Big Data Project - NYC Parking Tickets Analysis

PES2201800504	Aswin A Nair
PES2201800107	R.Darsini
PES2201800036	Ishaan Samant
PES2201800505	Nikita Ganvkar

ABSTRACT

Context

The NYC Department of Finance collects data on every parking ticket issued in NYC (~10M per year). This data is made publicly available to aid in ticket resolution and to guide policymakers.

Content of Dataset

The file is roughly organized by fiscal year (July 1 - June 30 of 2015-2016) with the exception of the initial dataset. The column attributes include information such as the vehicle ticketed, the ticket issued, location, and time.

AIM

The purpose of this project is to conduct exploratory data analysis and queries that will help us understand the data.

OBJECTIVES

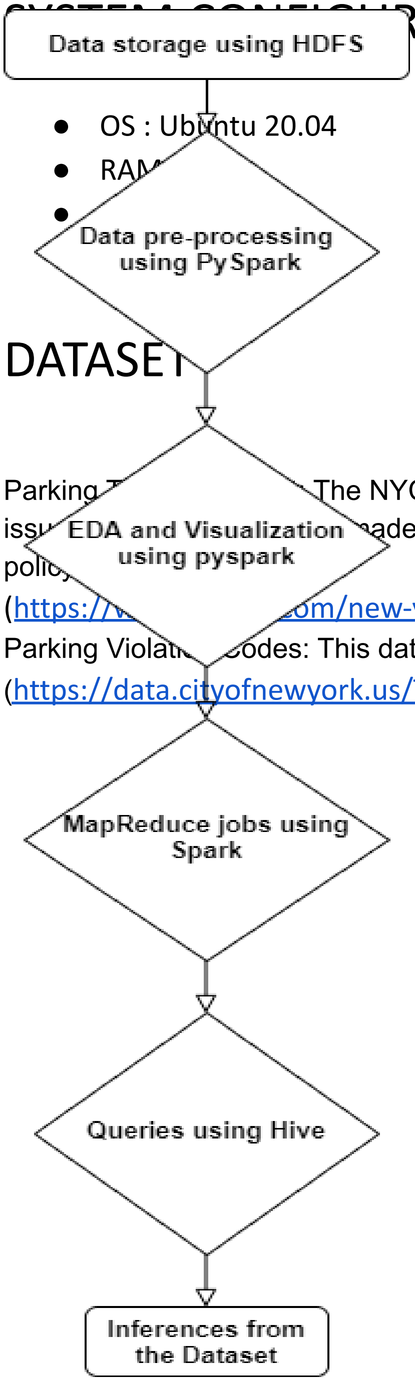
- Using Spark to perform EDA
- Executing queries with Hive and MapReduce
- Get basic inferences about the dataset

SYSTEM CONFIGURATION:

- OS : Ubuntu 20.04
- RAM
-

DATASET

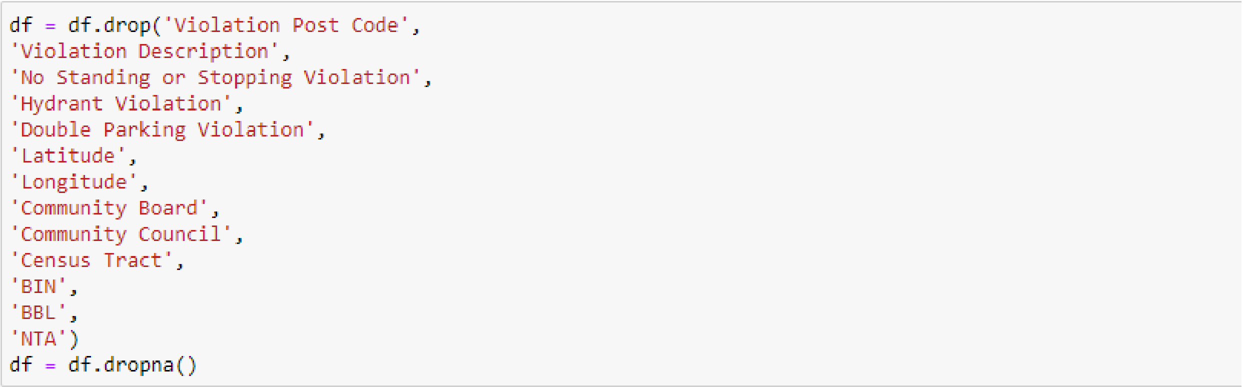
Parking T The NYC Department of Finance collects data on every parking ticket issu ade publicly available to aid in ticket resolution and to guide polic
(https://w om/new-york-city/nyc-parking-tickets/data)
Parking Violati odes: This dataset defines the parking violation codes in NYC and lists the fines.
(https://data.cityofnewyork.us/Transportation/DOF-Parking-Violation-Codes/ncbg-6agr)


Data storage using HDFS
Data pre-processing using PySpark
EDA and Visualization using pyspark
MapReduce jobs using Spark
Queries using Hive
Inferences from the Dataset

