

```

1 Introduction:
2
3 The assignment is meant for me to apply learnings of the module on Hive on a
  real-life dataset.
4 The objective is to familiarize me with necessary optimizations for my queries like
  selecting the appropriate table format,
5 using partitioned/bucketed tables and moreover acquainting me with the different
  functions of HIVE etc.
6
7 To be able to execute these queries , please follow the following steps:
8
9 Step 1. Log into AWS Console
10 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)
11 Step 3. Start the EC2 instance
12 Step 4. Once the instance is ON, log into EC2 terminal using Putty. For this you
  will require - Hostname or IP address,
13             which is the Public IP of your EC2 instance which has CDH installation.
             After logging in as 'ec2-user', switch to root user -
14             sudo -i command
15 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
16 Step 6: Execute following command on EC2 terminal:
17             aws s3 cp
             s3://hiveassignmentdatabde/Parking_Violations_Issued_-_Fiscal_Year_2017.csv
             s3://dee-hive-test
18 Step 7. Once this data is successfully copied , let us now start creating required
  database and tables in Hive using HUE.
19 Step 8. Hit the URL : <Public IP of Ec2 instance>:7180 , this will take you to
  Cloudera Manager login page.
20 Step 9. Login using credentials provided. Ensure that you have completed tuning of
  HIVE and Hue in Cloudera Manager by
21             previously following documents and sessions conducted for the same.
22 Step 10. Launch the Hue Editor. And now work on the queries one by one.
23
24 ##### 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:
25
26 create database IF not exists NYCParking location
  's3a://dee-hive-test/hiveAssignment/' ;
27
28
29 ##### Now, creating a table from the data. The below statement when executed will
  create the table.
30 ##### For creating the table with different fields and data types , I have followed
  the Data Dictionary
31 ##### present on:
https://data.cityofnewyork.us/City-Government/Parking-Violations-Issued-Fiscal-Year-2017/2bnn-yakx
32
33 create external IF NOT EXISTS table NYCParking.parking_ext(`SummonsNumber` bigint,
  `PlateID` string, `RegistrationState` string, `PlateType` string, `IssueDate`
  string, `ViolationCode` int,
34 `VehicleBodyType` string, `VehicleMake` string, `IssuingAgency` string, `StreetCode1`
  bigint, `StreetCode2` bigint, `StreetCode3` bigint, `VehicleExpirationDate` bigint,
35 `ViolationLocation` string, `ViolationPrecinct` int, `IssuerPrecinct` int, `IssuerCode`
  int, `IssuerCommand` string, `IssuerSquad` string, `ViolationTime` string,
36 `TimeFirstObserved` string, `ViolationCounty` string, `ViolationInFrontOfOrOpposite`
  string, `HouseNumber` string, `StreetName` string, `IntersectingStreet` string,
37 `DateFirstObserved` int, `LawSection` int, `SubDivision` string, `ViolationLegalCode`
  string, `DaysParkingInEffect` string,
38 `FromHoursInEffect` string, `ToHoursInEffect` string, `VehicleColor`
  string, `UnregisteredVehicle` string, `VehicleYear` int, `MeterNumber` string,
39 `FeetFromCurb` int, `ViolationPostCode` string, `ViolationDescription` string,
40 `NoStandingOrStoppingViolation` string, `HydrantViolation`
  string, `DoubleParkingViolation` string)
41 row format delimited fields terminated by ','
42 location 's3a://dee-hive-test/hiveAssignment/'
43 tblproperties ("skip.header.line.count"="1");
44
45
46 ##### Checking if table is successfully created in the database.
47

