HIVE ASSIGNMENT

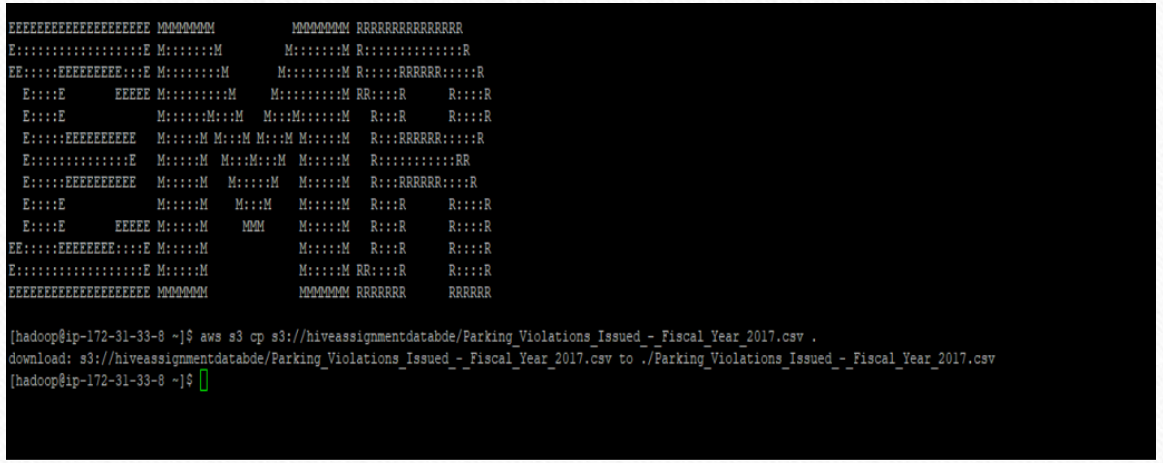
BY LOGESHWARAN V

Problem Statement

- New York City is a thriving metropolis, and like most other cities of its size, one of the biggest problems faced by its residents is the lack of parking space. The classic combination of a huge number of cars and 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 the NYC Open Data portal. Your job is to try and perform some analysis on this data in order to answer the questions that follow.

Copy Data to Hadoop

aws s3 cp s3://hiveassignmentdatabde/Parking_Violations_Issued_- _Fiscal_Year_2017.csv .



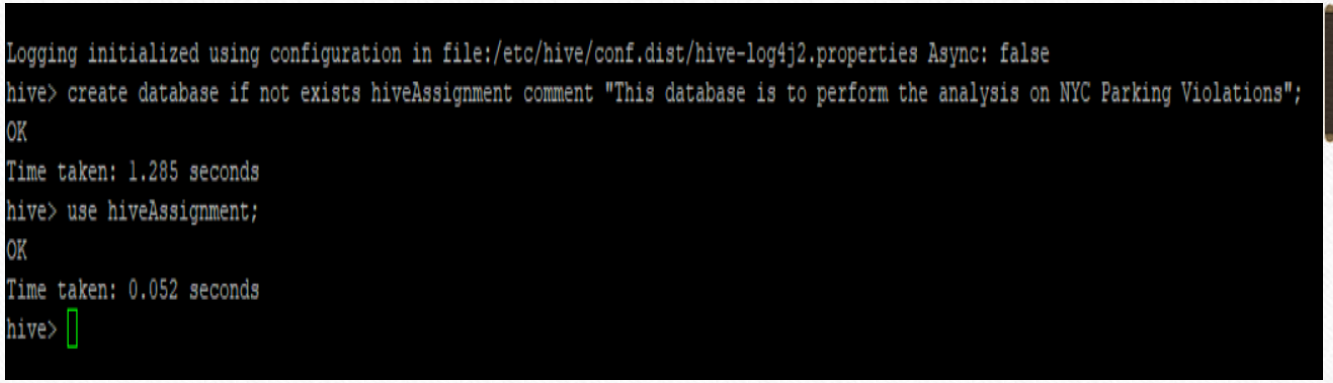
Creating and using the database

Create database:

create database if not exists hiveAssignment comment "This database is to perform the analysis on NYC Parking Violations";

