SWIGGY

FOOD DELIVERY APP

SWIGGY EDA

By Yuvaraj M

About the Company

Swiggy is one of India's leading food delivery platforms, founded in 2014 and headquartered in Bengaluru. It allows users to order food from local restaurants via its app and also offers grocery delivery through Swiggy Instamart and parcel services via Swiggy Genie. The company operates in over 500 cities and serves millions of customers across the country. Backed by major investors, Swiggy has expanded into quick commerce and acquired Dineout to enter the dining-out space. Its main competitor is Zomato, and both dominate India's online food delivery market.

EDA PART 2 ANALYSIS

Solving Business Problems

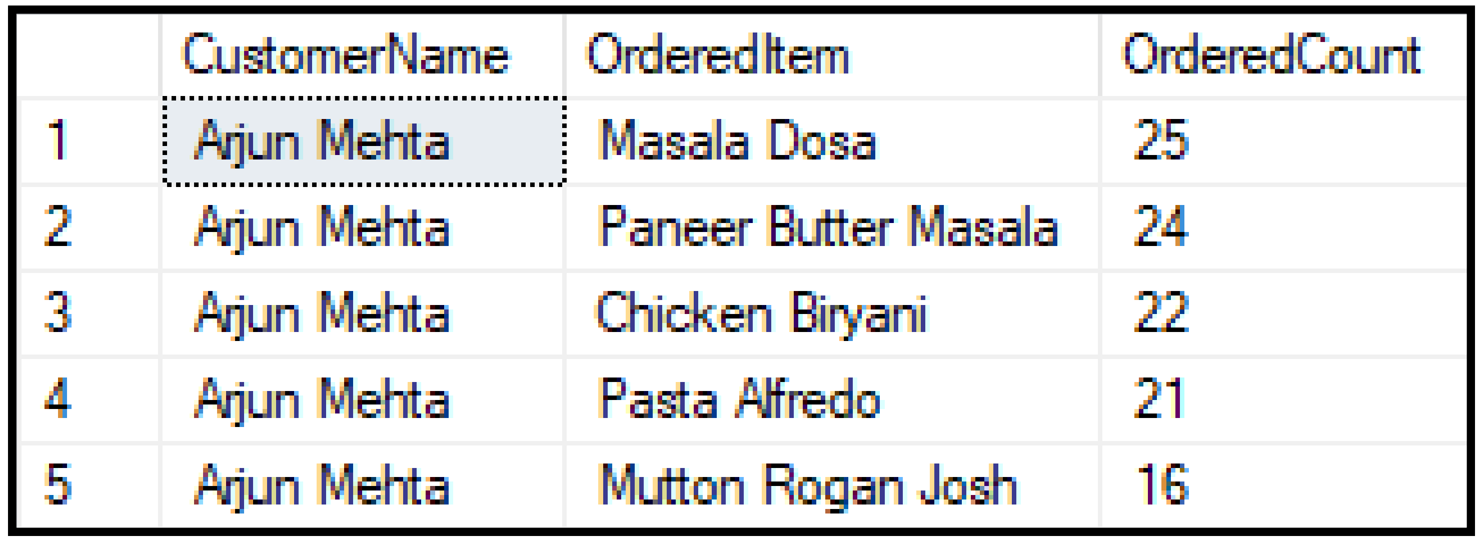
Solving Business Problems

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

Query

```
SELECT TOP 5
   c.customer_name AS CustomerName,
   o.order_item AS OrderedItem,
   COUNT(o.order_id) AS OrderedCount
FROM orders o
JOIN customers c ON o.customer_id = c.customer_id
WHERE
   c.customer_name = 'Arjun Mehta'
   AND YEAR(o.order_date) >= YEAR(GETDATE()) - 2
GROUP BY c.customer_name, o.order_item
order by count(order_id) desc
```

Output



	CustomerName	OrderedItem	OrderedCount
1	Arjun Mehta	Masala Dosa	25
2	Arjun Mehta	Paneer Butter Masala	24
3	Arjun Mehta	Chicken Biryani	22
4	Arjun Mehta	Pasta Alfredo	21
5	Arjun Mehta	Mutton Rogan Josh	16

