**ZOMATO SQL ANALYSIS**

**Data Import and Table Creation**

Successfully imported 5 CSV files into SQL Server to create the foundational database tables:

Orders

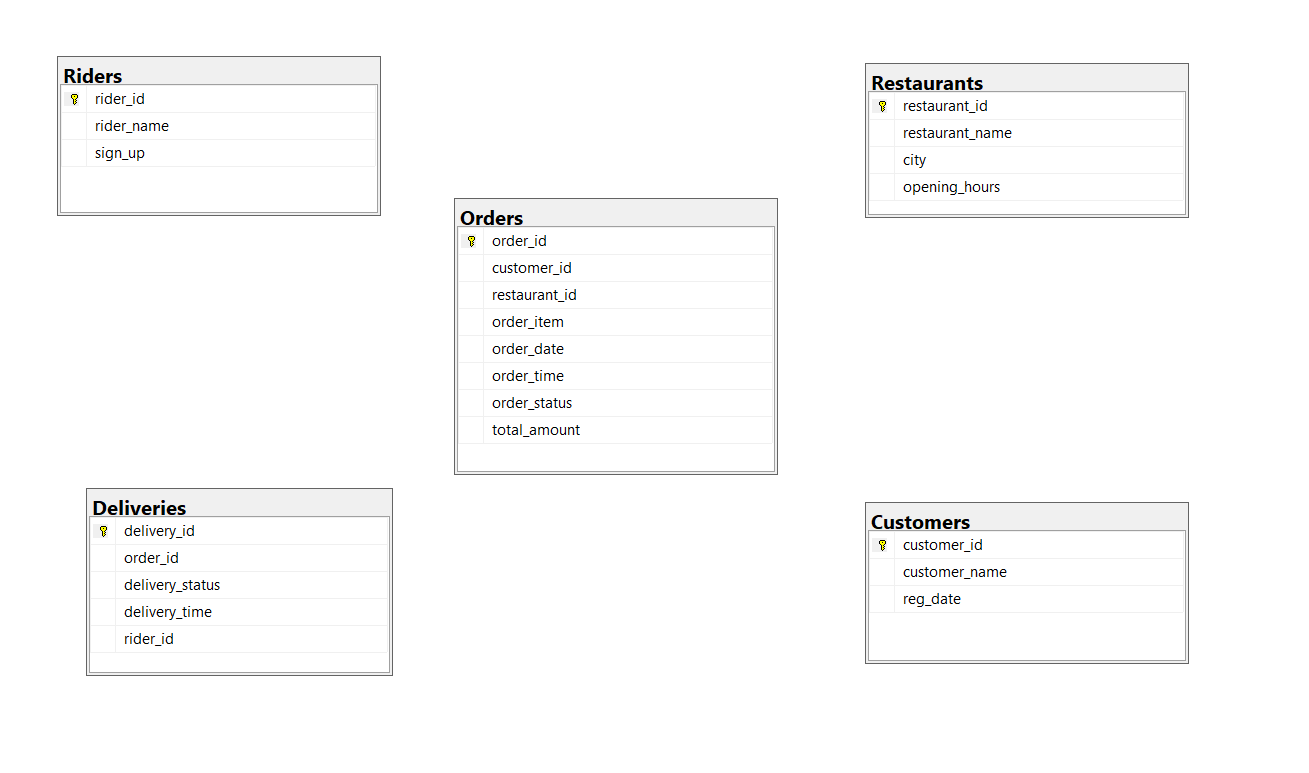
Customers

Deliveries

Restaurants

Riders

* Adjusted data types during import to ensure compatibility and efficiency (e.g., INT, VARCHAR, DATE).
* Applied necessary constraints such as PRIMARY KEY to enforce data integrity and uniqueness.



ALTER TABLE Deliveries

ADD CONSTRAINT FK\_Deliveries\_Orders

FOREIGN KEY (Order\_id) REFERENCES Orders(order\_id);

ALTER TABLE Deliveries

ADD CONSTRAINT FK\_Deliveries\_Orders

FOREIGN KEY (Order\_id) REFERENCES Orders(order\_id);

ALTER TABLE Orders

ADD CONSTRAINT FK\_Orders\_Customer\_id

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id);

ALTER TABLE Orders

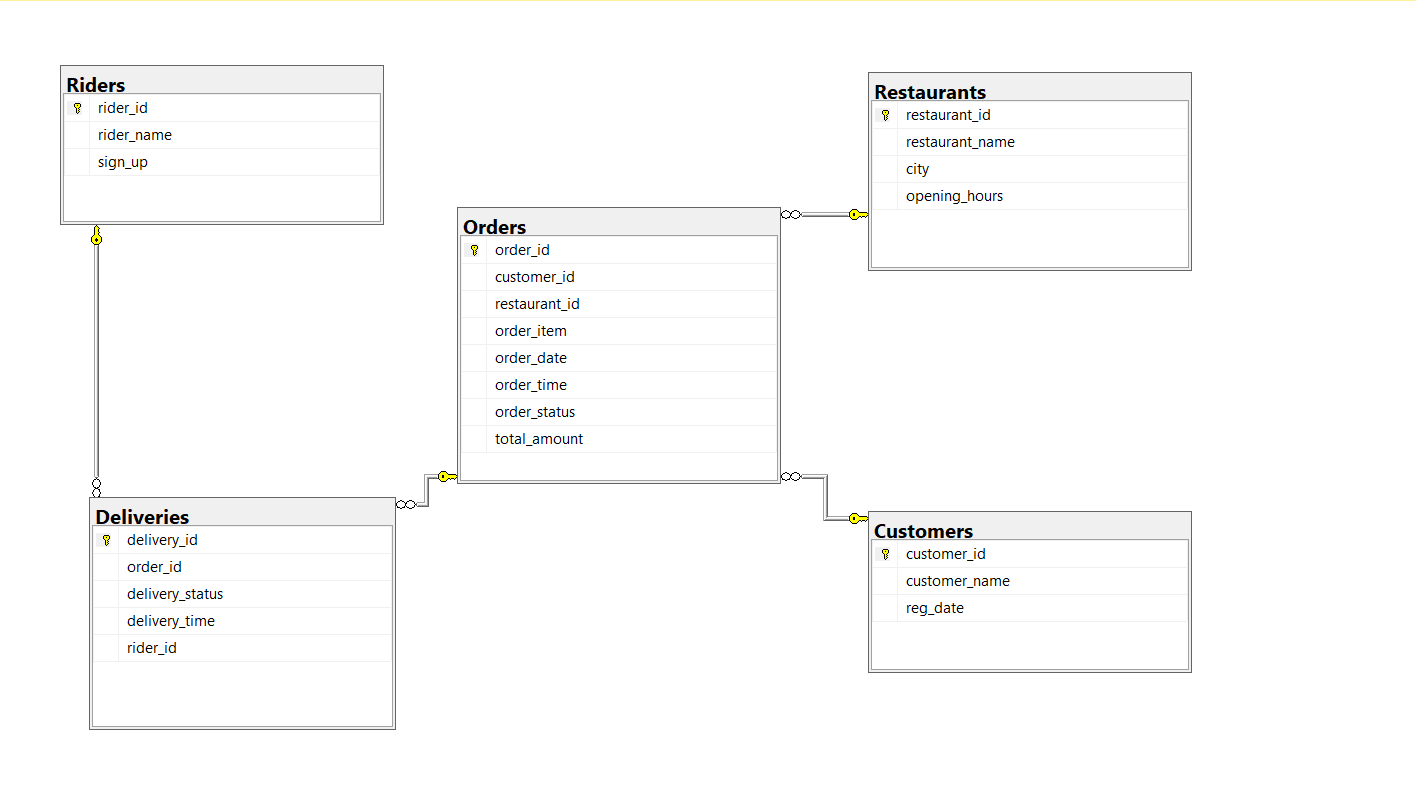
ADD CONSTRAINT FK\_Orders\_Restaurant\_id

FOREIGN KEY (restaurant\_id) REFERENCES Restaurants(restaurant\_id);

ALTER TABLE Deliveries

ADD CONSTRAINT FK\_Deliveries\_rider\_id

FOREIGN KEY (rider\_id) REFERENCES Riders(rider\_id);



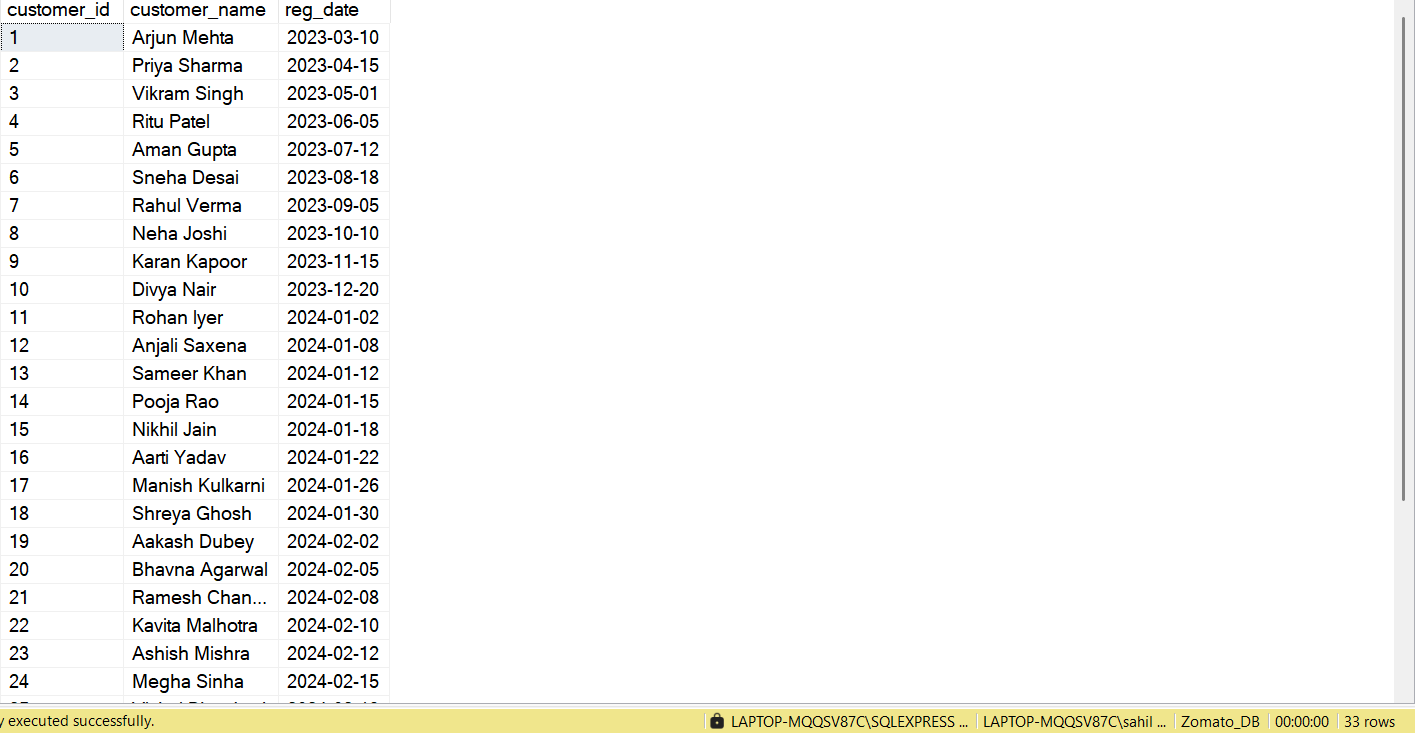
Established a relational database ready for data analysis, reporting, and optimization.