Use database:

use hiveAssignment;



Creating the table

create table if not exists parkingViolations (SummonsNumber int, PlateID string,RegistrationState string,PlateType string,IssueDate string,ViolationCode int,VehicleBodyType string, VehicleMake string,IssuingAgency string,StreetCode1 int,StreetCode2 int,StreetCode3 int,VehicleExpirationDate int, ViolationLocation string,ViolationPrecinct int, IssuerPrecinct int,IssuerCode int,IssuerCommand string,IssuerSquad string,ViolationTime string,TimeFirstObserved string,ViolationCounty string,ViolationInFrontOfOrOpposite string,HouseNumber string,StreetName string,IntersectingStreet string,DateFirstObserved int,LawSection int,SubDivision string,ViolationLegalCode string,DaysParkingInEffect string,FromHoursInEffect string,ToHoursInEffect string,VehicleColor string,UnregisteredVehicle string,VehicleYear int,MeterNumber string,FeetFromCurb int,ViolationPostCode string, ViolationDescription string,NoStandingorStoppingViolation string,HydrantViolation string,DoubleParkingViolation string) row format delimited fields terminated by ',' lines terminated by '\n' stored as textfile TBLPROPERTIES ("skip.header.line.count"="1");

```
hive> create table if not exists parkingViolations (
    > SummonsNumber int, PlateID string,RegistrationState string,PlateType string,IssueDate string,ViolationCode int,VehicleBodyType string, VehicleMake string,IssuingAgency string,StreetCode1 int,StreetCode2 int,StreetCode3 int,VehicleE
xpirationDate int, ViolationLocation string,ViolationPrecinct int, IssuerPrecinct int,IssuerCode int,IssuerCommand string,IssuerSquad string,ViolationTime string,TimeFirstObserved string,ViolationCounty string,ViolationInFrontOfOrOpposit
e string,HouseNumber string,StreetName string,IntersectingStreet string,DateFirstObserved int,LawSection int,SubDivision string,ViolationLegalCode string,DaysParkingInEffect string,FromHoursInEffect string,ToHoursInEffect string,VehicleC
olor string,UnregisteredVehicle string,VehicleYear int,MeterNumber string,FeetFromCurb int,ViolationPostCode string, ViolationDescription string,NoStandingorStoppingViolation string,HydrantViolation string,DoubleParkingViolation string )
 row format delimited fields terminated by ',' lines terminated by '\n' stored as textfile TBLPROPERTIES ("skip.header.line.count"="1");
OK
Time taken: 0.59 seconds
hive>
```

Loading the data into table

Load data to table:

load data local inpath '/home/hadoop/Parking_Violations_Issued_- _Fiscal_Year_2017.csv' into table parkingViolations ;

```
hive> load data local inpath '/home/hadoop/Parking_Violations_Issued_-_Fiscal_Year_2017.csv' into table parkingViolations ;
Loading data to table hiveassignment.parkingviolations
OK
Time taken: 4.253 seconds
```

Enabling Dynamic Partitioning

set hive.exec.dynamic.partition=true;

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

```
Time taken: 0.141 seconds
hive> set hive.exec.dynamic.partition=true;
hive> set hive.exec.dynamic.partition.mode= nonstring;
```

Part -1 Examine the data:

