**Hive project -2**

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.

The analysis can be divided into two parts:

Part-I: Examine the data

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 )

Part-II: Aggregation tasks

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

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

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)

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

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?

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.

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

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)

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.

#creating the table:

create table parking(

Summons\_Number string,

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\_Code1 string,

Street\_Code2 string,

Street\_Code3 string,

Vehicle\_Expiration\_Date string,

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\_Opposite string,

House\_Number string,

Street\_Name string,

Intersecting\_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 int,

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 local inpath 'file:///parking.csv' into table parking;

#creating table with ORC format

create table parking\_orc(

Summons\_Number string,

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\_Code1 string,

Street\_Code2 string,

Street\_Code3 string,

Vehicle\_Expiration\_Date string,

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\_Opposite string,

House\_Number string,

Street\_Name string,

Intersecting\_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 int,

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)

Stored as ORC;

#inserting data to ORC table only 2017 year data

insert overwrite table parking\_orc select \* from parking where year(to\_date(from\_unixtime(unix\_timestamp(issue\_Date,'MM/dd/yyy')))) = '2017' ;

#enabling vectorization

set hive.vectorized.execution.enabled = true;

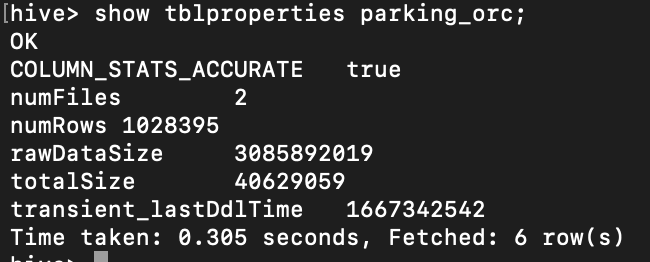
#**Vectorized query execution** streamlines operations by processing a block of 1024 rows at a time. Simple operations like arithmetic and comparisons are done by quickly iterating through the vectors in a tight loop, with no or very few function calls or conditional branches inside the loop.

**Note:**To use vectorized query execution, you must store your data in ORC format.

Part-I: Examine the data

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

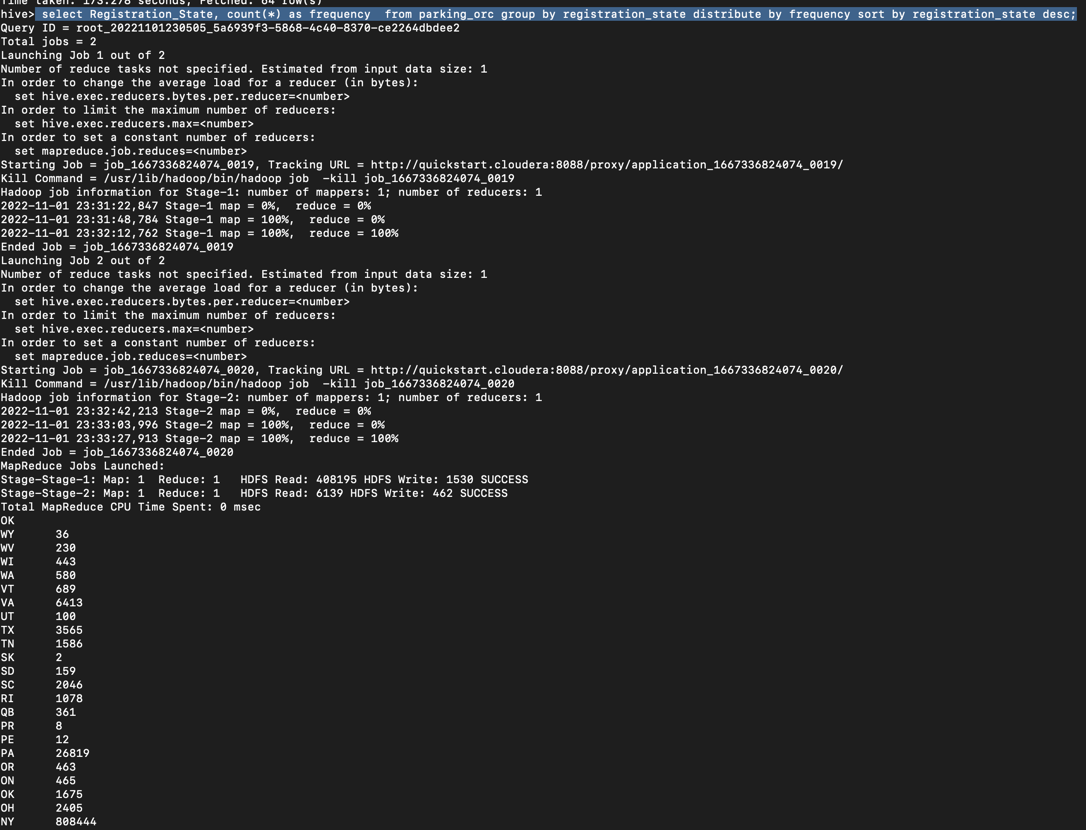
show tblproperties parking\_orc;