The design ensures:

* Data Integrity: Only valid data can be inserted or updated.
* Efficient Queries: Optimized structure for faster query performance.
* Database is now robust and scalable for real-world business applications.

**Exploratory Data Analysis**

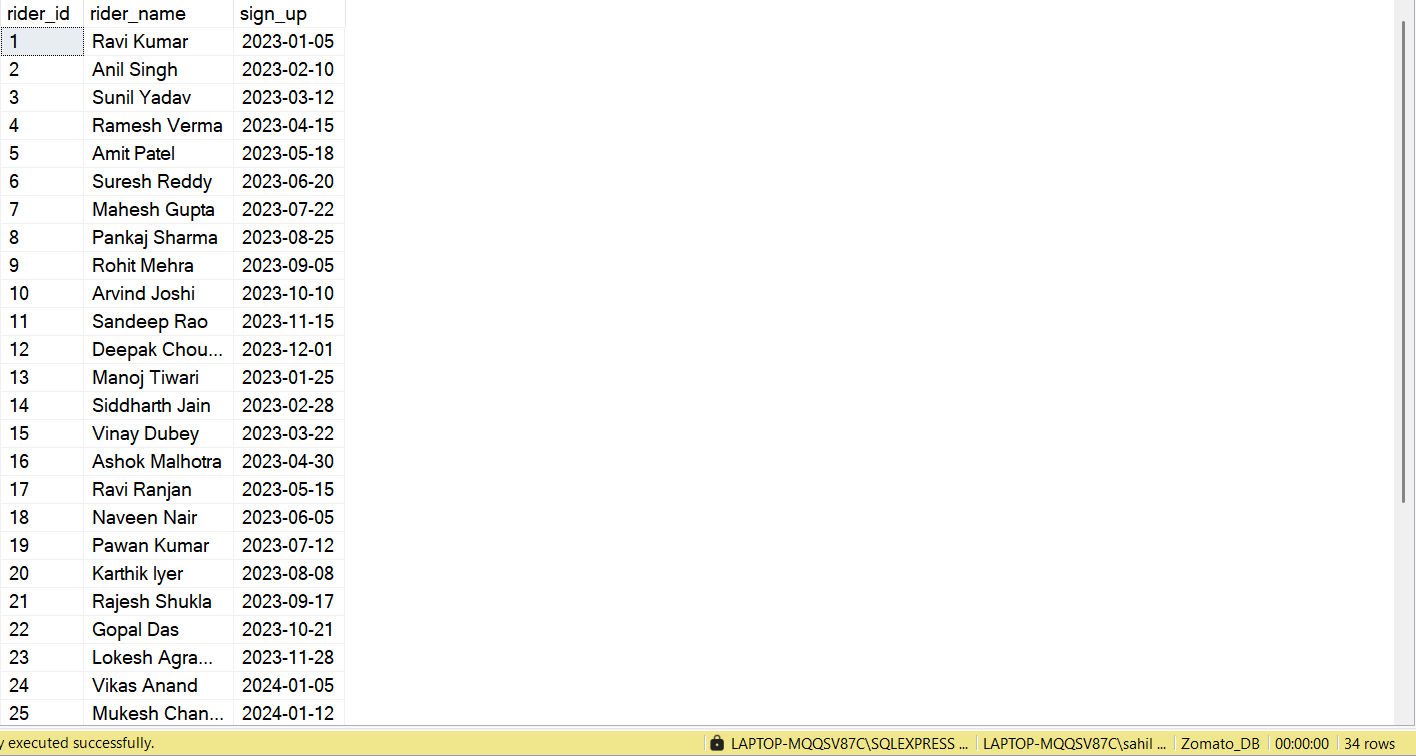
SELECT \* FROM Customers



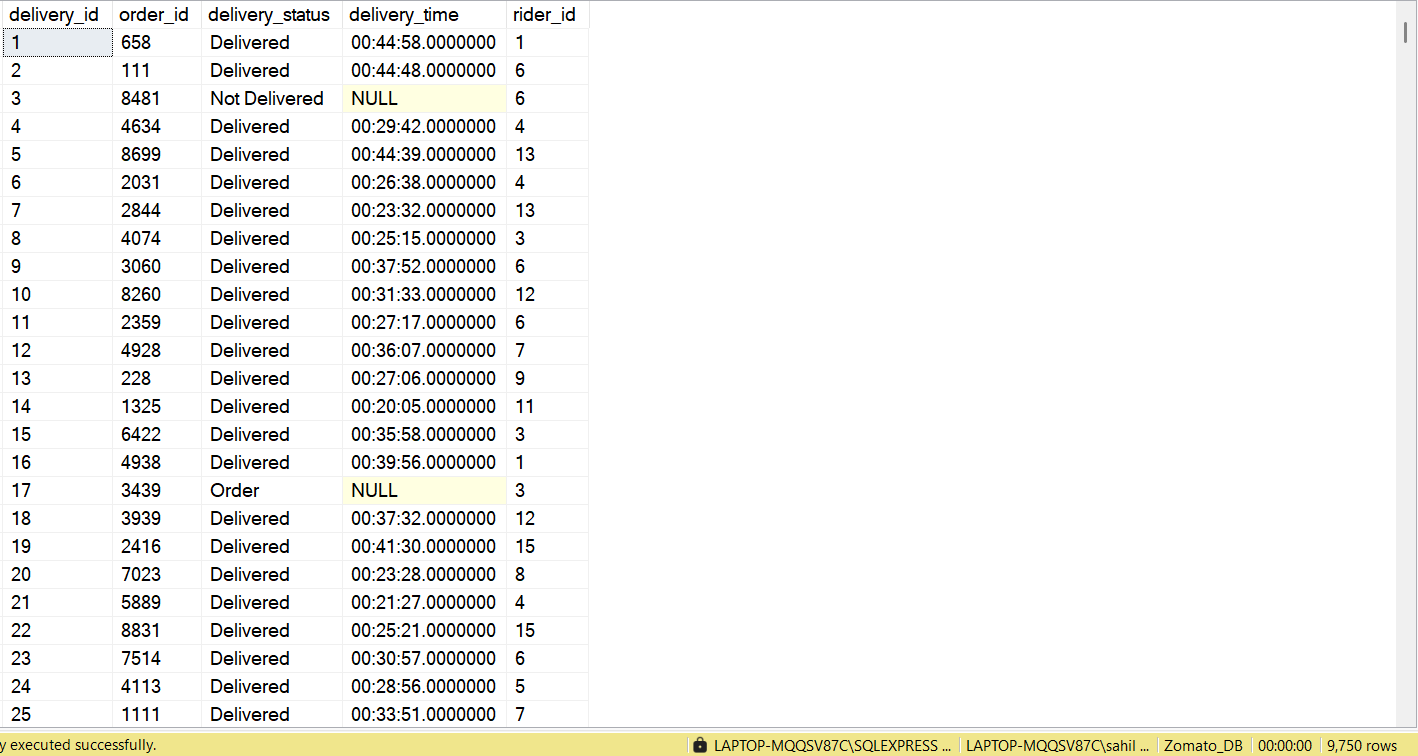
SELECT \* FROM Orders



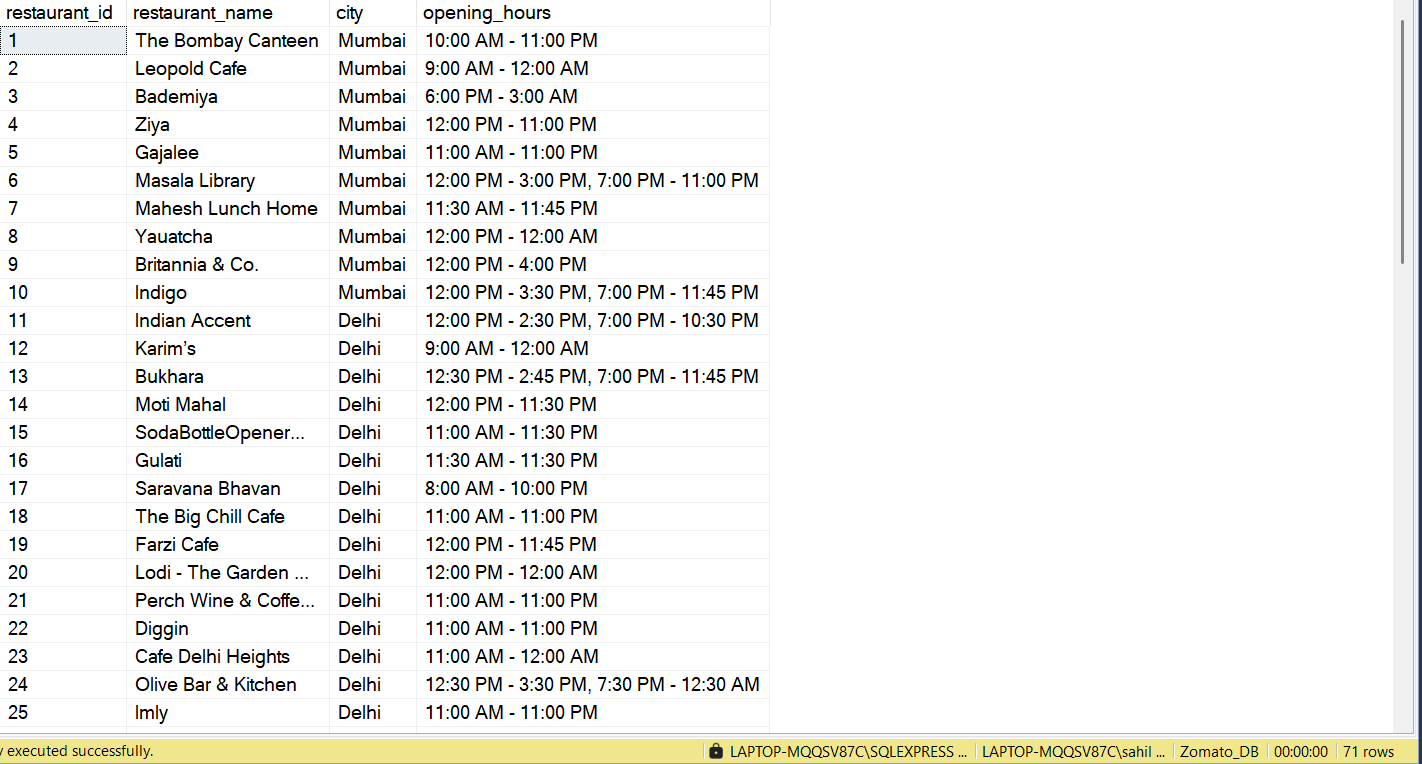
SELECT \* FROM Riders



SELECT \* FROM Deliveries



SELECT \* FROM Restaurants



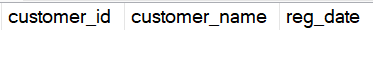
SELECT \* FROM Customers

WHERE

customer\_id IS NULL OR

customer\_name IS NULL OR

reg\_date IS NULL



SELECT \* FROM Orders

WHERE

order\_id IS NULL OR

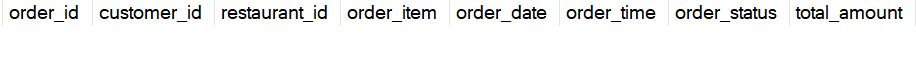
order\_item IS NULL OR

order\_date IS NULL OR

Order\_time IS NULL OR

order\_status IS NULL OR

total\_amount IS NULL



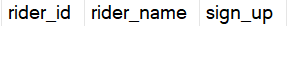
SELECT \* FROM Riders

WHERE

rider\_id IS NULL OR

rider\_name IS NULL OR

sign\_up IS NULL



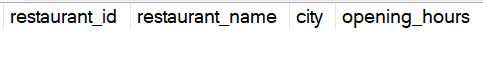
SELECT \* FROM Restaurants

WHERE

restaurant\_name IS NULL OR

opening\_hours IS NULL OR

city IS NULL



SELECT \* FROM Deliveries

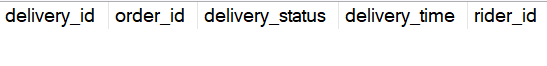
WHERE

delivery\_id IS NULL OR

order\_id IS NULL OR

rider\_id IS NULL OR

delivery\_status IS NULL



The last order's date in the Orders table

SELECT MAX(order\_date) Last\_Order\_Date FROM Orders



here we are considering Today's date as this

SELECT CAST(GETDATE() AS date) Todays\_Date



Q1 Write a query to find the top 5 most frequently ordered dishes by customer called "Arjun Mehta" in the last 2 year

SELECT DATEADD(YEAR,-2, CAST(GETDATE() AS date)) CurrentDate



SELECT customer\_name, Dishes, Rank\_of\_dishes FROM

( SELECT

c.customer\_id,

c.customer\_name,

o.order\_item as Dishes,

COUNT(o.order\_item) Nr\_of\_time\_item\_Order,

DENSE\_RANK() OVER(Order by COUNT(o.order\_item) DESC) Rank\_of\_dishes

FROM Orders o

JOIN Customers c

ON o.customer\_id = c.customer\_id

WHERE

c.customer\_name = 'Arjun Mehta'

AND

order\_date > DATEADD(YEAR,-2,CAST(GETDATE() AS date))

GROUP BY

c.customer\_id,

c.customer\_name,

o.order\_item

) t1

WHERE Rank\_of\_dishes <= 5



Q2 Popular Time slot: Identify the time Slots during which the most orders are placed. based on 2 hours interval

Here for 00:59:59--> 0 and 1:59:59-->1, so 0 to 1 is Slot of 2 hours

WITH Popular\_time\_slot

AS

(

SELECT \*,

CASE

