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.

SELECT \*

FROM trips\_sep

ORDER BY total\_amount DESC OFFSET 2

LIMIT 1;

*I solved this question by using an ORDER BY clause to sort the total amount in descending order. In addition, I have used the LIMIT clause to limit the number of rows to the 3rd row after excluding the first two rows in the sorted result. This has displayed the most expensive trip in September*

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

SELECT \*,

total\_amount /trip\_distance AS rate\_per\_mile FROM trips\_oct

WHERE trip\_distance != 0 ORDER BY rate\_per\_mile DESC

LIMIT 1;

*I solved this question by using the equation total\_amount/trip\_distance to reach at the total amount per mile. After that, the resulting column was sorted in the descending order by using an ORDER BY clause, followed by LIMIT clause to limit it with the highest expensive trip per mile.*

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

SELECT \*

FROM trips\_sep

ORDER BY tip\_amount DESC LIMIT 1;

*I solved this question by using an ORDER BY clause to sort the tip\_amount in descending order. In addition, I have used the LIMIT clause to limit the number of rows to the very first row in the sorted result. This has displayed the most generous trip in September*

1. Find the longest trip duration in September.

SELECT \*,

(lpep\_dropoff\_datetime - lpep\_pickup\_datetime) AS trip\_duration FROM trips\_sep

ORDER BY trip\_duration DESC

*I solved this question by using an ORDER BY clause to sort the trip\_duration in descending order. In addition, I have used the lpep\_dropoff\_datetime - lpep\_pickup\_datetime calculation to determine trip durations. Finally, the query displays the trips with the longest duration in September at the top of the result.*

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

SELECT

AVG(tip\_amount),

EXTRACT(HOUR FROM lpep\_dropoff\_datetime) as hour FROM trips\_sep

GROUP BY EXTRACT(HOUR FROM lpep\_dropoff\_datetime) ORDER BY hour

*I solved this question by using the AVG function to calculate the average tip\_amount for each hour. Additionally, I used the EXTRACT function to extract the hour from lpep\_dropoff\_datetime. The query groups the results by hour and sorts them in ascending order to display the average tips by hour in September.*

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

SELECT

COUNT(\*) AS no\_of\_trips, TO\_CHAR(lpep\_pickup\_datetime, 'Day') AS day\_of\_month FROM trips\_oct

GROUP BY TO\_CHAR(lpep\_pickup\_datetime, 'Day')

*I solved this question by using the COUNT function to calculate the total number of trips (no\_of\_trips) for each day of the week. Additionally, I used the TO\_CHAR function to extract the day of the week from lpep\_pickup\_datetime. The query groups the results by the day of the week.*

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

SELECT AVG(total\_amount), EXTRACT(HOUR from lpep\_pickup\_datetime) FROM trips\_oct

GROUP BY EXTRACT(HOUR from lpep\_pickup\_datetime) ORDER BY EXTRACT(HOUR from lpep\_pickup\_datetime)

*I solved this question by using the AVG function to calculate the average trip amount (total\_amount) for each hour of the day. Additionally, I used the EXTRACT(HOUR) function to extract the hour from lpep\_pickup\_datetime. The query groups the results by hour and sorts them in ascending order to display the average trip amount for each hour in October.*

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.

SELECT COUNT(\*), CASE

WHEN ratecodeid = 1 THEN 'Standard Rate' WHEN ratecodeid = 2 THEN 'JFK'

WHEN ratecodeid = 3 THEN 'Newark'

WHEN ratecodeid = 4 THEN 'Nassau or Westchester' WHEN ratecodeid = 5 THEN 'Negotiated Fare' WHEN ratecodeid = 6 THEN 'Group ride'

END AS rate\_identifier FROM trips\_sep

GROUP BY rate\_identifier; Answer: JFK

*I solved this question by using the COUNT function to calculate the total number of trips for each type of ratecodeid. Additionally, I used a CASE expression to map the ratecodeid values to meaningful names such as "Standard Rate," "JFK," "Newark," and so on. The query groups the results by the rate identifier and displays the count of trips for each category.*

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

WITH cte\_1 AS (SELECT driver\_id,

CASE

WHEN total\_amount >= 0 AND total\_amount < 10 THEN '0-9' WHEN total\_amount >= 10 AND total\_amount < 20 THEN '10-19' WHEN total\_amount >= 20 AND total\_amount < 30 THEN '20-29' WHEN total\_amount >= 30 AND total\_amount < 40 THEN '30-39' ELSE '40+'

END AS total\_amount\_range FROM trips\_sep)

SELECT driver\_id, total\_amount\_range,

COUNT(\*) OVER (PARTITION BY driver\_id, total\_amount\_range) AS trip\_count FROM cte\_1

GROUP BY driver\_id, total\_amount\_range;

*I solved this question by using the COUNT function to calculate the total number of trips within different fare ranges based on total\_amount. A CASE expression was utilized to categorize the total\_amount into specific ranges such as "0-9," "10-19," "20-29," etc. The query groups the results by these fare ranges and displays the count of trips for each category.*

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

WITH cte\_1 AS (SELECT driver\_id, total\_amount,

RANK() OVER(PARTITION BY driver\_id

ORDER BY total\_amount DESC) AS rnk FROM trips\_oct)

