Project soln 2

---

prashanth

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.

In an attempt to scientifically analyse this phenomenon, the NYC Police Department regularly collects data related to parking tickets. This data is made available by NYC Open Data portal. We will try and perform some analysis on this data.

Download Dataset - https://data.cityofnewyork.us/browse?q=parking+tickets

Note: Consider only the year 2017 for analysis and not the Fiscal year.

The analysis can be divided into two parts:

Part-I: Examine the data

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

```
SELECT COUNT(tickets)Parking_Violations_Issued ;
```

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

```
SELECT COUNT(DISTINCT Registration_State)FROM Parking_Violations_Issued;
```

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(*) FROM Parking_Violations_Issued WHERE `Street Code 1` = '' OR `Street Code 2` = '' OR `Street Code 3` = '';
```

Part-II: Aggregation tasks

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

```
SELECT `Violation Code`, COUNT(*) AS frequency FROM Parking_Violations_Issued  GROUP BY `Violation Code`  ORDER BY frequency 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 `Vehicle Body Type`, COUNT(*) AS frequency  FROM Parking_Violations_Issued  GROUP BY `Vehicle Body Type`  ORDER BY frequency 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)

```
SELECT `Violating Precinct`, COUNT(*) AS frequency FROM Parking_Violations_Issued GROUP BY `Violating Precinct` ORDER BY frequency DESCLIMIT 5;
```

b.) Issuer Precincts (this is the precinct that issued the ticket)

```
SELECT `Issuer Precinct`, COUNT(*) AS frequency FROM Parking_Violations_Issued GROUP BY `Issuer Precinct`ORDER BY frequency 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 Code`, `Issuer Precinct`, COUNT(*) AS frequency

FROM Parking_Violations_Issued

WHERE `Issuer Precinct` IN (

  SELECT `Issuer Precinct`

  FROM Parking_Violations_Issued

  GROUP BY `Issuer Precinct`
```

```
    ORDER BY COUNT(*) DESC
    LIMIT 3
)
GROUP BY `Violation Code`, `Issuer Precinct`
ORDER BY `Issuer Precinct`, frequency DESC;
```

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.

```
SELECT `Violation Code`,
    DATE_FORMAT(FROM_UNIXTIME(`Issue Date`), '%H') AS `Hour of Day`,
    COUNT(*) AS frequency
FROM Parking_Violations_Issued
GROUP BY `Violation Code`, `Hour of Day`
ORDER BY `Violation Code`, frequency DESC;
```

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 `Violation Code`,
    FLOOR(DATE_FORMAT(FROM_UNIXTIME(`Issue Date`), '%H') / 4) AS `Time Bin`,
    COUNT(*) AS frequency
FROM Parking_Violations_Issued
GROUP BY `Violation Code`, `Time Bin`
ORDER BY `Violation Code`, frequency 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 top_violations AS (
    SELECT `Violation Code`, COUNT(*) AS frequency
    FROM Parking_Violations_Issued
```

```
   GROUP BY `Violation Code`
   ORDER BY frequency DESC
   LIMIT 3
)
SELECT `Time Bin`, `Violation Code`, COUNT(*) AS frequency
FROM Parking_Violations_Issued
WHERE `Violation Code` IN (
   SELECT `Violation Code` FROM top_violations
)
GROUP BY `Time Bin`, `Violation Code`
ORDER BY `Time Bin`, frequency 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))

```
    SELECT
  CASE
   WHEN month(issue_date) BETWEEN 3 AND 5 THEN 'Spring'
   WHEN month(issue_date) BETWEEN 6 AND 8 THEN 'Summer'
   WHEN month(issue_date) BETWEEN 9 AND 11 THEN 'Fall'
   ELSE 'Winter'
  END AS season,
  count(*) AS frequency
FROM Parking_Violations_Issued
GROUP BY season;
```

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

```
        SELECT
  season,
  violation_code,
  frequency_rank
```

```
FROM (
  SELECT
    CASE
      WHEN month(issue_date) BETWEEN 3 AND 5 THEN 'Spring'
      WHEN month(issue_date) BETWEEN 6 AND 8 THEN 'Summer'
      WHEN month(issue_date) BETWEEN 9 AND 11 THEN 'Fall'
      ELSE 'Winter'
    END AS season,
    violation_code,
    COUNT(*) AS frequency,
    RANK() OVER (PARTITION BY season ORDER BY COUNT(*) DESC) AS frequency_rank
  FROM Parking_Violations_Issued
  GROUP BY season, violation_code
) t
WHERE frequency_rank <= 3;
```