WHEN DATEPART(HH,order\_time) BETWEEN 0 AND 1 THEN '00:00 - 02:00'

WHEN DATEPART(HH,order\_time) BETWEEN 2 AND 3 THEN '02:00 - 04:00'

WHEN DATEPART(HH,order\_time) BETWEEN 4 AND 5 THEN '04:00 - 06:00'

WHEN DATEPART(HH,order\_time) BETWEEN 6 AND 7 THEN '06:00 - 08:00'

WHEN DATEPART(HH,order\_time) BETWEEN 8 AND 9 THEN '08:00 - 10:00'

WHEN DATEPART(HH,order\_time) BETWEEN 10 AND 11 THEN '10:00 - 12:00'

WHEN DATEPART(HH,order\_time) BETWEEN 12 AND 13 THEN '12:00 - 14:00'

WHEN DATEPART(HH,order\_time) BETWEEN 14 AND 15 THEN '14:00 - 16:00'

WHEN DATEPART(HH,order\_time) BETWEEN 16 AND 17 THEN '16:00 - 18:00'

WHEN DATEPART(HH,order\_time) BETWEEN 18 AND 19 THEN '18:00 - 20:00'

WHEN DATEPART(HH,order\_time) BETWEEN 20 AND 21 THEN '20:00 - 22:00'

WHEN DATEPART(HH,order\_time) BETWEEN 22 AND 23 THEN '22:00 - 00:00'

END AS Time\_Slot

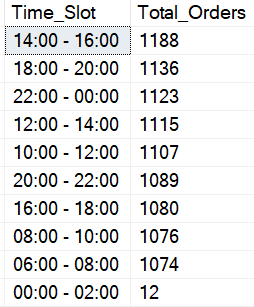
FROM Orders

)

SELECT Time\_Slot, COUNT(\*) Total\_Orders FROM Popular\_time\_slot

GROUP BY Time\_Slot

ORDER BY COUNT(order\_id) DESC



Q3 Order Value Analysis: Find the Average Order value per customer who has placed more than 750 orders

Return Customer\_name, and AOV(Average Order Value)

SELECT

c.customer\_id,

c.customer\_name,

CAST(AVG(O.total\_amount) AS decimal(10,2)) Avg\_Order\_Value

FROM Customers c

JOIN Orders o

ON c.customer\_id = o.customer\_id

GROUP BY c.customer\_id, c.customer\_name

HAVING

COUNT(o.order\_id) > 750

ORDER BY

AVG(total\_amount) DESC



Q4 High Value Customers: List the Customers who have spent more than 100K in total on food orders.

Return customer\_name, and customer\_id

SELECT

c.customer\_id,

c.customer\_name,

SUM(o.total\_amount) Total\_Spent

FROM Customers c

JOIN Orders o

ON c.customer\_id = o.customer\_id

GROUP BY c.customer\_id,

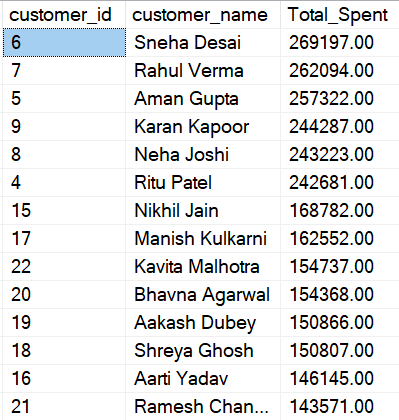
c.customer\_name

HAVING

SUM(o.total\_amount) > 100000

ORDER BY

SUM(o.total\_amount) DESC



Q5 Orders without Delivery: Write query to find orders that were placed but not delivered.

-Return each restaurant name, city and number of not delivered orders

Here we have to include both cases where orders was not fulfilled and Delivery status is "Not delivered"

SELECT

r.restaurant\_name,

r.city,

COUNT(o.order\_id) AS Total\_Not\_Delivered\_Orders

FROM

Orders o

LEFT JOIN Deliveries d

ON o.order\_id = d.order\_id

LEFT JOIN Restaurants r

ON r.restaurant\_id = o.restaurant\_id

WHERE

d.delivery\_status = 'Not Delivered'

