```
Introduction:

The assignment is meant for me to apply learnings of the module on Hive on a real-life dataset.
The objective is to familiarize me with necessary optimizations for my queries like selecting the appropriate table format,
using partitioned/bucketed tables and moreover acquainting me with the different functions of HIVE etc.

To be able to execute these queries , please follow the following steps:

Step 1. Log into AWS Console
Step 2. Go to S3 and create a bucket named - dee-hive-test (Steps for bucket creation are added in reference document:Creating S3 bucket)
Step 3. Start the EC2 instance
Step 4. Once the instance is ON, log into EC2 terminal using Putty. For this you will require - Hostname or IP address,
        which is the Public IP of your EC2 instance which has CDH installation. After logging in as 'ec2-user', switch to root user -
        sudo -i command
Step 5. Now, the first step is to copy the data for analysis, provided by Upgrad team, into our own S3 bucket i.e.  dee-hive-test
Step 6: Execute following command on EC2 terminal:
        aws s3 cp s3://hiveassignmentdatabde/Parking_Violations_Issued_-_Fiscal_Year_2017.csv s3://dee-hive-test
Step 7. Once this data is successfully copied , let us now start creating required database and tables in Hive using HUE.
Step 8. Hit the URL : <Public IP of Ec2 instance>:7180 , this will take you to Cloudera Manager login page.
Step 9. Login using credentials provided. Ensure that you have completed tuning of HIVE and Hue in Cloudera Manager by
        previously following documents and sessions conducted for the same.
Step 10. Launch the Hue Editor. And now work on the queries one by one.

#### I would like my tables to be placed and maintained in 1 database on an explicit location in S3 bucket , hence, starting with create database query:

create database IF not exists NYCParking location 's3a://dee-hive-test/hiveAssignment/' ;


#### Now, creating a table from the data. The below statement when executed will create the table.
#### For creating the table with different fields and data types , I have followed the Data Dictionary
#### present on: https://data.cityofnewyork.us/City-Government/Parking-Violations-Issued-Fiscal-Year-2017/2bnn-yakx

create external IF NOT EXISTS table NYCParking.parking_ext(`SummonsNumber` bigint, `PlateID` string, `RegistrationState` string, `PlateType` string,`IssueDate` string,`ViolationCode` int,
`VehicleBodyType` string,`VehicleMake` string,`IssuingAgency` string,`StreetCode1` bigint, `StreetCode2` bigint, `StreetCode3` bigint, `VehicleExpirationDate` bigint,
`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 ','
 location 's3a://dee-hive-test/hiveAssignment/'
 tblproperties ("skip.header.line.count"="1");


##### Checking if table is successfully created in the database.

```

```
SHOW TABLES in nycparking;

##### Checking if the table created contains any data or not ?

select * from nycparking.parking_ext;

###### Now populating the table created with the input data set that I had copied
into my S3 bucket:

load data inpath
'S3A://dee-hive-test/Parking_Violations_Issued_-_Fiscal_Year_2017.csv' overwrite
into table nycparking.parking_ext;

###### Checking how the data looks like in the table, if all the values are
correctly loaded into the table in the expected formats.

select * from nycparking.parking_ext limit 10;

##### Since we are going to query only on year 2017 data, I would be using
INSTR(IssueDate , '2017'). The method instr(string col, string substr)
##### which returns the position of the first occurence of substr in col value.
##### This method returns an int > 0 if the substr is found in the col value.
##### I would be using this method in my queries where I need to filter records of
only year - 2017
##### Below I am only testing if this string works or not.

select plateid , issuedate from nycparking.parking_ext where INSTR(IssueDate ,
'2017') > 0 limit 10;

#######################################################################################
######################################################################

Part-I: Examine the data

#######################################################################################
######################################################################

1. Find the total number of tickets for the year.
## Here , I am using count(Distinct) on all tickets (summonsnumber) to get the total
record count in dataset, after loading into table.
## My intention is to filter records only from year 2017.

Query:

select count(DISTINCT summonsnumber) as tickets_count from nycparking.parking_ext
where INSTR(IssueDate , '2017') > 0;

Output:

tickets_count
5431903

2. Find out how many unique states the cars which got parking tickets came from
## Here , I am using count(Distinct) on all states (violationcounty) to get the
count of those unique states whose cars got parking tickets in year 2017

Query:

select count(DISTINCT violationcounty) as unique_states from nycparking.parking_ext
where INSTR(IssueDate , '2017') > 0;

Output:

unique_states
12

3. Finding tickets from the dataset which do not have addresses on them
## In this case, if streetcode1, streetcode2 or streetcode3 is BLANK or may be has
'0' as the value which may be a default value for these address fields
## we would want to get the count of such parking tickets , also from year 2017 only.

Query:

