BAN 5501 Case Study New York City Taxi Questions Due December 3 by 11:59 PM

Note: There are two tables in this database; trips\_sep & trips\_oct.

*For each question, include a short explanation (about 2-3 sentences) about how you solved it. This should describe the approach taken and key steps in your solution.*

*Example SQL Code:*

*SELECT customer\_id, COUNT(\*) AS total\_rentals FROM rental*

*GROUP BY customer\_id;*

*Explanation: I solved this by using a GROUP BY clause to count the number of rentals for each customer, allowing me to summarize the data based on customer\_id. COUNT(\*) function returns the total number of rentals per customer.*

1)

1. Find the third-most expensive trip (Total Amount column) in September.

**Code**:

SELECT \*

FROM trips\_sep

ORDER BY total\_amount DESC

OFFSET 2 LIMIT 1;

**Explanation**:

Here I have ordered the total amount in descending order and used offset 2 to skip the first and second most expensive trips and limited one value

1. Find the most expensive trip per mile (Total Amount/Mile) in October.

**Code:**

SELECT trip\_distance, total\_amount,

CASE

WHEN trip\_distance > 1 THEN total\_amount/trip\_distance

ELSE total\_amount

END AS expensive\_trip\_per\_mile

FROM trips\_oct

ORDER BY expensive\_trip\_per\_mile DESC

LIMIT 1;

**Explanation:**

I have selected trip distance, total amount and used a case statement to calculate expensive trip per mile to avoid the zero division error and ordered the trips by descending order to retrieve the expensive trip and limited 1 value.

1. Find the most generous trip in September (highest tip).

**Code:**

SELECT MAX(tip\_amount)

FROM trips\_sep;

**Explanation:**

Selected the highest tip amount using aggregate function

1. Find the longest trip duration in September.

**Code:**

SELECT MAX(AGE(lpep\_dropoff\_datetime, lpep\_pickup\_datetime))

FROM trips\_sep

**Explanation:**

Used age function on drop time and pick up time to calculate duration and used an aggregate function retrieve highest value

1. Find the average tip amount by the hour in September.

**Code:**

SELECT round(AVG(tip\_amount)::numeric,2) AS avg\_tip\_per\_hour,

date\_part('hour',lpep\_dropoff\_datetime) AS hour

FROM trips\_sep

GROUP BY hour

ORDER BY hour;

**Explanation:**

I have used AVG function to calculate average tip amount and rounded it to 2 decimal point and grouped by hour

1. Find the number of trips by day of the week in October. (Create a "Day of Week" column, e.g., Monday, Tuesday, ..., Sunday).

**Code:**

SELECT

TO\_CHAR(lpep\_pickup\_datetime, 'Day') AS day\_of\_week,

COUNT(\*) AS num\_of\_trips

FROM trips\_oct

GROUP BY day\_of\_week;

**Explanation:**

I have extracted day as day\_of\_week and grouped by day\_of\_week.

1. Find the average trip amount by the hour in October.

**Code:**

SELECT date\_part('hour',lpep\_dropoff\_datetime) AS hour

round(AVG(total\_amount)::numeric,2) AS avg\_amt\_per\_hr,

FROM trips\_oct

GROUP BY hour

ORDER BY hour;

**Explanation:**

I have selected average trip amount and rounded it to 2 decimal point, have grouped by hour extracted using date\_part function and ordered by hour

1. Determine which airport welcomes more customers: JFK or EWR. Note: Use a CASE expression to retrieve the names as "JFK," "Group Ride," or "Newark" from the ***ratecodeid column***. Refer to the Data Dictionary file.

**Code:**

SELECT

CASE

WHEN ratecodeid = 2 THEN 'JFK'

WHEN ratecodeid = 3 THEN 'Newark'

WHEN ratecodeid = 6 THEN 'Group Ride'

ELSE 'Other'

END AS airport,

COUNT(\*) AS num\_of\_trips

FROM trips\_sep

GROUP BY airport

ORDER BY num\_of\_trips DESC;

**Explanation:**

I have used a case function to retrieve the Airports name JFK, Newark and Group Ride, grouped by airport and ordered by number of trips which is retrieved using count(\*).

1. Create buckets or price ranges for the total amount and find the number of trips in each price range for each driver in September.

