Objective:

The assignment is meant for you to apply learnings of the module on Hive on a real-life dataset. One of the major objectives of this assignment is gaining familiarity with how an analysis works in Hive and how you can gain insights from large datasets.

Problem statement:

New York City is a thriving metropolis and just like most other cities of similar size, one of the biggest problems its residents face is parking. The classic combination of a huge number of cars and a cramped geography is the exact recipe that leads to a large number of parking tickets.

- **first create an external csv table which points to the data**

```
create external table park_vio_ext_csv
(
SUMMONS_NUMBER int,
PLATE_ID string,
REGISTRATION_STATE string,
PLATE_TYPE string,
ISSUE_DATE string,
VIOLATION_CODE int,
VEHICLE_BODY_TYPE string,
VEHICLE_MAKE string,
ISSUING_AGENCY string,
STREET_CODE_1 int,
STREET_CODE_2 int,
STREET_CODE_3 int,
VEHICLE_EXPIRATION_DATE int,
VIOLATION_LOCATION string,
VIOLATION_PRECINCT int,
ISSUER_PRECINCT int,
ISSUER_CODE int,
```

```
ISSUER_COMMAND string,
ISSUER_SQUAD string,
VIOLATION_TIME string,
TIME_FIRST_OBSERVED string,
VIOLATION_COUNTRY string,
VIOLATION_INFRONT_OF_OR_OPPOSITE string,
HOUSE_NUMBER string,
STREET_NAME string,
INTERSECTING_STREET string,
DATE_FIRST_OBSERVED int,
LAW_SECTION int,
SUB_DIVISION string,
VIOLATION_LEGAL_CODE string,
DAYS_PARKING_IN_EFFECT string,
FROM_HOURS_IN_EFFECT string,
TO_HOURS_IN_EFFECT string,
VEHICLE_COLOR string,
UNREGISTERED_VEHICLE string,
VEHICLE_YEAR int,
METER_NUMBER string,
FEET_FROM_CURB int,
VIOLATION_POST_CODE string,
VIOLATION_DESCRIPTION string,
NO_STANDING_OR_STOPPING_VIOLATION string,
HYDRANT_VIOLATION string,
DOUBLE_PARKING_VIOLATION string
)
row format delimited
fields terminated by ','
tblproperties("skip.header.line.count"="1");
```

- **Load data into park_vio_ext_csv table**

```
load data local inpath 'file:///home/cloudera/Desktop/csv_files/Parking_Violations_Issued_-_Fiscal_Year_2017.csv' into table parking_vio_ext_csv;
```

- **ORC is the optimized file format used in Hive for better analysis.**
- **So, create an external orc table to load the csv data into it.**
- **And also perform partitioning and bucketing on top of it.**

```
create external table park_vio_ext_orc
(
SUMMONS_NUMBER int,
PLATE_ID string,
REGISTRATION_STATE string,
PLATE_TYPE string,
ISSUE_DATE string,
VIOLATION_CODE int,
VEHICLE_BODY_TYPE string,
VEHICLE_MAKE string,
ISSUING_AGENCY string,
STREET_CODE_1 int,
STREET_CODE_2 int,
STREET_CODE_3 int,
VEHICLE_EXPIRATION_DATE int,
VIOLATION_LOCATION string,
VIOLATION_PRECINCT int,
ISSUER_PRECINCT int,
ISSUER_CODE int,
ISSUER_COMMAND string,
ISSUER_SQUAD string,
VIOLATION_TIME string,
TIME_FIRST_OBSERVED string,
VIOLATION_INFRONT_OF_OR_OPPOSITE string,
HOUSE_NUMBER string,
```

```
26  STREET_NAME string,
27  INTERSECTING_STREET string,
28  DATE_FIRST_OBSERVED int,
29  LAW_SECTION int,
30  SUB_DIVISION string,
31  VIOLATION_LEGAL_CODE string,
32  DAYS_PARKING_IN_EFFECT string,
33  FROM_HOURS_IN_EFFECT string,
34  TO_HOURS_IN_EFFECT string,
35  VEHICLE_COLOR string,
36  UNREGISTERED_VEHICLE string,
37  VEHICLE_YEAR int,
38  METER_NUMBER string,
39  FEET_FROM_CURB int,
40  VIOLATION_POST_CODE string,
41  VIOLATION_DESCRIPTION string,
42  NO_STANDING_OR_STOPPING_VIOLATION string,
43  HYDRANT_VIOLATION string,
44  DOUBLE_PARKING_VIOLATION string
45  )
46  partitioned by (violation_country string)
47  clustered by (violation_code)
48  sorted by (violation_code)
49  into 5 buckets
50  stored as orc;
```