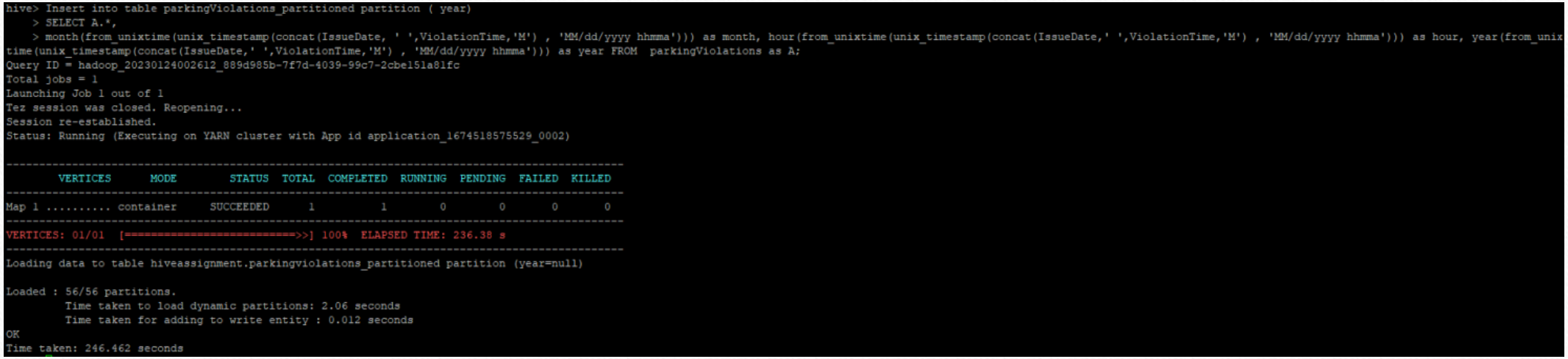
Create a table with year as partition:

create table if not exists parkingViolations_partitioned (SummonsNumber int, PlateID string,RegistrationState string,PlateType string,IssueDate string,ViolationCode int,VehicleBodyType string,VehicleMake string,IssuingAgency string,StreetCode1 int,StreetCode2 int,StreetCode3 int,VehicleExpirationDate int, ViolationLocation string,ViolationPrecinct int,IssuerPrecinct int,IssuerCode int,IssuerCommand string,IssuerSquad string,ViolationTime string,TimeFirstObserved string,ViolationCounty string,ViolationInFrontOfOrOpposite string,HouseNumber string,StreetName string,IntersectingStreet string,DateFirstObserved int,LawSection int,SubDivision string,ViolationLegalCode string,DaysParkingInEffect string, FromHoursInEffect string,ToHoursInEffect string,VehicleColor string,UnregisteredVehicle string,VehicleYear int,MeterNumber string,FeetFromCurb int,ViolationPostCode string, ViolationDescription string,NoStandingorStoppingViolation string,HydrantViolation string,DoubleParkingViolation string,month int, hour int) partitioned by (year int) row format delimited fields terminated by ',' lines terminated by '\n' stored as textfile TBLPROPERTIES ("skip.header.line.count"="1");

```
hive> create table if not exists parkingViolations_partitioned (
    > SummonsNumber int, PlateID string,RegistrationState string,PlateType string,IssueDate string,ViolationCode int,VehicleBodyType string,VehicleMake string,IssuingAgency string,StreetCode1 int,StreetCode2 int,StreetCode3 int,VehicleEx
pirationDate int, ViolationLocation string,ViolationPrecinct int,IssuerPrecinct int,IssuerCode int,IssuerCommand string,IssuerSquad string,ViolationTime string,TimeFirstObserved string,ViolationCounty string,ViolationInFrontOfOrOpposite
string,HouseNumber string,StreetName string,IntersectingStreet string,DateFirstObserved int,LawSection int,SubDivision string,ViolationLegalCode string,DaysParkingInEffect string, FromHoursInEffect string,ToHoursInEffect string,VehicleCo
lor string,UnregisteredVehicle string,VehicleYear int,MeterNumber string,FeetFromCurb int,ViolationPostCode string, ViolationDescription string,NoStandingorStoppingViolation string,HydrantViolation string,DoubleParkingViolation string,mo
nth int, hour int ) partitioned  by ( year int ) row format delimited fields terminated by ',' lines terminated by '\n' stored as textfile  TBLPROPERTIES ("skip.header.line.count"="1");
OK
Time taken: 0.173 seconds
```

Load data into table with year as partition:

Insert into table parkingViolations_partitioned partition (year) SELECT A.*, month(from_unixtime(unix_timestamp(concat(IssueDate, ' ',ViolationTime,'M') , 'MM/dd/yyyy hhmma'))) as month, hour(from_unixtime(unix_timestamp(concat(IssueDate,' ',ViolationTime,'M') , 'MM/dd/yyyy hhmma'))) as hour, year(from_unixtime(unix_timestamp(concat(IssueDate,' ',ViolationTime,'M') , 'MM/dd/yyyy hhmma'))) as year FROM parkingViolations as A;



Q1. Find the total number of tickets for the year.

