Table of Contents

Objective - The assignment is meant for you to apply learnings of the module on Hive on a real-life dataset. One of the major objectives of this assignment is gaining familiarity with how an analysis works in Hive and how you can gain insights from large datasets.

Problem Statement - New York City is a thriving metropolis and just like most other cities of similar size, one of the biggest problems its residents face is parking. The classic combination of a huge number of cars and a cramped geography is the exact recipe that leads to a large number of parking tickets.

In an attempt to scientifically analyse this phenomenon, the NYC Police Department regularly collects data related to parking tickets. This data is made available by NYC Open Data portal. We will try and perform some analysis on this data.

Download Dataset - https://data.cityofnewyork.us/browse?q=parking+tickets

Note: Consider only the year 2017 for analysis and not the Fiscal year.

Create database

```
CREATE database mini_project_2;
```

Create table schema

```
CREATE table ny_parking_violation_data
(
Summons_Number int,
Plate_ID string,
Registration_State string,
Plate_Type string,
Issue_Date date,
Violation_Code int,
Vehicle_Body_Type string,
Vehicle_Make string,
Issuing_Agency string,
Street_Code1 int,
Street_Code2 int,
Street_Code3 int,
Vehicle_Expiration_Date date,
Violation_Location string,
Violation_Precinct int,
Issuer_Precinct int,
Issuer_Code string,
Issuer_Command string,
Issuer_Squad string,
Violation_Time string,
Time_First_Observed string,
Violation_County string,
Violation_In_Front_Of_Or_Opposite string,
House_Number string,
Street_Name string,
Intersecting_Street string,
Date_First_Observed int,
Law_Section int,
Sub_Division string,
Violation_Legal_Code string,
Days_Parking_In_Effect string,
From_Hours_In_Effect string,
To_Hours_In_Effect string,
Vehicle_Color string,
Unregistered_Vehicle string,
Vehicle_Year string,
Meter_Number string,
Feet_From_Curb int,
Violation_Post_Code string,
Violation_Description string,
No_Standing_or_Stopping_Violation string,
Hydrant_Violation string,
Double_Parking_Violation string
)
row format delimited
fields terminated by ","
tblproperties ("skip.header.line.count=1");
```

Load data in the table

```
load data local inpath
'file:///home/cloudera/mini_project_2/Parking_Violations_Issued_Fiscal_Year_2017.csv' into
table ny_parking_violation_data;
```

Create partitioned table for ny_parking_violation_data

```
CREATE table ny_parking_violation_data_PARTITIONED
(
Summons_Number int,
Plate_ID string,
Registration_State string,
Plate_Type string,
Issue_Date date,
Violation_Code int,
Vehicle_Body_Type string,
Vehicle_Make string,
Issuing_Agency string,
Street_Code1 int,
Street_Code2 int,
Street_Code3 int,
Vehicle_Expiration_Date date,
Violation_Location string,

Violation_Precinct int,

Issuer_Precinct int,

Issuer_Code string,

Issuer_Command string,

Issuer_Squad string,

Violation_Time string,

Time_First_Observed string,

Violation_In_Front_Of_Or_Opposite string,

House_Number string,

Street_Name string,

Intersecting_Street string,

Date_First_Observed int,

Law_Section int,

Sub_Division string,

Violation_Legal_Code string,

Days_Parking_In_Effect string,

From_Hours_In_Effect string,

To_Hours_In_Effect string,

Vehicle_Color string,

Unregistered_Vehicle string,

Vehicle_Year string,

Meter_Number string,

Feet_From_Curb int,

Violation_Post_Code string,

Violation_Description string,

No_Standing_or_Stopping_Violation string,

Hydrant_Violation string,

Double_Parking_Violation string

)
```