```

```

48 SHOW TABLES in nycparking;
49
50 ##### Checking if the table created contains any data or not ?
51
52 select * from nycparking.parking_ext;
53
54 ##### Now populating the table created with the input data set that I had copied
into my S3 bucket:
55
56 load data inpath
'S3A://dee-hive-test/Parking_Violations_Issued_-_Fiscal_Year_2017.csv' overwrite
into table nycparking.parking_ext;
57
58 ##### Checking how the data looks like in the table, if all the values are
correctly loaded into the table in the expected formats.
59
60 select * from nycparking.parking_ext limit 10;
61
62 ##### 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)
63 ##### which returns the position of the first occurrence of substr in col value.
64 ##### This method returns an int > 0 if the substr is found in the col value.
65 ##### I would be using this method in my queries where I need to filter records of
only year - 2017
66 ##### Below I am only testing if this string works or not.
67
68 select plateid , issuedate from nycparking.parking_ext where INSTR(IssueDate ,
'2017') > 0 limit 10;
69
70 #####
#####
71
72 Part-I: Examine the data
73
74 #####
#####
75
76 1. Find the total number of tickets for the year.
77 ## Here , I am using count(Distinct) on all tickets (summonsnumber) to get the total
record count in dataset, after loading into table.
78 ## My intention is to filter records only from year 2017.
79
80 Query:
81
82 select count(DISTINCT summonsnumber) as tickets_count from nycparking.parking_ext
where INSTR(IssueDate , '2017') > 0;
83
84 Output:
85
86 tickets_count
87 5431903
88
89 2. Find out how many unique states the cars which got parking tickets came from
90 ## 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
91
92 Query:
93
94 select count(DISTINCT violationcounty) as unique_states from nycparking.parking_ext
where INSTR(IssueDate , '2017') > 0;
95
96 Output:
97
98 unique_states
99 12
100
101 3. Finding tickets from the dataset which do not have addresses on them
102 ## 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
103 ## we would want to get the count of such parking tickets , also from year 2017 only.
104
105 Query:
106

```

```

107 select count(DISTINCT summonsnumber) as withoutaddress_count from
nycparking.parking_ext
108 where (streetcode1 is null OR streetcode1 = '0' OR streetcode2 is null OR
streetcode2 = '0' OR streetcode3 is null OR streetcode3 = '0')
109 AND INSTR(IssueDate , '2017') > 0 ;
110
111 Output:
112
113 withoutaddress_count
114 1816816
115
116 ### Below is another way of writing the same query where I first filter records from
year 2017 only and then find the count of
117 ### such without address tickets from the set
118
119
120 select count(DISTINCT summonsnumber) as withoutaddress_count from
121 (select * from nycparking.parking_ext where INSTR(IssueDate , '2017') > 0) a
122 where (streetcode1 is null OR streetcode1 = '0' OR streetcode2 is null OR
streetcode2 = '0' OR streetcode3 is null OR streetcode3 = '0');
123
124
125 withoutaddress_count
126 1816816
127
128
129 #####
#####
130 Part-II: Aggregation tasks
131 #####
#####
132
133 1. Finding frequency of violation codes - find the top 5
134
135 ### Here, I get the occurrence (count) of all violationcodes which have taken place
in year 2017.
136 ### I then Group these by individual violationcodes and sort them in descending
order of their occurrence (count)
137 ### 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]
138
139 Query:
140
141 select violationcode , count(*) as violationcode_frequency from
nycparking.parking_ext where INSTR(IssueDate , '2017') > 0
142 group by (violationcode) order by violationcode_frequency DESC limit 5;
143
144 OUTPUT:
145
146 1. 21 768082
147 2. 36 662765
148 3. 38 542079
149 4. 14 476660
150 5. 20 319646
151
152 374.373 seconds
153
154
155 2. (i) Finding frequency of each vehicle body type getting a parking ticket
156 ### Here, I get the occurrence (count) of all parking violations by different
vehicle body types which have taken place in year 2017.
157 ### I then Group these by individual vehiclebodytype and sort them in descending
order of their occurrence (count)
158 ### 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]
159
160 Query:
161
162 select vehiclebodytype , count(*) as vehiclebody_tktcount from nycparking.parking_ext
163 where INSTR(IssueDate , '2017') > 0 group by (vehiclebodytype) order by
vehiclebody_tktcount DESC LIMIT 5;
164
165 OUTPUT:

```

```

166
167 vehiclebodytype vehiclebody_tktcount
168 1. SUBN 1883953
169 2. 4DSD 1547307
170 3. VAN 724025
171 4. DELV 358982
172 5. SDN 194197
173
174 375.669 seconds
175
176 2. (ii) Finding frequency of each vehicle make getting a parking ticket
177 ### Here, I get the occurrence (count) of all parking violations by different
vehicle make which have taken place in year 2017.
178 ### I then Group these by individual vehiclemake and sort them in descending order
of their occurrence (count)
179 ### 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]
180
181 Query:
182
183 select vehiclemake , count(*) as vehiclemake_tktcount from nycparking.parking_ext
184 where INSTR(IssueDate , '2017') > 0 group by (vehiclemake) order by
vehiclemake_tktcount DESC LIMIT 5;
185
186 Output:
187
188 vehiclemake vehiclemake_tktcount
189 1. FORD 636842
190 2. TOYOT 605290
191 3. HONDA 538884
192 4. NISSA 462017
193 5. CHEVR 356032
194
195 3. (1)Finding Violating Precincts with highest frequencies
196
197 ## Query which includes violationprecinct numbered '0' too in the result.
198 ## This pulls out all violationprecincts where maximum violations occurred in year
2017
199 ## It then groups the data by these violationprecincts and then sorts the data in
descending order of their occurrences(count)
200 ## 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]
201
202
203 Query:
204
205 select violationprecinct , count(*) as violationPrecinct_tktcount from
nycparking.parking_ext
206 where INSTR(IssueDate , '2017') > 0
207 group by (violationprecinct) order by violationPrecinct_tktcount DESC LIMIT 5;
208
209 Output:
210
211 violationprecinct violationprecinct_tktcount
212 1. 0 925596
213 2. 19 274443
214 3. 14 203552
215 4. 1 174702
216 5. 18 169131
217
218 #Query which does not include violation precinct numbered 0
219
220 Query:
221
222 select violationprecinct , count(*) as violationPrecinct_tktcount from
nycparking.parking_ext
223 where INSTR(IssueDate , '2017') > 0 AND violationprecinct > 0
224 group by (violationprecinct) order by violationPrecinct_tktcount DESC LIMIT 5;
225
226
227 3. (2)Finding Issuer Precincts with highest frequencies
228
229 ## Query which includes Issuerprecinct numbered '0' too in the result.

```

```

230  ## This pulls out all issuerprecincts where maximum tickets were issued in year 2017
231  ## It then groups the data by these issuerprecincts and then sorts the data in
    descending order of their occurrences(count)
232  ## 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]
233
234
235  Query:
236
237  select issuerprecinct , count(*) as issuerPrecinct_tktcount from
    nycparking.parking_ext
238  where INSTR(IssueDate , '2017') > 0
239  group by (issuerprecinct) order by issuerPrecinct_tktcount DESC LIMIT 5;
240
241  Output:
242
243  issuerprecinct issuerprecinct_tktcount
244  1. 0 1078403
245  2. 19 266959
246  3. 14 200494
247  4. 1 168740
248  5. 18 162994
249
250  #Query which does not include issuer precinct numbered '0'
251
252  Query:
253
254  select issuerprecinct , count(*) as issuerPrecinct_tktcount from
    nycparking.parking_ext
255  where INSTR(IssueDate , '2017') > 0 and issuerprecinct > 0
256  group by (issuerprecinct) order by issuerPrecinct_tktcount DESC LIMIT 5;
257
258  4.violation code frequency across 3 precincts which have issued the most number of
    tickets
259
260  ### Here my inner query first finds out those 3 issuerprecincts which issued maximum
    tickets in 2017
261  ### I then run a query to find all the violation codes issued by these 3 issuer
    precincts.
262  ### Finally, I visualize the result with combination of IssuerPrecinct, ViolationCode
    sorted in descending order of the occurrence (count)
263
264
265  Query:
266
267  select IssuerPrecinct, ViolationCode, count(*) as freq from nycparking.parking_ext
    where parking_ext.issuerprecinct in
268  (Select temp.IssuerPrecinct from (select IssuerPrecinct, count(*) freq from
269  nycparking.parking_ext where IssuerPrecinct>0 AND INSTR(issuedate , '2017') > 0
    group by IssuerPrecinct order by freq desc limit 3) as temp)
270  AND INSTR(parking_ext.issuedate , '2017') > 0
271  group by IssuerPrecinct, ViolationCode order by IssuerPrecinct, freq desc;
272
273  Output:
274
275  issuerprecinct  violationcode    freq
276  1      14      38354
277  1      16      19081
278  1      20      15408
279  1      46      12745
280  1      38      8535
281  1      17      7526
282  1      37      6470
283  1      31      5853
284  1      69      5672
285  1      19      5375
286  1      10      4712
287  1      40      4592
288  1      21      4055
289  1      71      3581
290  1      84      3310
291  1      42      2708
292  1      51      2223