Query :

Select count(*) as total_num_of_tickets from parkingViolations_partitioned where year = 2017;

Ans:

5431842

```
hive> Select count(*) as total_num_of_tickets from parkingViolations_partitioned where  year = 2017;
OK
total_num_of_tickets
5431842
Time taken: 0.116 seconds, Fetched: 1 row(s)
```

Q2. Find out the total number of states to which the cars with tickets belong. The count of states is mandatory here; providing the exact list of states is optional.

Query :

Select count(distinct RegistrationState) as total_num_of_states from parkingViolations_partitioned where year = 2017;

Answer:

65

```
hive> Select count(distinct RegistrationState) as total_num_of_states from parkingViolations_partitioned where  year = 2017;
Query ID = hadoop_20230124003747_ecld6e05-8b1f-4b54-b7b8-5ecc06e111b0
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1674518575529_0003)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 10.84 s
--------------------------------------------------------------------------------
OK
total_num_of_states
65
Time taken: 11.358 seconds, Fetched: 1 row(s)
```

Q3. Some parking tickets don't have addresses on them, which is a cause for concern. Find out the number of such tickets which have no addresses. (i.e. tickets where one of the Street Codes, i.e. "Street Code 1" or "Street Code 2" or "Street Code 3" is empty)

Query:

Select count(*) as num_of_tickets from parkingViolations_partitioned where year = 2017 and (StreetCode1 is null or StreetCode2 is null or StreetCode3 is null);

Answer:

For year 2017, there are no tickets which don't have address.

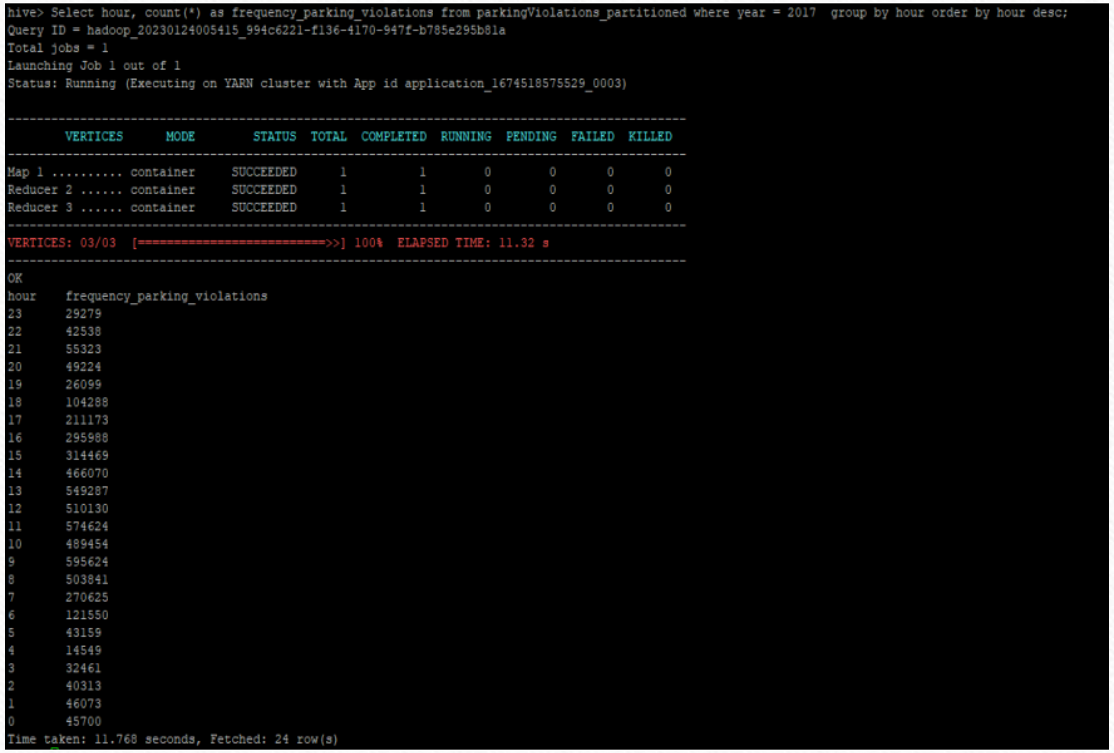
```
hive> Select count(*)  as num_of_tickets from parkingViolations_partitioned where  year = 2017 and
    > (StreetCode1 is null or  StreetCode2 is null or StreetCode3 is null);
Query ID = hadoop_20230124004500_f188aa96-880a-4771-af0f-d31872966be3
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1674518575529_0003)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 12.58 s
--------------------------------------------------------------------------------
OK
num_of_tickets
0
Time taken: 13.126 seconds, Fetched: 1 row(s)
hive>
```