```

```
select count(DISTINCT summonsnumber) as withoutaddress_count from nycparking.parking_ext
where (streetcode1 is null OR streetcode1 = '0' OR streetcode2 is null OR streetcode2 = '0' OR streetcode3 is null OR streetcode3 = '0')
AND INSTR(IssueDate , '2017') > 0 ;

Output:

withoutaddress_count
1816816

### Below is another way of writing the same query where I first filter records from year 2017 only and then find the count of
### such without address tickets from the set


select count(DISTINCT summonsnumber) as withoutaddress_count from
(select * from nycparking.parking_ext where INSTR(IssueDate , '2017') > 0) a
where (streetcode1 is null OR streetcode1 = '0' OR streetcode2 is null OR streetcode2 = '0' OR streetcode3 is null OR streetcode3 = '0');


withoutaddress_count
1816816


##############################################################################################################################################
Part-II: Aggregation tasks
##############################################################################################################################################

1. Finding frequency of violation codes - find the top 5

### Here, I get the occurrence (count) of all violationcodes which have taken place in year 2017.
### I then Group these by individual violationcodes and sort them in descending order of their occurrence (count)
### I then pick the top 5 [when my result set is in descending order of count, the LIMIT function would return me TOP n rows]

Query:

select violationcode , count(*) as violationcode_frequency from nycparking.parking_ext where INSTR(IssueDate , '2017') > 0
group by (violationcode) order by violationcode_frequency DESC limit 5;

OUTPUT:

1. 21 768082
2. 36 662765
3. 38 542079
4. 14 476660
5. 20 319646

374.373 seconds


2. (i) Finding frequency of each vehicle body type getting a parking ticket
### Here, I get the occurrence (count) of all parking violations by different vehicle body types which have taken place in year 2017.
### I then Group these by individual vehiclebodytype and sort them in descending order of their occurrence (count)
### I then pick the top 5 [when my result set is in descending order of count, the LIMIT function would return me TOP n rows]

Query:

select vehiclebodytype , count(*) as vehiclebody_tktcount from nycparking.parking_ext
where INSTR(IssueDate , '2017') > 0 group by (vehiclebodytype) order by vehiclebody_tktcount DESC LIMIT 5;

OUTPUT:
```

```
vehiclebodytype vehiclebody_tktcount
1. SUBN 1883953
2. 4DSD 1547307
3. VAN 724025
4. DELV 358982
5. SDN 194197

375.669 seconds

2. (ii) Finding frequency of each vehicle make getting a parking ticket
### Here, I get the occurrence (count) of all parking violations by different vehicle make which have taken place in year 2017.
### I then Group these by individual vehiclemake and sort them in descending order of their occurrence (count)
### I then pick the top 5 [when my result set is in descending order of count, the LIMIT function would return me TOP n rows]

Query:

select vehiclemake , count(*) as vehiclemake_tktcount from nycparking.parking_ext
where INSTR(IssueDate , '2017') > 0 group by (vehiclemake) order by vehiclemake_tktcount DESC LIMIT 5;

Output:

 vehiclemake vehiclemake_tktcount
1. FORD 636842
2. TOYOT 605290
3. HONDA 538884
4. NISSA 462017
5. CHEVR 356032

3. (1)Finding Violating Precincts with highest frequencies

## Query which includes violationprecinct numbered '0' too in the result.
## This pulls out all violationprecincts where maximum violations occurred in year 2017
## It then groups the data by these violationprecincts and then sorts the data in descending order of their occurrences(count)
## I then pick the top 5 [when my result set is in descending order of count, the LIMIT function would return me TOP n rows]


Query:

select violationprecinct , count(*) as violationPrecinct_tktcount from nycparking.parking_ext
where INSTR(IssueDate , '2017') > 0
group by (violationprecinct) order by violationPrecinct_tktcount DESC LIMIT 5;

Output:

violationprecinct violationprecinct_tktcount
1. 0 925596
2. 19 274443
3. 14 203552
4. 1 174702
5. 18 169131

#Query which does not include violation precinct numbered 0

Query:

select violationprecinct , count(*) as violationPrecinct_tktcount from nycparking.parking_ext
where INSTR(IssueDate , '2017') > 0 AND violationprecinct > 0
group by (violationprecinct) order by violationPrecinct_tktcount DESC LIMIT 5;


3. (2)Finding Issuer Precincts with highest frequencies

## Query which includes Issuerprecinct numbered '0' too in the result.
## This pulls out all issuerprecincts where maximum tickets were issued in year 2017
## It then groups the data by these issuerprecincts and then sorts the data in descending order of their occurrences(count)
## I then pick the top 5 [when my result set is in descending order of count, the LIMIT function would return me TOP n rows]
```