```

293	1	9	2206
294	1	70	2183
295	1	48	1907
296	1	53	1737
297	1	50	1374
298	1	13	1367
299	1	24	1193
300	1	74	1135
301	1	82	775
302	1	4	461
303	1	60	438
304	1	23	421
305	1	78	406
306	1	66	368
307	1	26	290
308	1	68	282
309	1	18	254
310	1	89	206
311	1	45	184
312	1	27	164
313	1	61	123
314	1	67	120
315	1	98	113
316	1	73	106
317	1	72	101
318	1	11	73
319	1	64	57
320	1	41	56
321	1	80	46
322	1	62	44
323	1	77	42
324	1	43	40
325	1	63	40
326	1	47	32
327	1	35	32
328	1	75	31
329	1	39	20
330	1	30	20
331	1	97	18
332	1	79	16
333	1	99	16
334	1	59	15
335	1	8	8
336	1	22	4
337	1	83	4
338	1	44	2
339	1	85	2
340	1	91	2
341	1	49	2
342	1	96	1
343	1	57	1
344	1	28	1
345	1	52	1
346	14	14	45036
347	14	69	30464
348	14	31	22555
349	14	47	18364
350	14	42	10027
351	14	46	7679
352	14	19	7030
353	14	84	6743
354	14	82	5052
355	14	40	3582
356	14	17	3534
357	14	38	3269
358	14	9	2874
359	14	20	2761
360	14	71	2757
361	14	13	2701
362	14	48	2439
363	14	89	1960
364	14	50	1824
365	14	11	1745

```

366 14 79 1495
367 14 70 1461
368 14 10 1319
369 14 37 1256
370 14 64 1070
371 14 23 1044
372 14 21 1029
373 14 53 953
374 14 24 946
375 14 16 940

```

```

376
377
378

```

379 5. Find out the properties of parking violations across different times of the day:  
380 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.

381  
382 ##### To make Violation time field value in a proper format like: 24 hour format with  
hour:min , I make use of below method:

383  
384 Method Used:

```

385
386 substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'), 'hhmma')),12,5)
387

```

388 ##### The string value loaded from dataset looks like : '1204P' . Now to convert it  
into proper format , first thing is concatenating 'M' after this.

389 ##### concat('1204P','M'); => 1204PM

390 ##### unix\_timestamp(concat(ViolationTime,'M'), 'hhmma') => Converts time string in  
format yyyy-MM-dd hhmma to Unix time stamp

391 ##### from\_unixtime() => Converts the number of seconds from unix epoch (1970-01-01  
00:00:00 UTC) to a string representing the

392 ##### 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

393 ##### The substring function returns HH:mm as we start at index 12 and go for a length  
5 from here.

394 ##### 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

```

```

395
396

```

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

```

398
399

```

400 (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]

401  
402 Query:

```

403
404 select violationprecinct , count(*) as violationPrecinct_tktcount from
nycparking.parking_ext
405 where INSTR(IssueDate , '2017') > 0 AND
hour(substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'),
'hhmma')),12)) = 8
406 group by (violationprecinct) order by violationPrecinct_tktcount DESC LIMIT 5;
407

```

408 (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]

409  
410 Query:

```

411
412 select issuerprecinct , count(*) as issuerPrecinct_tktcount from
nycparking.parking_ext
413 where INSTR(IssueDate , '2017') > 0 AND
hour(substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'),
'hhmma')),12)) = 18
414 group by (issuerprecinct) order by issuerPrecinct_tktcount DESC LIMIT 5;
415

```

416 (iii) How many tickets are issued at different times in a day. Find top 5 such slots.  
417

418 Query:

```

419
420 select count(*) as tickets_issued from nycparking.parking_ext
421 where INSTR(IssueDate , '2017') > 0 group by

```

