Hive Mini Project-2

Creating data schema (SAMPLE):

```
create table Parking_Violation
(
Summons_Number int,
Plate_ID varchar(10),
Registration_State varchar(2),
Issue_Date DATE ‘dd/mm/yyyy’,
.
.
.
.
.
PRIMARY KEY (Summons_Number)
)
row format delimited
field terminated by ‘,’ ;
```

Putting the data inside the table from HDFS .

```
LOAD DATA INPATH ‘file:///path/to/Parking_Violation.csv’ INTO TABLE
Parking_Violation;
```

Part-I: Examine the data

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

```
select count(*) from Parking_Violation
order by year(Issue_Date) desc;
```

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

```
select count(distinct Registration_state) from Parking_Violation;
```

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

```
select count(
case
WHEN street_code_1 IS NULL OR street_code_2 IS NULL OR street_code_3 IS
NULL
THEN 1
END) FROM Parking_Violation;
```

Part-II: Aggregation tasks

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

```
select count(distinct violation_code) as total_violations from Parking_Violation
order by total_violations desc
limit 5;
```

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

```
select Violation_body_type, count (distinct Violation_body_type) as
total_violations_per_body_type, Vehicle_make
from Parking_Violation
group by Vehicle_make
order by total_violations_per_body_type desc
limit 5;
```

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

    a.) Violating Precincts (this is the precinct of the zone where the violation occurred)
    b.) Issuer Precincts (this is the precinct that issued the ticket)

```
a —> select count(distinct Violating_Precincts) from Parking_Violation
       order by desc
       limit 5;

b —> select count(distinct Issuer Precincts) from Parking_Violation
        order by desc
        limit 5;
```

4.) Find the violation code frequency across 3 precincts which have issued the most number of tickets - do these precinct zones have an exceptionally high frequency of certain violation codes?

```
select Violation_Precinct, count(distinct Violation_Precinct) from
Parking_Violation
```

```
where Violation_Precinct>0
group by year, Violation_Precinct
order by 2 desc
limit 5;
```

5.) Find out the properties of parking violations across different times of the day: The Violation Time field is specified in a strange format. Find a way to make this into a time attribute that you can use to divide into groups.

```
UPDATE Parking_Violation
SET Violation_Hour = CAST(CONCAT(SUBSTR(Violation_Time, 1, 2), ':00:00') AS
TIMESTAMP);
```

6.) Divide 24 hours into 6 equal discrete bins of time. The intervals you choose are at your discretion. For each of these groups, find the 3 most commonly occurring violations

```
SELECT TimeGroup, Violation, COUNT(*) AS ViolationCount
FROM (
  SELECT
    CASE
      WHEN hour(Violation_Hour) >= 0 AND hour(Violation_Hour) < 4 THEN '00:00 -
03:59'
      WHEN hour(Violation_Hour) >= 4 AND hour(Violation_Hour) < 8 THEN '04:00 -
07:59'
      WHEN hour(Violation_Hour) >= 8 AND hour(Violation_Hour) < 12 THEN '08:00
- 11:59'
      WHEN hour(Violation_Hour) >= 12 AND hour(Violation_Hour) < 16 THEN '12:00
- 15:59'
      WHEN hour(Violation_Hour) >= 16 AND hour(Violation_Hour) < 20 THEN
'16:00 - 19:59'
      WHEN hour(Violation_Hour) >= 20 AND hour(Violation_Hour) < 24 THEN
'20:00 - 23:59'
    END AS TimeGroup,
    Violation
  FROM Parking_Violation
) AS grouped
GROUP BY TimeGroup, Violation
ORDER BY TimeGroup, ViolationCount DESC
```

7.) Now, try another direction. For the 3 most commonly occurring violation codes, find the most common times of day (in terms of the bins from the previous part)

```
WITH TopViolations AS (
  SELECT Violation, COUNT(*) AS ViolationCount
  FROM Parking_Violation
  GROUP BY Violation
  ORDER BY ViolationCount DESC
  LIMIT 3
)
SELECT tv.Violation, TimeGroup, COUNT(*) AS TimeGroupCount
FROM (
  SELECT
    CASE
      WHEN hour(Violation_Hour) >= 0 AND hour(Violation_Hour) < 4 THEN '00:00 -
03:59'
      WHEN hour(Violation_Hour) >= 4 AND hour(Violation_Hour) < 8 THEN '04:00 -
07:59'
      WHEN hour(Violation_Hour) >= 8 AND hour(Violation_Hour) < 12 THEN '08:00
- 11:59'
      WHEN hour(Violation_Hour) >= 12 AND hour(Violation_Hour) < 16 THEN '12:00
- 15:59'
      WHEN hour(Violation_Hour) >= 16 AND hour(Violation_Hour) < 20 THEN
'16:00 - 19:59'
      WHEN hour(Violation_Hour) >= 20 AND hour(Violation_Hour) < 24 THEN
'20:00 - 23:59'
    END AS TimeGroup,
    Violation
  FROM Parking_Violation
) AS grouped
JOIN TopViolations tv ON grouped.Violation = tv.Violation
GROUP BY tv.Violation, TimeGroup
ORDER BY tv.Violation, TimeGroupCount DESC;
```

8.) Let’s try and find some seasonality in this data

a.) First, divide the year into some number of seasons, and find frequencies of tickets for each season. (Hint: A quick Google search reveals the following seasons in NYC: Spring(March, April, March); Summer(June, July, August); Fall(September, October, November); Winter(December, January, February))

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

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

```
WITH SeasonTickets AS (
  SELECT
    CASE
      WHEN month(Issue_Date) IN (3, 4, 5) THEN 'Spring'
      WHEN month(Issue_Date) IN (6, 7, 8) THEN 'Summer'
      WHEN month(Issue_Date) IN (9, 10, 11) THEN 'Fall'
      WHEN month(Issue_Date) IN (12, 1, 2) THEN 'Winter'
    END AS Season,
    Violation
  FROM Parking_Violation
)
SELECT
  Season,
  Violation,
  COUNT(*) AS ViolationCount
FROM SeasonTickets
GROUP BY Season, Violation
ORDER BY Season, ViolationCount DESC
;

WITH SeasonTopViolations AS (
  SELECT
    Season,
    Violation,
    COUNT(*) AS ViolationCount,
    ROW_NUMBER() OVER (PARTITION BY Season ORDER BY COUNT(*) DESC) AS
rn
  FROM SeasonTickets
  GROUP BY Season, Violation
)
SELECT
  Season,
  Violation,
  ViolationCount
FROM SeasonTopViolations
WHERE rn <= 3
ORDER BY Season, ViolationCount DESC;
```