Query:

```
select issuerprecinct , count(*) as issuerPrecinct_tktcount from nycparking.parking_ext
where INSTR(IssueDate , '2017') > 0
group by (issuerprecinct) order by issuerPrecinct_tktcount DESC LIMIT 5;
```

Output:

```
issuerprecinct issuerprecinct_tktcount
1. 0 1078403
2. 19 266959
3. 14 200494
4. 1 168740
5. 18 162994
```

```
#Query which does not include issuer precinct numbered '0'
```

Query:

```
select issuerprecinct , count(*) as issuerPrecinct_tktcount from nycparking.parking_ext
where INSTR(IssueDate , '2017') > 0 and issuerprecinct > 0
group by (issuerprecinct) order by issuerPrecinct_tktcount DESC LIMIT 5;
```

4.violation code frequency across 3 precincts which have issued the most number of tickets

```
### Here my inner query first finds out those 3 issuerprecincts which issued maximum tickets in 2017
### I then run a query to find all the violation codes issued by these 3 issuer precincts.
### Finally, I visualize the result with combination of IssuerPrecinct,ViolationCode sorted in descending order of the occurrence (count)
```

Query:

```
select IssuerPrecinct,ViolationCode,count(*) as freq from nycparking.parking_ext where parking_ext.issuerprecinct in
(Select temp.IssuerPrecinct from (select IssuerPrecinct,count(*) freq from
 nycparking.parking_ext where IssuerPrecinct>0 AND INSTR(issuedate , '2017') > 0
 group by IssuerPrecinct order by freq desc limit 3) as temp)
 AND INSTR(parking_ext.issuedate , '2017') > 0
group by IssuerPrecinct,ViolationCode order by IssuerPrecinct,freq desc;
```

Output:

```
issuerprecinct  violationcode   freq
1   14  38354
1   16  19081
1   20  15408
1   46  12745
1   38  8535
1   17  7526
1   37  6470
1   31  5853
1   69  5672
1   19  5375
1   10  4712
1   40  4592
1   21  4055
1   71  3581
1   84  3310
1   42  2708
1   51  2223
```

| | | |
|---|---|---|
| 1 | 9 | 2206 |
| 1 | 70 | 2183 |
| 1 | 48 | 1907 |
| 1 | 53 | 1737 |
| 1 | 50 | 1374 |
| 1 | 13 | 1367 |
| 1 | 24 | 1193 |
| 1 | 74 | 1135 |
| 1 | 82 | 775 |
| 1 | 4 | 461 |
| 1 | 60 | 438 |
| 1 | 23 | 421 |
| 1 | 78 | 406 |
| 1 | 66 | 368 |
| 1 | 26 | 290 |
| 1 | 68 | 282 |
| 1 | 18 | 254 |
| 1 | 89 | 206 |
| 1 | 45 | 184 |
| 1 | 27 | 164 |
| 1 | 61 | 123 |
| 1 | 67 | 120 |
| 1 | 98 | 113 |
| 1 | 73 | 106 |
| 1 | 72 | 101 |
| 1 | 11 | 73 |
| 1 | 64 | 57 |
| 1 | 41 | 56 |
| 1 | 80 | 46 |
| 1 | 62 | 44 |
| 1 | 77 | 42 |
| 1 | 43 | 40 |
| 1 | 63 | 40 |
| 1 | 47 | 32 |
| 1 | 35 | 32 |
| 1 | 75 | 31 |
| 1 | 39 | 20 |
| 1 | 30 | 20 |
| 1 | 97 | 18 |
| 1 | 79 | 16 |
| 1 | 99 | 16 |
| 1 | 59 | 15 |
| 1 | 8 | 8 |
| 1 | 22 | 4 |
| 1 | 83 | 4 |
| 1 | 44 | 2 |
| 1 | 85 | 2 |
| 1 | 91 | 2 |
| 1 | 49 | 2 |
| 1 | 96 | 1 |
| 1 | 57 | 1 |
| 1 | 28 | 1 |
| 1 | 52 | 1 |
| 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
```

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.

#### To make Violation time field value in a proper format like: 24 hour format with hour:min , I make use of below method:

Method Used:

```
substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'), 'hhmma')),12,5)
```

#### The string value loaded from dataset looks like : '1204P' . Now to convert it into proper format , first thing is concatenating 'M' after this.
#### concat('1204P,'M');  => 1204PM
#### unix_timestamp(concat(ViolationTime,'M'), 'hhmma') => Converts time string in format yyyy-MM-dd hhmma to Unix time stamp
#### from_unixtime() => Converts the number of seconds from unix epoch (1970-01-01 00:00:00 UTC) to a string representing the
#### timestamp of that moment in the current system time zone in the format of "1970-01-01 00:00:00" which means - yyyy-MM-dd HH:mm:ss
#### The substring function retuns HH:mm as we start at index 12 and go for a length 5 from here.
#### Using hour(string date) : Returns the hour in int of the timestamp: hour('2009-07-30 12:58:59') = 12, hour('12:58:59') = 12

Hence,some of the properties of parking violations across different times of the day could be :

(i) Finding Violating Precincts with highest frequencies between 8-9 in morning. This can be any hour in the day. [Hour will be 8 from 8:00 to 8:59]

Query:

```
select violationprecinct , count(*) as violationPrecinct_tktcount from nycparking.parking_ext
where INSTR(IssueDate , '2017') > 0 AND hour(substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'), 'hhmma')),12)) = 8
group by (violationprecinct) order by violationPrecinct_tktcount DESC LIMIT 5;
```

(ii) Finding issuer precincts with highest no. of tickets issued between 6-7 in evening. This can be any time in the day. [Hour will be 18 from 18:00 to 18:59]

Query:

```
select issuerprecinct , count(*) as issuerPrecinct_tktcount from nycparking.parking_ext
where INSTR(IssueDate , '2017') > 0 AND hour(substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'), 'hhmma')),12)) = 18
group by (issuerprecinct) order by issuerPrecinct_tktcount DESC LIMIT 5;
```

(iii) How many tickets are issued at different times in a day. Find top 5 such slots.

Query:

```
select count(*) as tickets_issued from nycparking.parking_ext
where INSTR(IssueDate , '2017') > 0 group by
```

```
(hour(substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'), 'hhmma')),12)))
order by tickets_issued DESC LIMIT 5;


(iv) Find 3 most commonly occurring violations between 9 -11 PM

Query:

select violationcode , count(*) as frequency
from nycparking.parking_ext
where INSTR(IssueDate , '2017') > 0 AND hour(substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'), 'hhmma')),12)) BETWEEN 21 AND 23
group by violationcode order by frequency DESC limit 3;



6. Divide 24 hours into 6 equal discrete bins of time. For each of these groups, find the 3 most commonly occurring violations.
#### For this requirement, I would want to create a partitioned table segregated over a range such that
#### there are 6 equal sized partitions in 24 hours of the day when time is considered in 24 hour format of the day.
#### Since, I do not have any such column in my table, I will introduce one hour-range column and would partition the table based on that.
#### First step is to create a new partition table in my database and saving it on s3 bucket for later referencing and querying.

# dropping the table with this name if it existed

drop table if exists nycparking.parking_ext_part;

#setting the Hive properties to allow dynamic partitioning

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

#creating the table
#I have now taken Issue Date column as DATE as I am going to manipulate the data into DATE format.
#I have introduced 1 more column 'HourRange'of type int for partitioning


CREATE EXTERNAL TABLE if not exists NYCParking.parking_ext_part(
`SummonsNumber` bigint, `PlateID` string, `RegistrationState` string, `PlateType` string,`IssueDate` DATE,`ViolationCode` int,
`VehicleBodyType` string,`VehicleMake` string,`IssuingAgency` string,`StreetCode1` bigint, `StreetCode2` bigint, `StreetCode3` bigint,
`VehicleExpirationDate` bigint,
`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
)
PARTITIONED BY (HourRange int)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION 's3a://dee-hive-test/hiveAssignment/'
tblproperties ("skip.header.line.count"="1");