```

    (hour(substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'),
'hhmma')),12)))
422 order by tickets_issued DESC LIMIT 5;
423
424
425 (iv) Find 3 most commonly occurring violations between 9 -11 PM
426
427 Query:
428
429 select violationcode , count(*) as frequency
430 from nycparking.parking_ext
431 where INSTR(IssueDate , '2017') > 0 AND
hour(substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'),
'hhmma')),12)) BETWEEN 21 AND 23
432 group by violationcode order by frequency DESC limit 3;
433
434
435
436 6. Divide 24 hours into 6 equal discrete bins of time. For each of these groups,
find the 3 most commonly occurring violations.
437 ##### For this requirement, I would want to create a partitioned table segregated
over a range such that
438 ##### there are 6 equal sized partitions in 24 hours of the day when time is
considered in 24 hour format of the day.
439 ##### 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.
440 ##### First step is to create a new partition table in my database and saving it on
s3 bucket for later referencing and querying.
441
442 # dropping the table with this name if it existed
443
444 drop table if exists nycparking.parking_ext_part;
445
446 #setting the Hive properties to allow dynamic partitioning
447
448 set hive.exec.dynamic.partition =true;
449 set hive.exec.dynamic.partition.mode=nonstrict;
450
451 #creating the table
452 #I have now taken Issue Date column as DATE as I am going to manipulate the data
into DATE format.
453 #I have introduced 1 more column 'HourRange'of type int for partitioning
454
455
456 CREATE EXTERNAL TABLE if not exists NYCParking.parking_ext_part(
457 `SummonsNumber` bigint, `PlateID` string, `RegistrationState` string, `PlateType`
string, `IssueDate` DATE, `ViolationCode` int,
458 `VehicleBodyType` string, `VehicleMake` string, `IssuingAgency` string, `StreetCode1`
bigint, `StreetCode2` bigint, `StreetCode3` bigint,
459 `VehicleExpirationDate` bigint,
460 `ViolationLocation` string, `ViolationPrecinct` int, `IssuerPrecinct` int, `IssuerCode`
int, `IssuerCommand` string, `IssuersSquad` string, `ViolationTime` string,
`TimeFirstObserved` string, `ViolationCounty` string, `ViolationInFrontOfOrOpposite`
string, `HouseNumber` string, `StreetName` string, `IntersectingStreet` string,
461 `DateFirstObserved` int, `LawSection` int, `SubDivision` string, `ViolationLegalCode`
string, `DaysParkingInEffect` string,
462 `FromHoursInEffect` string, `ToHoursInEffect` string, `VehicleColor`
string, `UnregisteredVehicle` string, `VehicleYear` int, `MeterNumber` string,
463 `FeetFromCurb` int, `ViolationPostCode` string, `ViolationDescription` string,
464 `NoStandingOrStoppingViolation` string, `HydrantViolation`
string, `DoubleParkingViolation` string
465 )
466 PARTITIONED BY (HourRange int)
467 ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
468 LOCATION 's3a://dee-hive-test/hiveAssignment/'
469 tblproperties ("skip.header.line.count"="1");
470
471
472
473 #Inserting data into table
474 #Manipulating the Issue Date format from 'MM/dd/yyyy' to default HIVE date format -
'yyyy-MM-dd'. to_date() method casts it accordingly.
475 #Using substring() method to get the HH:mm:ss format in Violation time field

```



```

476 #The substring function returns HH:mm:ss as we start at index 12 and go till end from
477 here.
478 #Creating a range of partitions based on hour extracted from ViolationTime field
479 value.
480 #Using hour(string date) : Returns the hour in int of the timestamp:
481 hour('12:58:59') = 12, hour('23:58:59') = 23
482 #Filtering out those records which do belong to year 2017.
483
484 insert overwrite table nycparking.parking_ext_part partition(HourRange)
485 select SummonsNumber , PlateID, RegistrationState, PlateType
486 ,to_date(from_unixtime(UNIX_TIMESTAMP(IssueDate,'MM/dd/yyyy'))),ViolationCode ,
487 VehicleBodyType, VehicleMake ,IssuingAgency ,StreetCode1, StreetCode2, StreetCode3 ,
488 VehicleExpirationDate ,
489 ViolationLocation,ViolationPrecinct,IssuerPrecinct,IssuerCode,IssuerCommand,IssuerSquad,
490 substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'),
491 'hhmma')),12),TimeFirstObserved,ViolationCounty,ViolationInFrontOfOrOpposite,HouseNumber,
492 StreetName,IntersectingStreet,DateFirstObserved,LawSection,SubDivision,ViolationLegalCode,
493 DaysParkingInEffect,FromHoursInEffect,ToHoursInEffect,VehicleColor,UnregisteredVehicle ,VehicleYear
494 ,MeterNumber,FeetFromCurb,ViolationPostCode, ViolationDescription,
495 NoStandingOrStoppingViolation, HydrantViolation, DoubleParkingViolation ,
496 case
497     when hour(substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'),
498     'hhmma')),12)) BETWEEN 0 AND 3 THEN 1
499     when hour(substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'),
500     'hhmma')),12)) BETWEEN 4 AND 7 THEN 2
501     when hour(substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'),
502     'hhmma')),12)) BETWEEN 8 AND 11 THEN 3
503     when hour(substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'),
504     'hhmma')),12)) BETWEEN 12 AND 15 THEN 4
505     when hour(substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'),
506     'hhmma')),12)) BETWEEN 16 AND 19 THEN 5
507     when hour(substring(from_unixtime(unix_timestamp(concat(ViolationTime,'M'),
508     'hhmma')),12)) BETWEEN 20 AND 23 THEN 6
509     else 0
510 end as HourRange from nycparking.parking_ext where INSTR(IssueDate , '2017') > 0;
511
512 #Finding 3 most commonly occurring violations for time bin - 0 to 3
513
514 Query:
515
516 select violationcode , count(*) as frequency , rank() over (ORDER by frequency)
517 from nycparking.parking_ext_part
518 where hourrange = 1
519 group by violationcode order by frequency DESC LIMIT 3;
520
521 Output:
522
523 ViolationCode frequency
524 21 36965
525
526 40 25867
527
528 78 15527
529
530 #Finding 3 most commonly occurring violations for time bin - 4 to 7
531
532 Query:
533
534 select violationcode , count(*) as frequency
535 from nycparking.parking_ext_part
536 where hourrange = 2
537 group by violationcode order by frequency DESC limit 3;
538
539 Output:
540
541 violationcode frequency
542 14 74112
543
544 40 60651
545

```