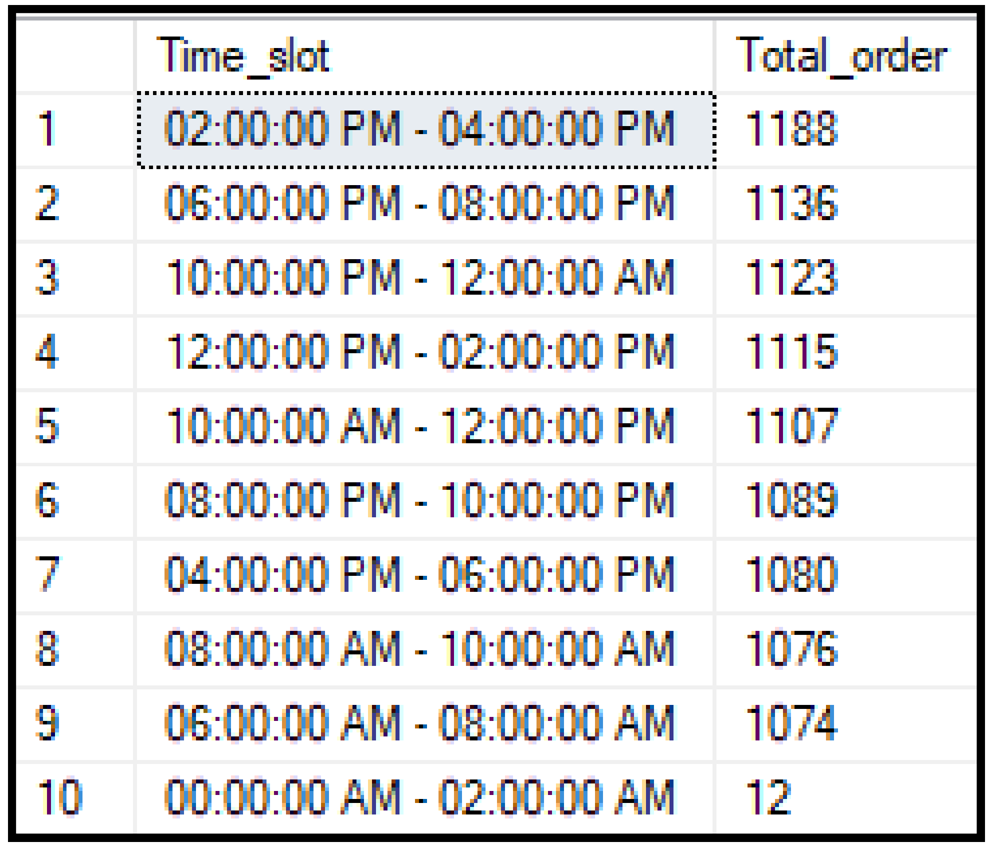
Solving Business Problems

->Identify the time slots during which the most orders are placed, based on 2-hour intervals

Query

```
WITH TimeSlot AS (
  SELECT
    order_id,
    CASE
      WHEN DATEPART(HOUR, order_time) BETWEEN 0 AND 1 THEN '00:00:00 AM - 02:00:00 AM'
      WHEN DATEPART(HOUR, order_time) BETWEEN 2 AND 3 THEN '02:00:00 AM - 04:00:00 AM'
      WHEN DATEPART(HOUR, order_time) BETWEEN 4 AND 5 THEN '04:00:00 AM - 06:00:00 AM'
      WHEN DATEPART(HOUR, order_time) BETWEEN 6 AND 7 THEN '06:00:00 AM - 08:00:00 AM'
      WHEN DATEPART(HOUR, order_time) BETWEEN 8 AND 9 THEN '08:00:00 AM - 10:00:00 AM'
      WHEN DATEPART(HOUR, order_time) BETWEEN 10 AND 11 THEN '10:00:00 AM - 12:00:00 PM'
      WHEN DATEPART(HOUR, order_time) BETWEEN 12 AND 13 THEN '12:00:00 PM - 02:00:00 PM'
      WHEN DATEPART(HOUR, order_time) BETWEEN 14 AND 15 THEN '02:00:00 PM - 04:00:00 PM'
      WHEN DATEPART(HOUR, order_time) BETWEEN 16 AND 17 THEN '04:00:00 PM - 06:00:00 PM'
      WHEN DATEPART(HOUR, order_time) BETWEEN 18 AND 19 THEN '06:00:00 PM - 08:00:00 PM'
      WHEN DATEPART(HOUR, order_time) BETWEEN 20 AND 21 THEN '08:00:00 PM - 10:00:00 PM'
      WHEN DATEPART(HOUR, order_time) BETWEEN 22 AND 23 THEN '10:00:00 PM - 12:00:00 AM'
    END AS Time_slot
  FROM orders
)
SELECT
  Time_slot,
  COUNT(order_id) AS Total_order
FROM TimeSlot
GROUP BY Time_slot
ORDER BY Total_order DESC;
```

Output



	Time_slot	Total_order
1	02:00:00 PM - 04:00:00 PM	1188
2	06:00:00 PM - 08:00:00 PM	1136
3	10:00:00 PM - 12:00:00 AM	1123
4	12:00:00 PM - 02:00:00 PM	1115
5	10:00:00 AM - 12:00:00 PM	1107
6	08:00:00 PM - 10:00:00 PM	1089
7	04:00:00 PM - 06:00:00 PM	1080
8	08:00:00 AM - 10:00:00 AM	1076
9	06:00:00 AM - 08:00:00 AM	1074
10	00:00:00 AM - 02:00:00 AM	12

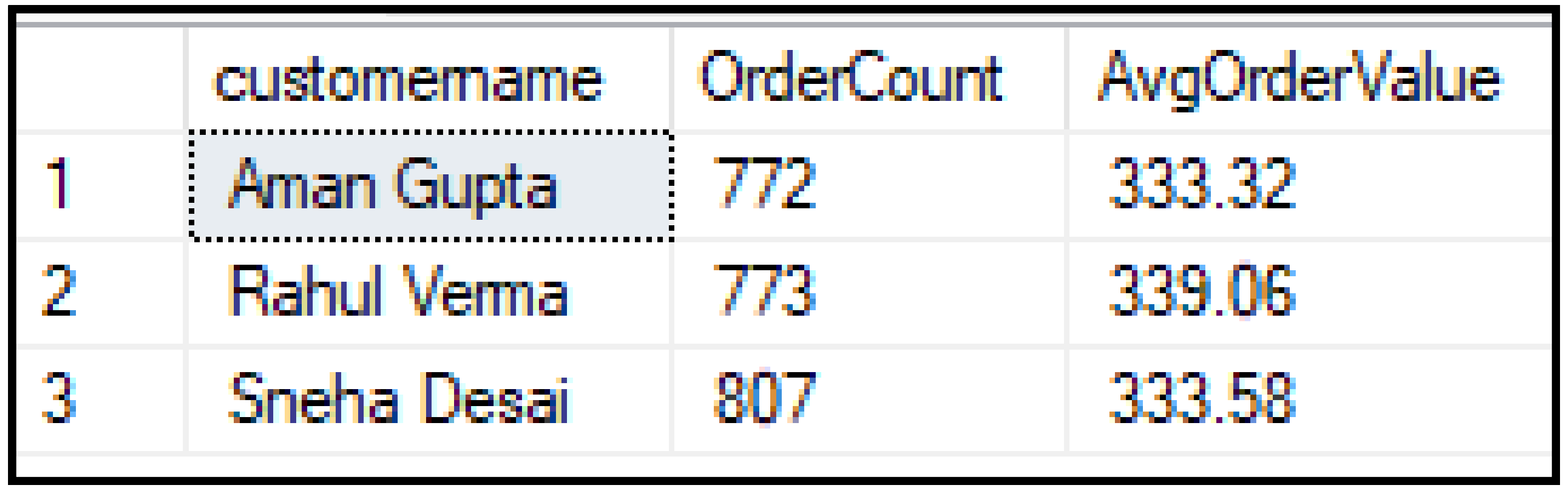
Solving Business Problems

->Find the average order value (AOV) per customer who has placed more than 750 orders

