Author:
Bhawesh Mehta
Data Enthusiastic

Project Name:

NYC Parking Tickets: An Exploratory Analysis Using Hive

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 analyze 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

Action:Export the data in CSV Format from the site given above

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

Before Going for the assignment >>

Let's Load the data into HDFS file storage system from local

The file named **parking_vio.csv** is currently in **local system**
Local system means the **file system of the container** i am in.

Please Note:

(**Local System** does not **refer** to the **local computer file system**)

```
# ls
Dockerfile  README.md              agent_performance.csv  docker-compose.yml  hadoop-hive.env  startup.sh
Makefile    agent_loging_report.csv  conf                 entrypoint.sh       parking_vio.csv  test.csv
#
```

Now I will be moving this file from container file system to hadoop file system

Before loading the file in hadoop environment let's check in the current directory in my hadoop environment.

```
# hadoop fs -ls /
Found 5 items
drwxr-xr-x   - root supergroup          0 2023-03-21 10:54 /inner_join
drwxr-xr-x   - root supergroup          0 2023-03-21 11:16 /join
drwxr-xr-x   - root supergroup          0 2023-03-21 10:47 /test
drwxrwxr-x   - root supergroup          0 2023-03-20 19:40 /tmp
drwxr-xr-x   - root supergroup          0 2023-03-21 10:59 /user
#
```

Now loading data into **hdfs** file system using **put** command

Here I have loaded the file inside the **local_to_hdfs** directory in hadoop

```
# hadoop fs -put /app/parking_vio.csv /local_to_hdfs
# hadoop fs -ls /
Found 6 items
drwxr-xr-x   - root supergroup          0 2023-03-21 10:54 /inner_join
drwxr-xr-x   - root supergroup          0 2023-03-21 11:16 /join
-rw-r--r--   3 root supergroup 2086913576 2023-03-31 19:02 /local_to_hdfs
drwxr-xr-x   - root supergroup          0 2023-03-21 10:47 /test
drwxrwxr-x   - root supergroup          0 2023-03-20 19:40 /tmp
drwxr-xr-x   - root supergroup          0 2023-03-21 10:59 /user
#
```

Let me show you the content inside the **/local_to_hdfs** directory of **hdfs**

To see the content you have to use

hadoop fs -cat /local_to_hdfs

Question arises why the .csv file is not created ?

Answer because we have not created the local_to_hdfs folder

Now let me delete this first

hadoop fs -rm rf /local_to_hdfs

First **create this** folder and use **put** command to load the file

```
# hadoop fs -mkdir /local_to_hdfs
# hadoop fs -put /app/parking_vio.csv /local_to_hdfs
# hadoop fs -ls /app/local_to_hdfs
ls: `/app/local_to_hdfs': No such file or directory
# hadoop fs -ls /local_to_hdfs
Found 1 items
-rw-r--r--   3 root supergroup 2086913576 2023-03-31 19:23 /local_to_hdfs/parking_vio.csv
#
```

Here clearly you can see that now **.csv file** can be seen inside **local_to_hdfs** file system

The analysis can be divided into two parts:

Part-I

Examine the data:

Let's Examine the data in the hadoop file system

1. Find the total number of tickets for the year.
2. Find out how many unique states the cars which got parking tickets came from.
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)

Approach 1

Using
Simple Internal table

Lets create a **internal table** named **parking**

Challenge here is that **date column** contains these values in these format

03-09-2017

01/18/2017

That is in mm-dd-yyyy or mm-dd-yyyy

So our **approach** will be to create a **temporary** table first defining them in **string** format and then **loading** the **data** from that **temp** table to the main table in data format in **hive**.
Before loading the data in main table we will replace / with - so that uniformity is maintained

So let's try this

By **Default hive stores** the date in **yyyy-MM-dd** format

Lets create a reference table named **violation_ref**

```
hive> create table violation_ref
    > (
    > summons_no int,
    > plate_id string,
    > registration_state string,
    > plate_type string,
    > issue_date string,
    > violation_code int,
    > vehicle_body_type string,
    > vehicle_make string,
    > issuing_agency string,
    > street_code_1 int,
    > street_code_2 int,
    > street_code_3 int,
    > vehicle_exp_date int,
    > violation_location int,
    > violation_precinct int,
    > issuer_precinct int,
    > issuer_code int,
    > issuer_command string,
    > issuer_squad string,
    > violation_time string,
    > time_first_observed string,
    > violation_county string,
    > violation_in_front_of_or_opp string,
    > house_no string,
```