#Inserting data into table
#Manipulating the Issue Date format from 'MM/dd/yyyy' to default HIVE date format - 'yyyy-MM-dd'. to_date() method casts it accordingly.
#Using substring() method to get the HH:mm:ss format in Violation time field
```

```
#The substring function retuns HH:mm:ss as we start at index 12 and go till end from here.
#Creating a range of partitions based on hour extracted from ViolationTime field value.
#Using hour(string date) : Returns the hour in int of the timestamp: hour('12:58:59') = 12, hour('23:58:59') = 23
#Filtering out those records which do belong to year 2017.


insert overwrite table nycparking.parking_ext_part partition(HourRange)
select SummonsNumber , PlateID, RegistrationState, PlateType ,to_date(from_unixtime(UNIX_TIMESTAMP(IssueDate,'MM/dd/yyyy'))),ViolationCode ,
VehicleBodyType, VehicleMake ,IssuingAgency ,StreetCode1, StreetCode2, StreetCode3 , VehicleExpirationDate ,
ViolationLocation,ViolationPrecinct,IssuerPrecinct,IssuerCode,IssuerCommand,IssuerSquad,
substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'), 'hhmma')),12),TimeFirstObserved,ViolationCounty,ViolationInFrontOfOrOpposite,HouseNumber,StreetName,IntersectingStreet,DateFirstObserved,LawSection,SubDivision,ViolationLegalCode,DaysParkingInEffect,FromHoursInEffect,ToHoursInEffect, VehicleColor,UnregisteredVehicle ,VehicleYear ,MeterNumber,FeetFromCurb,ViolationPostCode, ViolationDescription, NoStandingOrStoppingViolation, HydrantViolation, DoubleParkingViolation ,
case
    when hour(substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'), 'hhmma')),12)) BETWEEN 0 AND 3 THEN 1
    when hour(substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'), 'hhmma')),12)) BETWEEN 4 AND 7 THEN 2
    when hour(substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'), 'hhmma')),12)) BETWEEN 8 AND 11 THEN 3
    when hour(substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'), 'hhmma')),12)) BETWEEN 12 AND 15 THEN 4
    when hour(substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'), 'hhmma')),12)) BETWEEN 16 AND 19 THEN 5
    when hour(substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'), 'hhmma')),12)) BETWEEN 20 AND 23 THEN 6
    else 0
end as HourRange from nycparking.parking_ext where INSTR(IssueDate , '2017') > 0;

#Finding 3 most commonly occurring violations for time bin - 0 to 3

Query:

select violationcode , count(*) as frequency , rank() over (ORDER by frequency)
from nycparking.parking_ext_part
where hourrange = 1
group by violationcode order by frequency DESC LIMIT 3;

Output:

ViolationCode frequency
21 36965

40 25867

78 15527

#Finding 3 most commonly occurring violations for time bin - 4 to 7

Query:

select violationcode , count(*) as frequency
from nycparking.parking_ext_part
where hourrange = 2
group by violationcode order by frequency DESC limit 3;

Output:

violationcode frequency
14 74112

40 60651

```

```
21 57895


#Finding 3 most commonly occurring violations for time bin - 8 to 11

Query:

select violationcode , count(*) as frequency
from nycparking.parking_ext_part
where hourrange = 3
group by violationcode order by frequency DESC limit 3;

Output:

violationcode frequency
21 598050

36 348163

38 176570

#Finding 3 most commonly occurring violations for time bin - 12 to 15

Query:

select violationcode , count(*) as frequency
from nycparking.parking_ext_part
where hourrange = 4
group by violationcode order by frequency DESC limit 3;


Output:

violationcode frequency
36 286282

38 240719

37 167026

#Finding 3 most commonly occurring violations for time bin - 16 to 19

Query:

select violationcode , count(*) as frequency
from nycparking.parking_ext_part
where hourrange = 5
group by violationcode order by frequency DESC limit 3;


Output:

violationcode frequency
38 102855

14 75902

37 70345

#Finding 3 most commonly occurring violations for time bin - 20 to 23

Query:

select violationcode , count(*) as frequency
from nycparking.parking_ext_part
where hourrange = 6
group by violationcode order by frequency DESC limit 3;