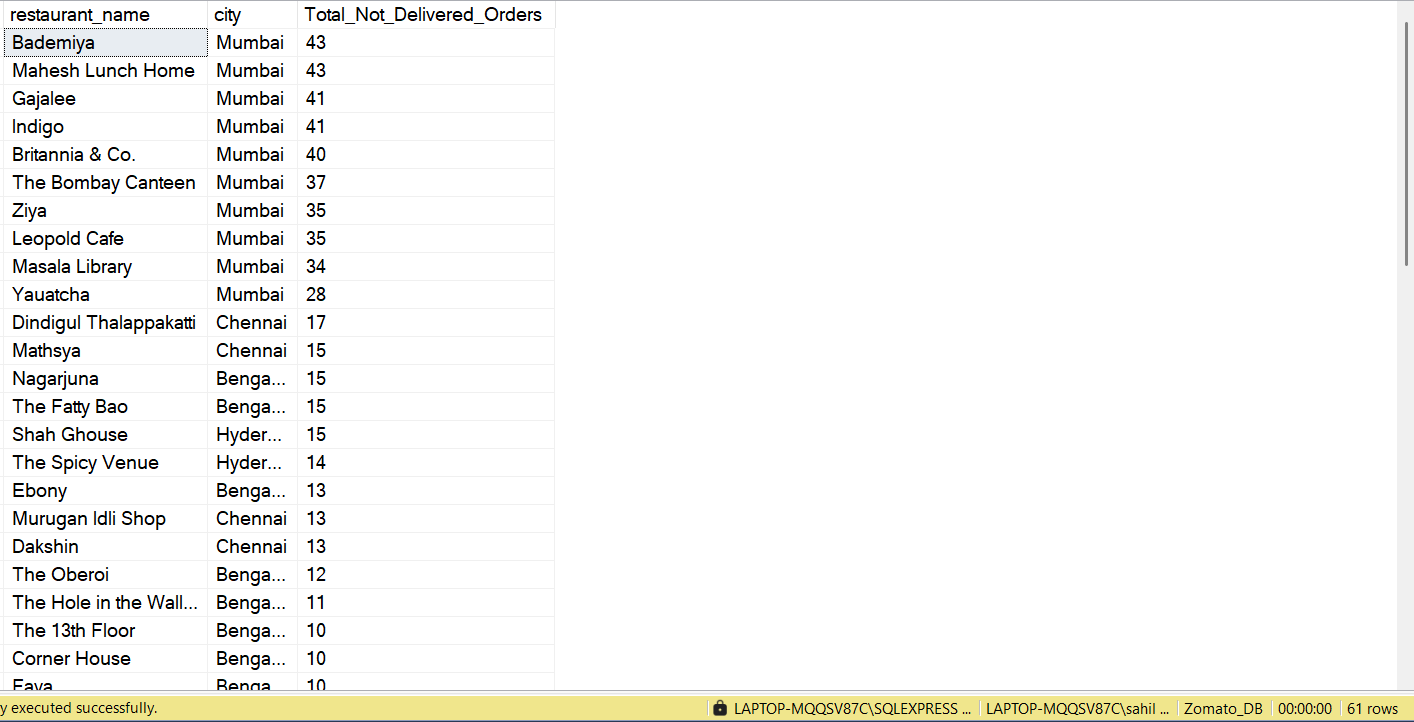
OR d.delivery\_status IS NULL -- Capture orders with no delivery entry

GROUP BY

r.restaurant\_name, r.city

ORDER BY

Total\_Not\_Delivered\_Orders DESC;



Q6 Restaurant Revenue Ranking: Rank restaurants by their total reveneu from the last year. including their name, Total Revenue, and rank within their city

SELECT

r.city,

r.restaurant\_name,

SUM(o.total\_amount) Revenue,

DENSE\_RANK() OVER(PARTITION BY city ORDER BY SUM(total\_amount) DESC) Rank\_of\_Restaurant

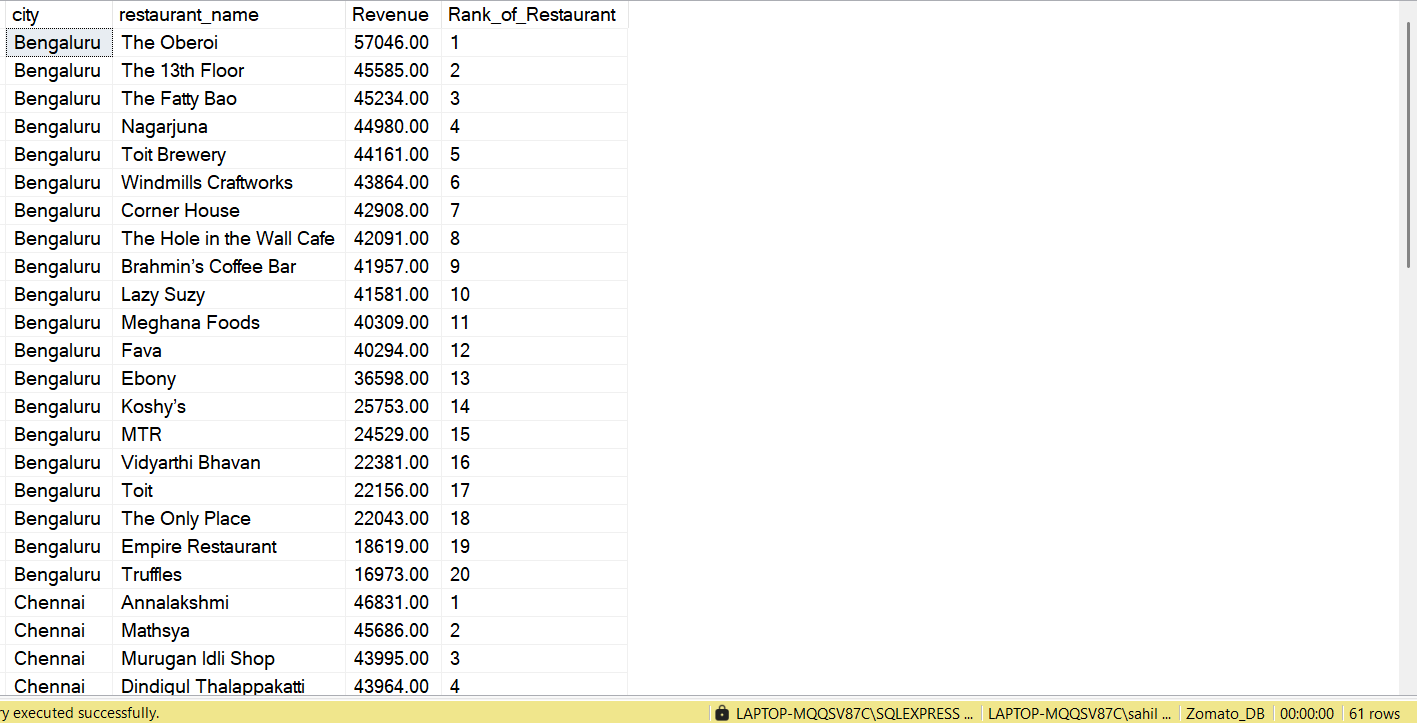
FROM Orders o

LEFT JOIN Restaurants r

ON r.restaurant\_id = o.restaurant\_id

WHERE YEAR(order\_date) < 2024

GROUP BY city, r.restaurant\_name



Top 3 Restaurant in their City based on Their Highest Revenue Revenue

WITH Rankin\_Table

AS (

SELECT

r.city City,

r.restaurant\_name Restaurant,

SUM(o.total\_amount) Revenue,

DENSE\_RANK() OVER(PARTITION BY city ORDER BY SUM(total\_amount) DESC) Rank\_of\_Restaurant

FROM Orders o

LEFT JOIN Restaurants r

ON r.restaurant\_id = o.restaurant\_id

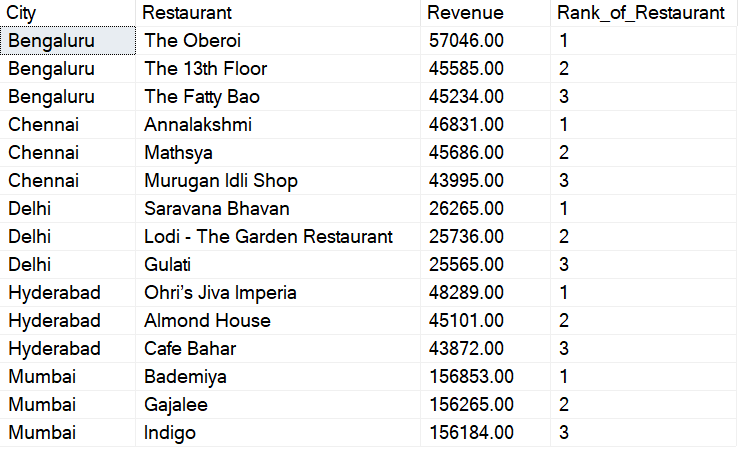
WHERE

YEAR(order\_date) < 2024

GROUP BY city, r.restaurant\_name)

SELECT \* FROM Rankin\_Table

WHERE Rank\_of\_Restaurant <= 3



Q7 Most popular dish by City:

Identify the Most Popular dish in each city based on the number of orders

WITH Most\_Popular\_dish

AS

(SELECT

city,

order\_item as Dishes,

COUNT(order\_id) Nr\_of\_Orders,

DENSE\_RANK() OVER(PARTITION BY city ORDER BY COUNT(order\_id)DESC) Rank\_of\_Dish

FROM Orders o

LEFT JOIN Restaurants r

ON r.restaurant\_id = o.restaurant\_id

GROUP BY city, order\_item

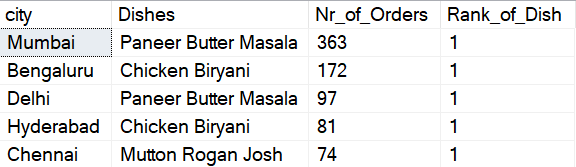
)

SELECT \* FROM Most\_Popular\_dish

WHERE Rank\_of\_Dish = 1

ORDER BY

Nr\_of\_Orders DESC



Q8 Customer Churn

Find Customers who haven't placed an order in 2024 but did in 2023

SELECT DISTINCT c.\* FROM Orders o

LEFT JOIN Customers c

ON o.customer\_id = c.customer\_id

WHERE

YEAR(o.order\_date) = 2023

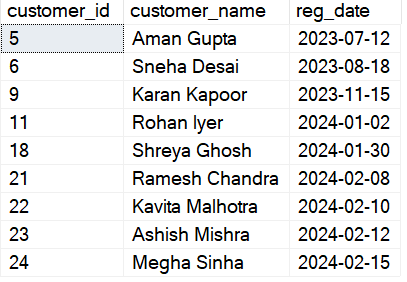
AND

c.customer\_id NOT IN

(SELECT DISTINCT customer\_id FROM Orders

WHERE YEAR(order\_date) = 2024)

ORDER BY c.customer\_id



Q9 Cancelled Rate Comparison:

Calculate and Compare the order Cancellation rate for each restaurant between the currrent year and previous year

WITH CANCELLED\_RATE

AS

(

SELECT

o.restaurant\_id,

COUNT(o.order\_id) AS Total\_Orders,

COUNT(CASE WHEN d.delivery\_id IS NULL THEN 1 END) AS Nr\_of\_Cancelled\_Orders

FROM Orders o

LEFT JOIN Deliveries d

ON d.order\_id = o.order\_id

WHERE YEAR(o.order\_date) = 2023

GROUP BY o.restaurant\_id

)

