Analysis of Los-Angeles Parking citation Data In Hadoop Environment Using Map-Reduce Techniques

Sindhujan Dhayalan,X17170265
Programming for Data Analytics project
M.Sc Data Analytics - National College of Ireland

***Abstract*—Parking in a busy city like Los-Angeles is troublesome. Chances are that if you drive in Los-Angeles, you are most likely to end up with a parking ticket. Which is a problem when the parking ticket issued in LA averages to $70 in the year 2017 and it doubles when paid late. Statistics show that the city generates large revenues from parking tickets as much that some percentage of the citys budget plan depends on revenues from parking tickets. This project is focused on analyzing big data available on parking tickets in LA open data portal. To analyze the available Big data, Big data processing environment with the combination of Hadoop distributed environment, Map-Reduce programming using Pig and Java, SQL: MySQL and NoSQL: HBase databases for data storage is used. The analysis is focused on finding hidden patterns which can be used to derive knowledge to lessen the burden on people driving caused by parking citations in much populated city like LA. Key findings were even though the parking tickets are expensive, Average ticket pricing has reduced in recent years. The Brand, Body-Style, Colour, Registered state plate of vehicles most involved in parking citations were found. The Top Route and Violation on which parking citations were issued are found.**

***Index Terms*—Hadoop, Map-Reduce, Pig, HBase, MySQL, JAVA, R Programming.**

I. Introduction

Parking tickets are issued for violating parking laws by parking enforcement official set by the State or the city. In cities, highly populated and busy 24/7, parking your vehicle becomes an issue. It becomes hard to find parking space and probably you will end up with a parking ticket. Cities like Los-Angeles, A single parking ticket can cause you more than $65 and late fees double the charges, there are complaints by people for reducing the high rate at parking ticket. It is very hard for everyone driving to pay for expensive parking tickets and people living in the city are unhappy and frustrated about it. But the city depends on high revenues from parking tickets. Published report[1] by City controller Ron Galperin states that LA generated $148 million in gross ticket revenues in the financial year 2015-2016 and Data from Kaggle[2] suggests in the year 2017, People paid a sum of $156 million on parking citations. Most of the revenues go to salaries and administrative costs. The remaining quarter of the revenues is spent on city services through general funds. Ron Galperin has states solutions such as Digital parking signs, Re-evaluation of Street sweeping schedules, Smartphone application on maintaining parking tickets rather than reducing the parking charges. This project is focused on analyzing the data available on Parking citations in LA city over the period of 2010 to 2018 using Hadoop distributed environment combined with MySQL and HBase databases and using Java and Pig Map-reduce programming. The data sourced from Kaggle is downloaded and stored in MySQL database then loaded to HDFS, where it is processed by Map-reduce programs using Java and Pig, later stored in HBase. The processed data is visualized using Rscript to identify hidden patterns, That can provide insights to reduce the burden of expensive parking tickets on people driving in the city of Los-Angeles. Data analysis is focused on identifying insights on following queries:

1) Average parking fine paid on Each day of a month, Each month of a year and Each year in the period of 2010-2018.
2) Total amount of fines paid by vehicles registered with different states apart from CA.
3) Brand of vehicles and their year of incident involved in parking citations.
4) Top 5 Routes, Violations and Month of the incident involved in parking citations.
5) The Body style, Colour, Brand and Routes of vehicles involved in most parking tickets in the year 2018.

The paper is organized as the following in the upcoming sections, Section 2: Related work, Section 3: Methodology, Section 4: Results followed by, Section 5, Conclusion and future work.

II. Related work

Research works using Hadoop MapReduce is mostly focused on Social media analysis, Crime analysis etc. Using Hadoop MapReduce for data analytics of parking citation proposed in this project is a novel approach. But there are many research using the same approach of MapReduce technology applied for this project in other research domains.

Research paper[1] discusses applying Big data analytics on crime data. The framework designed uses Apache sqoop to move data from MySQL database into HDFS.Apache sqoop connects with Relational database management systems and transfers large amount of data from RDBMS to HDFS.Pig Latin Script is used to run MapReduce program on the data stored in HDFS and the output from Pig is stored back to HDFS. The advantages of using Pig Script instead of JAVA script is the less complexity of Pig code. Hence reducing the overall time of development and testing. Pig scripts consume about 5% time compared to writing MapReduce Java programs. But execution time of Pig Scripts are 50% longer

[1] http://parking.controlpanel.la/

[2] https://www.kaggle.com/cityofLA/los-angeles-parking-citations

than that of Java MapReduce programs. The output from Pig MapReduce is later used for Visualization to find hidden patterns.

Paper[2] successfully implemented Big data analytics of E-Commerce consumer data on Hadoop platform to understand consumer consumption behaviour. The unstructured data is stored in HDFS as it provides Scalability and data security. Data stored in HDFS is processed using Hadoop's MapReduce framework. The processed data is then transferred to MySQL using Sqoop with help of shell command. Data stored in MySQL was visualized using technologies such as Eclipse platform, jsp, servlet, jdbc, jFreechart.

Hadoop MapReduce Framework on AWS cloud platform is used for data analysis of YouTube data in paper[3]. YouTube video data is collected using YouTube API and stored as CSV file format. The data collected is then fed to HDFS. MapReduce program coded in JAVA programming language is run on the data to perform summary operations to find the top 5 video categories, top 5 video uploaders etc. This output is stored back in HDFS and used for visualization to provide knowledge to end users.

In research paper[4], Author proposed a model to analyze weather logs quickly using Hadoop parallel processing. the pre-processed weather data is loaded into HDFS and MapReduce framework using Java programming is applied to analyze yearly maximum and minimum temperature. The output file from MapReduce is stored in HBase. The data in HBase is visualized in line charts to display the maximum and minimum temperature. The results were that Hadoop did not handle the integration of large number of small files into Large file as the size of the file was bigger than the HDFS block size.

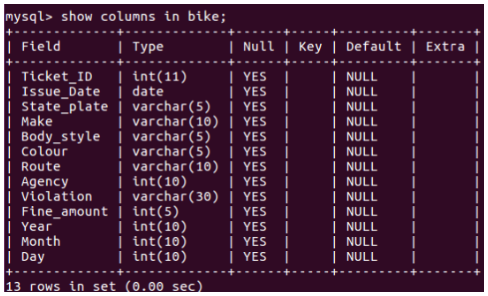
III. Methodology

A. Description of Dataset

The Parking citation Dataset for this project was sourced from Los Angeles open data platform[3], Also available on kaggle[4], the dataset has 9.26 million rows and 19 columns. The 19 columns contain data about the ticket number, Issued date, Issue time, Meter ID, Marked time, RP State plate, Plate expiry date, VIN, Make, Body style, Colour, Location, Route, Agency, Violation code, Violation description, Fine amount, Latitude and Longitude recorded from year 2010 to 2018. The dataset was created on 2018-08-16 and Last updated on 2019-04-23. The Challenges of using dataset is its size as Ms-excel cannot open 9.26 million rows and many records in the dataset contained null and duplicate values.