Part-II: Aggregation tasks

Q1. Find out the frequency 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.

Query:

```
Select hour, count(*) as
frequency_parking_violations from
parkingViolations_partitioned where
year = 2017 group by hour order by
hour desc;
```



```
hive> Select hour, count(*) as frequency_parking_violations from parkingViolations_partitioned where year = 2017  group by hour order by hour desc;
Query ID = hadoop_20230124005415_994c6221-f136-4170-947f-b785e295b81a
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1674518575529_0003)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   container     SUCCEEDED      1          1        0        0       0       0
Reducer 3 ......   container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 11.32 s
--------------------------------------------------------------------------------
OK
hour    frequency_parking_violations
23      29279
22      42538
21      55323
20      49224
19      26099
18      104288
17      211173
16      295988
15      314469
14      466070
13      549287
12      510130
11      574624
10      489454
9       595624
8       503841
7       270625
6       121550
5       43159
4       14549
3       32461
2       40313
1       46073
0       45700
Time taken: 11.768 seconds, Fetched: 24 row(s)
```

Q2. Divide 24 hours into six 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.

Query:

```
Select hour_bins, ViolationCode, frequency as num_violations,rank from (Select hour_bins,ViolationCode,frequency, dense_rank() over (partition by hour_bins order by frequency desc) as rank from ( Select hour_bins , ViolationCode, count(*) as frequency from ( Select case when hour in (0,1,2,3) then 'Late Hours' when hour in (4,5,6,7) then 'Early Morning' when hour in (8,9,10,11) then 'Morning' when hour in (12,13,14,15) then 'Afternoon' when hour in (16,17,18,19) then 'Evening' when hour in (20,21,22,23) then 'Night' End as hour_bins,ViolationCode from parkingViolations_partitioned where year = 2017 ) as binned_data group by hour_bins,ViolationCode ) as frequency_data) as rank_data where rank <=3 order by hour_bins;
```

```
hive> Select hour_bins, ViolationCode, frequency as num_violations,rank from
    > (Select hour_bins,ViolationCode,frequency, dense_rank() over (partition by hour_bins
    > order by frequency desc) as rank  from (
    > Select hour_bins , ViolationCode, count(*) as frequency from (
    > Select  case
    > when hour in (0,1,2,3) then 'Late Hours'
    > when hour in (4,5,6,7) then 'Early Morning'
    > when hour in (8,9,10,11) then 'Morning'
    > when hour in (12,13,14,15) then 'Afternoon'
    > when hour in (16,17,18,19) then 'Evening'
    > when hour in (20,21,22,23) then 'Night'
    > End as hour_bins,ViolationCode  from parkingViolations_partitioned  where year = 2017 )
    > as binned_data group by hour_bins,ViolationCode ) as frequency_data) as rank_data where
    >  rank <=3 order by hour_bins;
Query ID = hadoop_20230124011755_a07ba391-0024-4eb5-8da8-f498e8227439
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1674518575529_0005)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      6          6        0        0       0       0
Reducer 4 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 04/04  [==========================>>] 100%  ELAPSED TIME: 14.46 s
--------------------------------------------------------------------------------
OK
hour_bins       violationcode   num_violations  rank
Afternoon       36      286284  1
Afternoon       38      240721  2
Afternoon       37      167026  3
Early Morning   14      74113   1
Early Morning   40      60653   2
Early Morning   21      57897   3
Evening 38      102856  1
Evening 14      75902   2
Evening 37      70345   3
Late Hours      21      36966   1
Late Hours      40      25868   2
Late Hours      78      15529   3
Morning 21      598053  1
Morning 36      348165  2
Morning 38      176570  3
Night   7       26293   1
Night   40      22338   2
Night   14      21045   3
Time taken: 14.998 seconds, Fetched: 18 row(s)
hive>
```

