**Problem Statement:**

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

In an attempt to scientifically analyze this phenomenon, the NYC Police Department regularly collects data related to parking tickets. This data is made available by the [NYC Open Data](https://data.cityofnewyork.us/browse?q=parking+tickets) portal. Your job is to try and perform some analysis on this data, in order to answer the questions that follow.

You need to copy this data to your own bucket and then run the required queries

**Consider only the data for the year 2017 itself for your analysis, not the current year.**

The data dictionary is available on [this page](https://data.cityofnewyork.us/City-Government/Parking-Violations-Issued-Fiscal-Year-2017/2bnn-yakx).

The analysis can be divided into two parts:

**Part-I: Examine the data**

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

select substring(issue\_date,7,4),count(\*) from parking group by substring(issue\_date,7,4) having

substring(issue\_date,7,4)='2017';

### Find out the total number of states to which the cars with tickets belong. The count of states is mandatory here, providing the exact list of states is optional.

select registration\_state,count(\*) from parking group by registration\_state;

1. Some parking tickets don’t have addresses on them, which is a cause for concern. Find out the number of such tickets, which have no addresses. (i.e. tickets where one of the Street Codes, i.e. "Street Code 1" or "Street Code 2" or "Street Code 3" is empty)

select count(\*) from parking where street\_code1 IS NULL or street\_code2 IS NULL or street\_code3 IS NULL;

**Part-II: Aggregation tasks**

1. What are the top 5 most frequently occurring violation codes? (Note that frequency means the number of occurrences over a time period. The list should be in descending order)

SELECT Violation\_Code, count,rank FROM (SELECT Violation\_Code, COUNT(\*) AS count ,RANK() OVER (ORDER BY COUNT(\*) DESC) AS Rank FROM parking GROUP BY Violation\_Cod

e ) AS RankedViolations WHERE Rank <= 5;

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

select vehicle\_body\_type,count(vehicle\_body\_type) as total from parking group by vehicle\_body\_type order by total desc limit 5;

1. A precinct is a police station that has a certain zone of the city under its command. You will find two further classifications of precincts:
2. Violating Precincts - These are precincts where the violations have occurred. select violation\_precinct, count(\*) from parking group by violation\_precinct
3. Issuer Precincts - These are precincts that issued the tickets.

select issuer\_precinct,count(\*) from parking group by issuer\_precinct;

1. Find the top 5 Violating Precincts and Issuer Precincts by frequency.

SELECT precinct\_type, precinct, ticket\_count

FROM (

SELECT violation\_precinct AS precinct\_type, violation\_precinct AS precinct, COUNT(\*) AS ticket\_count

FROM parking

GROUP BY violation\_precinct

ORDER BY ticket\_count DESC

LIMIT 5

) AS violating\_precincts

UNION ALL

SELECT precinct\_type, precinct, ticket\_count

FROM (

SELECT issuer\_precinct AS precinct\_type, issuer\_precinct AS precinct, COUNT(\*) AS ticket\_count

FROM parking

GROUP BY issuer\_precinct

ORDER BY ticket\_count DESC

LIMIT 5

) AS issuer\_precincts;

1. Find the violation code frequency across the top 3 precincts which have issued the highest number of tickets. Do these precinct zones have an exceptionally high frequency of certain violation codes? If yes, list them.

select violation\_code, count(violation\_code) from parking where issuer\_code in (select issuer\_code From parking group by issuer\_code

order by count(issuer\_code) desc limit 3) group by violation\_code order by count(violation\_code) desc limit 3;

1. Find out the frequency 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

CASE

WHEN violation\_time LIKE '%A' AND CAST(SUBSTR(violation\_time, 1, 2) AS INT) < 6 THEN 'Early Morning'

WHEN violation\_time LIKE '%A' THEN 'Morning'

WHEN violation\_time LIKE '%P' AND CAST(SUBSTR(violation\_time, 1, 2) AS INT) < 5 THEN 'Afternoon'

WHEN violation\_time LIKE '%P' AND CAST(SUBSTR(violation\_time, 1, 2) AS INT) < 9 THEN 'Evening'

ELSE 'Night'