PIPELINE

SPARK

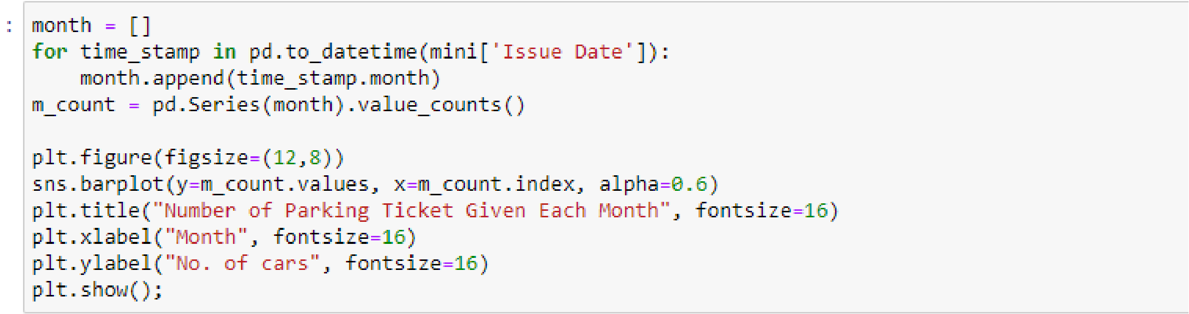
Preprocessing



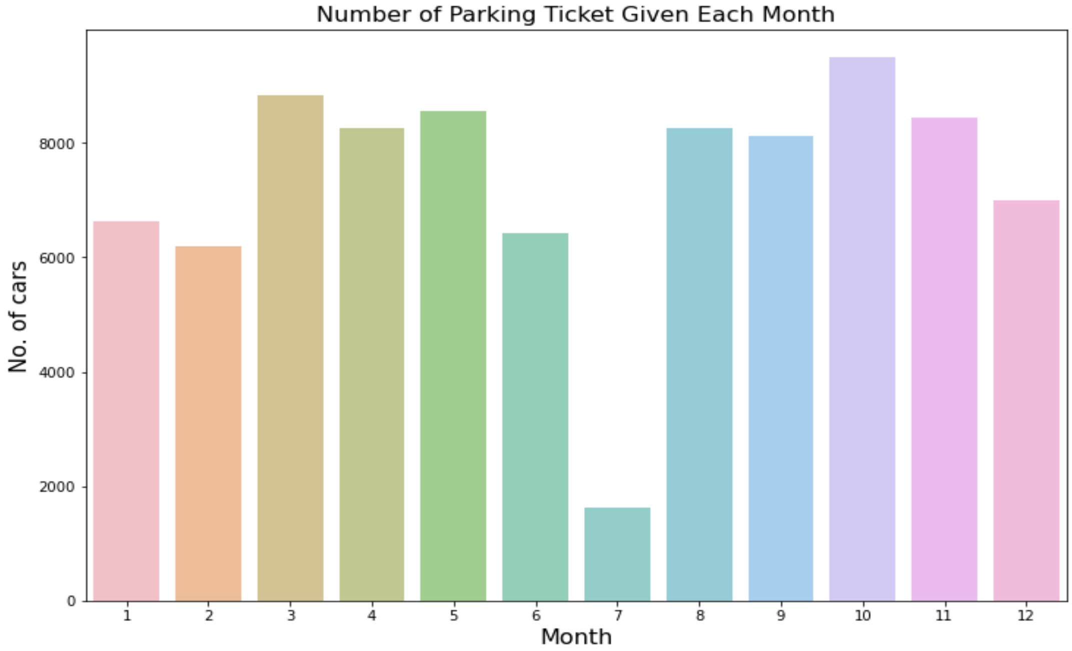
```
df = df.drop('Violation Post Code',
'Violation Description',
'No Standing or Stopping Violation',
'Hydrant Violation',
'Double Parking Violation',
'Latitude',
'Longitude',
'Community Board',
'Community Council',
'Census Tract',
'BIN',
'BBL',
'NTA')
df = df.dropna()
```

EDA - Visualizations

Parking Tickets given each Month

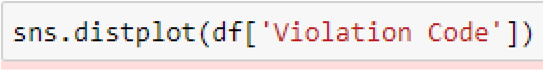


```
: month = []
  for time_stamp in pd.to_datetime(mini['Issue Date']):
      month.append(time_stamp.month)
  m_count = pd.Series(month).value_counts()

  plt.figure(figsize=(12,8))
  sns.barplot(y=m_count.values, x=m_count.index, alpha=0.6)
  plt.title("Number of Parking Ticket Given Each Month", fontsize=16)
  plt.xlabel("Month", fontsize=16)
  plt.ylabel("No. of cars", fontsize=16)
  plt.show();
```

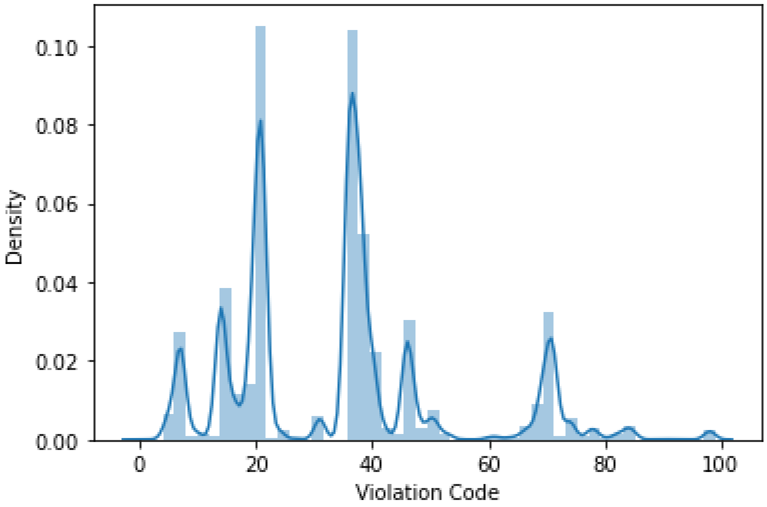


Number of Parking Ticket Given Each Month

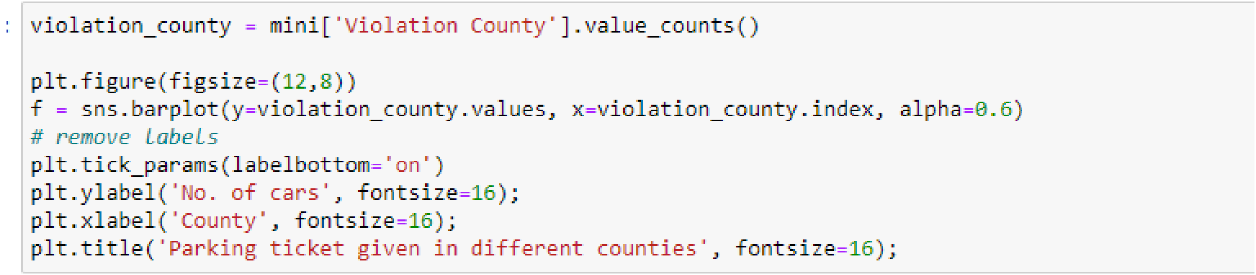
Violation Code Distribution



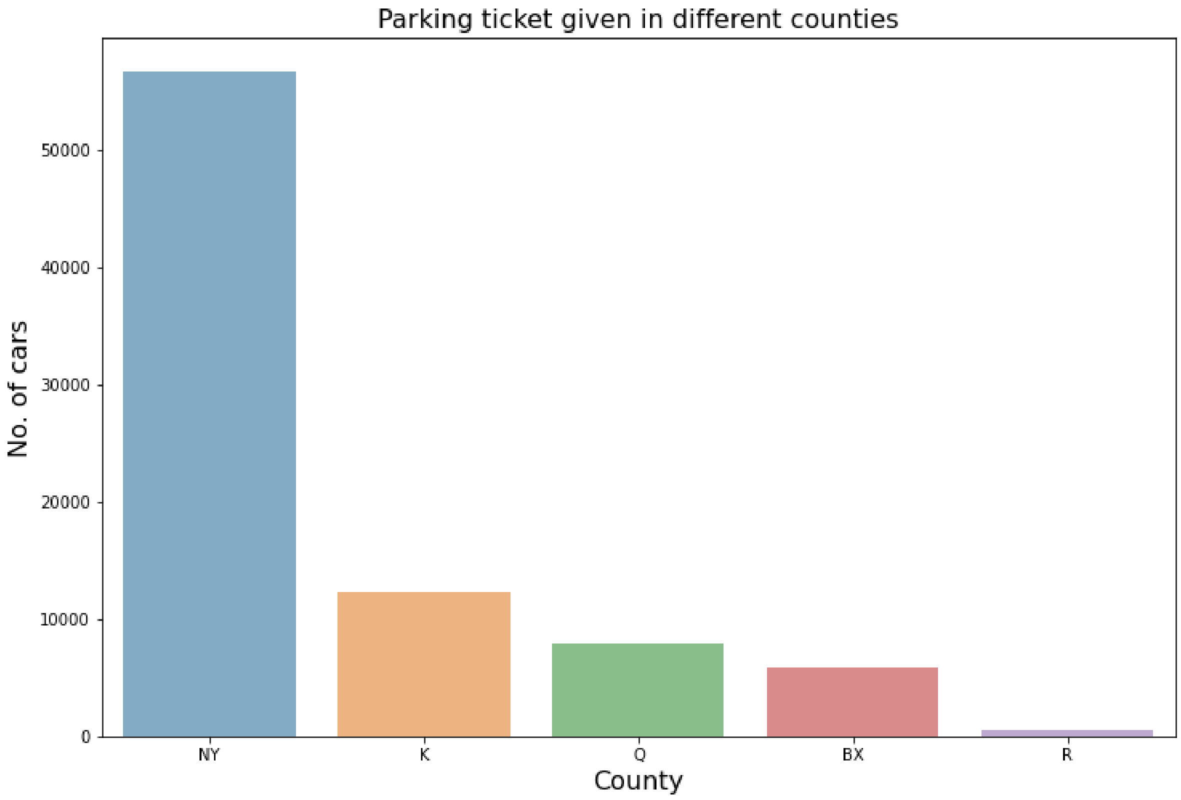
```
sns.distplot(df['Violation Code'])
```