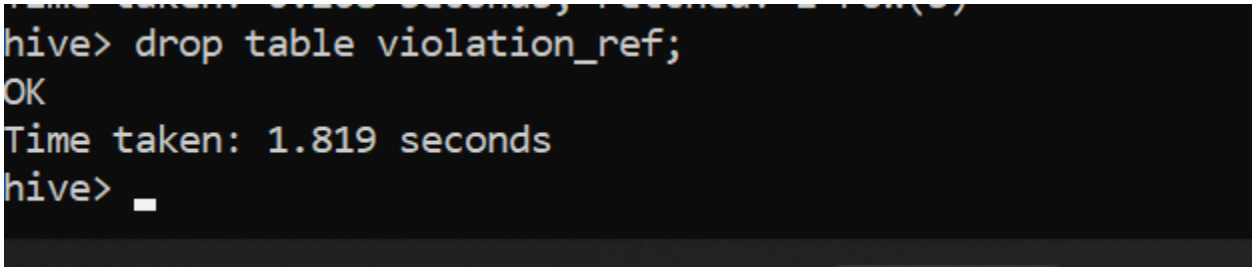
```
    > street_no string,
    > intersection_street string,
    > date_first_observed string,
    > law_section int,
    > sub_division string,
    > violation_legal_code string,
    > days_parking_in_effect string,
    > from_hours_in_effect string,
    > to_hours_in_effect string,
    > vehicle_color string,
    > unregistered_vehicle int,
    > vehicle_year string,
    > meter_no string,
    > feet_from_curb string,
    > 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 ',';
OK
Time taken: 8.546 seconds
hive>
```

Lets **load data** from our **hdfs file system** to this table named **violation_ref**

```
hive> load data inpath '/local_to_hdfs/parking_vio.csv' into table violation_ref;
Loading data to table hive_db.violation_ref
OK
Time taken: 2.949 seconds
hive>
```

We did a **mistake** here while creating the **table** we should have **escaped** the first row as it contains the **column name**

Let's **drop** the **violation_ref** table



Now again create **violation_ref** table

```
hive> create table violation_ref
    > (
    > summons_no int,
    > plate_id string,
    > registration_state string,
    > plate_type string,
    > issue_date string,
    > violation_code int,
    > vehicle_body_type string,
    > vehicle_make string,
    > issuing_agency string,
    > street_code_1 int,
    > street_code_2 int,
    > street_code_3 int,
    > vehicle_exp_date int,
    > violation_location int,
    > violation_precinct int,
    > issuer_precinct int,
    > issuer_code int,
    > issuer_command string,
    > issuer_squad string,
    > violation_time string,
    > time_first_observed string,
    > violation_county string,
    > violation_in_front_of_or_opp string,
    > house_no string,
    > street_no string,
    > intersection_street string,
    > date_first_observed string,
    > 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 int,
```

```
    > vehicle_year string,
    > meter_no string,
    > feet_from_curb string,
    > 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");
OK
Time taken: 0.18 seconds
hive>
```

Note how i have used **tblproperties(“skip.header.line.count”=”1”)**

Loading data from **hdfs** file system

Here I noticed one thing then when i dropped the table that the file inside the hdfs file system also got deleted

```
hive> load data inpath '/local_to_hdfs/parking_vio.csv' into table violation_ref ;
Loading data to table hive_db.violation_ref
OK
Time taken: 1.736 seconds
hive>
```

Checking if the table has data in **hdfs** file system or not

```
# hadoop fs -ls /user/hive/warehouse/hive_db.db
Found 6 items
drwxrwxr-x   - root supergroup          0 2023-03-21 07:10 /user/hive/warehouse/hive_db.db/agent_login_main
drwxrwxr-x   - root supergroup          0 2023-03-21 11:47 /user/hive/warehouse/hive_db.db/agent_login_main_part_buck
drwxrwxr-x   - root supergroup          0 2023-03-21 06:58 /user/hive/warehouse/hive_db.db/agent_login_ref
drwxrwxr-x   - root supergroup          0 2023-03-21 07:41 /user/hive/warehouse/hive_db.db/agent_performance_main
drwxrwxr-x   - root supergroup          0 2023-03-21 07:31 /user/hive/warehouse/hive_db.db/agent_performance_ref
drwxrwxr-x   - root supergroup          0 2023-04-07 17:12 /user/hive/warehouse/hive_db.db/violation_ref
# hadoop fs -ls /user/hive/warehouse/hive_db.db/violation_ref
Found 1 items
-rwxrwxr-x   3 root supergroup 2086913576 2023-04-07 17:11 /user/hive/warehouse/hive_db.db/violation_ref/parking_vio.csv
# hadoop fs -ls /user/hive/warehouse/hive_db.db/violation_ref/parking_vio.csv
```

Now lets create a **violation_main** table with **date column** defined

Note:
By **Default hive stores** the date in **yyyy-MM-dd** format

```
hive> insert into violation_main select
summons_no,plate_id,registration_state,plate_type,FROM_UNIXTIME(UNIX_TIMEST
AMP(issue_date, 'MM/dd/yyyy'), 'yyyy-MM-dd') as
issue_date,violation_code,vehicle_body_type,vehicle_make,issuing_agency,str
eet_code_1,street_code_2,street_code_3,vehicle_exp_date,violation_location,
violation_precinct,issuer_precinct,issuer_code,issuer_command,issuer_squad,
violation_time,time_first_observed,violation_county,violation_in_front_of_o
r_opp,house_no,street_no,intersection_street,date_first_observed,law_sectio
n,sub_division,violation_legal_code,days_parking_in_effect,from_hours_in_ef
fect,to_hours_in_effect,vehicle_color,unregistered_vehicle,vehicle_year,met
er_no,feet_from_curb,violation_post_code,violation_description,no_standing_
or_stopping_violation,hydrant_violation,double_parking_violation from
violation_ref;
```