```
531 21 57895
532
533
534 #Finding 3 most commonly occurring violations for time bin - 8 to 11
535
536 Query:
537
538 select violationcode , count(*) as frequency
539 from nycparking.parking_ext_part
540 where hourrange = 3
541 group by violationcode order by frequency DESC limit 3;
542
543 Output:
544
545 violationcode frequency
546 21 598050
547
548 36 348163
549
550 38 176570
551
552 #Finding 3 most commonly occurring violations for time bin - 12 to 15
553
554 Query:
555
556 select violationcode , count(*) as frequency
557 from nycparking.parking_ext_part
558 where hourrange = 4
559 group by violationcode order by frequency DESC limit 3;
560
561
562 Output:
563
564 violationcode frequency
565 36 286282
566
567 38 240719
568
569 37 167026
570
571 #Finding 3 most commonly occurring violations for time bin - 16 to 19
572
573 Query:
574
575 select violationcode , count(*) as frequency
576 from nycparking.parking_ext_part
577 where hourrange = 5
578 group by violationcode order by frequency DESC limit 3;
579
580
581 Output:
582
583 violationcode frequency
584 38 102855
585
586 14 75902
587
588 37 70345
589
590 #Finding 3 most commonly occurring violations for time bin - 20 to 23
591
592 Query:
593
594 select violationcode , count(*) as frequency
595 from nycparking.parking_ext_part
596 where hourrange = 6
597 group by violationcode order by frequency DESC limit 3;
598
599 Output:
600
601 violationcode frequency
602 7 26291
603
```

```

604 40 22338
605
606 14 21045
607
608
609
610 7. For the 3 most commonly occurring violation codes, find the most common times of
day
611
612 ### Here my inner query first finds out those 3 violationcodes which had maximum
tickets against them
613 ### 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
614 ### Finally, I visualize the result with combination of ViolationCode,hourrange
sorted in descending order of the occurrence (count)
615
616
617 Query:
618
619 select violationcode, hourrange , count(*) as freq from nycparking.parking_ext_part
where parking_ext_part.violationcode in
620 (select temp.violationcode from (select violationcode , count(*) as frequency
621 from nycparking.parking_ext_part group by violationcode ORDER by frequency DESC
LIMIT 3) as temp)
622 group by violationcode, hourrange order by violationcode,freq desc;
623
624 Output :
625
626 violationcode    hourrange    freq
627 21  3    598050
628 21  4    74693
629 21  2    57895
630 21  1    36965
631 21  5     264
632 21  6     189
633 21  0      17
634 36  3   348163
635 36  4   286282
636 36  2   14782
637 36  5   13534
638 38  4   240719
639 38  3   176570
640 38  5   102855
641 38  6    20347
642 38  2     1273
643 38  1     312
644
645
646 8. Divide the year into some number of seasons. Find some seasonality in data.
647
648 ##### For this requirement, I would want to create a partitioned table segregated
over a range such that
649 ##### there are 4 equal sized partitions in an year based on different seasons.
650 ##### 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.
651 ##### First step is to create a new partition table in my database and saving it on
s3 bucket for later referencing and querying.
652
653 # dropping the table with this name if it existed
654 drop table if exists nycparking.parking_ext_part_season;
655
656 #setting the Hive properties to allow dynamic partitioning
657
658 set hive.exec.dynamic.partition =true;
659 set hive.exec.dynamic.partition.mode=nonstrict;
660
661 #creating the table
662 #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.
663 #I have introduced 1 more column 'Season' of type string for partitioning
664
665 CREATE EXTERNAL TABLE if not exists NYCParking.parking_ext_part_season(
666 `SummonsNumber` bigint, `PlateID` string, `RegistrationState` string, `PlateType`

```

