Program Files\MongoDB\Server\4.2\bin>mongo
MongoDB shell version v4.2.3
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("7eebea7b-e374-4005-b376-1ad16a30f7dc") }
MongoDB server version: 4.2.3
Server has startup warnings:
2020-05-10T12:38:09.902+0545 I CONTROL [initandlisten]
2020-05-10T12:38:09.902+0545 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-05-10T12:38:09.902+0545 I CONTROL [initandlisten] **           Read and write access to data and configuration is unrestricted.
2020-05-10T12:38:09.902+0545 I CONTROL [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
```


3) Displaying data

```
db.bigcw.find().pretty()
```

```
> db.bigcw.find().pretty()
{
  "_id" : ObjectId("5eb7cd150c0293fd4dc74f68"),
  "gender" : "Male",
  "settlement" : "Urban",
  "report_date" : "1/20/2016",
  "report_year" : 2016,
  "age" : 31,
  "child_group" : 0,
  "adult_group" : 1,
  "disease" : "Ebola"
}
{
  "_id" : ObjectId("5eb7cd150c0293fd4dc74f69"),
  "gender" : "Female",
  "settlement" : "Rural",
  "report_date" : "7/5/2016",
  "report_year" : 2016,
  "age" : 66,
  "child_group" : 0,
  "adult_group" : 1,
  "disease" : "Ebola"
}
{
  "_id" : ObjectId("5eb7cd150c0293fd4dc74f6a"),
  "gender" : "Male",
  "settlement" : "Rural",
  "report_date" : "11/19/2016",
  "report_year" : 2016,
  "age" : 3,
  "child_group" : 1,
  "adult_group" : 0,
  "disease" : "Ebola"
}
{
  "_id" : ObjectId("5eb7cd150c0293fd4dc74f6b"),
  "gender" : "Female",
  "settlement" : "Rural",
  "report_date" : "8/26/2016",
  "report_year" : 2016,
  "age" : 5,
  "child_group" : 1,
  "adult_group" : 0,
  "disease" : "Ebola"
}
```

Total number of data has been counted to see the total cases of ebola in Nigeria in 2016.

```
> db.bigcw.find().count()
2796
>
```

Total male and female suffered by ebola:

From this we can analyze the total number of male and female who suffered from Ebola. From the extracted data, we can clearly see that the number of Females who suffered from Ebola is greater than Male i.e. 1321 Male and 1475 Female got Ebola. We can analyze that the Ebola was mostly contagious to female.

```
db.bigcw.count({gender:"Male"})
db.bigcw.count({gender:"Female"})
```

```
> db.bigcw.find().count()
2796
> db.bigcw.count({gender:"Male"})
1321
> db.bigcw.count({gender:"Female"})
1475
>
```

Total rural and urban area suffered by ebola:

Below figure shows the analysis done for the person suffered by Ebola in Rural and Urban areas. It shows that there were the maximum number of cases in Rural area than in Urban area. There were 1446 cases in Rural area whereas 1350 cases in Urban area. there were 96 case more in Rural area while comparing to Urban area. The cause behind this may be due to the smaller number of hospitals in Rural area.

```
Db.bigcw.count({settlement:"Rural"})
Db.bigcw.count({settlement:"Urban"})
```

```
> db.bigcw.count({settlement:"Rural"})
1446
> db.bigcw.count({settlement:"Urban"})
1350
>
```

We used aggregation to analyze the data to see the number of children aged between 1-17 suffered by Ebola. There were total of 741 cases of Ebola for child group. Among them, 352 cases were for Male Children and 389 for Female Children. We can analysis that the female children suffered more from Ebola.