Now let's check how **violation_main** table looks in **hdfs** file system

```
# hadoop fs -ls /user/hive/warehouse/hive_db.db/violation_main
Found 8 items
-rwxrwxr-x   3 root supergroup  259603163 2023-04-07 18:04 /user/hive/warehouse/hive_db.db/violation_main/000000_0
-rwxrwxr-x   3 root supergroup  259606380 2023-04-07 18:04 /user/hive/warehouse/hive_db.db/violation_main/000001_0
-rwxrwxr-x   3 root supergroup  259602811 2023-04-07 18:04 /user/hive/warehouse/hive_db.db/violation_main/000002_0
-rwxrwxr-x   3 root supergroup  259609772 2023-04-07 18:05 /user/hive/warehouse/hive_db.db/violation_main/000003_0
-rwxrwxr-x   3 root supergroup  259604818 2023-04-07 18:05 /user/hive/warehouse/hive_db.db/violation_main/000004_0
-rwxrwxr-x   3 root supergroup  259606283 2023-04-07 18:06 /user/hive/warehouse/hive_db.db/violation_main/000005_0
-rwxrwxr-x   3 root supergroup  259606952 2023-04-07 18:06 /user/hive/warehouse/hive_db.db/violation_main/000006_0
-rwxrwxr-x   3 root supergroup  201403532 2023-04-07 18:06 /user/hive/warehouse/hive_db.db/violation_main/000007_0
```

Notice how file is splitted into **multiple sub files** starting with **000000_0**

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

```
hive> select issue_date,count(*) from violation_main group by issue_date;
```

But we need the data for **2017** only

Sol let's modify this and create a **temp table** with data of year **2017** only

```
hive> create table parking_vio_2017
    > (
    > summons_no 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_code_1 int,
    > street_code_2 int,
    > street_code_3 int,
    > vehicle_exp_date int,
    > violation_location int,
    > violation_precinct int,
    > issuer_precinct int,
    > issuer_code int,
    > issuer_command string,
    > issuer_squad string,
    > violation_time string,
    > time_first_observed string,
    > violation_county string,
    > violation_in_front_of_or_opp string,
    > house_no string,
    > street_no string,
    > intersection_street string,
```

```
    > date_first_observed string,
    > 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 int,
    > vehicle_year string,
    > meter_no string,
    > feet_from_curb string,
    > 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 ',';
OK
Time taken: 12.221 seconds
hive>
```

```
hive> insert into parking_vio_2017 select
summons_no,plate_id,registration_state,plate_type,issue_date,violation_code
,vehicle_body_type,vehicle_make,issuing_agency,street_code_1,street_code_2,
street_code_3,vehicle_exp_date,violation_location,violation_precinct,issuer
_precinct,issuer_code,issuer_command,issuer_squad,violation_time,time_first
_observed,violation_county,violation_in_front_of_or_opp,house_no,street_no,
intersection_street,date_first_observed,law_section,sub_division,violation_
legal_code,days_parking_in_effect,from_hours_in_effect,to_hours_in_effect,v
ehicle_color,unregistered_vehicle,vehicle_year,meter_no,feet_from_curb,viol
ation_post_code,violation_description,no_standing_or_stopping_violation,hyd
rant_violation,double_parking_violation from violation_main where
year(issue_date)=2017;
```

Let's check if **data** in loaded in the **hdfs warehouse**

```
# hadoop fs -ls /user/hive/warehouse/hive_db.db/parking_vio_2017
```

```
Found 8 items
-rwxrwxr-x   3 root supergroup  133306082 2023-04-08 11:54
/user/hive/warehouse/hive_db.db/parking_vio_2017/000000_0
-rwxrwxr-x   3 root supergroup  133615823 2023-04-08 11:55
/user/hive/warehouse/hive_db.db/parking_vio_2017/000001_0
-rwxrwxr-x   3 root supergroup  133371174 2023-04-08 11:55
/user/hive/warehouse/hive_db.db/parking_vio_2017/000002_0
-rwxrwxr-x   3 root supergroup  133720470 2023-04-08 11:55
/user/hive/warehouse/hive_db.db/parking_vio_2017/000003_0
-rwxrwxr-x   3 root supergroup  133466345 2023-04-08 11:55
/user/hive/warehouse/hive_db.db/parking_vio_2017/000004_0
-rwxrwxr-x   3 root supergroup  133509997 2023-04-08 11:55
/user/hive/warehouse/hive_db.db/parking_vio_2017/000005_0
-rwxrwxr-x   3 root supergroup  133557192 2023-04-08 11:55
/user/hive/warehouse/hive_db.db/parking_vio_2017/000006_0
-rwxrwxr-x   3 root supergroup   80027489 2023-04-08 11:56
/user/hive/warehouse/hive_db.db/parking_vio_2017/000007_0
```