```

667 string,`IssueDate` DATE,`ViolationCode` int,
`VehicleBodyType` string,`VehicleMake` string,`IssuingAgency` string,`StreetCode1`
668 bigint,`StreetCode2` bigint,`StreetCode3` bigint,
669 `VehicleExpirationDate` bigint,
`ViolationLocation` string,`ViolationPrecinct` int,`IssuerPrecinct` int,`IssuerCode`
int,`IssuerCommand` string,`IssuerSquad` string,`ViolationTime` string,
`TimeFirstObserved` string,`ViolationCounty` string,`ViolationInFrontOfOrOpposite`
670 string,`HouseNumber` string,`StreetName` string,`IntersectingStreet` string,
`DateFirstObserved` int,`LawSection` int,`SubDivision` string,`ViolationLegalCode`
string,`DaysParkingInEffect` string,
671 `FromHoursInEffect` string,`ToHoursInEffect` string,`VehicleColor`
string,`UnregisteredVehicle` string,`VehicleYear` int,`MeterNumber` string,
672 `FeetFromCurb` int,`ViolationPostCode` string,`ViolationDescription` string,
673 `NoStandingOrStoppingViolation` string,`HydrantViolation`
string,`DoubleParkingViolation` string
674 )
675 PARTITIONED BY (Season string)
676 ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
677 LOCATION 's3a://dee-hive-test/hiveAssignment/'
678 tblproperties ("skip.header.line.count"="1");
679
680
681
682 #Inserting data into table
683 #unix_timestamp(IssueDate,'MM/dd/yyyy') => Converts time string in format MM/dd/yyyy
to Unix time stamp
684 #from_unixtime() => Converts the number of seconds from unix epoch (1970-01-01
00:00:00 UTC) to a string representing the
685 #Creating a range of partitions based on month extracted from IssueDate field value.
686 #to_date() method casts a date into Hive date format of yyyy-MM-dd format.
687 #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
688 #Filtering out those records which do belong to year 2017.
689
690
691
692 insert overwrite table nycparking.parking_ext_part_season partition(Season)
693 select SummonsNumber , PlateID, RegistrationState, PlateType
,to_date(from_unixtime(UNIX_TIMESTAMP(IssueDate,'MM/dd/yyyy'))),ViolationCode ,
694 VehicleBodyType, VehicleMake ,IssuingAgency ,StreetCode1, StreetCode2, StreetCode3 ,
VehicleExpirationDate ,
695 ViolationLocation,ViolationPrecinct,IssuerPrecinct,IssuerCode,IssuerCommand,IssuerSqua
d,
696 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 ,
697 case
698     when month(to_date(from_unixtime(UNIX_TIMESTAMP(IssueDate,'MM/dd/yyyy'))))
BETWEEN 3 AND 5 THEN 'Spring'
699     when month(to_date(from_unixtime(UNIX_TIMESTAMP(IssueDate,'MM/dd/yyyy'))))
BETWEEN 6 AND 8 THEN 'Summer'
700     when month(to_date(from_unixtime(UNIX_TIMESTAMP(IssueDate,'MM/dd/yyyy'))))
BETWEEN 9 AND 11 THEN 'Fall'
701     when month(to_date(from_unixtime(UNIX_TIMESTAMP(IssueDate,'MM/dd/yyyy')))) == 12
OR
702     month(to_date(from_unixtime(UNIX_TIMESTAMP(IssueDate,'MM/dd/yyyy')))) == 1 OR
703     month(to_date(from_unixtime(UNIX_TIMESTAMP(IssueDate,'MM/dd/yyyy')))) == 2 THEN
'Winter'
704     else 0
705 end as Season from nycparking.parking_ext where INSTR(IssueDate , '2017') > 0;
706
707
708
709 # Checking how many total records are there in this partition table
710
711 Query:
712
713 select count(*)
714 from nycparking.parking_ext_part_season;
715 Ans. 5431878

```