```
db.bigcw.aggregate (
    [
```

```

    {$match:{}},
    {$group: {_id:"$gender",total:{$sum:"$child_group"} } }
  ]
)

```

```

> db.bigcw.aggregate(
... [
... {$match:{}},
... {$group: {_id:"$gender",total:{$sum:"$child_group"} } }
... ]
... )
{ "_id" : "Male", "total" : 352 }
{ "_id" : "Female", "total" : 389 }
>

```

We used aggregation to analyze the data to see the number of adult aged between 18-76 suffered by Ebola. There were total of 2,055 cases of Ebola for adult group. Among them, 969 cases were for Male Adult and 1,086 for Female Adult. We can analysis that the female children suffered more from Ebola by 117 cases.

```

db.bigcw.aggregate(
  [
    {$match:{}},
    {$group: {_id:"$gender",total:{$sum:"$adult_group"}
    }
  ]
)

```

```

> db.bigcw.aggregate(
... [
... {$match:{}},
... {$group:{_id:"$gender",total:{$sum:"$adult_group"} } }
... ]
... )
{ "_id" : "Male", "total" : 969 }
{ "_id" : "Female", "total" : 1086 }
>

```

From below it is analysed that the average age group of the people suffered by Ebola. It is seen that the average age for both male and female is 36 years.

```

db.bigcw.aggregate(
  [
    {$group:{_id:"$gender",total:{$avg:"$age"} } }
  ]
)

```

```

> db.bigcw.aggregate(
... [
... {$group:{_id:"$gender",total:{$avg:"$age"} } }
... ]
... )
{ "_id" : "Male", "total" : 36.25662376987131 }
{ "_id" : "Female", "total" : 36.53423728813559 }
>

```

From below it is analysed that the oldest age group of the people suffered by Ebola. It is seen that the oldest age for both male and female is 76 years.

```

db.bigcw.aggregate(
    [
        {$group: {_id: "$gender", oldest: {$max: "$age"}}
    } ]
)

```

```

> db.bigcw.aggregate(
... [
... {$group: {_id: "$gender", oldest: {$max: "$age"}} } ]
... ]
... )
{ "_id" : "Male", "oldest" : 76 }
{ "_id" : "Female", "oldest" : 76 }
>

```

From below it is analysed that the youngest age group of the people suffered by Ebola. It is seen that the average age for both male and female is 1 year.

```

db.bigcw.aggregate(
    [
        {$group: {_id: "$gender", youngest: {$min: "$age"}} } ]
    ]
)

```

```

> db.bigcw.aggregate(
... [
... {$group: {_id: "$gender", youngest: {$min: "$age"}} } ]
... ]
... )
{ "_id" : "Male", "youngest" : 1 }
{ "_id" : "Female", "youngest" : 1 }
>

```

From all the above analysis, we can analysis that Female cases to suffer more Ebola is greater than male. Both for the Child and Adult case, Female suffered more from Ebola. There were 1475 cases for female Child and 1086 for Adult Female. Similarly, while, comparing between Rural and Urban area, there we more cases in Rural than in the Urban. There were 1446 cases in Rural area which is comparatively greater than Urban area having the case of 1350. It is also analyzed that the average age of the people who suffered from Ebola is 36 years.

7.2.3 Hadoop Mapreduce:

7.2.3.1 Explanation of code:

With the understanding from the workshops, we have applied MapReduce program to analyze total confirmed cases and confirmed death in each country in time spam of 3 years. We have encoded two mapper class with the name 'First_mapper' and 'Second_mapper' in order to split the data on the basis of comma ',' and add in the context. After the process of Mapper, Reducer input the values in a set of key-value pair and after that, the work of reducer is to combine the key and value pairs, apply filters to process the output.

```
1928579@hpd-srv:~/bigdatacwfinal$ hadoop jar EbolaHadoop.jar EbolaHadoop input_e/data_case.csv input_e/Ebola_death.csv output_e
```

Figure 36 Final MapReduce command

```
1928579@hpd-srv:~/bigdatacwfinal$ hdfs dfs -cat output_e/part-r-00000
Guinea 713850,441036
Italy 1,0
Liberia 41085,29050
Mali 14194,6875
Nigeria 21530,6960
Senegal 33440,9621
Sierra Leone 1921048,768620
Spain 35288,12946
United Kingdom 2,0
United States of America 32517,13071
```

Figure 37 Output of mapreduce in putty

Below are the logics of MapReduce to obtain the targeted result.