#no duplicates are in ticket numbers (i.e., select (\*),count(distinct(Summons\_Number)) from parking\_orc)

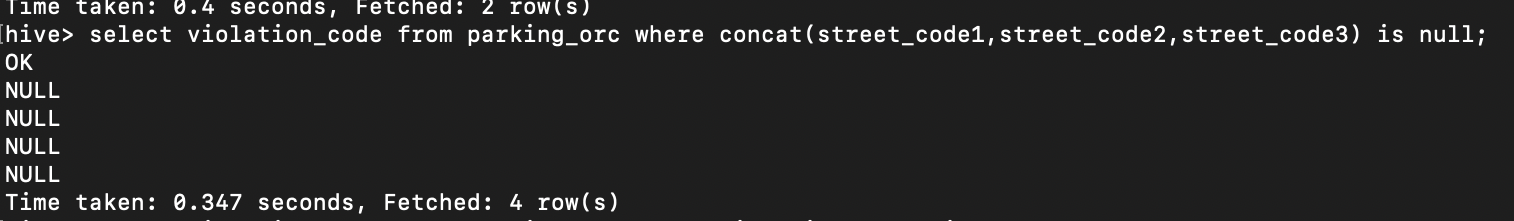
**2.) Find out how many unique states the cars which got parking tickets came from.**

select Registration\_State, count(\*) as frequency from parking\_orc group by registration\_state distribute by frequency sort by registration\_state desc;