B. Data Processing

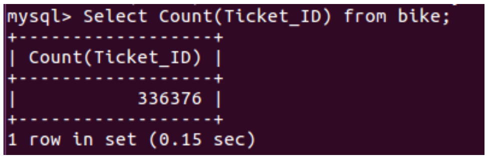
The Dataset was first pre-processed using RScript, Out of the 19 columns, 10 columns was considered for analysis on queries addressed in this project, they are ticket number, Issued date, RP State plate, Make, Body style, Colour, Route, Agency, Violation description, Fine amount. The tasks carried out using RScript were, Records with null values were removed, Column names were modified and Issued date column was divided into 3 separate columns: Year, Month and Day, Which will be useful input for Map-Reduce programs. Further, the pre-processed data is imported into MySQL database. For this purpose, first a database 'Parking' and table 'bike' was created in MySQL. Figure1 shows the created schema of 'bike' in MySQL database 'Parking'.



```
mysql> show columns in bike;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| Ticket_ID   | int(11)     | YES  |     | NULL    |       |
| Issue_Date  | date        | YES  |     | NULL    |       |
| State_plate | varchar(5)  | YES  |     | NULL    |       |
| Make        | varchar(10) | YES  |     | NULL    |       |
| Body_style  | varchar(5)  | YES  |     | NULL    |       |
| Colour      | varchar(5)  | YES  |     | NULL    |       |
| Route       | varchar(10) | YES  |     | NULL    |       |
| Agency      | int(10)     | YES  |     | NULL    |       |
| Violation   | varchar(30) | YES  |     | NULL    |       |
| Fine_amount | int(5)      | YES  |     | NULL    |       |
| Year        | int(10)     | YES  |     | NULL    |       |
| Month       | int(10)     | YES  |     | NULL    |       |
| Day         | int(10)     | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
13 rows in set (0.00 sec)
```

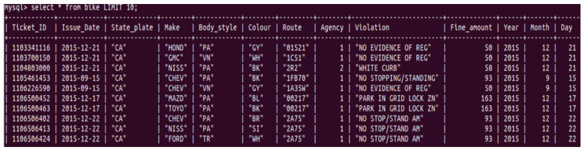
Fig. 1. Schema created in MySQL database

On querying the table, Duplicate Ticket ID records were found, which were removed using MySQL command and finally, A table with 336376 records and 13 fields were produced for use in this project. The Count and Sample of the table created is displayed in figure2 and figure3.



```
mysql> Select Count(Ticket_ID) from bike;
+------------------+
| Count(Ticket_ID) |
+------------------+
|           336376 |
+------------------+
1 row in set (0.15 sec)
```

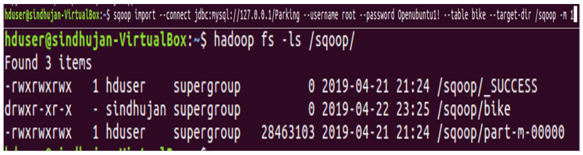
Fig. 2. Table 'bike' count



```
mysql> select * from bike LIMIT 10;
+------------+------------+-------------+--------+------------+--------+---------+--------+------------------------+-------------+------+-------+-----
| Ticket_ID  | Issue_Date | State_plate | Make   | Body_style | Colour | Route   | Agency | Violation              | Fine_amount | Year | Month | Day
+------------+------------+-------------+--------+------------+--------+---------+--------+------------------------+-------------+------+-------+-----
| 1103341116 | 2015-12-21 | "CA"        | "HOND" | "PA"       | "GY"   | "01521" |      1 | "NO EVIDENCE OF REG"   |          50 | 2015 |    12 |  21
| 1103700150 | 2015-12-21 | "CA"        | "GMC"  | "VN"       | "WH"   | "1C51"  |      1 | "NO EVIDENCE OF REG"   |          50 | 2015 |    12 |  21
| 1104803000 | 2015-12-21 | "CA"        | "NISS" | "PA"       | "BK"   | "2R2"   |      2 | "WHITE CURB"           |          58 | 2015 |    12 |  21
| 1105461453 | 2015-09-15 | "CA"        | "CHEV" | "PA"       | "BK"   | "1FB70" |      1 | "NO STOPPING/STANDING" |          93 | 2015 |     9 |  15
| 1106226590 | 2015-09-15 | "CA"        | "CHEV" | "VN"       | "GY"   | "1A35W" |      1 | "NO EVIDENCE OF REG"   |          50 | 2015 |     9 |  15
| 1106500452 | 2015-12-17 | "CA"        | "MAZD" | "PA"       | "BL"   | "00217" |      1 | "PARK IN GRID LOCK ZN" |         163 | 2015 |    12 |  17
| 1106500463 | 2015-12-17 | "CA"        | "TOYO" | "PA"       | "BK"   | "00217" |      1 | "PARK IN GRID LOCK ZN" |         163 | 2015 |    12 |  17
| 1106506402 | 2015-12-22 | "CA"        | "CHEV" | "PA"       | "BR"   | "2A75"  |      1 | "NO STOP/STAND AM"     |          93 | 2015 |    12 |  22
| 1106506413 | 2015-12-22 | "CA"        | "NISS" | "PA"       | "SI"   | "2A75"  |      1 | "NO STOP/STAND AM"     |          93 | 2015 |    12 |  22
| 1106506424 | 2015-12-22 | "CA"        | "FORD" | "TR"       | "WH"   | "2A75"  |      1 | "NO STOP/STAND AM"     |          93 | 2015 |    12 |  22
```

Fig. 3. MySQL table records - Sample

The pre-processed data in MySQL database is moved and stored in HDFS file system using Sqoop as shown in figure4. Sqoop is a tool designed for transferring big data between Hadoop and RDBMS. Sqoop uses MapReduce to import and export the data, providing fault tolerance and parallel processing[5].



```
hduser@sindhujan-VirtualBox:~$ sqoop import --connect jdbc:mysql://127.0.0.1/Parking --username root --password Openubuntu1! --table bike --target-dir /sqoop -m 1
hduser@sindhujan-VirtualBox:~$ hadoop fs -ls /sqoop/
Found 3 items
-rwxrwxrwx   1 hduser    supergroup          0 2019-04-21 21:24 /sqoop/_SUCCESS
drwxr-xr-x   - sindhujan supergroup          0 2019-04-22 23:25 /sqoop/bike
-rwxrwxrwx   1 hduser    supergroup   28463103 2019-04-21 21:24 /sqoop/part-m-00000
```

Fig. 4. MySQL to HDFS using SQOOP

[3] https://data.lacity.org/A-Well-Run-City/Parking-Citations/wjz9-h9np

[4] https://www.kaggle.com/cityofLA/los-angeles-parking-citations

The dataset in HDFS was processed using 3 Java Map-Reduce programs using eclipse platform and 2 Pig Map-Reduce programs on the basis of the queries to be addressed in the project. Output from the Map-Reduce programs were stored in HDFS and then transferred to HBase. The files are then exported to perform visualization using RScript. Finally the visualization produced was observed and knowledge were derived to answer the queries mentioned in section I. The process flow of the project is displayed in Figure5.