```

import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class EbolaHadoop {
    public static class First_mapper extends Mapper <Object, Text, Text, Text>
    {
        public void map(Object key, Text value, Context context)
            throws IOException, InterruptedException {
            String record = value.toString();
            String[] parts = record.split(",");
            // 0: Key (county) 1: Year 2: figure
            // default to 0 if null value
            if (parts.length >= 1 )
                context.write(new Text(parts[0]), new Text("first\t" +
parts[1]));
            else
                context.write(new Text(parts[0]), new Text("first\t" + 0));
        } // map method
    } // First_mapper

    public static class Second_mapper extends Mapper <Object, Text, Text, Text>
    {
        public void map(Object key, Text value, Context context)
            throws IOException, InterruptedException {
            String record = value.toString();
            String[] parts = record.split(",");
            // 0: Key (county) 1: Year 2: figure
            // default to 0 if null value
            if (parts.length >= 1)
                context.write(new Text(parts[0]), new Text("second\t" +
parts[1]));
            else
                context.write(new Text(parts[0]), new Text("second\t" + 0));
        } // map method
    } // Second_mapper

    public static class EbolaReducer extends Reducer <Text, Text, Text, Text> {
        public void reduce(Text key, Iterable<Text> values, Context context)
            throws IOException, InterruptedException {
            String firstName = "";
            String secondName = "";
            int firstTotal = 0;
            int secondTotal = 0;

```

```

for (Text t : values) {
    String parts[] = t.toString().split("\t");
    if (parts[0].equals("second")) {

        secondTotal += Integer.parseInt(parts[1]);
    } // if
    else if (parts[0].equals("first")) {

        firstTotal += Integer.parseInt(parts[1]);
    } // else
} // for loop

    String str = String.format("%d,%d", firstTotal, secondTotal);
    context.write(new Text(key), new Text(str));
} // reduce method
} // JoinReducer

public static void main(String[] args) throws Exception {

    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "Join datasets");
    job.setJarByClass(EbolaHadoop.class);
    job.setReducerClass(EbolaReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);

    MultipleInputs.addInputPath(job, new Path(args[0]), TextInputFormat.class,
First_mapper.class);
    MultipleInputs.addInputPath(job, new Path(args[1]), TextInputFormat.class,
Second_mapper.class);