Query

```
SELECT
 C.customer_name as customername ,
 count(o.order_id) as OrderCount,
 ROUND(AVG(o.total_amount),2) as AvgOrderValue
FROM orders o join customers c
on o.customer_id = c.customer_id
GROUP BY C.customer_name
HAVING count(o.order_id) > 750
```

Output



	customername	OrderCount	AvgOrderValue
1	Aman Gupta	772	333.32
2	Rahul Verma	773	339.06
3	Sneha Desai	807	333.58

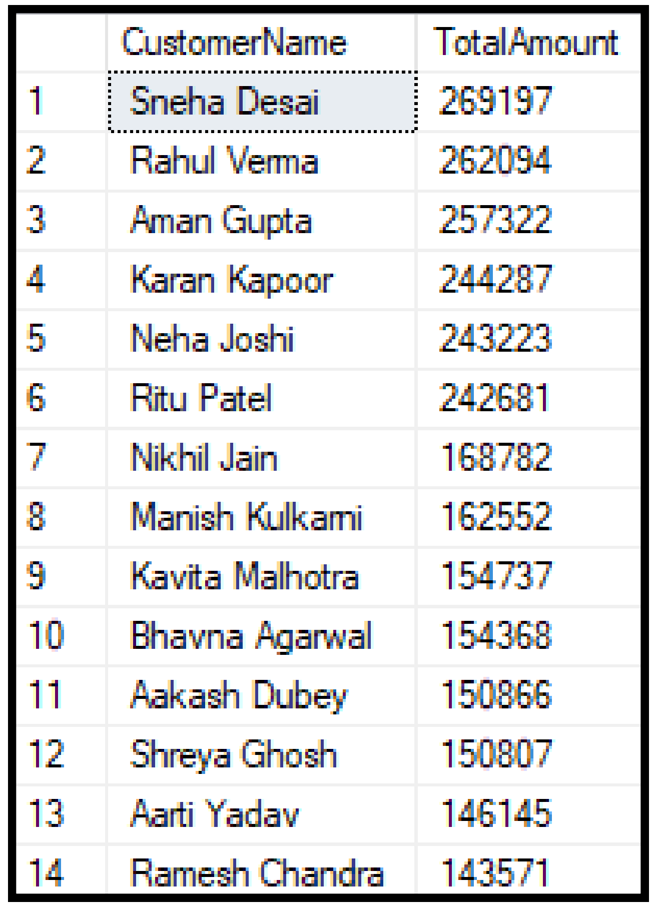
Solving Business Problems

->List the customers who have spent more than 100K in total on food orders.

Query

```
SELECT
c.customer_name as CustomerName,
SUM(Total_amount) as TotalAmount
FROM orders o join customers c on o.customer_id = c.customer_id
GROUP BY c.customer_name
HAVING SUM(Total_amount) > 100000
ORDER BY TotalAmount DESC
```

Output



	CustomerName	TotalAmount
1	Sneha Desai	269197
2	Rahul Verma	262094
3	Aman Gupta	257322
4	Karan Kapoor	244287
5	Neha Joshi	243223
6	Ritu Patel	242681
7	Nikhil Jain	168782
8	Manish Kulkarni	162552
9	Kavita Malhotra	154737
10	Bhavna Agarwal	154368
11	Aakash Dubey	150866
12	Shreya Ghosh	150807
13	Aarti Yadav	146145
14	Ramesh Chandra	143571

