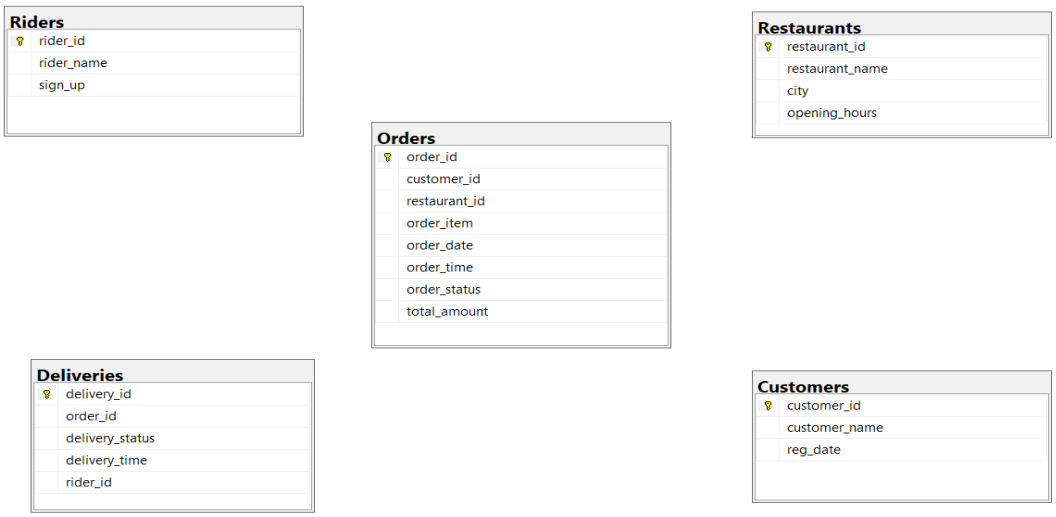
ZOMATO SQL ANALYSIS

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.



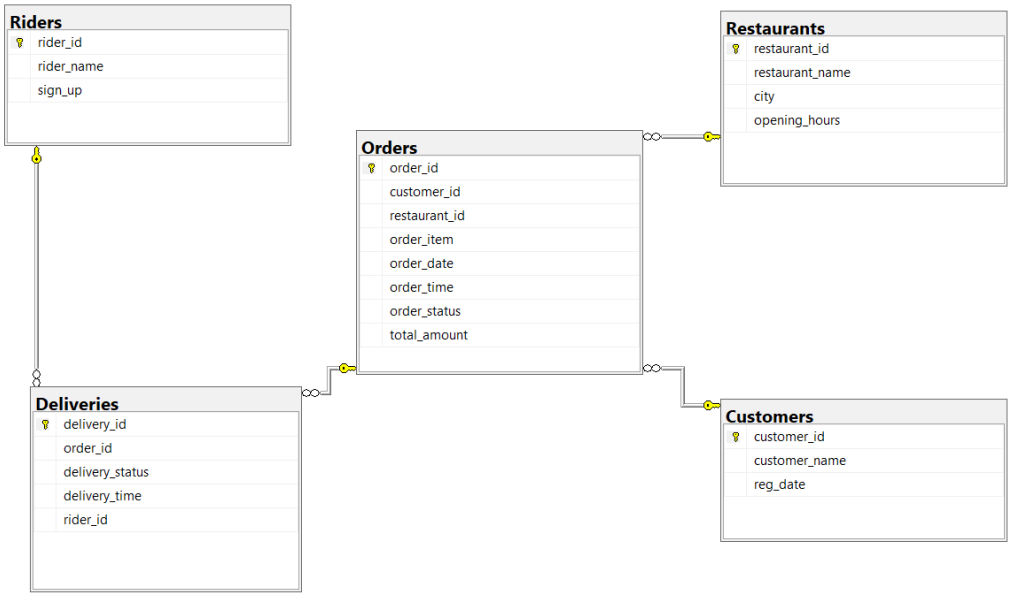

```
ALTER TABLE Deliveries
ADD CONSTRAINT FK_Deliveries_Orders
FOREIGN KEY (Order_id) REFERENCES Orders(order_id);

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
```