SELECT

restaurant\_id,

Total\_Orders,

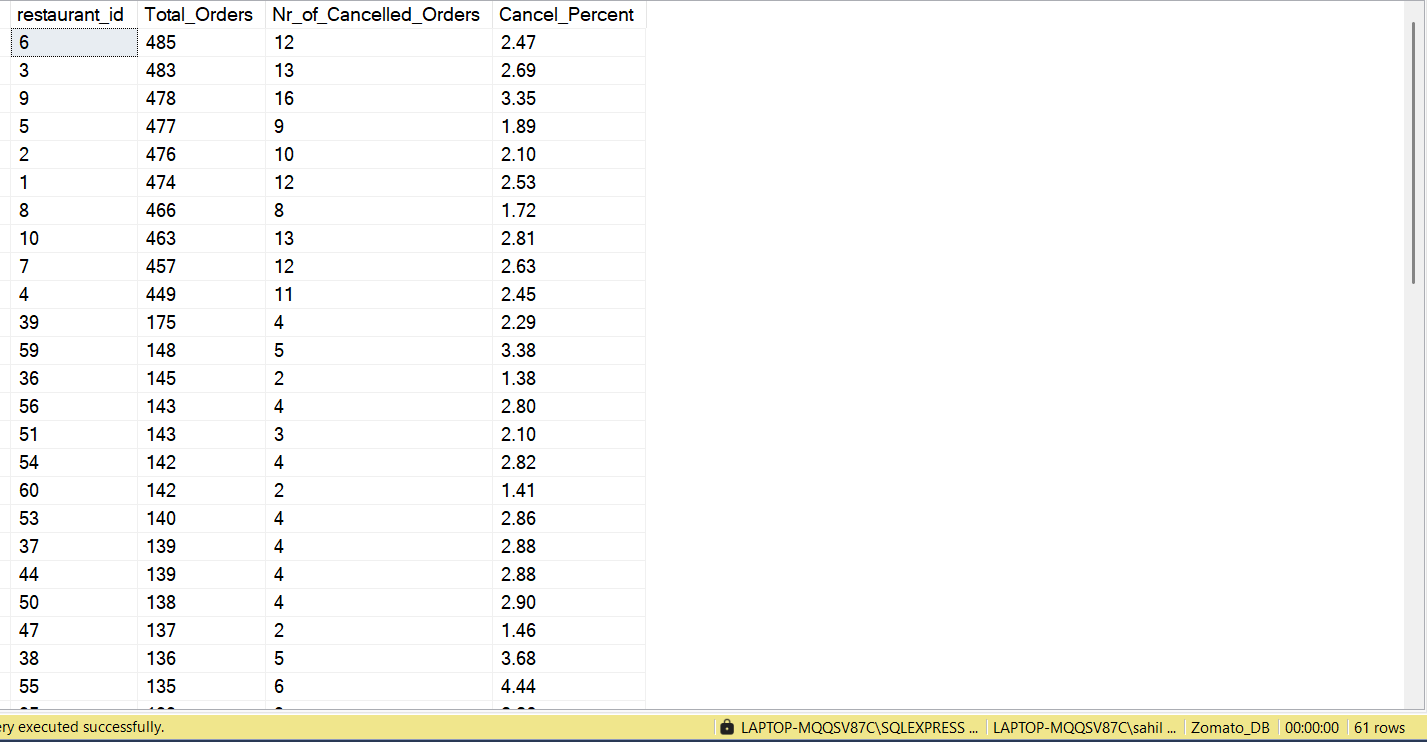
Nr\_of\_Cancelled\_Orders,

CAST(CAST(Nr\_of\_Cancelled\_Orders AS decimal(10,2))

/ CAST(Total\_Orders AS decimal(10,2)) \* 100 AS decimal(10,2)) AS Cancel\_rate

FROM CANCELLED\_RATE

ORDER BY Total\_Orders DESC



-- Now IF I want to Check Cancellation rate for year 2024

WITH CANCELLED\_RATE

AS

(

SELECT

o.restaurant\_id,

COUNT(o.order\_id) AS Total\_Orders,

COUNT(CASE WHEN d.delivery\_id IS NULL THEN 1 END) AS Nr\_of\_Cancelled\_Orders

FROM Orders o

LEFT JOIN Deliveries d

ON d.order\_id = o.order\_id

WHERE YEAR(o.order\_date) = 2024

GROUP BY o.restaurant\_id

)

SELECT

restaurant\_id,

Total\_Orders,

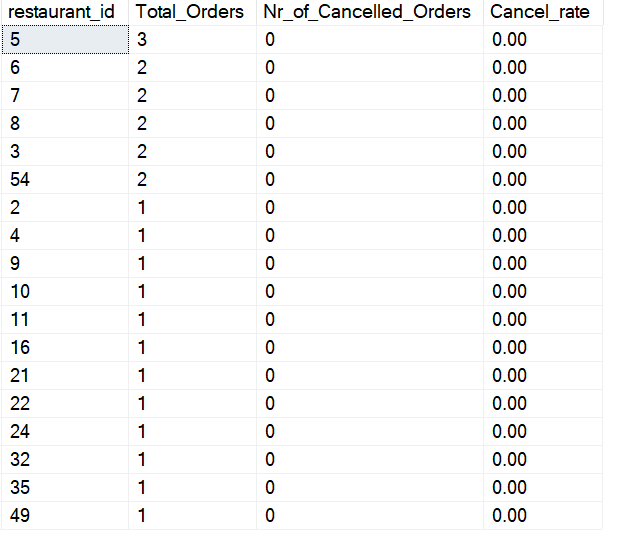
Nr\_of\_Cancelled\_Orders,

CAST(CAST(Nr\_of\_Cancelled\_Orders AS decimal(10,2))

/ CAST(Total\_Orders AS decimal(10,2)) \* 100 AS decimal(10,2)) AS Cancel\_rate

FROM CANCELLED\_RATE

ORDER BY Total\_Orders DESC



-- Final Solution of Quesiton 9, Rearranging CTEs

WITH CANCELLED\_RATE\_2023

AS

(

SELECT

o.restaurant\_id,

COUNT(o.order\_id) AS Total\_Orders,

COUNT(CASE WHEN d.delivery\_id IS NULL THEN 1 END) AS Nr\_of\_Cancelled\_Orders

FROM Orders o

LEFT JOIN Deliveries d

ON d.order\_id = o.order\_id

WHERE YEAR(o.order\_date) = 2023

GROUP BY o.restaurant\_id

),

CANCELLED\_RATE\_2024

AS

(

SELECT

o.restaurant\_id,

COUNT(o.order\_id) AS Total\_Orders

COUNT(CASE WHEN d.delivery\_id IS NULL THEN 1 END) AS Nr\_of\_Cancelled\_Orders

FROM Orders o

LEFT JOIN Deliveries d

ON d.order\_id = o.order\_id

WHERE YEAR(o.order\_date) = 2024

GROUP BY o.restaurant\_id

),

Last\_Year\_Data

AS

(

SELECT

restaurant\_id,

Total\_Orders,

Nr\_of\_Cancelled\_Orders,

CAST(CAST(Nr\_of\_Cancelled\_Orders AS decimal(10,2))

/ CAST(Total\_Orders AS decimal(10,2)) \* 100 AS decimal(10,2)) AS Cancel\_Percent

FROM CANCELLED\_RATE\_2023

),

Current\_year\_Data

AS

(

SELECT

restaurant\_id,

Total\_Orders,

Nr\_of\_Cancelled\_Orders,

CAST(CAST(Nr\_of\_Cancelled\_Orders AS decimal(10,2))

/ CAST(Total\_Orders AS decimal(10,2)) \* 100 AS decimal(10,2)) AS Cancel\_Percent

FROM CANCELLED\_RATE\_2024

)

SELECT

cy.restaurant\_id,

ly.Cancel\_Percent Cancellation\_Percent\_of\_2023 ,

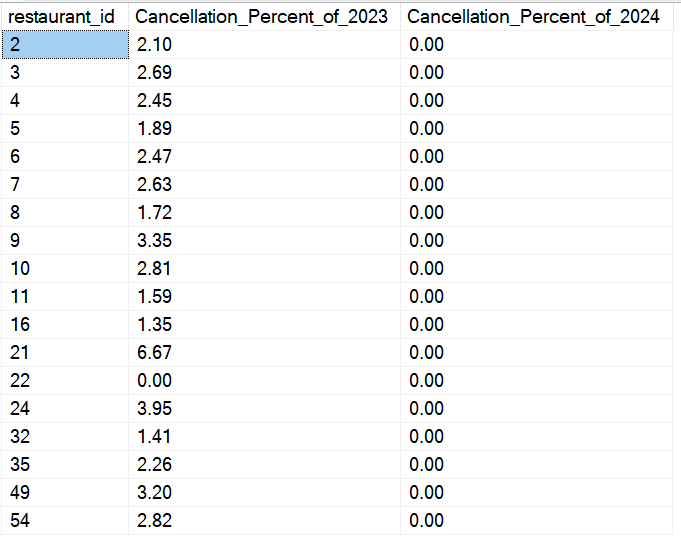
cy.Cancel\_Percent Cancel\_Percent\_of\_2024

FROM Current\_year\_Data cy

JOIN Last\_Year\_Data ly

ON cy.restaurant\_id = ly.restaurant\_id

ORDER BY cy.restaurant\_id



Q10 Rider Average Delivery Time

Determine each rider's average delivery time

WITH Riders\_Avg\_Delivery\_Time

AS

(

SELECT

r.rider\_id,

r.rider\_name,

o.order\_time,

d.delivery\_time,

CAST(CASE

WHEN d.delivery\_time < o.order\_time THEN (1440 – ABS(DATEDIFF(MINUTE,order\_time,d.delivery\_time)))

ELSE ABS(DATEDIFF(MINUTE,order\_time,d.delivery\_time))

END as decimal(10,2)) Time\_Taken\_to\_deliver

FROM Orders o

LEFT JOIN Deliveries d ON o.order\_id = d.order\_id

LEFT JOIN Riders r ON d.rider\_id = r.rider\_id

WHERE d.delivery\_status = 'Delivered'

)