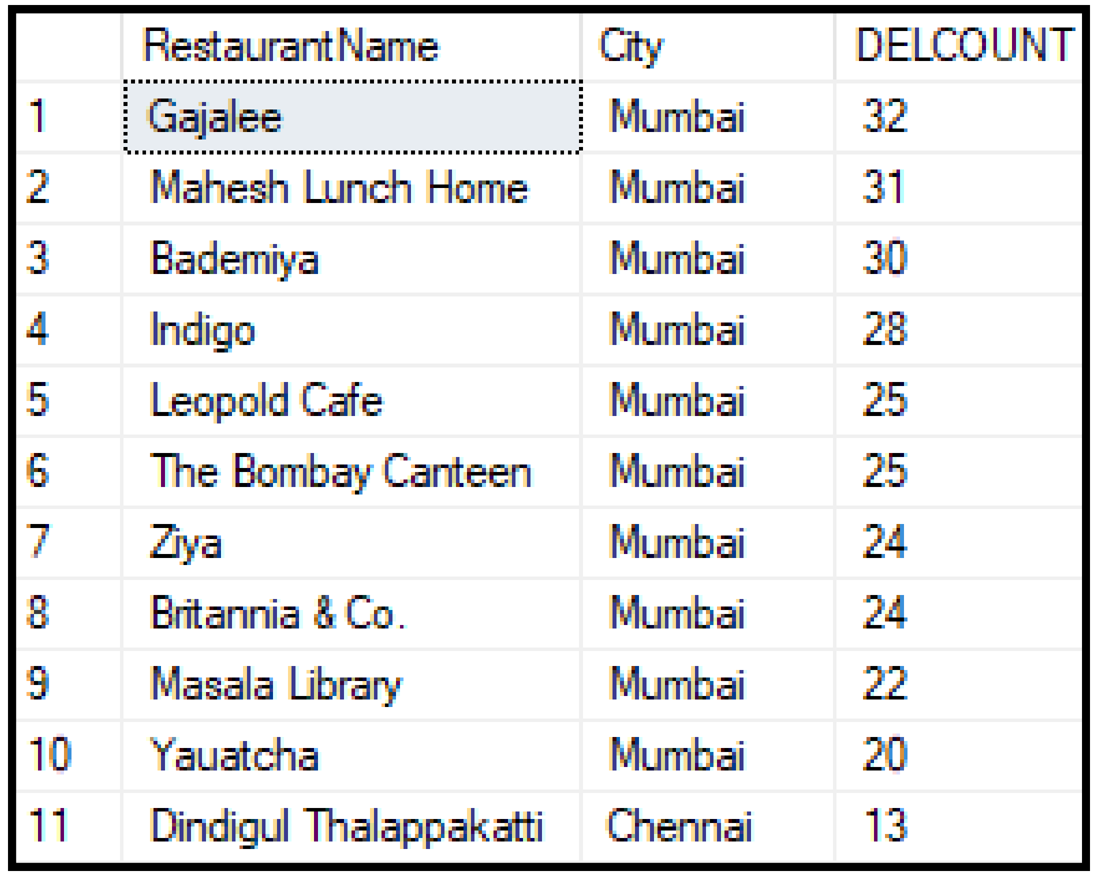
Solving Business Problems

-> Write a query to find orders that were placed but not delivered. Return: restaurant_name, city, and the number of not delivered orders.

Query

```
SELECT
   r.restaurant_name AS RestaurantName,
   r.city AS City,
   SUM (CASE
          WHEN d.delivery_status = 'Not Delivered'
          THEN 1 ELSE 0
       END) AS DeliveryCount
FROM orders o
JOIN deliveries d ON o.order_id = d.order_id
JOIN restaurants r ON o.restaurant_id = r.restaurant_id
where order_status = 'Completed' and delivery_status='Not Delivered'
GROUP BY r.restaurant_name, r.city
ORDER BY DeliveryCount DESC;
```

Output



	RestaurantName	City	DELCOUNT
1	Gajalee	Mumbai	32
2	Mahesh Lunch Home	Mumbai	31
3	Bademiya	Mumbai	30
4	Indigo	Mumbai	28
5	Leopold Cafe	Mumbai	25
6	The Bombay Canteen	Mumbai	25
7	Ziya	Mumbai	24
8	Britannia & Co.	Mumbai	24
9	Masala Library	Mumbai	22
10	Yauatcha	Mumbai	20
11	Dindigul Thalappakatti	Chennai	13

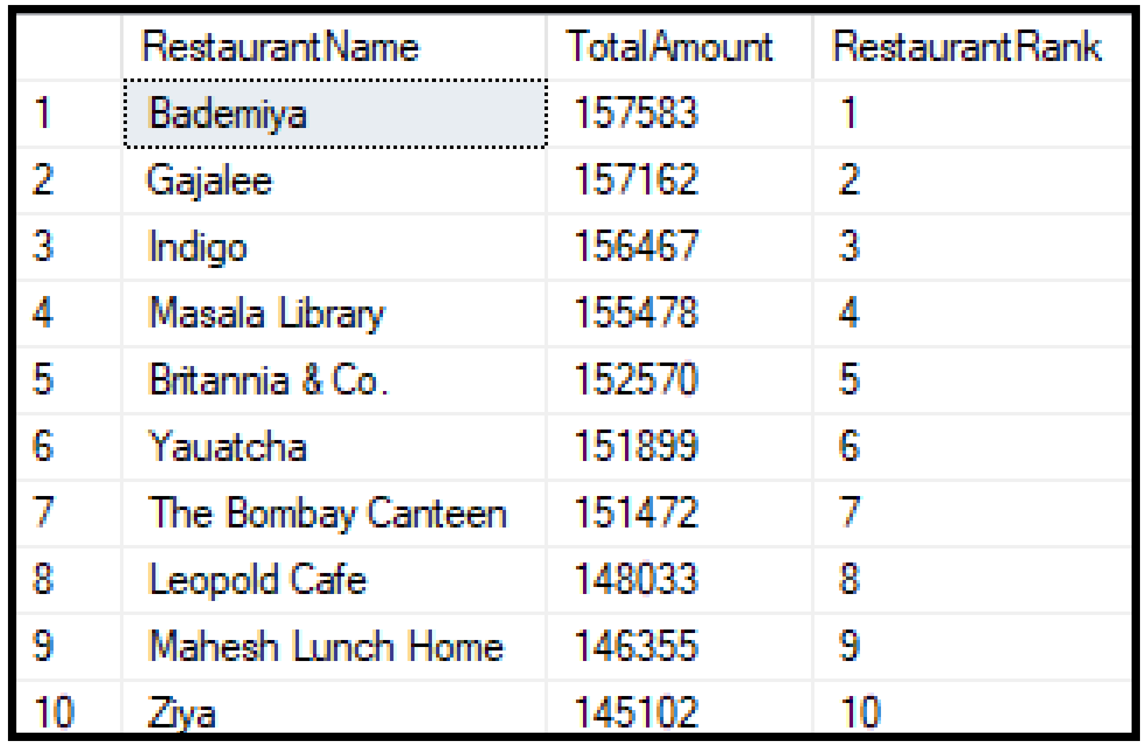
Solving Business Problems

->Rank restaurants by their total revenue from the last 2 year.