customer_id	customer_name	reg_date
1	Arjun Mehta	2023-03-10
2	Priya Sharma	2023-04-15
3	Vikram Singh	2023-05-01
4	Ritu Patel	2023-06-05
5	Aman Gupta	2023-07-12
6	Sneha Desai	2023-08-18
7	Rahul Verma	2023-09-05
8	Neha Joshi	2023-10-10
9	Karan Kapoor	2023-11-15
10	Divya Nair	2023-12-20
11	Rohan Iyer	2024-01-02
12	Anjali Saxena	2024-01-08
13	Sameer Khan	2024-01-12
14	Pooja Rao	2024-01-15
15	Nikhil Jain	2024-01-18
16	Aarti Yadav	2024-01-22
17	Manish Kulkarni	2024-01-26
18	Shreya Ghosh	2024-01-30
19	Aakash Dubey	2024-02-02
20	Bhavna Agarwal	2024-02-05
21	Ramesh Chan...	2024-02-08
22	Kavita Malhotra	2024-02-10
23	Ashish Mishra	2024-02-12
24	Megha Sinha	2024-02-15

executed successfully. LAPTOP-MQQSV87C\SQLEXPRESS ... LAPTOP-MQQSV87C\sahil ... Zomato_DB 00:00:00 33 rows

```
SELECT * FROM Orders
```

order_id	customer_id	restaurant_id	order_item	order_date	order_time	order_status	total_amount
1	13	6	Chicken Biryani	2024-01-01	12:30:00.0000000	Completed	242.00
2	2	5	Vegetable Fried Rice	2024-01-02	13:00:00.0000000	Completed	288.00
3	15	6	Chicken Biryani	2024-01-03	09:30:00.0000000	Completed	244.00
4	3	7	Prawn Masala	2024-01-04	14:15:00.0000000	Completed	477.00
5	1	35	Paneer Butter Masala	2024-01-05	19:00:00.0000000	Completed	293.00
6	12	5	Masala Dosa	2024-01-06	12:45:00.0000000	Completed	286.00
7	10	11	Mutton Rogan Josh	2024-01-07	18:30:00.0000000	Completed	267.00
8	1	5	Vegetable Fried Rice	2024-01-08	13:30:00.0000000	Completed	323.00
9	8	9	Egg Curry	2024-01-09	12:00:00.0000000	Completed	217.00
10	16	16	Chicken Biryani	2024-01-10	19:45:00.0000000	Completed	220.00
11	17	7	Paneer Butter Masala	2024-01-11	13:15:00.0000000	Completed	411.00
12	16	2	Vegetable Fried Rice	2024-01-12	10:00:00.0000000	Completed	228.00
13	4	21	Chicken Shawarma	2024-01-13	20:00:00.0000000	Completed	228.00
14	7	3	Vegetable Fried Rice	2024-01-14	12:30:00.0000000	Completed	320.00
15	17	8	Chicken Biryani	2024-01-15	13:00:00.0000000	Completed	236.00
16	19	24	Pizza	2024-01-16	19:30:00.0000000	Completed	493.00
17	14	49	Mutton Biryani	2024-01-17	18:00:00.0000000	Completed	423.00
18	1	4	Masala Dosa	2024-01-18	13:45:00.0000000	Completed	475.00
19	15	3	Chole Bhature	2024-01-19	14:00:00.0000000	Completed	410.00
20	16	54	Lamb Kebab	2024-01-20	20:15:00.0000000	Completed	495.00
21	15	32	Mutton Rogan Josh	2024-01-21	18:45:00.0000000	Completed	287.00
22	15	22	Paneer Butter Masala	2024-01-22	17:30:00.0000000	Completed	313.00
23	17	10	Masala Dosa	2024-01-23	13:00:00.0000000	Completed	283.00

executed successfully. LAPTOP-MQQSV87C\SQLEXPRESS ... LAPTOP-MQQSV87C\sahil ... Zomato_DB 00:00:00 10,000 rows

```
SELECT * FROM Riders
```

rider_id	rider_name	sign_up
1	Ravi Kumar	2023-01-05
2	Anil Singh	2023-02-10
3	Sunil Yadav	2023-03-12
4	Ramesh Verma	2023-04-15
5	Amit Patel	2023-05-18
6	Suresh Reddy	2023-06-20
7	Mahesh Gupta	2023-07-22
8	Pankaj Sharma	2023-08-25
9	Rohit Mehra	2023-09-05
10	Arvind Joshi	2023-10-10
11	Sandeep Rao	2023-11-15
12	Deepak Chou...	2023-12-01
13	Manoj Tiwari	2023-01-25
14	Siddharth Jain	2023-02-28
15	Vinay Dubey	2023-03-22
16	Ashok Malhotra	2023-04-30
17	Ravi Ranjan	2023-05-15
18	Naveen Nair	2023-06-05
19	Pawan Kumar	2023-07-12
20	Karthik Iyer	2023-08-08
21	Rajesh Shukla	2023-09-17
22	Gopal Das	2023-10-21
23	Lokesh Agra...	2023-11-28
24	Vikas Anand	2024-01-05
25	Mukesh Chan...	2024-01-12