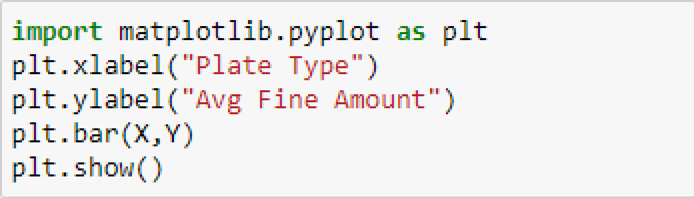
Parking Ticket per County



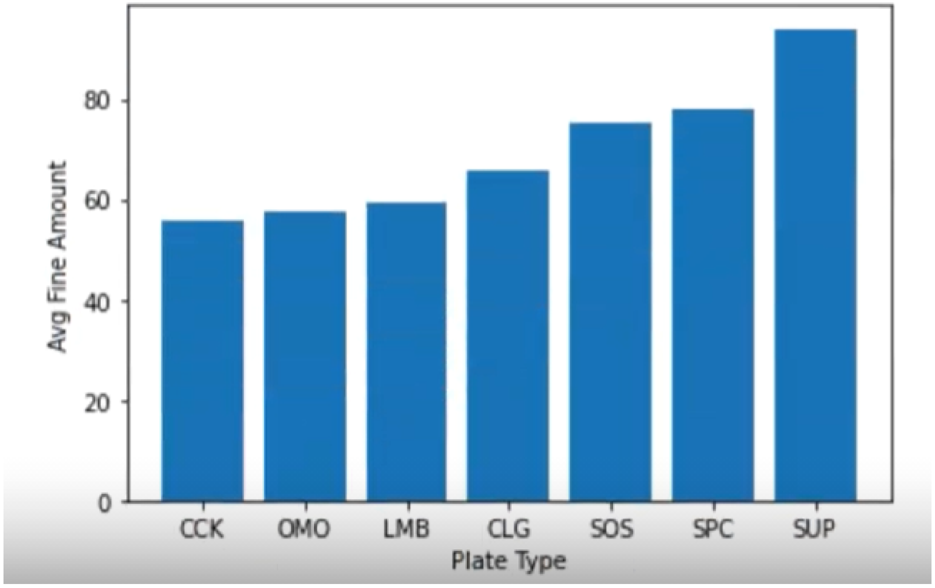
```
violation_county = mini['Violation County'].value_counts()

plt.figure(figsize=(12,8))
f = sns.barplot(y=violation_county.values, x=violation_county.index, alpha=0.6)
# remove labels
plt.tick_params(labelbottom='on')
plt.ylabel('No. of cars', fontsize=16);
plt.xlabel('County', fontsize=16);
plt.title('Parking ticket given in different counties', fontsize=16);
```



Average Amount of Fine for the top Plate Types

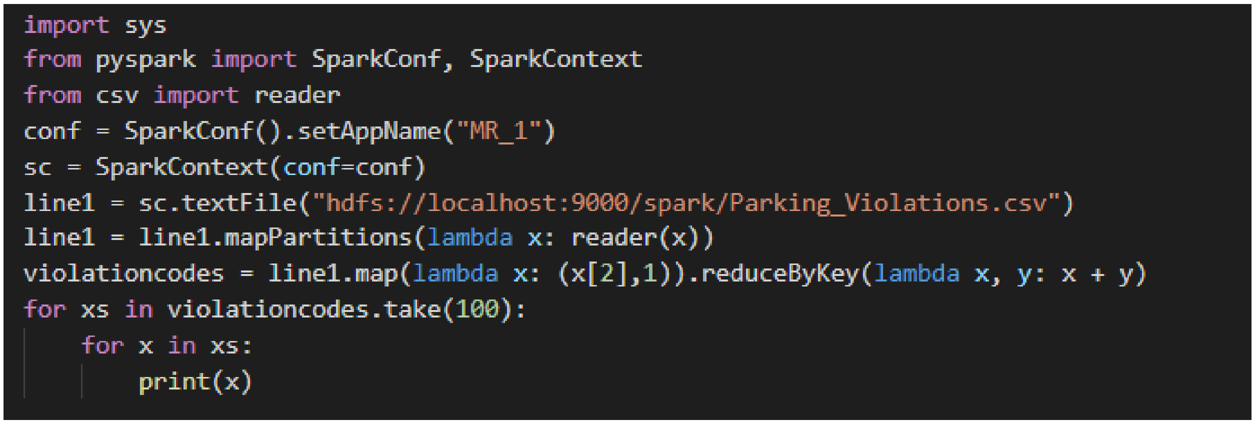


```
import matplotlib.pyplot as plt
plt.xlabel("Plate Type")
plt.ylabel("Avg Fine Amount")
plt.bar(X,Y)
plt.show()
```



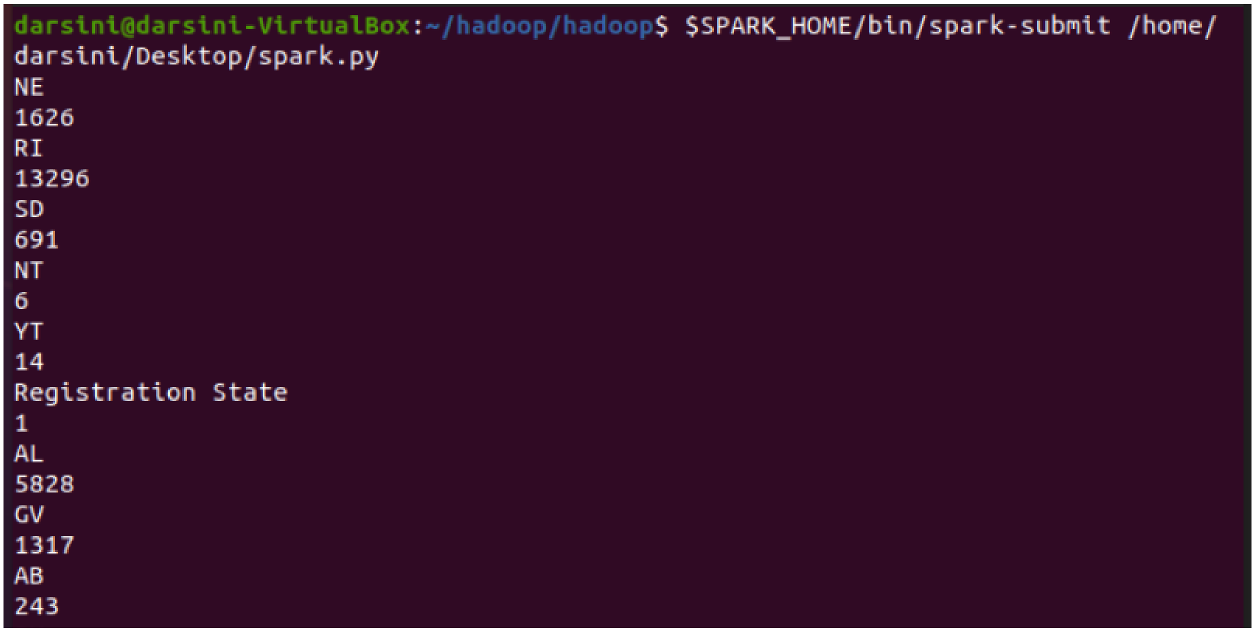
MAP REDUCE

Query: Number of Violations per Registration State



```
import sys
from pyspark import SparkConf, SparkContext
from csv import reader
conf = SparkConf().setAppName("MR_1")
sc = SparkContext(conf=conf)
line1 = sc.textFile("hdfs://localhost:9000/spark/Parking_Violations.csv")
line1 = line1.mapPartitions(lambda x: reader(x))
violationcodes = line1.map(lambda x: (x[2],1)).reduceByKey(lambda x, y: x + y)
for xs in violationcodes.take(100):
    for x in xs:
        print(x)
```