SELECT

driver\_id, total\_amount, rnk

FROM cte\_1

WHERE rnk IN(1, 2, 3)

*I solved this question by using a Common Table Expression (CTE) to first rank the trips for each driver based on the total\_amount in descending order. The RANK() window function was used*

*with the PARTITION BY driver\_id clause to rank the trips for each driver separately. After that, the main query selects the top 3 trips (with rank 1, 2, and 3) for each driver, displaying the driver ID, total amount for the trip, and the rank.*

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

WITH cte\_1 AS (SELECT driver\_id, total\_amount,

ROW\_NUMBER() OVER(PARTITION BY driver\_id

ORDER BY total\_amount ASC) AS rnk FROM trips\_oct)

SELECT

driver\_id, total\_amount, rnk

FROM cte\_1

WHERE (rnk BETWEEN 1 AND 10)

AND driver\_id = 1

*I solved this question by using a* Common Table Expression (CTE) *to first rank the trips for each driver based on the* total\_amount *in ascending order using the* ROW\_NUMBER() *window function. The* PARTITION BY driver\_id *clause ensures that the ranking is calculated separately for each driver. The main query then selects the top 10 lowest fare trips for* Driver 1 *(with ranks 1 to 10) and displays the driver ID, total amount for the trip, and the rank.*

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

SELECT

driver\_id, trip\_id,

SUM(total\_amount) OVER (PARTITION BY driver\_id ORDER BY lpep\_pickup\_datetime

ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)

FROM trips\_oct WHERE driver\_id = 1

*I solved this question by using the* SUM() *window function to calculate the cumulative earnings for* Driver 1 *after each trip. The query partitions the data by* **driver\_id** *and orders it by* lpep\_pickup\_datetime *to ensure the running total is calculated in chronological order. The* ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW *clause ensures that the sum starts from the first row and accumulates up to the current row, providing the cumulative earnings for each trip.*

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.

SELECT DISTINCT driver\_id FROM trips\_oct

WHERE driver\_id NOT IN (SELECT DISTINCT driver\_id FROM trips\_sep)

*I solved this question by using the* ***NOT IN*** *operator along with a* ***subquery*** *to identify drivers who appear exclusively in the trips\_oct table and not in the trips\_sep table. The subquery retrieves all distinct driver\_id values from the trips\_sep table, and the main query filters out these IDs from the trips\_oct table. This approach ensures that only drivers present in October but absent in September are included in the result*

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.

WITH CTE\_1 AS

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

FROM trips\_sep) SELECT

total\_amount\_sept, total\_amount\_oct,

ROUND((total\_amount\_oct - total\_amount\_sept)::NUMERIC, 2) AS difference FROM CTE\_1

*I solved this question by using a CTE to calculate the total earnings for both September and October. The SUM functions are used within the CTEs for each month. The difference between October and September earnings is calculated and rounded to two decimal places. The final results display the earnings for both months and their difference*.

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.

WITH cte\_1 AS

(SELECT SUM(total\_amount) AS total\_amount\_sept, driver\_id FROM trips\_sep

GROUP BY driver\_id),

cte\_2 AS

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

GROUP BY driver\_id)

SELECT

oct.driver\_id, total\_amount\_oct, total\_amount\_sept,

ROUND((total\_amount\_oct - COALESCE(total\_amount\_sept, 0))::NUMERIC, 2) AS difference FROM cte\_1 AS SEPT

FULL JOIN cte\_2 AS OCT

ON OCT.driver\_id = SEPT.driver\_id ORDER BY OCT.driver\_id

*I solved this question by using two CTEs to calculate total earnings for each driver in September and October. A FULL JOIN combines the results, with COALESCE handling null values. The difference in earnings between October and September is calculated and rounded to two decimal places. The results are ordered by driver\_id*

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.

WITH cte\_1 AS (SELECT

TO\_CHAR(lpep\_pickup\_datetime, 'Day') AS day\_of\_week, SUM(total\_amount) AS total\_revenue\_sept

FROM trips\_sep

GROUP BY TO\_CHAR(lpep\_pickup\_datetime, 'Day')),

cte\_2 AS (SELECT

TO\_CHAR(lpep\_pickup\_datetime, 'Day') AS day\_of\_week, SUM(total\_amount) AS total\_revenue\_oct

FROM trips\_oct

GROUP BY TO\_CHAR(lpep\_pickup\_datetime, 'Day'))

SELECT

cte\_1.day\_of\_week, total\_revenue\_sept, total\_revenue\_oct,

total\_revenue\_oct - total\_revenue\_sept AS total\_change\_in\_revenue FROM cte\_1 AS cte\_1

INNER JOIN cte\_2 ON

cte\_1.day\_of\_week = cte\_2.day\_of\_week ORDER BY total\_revenue\_oct DESC

*I solved this query by using two CTEs to calculate the total revenue generated for each day of the week in September and October. An INNER JOIN is applied to combine the results based on the day of the week. The difference in total revenue between October and September is computed and displayed as total\_change\_in\_revenue. The final results are ordered in descending order of*

*total revenue October.*