executed successfully. LAPTOP-MQQSV87C\SQLEXPRESS ... LAPTOP-MQQSV87C\sahil ... Zomato_DB 00:00:00 34 rows

```
SELECT * FROM Deliveries
```

delivery_id	order_id	delivery_status	delivery_time	rider_id
1	658	Delivered	00:44:58.0000000	1
2	111	Delivered	00:44:48.0000000	6
3	8481	Not Delivered	NULL	6
4	4634	Delivered	00:29:42.0000000	4
5	8699	Delivered	00:44:39.0000000	13
6	2031	Delivered	00:26:38.0000000	4
7	2844	Delivered	00:23:32.0000000	13
8	4074	Delivered	00:25:15.0000000	3
9	3060	Delivered	00:37:52.0000000	6
10	8260	Delivered	00:31:33.0000000	12
11	2359	Delivered	00:27:17.0000000	6
12	4928	Delivered	00:36:07.0000000	7
13	228	Delivered	00:27:06.0000000	9
14	1325	Delivered	00:20:05.0000000	11
15	6422	Delivered	00:35:58.0000000	3
16	4938	Delivered	00:39:56.0000000	1
17	3439	Order	NULL	3
18	3939	Delivered	00:37:32.0000000	12
19	2416	Delivered	00:41:30.0000000	15
20	7023	Delivered	00:23:28.0000000	8
21	5889	Delivered	00:21:27.0000000	4
22	8831	Delivered	00:25:21.0000000	15
23	7514	Delivered	00:30:57.0000000	6
24	4113	Delivered	00:28:56.0000000	5
25	1111	Delivered	00:33:51.0000000	7

executed successfully. LAPTOP-MQQSV87C\SQLEXPRESS ... LAPTOP-MQQSV87C\sahil ... Zomato_DB 00:00:00 9,750 rows

```
SELECT * FROM Restaurants
```

restaurant_id	restaurant_name	city	opening_hours
1	The Bombay Canteen	Mumbai	10:00 AM - 11:00 PM
2	Leopold Cafe	Mumbai	9:00 AM - 12:00 AM
3	Bademiya	Mumbai	6:00 PM - 3:00 AM
4	Ziya	Mumbai	12:00 PM - 11:00 PM
5	Gajalee	Mumbai	11:00 AM - 11:00 PM
6	Masala Library	Mumbai	12:00 PM - 3:00 PM, 7:00 PM - 11:00 PM
7	Mahesh Lunch Home	Mumbai	11:30 AM - 11:45 PM
8	Yauatcha	Mumbai	12:00 PM - 12:00 AM
9	Britannia & Co.	Mumbai	12:00 PM - 4:00 PM
10	Indigo	Mumbai	12:00 PM - 3:30 PM, 7:00 PM - 11:45 PM
11	Indian Accent	Delhi	12:00 PM - 2:30 PM, 7:00 PM - 10:30 PM
12	Karim's	Delhi	9:00 AM - 12:00 AM
13	Bukhara	Delhi	12:30 PM - 2:45 PM, 7:00 PM - 11:45 PM
14	Moti Mahal	Delhi	12:00 PM - 11:30 PM
15	SodaBottleOpener...	Delhi	11:00 AM - 11:30 PM
16	Gulati	Delhi	11:30 AM - 11:30 PM
17	Saravana Bhavan	Delhi	8:00 AM - 10:00 PM
18	The Big Chill Cafe	Delhi	11:00 AM - 11:00 PM
19	Farzi Cafe	Delhi	12:00 PM - 11:45 PM
20	Lodi - The Garden ...	Delhi	12:00 PM - 12:00 AM
21	Perch Wine & Coffe...	Delhi	11:00 AM - 11:00 PM
22	Diggin	Delhi	11:00 AM - 11:00 PM
23	Cafe Delhi Heights	Delhi	11:00 AM - 12:00 AM
24	Olive Bar & Kitchen	Delhi	12:30 PM - 3:30 PM, 7:30 PM - 12:30 AM
25	Imly	Delhi	11:00 AM - 11:00 PM


```sql
SELECT * FROM Customers
WHERE
	customer_id IS NULL OR
	customer_name IS NULL OR
	reg_date IS NULL
```