- **set some hive properties for dynamic_partition and bucketing**

```
1  set hive.exec.dynamic.partition=true;
2  set hive.exec.dynamic.partition.mode=nonstrict;
3  set hive.enforce.bucketing=true;
```

- **overwrite the orc table with csv table**
- **Filter the data of year 2017.**

```
insert into park_vio_ext_orc partition(violation_country)
select
SUMMONS_NUMBER,
PLATE_ID ,
REGISTRATION_STATE,
PLATE_TYPE,
ISSUE_DATE ,
VIOLATION_CODE,
VEHICLE_BODY_TYPE,
VEHICLE_MAKE,
ISSUING_AGENCY,
STREET_CODE_1,
STREET_CODE_2 ,
STREET_CODE_3,
VEHICLE_EXPIRATION_DATE,
VIOLATION_LOCATION,
VIOLATION_PRECINCT,
ISSUER_PRECINCT,
ISSUER_CODE,
ISSUER_COMMAND,
ISSUER_SQUAD,
VIOLATION_TIME,
TIME_FIRST_OBSERVED,
VIOLATION_INFRONT_OF_OR_OPPOSITE,
HOUSE_NUMBER,
STREET_NAME,
INTERSECTING_STREET,
DATE_FIRST_OBSERVED,
LAW_SECTION,
SUB_DIVISION,
VIOLATION_LEGAL_CODE,
DAYS_PARKING_IN_EFFECT,
FROM_HOURS_IN_EFFECT,
TO_HOURS_IN_EFFECT,
VEHICLE_COLOR,
UNREGISTERED_VEHICLE,
VEHICLE_YEAR,
METER_NUMBER,
FEET_FROM_CURB,
VIOLATION_POST_CODE,
VIOLATION_DESCRIPTION,
NO_STANDING_OR_STOPPING_VIOLATION,
HYDRANT_VIOLATION,
DOUBLE_PARKING_VIOLATION,
VIOLATION_COUNTRY
from park_vio_ext_csv
where issue_date like '%2017';
```

Part-I: Examine the data

1.)

```
1
2
3
```

```
/* total tickets */

select count(*) as total_tickets from park_viol_ext_orc;
```

2.)

```
1
2
```

```
/* unique registration states */
select count(distinct(registration_state)) as number_of_states from
park_vio_ext_orc;
```

3.)

```
1
2
3
4
```

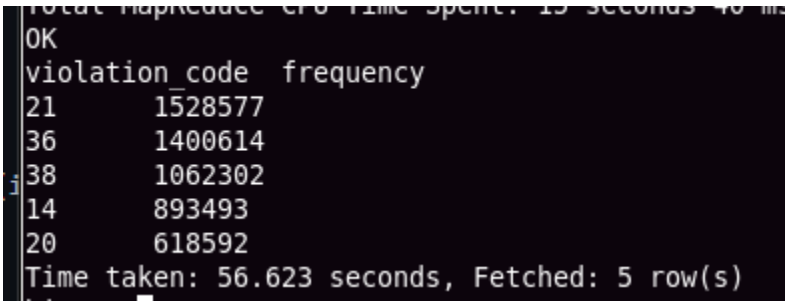
```
/* parking tickets don't have addresses on them */

select count(*) as tickets_not_having_address from park_viol_ext_orc
where street_code_1 = 0 or street_code_2 = 0 or street_code_3 = 0;
```

Part-II: Aggregation tasks

1.)

- **frequency of violation codes - find the top 5**

```
1  /* group the data by  violation code to find the frequency of each code
2  */
3
4  select violation_code , count(*) as frequency from park_vio_ext_orc
5  group by violation_code
6  sort by frequency desc
7  limit 5;
```



2.)

a.

- **Frequencies of vehicle body top 5**

```
1  /*  group the data by vehicle body to find the frequency */
2
3  select vehicle_body_type, count(*) as ticket_count
4  from park_vio_ext_orc
5  group by vehicle_body_type
6  sort by ticket_count desc
7  limit 5;
```

```
OK
vehicle_body_type       ticket_count
SUBN    3719796
4DSD    3082006
VAN     1411964
DELV    687324
SDN     438191
Time taken: 54.299 seconds, Fetched: 5 row(s)
```

b.