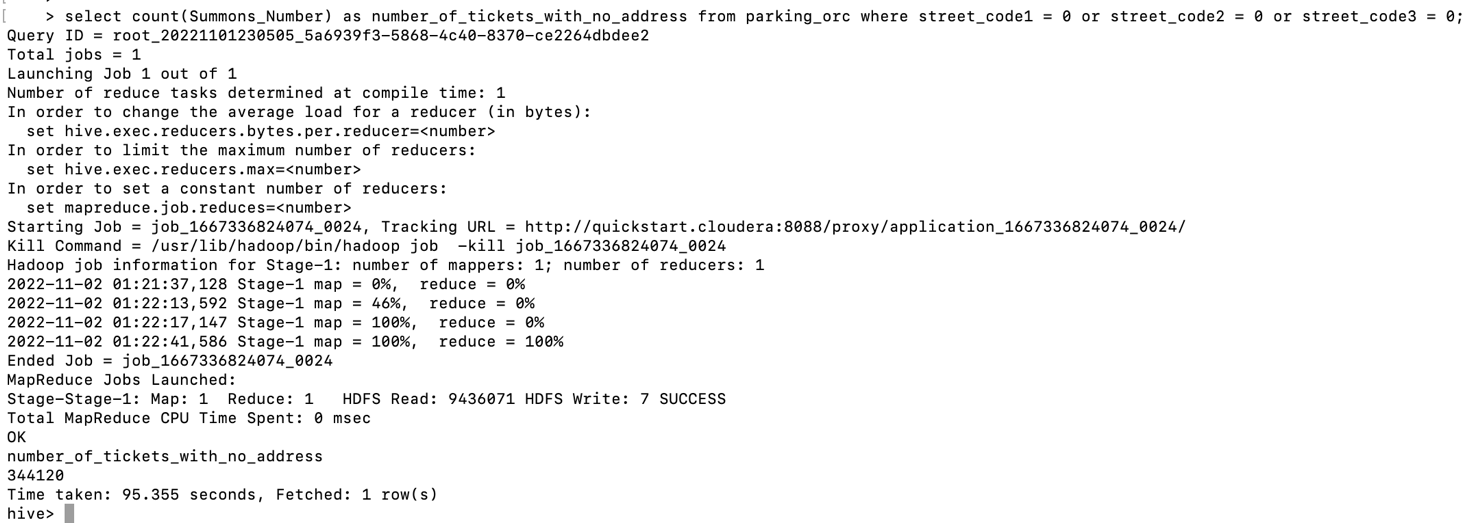


3.) Some parking tickets don’t have addresses on them, which is cause for concern. Find out how many such tickets there are(i.e. tickets where either "Street Code 1" or "Street Code 2" or "Street Code 3" is empty )

select Summons\_Number from parking\_orc where concat(street\_code1,street\_code2,street\_code3) is null;



select count(Summons\_Number) as number\_of\_tickets\_with\_no\_address from parking\_orc where street\_code1 = 0 or street\_code2 = 0 or street\_code3 = 0;