customer_id	customer_name	reg_date

```sql
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
```

order_id	customer_id	restaurant_id	order_item	order_date	order_time	order_status	total_amount

```sql
SELECT * FROM Riders
WHERE
	rider_id IS NULL OR
	rider_name IS NULL OR
	sign_up IS NULL
```

rider_id	rider_name	sign_up

```sql
SELECT * FROM Restaurants
WHERE
	restaurant_name IS NULL OR
	opening_hours IS NULL OR
	city IS NULL
```

restaurant_id	restaurant_name	city	opening_hours

```sql
SELECT * FROM Deliveries
WHERE
	delivery_id IS NULL OR
	order_id IS NULL OR
	rider_id IS NULL OR
	delivery_status IS NULL
```

delivery_id	order_id	delivery_status	delivery_time	rider_id

```sql
The last order's date in the Orders table
SELECT MAX(order_date) Last_Order_Date FROM Orders
```

Last_Order_Date
2024-01-25

```sql
here we are considering Today's date as this
SELECT CAST(GETDATE() AS date) Todays_Date
```

Todays_Date
2025-01-22

```sql
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
```

CurrentDate
2023-01-22

```sql
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
```

customer_name	Dishes	Rank_of_dishes
Arjun Mehta	Masala Dosa	1
Arjun Mehta	Paneer Butter Masala	1
Arjun Mehta	Pasta Alfredo	2
Arjun Mehta	Chicken Biryani	3
Arjun Mehta	Mutton Rogan Josh	4
Arjun Mehta	Mutton Biryani	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
```

Time_Slot	Total_Orders
14:00 - 16:00	1188
18:00 - 20:00	1136
22:00 - 00:00	1123
12:00 - 14:00	1115
10:00 - 12:00	1107
20:00 - 22:00	1089
16:00 - 18:00	1080
08:00 - 10:00	1076
06:00 - 08:00	1074
00:00 - 02:00	12

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)

```sql
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
```

customer_id	customer_name	Avg_Order_Value
7	Rahul Verma	339.06
6	Sneha Desai	333.58
5	Aman Gupta	333.32

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

```sql
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
```

customer_id	customer_name	Total_Spent
6	Sneha Desai	269197.00
7	Rahul Verma	262094.00
5	Aman Gupta	257322.00
9	Karan Kapoor	244287.00
8	Neha Joshi	243223.00
4	Ritu Patel	242681.00
15	Nikhil Jain	168782.00
17	Manish Kulkarni	162552.00
22	Kavita Malhotra	154737.00
20	Bhavna Agarwal	154368.00
19	Aakash Dubey	150866.00
18	Shreya Ghosh	150807.00
16	Aarti Yadav	146145.00
21	Ramesh Chan...	143571.00

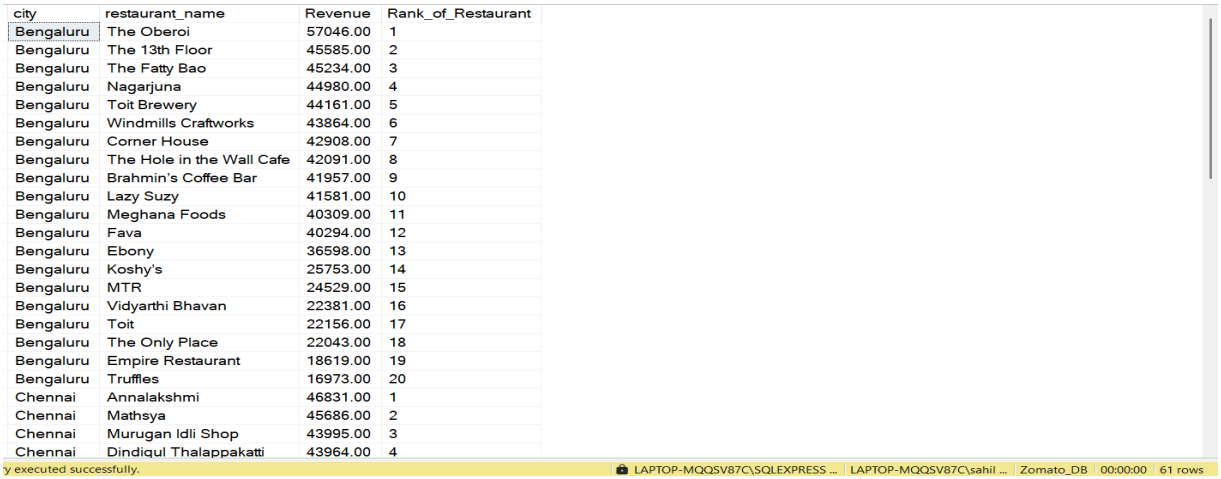
Q5 Orders without Delivery: Write query to find orders that were placed but not delivered.
-Return each restaurant name, city and number of not delivered orders