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

```
hive> select distinct registration_state from violation_main;
```

```
hive> select registration_state,count(*) as instance from parking_vio_2017
group by registration_state order by instance desc;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the
future versions. Consider using a different execution engine (i.e. spark,
tez) or using Hive 1.X releases.
Query ID = root_20230408120319_8c69e2b9-a0d2-4432-a51e-74cc7f583b36
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 4
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
```

```
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-08 12:03:23,028 Stage-1 map = 0%,  reduce = 0%
2023-04-08 12:03:27,707 Stage-1 map = 100%,  reduce = 0%
2023-04-08 12:03:37,722 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1615623838_0004
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-08 12:03:39,869 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local1643808020_0005
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 55123329229 HDFS Write: 8116602392 SUCCESS
Stage-Stage-2:  HDFS Read: 14141225856 HDFS Write: 2029150598 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
registration_state      instance
NY      4273941
NJ      475824
PA      140284
CT      70403
FL      69468
IN      45525
MA      38941
VA      34367
MD      30213
NC      27152
TX      18827
IL      18666
GA      17537
99      16055
AZ      12379
OH      12281
CA      12152
```

```
ME      10806
SC      10394
MN      10083
OK      9088
TN      8514
DE      7905
MI      7231
RI      5814
NH      4119
VT      3683
AL      3178
WA      3052
OR      2622
MO      2483
ON      2460
WI      2127
QB      1998
IA      1938
DC      1929
CO      1841
KY      1795
DP      1794
LA      1689
MS      1582
WV      1265
AR      994
SD      859
NM      792
ID      763
NV      725
KS      706
NE      704
UT      561
MT      505
GV      348
NS      322
AK      298
ND      254
WY      188
HI      156
AB      79
PE      61
NB      57
```

```
BC      54
PR      38
MB      17
SK      9
FO      8
Time taken: 20.127 seconds, Fetched: 65 row(s)
```

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)

```
hive> select count(*) from parking_vio_2017 where street_code_1 is null or
street_code_2 is null or street_code_3 is null or street_code_1=0 or
street_code_2=0 or street_code_3=0;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the
future versions. Consider using a different execution engine (i.e. spark,
tez) or using Hive 1.X releases.
Query ID = root_20230408122406_546bfba9-608e-49bb-b6cc-c6ce23ab817d
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-08 12:24:08,395 Stage-1 map = 0%,  reduce = 0%
2023-04-08 12:24:11,784 Stage-1 map = 100%,  reduce = 0%
2023-04-08 12:24:18,791 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1069774194_0007
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 44057236165 HDFS Write: 5072876495 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
_c0
1816816
Time taken: 12.794 seconds, Fetched: 1 row(s)
hive>
```

Part-II: Aggregation tasks

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

```
hive> select violation_code,count(*) as instance from parking_vio_2017
group by violation_code order by instance desc limit 5;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the
future versions. Consider using a different execution engine (i.e. spark,
tez) or using Hive 1.X releases.
Query ID = root_20230408123022_9378794a-2f09-4e0b-9bad-3113d0113f09
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 4
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-08 12:30:31,657 Stage-1 map = 0%,  reduce = 0%
2023-04-08 12:30:40,682 Stage-1 map = 100%,  reduce = 0%
2023-04-08 12:30:46,688 Stage-1 map = 33%,  reduce = 0%
2023-04-08 12:30:48,710 Stage-1 map = 100%,  reduce = 0%
2023-04-08 12:30:58,737 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local488446953_0008
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-08 12:31:01,739 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local1353223834_0009
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 79473118957 HDFS Write: 8116602392 SUCCESS
```

```
Stage-Stage-2:  HDFS Read: 20228673288 HDFS Write: 2029150598 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
violation_code  instance
21      768082
36      662765
38      542079
14      476660
20      319646
Time taken: 38.911 seconds, Fetched: 5 row(s)
hive>
```

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

For Vehicle body type

```
hive> select vehicle_body_type,count(*) as instance from parking_vio_2017
group by vehicle_body_type order by instance desc limit 5;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the
future versions. Consider using a different execution engine (i.e. spark,
tez) or using Hive 1.X releases.
Query ID = root_20230408123551_a96f78b5-6b0a-4fae-8fd6-54ec188a7fc3
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 4
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-08 12:35:54,477 Stage-1 map = 0%,  reduce = 0%
2023-04-08 12:36:00,484 Stage-1 map = 8%,  reduce = 0%
2023-04-08 12:36:03,487 Stage-1 map = 100%,  reduce = 0%
2023-04-08 12:36:09,497 Stage-1 map = 25%,  reduce = 0%
2023-04-08 12:36:12,502 Stage-1 map = 100%,  reduce = 0%
2023-04-08 12:36:17,508 Stage-1 map = 75%,  reduce = 0%
```

```
2023-04-08 12:36:18,509 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local198719902_0010
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-08 12:36:20,251 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local457352855_0011
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 87589715533 HDFS Write: 8116602392 SUCCESS
Stage-Stage-2:  HDFS Read: 22257822432 HDFS Write: 2029150598 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
vehicle_body_type       instance
SUBN    1883953
4DSD    1547307
VAN     724025
DELV    358982
SDN     194197
Time taken: 28.821 seconds, Fetched: 5 row(s)
hive>
```