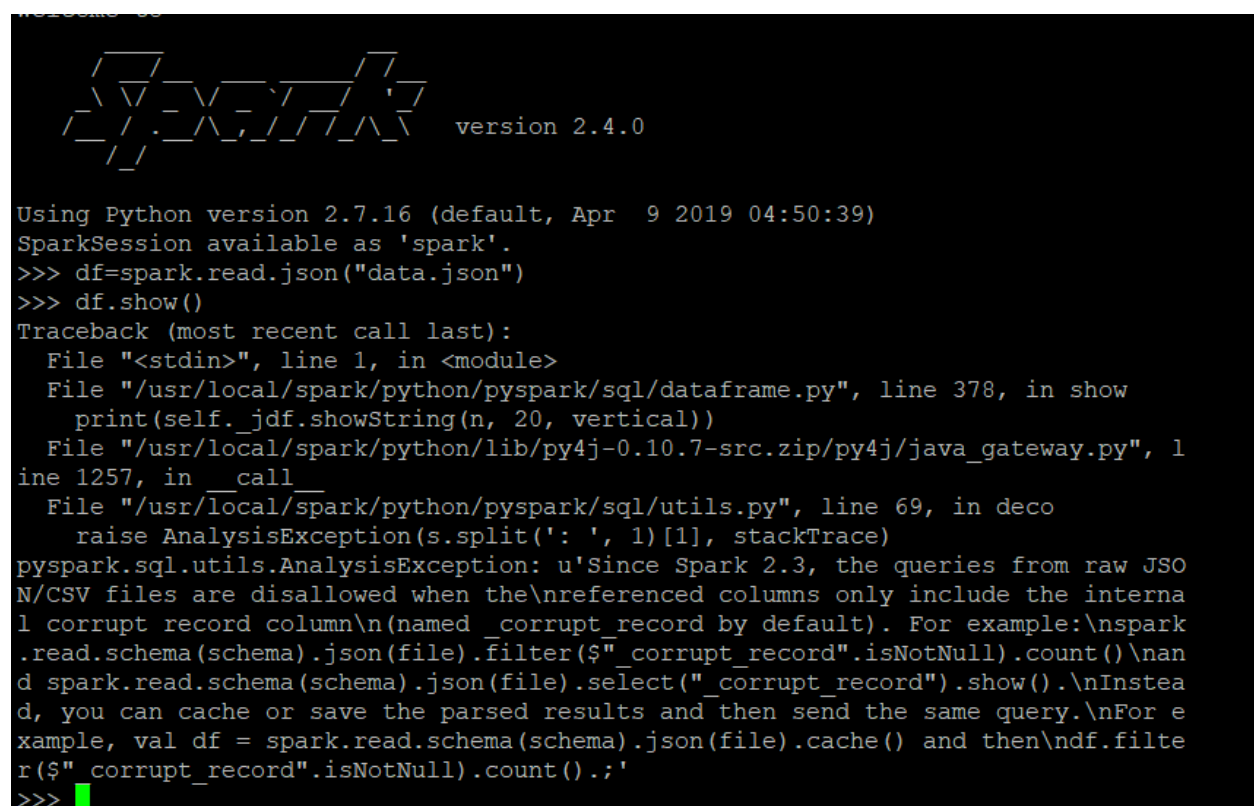
    Path outputPath = new Path(args[2]);
    FileOutputFormat.setOutputPath(job, outputPath);
    // Delete the output directory
    outputPath.getFileSystem(conf).delete(outputPath, true);

    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}

```


7.2.3.2 Spark

Json dataset analyzed in Mongo DB was used in spark. Because of the problem in Json dataset while importing in spark, dataset was converted into .csv format and processed each step. As the original Json format dataset was getting hard to find, we converted one csv format to .json to compute in MongoDB and it ran without any error. But unfortunately, while importing the same Json file in Spark, because of unknow reason, the json data could not ran in spark that is why we used original .csv file.



```

version 2.4.0

Using Python version 2.7.16 (default, Apr  9 2019 04:50:39)
SparkSession available as 'spark'.
>>> df=spark.read.json("data.json")
>>> df.show()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 378, in show
    print(self._jdf.showString(n, 20, vertical))
  File "/usr/local/spark/python/lib/py4j-0.10.7-src.zip/py4j/java_gateway.py", line 1257, in __call__
  File "/usr/local/spark/python/pyspark/sql/utils.py", line 69, in deco
    raise AnalysisException(s.split(':', 1)[1], stackTrace)
pyspark.sql.utils.AnalysisException: u'Since Spark 2.3, the queries from raw JSON/CSV files are disallowed when the\nreferenced columns only include the internal corrupt record column\n(named _corrupt_record by default). For example:\nspark.read.schema(schema).json(file).filter($"_corrupt_record".isNotNull).count()\nand spark.read.schema(schema).json(file).select("_corrupt_record").show().\nInstead, you can cache or save the parsed results and then send the same query.\nFor example, val df = spark.read.schema(schema).json(file).cache() and then\ndf.filter($"_corrupt_record".isNotNull).count().;'
>>>
```

Figure 38 Error in Json file

Similar to MongoDB, we have analyzed the death case of Ebola in the country Nigeria. It analyzed the death case on the basis of gender, settlement area and age-group.