```sql
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;
```

restaurant_name	city	Total_Not_Delivered_Orders
Bademiya	Mumbai	43
Mahesh Lunch Home	Mumbai	43
Gajalee	Mumbai	41
Indigo	Mumbai	41
Britannia & Co.	Mumbai	40
The Bombay Canteen	Mumbai	37
Ziya	Mumbai	35
Leopold Cafe	Mumbai	35
Masala Library	Mumbai	34
Yauatcha	Mumbai	28
Dindigul Thalappakatti	Chennai	17
Mathsya	Chennai	15
Nagarjuna	Benga...	15
The Fatty Bao	Benga...	15
Shah Ghouse	Hyder...	15
The Spicy Venue	Hyder...	14
Ebony	Benga...	13
Murugan Idli Shop	Chennai	13
Dakshin	Chennai	13
The Oberoi	Benga...	12
The Hole in the Wall...	Benga...	11
The 13th Floor	Benga...	10
Corner House	Benga...	10
Fava	Benga...	10

y executed successfully. | LAPTOP-MQQSV87C\SQLEXPRESS ... | LAPTOP-MQQSV87C\sahil ... | Zomato_DB | 00:00:00 | 61 rows

```sql
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
```



city	restaurant_name	Revenue	Rank_of_Restaurant
Bengaluru	The Oberoi	57046.00	1
Bengaluru	The 13th Floor	45585.00	2
Bengaluru	The Fatty Bao	45234.00	3
Bengaluru	Nagarjuna	44980.00	4
Bengaluru	Toit Brewery	44161.00	5
Bengaluru	Windmills Craftworks	43864.00	6
Bengaluru	Corner House	42908.00	7
Bengaluru	The Hole in the Wall Cafe	42091.00	8
Bengaluru	Brahmin's Coffee Bar	41957.00	9
Bengaluru	Lazy Suzy	41581.00	10
Bengaluru	Meghana Foods	40309.00	11
Bengaluru	Fava	40294.00	12
Bengaluru	Ebony	36598.00	13
Bengaluru	Koshy's	25753.00	14
Bengaluru	MTR	24529.00	15
Bengaluru	Vidyarthi Bhavan	22381.00	16
Bengaluru	Toit	22156.00	17
Bengaluru	The Only Place	22043.00	18
Bengaluru	Empire Restaurant	18619.00	19
Bengaluru	Truffles	16973.00	20
Chennai	Annalakshmi	46831.00	1
Chennai	Mathsya	45686.00	2
Chennai	Murugan Idli Shop	43995.00	3
Chennai	Dindigul Thalappakatti	43964.00	4


```
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
```

City	Restaurant	Revenue	Rank_of_Restaurant
Bengaluru	The Oberoi	57046.00	1
Bengaluru	The 13th Floor	45585.00	2
Bengaluru	The Fatty Bao	45234.00	3
Chennai	Annalakshmi	46831.00	1
Chennai	Mathsya	45686.00	2
Chennai	Murugan Idli Shop	43995.00	3
Delhi	Saravana Bhavan	26265.00	1
Delhi	Lodi - The Garden Restaurant	25736.00	2
Delhi	Gulati	25565.00	3
Hyderabad	Ohri's Jiva Imperia	48289.00	1
Hyderabad	Almond House	45101.00	2
Hyderabad	Cafe Bahar	43872.00	3
Mumbai	Bademiya	156853.00	1
Mumbai	Gajalee	156265.00	2
Mumbai	Indigo	156184.00	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
```

```sql
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
```