Part-II: Aggregation tasks

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

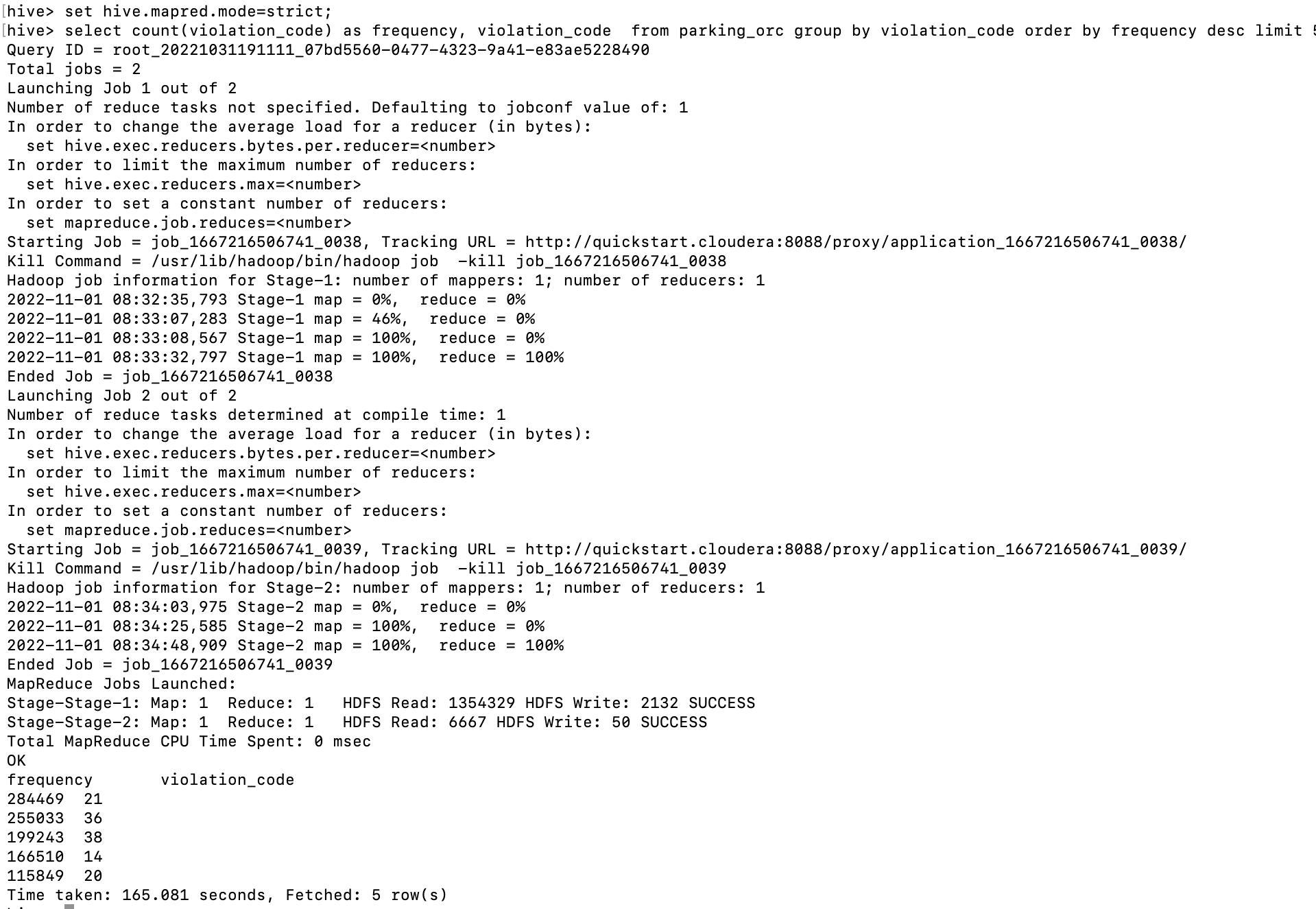
#setting mode to strict

set hive.mapred.mode=strict;

Why strict mode ?

The reason is that in order to impose total order of all results, there has to be one reducer to sort the final output. If the number of rows in the output is too large, the single reducer could take a very long time to finish.

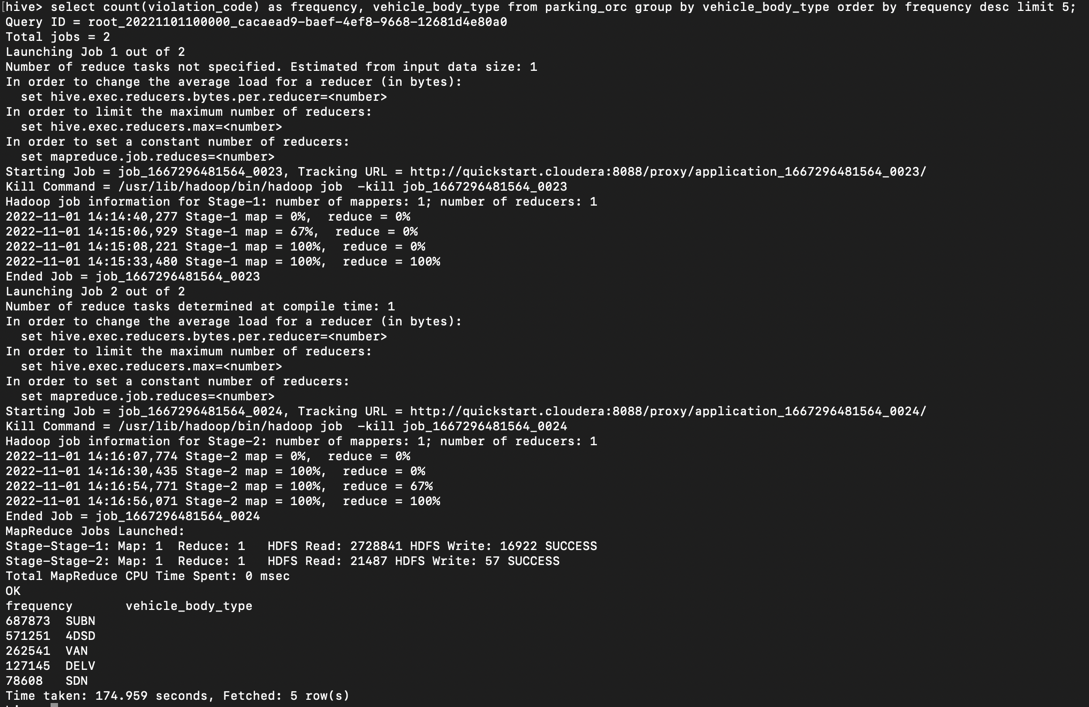
select count(violation\_code) as frequency, violation\_code from parking\_orc group by violation\_code order by frequency desc limit 5;