END AS time\_of\_day,

COUNT(\*) AS violation\_count

FROM parking

GROUP BY

CASE

WHEN violation\_time LIKE '%A' AND CAST(SUBSTR(violation\_time, 1, 2) AS INT) < 6 THEN 'Early Morning'

WHEN violation\_time LIKE '%A' THEN 'Morning'

WHEN violation\_time LIKE '%P' AND CAST(SUBSTR(violation\_time, 1, 2) AS INT) < 5 THEN 'Afternoon'

WHEN violation\_time LIKE '%P' AND CAST(SUBSTR(violation\_time, 1, 2) AS INT) < 9 THEN 'Evening'

ELSE 'Night'

END

ORDER BY violation\_count DESC;

1. 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

CASE

WHEN CAST(SUBSTR(violation\_time, 1, 2) AS INT) < 4 THEN '12:00 AM - 4:00 AM'

WHEN CAST(SUBSTR(violation\_time, 1, 2) AS INT) < 8 THEN '4:00 AM - 8:00 AM'

WHEN CAST(SUBSTR(violation\_time, 1, 2) AS INT) < 12 THEN '8:00 AM - 12:00 PM'

WHEN CAST(SUBSTR(violation\_time, 1, 2) AS INT) < 16 THEN '12:00 PM - 4:00 PM'

WHEN CAST(SUBSTR(violation\_time, 1, 2) AS INT) < 20 THEN '4:00 PM - 8:00 PM'

ELSE '8:00 PM - 12:00 AM'

END AS time\_bin,

violation\_code,

COUNT(\*) AS violation\_count

FROM parking

GROUP BY

CASE

WHEN CAST(SUBSTR(violation\_time, 1, 2) AS INT) < 4 THEN '12:00 AM - 4:00 AM'

WHEN CAST(SUBSTR(violation\_time, 1, 2) AS INT) < 8 THEN '4:00 AM - 8:00 AM'

WHEN CAST(SUBSTR(violation\_time, 1, 2) AS INT) < 12 THEN '8:00 AM - 12:00 PM'

WHEN CAST(SUBSTR(violation\_time, 1, 2) AS INT) < 16 THEN '12:00 PM - 4:00 PM'

WHEN CAST(SUBSTR(violation\_time, 1, 2) AS INT) < 20 THEN '4:00 PM - 8:00 PM'

ELSE '8:00 PM - 12:00 AM'

END,

violation\_code

ORDER BY time\_bin, violation\_count DESC limit 3;

1. 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).

SELECT

parking.violation\_code,

CASE

WHEN CAST(parking.violation\_time AS INT) >= 500 AND CAST(parking.violation\_time AS INT) < 1200 THEN 'Morning'

WHEN CAST(parking.violation\_time AS INT) >= 1200 AND CAST(parking.violation\_time AS INT) < 1700 THEN 'Afternoon'

WHEN CAST(parking.violation\_time AS INT) >= 1700 AND CAST(parking.violation\_time AS INT) < 2100 THEN 'Evening'

ELSE 'Night'

END AS time\_of\_day,

COUNT(\*) AS violation\_frequency

FROM parking

WHERE parking.violation\_code IN (

SELECT violation\_code

FROM parking

GROUP BY violation\_code

ORDER BY COUNT(\*) DESC

LIMIT 3

)

GROUP BY parking.violation\_code, time\_of\_day

ORDER BY violation\_frequency DESC;

Let’s try and find some seasonality in this data:

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

SELECT

SUM(CASE

WHEN issue\_date >= '01/03/2016' AND issue\_date <= '31/05/2016' THEN 1

ELSE 0

END) AS spring,

SUM(CASE

WHEN issue\_date >= '01/06/2016' AND issue\_date <= '31/08/2016' THEN 1

ELSE 0

END) AS summer,

SUM(CASE

WHEN issue\_date >= '01/09/2016' AND issue\_date <= '30/11/2016' THEN 1

ELSE 0

END) AS fall,

SUM(CASE

WHEN issue\_date >= '01/12/2015' AND issue\_date <= '29/02/2016' THEN 1

WHEN issue\_date >= '01/12/2016' AND issue\_date <= '28/02/2017' THEN 1

ELSE 0

END) AS winter

FROM parking;

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