Output

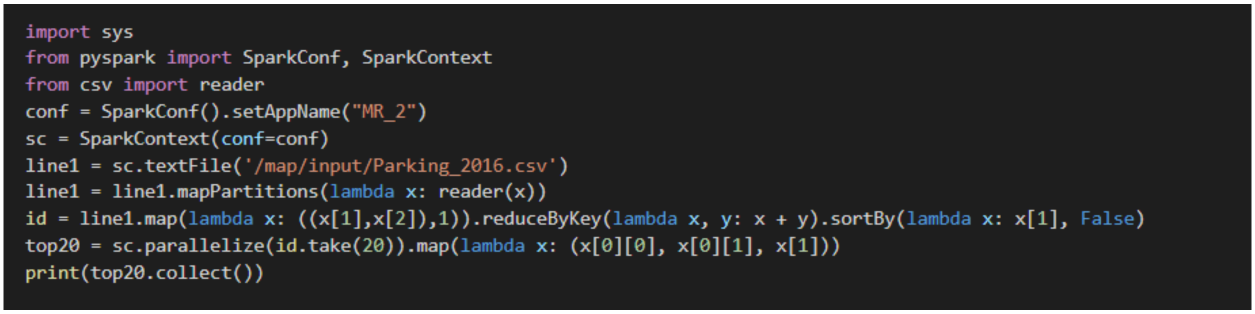


```
darsini@darsini-VirtualBox:~/hadoop/hadoop$ $SPARK_HOME/bin/spark-submit /home/
darsini/Desktop/spark.py
NE
1626
RI
13296
SD
691
NT
6
YT
14
Registration State
1
AL
5828
GV
1317
AB
243
```

This shows that the state RI (Rhode Island) has the maximum number of violation with about 13,296 violations

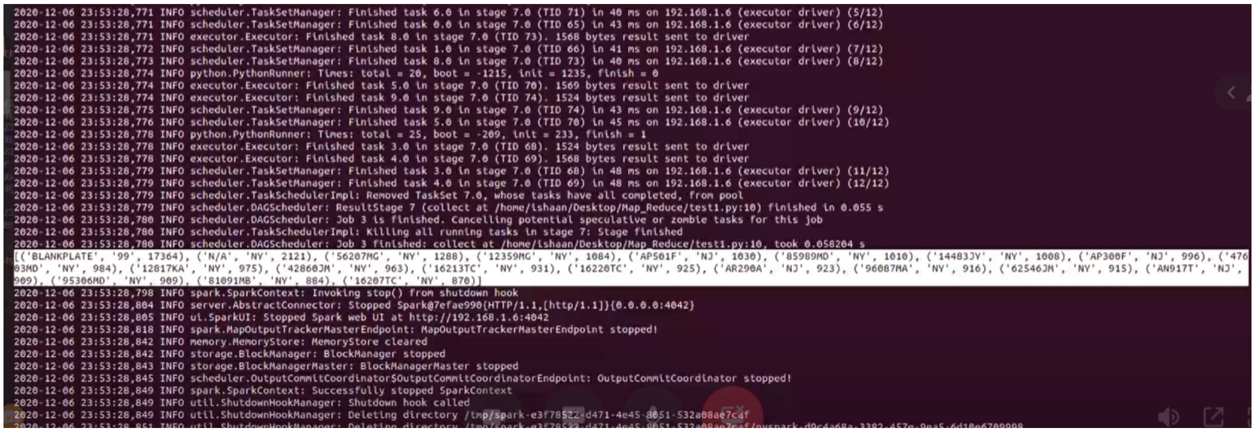
Query: Top 20 vehicles in terms of total violations

MapReduce



```
import sys
from pyspark import SparkConf, SparkContext
from csv import reader
conf = SparkConf().setAppName("MR_2")
sc = SparkContext(conf=conf)
line1 = sc.textFile('/map/input/Parking_2016.csv')
line1 = line1.mapPartitions(lambda x: reader(x))
id = line1.map(lambda x: ((x[1],x[2]),1)).reduceByKey(lambda x, y: x + y).sortBy(lambda x: x[1], False)
top20 = sc.parallelize(id.take(20)).map(lambda x: (x[0][0], x[0][1], x[1]))
print(top20.collect())
```