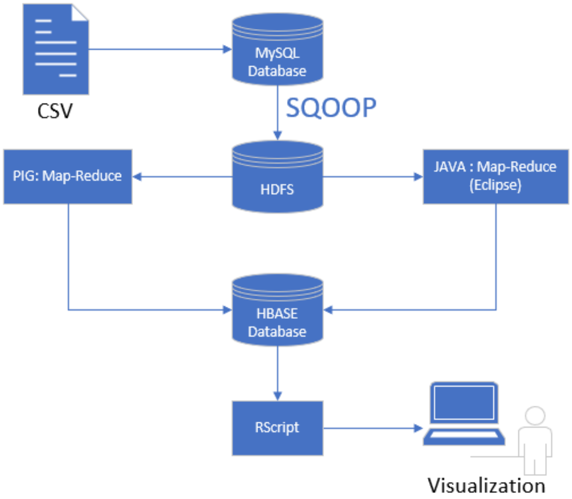


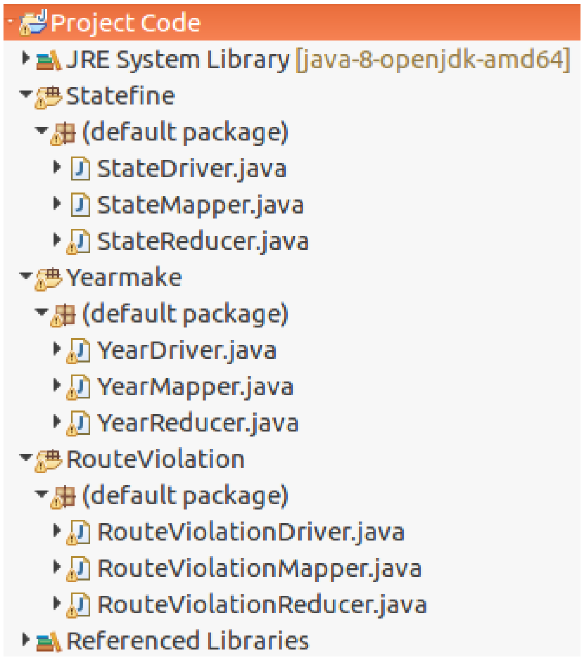
Fig. 5. Process flow

MapReduce tasks performed using JAVA program and Pig Scripts are explained in the following sections.

C. JAVA Map-Reduce

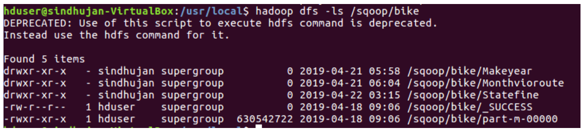
MapReduce is a programming model used to process big data. The Map function process key/value pair as input and generate a set of intermediate key/value pairs. Then the reduce function merges all intermediate values associated with same intermediate key. Google uses MapReduce algorithms for processing graphs, texts, Machine learning and performing statistical operations. MapReduce is fault tolerant and is a very useful tool in big data analysis[6]. For this project, JAVA Map-Reduce programs were performed using Eclipse-Hadoop integration. The input file path is from the HDFS and the output file of Eclipse is stored back in HDFS for the 3 tasks discussed below:
Three Classes: Driver, Mapper and Reducer were used to perform the Map-Reduce task. Figure6 displays the classes created for performing the tasks.

Task1 - State & Fine
Objective of the task is to find the total amount of fine payed by vehicles with different registration state plate. For this purpose, Words in the fields 'State' as Key and the total amount of fine paid by that state is produced as value output.
Input : State and Fine & Output: Key: State & Value: Fine
Task2 - Count(Make,Year)
Objective of the task is to count different Make of vehicle reported in each year on parking citation. For this purpose, Words in the fields 'Make' and 'Year' are concatenated as Key and their occurrence together is counted and produced as value output.
Input : Make and Year
Output: Key: (Make,Year) & Value: Count(Make,Year)
Task3 - Count(Month,Violence,Route)
Objective of the task is to count number of parking violation reported in every route in each month. For this purpose, Words in the fields 'Month','Violation' and 'Route' are concatenated as Key and their occurrence together is counted and produced as value output.
Input : Month, Violation and Route
Output:Key:(Month,Violation,Route)
Value: Count(Month,Violation,Route)



```
Project Code
  JRE System Library [java-8-openjdk-amd64]
  Statefine
    (default package)
      StateDriver.java
      StateMapper.java
      StateReducer.java
  Yearmake
    (default package)
      YearDriver.java
      YearMapper.java
      YearReducer.java
  RouteViolation
    (default package)
      RouteViolationDriver.java
      RouteViolationMapper.java
      RouteViolationReducer.java
  Referenced Libraries
```

Fig. 6. Java Map-Reduce Classes in Eclipse

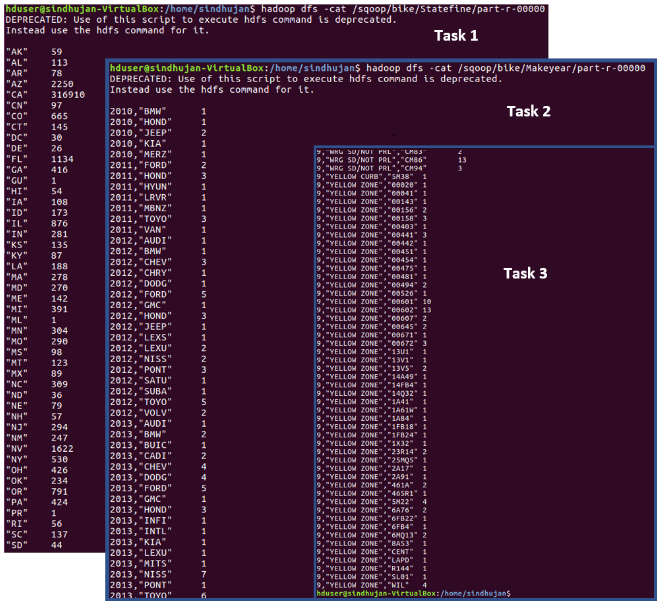


```
hduser@sindhujan-VirtualBox:/usr/local$ hadoop dfs -ls /sqoop/bike
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Found 5 items
drwxr-xr-x   - sindhujan supergroup          0 2019-04-21 05:58 /sqoop/bike/Makeyear
drwxr-xr-x   - sindhujan supergroup          0 2019-04-21 06:04 /sqoop/bike/Monthvioroute
drwxr-xr-x   - sindhujan supergroup          0 2019-04-22 03:15 /sqoop/bike/Statefine
-rw-r--r--   1 hduser    supergroup          0 2019-04-18 09:06 /sqoop/bike/_SUCCESS
-rwxr-xr-x   1 hduser    supergroup  630542722 2019-04-18 09:06 /sqoop/bike/part-m-00000
```

Fig. 7. Output of Java Map-Reduce stored in HDFS

D. Pig Map-Reduce

Apache Pig is a high level language platform used for analyzing large data sets. Pig Latin can execute Hadoop jobs using MapReduce[7]. Pig-MapReduce process was carried out by creating two Pig Latin scripts for Task 4(pig_cmd.pig) and Task5(pig_cmdcount.pig) and the scripts were run on Apache Pig Grunt shell. The input path was main source dataset stored in HDFS and Output was stored back in HDFS system.



```
hduser@sindhujan-VirtualBox:/home/sindhujan$ hadoop dfs -cat /sqoop/bike/Statefine/part-r-00000
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

"AK"    59
"AL"    113
"AR"    78
"AZ"    2250
"CA"    316910
"CN"    97
"CO"    665
"CT"    145
"DC"    30
"DE"    26
"FL"    1134
"GA"    416
"GU"    1
"HI"    54
"IA"    108
"ID"    173
"IL"    876
"IN"    281
"KS"    135
"KY"    87
"LA"    188
"MA"    278
"MD"    270
"ME"    142
"MI"    391
"ML"    1
"MN"    304
"MO"    290
"MS"    98
"MT"    123
"MX"    89
"NC"    309
"ND"    36
"NE"    79
"NH"    57
"NJ"    294
"NM"    247
"NV"    1622
"NY"    530
"OH"    426
"OK"    234
"OR"    791
"PA"    424
"PR"    1
"RI"    56
"SC"    137
"SD"    44
```