For vehicle make

```
hive> select vehicle_make,count(*) as instance from parking_vio_2017 group
by vehicle_make order by instance desc limit 5;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the
future versions. Consider using a different execution engine (i.e. spark,
tez) or using Hive 1.X releases.
Query ID = root_20230408123750_9647cd58-0775-4c05-af32-27742608aed6
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 4
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
```

```
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-08 12:37:54,203 Stage-1 map = 0%,  reduce = 0%
2023-04-08 12:37:56,206 Stage-1 map = 100%,  reduce = 0%
2023-04-08 12:38:03,214 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1454523152_0012
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-08 12:38:05,257 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local1060513731_0013
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 95706312109 HDFS Write: 8116602392 SUCCESS
Stage-Stage-2:  HDFS Read: 24286971576 HDFS Write: 2029150598 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
vehicle_make    instance
FORD    636842
TOYOT   605290
HONDA   538884
NISSA   462017
CHEVR   356032
Time taken: 14.345 seconds, Fetched: 5 row(s)
hive>
```

3) 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)

```
hive> select violation_precinct,count(*) as instances from parking_vio_2017
group by violation_precinct order by instances desc limit 5;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the
future versions. Consider using a different execution engine (i.e. spark,
```

```
tez) or using Hive 1.X releases.
Query ID = root_20230408124537_11dc8925-f0f4-4397-8eee-b5be88d363f3
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 4
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-08 12:45:40,763 Stage-1 map = 0%,  reduce = 0%
2023-04-08 12:45:43,765 Stage-1 map = 100%,  reduce = 0%
2023-04-08 12:45:49,771 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1258832131_0014
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-08 12:45:51,660 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local1685272680_0015
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 103822908685 HDFS Write: 8116602392 SUCCESS
Stage-Stage-2:  HDFS Read: 26316120720 HDFS Write: 2029150598 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
violation_precinct      instances
0       925596
19      274443
14      203552
1       174702
18      169131
Time taken: 13.861 seconds, Fetched: 5 row(s)
hive>
```

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

```
hive> select issuer_precinct,count(*) as instances from parking_vio_2017
group by issuer_precinct order by instances desc limit 5;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the
future versions. Consider using a different execution engine (i.e. spark,
tez) or using Hive 1.X releases.
Query ID = root_20230408124648_200df340-8b5c-4e61-9eb4-5a864c9ab815
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 4
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-08 12:46:50,404 Stage-1 map = 0%,  reduce = 0%
2023-04-08 12:46:52,406 Stage-1 map = 100%,  reduce = 0%
2023-04-08 12:46:58,414 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local523335232_0016
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-08 12:47:00,186 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local1790657867_0017
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 111939505261 HDFS Write: 8116602392 SUCCESS
Stage-Stage-2:  HDFS Read: 28345269864 HDFS Write: 2029150598 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
issuer_precinct instances
0       1078403
19      266959
14      200494
```

```
1       168740
18      162994
Time taken: 11.936 seconds, Fetched: 5 row(s)
hive>
```

4) 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?

```
hive> select issuer_precinct,violation_code,count(*) as instance from
parking_vio_2017 group by issuer_precinct,violation_code order by
issuer_precinct,instance desc;


hive> select issuer_precinct,count(*) as instance from parking_vio_2017
where issuer_precinct!=0 group by issuer_precinct,violation_code order by
instance desc limit 3;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the
future versions. Consider using a different execution engine (i.e. spark,
tez) or using Hive 1.X releases.
Query ID = root_20230408131017_0062681a-3eaa-4524-8013-469a72d4e365
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 4
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-08 13:10:20,140 Stage-1 map = 0%,  reduce = 0%
2023-04-08 13:10:22,142 Stage-1 map = 100%,  reduce = 0%
2023-04-08 13:10:28,147 Stage-1 map = 75%,  reduce = 0%
2023-04-08 13:10:29,150 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1999220878_0027
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
```

```
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-08 13:10:30,934 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local1703760097_0028
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 160639084717 HDFS Write: 8116602392 SUCCESS
Stage-Stage-2:  HDFS Read: 40520164728 HDFS Write: 2029150598 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
issuer_precinct instance
18      50150
19      48444
14      45036
Time taken: 12.999 seconds, Fetched: 3 row(s)
hive>
```