Output

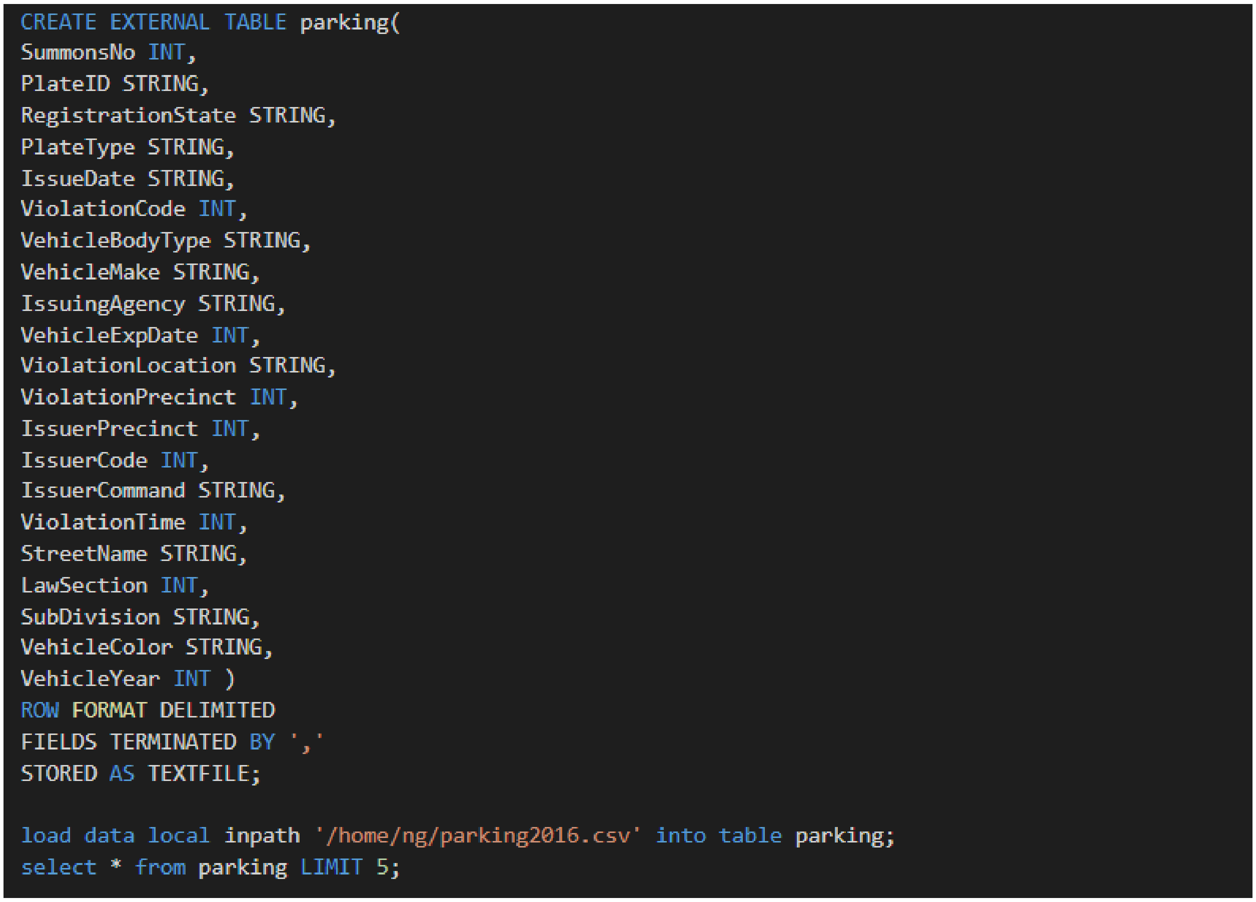


```
2020-12-06 23:53:28,771 INFO scheduler.TaskSetManager: Finished task 6.0 in stage 7.0 (TID 71) in 40 ms on 192.168.1.6 (executor driver) (5/12)
2020-12-06 23:53:28,771 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 7.0 (TID 65) in 43 ms on 192.168.1.6 (executor driver) (6/12)
2020-12-06 23:53:28,771 INFO executor.Executor: Finished task 8.0 in stage 7.0 (TID 73). 1568 bytes result sent to driver
2020-12-06 23:53:28,772 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 7.0 (TID 66) in 41 ms on 192.168.1.6 (executor driver) (7/12)
2020-12-06 23:53:28,773 INFO scheduler.TaskSetManager: Finished task 8.0 in stage 7.0 (TID 73) in 40 ms on 192.168.1.6 (executor driver) (8/12)
2020-12-06 23:53:28,774 INFO python.PythonRunner: Times: total = 20, boot = -1215, init = 1235, finish = 0
2020-12-06 23:53:28,774 INFO executor.Executor: Finished task 5.0 in stage 7.0 (TID 70). 1569 bytes result sent to driver
2020-12-06 23:53:28,774 INFO executor.Executor: Finished task 9.0 in stage 7.0 (TID 74). 1524 bytes result sent to driver
2020-12-06 23:53:28,775 INFO scheduler.TaskSetManager: Finished task 9.0 in stage 7.0 (TID 74) in 43 ms on 192.168.1.6 (executor driver) (9/12)
2020-12-06 23:53:28,776 INFO scheduler.TaskSetManager: Finished task 5.0 in stage 7.0 (TID 70) in 45 ms on 192.168.1.6 (executor driver) (10/12)
2020-12-06 23:53:28,778 INFO python.PythonRunner: Times: total = 25, boot = -209, init = 233, finish = 1
2020-12-06 23:53:28,778 INFO executor.Executor: Finished task 3.0 in stage 7.0 (TID 68). 1524 bytes result sent to driver
2020-12-06 23:53:28,778 INFO executor.Executor: Finished task 4.0 in stage 7.0 (TID 69). 1568 bytes result sent to driver
2020-12-06 23:53:28,779 INFO scheduler.TaskSetManager: Finished task 3.0 in stage 7.0 (TID 68) in 48 ms on 192.168.1.6 (executor driver) (11/12)
2020-12-06 23:53:28,779 INFO scheduler.TaskSetManager: Finished task 4.0 in stage 7.0 (TID 69) in 48 ms on 192.168.1.6 (executor driver) (12/12)
2020-12-06 23:53:28,779 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 7.0, whose tasks have all completed, from pool
2020-12-06 23:53:28,779 INFO scheduler.DAGScheduler: ResultStage 7 (collect at /home/ishaan/Desktop/Map_Reduce/test1.py:10) finished in 0.055 s
2020-12-06 23:53:28,780 INFO scheduler.DAGScheduler: Job 3 is finished. Cancelling potential speculative or zombie tasks for this job
2020-12-06 23:53:28,780 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in stage 7: Stage finished
2020-12-06 23:53:28,780 INFO scheduler.DAGScheduler: Job 3 finished: collect at /home/ishaan/Desktop/Map_Reduce/test1.py:10, took 0.058204 s
[('BLANKPLATE', '99', 17364), ('N/A', 'NY', 2121), ('56207MG', 'NY', 1288), ('12359MG', 'NY', 1084), ('AP501F', 'NJ', 1030), ('85989MD', 'NY', 1010), ('14483JY', 'NY', 1008), ('AP300F', 'NJ', 996), ('476
03MD', 'NY', 984), ('12817KA', 'NY', 975), ('42860JM', 'NY', 963), ('16213TC', 'NY', 931), ('16220TC', 'NY', 925), ('AR290A', 'NJ', 923), ('96087MA', 'NY', 916), ('62546JM', 'NY', 915), ('AN917T', 'NJ',
909), ('95306MD', 'NY', 909), ('81091MB', 'NY', 884), ('16207TC', 'NY', 870)]
2020-12-06 23:53:28,798 INFO spark.SparkContext: Invoking stop() from shutdown hook
2020-12-06 23:53:28,804 INFO server.AbstractConnector: Stopped Spark@7efae990{HTTP/1.1,[http/1.1]}{0.0.0.0:4042}
2020-12-06 23:53:28,805 INFO ui.SparkUI: Stopped Spark web UI at http://192.168.1.6:4042
2020-12-06 23:53:28,818 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
2020-12-06 23:53:28,842 INFO memory.MemoryStore: MemoryStore cleared
2020-12-06 23:53:28,842 INFO storage.BlockManager: BlockManager stopped
2020-12-06 23:53:28,843 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
2020-12-06 23:53:28,845 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
2020-12-06 23:53:28,849 INFO spark.SparkContext: Successfully stopped SparkContext
2020-12-06 23:53:28,849 INFO util.ShutdownHookManager: Shutdown hook called
2020-12-06 23:53:28,849 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-e3f78522-d471-4e45-8051-532a08ae7caf
2020-12-06 23:53:28,851 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-e3f78522-d471-4e45-8051-532a08ae7caf/pyspark-d9c4a68a-3382-457e-9aa5-6d10e6709998
```