Use the following price ranges:

* + 0 <= Total Amount < 10
  + 10 <= Total Amount < 20
  + 20 <= Total Amount < 30
  + 30 <= Total Amount < 40
  + Total Amount >= 40; ELSE

**Code:**

SELECT driver\_id,

CASE

WHEN total\_amount < 10 THEN '0-10'

WHEN total\_amount < 20 THEN '10-20'

WHEN total\_amount < 30 THEN '20-30'

WHEN total\_amount < 40 THEN '30-40'

ELSE '40+'

END AS price\_range,

COUNT(\*) AS num\_of\_trips

FROM trips\_sep

GROUP BY driver\_id, price\_range

ORDER BY driver\_id, price\_range;

**Explanation:**

I have used case function for the price range and grouped by price\_range and driver\_id

1. Write a query to find the top three highest total amounts for each driver in October.

**Code:**

SELECT t1.driver\_id, t1.total\_amount

FROM trips\_oct t1

JOIN trips\_oct t2

ON t1.driver\_id = t2.driver\_id

AND t1.total\_amount <= t2.total\_amount

GROUP BY t1.driver\_id, t1.total\_amount

HAVING COUNT(\*) <= 3

ORDER BY t1.driver\_id, t1.total\_amount DESC;

**Explanation:**

I have used Self Join to to rank each trip by comparing its total amount to others of the same driver. It filters to include only the top three highest-earning trips per driver in October.

1. Find the 10 lowest total amounts for Driver 1 in October.

**Code:**

SELECT driver\_id, total\_amount

FROM trips\_oct

WHERE driver\_id = 1

ORDER BY total\_amount ASC

LIMIT 10;

**Explanation:**

I have selected driver\_id and total\_amount with a Where clause I have ordered by total amount and limited to 10 values

1. Write a query to track the cumulative earnings of Driver 1 after each trip in October. (Hint: Running total, Window functions).

**Code**:

SELECT driver\_id,

SUM(total\_amount) OVER (

PARTITION BY driver\_id

ORDER BY lpep\_pickup\_datetime

) AS cumulative\_earnings

FROM trips\_oct

WHERE driver\_id = 1;

**Explanation:**

I have used a window function and summed the total\_amount and ordered by pickup date

And partitioned by driver id and filtered the results for driver\_id 1

1. Is there any new driver in October? (Hint: Find drivers who exist in the October table but not in the September table)

Note: Return unique driver IDs.

**Code:**

SELECT DISTINCT driver\_id

FROM trips\_oct

WHERE driver\_id NOT IN (

SELECT DISTINCT driver\_id

FROM trips\_sep

);

**Explanation:**

I have used distinct function to get unique driver id and filtered out results for new driver

1. Find the total amount difference between September and October. In the output, display the total amount for September, the total amount for October, and the difference between the two.

**Code**:

SELECT

(SELECT SUM(total\_amount) FROM trips\_sep) AS total\_september\_amount,

(SELECT SUM(total\_amount) FROM trips\_oct) AS total\_october\_amount,

(SELECT SUM(total\_amount) FROM trips\_oct) - (SELECT SUM(total\_amount) FROM trips\_sep) AS amount\_diff;

**Explanation:**

I used subqueries to calculate the total amounts for both months and determined the difference by subtracting September's total from October's total.

1. Find the total revenue (Total Amount) for each driver in both September and October.

Ensure the output displays the total revenue for September, the total revenue for October, and the difference between the two. Sort the total revenue in descending order.

**Code:**

WITH sep\_total as (

SELECT round(sum(total\_amount)::numeric, 2) as sep\_amount from trips\_sep ), oct\_total as ( SELECT round(sum(total\_amount)::numeric, 2) as oct\_amount from trips\_oct )

SELECT sep\_amount, oct\_amount, (sep\_amount-oct\_amount) as difference

FROM sep\_total, oct\_total;

**Explanation**:

Here I have used cte’s to retrieve total amount from sep and oct table and then used them to find the difference. Used round function with casting the double precision to numeric to show the output in 2 decimals.

1. Find the total revenue (Total Amount) by day of the week (Monday, Tuesday, ..., Sunday) for both September and October. Ensure the output displays the total revenue for each day of the week in September, the total revenue for each day of the week in October, and the difference between the two. Sort the total revenue in descending order.

**Code**:

WITH sep\_amt as (

SELECT round(sum(total\_amount)::numeric, 2)as sep\_rev, driver\_id

FROM trips\_sep GROUP BY driver\_id ),

oct\_amt as ( select round(sum(total\_amount)::numeric, 2)as oct\_rev, driver\_id FROM trips\_oct group by driver\_id )

SELECT o.driver\_id, s.sep\_rev, o.oct\_rev,

CASE

WHEN sep\_rev IS NULL THEN oct\_rev

ELSE (o.oct\_rev - s.sep\_rev)

END AS revenue\_diff

FROM oct\_amt o

LEFT JOIN sep\_amt s

ON o.driver\_id = s.driver\_id

ORDER BY oct\_rev desc;

**Explanation:**

I used CTEs to calculate the total amount for each month for the respective drivers. Then, I performed a left join on the CTEs to include all drivers from October. Finally, I used a CASE expression to handle null values for the revenue of new drivers in September.