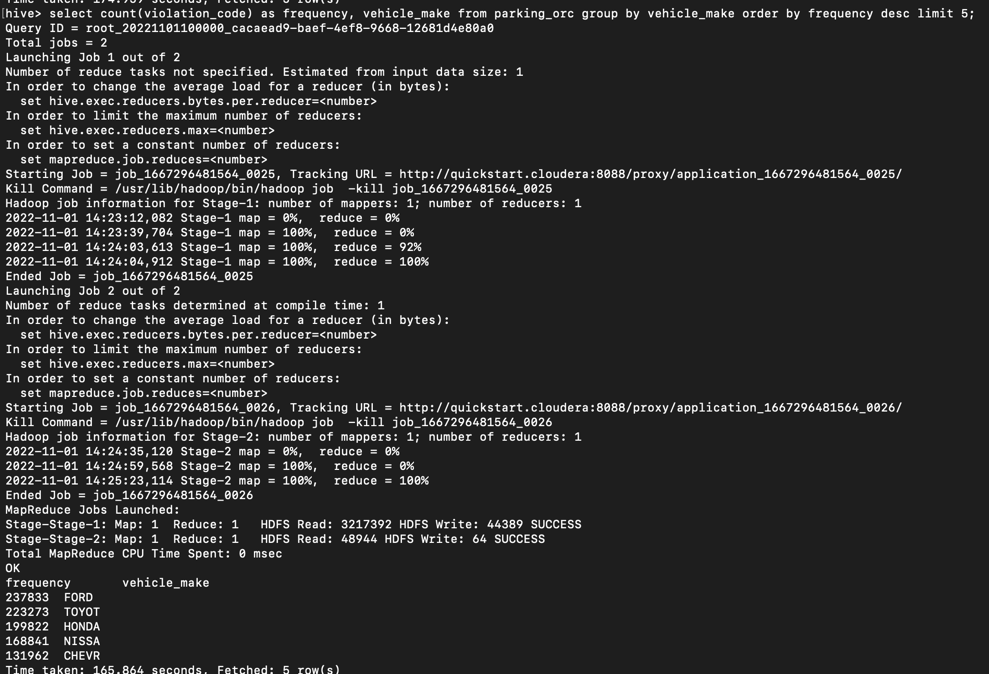
**2.) How often does each vehicle body type get a parking ticket? How about the vehicle make? (find the top 5 for both)**

set hive.mapred.mode=strict;

select count(violation\_code) as frequency, vehicle\_body\_type from parking\_orc group by vehicle\_body\_type order by frequency desc limit 5;



select vehicle\_make , count(violation\_code) as frequency from parking\_orc group by vehicle\_make order by frequency desc limit 5;

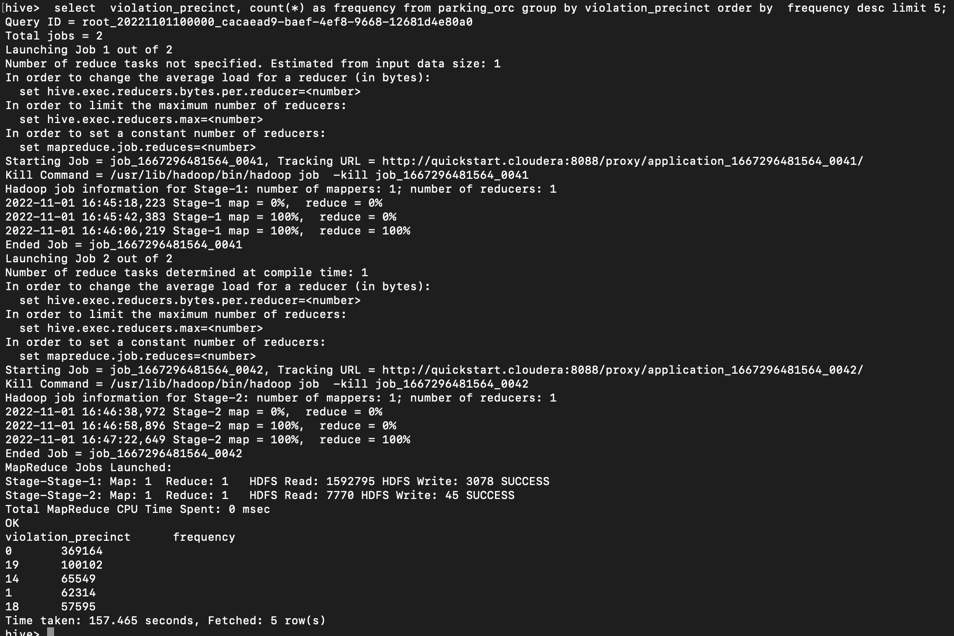


**3.) A precinct is a police station that has a certain zone of the city under its command. Find the (5 highest) frequencies of:**