issuer_precinct are 18,19,14

Now let's examine **violation code** frequency in **18,19,14 issuer_precinct**

```
hive> select issuer_precinct,violation_code,count(*) as instance from
parking_vio_2017 where issuer_precinct in (18,19,14) group by
issuer_precinct,violation_code order by issuer_precinct,instance desc;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the
future versions. Consider using a different execution engine (i.e. spark,
tez) or using Hive 1.X releases.
Query ID = root_20230408131311_c3b7e27c-4880-4695-987e-ca09a6a842e9
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 4
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-08 13:13:13,474 Stage-1 map = 0%,  reduce = 0%
```

```
2023-04-08 13:13:15,475 Stage-1 map = 100%,  reduce = 0%
2023-04-08 13:13:21,483 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1448052520_0029
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-04-08 13:13:23,296 Stage-2 map = 100%,  reduce = 100%
Ended Job = job_local168259142_0030
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 168755681293 HDFS Write: 8116602392 SUCCESS
Stage-Stage-2:  HDFS Read: 42549313872 HDFS Write: 2029150598 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
issuer_precinct violation_code  instance
14      14      45036
14      69      30464
14      31      22555
14      47      18364
14      42      10027
14      46      7679
14      19      7030
14      84      6743
14      82      5052
14      40      3582
14      17      3534
14      38      3269
14      9       2874
14      20      2761
14      71      2757
14      13      2701
14      48      2439
14      89      1960
14      50      1824
14      11      1745
14      79      1495
14      70      1461
14      10      1319
```

```
14      37      1256
14      64      1070
14      23      1044
14      21      1029
14      53      953
14      24      946
14      16      940
14      74      768
14      35      675
14      8       588
14      51      559
14      52      549
14      45      526
14      73      253
14      1       247
14      3       241
14      30      239
14      78      227
14      18      213
14      26      212
14      72      200
14      85      152
14      77      138
14      83      108
14      49      84
14      61      74
14      98      67
14      62      66
14      41      63
14      67      62
14      75      43
14      60      41
14      43      40
14      2       32
14      66      22
14      59      21
14      22      19
14      68      12
14      80      8
14      39      7
14      27      7
14      29      4
14      12      4
```

```
14      4       4
14      81      2
14      99      2
14      86      1
14      0       1
14      56      1
14      91      1
14      96      1
14      54      1
18      14      50150
18      69      20189
18      47      14107
18      31      11893
18      46      7863
18      42      6190
18      38      6176
18      84      5189
18      19      4580
18      20      4114
18      40      3350
18      16      2632
18      82      2242
18      37      2236
18      11      2059
18      79      2006
18      71      1977
18      13      1816
18      17      1653
18      21      1452
18      10      1373
18      70      1101
18      35      1080
18      23      841
18      24      683
18      9       634
18      48      624
18      50      602
18      45      601
18      74      518
18      53      431
18      64      428
18      51      359
18      78      279
```

```
18      8       215
18      98      173
18      77      123
18      18      108
18      66      99
18      73      92
18      72      80
18      89      79
18      49      69
18      75      56
18      26      53
18      85      52
18      39      50
18      1       46
18      60      42
18      41      35
18      83      33
18      61      27
18      43      18
18      68      15
18      67      15
18      30      14
18      3       14
18      22      13
18      62      9
18      59      9
18      80      7
18      2       3
18      29      3
18      99      3
18      33      2
18      0       2
18      12      1
18      52      1
18      15      1
18      96      1
18      95      1
18      81      1
18      27      1
19      46      48444
19      38      36386
19      37      36056
19      14      29797
```

```
19      21      28414
19      20      14629
19      40      11416
19      16      9926
19      71      7493
19      19      6856
19      10      5643
19      84      4910
19      70      4459
19      18      3148
19      69      2910
19      31      2080
19      53      1736
19      50      1483
19      17      1464
19      48      1460
19      74      1329
19      24      1029
19      42      903
19      82      888
19      47      702
19      51      539
19      9       480
19      13      445
19      64      389
19      45      241
19      23      207
19      78      189
19      11      183
19      98      92
19      75      91
19      85      75
19      72      63
19      61      60
19      83      50
19      73      44
19      41      43
19      39      27
19      30      27
19      60      26
19      8       24
19      79      13
19      43      13
```

```
19      52      12
19      68      11
19      35      7
19      49      6
19      67      5
19      66      4
19      26      4
19      81      3
19      62      3
19      33      3
19      80      3
19      99      3
19      59      2
19      0       2
19      4       1
19      89      1
19      77      1
19      91      1
19      29      1
19      2       1
19      22      1
19      32      1
19      12      1
Time taken: 11.828 seconds, Fetched: 218 row(s)
hive>
```

Precinct 18 and Precinct 14 has more less similar top violation code.

But Precinct 19 has very different top violation code.

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.

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

```
select summons_no, violation_code , 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
from parking_vio_2017 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')
)
```

Now Let's find **3 most** commonly **occurring** violations
For violation_time_bin=1

```
with bin_table as (
select summons_no, violation_code , 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
from parking_vio_2017 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')
)
)

select violation_time_bin,violation_code,count(*) as instance from
bin_table
where violation_time_bin=1
group by violation_time_bin,violation_code
order by instance desc
limit 3

violation_time_bin      violation_code  instance
1       21      36957
1       40      25866
1       78      15528
Time taken: 16.955 seconds, Fetched: 3 row(s)
hive>
```

Similarly trying for violation_time_bin for 2,3,4,5,6

Three most **commonly violation_codes** are

21,36,38

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)

```
with bin_table as (
select summons_no, violation_code , 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
from parking_vio_2017 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')
)
)

select violation_time_bin,count(*) as instance from bin_table
where violation_code in (21,36,38)
group by violation_time_bin
order by instance desc
limit 3
```

```
violation_time_bin      instance
3       1122795
4       601699
5       116648
Time taken: 16.351 seconds, Fetched: 3 row(s)
hive>
```