HIVE

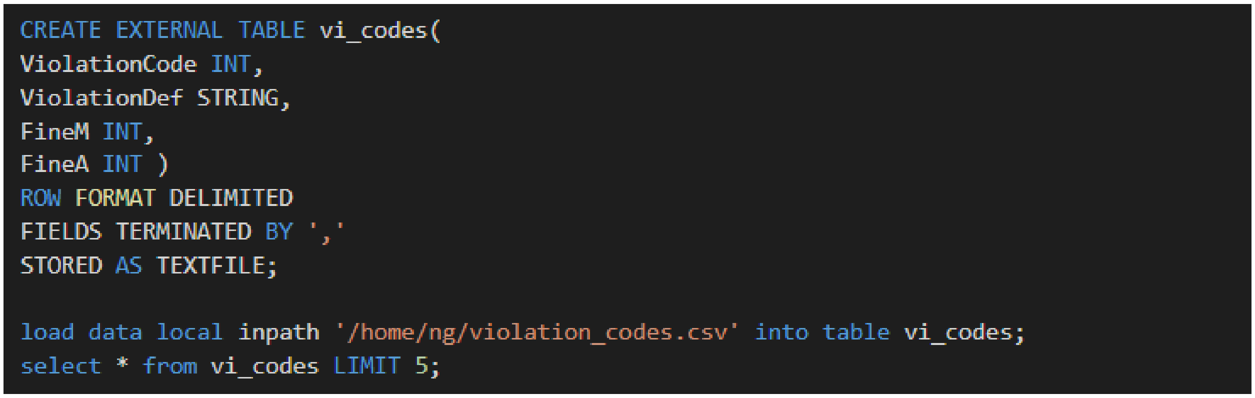
Loading two tables:

- Parking Ticket Details (from Kaggle)



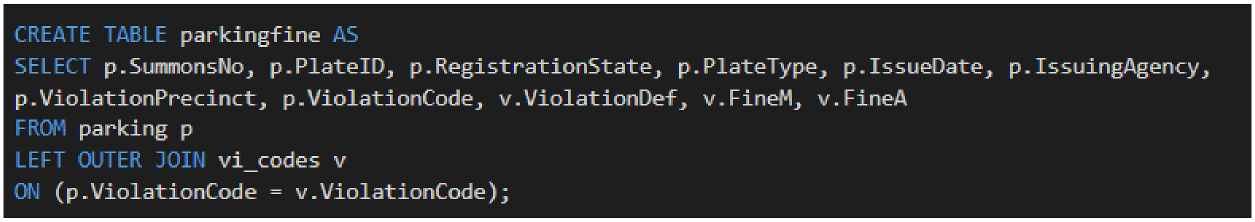
```
CREATE EXTERNAL TABLE parking(
SummonsNo INT,
PlateID STRING,
RegistrationState STRING,
PlateType STRING,
IssueDate STRING,
ViolationCode INT,
VehicleBodyType STRING,
VehicleMake STRING,
IssuingAgency STRING,
VehicleExpDate INT,
ViolationLocation STRING,
ViolationPrecinct INT,
IssuerPrecinct INT,
IssuerCode INT,
IssuerCommand STRING,
ViolationTime INT,
StreetName STRING,
LawSection INT,
SubDivision STRING,
VehicleColor STRING,
VehicleYear INT )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

load data local inpath '/home/ng/parking2016.csv' into table parking;
select * from parking LIMIT 5;
```

- Details of Violation Codes



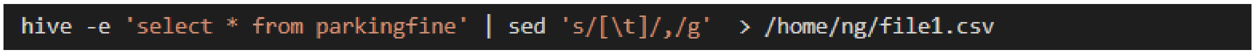
```
CREATE EXTERNAL TABLE vi_codes(
ViolationCode INT,
ViolationDef STRING,
FineM INT,
FineA INT )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

load data local inpath '/home/ng/violation_codes.csv' into table vi_codes;
select * from vi_codes LIMIT 5;
```

- Combining two datasets on violation code



```
CREATE TABLE parkingfine AS
SELECT p.SummonsNo, p.PlateID, p.RegistrationState, p.PlateType, p.IssueDate, p.IssuingAgency,
p.ViolationPrecinct, p.ViolationCode, v.ViolationDef, v.FineM, v.FineA
FROM parking p
LEFT OUTER JOIN vi_codes v
ON (p.ViolationCode = v.ViolationCode);
```

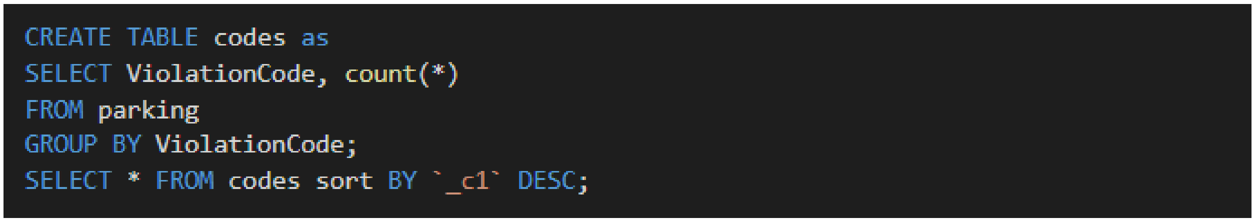
- Exporting dataset to be used



```
hive -e 'select * from parkingfine' | sed 's/[\t]/,/g'  > /home/ng/file1.csv
```

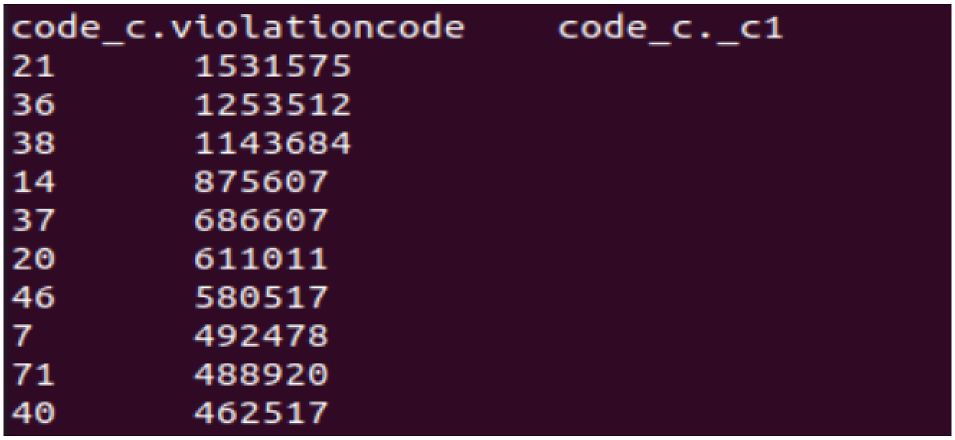
Queries:

Violation code, the number of violations that have this code



```
CREATE TABLE codes as
SELECT ViolationCode, count(*)
FROM parking
GROUP BY ViolationCode;
SELECT * FROM codes sort BY `_c1` DESC;
```

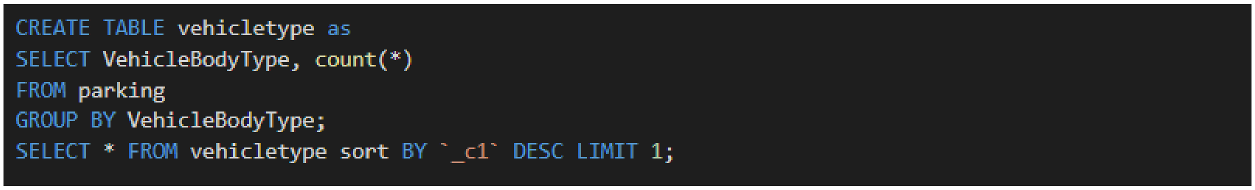
Output



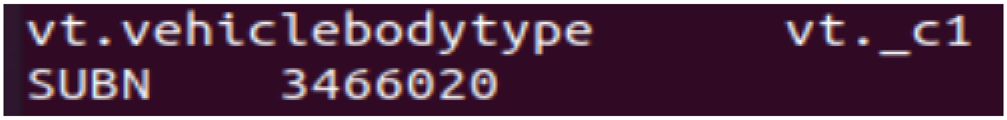
```
code_c.violationcode    code_c._c1
21        1531575
36        1253512
38        1143684
14        875607
37        686607
20        611011
46        580517
7         492478
71        488920
40        462517
```

Violation Code 21 which states that *Street Cleaning: No **parking** where **parking** is not allowed by sign, street marking or traffic control device* (description according to the nyc government site) is the most violated code of all the rules.