city	Dishes	Nr_of_Orders	Rank_of_Dish
Mumbai	Paneer Butter Masala	363	1
Bengaluru	Chicken Biryani	172	1
Delhi	Paneer Butter Masala	97	1
Hyderabad	Chicken Biryani	81	1
Chennai	Mutton Rogan Josh	74	1

```sql
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
```

customer_id	customer_name	reg_date
5	Aman Gupta	2023-07-12
6	Sneha Desai	2023-08-18
9	Karan Kapoor	2023-11-15
11	Rohan Iyer	2024-01-02
18	Shreya Ghosh	2024-01-30
21	Ramesh Chandra	2024-02-08
22	Kavita Malhotra	2024-02-10
23	Ashish Mishra	2024-02-12
24	Megha Sinha	2024-02-15

```sql
Q9 Cancelled Rate Comparison:
Calculate and Compare the order Cancellation rate for each restaurant between the
currrent year and previous year
```

```sql
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
```

restaurant_id	Total_Orders	Nr_of_Cancelled_Orders	Cancel_Percent
6	485	12	2.47
3	483	13	2.69
9	478	16	3.35
5	477	9	1.89
2	476	10	2.10
1	474	12	2.53
8	466	8	1.72
10	463	13	2.81
7	457	12	2.63
4	449	11	2.45
39	175	4	2.29
59	148	5	3.38
36	145	2	1.38
56	143	4	2.80
51	143	3	2.10
54	142	4	2.82
60	142	2	1.41
53	140	4	2.86
37	139	4	2.88
44	139	4	2.88
50	138	4	2.90
47	137	2	1.46
38	136	5	3.68
55	135	6	4.44

ry executed successfully. | LAPTOP-MQQSV87C\SQLEXPRESS ... | LAPTOP-MQQSV87C\sahil ... | Zomato_DB | 00:00:00 | 61 rows

```sql
--  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
```

restaurant_id	Total_Orders	Nr_of_Cancelled_Orders	Cancel_rate
5	3	0	0.00
6	2	0	0.00
7	2	0	0.00
8	2	0	0.00
3	2	0	0.00
54	2	0	0.00
2	1	0	0.00
4	1	0	0.00
9	1	0	0.00
10	1	0	0.00
11	1	0	0.00
16	1	0	0.00
21	1	0	0.00
22	1	0	0.00
24	1	0	0.00
32	1	0	0.00
35	1	0	0.00
49	1	0	0.00

```
-- 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
```

restaurant_id	Cancellation_Percent_of_2023	Cancellation_Percent_of_2024
2	2.10	0.00
3	2.69	0.00
4	2.45	0.00
5	1.89	0.00
6	2.47	0.00
7	2.63	0.00
8	1.72	0.00
9	3.35	0.00
10	2.81	0.00
11	1.59	0.00
16	1.35	0.00
21	6.67	0.00
22	0.00	0.00
24	3.95	0.00
32	1.41	0.00
35	2.26	0.00
49	3.20	0.00
54	2.82	0.00

```
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
```

rider_id	rider_name	Avg_Time_By_Riders_in_MINs
1	Ravi Kumar	51.78
2	Anil Singh	51.65
3	Sunil Yadav	50.18
4	Ramesh Verma	50.33
5	Amit Patel	33.80
6	Suresh Reddy	33.81
7	Mahesh Gupta	32.43
8	Pankaj Sharma	32.98
9	Rohit Mehra	34.96
10	Arvind Joshi	35.13
11	Sandeep Rao	36.11
12	Deepak Choudhary	34.98
13	Manoj Tiwari	35.06
14	Siddharth Jain	34.63
15	Vinay Dubey	34.92

```
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
```