Bins **3, 4, 5** are having most violations

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))

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

```
with season_data as (
select case when month(issue_date) in (1,2,12) then 'winter' when
month(issue_date) in (3,4,5) then 'spring' when month(issue_date) in
(6,7,8) then 'summer' else 'fall' end as season,violation_code,count(*) as
instances
from parking_vio_2017
group by case when month(issue_date) in (1,2,12) then 'winter' when
month(issue_date) in (3,4,5) then 'spring' when month(issue_date) in
(6,7,8) then 'summer' else 'fall' end,violation_code
)
select * from (
select season,violation_code,instances,dense_rank() over (partition by
season,violation_code order by instances desc) as rank
from season_data
)b


b.season        b.violation_code        b.instances     b.rank
fall    46      231     1
fall    21      128     2
fall    40      116     3
spring  21      402424  1
spring  36      344834  2
spring  38      271167  3
summer  21      127350  1
summer  36      96663   2
summer  38      83518   3
winter  21      238180  1
winter  36      221268  2
winter  38      187386  3
Time taken: 13.933 seconds, Fetched: 12 row(s)
hive>
```

Now let's solve all above questions with some optimization techniques

Hive is better for **orc type format**

So here we will create an **orc table** with **dynamic partition** on **issue_date,violation_code and bucketing on summons_no**

Here We will try to implement all the concepts learnt

Step 1
Lets create an **orc table** named **parking_vio_2017_orc** from **reference** table **parking_vio_2017**

```
hive> create table parking_vio_2017_orc
    > (
    > summons_no 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_code_1 int,
    > street_code_2 int,
    > street_code_3 int,
    > vehicle_exp_date int,
    > violation_location int,
    > violation_precinct int,
    > issuer_precinct int,
    > issuer_code int,
    > issuer_command string,
    > issuer_squad string,
    > violation_time string,
    > time_first_observed string,
    > violation_county string,
    > violation_in_front_of_or_opp string,
    > house_no string,
    > street_no string,
```

```
    > intersection_street string,
    > date_first_observed string,
    > 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 int,
    > vehicle_year string,
    > meter_no string,
    > feet_from_curb string,
    > violation_post_code string,
    > violation_description string,
    > no_standing_or_stopping_violation string,
    > hydrant_violation string,
    > double_parking_violation string
    > )
    > stored as orc;
OK
Time taken: 6.189 seconds
hive>
```

Lets **load data** into this **orc table** from the ref table named **parking_vio_2017**

```
hive> from parking_vio_2017 insert overwrite table parking_vio_2017_orc
select *;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the
future versions. Consider using a different execution engine (i.e. spark,
tez) or using Hive 1.X releases.
Query ID = root_20230408183337_5e13629f-b635-4151-8dc9-5e1decd2870c
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Job running in-process (local Hadoop)
2023-04-08 18:33:40,787 Stage-1 map = 0%,  reduce = 0%
2023-04-08 18:33:58,800 Stage-1 map = 12%,  reduce = 0%
2023-04-08 18:34:11,817 Stage-1 map = 100%,  reduce = 0%
2023-04-08 18:34:17,822 Stage-1 map = 25%,  reduce = 0%
2023-04-08 18:34:26,828 Stage-1 map = 37%,  reduce = 0%
2023-04-08 18:34:40,838 Stage-1 map = 100%,  reduce = 0%
```

```
2023-04-08 18:34:46,844 Stage-1 map = 50%,  reduce = 0%
2023-04-08 18:34:54,851 Stage-1 map = 62%,  reduce = 0%
2023-04-08 18:35:08,870 Stage-1 map = 100%,  reduce = 0%
2023-04-08 18:35:14,874 Stage-1 map = 75%,  reduce = 0%
2023-04-08 18:35:23,948 Stage-1 map = 91%,  reduce = 0%
2023-04-08 18:35:29,952 Stage-1 map = 100%,  reduce = 0%
Ended Job = job_local1833751516_0015
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to directory
hdfs://namenode:8020/user/hive/warehouse/hive_db.db/parking_vio_2017_orc/.h
ive-staging_hive_2023-04-08_18-33-37_677_1074309894541524570-1/-ext-10000
Loading data to table hive_db.parking_vio_2017_orc
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 35083110397 HDFS Write: 510857609 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
parking_vio_2017.summons_no     parking_vio_2017.plate_id
parking_vio_2017.registration_state     parking_vio_2017.plate_type
parking_vio_2017.issue_date     parking_vio_2017.violation_code
parking_vio_2017.vehicle_body_type      parking_vio_2017.vehicle_make
parking_vio_2017.issuing_agency parking_vio_2017.street_code_1
parking_vio_2017.street_code_2      parking_vio_2017.street_code_3
parking_vio_2017.vehicle_exp_date       parking_vio_2017.violation_location
parking_vio_2017.violation_precinct parking_vio_2017.issuer_precinct
parking_vio_2017.issuer_code    parking_vio_2017.issuer_command
parking_vio_2017.issuer_squad   parking_vio_2017.violation_time
parking_vio_2017.time_first_observed    parking_vio_2017.violation_county
parking_vio_2017.violation_in_front_of_or_opp   parking_vio_2017.house_no
parking_vio_2017.street_no  parking_vio_2017.intersection_street
parking_vio_2017.date_first_observed    parking_vio_2017.law_section
parking_vio_2017.sub_division   parking_vio_2017.violation_legal_code
parking_vio_2017.days_parking_in_effect
parking_vio_2017.from_hours_in_effect   parking_vio_2017.to_hours_in_effect
parking_vio_2017.vehicle_color      parking_vio_2017.unregistered_vehicle
parking_vio_2017.vehicle_year   parking_vio_2017.meter_no
parking_vio_2017.feet_from_curb parking_vio_2017.violation_post_code
parking_vio_2017.violation_description
parking_vio_2017.no_standing_or_stopping_violation
parking_vio_2017.hydrant_violation
parking_vio_2017.double_parking_violation
Time taken: 115.373 seconds
```

```
hive>
```