Which vehicle body type is most likely to get a parking ticket



```
CREATE TABLE vehicletype as
SELECT VehicleBodyType, count(*)
FROM parking
GROUP BY VehicleBodyType;
SELECT * FROM vehicletype sort BY `_c1` DESC LIMIT 1;
```



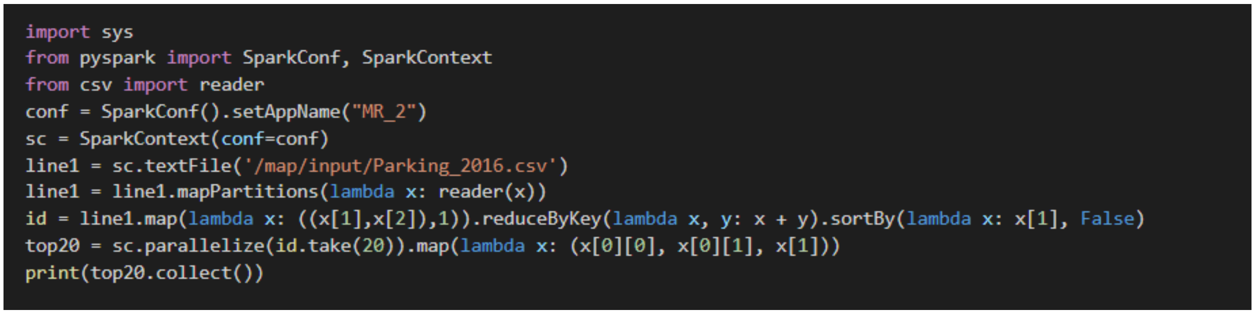
```
vt.vehiclebodytype      vt._c1
SUBN    3466020
```

SUBN - Suburban Vehicles have the highest count of parking tickets

HIVE vs MapReduce

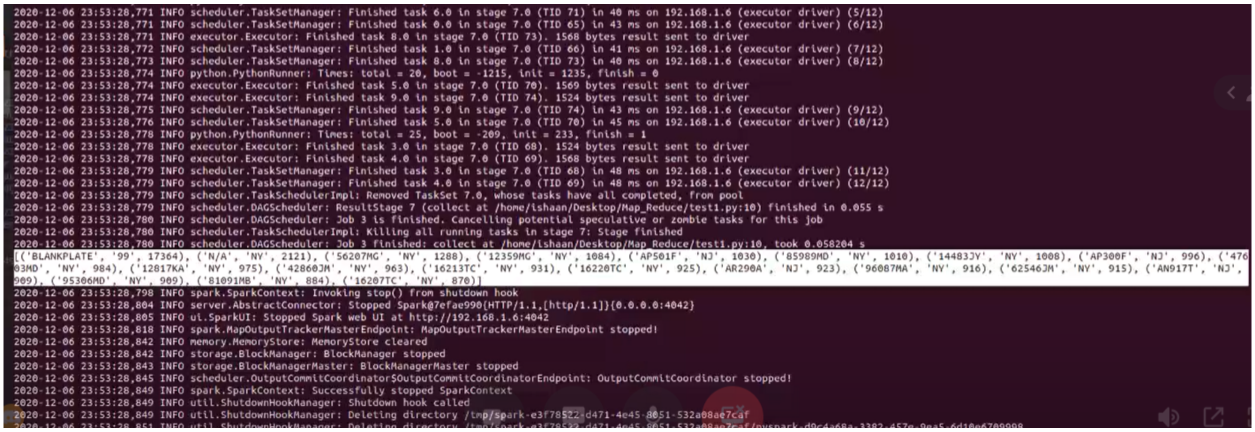
Chosen Query: Top 20 vehicles in terms of total violations

MapReduce



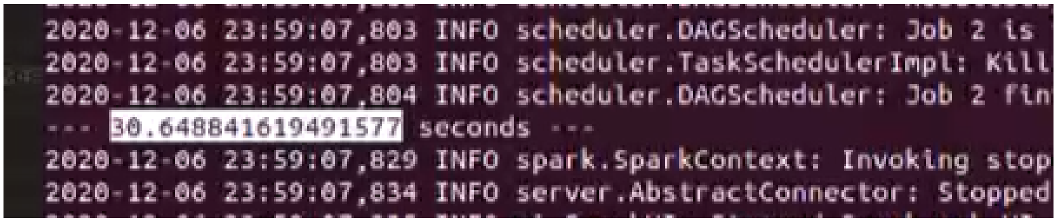
```
import sys
from pyspark import SparkConf, SparkContext
from csv import reader
conf = SparkConf().setAppName("MR_2")
sc = SparkContext(conf=conf)
line1 = sc.textFile('/map/input/Parking_2016.csv')
line1 = line1.mapPartitions(lambda x: reader(x))
id = line1.map(lambda x: ((x[1],x[2]),1)).reduceByKey(lambda x, y: x + y).sortBy(lambda x: x[1], False)
top20 = sc.parallelize(id.take(20)).map(lambda x: (x[0][0], x[0][1], x[1]))
print(top20.collect())
```