Output:

violationcode frequency
7 26291

```

```
40 22338

14 21045
```

7. For the 3 most commonly occurring violation codes, find the most common times of day

```
### Here my inner query first finds out those 3 violationcodes which had maximum tickets against them
### I then run a query to find all the hour ranges (time bins created in above question) at which these violationcodes had tickets issued against them
### Finally, I visualize the result with combination of ViolationCode,hourrange sorted in descending order of the occurrence (count)
```

Query:

```
select violationcode, hourrange , count(*) as freq from nycparking.parking_ext_part where parking_ext_part.violationcode in
(Select temp.violationcode from (select violationcode , count(*) as frequency
from nycparking.parking_ext_part group by violationcode ORDER by frequency DESC LIMIT 3) as temp)
group by violationcode, hourrange order by violationcode,freq desc;
```

Output :

```
violationcode   hourrange   freq
21  3   598050
21  4   74693
21  2   57895
21  1   36965
21  5   264
21  6   189
21  0   17
36  3   348163
36  4   286282
36  2   14782
36  5   13534
38  4   240719
38  3   176570
38  5   102855
38  6   20347
38  2   1273
38  1   312
```

8.  Divide the year into some number of seasons. Find some seasonality in data.

```
#### For this requirement, I would want to create a partitioned table segregated over a range such that
#### there are 4 equal sized partitions in an year based on different seasons.
#### Since, I do not have any such column in my table, I will introduce one Season column and would partition the table based on that.
#### First step is to create a new partition table in my database and saving it on s3 bucket for later referencing and querying.

# dropping the table with this name if it existed
drop table if exists nycparking.parking_ext_part_season;

#setting the Hive properties to allow dynamic partitioning

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

#creating the table
#I have now taken Issue Date column as DATE as I am going to manipulate the data into DATE format and then extract month out of it.
#I have introduced 1 more column 'Season' of type string for partitioning

CREATE EXTERNAL TABLE if not exists NYCParking.parking_ext_part_season(
`SummonsNumber` bigint, `PlateID` string, `RegistrationState` string, `PlateType`
string,`IssueDate` DATE,`ViolationCode` int,
`VehicleBodyType` string,`VehicleMake` string,`IssuingAgency` string,`StreetCode1`
bigint, `StreetCode2` bigint, `StreetCode3` bigint,
`VehicleExpirationDate` bigint,
`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
)
PARTITIONED BY (Season string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION 's3a://dee-hive-test/hiveAssignment/'
tblproperties ("skip.header.line.count"="1");



#Inserting data into table
#unix_timestamp(IssueDate,'MM/dd/yyyy') => Converts time string in format MM/dd/yyyy
to Unix time stamp
#from_unixtime() => Converts the number of seconds from unix epoch (1970-01-01
00:00:00 UTC) to a string representing the
#Creating a range of partitions based on month extracted from IssueDate field value.
#to_date() method casts a date into Hive date format of yyyy-MM-dd format.
#Using month(string date) : Returns the month part of a date or a timestamp string:
month(“1970-11-01 00:00:00”) = 11, month(“1970-11-01”) = 11
#Filtering out those records which do belong to year 2017.



insert overwrite table nycparking.parking_ext_part_season partition(Season)
select SummonsNumber , PlateID, RegistrationState, PlateType
,to_date(from_unixtime(UNIX_TIMESTAMP(IssueDate,'MM/dd/yyyy'))),ViolationCode ,
VehicleBodyType, VehicleMake ,IssuingAgency ,StreetCode1, StreetCode2, StreetCode3 ,
VehicleExpirationDate ,
ViolationLocation,ViolationPrecinct,IssuerPrecinct,IssuerCode,IssuerCommand,IssuerSqua
d,
ViolationTime,TimeFirstObserved,ViolationCounty,ViolationInFrontOfOrOpposite,HouseNumb
er,StreetName,IntersectingStreet,DateFirstObserved,LawSection,SubDivision,ViolationLeg
alCode,DaysParkingInEffect,FromHoursInEffect,ToHoursInEffect,
VehicleColor,UnregisteredVehicle ,VehicleYear
,MeterNumber,FeetFromCurb,ViolationPostCode, ViolationDescription,
NoStandingOrStoppingViolation, HydrantViolation, DoubleParkingViolation ,
case
    when month(to_date(from_unixtime(UNIX_TIMESTAMP(IssueDate,'MM/dd/yyyy'))))
    BETWEEN 3 AND 5 THEN 'Spring'
    when month(to_date(from_unixtime(UNIX_TIMESTAMP(IssueDate,'MM/dd/yyyy'))))
    BETWEEN 6 AND 8 THEN 'Summer'
    when month(to_date(from_unixtime(UNIX_TIMESTAMP(IssueDate,'MM/dd/yyyy'))))
    BETWEEN 9 AND 11 THEN 'Fall'
    when month(to_date(from_unixtime(UNIX_TIMESTAMP(IssueDate,'MM/dd/yyyy')))) == 12
    OR
    month(to_date(from_unixtime(UNIX_TIMESTAMP(IssueDate,'MM/dd/yyyy')))) == 1 OR
    month(to_date(from_unixtime(UNIX_TIMESTAMP(IssueDate,'MM/dd/yyyy')))) == 2 THEN
    'Winter'
    else 0
end as Season from nycparking.parking_ext where INSTR(IssueDate , '2017') > 0;