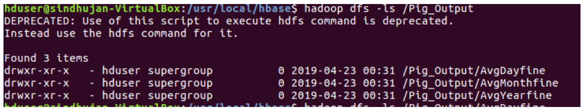
```
hduser@sindhujan-VirtualBox:/home/sindhujan$ hadoop dfs -cat /sqoop/bike/Makeyear/part-r-00000
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

2010,"BMW"      1
2010,"HOND"     1
2010,"JEEP"     2
2010,"KIA"      1
2010,"MERZ"     1
2011,"FORD"     2
2011,"HOND"     3
2011,"HYUN"     1
2011,"LRVR"     1
2011,"MBNZ"     1
2011,"TOYO"     3
2011,"VAN"      1
2012,"AUDI"     1
2012,"BMW"      1
2012,"CHEV"     3
2012,"CHRY"     1
2012,"DODG"     1
2012,"FORD"     5
2012,"GMC"      1
2012,"HOND"     3
2012,"JEEP"     1
2012,"LEXS"     1
2012,"LEXU"     2
2012,"NISS"     2
2012,"PONT"     3
2012,"SATU"     1
2012,"SUBA"     1
2012,"TOYO"     5
2012,"VOLV"     2
2013,"AUDI"     1
2013,"BMW"      2
2013,"BUIC"     1
2013,"CADI"     2
2013,"CHEV"     4
2013,"DODG"     4
2013,"FORD"     5
2013,"GMC"      1
2013,"HOND"     3
2013,"INFI"     1
2013,"INTL"     1
2013,"KIA"      1
2013,"LEXU"     1
2013,"MITS"     1
2013,"NISS"     7
2013,"PONT"     1
2013,"TOYO"     6
```

```
9,"WRG SD/NOT PRL","CM83"       2
9,"WRG SD/NOT PRL","CM86"       13
9,"WRG SD/NOT PRL","CM94"       3
9,"YELLOW CURB","5M38"  1
9,"YELLOW ZONE","00020" 1
9,"YELLOW ZONE","00041" 1
9,"YELLOW ZONE","00143" 1
9,"YELLOW ZONE","00156" 2
9,"YELLOW ZONE","00158" 3
9,"YELLOW ZONE","00403" 1
9,"YELLOW ZONE","00441" 3
9,"YELLOW ZONE","00442" 1
9,"YELLOW ZONE","00451" 1
9,"YELLOW ZONE","00454" 1
9,"YELLOW ZONE","00475" 1
9,"YELLOW ZONE","00481" 1
9,"YELLOW ZONE","00494" 2
9,"YELLOW ZONE","00526" 1
9,"YELLOW ZONE","00601" 10
9,"YELLOW ZONE","00602" 13
9,"YELLOW ZONE","00607" 2
9,"YELLOW ZONE","00645" 2
9,"YELLOW ZONE","00671" 1
9,"YELLOW ZONE","00672" 3
9,"YELLOW ZONE","13U1"  1
9,"YELLOW ZONE","13V1"  1
9,"YELLOW ZONE","13V5"  2
9,"YELLOW ZONE","14A49" 1
9,"YELLOW ZONE","14FB4" 1
9,"YELLOW ZONE","14Q32" 1
9,"YELLOW ZONE","1A41"  1
9,"YELLOW ZONE","1A61W" 1
9,"YELLOW ZONE","1A84"  1
9,"YELLOW ZONE","1FB18" 1
9,"YELLOW ZONE","1FB24" 1
9,"YELLOW ZONE","1X32"  1
9,"YELLOW ZONE","23R14" 2
9,"YELLOW ZONE","25MQ5" 1
9,"YELLOW ZONE","2A17"  1
9,"YELLOW ZONE","2A91"  1
9,"YELLOW ZONE","461A"  2
9,"YELLOW ZONE","465R1" 1
9,"YELLOW ZONE","5M22"  4
9,"YELLOW ZONE","6A76"  2
9,"YELLOW ZONE","6FB22" 1
9,"YELLOW ZONE","6FB4"  1
9,"YELLOW ZONE","6MQ13" 2
9,"YELLOW ZONE","8A53"  1
9,"YELLOW ZONE","CENT"  1
9,"YELLOW ZONE","LAPD"  1
9,"YELLOW ZONE","R144"  1
9,"YELLOW ZONE","SL01"  1
9,"YELLOW ZONE","WIL"   4
hduser@sindhujan-VirtualBox:/home/sindhujan$
```

Fig. 8. Output of Task1, Task2 and Task3

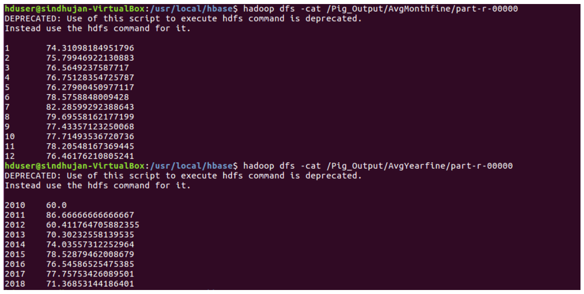
Task4 - Average Fine on Days, Months and Year

Objective of Task 4 is to calculate the average fine paid on daily, Monthly and yearly basis over a period from 2010 to 2018. Three output files were stored in HDFS in this task for each Days, Months and Years respectively. Figure9 displays the path of output files stored in HDFS. Figure10 and Figure11 displays the Output of Task 4 Map-Reduce program.



```
hduser@sindhujan-VirtualBox:/usr/local/hbase$ hadoop dfs -ls /Pig_Output
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Found 3 items
drwxr-xr-x   - hduser supergroup          0 2019-04-23 00:31 /Pig_Output/AvgDayfine
drwxr-xr-x   - hduser supergroup          0 2019-04-23 00:31 /Pig_Output/AvgMonthfine
drwxr-xr-x   - hduser supergroup          0 2019-04-23 00:31 /Pig_Output/AvgYearfine
```

Fig. 9. Pig-Task4 Output files stored in HDFS

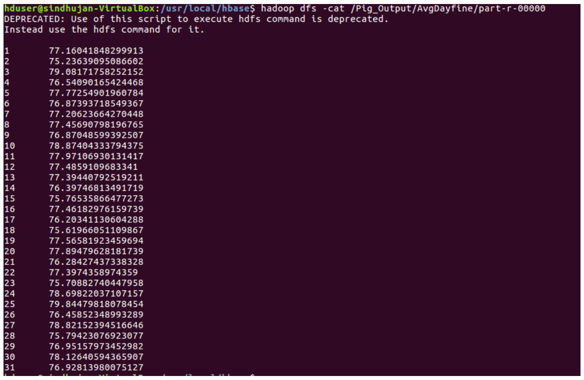


```
hduser@sindhujan-VirtualBox:/usr/local/hbase$ hadoop dfs -cat /Pig_Output/AvgMonthfine/part-r-00000
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

1       74.31098184951796
2       75.79946922130883
3       76.5649237587717
4       76.75128354725787
5       76.27900450977117
6       78.5758848009428
7       82.28599292388643
8       79.69558162177199
9       77.43357123250068
10      77.71493536720736
11      78.20548167369445
12      76.46176210805241
hduser@sindhujan-VirtualBox:/usr/local/hbase$ hadoop dfs -cat /Pig_Output/AvgYearfine/part-r-00000
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

2010    60.0
2011    86.66666666666667
2012    60.411764705882355
2013    70.30232558139535
2014    74.03557312252964
2015    78.52879462008679
2016    76.54586525475385
2017    77.75753426089501
2018    71.36853144186401
```