```

716
717
718 8. (1) find frequencies of tickets for each season.
719
720 #No of tickets issued in Fall Season:
721 #Finding total tickets based on partition = Fall
722
723 Query:
724
725 select count(DISTINCT summonsnumber) as tickets_fall
726 from nycparking.parking_ext_part_season
727 where season = 'Fall';
728
729 Output:
730
731 tickets_fall
732 978
733
734
735 #No of tickets issued in Spring Season:
736 #Finding total tickets based on partition = Spring
737
738 Query:
739
740 select count(DISTINCT summonsnumber) as tickets_spring
741 from nycparking.parking_ext_part_season where season = 'Spring';
742
743
744 Output:
745
746 tickets_spring
747 2873372
748
749
750 #No of tickets issued in Summer Season:
751 #Finding total tickets based on partition = Summer
752
753
754 Query:
755
756 select count(DISTINCT summonsnumber) as tickets_summer
757 from nycparking.parking_ext_part_season where season = 'Summer';
758
759 Output:
760
761 tickets_summer
762 852856
763
764
765 #No of tickets issued in Winter Season:
766 #Finding total tickets based on partition = Winter
767
768
769 Query:
770
771 select count(DISTINCT summonsnumber) as tickets_winter
772 from nycparking.parking_ext_part_season where season = 'Winter';
773
774 Output:
775
776 tickets_winter
777 1704672
778
779 8. (2) find the 3 most common violations for each of these seasons
780
781 #Finding the most common violation codes against which tickets are issued in
partition Fall
782 #I am ordering the set into descending order of occurrence(count) and then picking
top 3 by using -> LIMIT 3
783
784 Query:
785
786 select violationcode , count(*) as frequency

```

```

787 from nycparking.parking_ext_part_season
788 where season = 'Fall'
789 group by violationcode order by frequency DESC limit 3;
790
791 Output:
792
793 violationcode    frequency
794 46    230
795 21    128
796 40    116
797
798 #Finding the most common violation codes against which tickets are issued in
partition Spring
799 #I am ordering the set into descending order of occurrence(count) and then picking
top 3 by using -> LIMIT 3
800
801 Query:
802
803 select violationcode , count(*) as frequency
804 from nycparking.parking_ext_part_season
805 where season = 'Spring'
806 group by violationcode order by frequency DESC limit 3;
807
808
809 Output:
810
811 violationcode    frequency
812 21    402423
813 36    344834
814 38    271167
815
816
817 #Finding the most common violation codes against which tickets are issued in
partition Summer
818 #I am ordering the set into descending order of occurrence(count) and then picking
top 3 by using -> LIMIT 3
819
820 Query:
821
822 select violationcode , count(*) as frequency
823 from nycparking.parking_ext_part_season
824 where season = 'Summer'
825 group by violationcode order by frequency DESC limit 3;
826
827 Output:
828
829 violationcode    frequency
830 21    127349
831 36    96663
832 38    83517
833
834
835 #Finding the most common violation codes against which tickets are issued in
partition Winter
836 #I am ordering the set into descending order of occurrence(count) and then picking
top 3 by using -> LIMIT 3
837
838 Query:
839
840 select violationcode , count(*) as frequency
841 from nycparking.parking_ext_part_season
842 where season = 'Winter'
843 group by violationcode order by frequency DESC limit 3;
844
845 Output:
846
847 violationcode    frequency
848 21    238178
849 36    221267
850 38    187386
851
852
853 #####

```

```
#####
854 #####                               End of Assignment
Questions #####
855
856 Best Practices:
857
858 1. You should now drop the tables and database if you do not need it any further.
859 2. Use DROP Table and then DROP database commands to drop the non-required items.
860 3. You should now STOP your EC2 instance and come out of the EC2 terminal.
861 4. You should now delete the buckets created on S3 if you do not need those any
    further.
862
863
864 #####                               End of
Document #####
```