# Checking how many total records are there in this partition table

Query:

select count(*)
from nycparking.parking_ext_part_season;
Ans. 5431878
```

8. (1) find frequencies of tickets for each season.

#No of tickets issued in Fall Season:
#Finding total tickets based on partition = Fall

Query:

```
select count(DISTINCT summonsnumber) as tickets_fall
from nycparking.parking_ext_part_season
where season = 'Fall';
```

Output:

```
tickets_fall
978
```

#No of tickets issued in Spring Season:
#Finding total tickets based on partition = Spring

Query:

```
select count(DISTINCT summonsnumber) as tickets_spring
from nycparking.parking_ext_part_season where season = 'Spring';
```

Output:

```
tickets_spring
2873372
```

#No of tickets issued in Summer Season:
#Finding total tickets based on partition = Summer

Query:

```
select count(DISTINCT summonsnumber) as tickets_summer
from nycparking.parking_ext_part_season where season = 'Summer';
```

Output:

```
tickets_summer
852856
```

#No of tickets issued in Winter Season:
#Finding total tickets based on partition = Winter

Query:

```
select count(DISTINCT summonsnumber) as tickets_winter
from nycparking.parking_ext_part_season where season = 'Winter';
```

Output:

```
tickets_winter
1704672
```

8. (2) find the 3 most common violations for each of these seasons

#Finding the most common violation codes against which tickets are issued in partition Fall
#I am ordering the set into descending order of occurrence(count) and then picking top 3 by using -> LIMIT 3

Query:

```
select violationcode , count(*) as frequency
```

```
from nycparking.parking_ext_part_season
where season = 'Fall'
group by violationcode order by frequency DESC limit 3;

Output:

violationcode   frequency
46  230
21  128
40  116

#Finding the most common violation codes against which tickets are issued in partition Spring
#I am ordering the set into descending order of occurrence(count) and then picking top 3 by using -> LIMIT 3

Query:

select violationcode , count(*) as frequency
from nycparking.parking_ext_part_season
where season = 'Spring'
group by violationcode order by frequency DESC limit 3;


Output:

violationcode   frequency
21  402423
36  344834
38  271167


#Finding the most common violation codes against which tickets are issued in partition Summer
#I am ordering the set into descending order of occurrence(count) and then picking top 3 by using -> LIMIT 3

Query:

select violationcode , count(*) as frequency
from nycparking.parking_ext_part_season
where season = 'Summer'
group by violationcode order by frequency DESC limit 3;

Output:

violationcode   frequency
21  127349
36  96663
38  83517


#Finding the most common violation codes against which tickets are issued in partition Winter
#I am ordering the set into descending order of occurrence(count) and then picking top 3 by using -> LIMIT 3

Query:

select violationcode , count(*) as frequency
from nycparking.parking_ext_part_season
where season = 'Winter'
group by violationcode order by frequency DESC limit 3;

Output:

violationcode   frequency
21  238178
36  221267
38  187386


#########################################################################################
```

```
##########################################################################
################################################          End of Assignment
Questions    ######################################################################

Best Practices:

1. You should now drop the tables and database if you do not need it any further.
2. Use DROP Table and then DROP database commands to drop the non-required items.
3. You should now STOP your EC2 instance and come out of the EC2 terminal.
4. You should now delete the buckets created on S3 if you do not need those any
further.


###########################################################       End of
Document           ################################################################
```