Query

```
SELECT
 r.restaurant_name as RestaurantName,
 sum(O.total_amount) as TotalAmount,
 RANK() OVER(ORDER BY sum(O.total_amount) DESC ) as RestaurantRank
FROM orders o join restaurants r
on o.restaurant_id = r.restaurant_id
GROUP BY r.restaurant_name
HAVING sum(O.total_amount) IS NOT NULL
```

Output



	RestaurantName	TotalAmount	RestaurantRank
1	Bademiya	157583	1
2	Gajalee	157162	2
3	Indigo	156467	3
4	Masala Library	155478	4
5	Britannia & Co.	152570	5
6	Yauatcha	151899	6
7	The Bombay Canteen	151472	7
8	Leopold Cafe	148033	8
9	Mahesh Lunch Home	146355	9
10	Ziya	145102	10

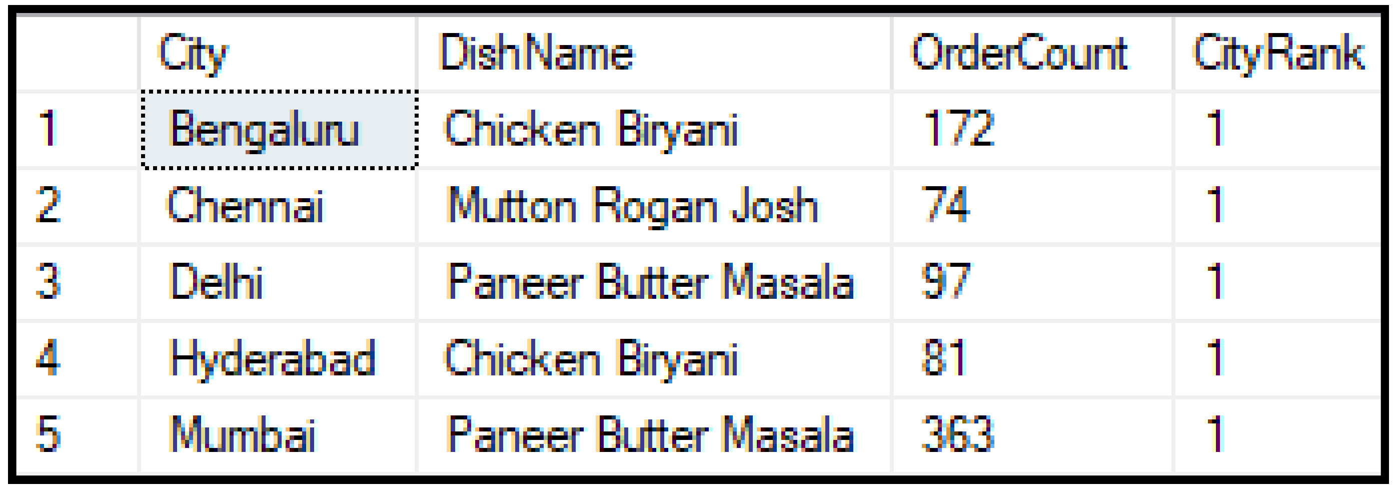
Solving Business Problems

->Identify the most popular dish in each city based on the number of orders

Query

```
WITH cityranks AS (
   SELECT
      r.city AS City,
      o.order_item AS DishName,
      COUNT(o.order_id) AS OrderCount,
      DENSE_RANK() OVER (
         PARTITION BY r.city
         ORDER BY COUNT(o.order_id) DESC
      ) AS CityRank
   FROM orders o
   JOIN restaurants r ON o.restaurant_id = r.restaurant_id
   GROUP BY r.city, o.order_item
)
SELECT *
FROM cityranks
WHERE CityRank = 1;
```

Output



	City	DishName	OrderCount	CityRank
1	Bengaluru	Chicken Biryani	172	1
2	Chennai	Mutton Rogan Josh	74	1
3	Delhi	Paneer Butter Masala	97	1
4	Hyderabad	Chicken Biryani	81	1
5	Mumbai	Paneer Butter Masala	363	1