```

>>> sqlDF=spark.sql("SELECT COUNT(settlement)from Ebola Where settlement='Rural' ").show()
+-----+
|count(settlement)|
+-----+
|          1446|
+-----+

>>> sqlDF=spark.sql("SELECT COUNT(settlement)from Ebola Where settlement='Urban' ").show()
+-----+
|count(settlement)|
+-----+
|          1350|
+-----+

```

Figure 39 Death on the basic of settlement area

Ebola was spread in rural area more and in urban area. Rural area had 1446 death count whereas Urban had 1350 death count.

```

>>> sqlDF=spark.sql("SELECT COUNT(gender)from Ebola Where gender='Female' ").show(20,False)
+-----+
|count(gender)|
+-----+
|1475      |
+-----+

>>> sqlDF=spark.sql("SELECT COUNT(gender)from Ebola Where gender='Male' ").show(20,False)
+-----+
|count(gender)|
+-----+
|1321      |
+-----+

```

Figure 40 death count on the basis of gender

It can be seen female were affected the most than male.

```

>>> sqlDF=spark.sql("SELECT SUM(child_group)from Ebola").show()
+-----+
|sum(CAST(child_group AS DOUBLE))|
+-----+
|          741.0|
+-----+

>>> sqlDF=spark.sql("SELECT SUM(adult_group)from Ebola").show()
+-----+
|sum(CAST(adult_group AS DOUBLE))|
+-----+
|        2055.0|
+-----+

```

Figure 41 death on the basis of age-group

Adult were infected the most than child age group.

7.3 Visualizations:

The visualizations are made using Excel charts and Jupyter note book using seaborn and matplotlib libraries. There are simple charts such as Bargarh and Pie-chart because the data can be analyzed from that method of visualization in better way. The pie chart, bar graph and line chart were made using pivot table. Demonstration of how it was made is shown below:

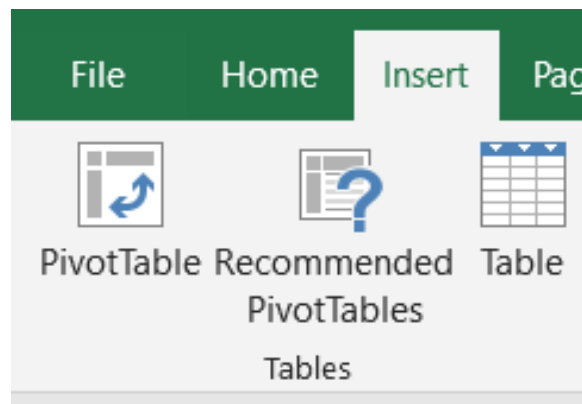


Figure 42 :Pivot table option from Insert is selected

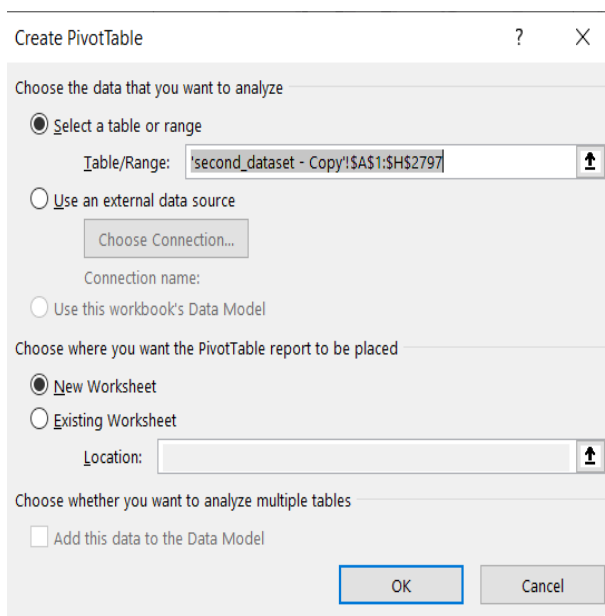


Figure 43:Ok option selected

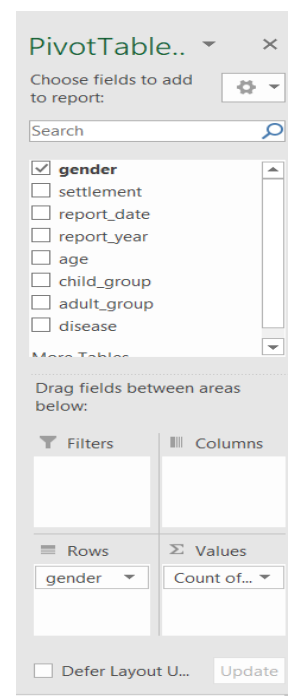


Figure 44:Selected Preferences

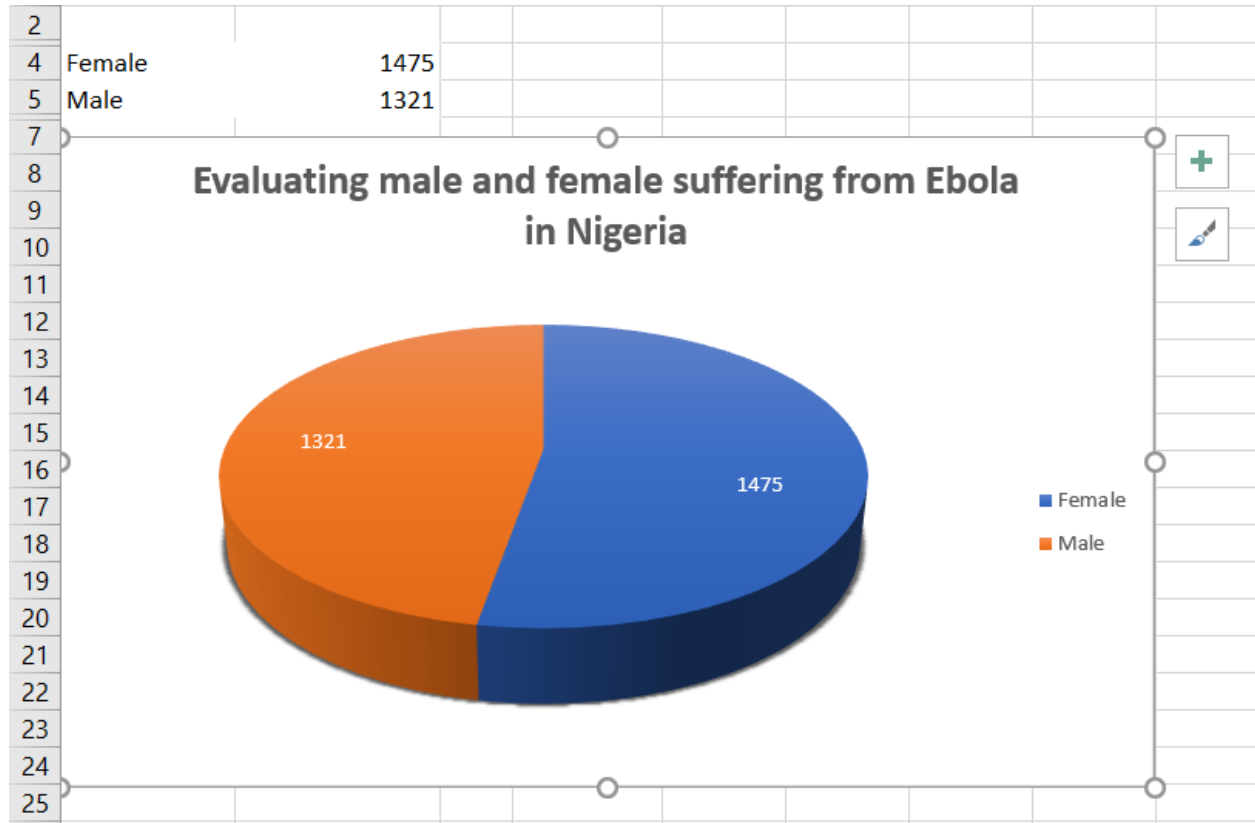


Figure 45: From Chart, Pie chart option selected

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# import missingno as msno
```