- **Frequencies of vehicle make top 5**

```
1  /* group the data by vehicle body to find the frequency */
2
3  select vehicle_make, count(*) as ticket_count
4  from park_vio_ext_orc
5  group by vehicle_make
6  sort by ticket_count desc
7  limit 5;
```

```
Total MapReduce CPU Time Spent: 16 seconds 800
OK
vehicle_make    ticket_count
FORD    1280956
TOYOT   1211447
HONDA   1079237
NISSA   918590
CHEVR   714654
Time taken: 53.409 seconds, Fetched: 5 row(s)
```

3.)

a.

➢ **Frequencies of Violation precincts**

```
1  /* group the data by violation precinct and find frequencies of each */
2
3  select violation_precinct, count(*) as frequency
4  from park_vio_ext_orc
5  group by violation_precinct
6  sort by frequency desc
7  limit 5;
```

```
OK
violation_precinct      frequency
0       2072400
19      535671
14      352450
1       331810
18      306920
Time taken: 55.659 seconds, Fetched: 5 row(s)
```

b.

➢ **Frequencies of issuer precincts**

```
1  /* group by issuer precinct and find the frequencies of each */
2
3  select issuer_precinct, count(*) as frequency
4  from park_vio_ext_orc
5  group by issuer_precinct
6  sort by frequency desc
7  limit 5;
```

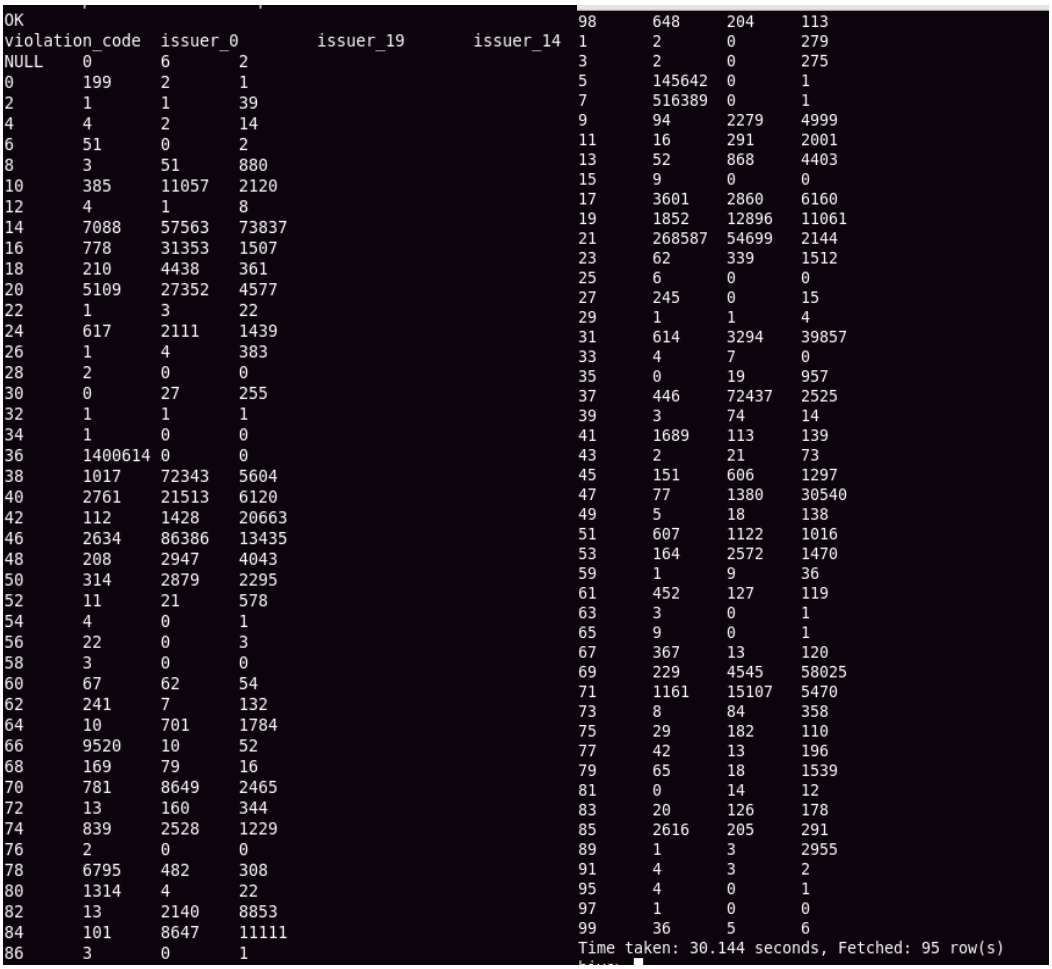
```
OK
issuer_precinct frequency
0       2388475
19      521513
14      344977
1       321170
18      296554
```

4.)