SELECT

rider\_id,

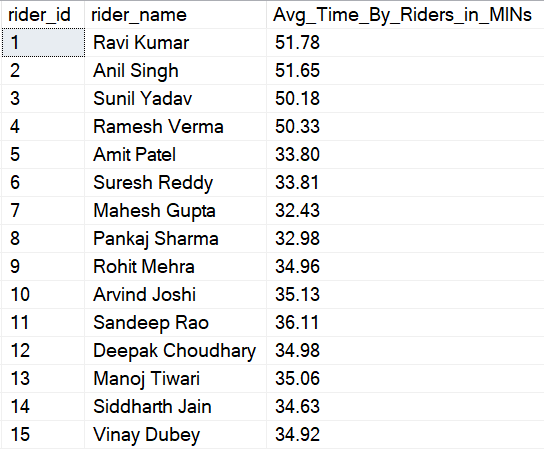
rider\_name,

CAST(ROUND(AVG(Time\_Taken\_to\_deliver),2) AS decimal(10,2)) AS Avg\_Time\_By\_Riders\_in\_MINs

FROM Riders\_Avg\_Delivery\_Time

GROUP BY rider\_id, rider\_name

ORDER BY rider\_id



Q11 Monthly Restaurant Growth Ratio:

Calculate each restaurant's growth ratio based on the total number of delivered orders since its joining

WITH Growth\_Rate\_of\_Delived\_Orders

AS

(

SELECT

o.restaurant\_id,

YEAR(o.order\_date) Order\_year,

MONTH(o.order\_date) Order\_Month,

FORMAT(o.order\_date, 'MMM yyyy') AS Month\_year,

CAST(COUNT(d.delivery\_id) AS decimal(10,2)) AS Current\_Month\_Orders\_Delivered,

CAST(LAG(COUNT(d.delivery\_id)) OVER(PARTITION BY o.restaurant\_id

ORDER BY YEAR(o.order\_date),MONTH(o.order\_date)) AS decimal(10,2)) Prev\_Month\_Orders\_delivered

FROM Orders o

LEFT JOIN Deliveries d

ON o.order\_id = d.order\_id

WHERE d.delivery\_status = 'Delivered'

GROUP BY

o.restaurant\_id,

YEAR(o.order\_date),

MONTH(o.order\_date),

FORMAT(o.order\_date, 'MMM yyyy')

)

SELECT

restaurant\_id,

Month\_year,

Current\_Month\_Orders\_Delivered,

Prev\_Month\_Orders\_delivered,

ROUND(CAST((Current\_Month\_Orders\_Delivered - Prev\_Month\_Orders\_delivered)/Prev\_Month\_Orders\_delivered \* 100 AS decimal(10,2)),2) as Grow\_Rate\_in\_Orders\_Delivered

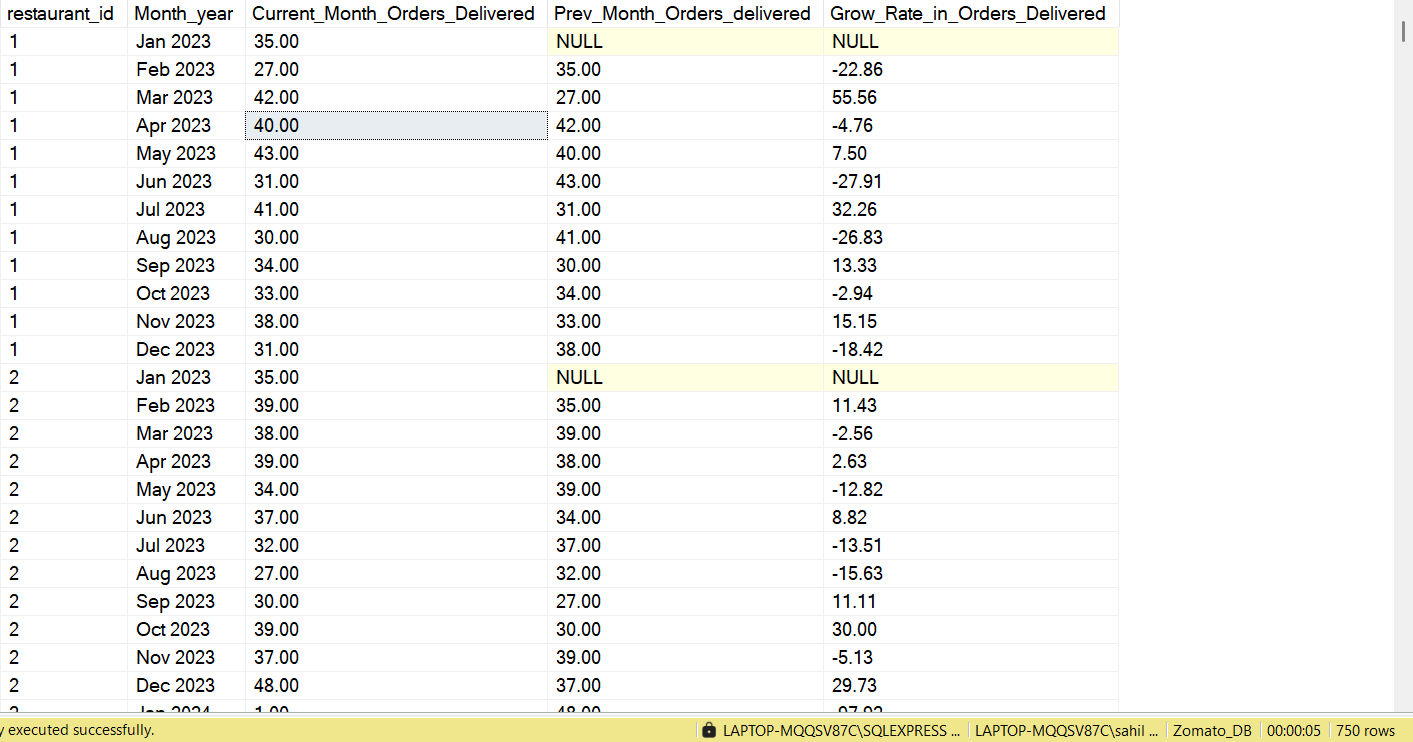
FROM Growth\_Rate\_of\_Delived\_Orders

ORDER BY

restaurant\_id,

Order\_year,

Order\_Month



Q12 Customer Segmentations:

(1) Segment Customers into "Gold" or "Silver" groups based on their total spending

(2) Compare to the Average Order Value

If Customer's total spending exceeds AOV Label them with gold other wise label them as silver

Write a Query to Determine each segment's total number of orders and total revenue

SELECT

Customer\_Category,

SUM(Total\_Spend) Total\_Revenue,

SUM(Nr\_of\_Orders) Total\_Orders

FROM

(

SELECT

c.customer\_name,

COUNT(o.order\_id) Nr\_of\_Orders,

SUM(o.total\_amount) Total\_Spend,

CASE

WHEN SUM(o.total\_amount)> (SELECT AVG(total\_amount) from Orders) THEN 'Gold'

ELSE 'Silver'

END as Customer\_Category

FROM Orders o

JOIN Customers c

ON c.customer\_id = o.customer\_id

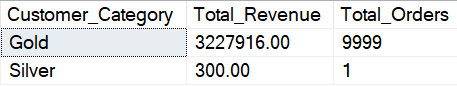
GROUP BY

c.customer\_id,

c.customer\_name

) as t2

GROUP BY Customer\_Category



Q13 Rider Monthly Earning:

Calculate each ride's total monthly earnings, assuming they earn 8% of the Delivered Order Amount

WITH Riders\_Monthly\_Earning

AS

(SELECT

rd.rider\_id ,

rd.rider\_name,

YEAR(o.order\_date) Order\_year,

Month(o.order\_date) Order\_Month,

FORMAT(o.order\_date,'MMMM yyyy') Month\_year,

CAST(SUM(total\_amount) \* 0.08 AS decimal(10,2)) Total\_Earning\_of\_Rider

FROM Orders o

LEFT JOIN Deliveries d

ON d.order\_id = o.order\_id

LEFT JOIN Riders rd

ON rd.rider\_id = d.rider\_id

WHERE d.delivery\_status = 'Delivered'

GROUP BY

rd.rider\_id,

rd.rider\_name,

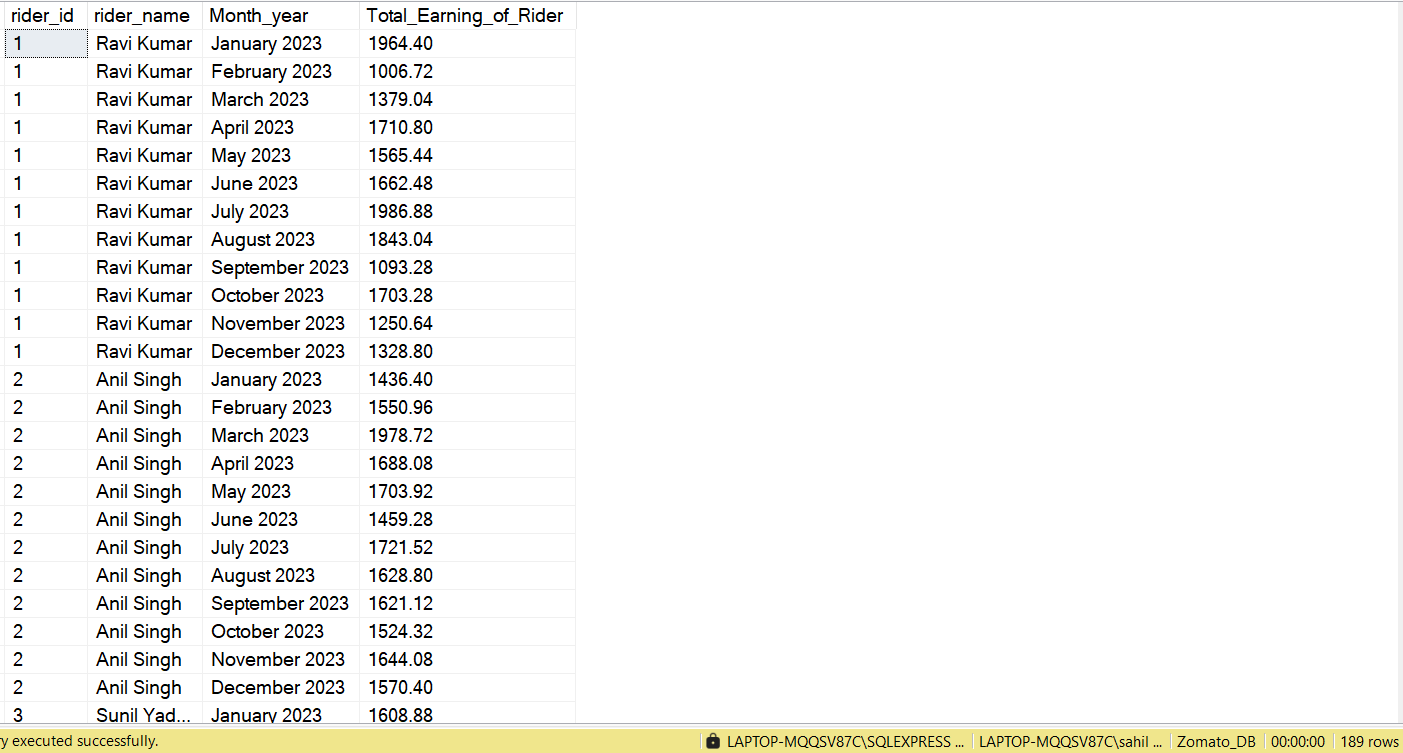
YEAR(o.order\_date),

Month(o.order\_date),

FORMAT(o.order\_date,'MMMM yyyy')