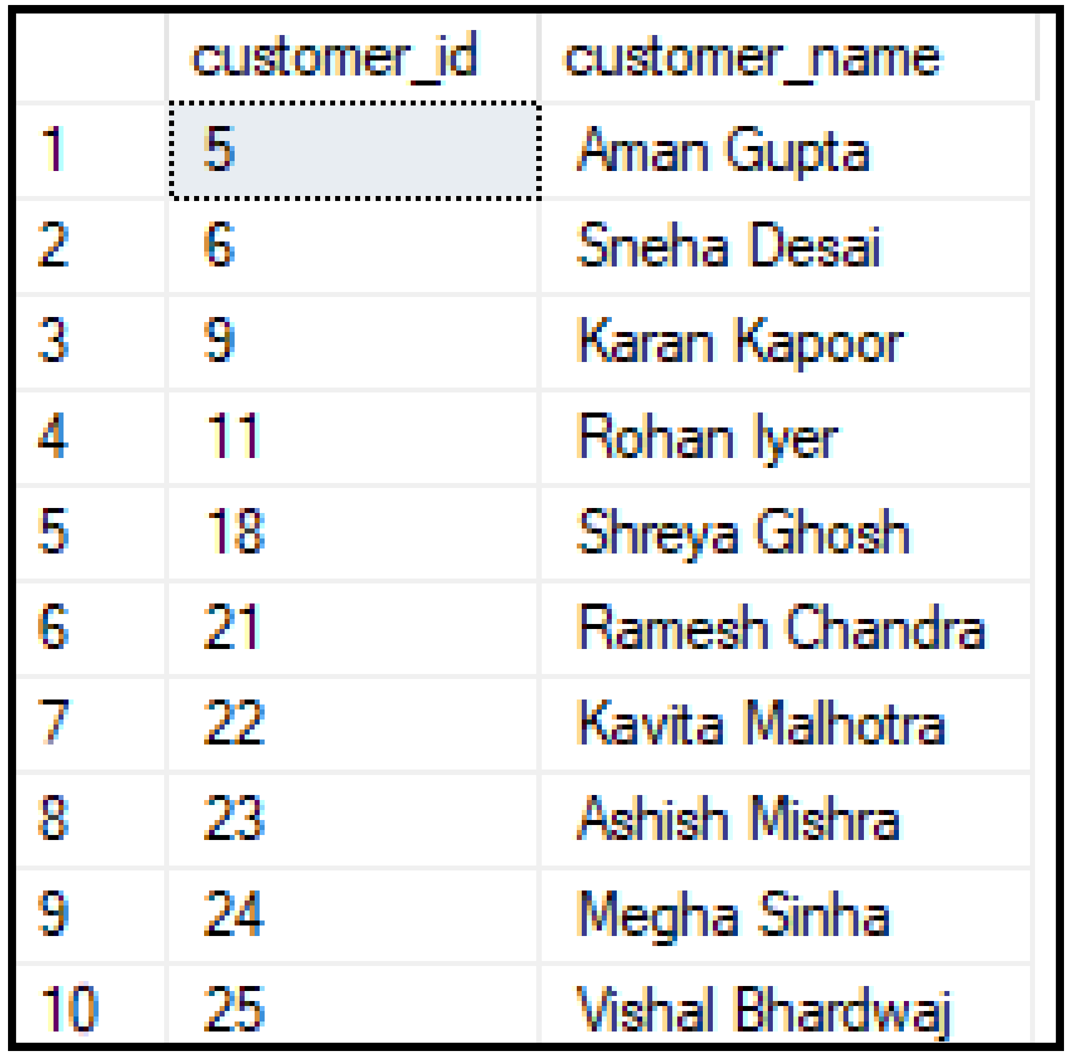
Solving Business Problems

->Find customers who haven’t placed an order in 2024 but did in 2023

Query

```
WITH Customer as (
SELECT c.customer_id
FROM orders O JOIN customers C
ON O.customer_id = C.customer_id
WHERE YEAR(O.order_date) in ('2024')
)
SELECT DISTINCT  c.customer_id,c.customer_name
FROM orders O JOIN customers C
on O.customer_id = O.customer_id
WHERE YEAR(o.order_date) IN ('2023')
and
c.customer_id not in (select customer_id from Customer)
```

Output



	customer_id	customer_name
1	5	Aman Gupta
2	6	Sneha Desai
3	9	Karan Kapoor
4	11	Rohan Iyer
5	18	Shreya Ghosh
6	21	Ramesh Chandra
7	22	Kavita Malhotra
8	23	Ashish Mishra
9	24	Megha Sinha
10	25	Vishal Bhardwaj

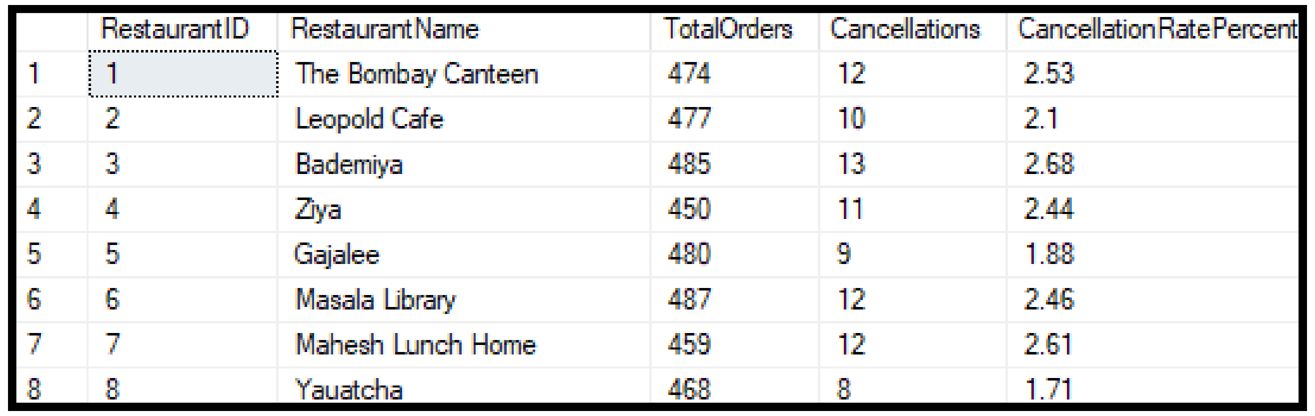
Solving Business Problems

->Calculate the cancellation rate for each restaurant between the 2023 and 2024

Query

```
WITH OrderCounts AS (
    SELECT
        r.restaurant_id AS RestaurantID,
        r.restaurant_name AS RestaurantName,
        COUNT(o.order_id) AS TotalOrders
    FROM orders o
    JOIN restaurants r ON o.restaurant_id = r.restaurant_id
    WHERE YEAR(o.order_date) BETWEEN 2023 AND 2024
    GROUP BY r.restaurant_id, r.restaurant_name
),
CancellationCounts AS (
    SELECT
        r.restaurant_id AS RestaurantID,
        r.restaurant_name AS RestaurantName,
        COUNT(o.order_id) AS Cancellations
    FROM orders o
    JOIN restaurants r ON o.restaurant_id = r.restaurant_id
    WHERE o.order_status = 'Not Fulfilled'
      AND YEAR(o.order_date) BETWEEN 2023 AND 2024
    GROUP BY r.restaurant_id, r.restaurant_name
)
SELECT
 oc.RestaurantID,
 oc.RestaurantName,
 oc.TotalOrders,
 COALESCE(cc.Cancellations, 0) AS Cancellations,
 ROUND(CAST(COALESCE(cc.Cancellations, 0) AS
FLOAT) /
 oc.TotalOrders * 100, 2) AS CancellationRatePercent
FROM OrderCounts oc
LEFT JOIN CancellationCounts cc
 ON oc.RestaurantID = cc.RestaurantID
ORDER BY oc.RestaurantID;
```

Output



	RestaurantID	RestaurantName	TotalOrders	Cancellations	CancellationRatePercent
1	1	The Bombay Canteen	474	12	2.53
2	2	Leopold Cafe	477	10	2.1
3	3	Bademiya	485	13	2.68
4	4	Ziya	450	11	2.44
5	5	Gajalee	480	9	1.88
6	6	Masala Library	487	12	2.46
7	7	Mahesh Lunch Home	459	12	2.61
8	8	Yauatcha	468	8	1.71