Fig. 10. Pig-Task4 Average fine on each Month & Year

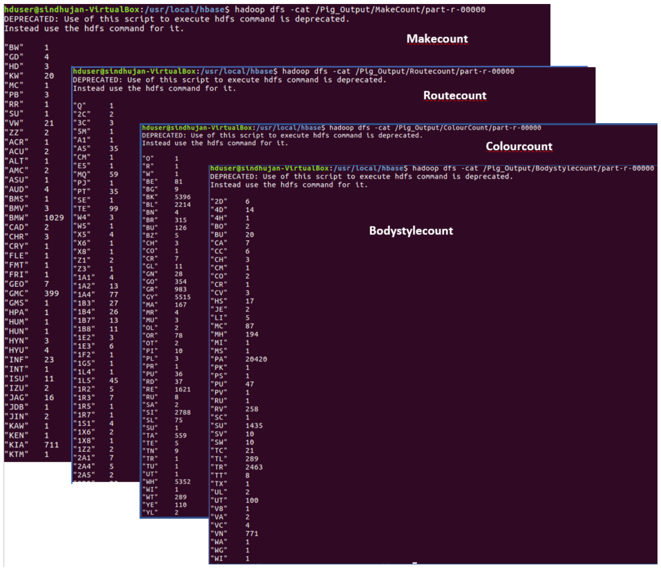
Task5 - Count of Body-style, Make, Colour & Route

Objective of Task5 is to calculate the Count of Body-style, Make, Colour and Route of vehicles involved in parking violation in the year 2018. Four output files were stored in HDFS in this task for Body-style, Make, Colour and Route respectively. Figure12 displays Collage screen-shots of Output of MapReduce performed in Task5.



```
hduser@sindhujan-VirtualBox:/usr/local/hbase$ hadoop dfs -cat /Pig_Output/AvgDayfine/part-r-00000
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

1       77.16041848299913
2       75.23639095086602
3       79.08171758252152
4       76.54090165424468
5       77.77254901960784
6       76.87393718549367
7       77.20623664270448
8       77.45690798196765
9       76.87048599392507
10      78.87404333794375
11      77.97106930131417
12      77.4859109683341
13      77.39440792519211
14      76.39746813491719
15      75.76535866477273
16      77.46182976159739
17      76.20341130604288
18      75.61966051109867
19      77.56581923459694
20      77.89479628181739
21      76.28427437338328
22      77.3974358974359
23      75.70882740447958
24      78.69822037107157
25      79.84479818078454
26      76.45852348993289
27      78.82152394516646
28      75.79423076923077
29      76.95157973452982
30      78.12640594365907
31      76.92813980075127
```

Fig. 11. Pig-Task4 Average fine on each day of the month



```
hduser@sindhujan-VirtualBox:/usr/local/hbase$ hadoop dfs -cat /Pig_Output/MakeCount/part-r-00000
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

"BW"    1
"GD"    4
"HD"    3
"KW"    20
"MC"    1
"PB"    3
"RR"    1
"SU"    1
"VW"    21
"ZZ"    2
"ACR"   1
"ACU"   2
"ALT"   1
"AMC"   2
"ASU"   1
"AUD"   4
"BMS"   1
"BMV"   3
"BMW"   1029
"CAD"   2
"CHR"   3
"CRY"   1
"FLE"   1
"FMT"   1
"FRI"   1
"GEO"   7
"GMC"   399
"GMS"   1
"HPA"   1
"HUM"   1
"HUN"   1
"HYN"   3
"HYU"   4
"INF"   23
"INT"   1
"ISU"   11
"IZU"   2
"JAG"   16
"JDB"   1
"JIN"   2
"KAW"   1
"KEN"   1
"KIA"   711
"KTM"   1
```

```
hduser@sindhujan-VirtualBox:/usr/local/hbase$ hadoop dfs -cat /Pig_Output/Routecount/part-r-00000
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

"Q"     1
"2C"    2
"3C"    3
"5M"    1
"A1"    1
"A5"    35
"CM"    1
"E5"    1
"MQ"    59
"PJ"    1
"PT"    35
"SE"    1
"TE"    99
"W4"    3
"W5"    1
"X5"    4
"X6"    1
"X8"    1
"Z1"    2
"Z3"    1
"1A1"   4
"1A2"   13
"1A4"   77
"1B3"   27
"1B4"   26
"1B7"   13
"1B8"   11
"1E2"   3
"1E3"   6
"1F2"   1
"1G5"   1
"1L4"   1
"1L5"   45
"1R2"   5
"1R3"   7
"1R5"   1
"1R7"   1
"1S1"   4
"1X6"   2
"1X8"   1
"1Z2"   2
"2A1"   7
"2A4"   5
"2A5"   2
```

```
hduser@sindhujan-VirtualBox:/usr/local/hbase$ hadoop dfs -cat /Pig_Output/ColourCount/part-r-00000
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

"O"     1
"R"     1
"W"     1
"BE"    81
"BG"    9
"BK"    5396
"BL"    2214
"BN"    4
"BR"    315
"BU"    126
"BZ"    5
"CH"    3
"CO"    1
"CR"    7
"GL"    11
"GN"    28
"GO"    354
"GR"    983
"GY"    5515
"MA"    167
"MR"    4
"MU"    3
"OL"    2
"OR"    78
"OT"    2
"PI"    10
"PL"    3
"PR"    1
"PU"    36
"RD"    37
"RE"    1621
"RU"    8
"SA"    2
"SI"    2788
"SL"    75
"SU"    1
"TA"    559
"TE"    5
"TN"    9
"TR"    1
"TU"    1
"UT"    1
"WH"    5352
"WI"    1
"WT"    289
"YE"    110
"YL"    2
```

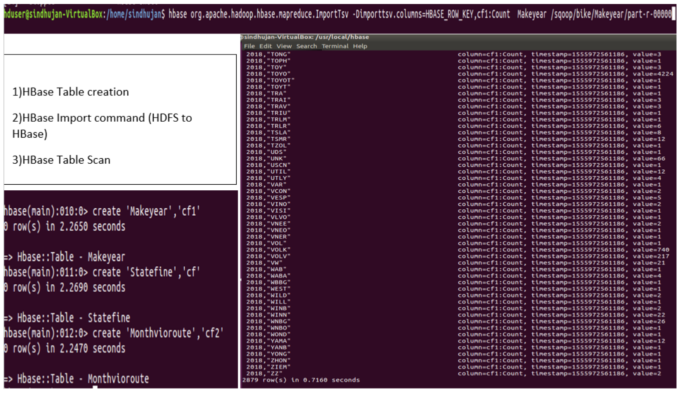
```
hduser@sindhujan-VirtualBox:/usr/local/hbase$ hadoop dfs -cat /Pig_Output/Bodystylecount/part-r-00000
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

"2D"    6
"4D"    14
"4H"    1
"BO"    2
"BU"    20
"CA"    7
"CC"    6
"CH"    3
"CM"    1
"CO"    2
"CR"    1
"CV"    3
"HS"    17
"JE"    2
"LI"    5
"MC"    87
"MH"    194
"MI"    1
"MS"    1
"PA"    20420
"PK"    1
"PS"    1
"PU"    47
"PV"    1
"RU"    1
"RV"    258
"SC"    1
"SU"    1435
"SV"    10
"SW"    10
"TC"    21
"TL"    289
"TR"    2463
"TT"    8
"TX"    1
"UL"    2
"UT"    100
"V8"    1
"VA"    2
"VC"    4
"VN"    771
"WA"    1
"WG"    1
"WI"    1
```

Fig. 12. Pig-Task5 Output

E. Apache HBase

HBase is an open source, NoSQL column based database written in Java. It runs on top of HDFS (Hadoop distributed file system). It is mainly used to store large data with sparsity in a fault tolerant way. Data is stored in column based model in the form key/Value pair and MapReduce can be used to query the data. All the columns made are grouped into one column family[8].