Figure 46: Imported python visualization libraries

```
ebola2= pd.read_csv('C:\\Users\\msi\\Downloads\\second_dataset.csv')
ebola2.head()
```

	gender	settlement	report_date	report_year	age	child_group	groups	disease
0	Male	Rural	11/19/2016	2016	3	1	0	Ebola
1	Female	Rural	11/23/2016	2016	10	1	0	Ebola
2	Male	Urban	3/9/2016	2016	9	1	0	Ebola
3	Male	Urban	10/23/2016	2016	66	0	1	Ebola
4	Female	Urban	2/9/2016	2016	35	0	1	Ebola

```
ebola3=ebola2.rename(columns={'adult_group':'groups'})
ebola3.head()
```

```
sns.relplot(x='gender', y='age', hue='age',size='child_group',col='groups', data=ebola3)
```

<seaborn.axisgrid.FacetGrid at 0x1d08b42f3a0>

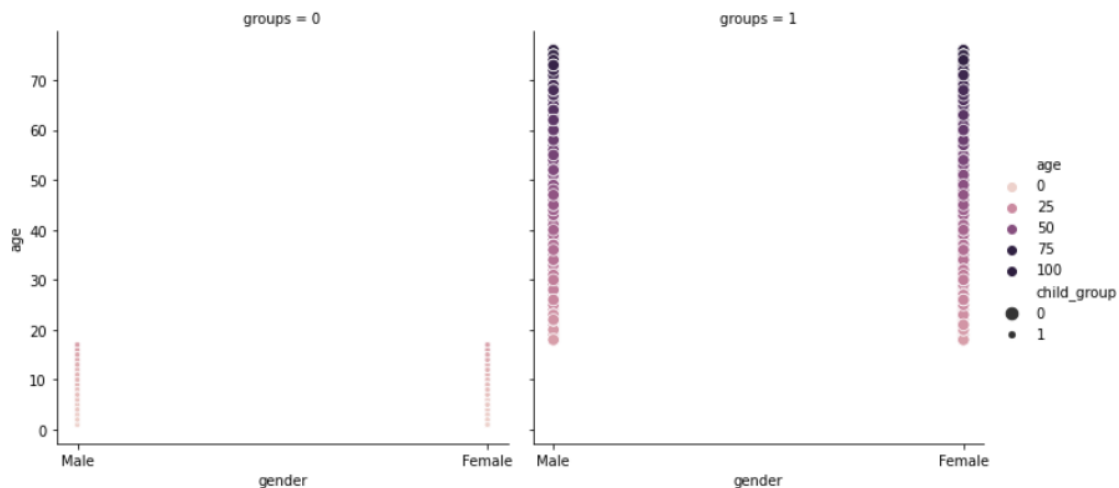
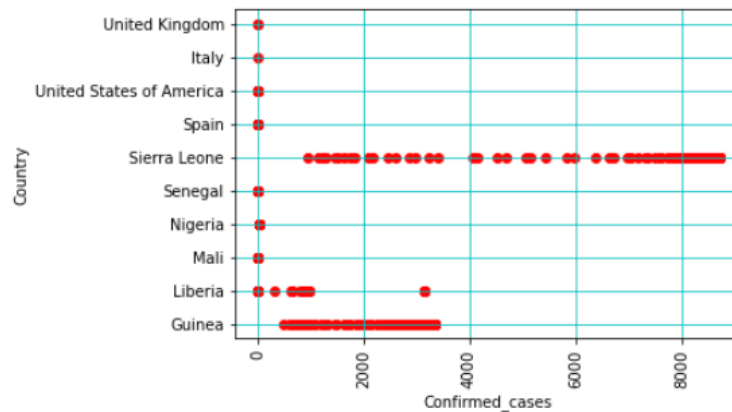