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

set hive.mapred.mode=strict;

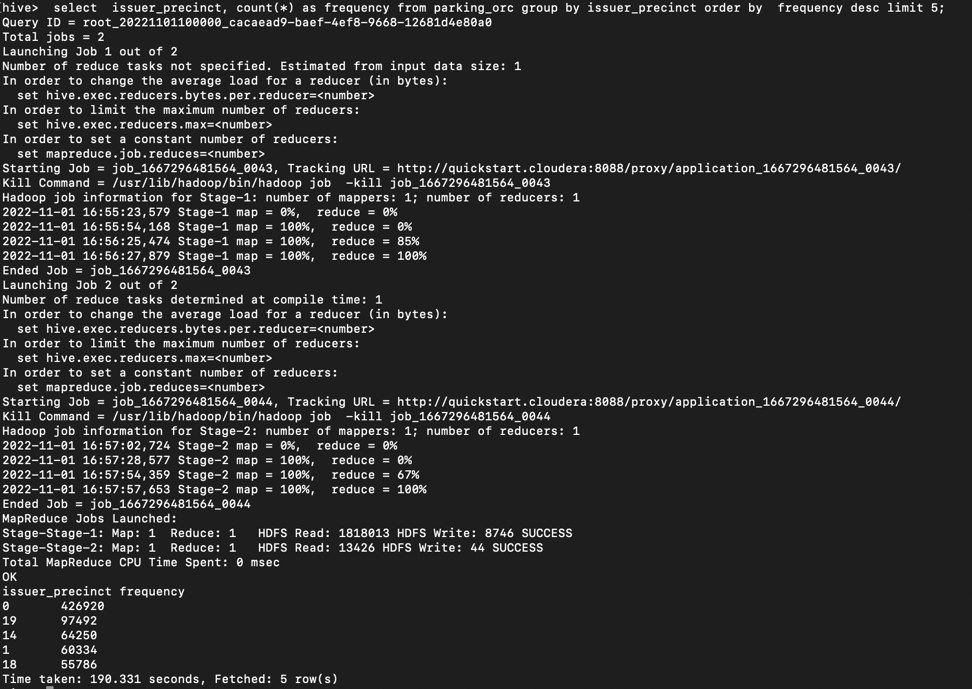
select violation\_precinct, count(\*) frequency from parking\_orc group by violation\_precinct order by frequency desc;



We believe that “violation\_precinct=0” is something null/empty cell

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

select issuer\_precinct, count(\*) as frequency from parking\_orc group by issuer\_precinct order by frequency desc limit 5;