- **Find the violation code frequency across 3 precincts which have issued the most number of tickets**

```
1   /* find the top 3 issuer_precinct */
2
3   select issuer_precinct,count(*) as frequency
4   from park_vio_ext_orc
5   group by issuer_precinct
6   sort by frequency desc
7   limit 3;
```

```
 1   /* from the above queryissuer_precincts are 0, 19,14. Based on the
 2   result perform  group by violation_code  */
 3
 4   select violation_code,
 5   sum(case when issuer_precinct = 0 then 1 else 0 end) as issuer_0,
 6   sum(case when issuer_precinct = 19 then 1 else 0 end) as issuer_19,
 7   sum(case when issuer_precinct = 14 then 1 else 0 end) as issuer_14
 8   from park_vio_ext_orc a
 9   where a.issuer_precinct in (0,19,14)
10   group by violation_code;
```



```
OK
violation_code  issuer_0        issuer_19       issuer_14
NULL    0       6       2
0       199     2       1
2       1       1       39
4       4       2       14
6       51      0       2
8       3       51      880
10      385     11057   2120
12      4       1       8
14      7088    57563   73837
16      778     31353   1507
18      210     4438    361
20      5109    27352   4577
22      1       3       22
24      617     2111    1439
26      1       4       383
28      2       0       0
30      0       27      255
32      1       1       1
34      1       0       0
36      1400614 0       0
38      1017    72343   5604
40      2761    21513   6120
42      112     1428    20663
46      2634    86386   13435
48      208     2947    4043
50      314     2879    2295
52      11      21      578
54      4       0       1
56      22      0       3
58      3       0       0
60      67      62      54
62      241     7       132
64      10      701     1784
66      9520    10      52
68      169     79      16
70      781     8649    2465
72      13      160     344
74      839     2528    1229
76      2       0       0
78      6795    482     308
80      1314    4       22
82      13      2140    8853
84      101     8647    11111
86      3       0       1
98      648     204     113
1       2       0       279
3       2       0       275
5       145642  0       1
7       516389  0       1
9       94      2279    4999
11      16      291     2001
13      52      868     4403
15      9       0       0
17      3601    2860    6160
19      1852    12896   11061
21      268587  54699   2144
23      62      339     1512
25      6       0       0
27      245     0       15
29      1       1       4
31      614     3294    39857
33      4       7       0
35      0       19      957
37      446     72437   2525
39      3       74      14
41      1689    113     139
43      2       21      73
45      151     606     1297
47      77      1380    30540
49      5       18      138
51      607     1122    1016
53      164     2572    1470
59      1       9       36
61      452     127     119
63      3       0       1
65      9       0       1
67      367     13      120
69      229     4545    58025
71      1161    15107   5470
73      8       84      358
75      29      182     110
77      42      13      196
79      65      18      1539
81      0       14      12
83      20      126     178
85      2616    205     291
89      1       3       2955
91      4       3       2
95      4       0       1
97      1       0       0
99      36      5       6
Time taken: 30.144 seconds, Fetched: 95 row(s)
```

5.)