restaurant_id	Month_year	Current_Month_Orders_Delivered	Prev_Month_Orders_delivered	Grow_Rate_in_Orders_Delivered
1	Jan 2023	35.00	NULL	NULL
1	Feb 2023	27.00	35.00	-22.86
1	Mar 2023	42.00	27.00	55.56
1	Apr 2023	40.00	42.00	-4.76
1	May 2023	43.00	40.00	7.50
1	Jun 2023	31.00	43.00	-27.91
1	Jul 2023	41.00	31.00	32.26
1	Aug 2023	30.00	41.00	-26.83
1	Sep 2023	34.00	30.00	13.33
1	Oct 2023	33.00	34.00	-2.94
1	Nov 2023	38.00	33.00	15.15
1	Dec 2023	31.00	38.00	-18.42
2	Jan 2023	35.00	NULL	NULL
2	Feb 2023	39.00	35.00	11.43
2	Mar 2023	38.00	39.00	-2.56
2	Apr 2023	39.00	38.00	2.63
2	May 2023	34.00	39.00	-12.82
2	Jun 2023	37.00	34.00	8.82
2	Jul 2023	32.00	37.00	-13.51
2	Aug 2023	27.00	32.00	-15.63
2	Sep 2023	30.00	27.00	11.11
2	Oct 2023	39.00	30.00	30.00
2	Nov 2023	37.00	39.00	-5.13
2	Dec 2023	48.00	37.00	29.73

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
```

Customer_Category	Total_Revenue	Total_Orders
Gold	3227916.00	9999
Silver	300.00	1

```
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
```

rider_id	rider_name	Month_year	Total_Earning_of_Rider
1	Ravi Kumar	January 2023	1964.40
1	Ravi Kumar	February 2023	1006.72
1	Ravi Kumar	March 2023	1379.04
1	Ravi Kumar	April 2023	1710.80
1	Ravi Kumar	May 2023	1565.44
1	Ravi Kumar	June 2023	1662.48
1	Ravi Kumar	July 2023	1986.88
1	Ravi Kumar	August 2023	1843.04
1	Ravi Kumar	September 2023	1093.28
1	Ravi Kumar	October 2023	1703.28
1	Ravi Kumar	November 2023	1250.64
1	Ravi Kumar	December 2023	1328.80
2	Anil Singh	January 2023	1436.40
2	Anil Singh	February 2023	1550.96
2	Anil Singh	March 2023	1978.72
2	Anil Singh	April 2023	1688.08
2	Anil Singh	May 2023	1703.92
2	Anil Singh	June 2023	1459.28
2	Anil Singh	July 2023	1721.52
2	Anil Singh	August 2023	1628.80
2	Anil Singh	September 2023	1621.12
2	Anil Singh	October 2023	1524.32
2	Anil Singh	November 2023	1644.08
2	Anil Singh	December 2023	1570.40
3	Sunil Yad...	January 2023	1608.88

y executed successfully. LAPTOP-MQQSV87C\SQLEXPRESS ... LAPTOP-MQQSV87C\sahil ... Zomato_DB 00:00:00 189 rows

```
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
```

Rider_id	STARS	Total_Stars
1	3 Star	717
2	3 Star	767
3	3 Star	721
4	3 Star	683
5	3 Star	328
6	3 Star	286
7	3 Star	288
8	3 Star	321
9	3 Star	607
9	5 star	50
9	4 Star	42
10	3 Star	650
10	5 star	47
10	4 Star	38
11	3 Star	616
11	4 Star	44
11	5 star	30
12	3 Star	632
12	5 star	44
12	4 Star	35
13	3 Star	623
13	4 Star	50
13	5 star	40
14	3 Star	556
14	5 star	52

y executed successfully. LAPTOP-MQQSV87C\SQLEXPRESS ... LAPTOP-MQQSV87C\sahil ... Zomato_DB 00:00:00 29 rows

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
```

Restaurant	Weekday_name	Nr_of_Orders	Rank_of_Week_Day
Britannia & Co.	Saturday	85	1
Bademiya	Wednesday	82	1
The Bombay Canteen	Monday	81	1
Indigo	Sunday	79	1
Ziya	Sunday	79	1
Gajalee	Thursday	78	1
Yauatcha	Thursday	77	1
Masala Library	Monday	74	1
Leopold Cafe	Monday	73	1
Mahesh Lunch Home	Friday	71	1
Mahesh Lunch Home	Thursday	71	1
Mahesh Lunch Home	Saturday	71	1
The Hole in the Wall...	Saturday	37	1
The Oberoi	Friday	31	1
Almond House	Thursday	28	1
Almond House	Tuesday	28	1
Dakshin	Friday	28	1
Ebony	Wednesday	27	1
The Spicy Venue	Wednesday	27	1
Paradise Biryani	Friday	26	1
Brahmin's Coffee Bar	Monday	26	1
Nagarjuna	Monday	26	1