**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?**

#from the above results we see that 19,14,1 are top most precinct zones

#create bucketing

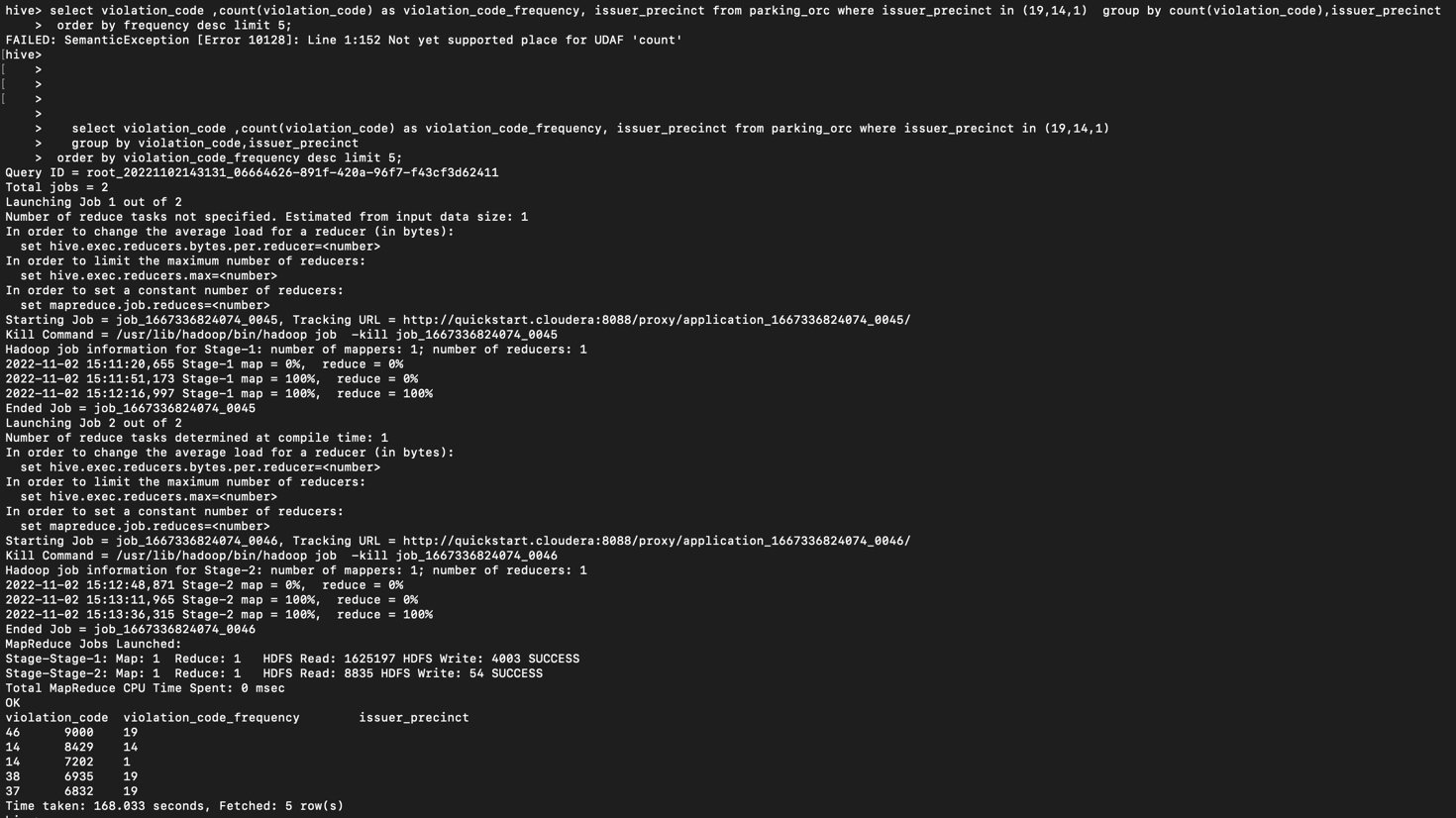
create table issuer\_precinct\_bucket ( violation\_code int , issuer\_precinct int) clustered by (issuer\_precinct) sorted by (issuer\_precinct) into 3 buckets stored as ORC;

#insert data into bucketed table

insert overwrite table issuer\_precinct\_bucket select violation\_code,issuer\_precinct from parking\_orc;

select violation\_code ,count(violation\_code)as violation\_code\_frequency, issuer\_precinct from parking\_orc where issuer\_precinct in (19,14,1)

group by violation\_code,issuer\_precinct order by violation\_code\_frequency desc limit 5;



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.

**#createing table for time related data**

create table time\_violations(summons\_number string, violation\_code int, violation\_time string) stored as ORC;

**#inserting data with from parking\_orc # time data is converted into 24 hours**

insert overwrite table time\_violations select summons\_number , violation\_code, from\_unixtime(unix\_timestamp(regexp\_replace(violation\_time,'\P','\\PM'), 'hhmmaa'), 'HH:mm aa') from parking\_orc where violation\_time like '%P' ;

insert into table time\_violations select summons\_number , violation\_code,from\_unixtime(unix\_timestamp(violation\_time, 'hhmm'), 'HH:mm aa') from parking\_orc where violation\_time like '%A' ;