- **Divide time based on AM and PM**
- **Find the frequencies for them.**

```
1   /*
2   1.) filter data where violation_time has A or P at the end.
3   2.) And consider the data where hour in violation_time is from 1 - 12
4   3.) Append the date time and convert them into unix_timestamp.
5   4.) Now perform group by on violation_code which results the count of
6   each violation_code in AM and PM
7    */
8
9   with data_2017 as
10  (
11  select registration_state, issue_date,
12  violation_code, violation_precinct,
13  issuer_precinct, violation_time, violation_country
14  from park_vio_ext_orc
15  where (violation_time like '%P' or violation_time like '%A') and
16  (cast(substring(violation_time,1,2) as int) > 0 and
```

```
cast(substring(violation_time,1,2) as int) <=12)
),
change_time as
(
select registration_state,issue_date,
violation_code, violation_precinct, issuer_precinct,
concat(substring(violation_time,1,2),':',substring(violation_time,3,2)
,':00 ',substring(violation_time,5,1),'M') as violation_time,
violation_country from data_2017
)
select violation_code,
sum(case when violation_time like '%AM' then 1 else 0 end) as AM,
sum(case when violation_time like '%PM' then 1 else 0 end) as PM
from change_date
group by violation_code;
```

```
Total MapReduce CPU Time Spent: 44 seconds 0
OK
violation_code  am      pm
0       199     28
2       72      5
4       440     81
6       170     4
8       1318    83
10      23192   2651
12      51      2
14      440657  31952
16      68748   5951
18      9601    584
20      293917  23555
22      76      5
24      35892   2483
26      633     27
28      5       0
30      516     37
32      13      1
34      11      0
36      560774  101991
38      486033  56040
40      255463  18252
42      27351   4657
44      4       0
46      278826  32467
48      36642   4330
50      49525   3889
52      946     55
54      3       0
56      333     27
58      12      1
60      3505    168
62      2548    229
64      5935    810
66      12402   480
68      22764   2190
70      130605  13630
72      5219    274
74      54563   3968
76      16      2
78      23307   1668
80      1884    194
82      15634   1546
```

6.)

- **create external table to store only required fileds from main data after applying all the filtering.**

```
create external table filtered_data_orc
(
registration_state string,
date_time string,
violation_code int,
violation_precinct int,
issuer_precinct int,
violation_country string
)
stored as orc;
```

- **The below code is written with the help of common table expressions.**
- **1.) filter the data**
- **2.) convert the date in string format to unix_timestamp and time to hh:mm:ss format**
- **3.) combine the time and date and convert both to unix_time**
- **4.) insert the final modified data into filtered_data_orc table**

```
with data_2017 as
(
select registration_state, issue_date, violation_code, violation_precinct,
issuer_precinct, violation_time, violation_country
from park_vio_ext_orc
where (violation_time like '%P' or violation_time like '%A') and
(cast(substring(violation_time,1,2) as int) > 0 and
cast(substring(violation_time,1,2) as int) <=12)
),
change_date as
(
select registration_state,
from_unixtime(unix_timestamp(issue_date,'MM/dd/yyyy'),'yyyy-MM-dd') as
date,
violation_code, violation_precinct, issuer_precinct,
concat(substring(violation_time,1,2),':',substring(violation_time,3,2),':00
',substring(violation_time,5,1),'M') as violation_time,
violation_country
from data_2017
),
combine_date_time as
(
select  registration_state,
```

```
from_unixtime(unix_timestamp(concat(date,violation_time),'yyyy-MM-
ddhh:mm:ss a'),'yyyy-MM-dd hh:mm:ss a') as date_time,
violation_code, violation_precinct, i
ssuer_precinct,violation_country
from change_date
)
from combine_date_time
insert overwrite table filtered_data_orc select *;
```

- **Now from the filtered_data_orc table group the hours into six groups.**
- **Find the frequency of each violation_code in that particular interval group**

```
with time_groups as
(
select violation_code ,
sum(case when (date_time like '%AM' and hour(date_time) in (12,1,2,3)) then 1 else
0 end) as 12_AM_to_4_AM,
sum(case when (date_time like '%AM' and hour(date_time) in (4,5,6,7)) then 1 else 0
end) as 4_AM_to_8_AM,
sum(case when (date_time like '%AM' and hour(date_time) in (8,9,10,11)) then 1 else
0 end) as 8_AM_to_12_PM,
sum(case when (date_time like '%PM' and hour(date_time) in (12,1,2,3)) then 1 else
0 end) as 12_PM_to_4_PM,
sum(case when (date_time like '%PM' and hour(date_time) in (4,5,6,7)) then 1 else 0
end) as 4_PM_to_8_PM,
sum(case when (date_time like '%PM' and hour(date_time) in (8,9,10,11)) then 1 else
0 end) as 8_PM_to_12_PM
from filtered_data_orc
group by violation_code
)
select violation_code, 12_AM_to_4_AM
from time_groups
sort by 12_AM_to_4_AM desc
limit 3;
```