In this project, The Output from the Map-Reduce programs stored in HDFS is moved to HBase Database. Initially, HBase tables are created with table name and column family. Later, using the ImportTsv command, The data in HDFS file is stored into the HBase table. Figure13 displays the Commands used for importing the data into the table and imported data present in the HBase table.



```
hduser@sindhujan-VirtualBox:/home/sindhujan$ hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.columns=HBASE_ROW_KEY,cf1:Count  Makeyear /sqoop/bike/Makeyear/part-r-00000
```

1)HBase Table creation

2)HBase Import command (HDFS to HBase)

3)HBase Table Scan

```
hbase(main):010:0> create 'Makeyear','cf1'
0 row(s) in 2.2650 seconds

=> Hbase::Table - Makeyear
hbase(main):011:0> create 'Statefine','cf'
0 row(s) in 2.2690 seconds

=> Hbase::Table - Statefine
hbase(main):012:0> create 'Monthvioroute','cf2'
0 row(s) in 2.2470 seconds

=> Hbase::Table - Monthvioroute
```

sindhujan-VirtualBox: /usr/local/hbase
File Edit View Search Terminal Help

```
 2018,"TONG"                                       column=cf1:Count, timestamp=1555972561186, value=3
 2018,"TOPH"                                       column=cf1:Count, timestamp=1555972561186, value=1
 2018,"TOY"                                        column=cf1:Count, timestamp=1555972561186, value=3
 2018,"TOYO"                                       column=cf1:Count, timestamp=1555972561186, value=4224
 2018,"TOYOT"                                      column=cf1:Count, timestamp=1555972561186, value=1
 2018,"TOYT"                                       column=cf1:Count, timestamp=1555972561186, value=1
 2018,"TRA"                                        column=cf1:Count, timestamp=1555972561186, value=1
 2018,"TRAI"                                       column=cf1:Count, timestamp=1555972561186, value=3
 2018,"TRAV"                                       column=cf1:Count, timestamp=1555972561186, value=3
 2018,"TRIU"                                       column=cf1:Count, timestamp=1555972561186, value=1
 2018,"TRLM"                                       column=cf1:Count, timestamp=1555972561186, value=1
 2018,"TRLR"                                       column=cf1:Count, timestamp=1555972561186, value=6
 2018,"TSLA"                                       column=cf1:Count, timestamp=1555972561186, value=8
 2018,"TSMR"                                       column=cf1:Count, timestamp=1555972561186, value=12
 2018,"TZOL"                                       column=cf1:Count, timestamp=1555972561186, value=1
 2018,"UDS"                                        column=cf1:Count, timestamp=1555972561186, value=1
 2018,"UNK"                                        column=cf1:Count, timestamp=1555972561186, value=66
 2018,"USCN"                                       column=cf1:Count, timestamp=1555972561186, value=1
 2018,"UTIL"                                       column=cf1:Count, timestamp=1555972561186, value=12
 2018,"UTLY"                                       column=cf1:Count, timestamp=1555972561186, value=4
 2018,"VAR"                                        column=cf1:Count, timestamp=1555972561186, value=1
 2018,"VCON"                                       column=cf1:Count, timestamp=1555972561186, value=2
 2018,"VESP"                                       column=cf1:Count, timestamp=1555972561186, value=5
 2018,"VINO"                                       column=cf1:Count, timestamp=1555972561186, value=2
 2018,"VISI"                                       column=cf1:Count, timestamp=1555972561186, value=1
 2018,"VLVO"                                       column=cf1:Count, timestamp=1555972561186, value=1
 2018,"VNEE"                                       column=cf1:Count, timestamp=1555972561186, value=2
 2018,"VNEO"                                       column=cf1:Count, timestamp=1555972561186, value=1
 2018,"VNER"                                       column=cf1:Count, timestamp=1555972561186, value=1
 2018,"VOL"                                        column=cf1:Count, timestamp=1555972561186, value=1
 2018,"VOLK"                                       column=cf1:Count, timestamp=1555972561186, value=740
 2018,"VOLV"                                       column=cf1:Count, timestamp=1555972561186, value=217
 2018,"VW"                                         column=cf1:Count, timestamp=1555972561186, value=21
 2018,"WAB"                                        column=cf1:Count, timestamp=1555972561186, value=1
 2018,"WABA"                                       column=cf1:Count, timestamp=1555972561186, value=4
 2018,"WBBG"                                       column=cf1:Count, timestamp=1555972561186, value=1
 2018,"WEST"                                       column=cf1:Count, timestamp=1555972561186, value=1
 2018,"WILD"                                       column=cf1:Count, timestamp=1555972561186, value=2
 2018,"WILL"                                       column=cf1:Count, timestamp=1555972561186, value=1
 2018,"WINB"                                       column=cf1:Count, timestamp=1555972561186, value=2
 2018,"WINN"                                       column=cf1:Count, timestamp=1555972561186, value=22
 2018,"WNBG"                                       column=cf1:Count, timestamp=1555972561186, value=26
 2018,"WNBO"                                       column=cf1:Count, timestamp=1555972561186, value=1
 2018,"WOND"                                       column=cf1:Count, timestamp=1555972561186, value=1
 2018,"YAMA"                                       column=cf1:Count, timestamp=1555972561186, value=12
 2018,"YANB"                                       column=cf1:Count, timestamp=1555972561186, value=1
 2018,"YONG"                                       column=cf1:Count, timestamp=1555972561186, value=1
 2018,"ZHON"                                       column=cf1:Count, timestamp=1555972561186, value=1
 2018,"ZIEM"                                       column=cf1:Count, timestamp=1555972561186, value=1
 2018,"ZZ"                                         column=cf1:Count, timestamp=1555972561186, value=2
2879 row(s) in 0.7160 seconds
```

Fig. 13. HBase Table creation & Imported data in HBase table

IV. Results

The Output files stored in HBase Database were visualized using RScript to gain knowledge, that was used to address the queries mentioned in section I. This section discusses results of visualization of the outputs from MapReduce tasks.

A. Task 1 - State & Fine

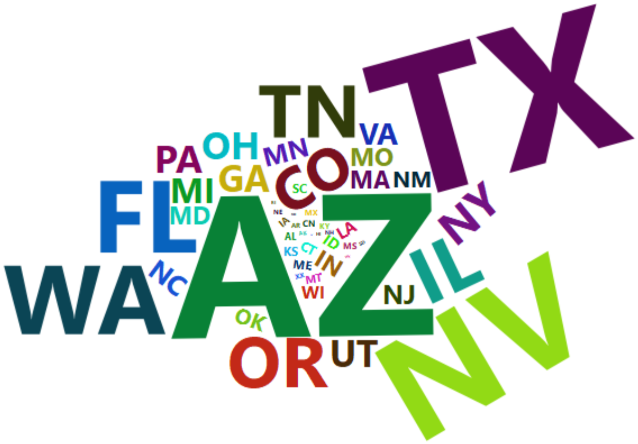


Fig. 14. Word Cloud of State plates with most fines

From the output of Task1, It was observed that vehicles registered with 'CA' state plate had 316910 records (10 times) more parking violations than other states. This is not surprising as the dataset is based on the Parking citations in Los-Angeles which is a city in CA. Being an outlier this value was ignored while visualizing and Aim was to find the vehicles registered outside CA having more parking violation in Los Angeles. Observing Word cloud in figure14, It is inferred that vehicles with state plate registered in AZ, TX and NV are vehicles involved in parking violations apart from CA.