Q3. 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).

Query:

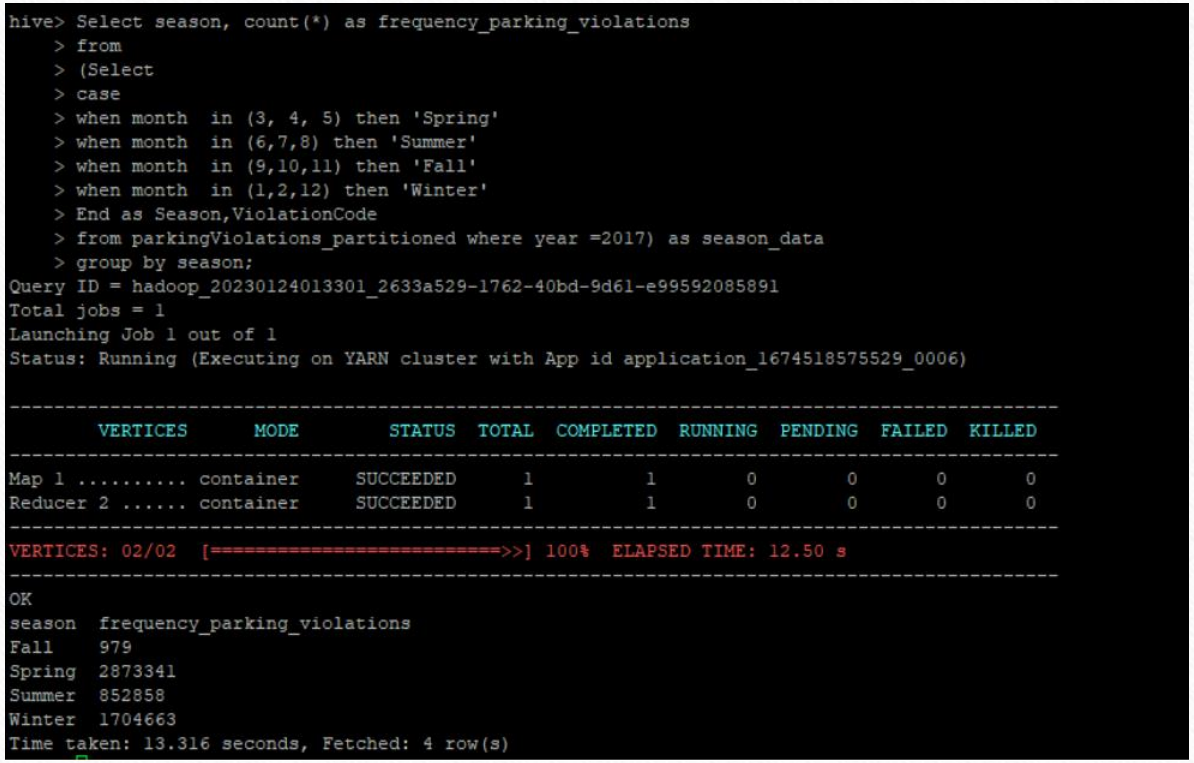
```
Select ViolationCode,hour_bins, frequency as num_violations,rank from (Select ViolationCode,hour_bins, frequency, dense_rank() over (partition by ViolationCode order by frequency desc) as rank from ( Select ViolationCode,hour_bins, count(*) as frequency from ( Select case when hour in (0,1,2,3) then 'Late Hours' when hour in (4,5,6,7) then 'Early Morning' when hour in (8,9,10,11) then 'Morning' when hour in (12,13,14,15) then 'Afternoon' when hour in (16,17,18,19) then 'Evening' when hour in (20,21,22,23) then 'Night' End as hour_bins,a.ViolationCode as ViolationCode from (Select hour,ViolationCode from parkingViolations_partitioned where year =2017 ) as a inner join (Select ViolationCode,count(*) as num_of_violations from parkingViolations_partitioned where year =2017 group by ViolationCode order by num_of_violations desc limit 3) b on a.ViolationCode = b.ViolationCode ) as data group by ViolationCode ,hour_bins) as frequency_data ) as rank_data where rank =1;
```

```
hive> Select  ViolationCode,hour_bins, frequency as num_violations,rank from
    > (Select ViolationCode,hour_bins, frequency, dense_rank() over (partition by ViolationCode order by frequency desc) as rank
    > from ( Select ViolationCode,hour_bins , count(*) as frequency from (
    > Select case
    > when hour in (0,1,2,3) then 'Late Hours'
    > when hour in (4,5,6,7) then 'Early Morning'