Figure 47:Relplot example

7.3.1 Scatter plot analysis of csv data

```
plt.xlabel('Confirmed_cases')
plt.ylabel('Country')
plt.scatter(ebola.Confirmed_cases,ebola.Country,color='r')
# plt.scatter(ebola.Confirmed_deaths,ebola.Country,color='b')
plt.grid(True,color='c')
plt.xticks(rotation=90)
```

(array([-2000., 0., 2000., 4000., 6000., 8000., 10000.]),
<a list of 7 Text major ticklabel objects>)



```
: plt.xlabel('Confirmed_deaths')
plt.ylabel('Country')
# plt.scatter(ebola.Confirmed_cases,ebola.Country,color='r')
plt.scatter(ebola.Confirmed_deaths,ebola.Country,color='b')
plt.grid(True,color='c')
plt.xticks(rotation=90)
```

: (array([-500., 0., 500., 1000., 1500., 2000., 2500., 3000., 3500.,
4000., 4500.]),
<a list of 11 Text major ticklabel objects>)

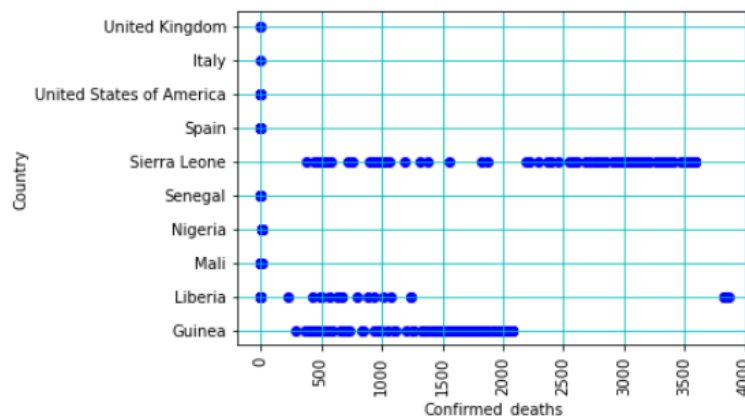


Figure 48:Scatter plot analysis of confirmed cases and deaths

8 References

Beal, V., 2020. *Big Data*. [Online]

Available at: https://www.webopedia.com/TERM/B/big_data.html

[Accessed 13 05 2020].

Johari, A., 2019. *Python NumPy Tutorial – Learn NumPy Arrays With Examples*. [Online]

Available at: edureka.co/blog/python-numpy-tutorial/

[Accessed 15 May 2020].

learnpython.org, 2020. *Pandas Basics*. [Online]

Available at: https://www.learnpython.org/en/Pandas_Basics

[Accessed 15 May 2020].

Lee, C. H. & Yoon, H.-J., 2017. Medical big data: promise and challenges. *Department of Biomedical Engineering, Seoul National University*, 36(1), pp. 3-11.

Peng, L. & Lei, 2004. A review of Missing Data Treatment Methods. *A review of Missing Data Treatment Methods*, Volume 33, pp. 1-8.

Peng, L. & Lei, L., 2004. A Review of Missing Data Treatment Methods. *A Review of Missing Data Treatment Methods*, Volume 33, pp. 1-8.

Rijmenam, D. M. v., 2013. *A Short History of Big Data*. [Online]

Available at: <https://datafloq.com/read/big-data-history/239>

[Accessed 5 May 2020].