)

SELECT rider\_id, rider\_name, Month\_year, Total\_Earning\_of\_Rider FROM Riders\_Monthly\_Earning



Q 14 Rider Rating Analysis:

Find the number of 5 Star. 4 star, and 3 star rating Each riders has.

Riders recieve this rating based on delivery time

IF orders are delivered less than 15 Minutes of order recieved time the rider get 5 star rating.

IF they delivery is 15 to 20 Minute then they get a 4 star rating

IF they deliver after 20 Minute they get 3 star rating.

WITH Main\_cte

AS

(

SELECT

d.rider\_id as Rider\_id,

o.order\_time Order\_time,

d.delivery\_time,

CASE

WHEN d.delivery\_time < o.order\_time THEN (1440 - ABS(DATEDIFF(MINUTE,order\_time,d.delivery\_time)))

ELSE ABS(DATEDIFF(MINUTE,order\_time,d.delivery\_time))

END Time\_taken\_to\_Deliver

FROM Orders o

JOIN Deliveries d

ON o.order\_id = d.order\_id

WHERE d.delivery\_status = 'Delivered'

),

Final

AS

(

SELECT Rider\_id,

CASE

WHEN Time\_taken\_to\_Deliver <=15 THEN '5 star'

WHEN Time\_taken\_to\_Deliver > 15 AND Time\_taken\_to\_Deliver <= 20 THEN '4 Star'

ELSE '3 Star'

END STARS,

Time\_taken\_to\_Deliver

FROM Main\_cte

)

SELECT

Rider\_id,

STARS,

COUNT(STARS) Total\_Stars

FROM Final

GROUP BY

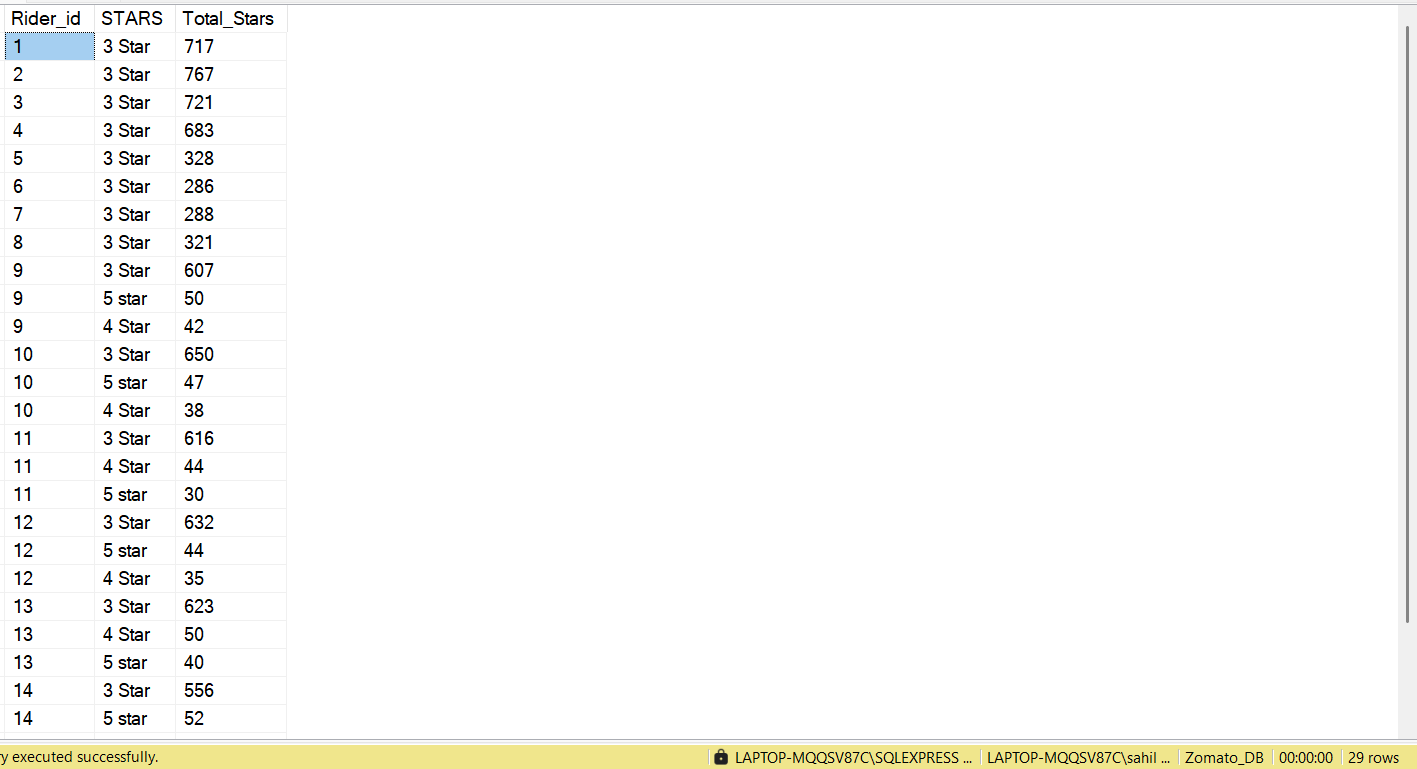
Rider\_id,

STARS

ORDER BY

Rider\_id,

COUNT(STARS) DESC



Q 15 Order Frequency by Day:

Analyze order fequency per day of the week and identify the peak day for each restaurant

WITH Peak\_day\_for\_Restaurant

AS

(

SELECT

o.restaurant\_id,

r.restaurant\_name as Restaurant,

DATEPART(WEEKDAY,o.order\_date) Week\_number,

DATENAME(WEEKDAY, o.order\_date) Weekday\_name,

COUNT(o.order\_id) Nr\_of\_Orders,

DENSE\_RANK() OVER(PARTITION BY o.restaurant\_id

ORDER BY COUNT(o.order\_id) DESC ) as Rank\_of\_Week\_Day

FROM Orders o

LEFT JOIN Restaurants r

ON r.restaurant\_id = o.restaurant\_id

GROUP BY o.restaurant\_id,

r.restaurant\_name,

DATENAME(WEEKDAY, o.order\_date),

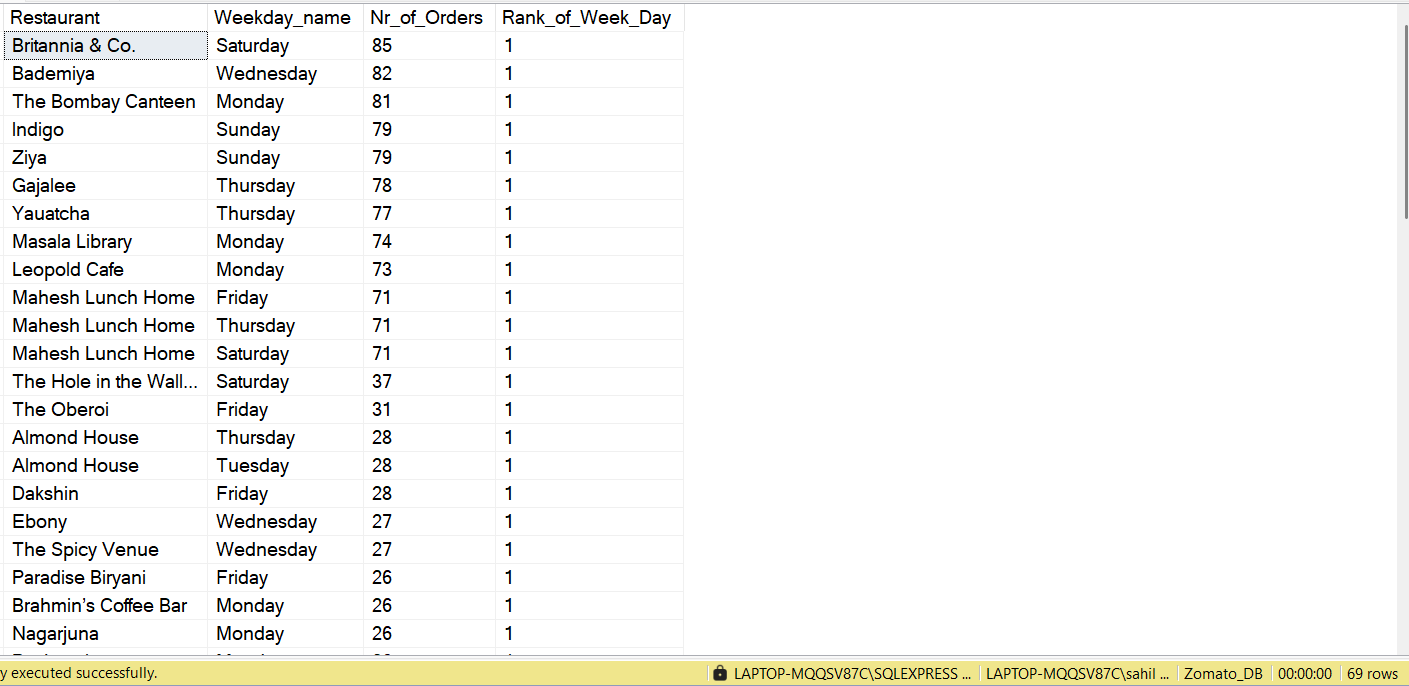
DATEPART(WEEKDAY,o.order\_date)