    > when hour in (8,9,10,11) then 'Morning'
    > when hour in (12,13,14,15) then 'Afternoon'
    > when hour in (16,17,18,19) then 'Evening'
    > when hour in (20,21,22,23) then 'Night'
    > End as hour_bins,a.ViolationCode as  ViolationCode from
    > (Select hour,ViolationCode from parkingViolations_partitioned where year =2017 ) as a  inner join
    > (Select ViolationCode,count(*) as num_of_violations from parkingViolations_partitioned where year =2017
    > group by ViolationCode order by num_of_violations desc limit 3) b on a.ViolationCode = b.ViolationCode ) as data group by ViolationCode ,hour_bins) as frequency_data ) as rank_data where rank =1;
Query ID = hadoop_20230124012802_7c9e6c18-3171-4d25-be47-9df9b751e901
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1674518575529_0006)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Map 4 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      6          6        0        0       0       0
Reducer 5 ...... container     SUCCEEDED      1          1        0        0       0       0
Reducer 6 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 06/06  [==========================>>] 100%  ELAPSED TIME: 22.38 s
--------------------------------------------------------------------------------
OK
violationcode   hour_bins       num_violations  rank
36      Morning 348165  1
38      Afternoon       240721  1
21      Morning 598053  1
Time taken: 23.342 seconds, Fetched: 3 row(s)
hive>
```