B. Task 2 - Make & Year

Task 2 output file consists of the count of Year and Make of vehicle with most parking violations. Word cloud was again used to find the most occurring terms and it was observed from figure15 that vehicles of brands 'FORD', 'CHEV' and 'NIS' were mostly involved in parking violations and Parking violations were more frequent in year 2015 compared to other years.

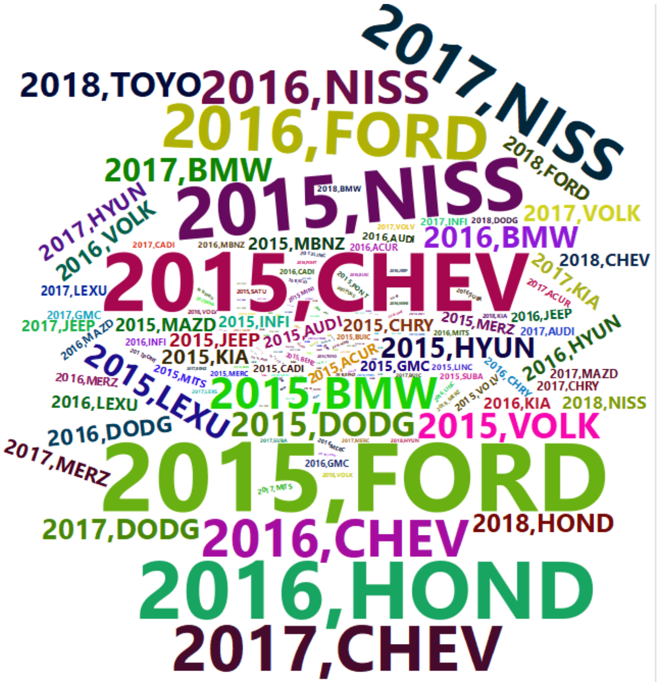


Fig. 15. Word Cloud of Make,Year with most fines

C. Task 3 - Month, Violation & Route

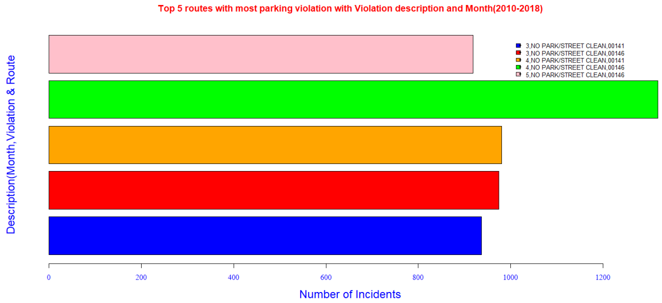


Fig. 16. Horizontal bar-plot of top 5 routes with most parking violation with violation description and Month

Task 3 objective was to find the frequently occurring parking violation incident, its location and the month at which these incident happen. It is inferred from the figure16 that 'NO PARK/STREET CLEAN' occurs frequently in the routes '00141' & '00146' around the months between March to May over the period from 2010 to 2018.

D. Task 4 - Average fine on each Day, Month and Year

Task 4 is carried out to find the Day of the Month, Month of the Year and Year among other Years with most average fine. From the line graphs visualized, It can be inferred that, As of the Day of the Month in figure17, the Average fine value

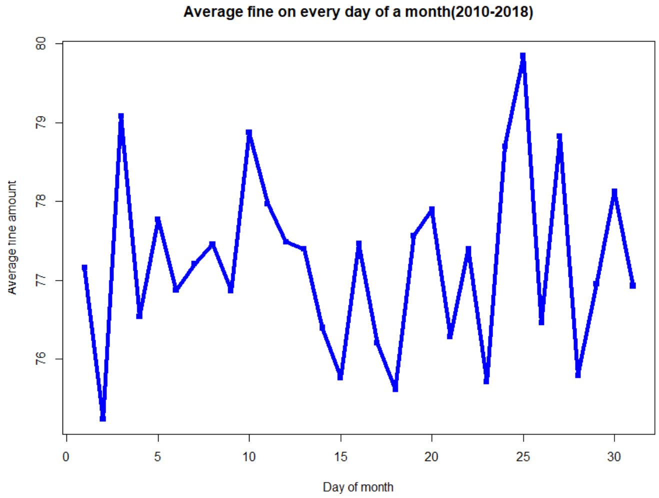


Fig. 17. Average fine every day of Month

is unstable, But the most fine is paid on 25th date of every month.

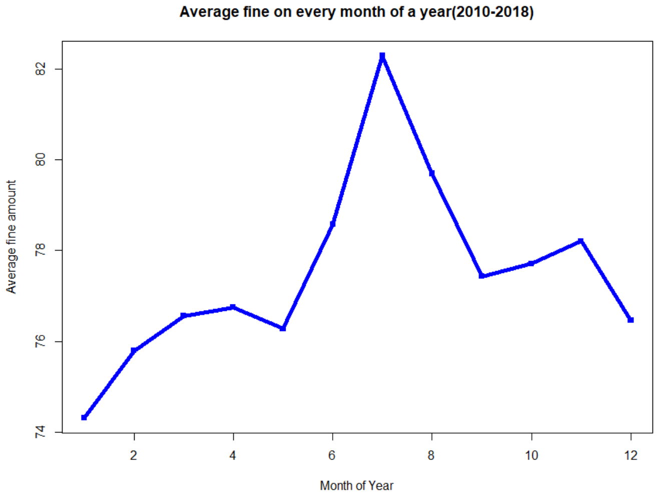


Fig. 18. Average fine on every month of year

Figure18 shows that Most of the fines are paid on months from July to August and July being the month fines are paid the most. Figure19 shows that the average amount of fines paid were highest in 2011 and are gradually decreasing in the recent years from 2016 to 2018.

E. Task 5 - Body-style, Make, Colour & Route

Task 5 output file contains the count of Body-style, Make and colour of vehicles involved in parking violation incidents and the routes in the violation incidents have happened in year 2018. Figure20 shows top 10 body-style of vehicles involved in parking violation in year 2018 and it is inferred that most of vehicles with body style 'PA'(20420 records) commit about 10 times more parking violations than other body-style of vehicles.

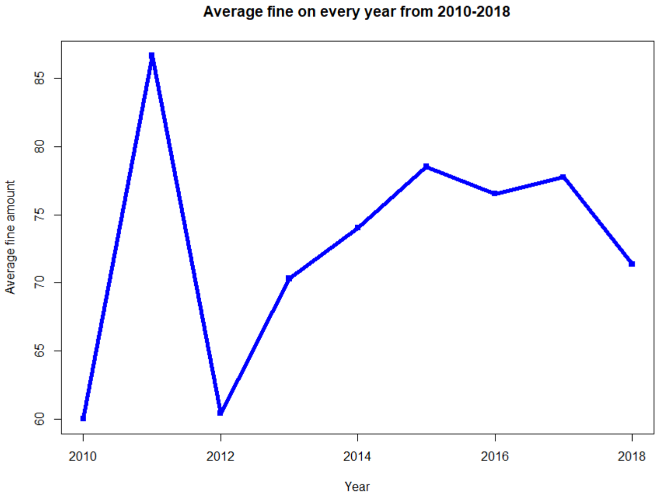


Fig. 19. Average fine on every year(2010-2018)

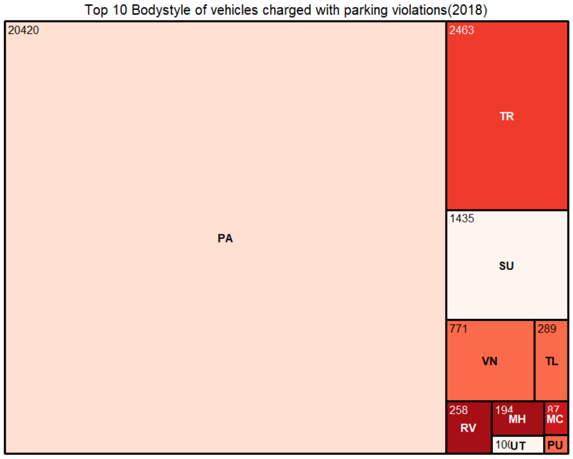


Fig. 20. Top 10 Body-style of vehicles with most parking violations(2018)

Figure21 shows the top 5 colours of vehicles involved in most parking violations and it is inferred from the circle packing that vehicles with colour black, white and grey are involved the most in year 2018.