Let's check in hadoop environment how this orc file looks like

```
# hadoop fs -ls /user/hive/warehouse/hive_db.db/parking_vio_2017_orc
Found 4 items
-rwxrwxr-x   3 root supergroup   52140329 2023-04-08 18:34 /user/hive/warehouse/hive_db.db/parking_vio_2017_orc/000000_0
-rwxrwxr-x   3 root supergroup   52115932 2023-04-08 18:34 /user/hive/warehouse/hive_db.db/parking_vio_2017_orc/000001_0
-rwxrwxr-x   3 root supergroup   52118703 2023-04-08 18:35 /user/hive/warehouse/hive_db.db/parking_vio_2017_orc/000002_0
-rwxrwxr-x   3 root supergroup   41710121 2023-04-08 18:35 /user/hive/warehouse/hive_db.db/parking_vio_2017_orc/000003_0
#
```

Its taking very less time to query now

```
hive> select count(*) from parking_vio_2017_orc ;
OK
_c0
5431903
Time taken: 0.248 seconds, Fetched: 1 row(s)
hive>
```

Now let's create partitions and buckets

I am going to do **dynamic partition** on **issue_date,violation_code** and **bucketing** on summon_no

```
hive> create table parking_vio_2017_orc_part
    > (
    > summons_no int,
    > plate_id string,
    > registration_state string,
    > plate_type string,
    > vehicle_body_type string,
    > vehicle_make string,
    > issuing_agency string,
    > street_code_1 int,
    > street_code_2 int,
    > street_code_3 int,
```

```
    > vehicle_exp_date int,
    > violation_location int,
    > violation_precinct int,
    > issuer_precinct int,
    > issuer_code int,
    > issuer_command string,
    > issuer_squad string,
    > violation_time string,
    > time_first_observed string,
    > violation_county string,
    > violation_in_front_of_or_opp string,
    > house_no string,
    > street_no string,
    > intersection_street string,
    > date_first_observed string,
    > 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 int,
    > vehicle_year string,
    > meter_no string,
    > feet_from_curb string,
    > violation_post_code string,
    > violation_description string,
    > no_standing_or_stopping_violation string,
    > hydrant_violation string,
    > double_parking_violation string
    > )
    > partitioned by (issue_date date,violation_code int)
    > clustered by (summons_no)
    > sorted by (summons_no)
    > into 3 buckets;
OK
Time taken: 0.526 seconds
hive>
```

```
insert overwrite table parking_vio_2017_orc_part_buck partition
(issue_date,violation_code) select
summons_no,plate_id,registration_state,plate_type,vehicle_body_type,vehicle
_make,issuing_agency,street_code_1,street_code_2,street_code_3,vehicle_exp_
date,violation_location,violation_precinct,issuer_precinct,issuer_code,issu
er_command,issuer_squad,violation_time,time_first_observed,violation_county
,violation_in_front_of_or_opp,house_no,street_no,intersection_street,date_f
irst_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_no,feet_from_curb,violation_post_code,violation_
description,no_standing_or_stopping_violation,hydrant_violation,double_park
ing_violation,issue_date,violation_code from parking_vio_2017_orc
```

Now let's check how the **partitions** and bucketing looks in **hdfs** file system

```
#  hadoop fs -ls /user/hive/warehouse/hive_db.db/parking_vio_2017_orc_part_buck/issue_date=2017-02-07/violation_code=21
Found 3 items
-rwxrwxr-x   3 root supergroup        189 2023-04-08 19:39 /user/hive/warehouse/hive_db.db/parking_vio_2017_orc_part_buck/issue_date=2017-02-07/violation_code=21/000000_0
-rwxrwxr-x   3 root supergroup          0 2023-04-08 19:39 /user/hive/warehouse/hive_db.db/parking_vio_2017_orc_part_buck/issue_date=2017-02-07/violation_code=21/000001_0
-rwxrwxr-x   3 root supergroup          0 2023-04-08 19:39 /user/hive/warehouse/hive_db.db/parking_vio_2017_orc_part_buck/issue_date=2017-02-07/violation_code=21/000002_0
#
```