Q4.1: First, divide the year into seasons, and find the frequencies of tickets for each season.

Query:

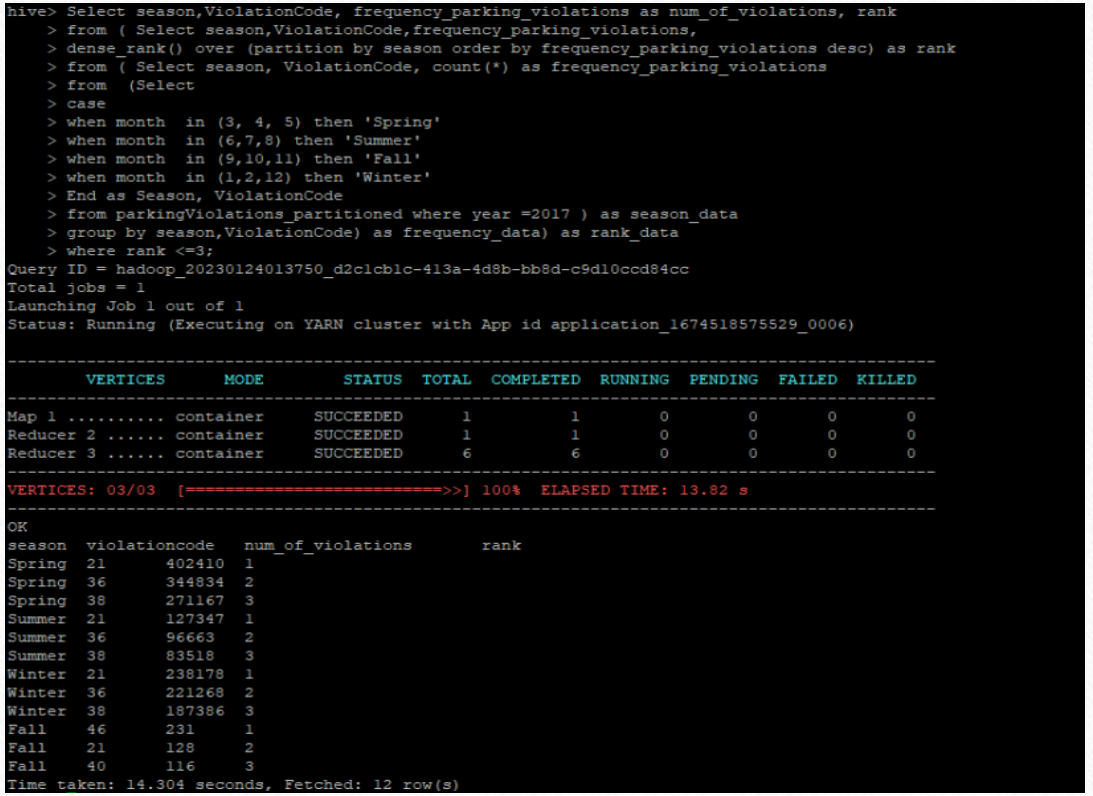
```
Select season, count(*) as frequency_parking_violations from (Select case when month in (3, 4, 5) then 'Spring' when month in (6,7,8) then 'Summer' when month in (9,10,11) then 'Fall' when month in (1,2,12) then 'Winter' End as Season,ViolationCode from parkingViolations_partitioned where year =2017) as season_data group by season;
```



Q4.2: find the 3 most common violations for each of these seasons.

Query:

```
Select season,ViolationCode,
frequency_parking_violations as num_of_violations,
rank from ( Select
season,ViolationCode,frequency_parking_violations,
dense_rank() over (partition by season order by
frequency_parking_violations desc) as rank from (
Select season, ViolationCode, count(*) as
frequency_parking_violations from (Select case when
month in (3, 4, 5) then 'Spring' when month in (6,7,8)
then 'Summer' when month in (9,10,11) then 'Fall'
when month in (1,2,12) then 'Winter' End as Season,
ViolationCode from parkingViolations_partitioned
where year =2017 ) as season_data group by
season,ViolationCode) as frequency_data) as rank_data
where rank <=3;
```



```
hive> Select season,ViolationCode, frequency_parking_violations as num_of_violations, rank
    > from ( Select season,ViolationCode,frequency_parking_violations,
    > dense_rank() over (partition by season order by frequency_parking_violations desc) as rank
    > from ( Select season, ViolationCode, count(*) as frequency_parking_violations
    > from  (Select
    > case
    > when month  in (3, 4, 5) then 'Spring'
    > when month  in (6,7,8) then 'Summer'
    > when month  in (9,10,11) then 'Fall'
    > when month  in (1,2,12) then 'Winter'
    > End as Season, ViolationCode
    > from parkingViolations_partitioned where year =2017 ) as season_data
    > group by season,ViolationCode) as frequency_data) as rank_data
    > where rank <=3;
Query ID = hadoop_20230124013750_d2c1cb1c-413a-4d8b-bb8d-c9d10ccd84cc
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1674518575529_0006)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      6          6        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 13.82 s
----------------------------------------------------------------------------------------------
OK
season  violationcode   num_of_violations       rank
Spring  21      402410  1
Spring  36      344834  2
Spring  38      271167  3
Summer  21      127347  1
Summer  36      96663   2
Summer  38      83518   3
Winter  21      238178  1
Winter  36      221268  2
Winter  38      187386  3
Fall    46      231     1
Fall    21      128     2
Fall    40      116     3
Time taken: 14.304 seconds, Fetched: 12 row(s)
```

Thank You