**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**

**#dividing data in 6 bins #creating view and saving data in view**

Create view time\_data as select violation\_code, case

when substring(violation\_time,1,2) in ('00','01','02','03') and upper(substring(violation\_time,-2))=("AM") then 'bin\_1'

when substring(violation\_time,1,2) in ('04','05','06','07') and upper(substring(violation\_time,-2))=("AM") then 'bin\_2'

when substring(violation\_time,1,2) in ('08','09','10','11') and upper(substring(violation\_time,-2))=("AM") then 'bin\_3'

when substring(violation\_time,1,2) in ('12','13','14','15') and upper(substring(violation\_time,-2))=("PM") then 'bin\_4'

when substring(violation\_time,1,2) in ('16','17','18','19') and upper(substring(violation\_time,-2))=("PM") then 'bin\_5'

when substring(violation\_time,1,2) in ('20','21','22','23') and upper(substring(violation\_time,-2))=("PM") then 'bin\_6'

else null end as violation\_bins

from time\_violations where violation\_time is not null or (substring(violation\_time,1,2) in ('00','01','02','03', '04','05','06','07', '08','09','10','11', '12','13','14','15', '16','17','18','19', '20','21','22','23' ) and upper(substring(violation\_time,-2)) in ('PM', 'AM'));

**#creating partitioned table**

create table time\_partition (violation\_code int) partitioned by (violation\_bin string) stored as ORC;

**#loading data to partitioned table from view**

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table time\_partition partition (violation\_bin) select violation\_code, violation\_bins from time\_data;

**# 3 most commonly occurring violations in each bin**

select violation\_code, count(violation\_code) as frequency from time\_partition where violation\_bin = 'bin\_1' group by violation\_code distribute by frequency sort by frequency desc limit 3;

select violation\_code, count(violation\_code) as frequency from time\_partition where violation\_bin = 'bin\_2' group by violation\_code distribute by frequency sort by frequency desc limit 3;

select violation\_code, count(violation\_code) as frequency from time\_partition where violation\_bin = 'bin\_3' group by violation\_code distribute by frequency sort by frequency desc limit 3;

select violation\_code, count(violation\_code) as frequency from time\_partition where violation\_bin = 'bin\_4' group by violation\_code distribute by frequency sort by frequency desc limit 3;

select violation\_code, count(violation\_code) as frequency from time\_partition where violation\_bin = 'bin\_5' group by violation\_code distribute by frequency sort by frequency desc limit 3;

select violation\_code, count(violation\_code) as frequency from time\_partition where violation\_bin = 'bin\_6' group by violation\_code distribute by frequency sort by frequency 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, count(violation\_code) as a from time\_date group by violation\_code distribute by a sort by a desc limit 5;

select violation\_bins, count(violation\_bins) from time\_data where violation\_code in ('21','36','38') group by violation\_bins ;

**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))**

**#creating view and saving data in view**

create view seasons\_view as Select violation\_code, issue\_date, case

when substring(issue\_Date,1,2) in ('12','01','02') then 'winter'

when substring(issue\_Date,1,2) in ('03','04','05') then 'spring'

when substring(issue\_Date,1,2) in ('06','07','08') then 'summer'

when substring(issue\_Date,1,2) in ('09','10','11') then 'fall'

else 'not\_defined' end as seasons

from parking\_orc where issue\_date is not null or ((length(issue\_date)=10 and substring(issue\_date,1,2) in ('12','01','02','03','04','05','06','07','08','09','10','11'))) ;

#**finding the frequencies of tickets for each season**

select count(violation\_code) as ticket\_count, seasons

from seasons\_view

group by seasons order by ticket\_count desc

**#creating partition table**

create table season\_partition(violation\_code int, issue\_date string) partitioned by (seasons string) stored as ORC;

**#inserting data from view to partitioned table**

insert overwrite table season\_partition partition (seasons) select \* from seasons\_view;

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

select violation\_code, count(violation\_code) as frequency from season\_partition where seasons = 'winter' group by violation\_code distribute by frequency sort by frequency limit 3;

select violation\_code, count(violation\_code) as frequency from season\_partition where seasons = 'fall'group by violation\_code distribute by frequency sort by frequency limit 3;

select violation\_code, count(violation\_code) as frequency from season\_partition where seasons = 'summer' group by violation\_code distribute by frequency sort by frequency limit 3;

select violation\_code, count(violation\_code) as frequency from season\_partition where seasons = 'spring' group by violation\_code distribute by frequency sort by frequency limit 3;

Note: Please ensure you make necessary optimizations to your queries like selecting the appropriate table format, using partitioned/bucketed tables. Marks will be awarded for keeping the performance also in mind.