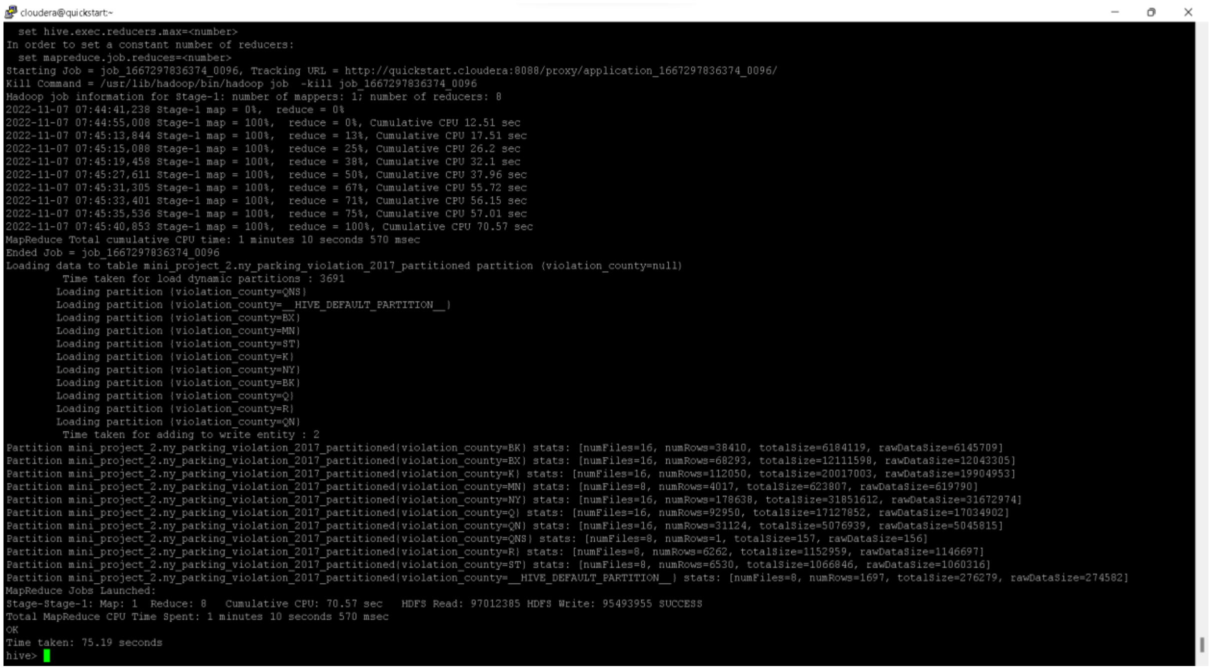
```
PARTITIONED by (violation_county string)
clustered by (violation_code) sorted by (violation_code) into 8 buckets
tblproperties ("skip.header.line.count=1");
```

Set properties for dynamic partitioning

```
Set hive.exec.dynamic.partition=true;
Set hive.exec.dynamic.partition.mode=nonstrict;
Set hive.enforce.bucketing=true;
Set hive.vectorization.enable=true;
```