y executed successfully. | LAPTOP-MQQSV87C\SQLEXPRESS ... | LAPTOP-MQQSV87C\sahil ... | Zomato_DB | 00:00:00 | 69 rows

```
 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
```

customer_id	customer_name	Customer_Lifetime_Value
6	Sneha Desai	269197.00
7	Rahul Verma	262094.00
5	Aman Gupta	257322.00
9	Karan Kapoor	244287.00
8	Neha Joshi	243223.00
4	Ritu Patel	242681.00
15	Nikhil Jain	168782.00
17	Manish Kulkarni	162552.00
22	Kavita Malhotra	154737.00
20	Bhavna Agarwal	154368.00
19	Aakash Dubey	150866.00
18	Shreya Ghosh	150807.00
16	Aarti Yadav	146145.00
21	Ramesh Chan...	143571.00
1	Arjun Mehta	66286.00
12	Anjali Saxena	64788.00
3	Vikram Singh	59750.00
10	Divya Nair	59235.00
11	Rohan Iyer	58870.00
13	Sameer Khan	56233.00
2	Priya Sharma	55936.00
14	Pooja Rao	55439.00
23	Ashish Mishra	747.00
24	Megha Sinha	300.00

y executed successfully. LAPTOP-MQQSV87C\SQLEXPRESS ... LAPTOP-MQQSV87C\sahil ... Zomato_DB 00:00:00 24 rows

```
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
```

Year	Month_Name	Prev_Month_Sales	Current_Month_Sales	Percent_Growth_In_Sales
2023	January	NULL	275656.00	NULL
2023	February	7944.00	233439.00	2838.56
2023	March	233439.00	288309.00	23.51
2023	April	288309.00	271615.00	-5.79
2023	May	271615.00	276827.00	1.92
2023	June	276827.00	247780.00	-10.49
2023	July	247780.00	284818.00	14.95
2023	August	284818.00	255234.00	-10.39
2023	September	255234.00	259743.00	1.77
2023	October	259743.00	286519.00	10.31
2023	November	286519.00	264235.00	-7.78
2023	December	264235.00	276097.00	4.49
2024	January	275656.00	7944.00	-97.12

```
 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
```

Rider_Id	Rider_Name	Avg_Time_Taken_to_Deliver
1	Ravi Kumar	51.78
2	Anil Singh	51.65
4	Ramesh Verma	50.33
3	Sunil Yadav	50.18
11	Sandeep Rao	36.11
10	Arvind Joshi	35.13
13	Manoj Tiwari	35.06
12	Deepak Choudhary	34.98
9	Rohit Mehra	34.96
15	Vinay Dubey	34.92
14	Siddharth Jain	34.63
6	Suresh Reddy	33.81
5	Amit Patel	33.80
8	Pankaj Sharma	32.98
7	Mahesh Gupta	32.43

Min_Avg_Time_taken_to_deliver	Max_Avg_Time_taken_to_deliver
32.43	51.78

Q19 Order Item Popularity :
Track the Popularity of specific order items over time and identify seasonal demand spike

```sql
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
```

order_item	Season	Nr_of_Orders
Burger	Monsoon	127
Burger	Winter	112
Burger	Summer	103
Burger	Autumn	86
Butter Chicken	Monsoon	101
Butter Chicken	Summer	89
Butter Chicken	Winter	87
Butter Chicken	Autumn	56
Chicken Biryani	Monsoon	251
Chicken Biryani	Summer	197
Chicken Biryani	Winter	190
Chicken Biryani	Autumn	116
Chicken Sha...	Monsoon	130
Chicken Sha...	Summer	120
Chicken Sha...	Winter	112
Chicken Sha...	Autumn	64
Chicken Tikka	Monsoon	37
Chicken Tikka	Summer	34
Chicken Tikka	Winter	30
Chicken Tikka	Autumn	10
Chole Bhature	Monsoon	144
Chole Bhature	Summer	104
Chole Bhature	Winter	96
Chole Bhature	Autumn	66

y executed successfully. LAPTOP-MQQSV87C\SQLEXPRESS ... LAPTOP-MQQSV87C\sahil ... Zomato_DB 00:00:00 92 rows

```
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
```

city	Total_Revenue	Rank_of_City_by_Revenue
Mumbai	1517450.00	1
Bengaluru	719062.00	2
Delhi	393896.00	3
Hyderabad	305467.00	4
Chennai	284397.00	5