- **Similarly for all the time intervals**

```
1   /* similarly for all other groups */
 2
 3   select violation_code, 4_AM_to_8_AM
 4   from time_groups
 5   sort by 4_AM_to_8_AM desc
 6   limit 3;
 7
 8
 9   select violation_code, 8_AM_to_12_PM
10   from time_groups
11   sort by 8_AM_to_12_PM  desc
```

```
12    limit 3;
13
14
15    select violation_code, 12_PM_to_4_PM
16    from time_groups
17    sort by 12_PM_to_4_PM  desc
18    limit 3;
19
20
21    select violation_code, 4_PM_to_8_PM
22    from time_groups
23    sort by 4_PM_to_8_PM desc
24    limit 3;
25
26
27    select violation_code, 8_PM_to_12_PM
28    from time_groups
29    sort by 8_PM_to_12_PM desc
30    limit 3;
```

```
OK
violation_code  12_am_to_4_am
21      26444
40      22420
78      13737
```

7.)

- **first find the 3 most commonly occurring violation codes**

```
1   select  violation_code, count(*) as frequency
2   from filtered_data_orc
3   group by violation_code
4   sort by frequency desc
5   limit 3;
```

- **So, the top 3 violation_codes are 21, 36, 38**
- **And the frequencies of these top 3 violation_code in all the intervals**

```
select violation_code,
sum(case when (date_time like '%AM' and hour(date_time) in (12,1,2,3)) then 1 else
0 end) as 12_AM_to_4_AM,
sum(case when (date_time like '%AM' and hour(date_time) in (4,5,6,7)) then 1 else 0
end) as 4_AM_to_8_AM,
sum(case when (date_time like '%AM' and hour(date_time) in (8,9,10,11)) then 1 else
0 end) as 8_AM_to_12_PM,
sum(case when (date_time like '%PM' and hour(date_time) in (12,1,2,3)) then 1 else
```

```
0 end) as 12_PM_to_4_PM,
sum(case when (date_time like '%PM' and hour(date_time) in (4,5,6,7)) then 1 else 0
end) as 4_PM_to_8_PM,
sum(case when (date_time like '%PM' and hour(date_time) in (8,9,10,11)) then 1 else
0 end) as 8_PM_to_12_PM
from filtered_data_orc
where violation_code in (21,36,38)
group by violation_code;
```

```
OK
violation_code  12_am_to_4_am   4_am_to_8_am    8_am_to_12_pm   12_pm_to_4_pm   4_pm_to_8_pm    8_pm_to_12_pm
21      26444   57893   598053  74691   259     184
36      0       14782   348165  286284  13534   0
38      307     1273    176570  240721  102855  20347
Time taken: 29.602 seconds, Fetched: 3 row(s)
```

```
OK
violation_code  frequency
21      757535
36      662765
38      542073
Time taken: 62.221 seconds, Fetched: 3 row(s)
```

8)

a.)

- **frequency of tickets for each of the seasons**

```
select
sum(case when month(date_time) in (3,4,5) then 1 else 0 end) as spring,
sum(case when month(date_time) in (6,7,8) then 1 else 0 end) as summer,
sum(case when month(date_time) in (9,10,11) then 1 else 0 end) as fall,
sum(case when month(date_time) in (12,1,2) then 1 else 0 end) as winter
from filtered_data_orc;
```

b.)

- **3 most common violations in these seasons**

```
with cte as
(
select violation_code,
sum(case when month(date_time) in (3,4,5) then 1 else 0 end) as
spring,
sum(case when month(date_time) in (6,7,8) then 1 else 0 end) as
summer,
sum(case when month(date_time) in (9,10,11) then 1 else 0 end) as
fall,
```

```
sum(case when month(date_time) in (12,1,2) then 1 else 0 end) as
winter
from filtered_data_orc
group by violation_code
)
/* This selects for spring season */
select violation_code,spring
from cte
order by spring desc
limit 3;
```

- **Similarly for all the seasons**

```
1   /*  similarly for all the seasons */
 2
 3   select violation_code,summer
 4   from cte
 5   order by summer desc
 6   limit 3;
 7
 8   select violation_code,fall
 9   from cte
10   order by fall desc
11   limit 3;
12
13
14   select violation_code,winter
15   from cte
16   order by winter desc
17   limit 3;
```

```
OK
spring  summer  fall    winter
2858089 848432  970     1695713
```