Load the data in the partitioned table FROM ny_parking_violation_data_partitioned

```
Insert into ny_parking_violation_data_PARTITIONED partition(violation_county)
SELECT Summons_Number, Plate_ID, Registration_State, Plate_Type, Issue_Date,
Violation_Code, Vehicle_Body_Type, Vehicle_Make, Issuing_Agency, Street_Code1,
Street_Code2, Street_Code3, Vehicle_Expiration_Date, Violation_Location, Violation_Precinct,
Issuer_Precinct, Issuer_Code, Issuer_Command, Issuer_Squad, Violation_Time,
Time_First_Observed, Violation_In_Front_Of_Or_Opposite,House_Number, Street_Name,
Intersecting_Street, Date_First_Observed, Law_Section, Sub_Division, Violation_Legal_Code,
Days_Parking_In_Effect, FROM_Hours_In_Effect, To_Hours_In_Effect, Vehicle_Color,
Unregistered_Vehicle, Vehicle_Year, Meter_Number, Feet_From_Curb, Violation_Post_Code,
Violation_Description,  No_Standing_or_Stopping_Violation, Hydrant_Violation,
Double_Parking_Violation , Violation_County
FROM ny_parking_violation_data WHERE year(issue_date) = ‘2017’
```



```
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1667297836374_0096, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1667297836374_0096/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1667297836374_0096
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 8
2022-11-07 07:44:41,238 Stage-1 map = 0%,  reduce = 0%
2022-11-07 07:44:55,008 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 12.51 sec
2022-11-07 07:45:13,844 Stage-1 map = 100%,  reduce = 13%, Cumulative CPU 17.51 sec
2022-11-07 07:45:15,088 Stage-1 map = 100%,  reduce = 25%, Cumulative CPU 26.2 sec
2022-11-07 07:45:19,458 Stage-1 map = 100%,  reduce = 38%, Cumulative CPU 32.1 sec
2022-11-07 07:45:27,611 Stage-1 map = 100%,  reduce = 50%, Cumulative CPU 37.96 sec
2022-11-07 07:45:31,305 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 55.72 sec
2022-11-07 07:45:33,401 Stage-1 map = 100%,  reduce = 71%, Cumulative CPU 56.15 sec
2022-11-07 07:45:35,536 Stage-1 map = 100%,  reduce = 75%, Cumulative CPU 57.01 sec
2022-11-07 07:45:40,853 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 70.57 sec
MapReduce Total cumulative CPU time: 1 minutes 10 seconds 570 msec
Ended Job = job_1667297836374_0096
Loading data to table mini_project_2.ny_parking_violation_2017_partitioned partition (violation_county=null)
         Time taken for load dynamic partitions : 3691
        Loading partition {violation_county=QNS}
        Loading partition {violation_county=__HIVE_DEFAULT_PARTITION__}
        Loading partition {violation_county=BX}
        Loading partition {violation_county=MN}
        Loading partition {violation_county=ST}
        Loading partition {violation_county=K}
        Loading partition {violation_county=NY}
        Loading partition {violation_county=BK}
        Loading partition {violation_county=Q}
        Loading partition {violation_county=R}
        Loading partition {violation_county=QN}
         Time taken for adding to write entity : 2
Partition mini_project_2.ny_parking_violation_2017_partitioned{violation_county=BK} stats: [numFiles=16, numRows=38410, totalSize=6184119, rawDataSize=6145709]
Partition mini_project_2.ny_parking_violation_2017_partitioned{violation_county=BX} stats: [numFiles=16, numRows=68293, totalSize=12111598, rawDataSize=12043305]
Partition mini_project_2.ny_parking_violation_2017_partitioned{violation_county=K} stats: [numFiles=16, numRows=112050, totalSize=20017003, rawDataSize=19904953]
Partition mini_project_2.ny_parking_violation_2017_partitioned{violation_county=MN} stats: [numFiles=8, numRows=4017, totalSize=623807, rawDataSize=619790]
Partition mini_project_2.ny_parking_violation_2017_partitioned{violation_county=NY} stats: [numFiles=16, numRows=178638, totalSize=31851612, rawDataSize=31672974]
Partition mini_project_2.ny_parking_violation_2017_partitioned{violation_county=Q} stats: [numFiles=16, numRows=92950, totalSize=17127852, rawDataSize=17034902]
Partition mini_project_2.ny_parking_violation_2017_partitioned{violation_county=QN} stats: [numFiles=16, numRows=31124, totalSize=5076939, rawDataSize=5045815]
Partition mini_project_2.ny_parking_violation_2017_partitioned{violation_county=QNS} stats: [numFiles=8, numRows=1, totalSize=157, rawDataSize=156]
Partition mini_project_2.ny_parking_violation_2017_partitioned{violation_county=R} stats: [numFiles=8, numRows=6262, totalSize=1152959, rawDataSize=1146697]
Partition mini_project_2.ny_parking_violation_2017_partitioned{violation_county=ST} stats: [numFiles=8, numRows=6530, totalSize=1066846, rawDataSize=1060316]
Partition mini_project_2.ny_parking_violation_2017_partitioned{violation_county=__HIVE_DEFAULT_PARTITION__} stats: [numFiles=8, numRows=1697, totalSize=276279, rawDataSize=274582]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 8   Cumulative CPU: 70.57 sec   HDFS Read: 97012385 HDFS Write: 95493955 SUCCESS
Total MapReduce CPU Time Spent: 1 minutes 10 seconds 570 msec
OK
Time taken: 75.19 seconds
hive> 
```

The analysis can be divided into two parts:

Part-I: Examine the data

1.) Find the total number of tickets for the year.

```
SELECT count(distinct summons_number) as Total_violations
FROM ny_parking_violation_data_partitioned;
```

2.) Find out how many unique states the cars which got parking tickets came from.

```
SELECT registration_state, count(distinct plate_id) as Total_cars_by_state_violation

FROM ny_parking_violation_data_partitioned

 GROUP BY registration_state order by Total_cars_by_state_violation desc;
```

3.) Some parking tickets don't have addresses on them, which is cause for concern. Find out how many such tickets there are (i.e. tickets WHERE either "Street Code 1" or "Street Code 2" or "Street Code 3" is empty)

```
SELECT registration_state, count(summons_number) as Tickets_w_o_add

FROM ny_parking_violation_data

WHERE street_code1 = 0 or street_code2 = 0

OR street_code3 = 0 GROUP BY registration_state order by Tickets_w_o_add desc;
```

Part-II: Aggregation tasks

1.) How often does each violation code occur? (frequency of violation codes - find the top 5)

```
SELECT violation_code, count(violation_code) as frequency_of_violation_code

FROM ny_parking_violation_data_partitioned

GROUP BY violation_code order_by frequency_of_violation_code desc limit 5;
```