)

SELECT Restaurant, Weekday\_name, Nr\_of\_Orders, Rank\_of\_Week\_Day FROM Peak\_day\_for\_Restaurant

WHERE Rank\_of\_Week\_Day = 1

ORDER BY Nr\_of\_Orders DESC



Q16 Customer Lifetime value(CLV)

Calculate the Total Revenue Generated by each customer over all their orders

SELECT

c.customer\_id,

c.customer\_name,

SUM(o.total\_amount) Customer\_Lifetime\_Value FROM Orders o

JOIN Customers c

ON c.customer\_id = o.customer\_id

GROUP BY c.customer\_id, c.customer\_name

ORDER BY

SUM(o.total\_amount) DESC



Q 17 Monthly Sales Trends:

Identify Sales Trends by Comparing each month's total Sales to the previous months

WITH Monthly\_Sales\_Trends

AS

(

SELECT

YEAR(order\_date) Year,

MONTH(Order\_date) Month\_Number,

DATENAME(MONTH,Order\_date) Month\_Name,

SUM(total\_amount) Current\_Month\_Sales,

LAG(SUM(total\_amount)) OVER(ORDER BY MONTH(Order\_date)) Prev\_Month\_Sales

FROM Orders

GROUP BY

DATENAME(MONTH,Order\_date),

YEAR(order\_date),

MONTH(Order\_date)

)

SELECT

Year,

Month\_Name,

Prev\_Month\_Sales,

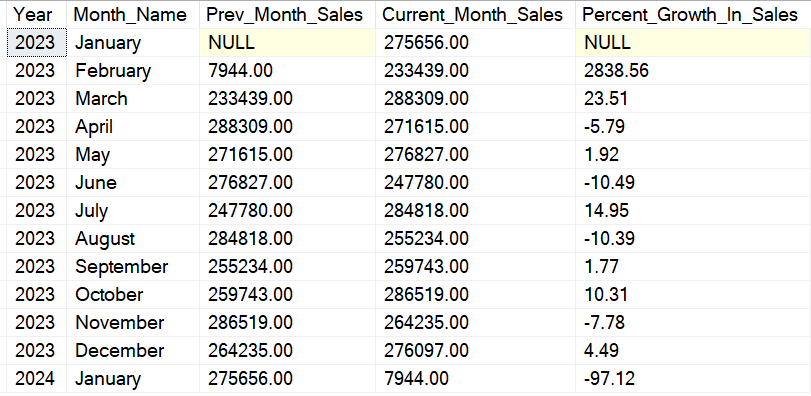
Current\_Month\_Sales,

CAST(ROUND((Current\_Month\_Sales- Prev\_Month\_Sales)/Prev\_Month\_Sales\* 100,2)AS decimal(10,2)) Percent\_Growth\_In\_Sales

FROM Monthly\_Sales\_Trends

ORDER BY

Year, Month\_Number



Q 18 Rider Effeciency

Evaluate rider Effeciency by determining Average Delivery times and Identifying those with lowest And highest Average Delivery time

SELECT -- By this You will get Minimum and Maximum Average time taken to deliver

MIN(Avg\_Time\_Taken\_to\_Deliver) Min\_Avg\_Time\_taken\_to\_deliver,

MAX(Avg\_Time\_Taken\_to\_Deliver) Max\_Avg\_Time\_taken\_to\_deliver

FROM

( -- By this Subquery you will get Average Time taken by Riders

SELECT Rider\_Id, Rider\_Name, CAST(ROUND(Avg\_Time\_taken\_to\_Deliver,2) AS decimal(10,2)) AS Avg\_Time\_Taken\_to\_Deliver FROM

(

SELECT

r.rider\_id as Rider\_Id,

rider\_name as Rider\_Name,

AVG(CAST(CASE

WHEN d.delivery\_time < o.order\_time THEN (1440 - ABS(DATEDIFF(MINUTE,order\_time,d.delivery\_time)))

ELSE ABS(DATEDIFF(MINUTE,order\_time,d.delivery\_time))

END as decimal(10,2))) Avg\_Time\_taken\_to\_Deliver

FROM Orders o

LEFT JOIN Deliveries d

ON o.order\_id = d.order\_id

LEFT JOIN Riders r

ON r.rider\_id = d.rider\_id

WHERE

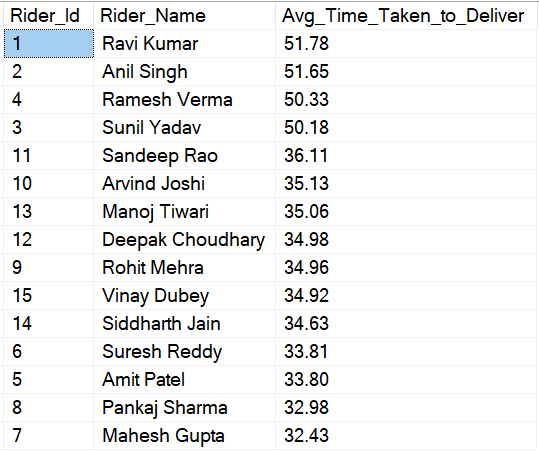
d.delivery\_status = 'Delivered'

GROUP BY r.rider\_id,rider\_name

) as t1

-- ORDER BY Avg\_Time\_taken\_to\_Deliver DESC

)t2





Q19 Order Item Popularity :

Track the Popularity of specific order items over time and identify seasonal demand spike

SELECT order\_item,Season, COUNT(order\_id) Nr\_of\_Orders

FROM (

SELECT

\*,

CASE

WHEN MONTH(order\_date) BETWEEN 3 AND 5 THEN 'Summer'

WHEN MONTH(order\_date) BETWEEN 6 AND 9 THEN 'Monsoon'

WHEN MONTH(order\_date) BETWEEN 10 AND 11 THEN 'Autumn'

WHEN MONTH(order\_date) IN ( 11, 12, 1, 2) THEN 'Winter'

END Season

FROM Orders

) t1

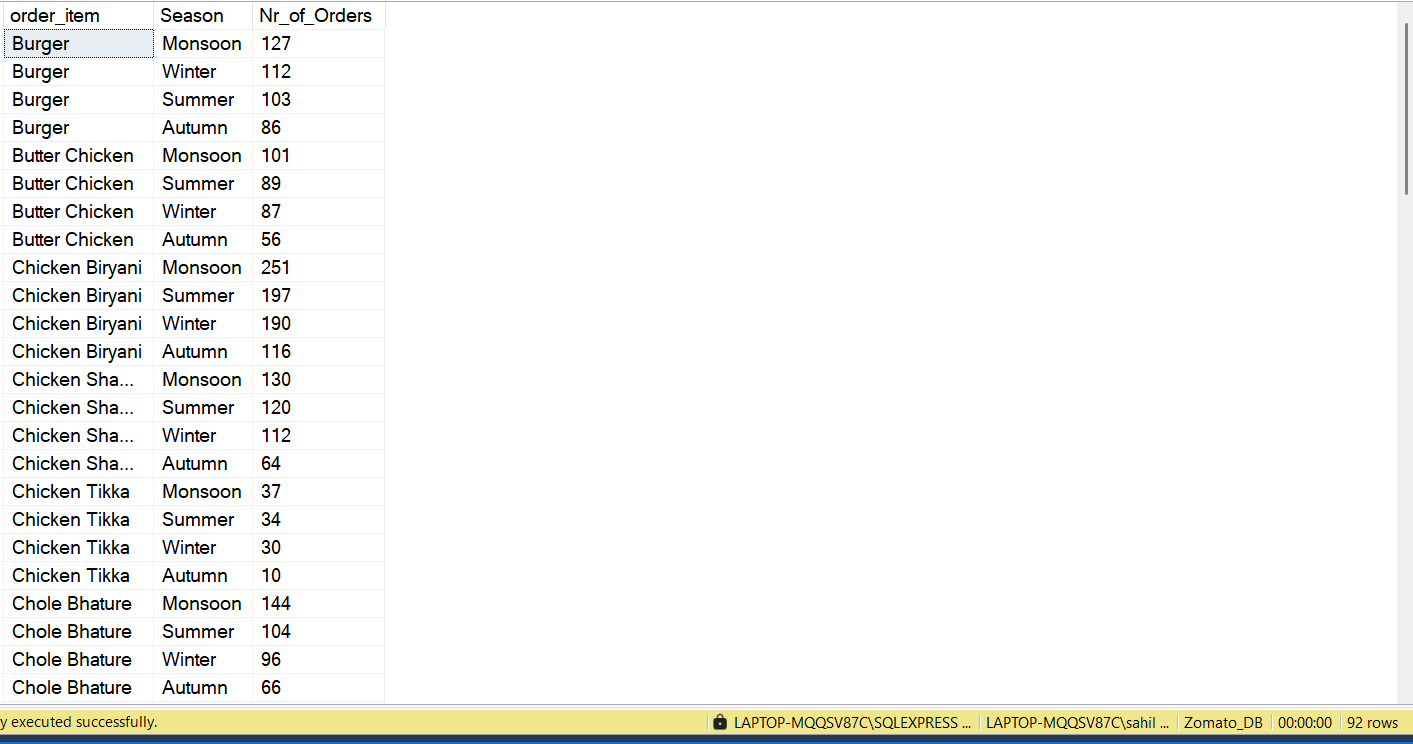
GROUP BY order\_item,

Season

ORDER BY

order\_item,

Nr\_of\_Orders DESC



Q 20 Rank each City based on the Total revenue for last year 2023

SELECT

r.city,

SUM(o.total\_amount) Total\_Revenue,

RANK() OVER(ORDER BY SUM(o.total\_amount) DESC) Rank\_of\_City\_by\_Revenue

FROM Orders o

LEFT JOIN Restaurants r

ON r.restaurant\_id = o.restaurant\_id

WHERE YEAR(order\_date) = 2023

GROUP BY r.city