From Bar chart in figure22, it is inferred that people using 'TOYOTA(TOYO)' brand of vehicles are having trouble with parking the most in the year 2018. the gap between TOYOTA and other brands are significantly large(¿1000). Figure23 displays the Top 10 routes with most parking violations, it is inferred from the graphical representation that Route '00500' has 2719 parking citations in year 2018, The value is 1500 incident more than the second route '00001'.

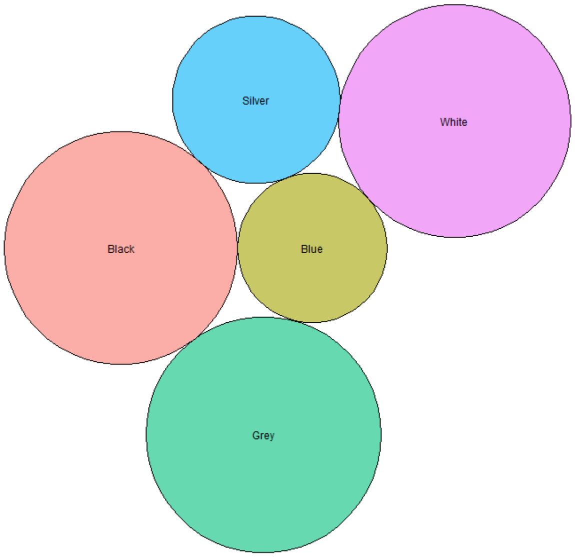


Fig. 21. Top 5 Colour of vehicles involved in Parking violations(2018)

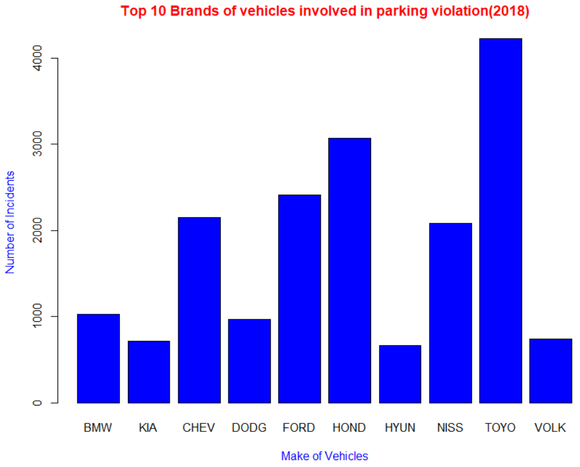


Fig. 22. Top 10 Brands of vehicles involved in Parking violations(2018)

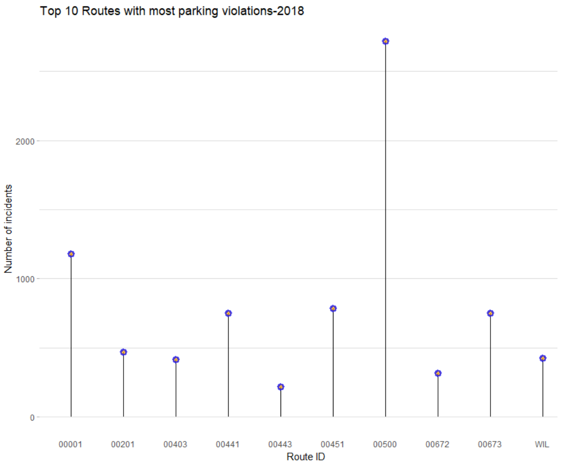


Fig. 23. Top 10 Routes with most Parking violations(2018)

V. Conclusion and Future Work

From the previous section, a lot of insights were gathered, Visualizing Data showed many interesting results. Apart from ’CA’ registered vehicles, vehicles registered under ’AZ’,’TX’ and ’NV’ commit the most parking violation in Los-Angeles. The most violations are made by ’CHEV’,’FORD’ and ’NISS’ and frequently in the year 2015. Most violations made were ’NO PARK/STREET CLEAN’ in the routes ’00141’ and ’00146’ around the months March to May in year range(2010 to 2018). It was found that average fines paid were highest on 25th of each month, Months of July had highest average paid fine and In year 2011, people were paying on average 86 American dollars per parking violation ticket, which has gradually decreased in the recent years. Visualizing data on recent year of 2018, it was found that people using vehicles designed with ’PA’ body commit the most parking violations, vehicles colour with either Black, Grey or White are involved the most in parking violations. ’TOYOTA(TOYO)’ vehicle users were involved the most in Parking violations and the most violations occurred in Route ’00500’. The insights from recent years is that parking violations have decreased compared to previous years. Precautionary measures have been taken on routes ’00141’ and ’00146’ which are overall most parking violated routes, but they do not appear in the top 10 parking violated routes in 2018. Using the Hadoop MapReduce environment, we were successfully able to answer all the queries mentioned in the section I. The advantage here is most of the contributors to parking violations at the 1st place have a large gap from the other contributors, hence the government can focus on reducing the causes of parking violation by targeting the top contributors in each attributes as priority. thereby acting on the top most contributors can significantly reduce the overall parking violation problems. Challenges faced were the processing of 9 million rows. the system performance was low while processing of huge amount of data. Future work can be done on analyzing other attributes such as Agency, Issued time, latitude, longitude which were not used in this project to get more interesting knowledge and discover hidden patterns causing parking violations. In future, an approach to use the entire 9 million rows can be attempted.

References

[1] A. Jain and V. Bhatnagar, “Crime data analysis using pig with hadoop,” *Procedia Computer Science*, vol. 78, pp. 571 – 578, 2016, 1st International Conference on Information Security Privacy 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S187705091600106X

[2] F. Shao and J. Yao, “The establishment of data analysis model about e-commerces behavior based on hadoop platform,” in *2018 International Conference on Intelligent Transportation, Big Data Smart City (ICITBS)*, Jan 2018, pp. 436–439.

[3] P. Merla and Y. Liang, “Data analysis using hadoop mapreduce environment,” in *2017 IEEE International Conference on Big Data (Big Data)*, Dec 2017, pp. 4783–4785.

[4] H. Wu, “Decision analysis of the weather log by hadoop,” in *International Conference on Communication and Electronic Information Engineering (CEIE 2016)*. Atlantis Press, 2016/10. [Online]. Available: https://doi.org/10.2991/ceie-16.2017.3

[5] A. Scoop, “Apache sqoop,” 2019. [Online]. Available: https://sqoop.apache.org/

[6] J. Dean and S. Ghemawat, “Mapreduce: a flexible data processing tool,” *Communications of the ACM*, vol. 53, no. 1, pp. 72–77, 2010.

[7] A. Pig, “Welcome to apache pig!” [Online]. Available: https://pig.apache.org/

[8] L. George, *HBase: The Definitive Guide: Random Access to Your Planet-Size Data*. O’Reilly, Aug 2011. [Online]. Available: https://www.dawsonera.com:443/abstract/9781449315771