2.) How often does each vehicle body type get a parking ticket? How about the vehicle make? (find the top 5 for both)

```
SELECT vehicle_body_type, count(distinct summons_number) total_tickets

FROM ny_parking_violation_data_partitioned

GROUP BY vehicle_body_type order by total_tickets desc limit 5;
```

```
SELECT vehicle_make, count(distinct summons_number) total_tickets

FROM ny_parking_violation_data_ partitioned

GROUP BY vehicle_make order by total_tickets desc limit 5;
```

4.) A precinct is a police station that has a certain zone of the city under its command. Find the (5 highest) frequencies of:

a.) Violating Precincts (this is the precinct of the zone WHERE the violation occurred)

```
SELECT violation_precinct, count(summons_number) as total_tickets

FROM ny_parking_violation_data_ partitioned

GROUP BY violation_precinct order by total_tickets desc limit 5;
```

b.) Issuer Precincts (this is the precinct that issued the ticket)

```
SELECT issuer_precinct, count(summons_number) as total_tickets

FROM ny_parking_violation_data_ partitioned
```

```
GROUP BY issuer_precinct order by total_tickets desc limit 5;
```

5.) Find the violation code frequency across 3 precincts which have issued the most number of tickets - do these precinct zones have an exceptionally high frequency of certain violation codes?

For top 3 precinct with highest no. of tickets issued –

```
SELECT issuer_precinct, count(violation_code) as total_tickets
FROM ny_parking_violation_data_partitioned
GROUP BY issuer_precinct , order by total_tickets desc limit 7;
```

(**limit by 7 because, to exclude records with missing precinct data**)

- ✓ Precincts with highest tickets are 19, 14 & 18 in respective order

Here is the breakdown of highest numbers of violations by violation_code for each precincts mentioned above.

- Precinct – 19

```
SELECT violation_code, count(violation_code) as total_tickets
FROM ny_parking_violation_data_partitioned
WHERE issuer_precinct = 19
GROUP BY violation_code order by total_tickets desc limit 3;
```

- Precinct – 14

```
SELECT violation_code, count(violation_code) as total_tickets
FROM ny_parking_violation_data_partitioned
WHERE issuer_precinct = 19
GROUP BY violation_code order by total_tickets desc limit 3;
```

- Precinct – 18

```
SELECT violation_code, count(violation_code) as total_tickets
FROM ny_parking_violation_data_partitioned
WHERE issuer_precinct = 19
```

```
GROUP BY violation_code order by total_tickets desc limit 3;
```

Alternatively, below code could be used to get results for all 3 precincts together

```
SELECT Issuer_Precinct,Violation_Code, count(*) as Total_tickets

FROM ny_parking_violation_data

WHERE Issuer_Precinct in (18,19,14)

GROUP BY Issuer_Precinct,Violation_Code order by Total_tickets desc limit 10;
```

5.) Find out the properties of parking violations across different times of the day: The Violation Time field is specified in a strange format. Find a way to make this into a time attribute that you can use to divide into groups.

```
SELECT FROM_UNIXTIME(UNIX_TIMESTAMP(REGEXP_EXTRACT(violation_time,'(.*)[A-
Z]',1),'HHmm'),"HH:mm") as time FROM ny_parking_violation_data limit 5;
```

6.) Divide 24 hours into 6 equal discrete bins of time. The intervals you choose are at your discretion. For each of these groups, find the 3 most commonly occurring violations

```
CREATE VIEW nyparking_violation PARTITIONED on (Violation_Code) AS

SELECT Summons_Number, Violation_Time, Issuer_Precinct,

CASE

WHEN SUBSTRING(Violation_Time,1,2) IN ('00','01','02','03','12')

AND UPPER(SUBSTRING (Violation_Time,-1))='A' THEN 1

WHEN SUBSTRING(Violation_Time,1,2) IN ('04','05','06','07')

AND UPPER(SUBSTRING (Violation_Time,-1))='A' THEN 2

WHEN SUBSTRING(Violation_Time,1,2) IN ('08','09','10','11')

AND UPPER(SUBSTRING (Violation_Time,-1))='A' THEN 3

WHEN substring(Violation_Time,1,2) IN ('12','00','01','02','03') AND

UPPER(SUBSTRING (Violation_Time,-1))='P' THEN 4

WHEN SUBSTRING(Violation_Time,1,2) IN ('04','05','06','07') AND

UPPER(SUBSTRING(Violation_Time,-1))='P' THEN 5

WHEN SUBSTRING(Violation_Time,1,2) IN ('08','09','10','11') AND
```

```
UPPER(SUBSTRING(Violation_Time,-1))='P' THEN 6

ELSE NULL

END AS Violation_Time_bin, Violation_Code

FROM ny_parking_violation_data

WHERE Violation_Time is NOT NULL OR (length(Violation_Time)=5

AND UPPER(SUBSTRING(Violation_Time,-1))in ('A','P')

AND SUBSTRING(Violation_Time,1,2) in ('00','01','02','03','04','05','06','07',
'08','09','10','11','12'));
```