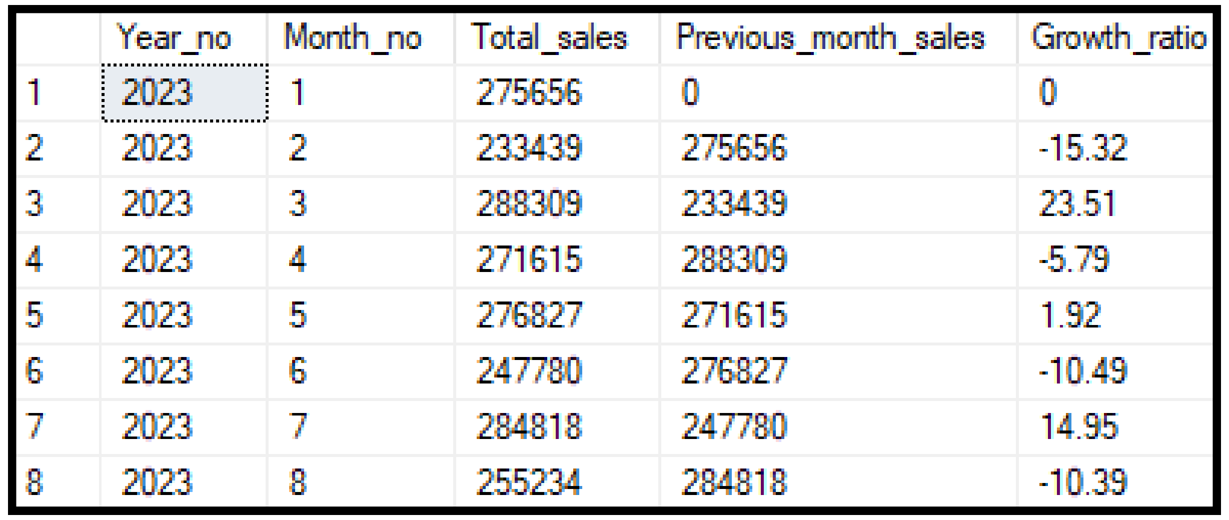
Solving Business Problems

->Identify sales trends by comparing each month's total sales to the previous month..

Query

```
WITH MonthlySales AS (
  SELECT
    YEAR(order_date) AS Year_no,
    MONTH(order_date) AS Month_no,
    SUM(total_amount) AS Total_sales,
    COALESCE(LAG(SUM(total_amount))
    OVER (ORDER BY YEAR(order_date), MONTH(order_date)),0) AS Previous_month_sales
  FROM orders
  GROUP BY YEAR(order_date), MONTH(order_date)
)
SELECT
  Year_no,
  Month_no,
  Total_sales,
  Previous_month_sales,
 COALESCE(ROUND(
      (CAST(Total_sales - Previous_month_sales AS FLOAT) / NULLIF(Previous_month_sales, 0)) * 100,
      2
    ),0) AS Growth_ratio
FROM MonthlySales
ORDER BY Year_no, Month_no;
```

Output



	Year_no	Month_no	Total_sales	Previous_month_sales	Growth_ratio
1	2023	1	275656	0	0
2	2023	2	233439	275656	-15.32
3	2023	3	288309	233439	23.51
4	2023	4	271615	288309	-5.79
5	2023	5	276827	271615	1.92
6	2023	6	247780	276827	-10.49
7	2023	7	284818	247780	14.95
8	2023	8	255234	284818	-10.39

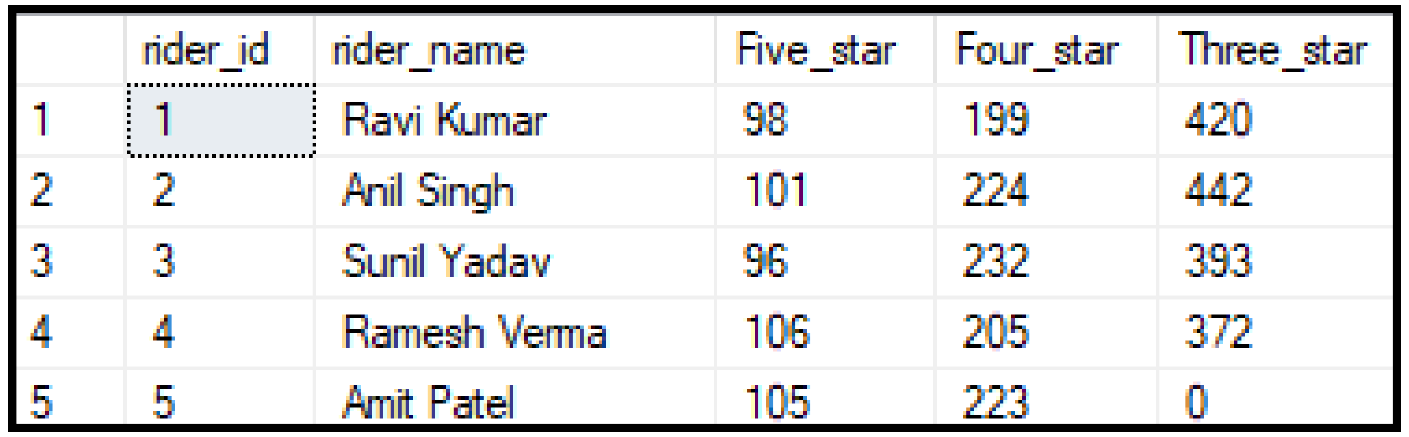
Solving Business Problems

->Find the number of 5-star, 4-star, and 3-star ratings each rider has. Riders receive ratings based on delivery time: ● 5-star: Delivered in less than 30 minutes ,● 4-star: Delivered between 30 and 45 minutes,● 3-star: Delivered after 45 minutes