Output



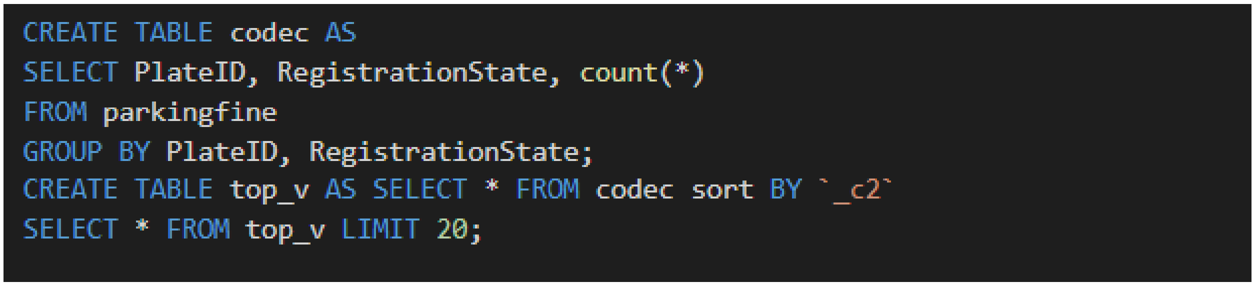
```
2020-12-06 23:53:28,771 INFO scheduler.TaskSetManager: Finished task 6.0 in stage 7.0 (TID 71) in 40 ms on 192.168.1.6 (executor driver) (5/12)
2020-12-06 23:53:28,771 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 7.0 (TID 65) in 43 ms on 192.168.1.6 (executor driver) (6/12)
2020-12-06 23:53:28,771 INFO executor.Executor: Finished task 8.0 in stage 7.0 (TID 73). 1568 bytes result sent to driver
2020-12-06 23:53:28,772 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 7.0 (TID 66) in 41 ms on 192.168.1.6 (executor driver) (7/12)
2020-12-06 23:53:28,773 INFO scheduler.TaskSetManager: Finished task 8.0 in stage 7.0 (TID 73) in 40 ms on 192.168.1.6 (executor driver) (8/12)
2020-12-06 23:53:28,774 INFO python.PythonRunner: Times: total = 20, boot = -1215, init = 1235, finish = 0
2020-12-06 23:53:28,774 INFO executor.Executor: Finished task 5.0 in stage 7.0 (TID 70). 1569 bytes result sent to driver
2020-12-06 23:53:28,774 INFO executor.Executor: Finished task 9.0 in stage 7.0 (TID 74). 1524 bytes result sent to driver
2020-12-06 23:53:28,775 INFO scheduler.TaskSetManager: Finished task 9.0 in stage 7.0 (TID 74) in 43 ms on 192.168.1.6 (executor driver) (9/12)
2020-12-06 23:53:28,776 INFO scheduler.TaskSetManager: Finished task 5.0 in stage 7.0 (TID 70) in 45 ms on 192.168.1.6 (executor driver) (10/12)
2020-12-06 23:53:28,778 INFO python.PythonRunner: Times: total = 25, boot = -209, init = 233, finish = 1
2020-12-06 23:53:28,778 INFO executor.Executor: Finished task 3.0 in stage 7.0 (TID 68). 1524 bytes result sent to driver
2020-12-06 23:53:28,778 INFO executor.Executor: Finished task 4.0 in stage 7.0 (TID 69). 1568 bytes result sent to driver
2020-12-06 23:53:28,779 INFO scheduler.TaskSetManager: Finished task 3.0 in stage 7.0 (TID 68) in 48 ms on 192.168.1.6 (executor driver) (11/12)
2020-12-06 23:53:28,779 INFO scheduler.TaskSetManager: Finished task 4.0 in stage 7.0 (TID 69) in 48 ms on 192.168.1.6 (executor driver) (12/12)
2020-12-06 23:53:28,779 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 7.0, whose tasks have all completed, from pool
2020-12-06 23:53:28,779 INFO scheduler.DAGScheduler: ResultStage 7 (collect at /home/ishaan/Desktop/Map_Reduce/test1.py:10) finished in 0.055 s
2020-12-06 23:53:28,780 INFO scheduler.DAGScheduler: Job 3 is finished. Cancelling potential speculative or zombie tasks for this job
2020-12-06 23:53:28,780 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in stage 7: Stage finished
2020-12-06 23:53:28,780 INFO scheduler.DAGScheduler: Job 3 finished: collect at /home/ishaan/Desktop/Map_Reduce/test1.py:10, took 0.058204 s
[('BLANKPLATE', '99', 17364), ('N/A', 'NY', 2121), ('56207MG', 'NY', 1288), ('12359MG', 'NY', 1084), ('AP501F', 'NJ', 1030), ('85989MD', 'NY', 1010), ('14483JY', 'NY', 1008), ('AP300F', 'NJ', 996), ('476
03MD', 'NY', 984), ('12817KA', 'NY', 975), ('42860JM', 'NY', 963), ('16213TC', 'NY', 931), ('16220TC', 'NY', 925), ('AR290A', 'NJ', 923), ('96087MA', 'NY', 916), ('62546JM', 'NY', 915), ('AN917T', 'NJ',
909), ('95306MD', 'NY', 909), ('81091MB', 'NY', 884), ('16207TC', 'NY', 870)]
2020-12-06 23:53:28,798 INFO spark.SparkContext: Invoking stop() from shutdown hook
2020-12-06 23:53:28,804 INFO server.AbstractConnector: Stopped Spark@7efae990{HTTP/1.1,[http/1.1]}{0.0.0.0:4042}
2020-12-06 23:53:28,805 INFO ui.SparkUI: Stopped Spark web UI at http://192.168.1.6:4042
2020-12-06 23:53:28,818 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
2020-12-06 23:53:28,842 INFO memory.MemoryStore: MemoryStore cleared
2020-12-06 23:53:28,842 INFO storage.BlockManager: BlockManager stopped
2020-12-06 23:53:28,843 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
2020-12-06 23:53:28,845 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
2020-12-06 23:53:28,849 INFO spark.SparkContext: Successfully stopped SparkContext
2020-12-06 23:53:28,849 INFO util.ShutdownHookManager: Shutdown hook called
2020-12-06 23:53:28,849 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-e3f78522-d471-4e45-8051-532a08ae7caf
2020-12-06 23:53:28,851 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-e3f78522-d471-4e45-8051-532a08ae7caf/pyspark-d9c4a68a-3382-457e-9ea5-6d10e6709998
```

Time Taken:



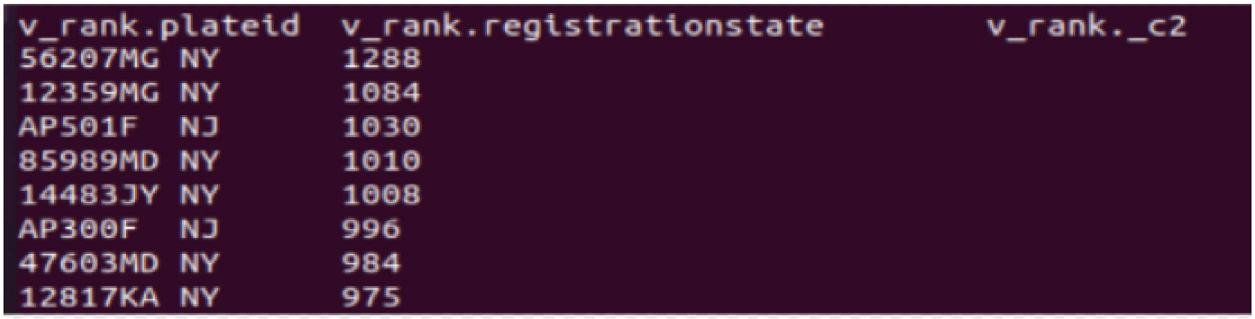
```
2020-12-06 23:59:07,803 INFO scheduler.DAGScheduler: Job 2 is
2020-12-06 23:59:07,803 INFO scheduler.TaskSchedulerImpl: Kill
2020-12-06 23:59:07,804 INFO scheduler.DAGScheduler: Job 2 fin
--- 30.648841619491577 seconds ---
2020-12-06 23:59:07,829 INFO spark.SparkContext: Invoking stop
2020-12-06 23:59:07,834 INFO server.AbstractConnector: Stopped
```

HIVE



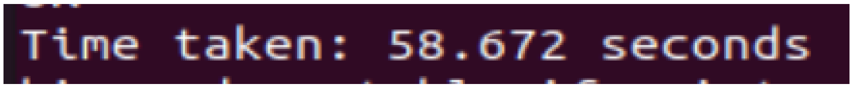
```
CREATE TABLE codec AS
SELECT PlateID, RegistrationState, count(*)
FROM parkingfine
GROUP BY PlateID, RegistrationState;
CREATE TABLE top_v AS SELECT * FROM codec sort BY `_c2`
SELECT * FROM top_v LIMIT 20;
```

Output



```
v_rank.plateid  v_rank.registrationstate        v_rank._c2
56207MG  NY      1288
12359MG  NY      1084
AP501F   NJ      1030
85989MD  NY      1010
14483JY  NY      1008
AP300F   NJ      996
47603MD  NY      984
12817KA  NY      975
```

Time Taken



```
Time taken: 58.672 seconds
```