Queries to break down of total tickets issued divided in each view

- Bin 1

```
SELECT violation_code, count(*) tickets_issued

FROM nyparking_violation

WHERE violation_Time_bin == 1 GROUP BY violation_code ORDER BY tickets_issued desc LIMIT
3;
```

- Bin 2

```
SELECT violation_code, count(*) tickets_issued

FROM nyparking_violation

WHERE violation_Time_bin == 2 GROUP BY violation_code ORDER BY tickets_issued desc LIMIT
3;
```

- Bin 3

```
SELECT violation_code, count(*) tickets_issued

FROM nyparking_violation

WHERE violation_Time_bin == 3 GROUP BY violation_code ORDER BY tickets_issued desc LIMIT
3;
```

- Bin 4

```
SELECT violation_code, count(*) tickets_issued

FROM nyparking_violation

WHERE violation_Time_bin == 4 GROUP BY violation_code ORDER BY tickets_issued desc LIMIT
3;
```

- Bin 5

```
SELECT violation_code, count(*) tickets_issued
```

```
FROM nyparking_violation
WHERE violation_Time_bin == 5 GROUP BY violation_code ORDER BY tickets_issued desc LIMIT 3;
```

- Bin 6

```
SELECT violation_code, count(*) tickets_issued
FROM nyparking_violation
WHERE violation_Time_bin == 6 GROUP BY violation_code ORDER BY tickets_issued desc LIMIT 3;
```

7.) Now, try another direction. For the 3 most commonly occurring violation codes, find the most common times of day (in terms of the bins from the previous part)

```
SELECT violation_code Violations, Violation_Time_bin  Time_Bin, count(*) AS Tickets_Issued
FROM nyparking_violation
GROUP BY violation_code, Violation_Time_bin ORDER BY Tickets_Issued desc LIMIT 10;
```

8.) Let's try and find some seasonality in this data

a.) First, divide the year into some number of seasons, and find frequencies of tickets for each season. (Hint: A quick Google search reveals the following seasons in NYC: Spring (March, April, March); Summer (June, July, August); Fall (September, October, November); Winter (December, January, February))

Consider Seasons as below and divide these into Views

Spring: March, April, May

Summer: June, July, August

Fall: September, October, November

Winter: December, January, February

By Create View on table with bins for each season-

```
CREATE VIEW parking_violation_by_seasons_bin  AS
SELECT Summons_number,
CASE
WHEN SUBSTRING(issue_date,4,2) IN ('03', '04', '05') THEN 'Spring'
WHEN SUBSTRING(issue_date,4,2) IN ('06', '07', '08') THEN 'Summer'
WHEN SUBSTRING(issue_date,4,2) IN ('09', '10', '11') THEN 'Fall'
WHEN SUBSTRING(issue_date,4,2) IN ('12', '01', '02') THEN 'Winter'
ELSE 'Unknown' END AS Season, violation_code
FROM ny_parking_violation_2017;
```

b.) Then, find the 3 most common violations for each of these seasons.

- Most common violation for Spring

```
SELECT violation_code, COUNT(*) AS Tickets_issued
FROM parking_violation_by_seasons_bin WHERE Season = 'Spring'
GROUP BY violation_code ORDER BY Tickets_issued desc LIMIT 3;
```

- Most common violation for Summer

```
SELECT violation_code, COUNT(*) AS Tickets_issued
FROM parking_violation_by_seasons_bin WHERE Season = 'Summer'
GROUP BY violation_code ORDER BY Tickets_issued desc LIMIT 3;
```

- Most common violation for Fall

```
SELECT violation_code, COUNT(*) AS Tickets_issued
FROM parking_violation_by_seasons_bin WHERE Season = 'Fall'
GROUP BY violation_code ORDER BY Tickets_issued desc LIMIT 3;
```

- Most common violation for Winter

```
SELECT violation_code, COUNT(*) AS Tickets_issued
FROM parking_violation_by_seasons_bin WHERE Season = 'Winter '
GROUP BY violation_code ORDER BY Tickets_issued desc LIMIT 3;
```