Query

```
WITH diff AS (
  SELECT
    riders.rider_id,
    riders.rider_name,
    CASE
      WHEN delivery_time > order_time
      THEN DATEDIFF(MINUTE, order_time, delivery_time)
      ELSE DATEDIFF(MINUTE, order_time, DATEADD(DAY, 1, CAST(delivery_time AS DATETIME)))
    END AS Delivery_min
  FROM riders
  LEFT JOIN deliveries ON riders.rider_id = deliveries.rider_id
  LEFT JOIN orders ON orders.order_id = deliveries.order_id
  WHERE orders.Order_status = 'completed'
   AND deliveries.delivery_status = 'delivered'
)
SELECT
  rider_id,
  rider_name,
  SUM(CASE WHEN Delivery_min < 30 THEN 1 ELSE 0 END) AS Five_star,
  SUM(CASE WHEN Delivery_min BETWEEN 30 AND 45 THEN 1 ELSE 0 END) AS Four_star,
  SUM(CASE WHEN Delivery_min > 45 THEN 1 ELSE 0 END) AS Three_star
FROM diff  -- Use the CTE here
GROUP BY rider_id, rider_name
ORDER BY rider_id;
```

Output



	rider_id	rider_name	Five_star	Four_star	Three_star
1	1	Ravi Kumar	98	199	420
2	2	Anil Singh	101	224	442
3	3	Sunil Yadav	96	232	393
4	4	Ramesh Verma	106	205	372
5	5	Amit Patel	105	223	0

Solving Business Problems

->Segment customers into 'Gold' or 'Silver' groups based on their total spending compared to the average order value (AOV). If a customer's total spending exceeds the AOV,label them as 'Gold'; otherwise, label them as 'Silver'. Return: The total number of orders and total revenue for each segment.

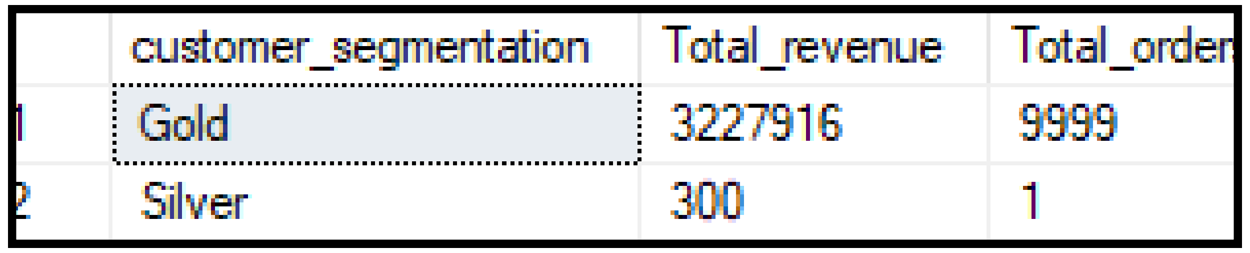
Query

```
WITH CusAvg AS (
   SELECT
      customers.customer_id,
      ROUND(AVG(total_amount), 2) AS AOV
   FROM customers
   INNER JOIN orders ON customers.customer_id = orders.customer_id
   GROUP BY customers.customer_id
),

cust_seg AS (
   SELECT
      CusAvg.customer_id,
      CASE
         WHEN SUM(orders.total_amount) > CusAvg.AOV THEN 'Gold'
         ELSE 'Silver'
      END AS customer_segmentation
   FROM CusAvg
   INNER JOIN orders ON CusAvg.customer_id = orders.customer_id
   GROUP BY CusAvg.customer_id, CusAvg.AOV
)
```

```
SELECT
 cust_seg.customer_segmentation,
 SUM(orders.total_amount) AS Total_revenue,
 COUNT(orders.order_id) AS Total_orders
FROM cust_seg
INNER JOIN orders ON cust_seg.customer_id = orders.customer_id
GROUP BY cust_seg.customer_segmentation;
```

Output



	customer_segmentation	Total_revenue	Total_order
1	Gold	3227916	9999
2	Silver	300	1

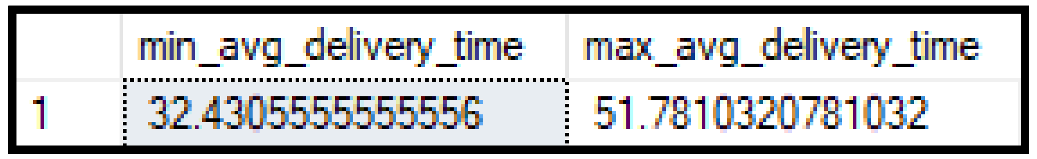
Solving Business Problems

->Evaluate rider efficiency by determining average delivery times and identifying those with the lowest and highest averages.

Query

```
WITH cte AS (
  SELECT
    r.rider_id,
    r.rider_name,
    CASE
      WHEN
        CAST(d.delivery_time AS DATETIME) > CAST(o.order_time AS DATETIME)
      THEN
        DATEDIFF(MINUTE, CAST(o.order_time AS DATETIME), CAST(d.delivery_time AS DATETIME))
      ELSE
        DATEDIFF(
          MINUTE,
          CAST(o.order_time AS DATETIME),
          DATEADD(DAY, 1, CAST(d.delivery_time AS DATETIME))
        )
    END AS delivery_time_took
  FROM
    riders r
    LEFT JOIN deliveries d ON r.rider_id = d.rider_id
    LEFT JOIN orders o ON o.order_id = d.order_id
  WHERE
    o.order_status = 'completed'
    AND d.delivery_status = 'delivered'
),
avg_deliv_time AS (
 SELECT
 rider_id,
 rider_name,
 AVG(CAST(delivery_time_took AS FLOAT)) AS avg_delivery_time_took
 FROM
 cte
 GROUP BY
 rider_id,
 rider_name
)
SELECT
 MIN(avg_delivery_time_took) AS min_avg_delivery_time,
 MAX(avg_delivery_time_took) AS max_avg_delivery_time
FROM
 avg_deliv_time;
```

Output



	min_avg_delivery_time	max_avg_delivery_time
1	32.4305555555556	51.7810320781032

Thank you