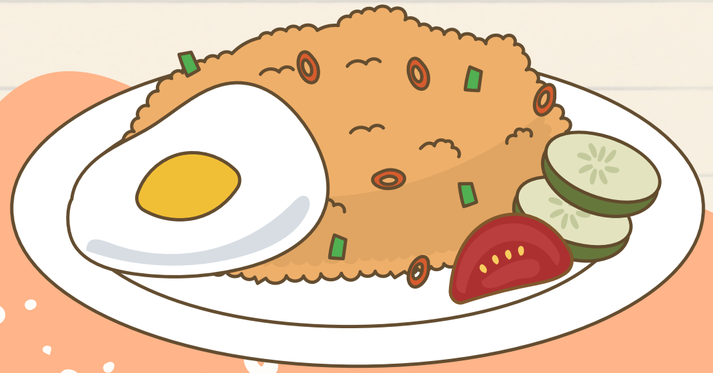
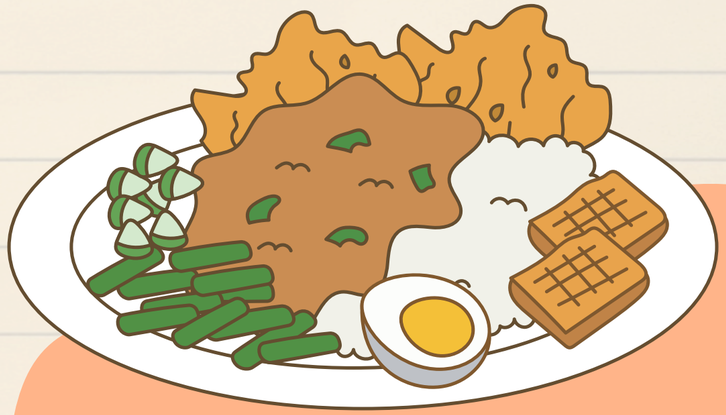
SWIGGY
SQL ANALYSIS

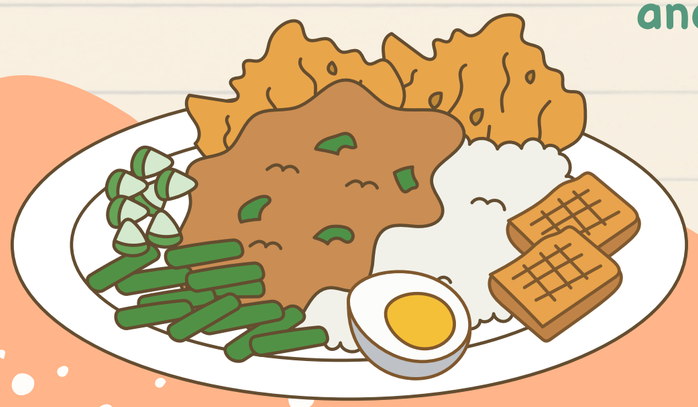


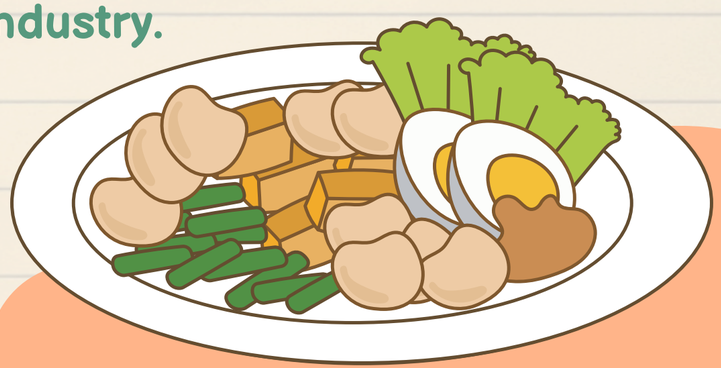


OVERVIEW

This project showcases a comprehensive MySQL-based data analysis of Swiggy, one of India's leading online food delivery platforms. Leveraging a structured relational database, the analysis explores key datasets involving customers, restaurants, orders, delivery partners, and ratings.

The objective is to demonstrate how data-driven decision-making can be supported by relational database systems in a real-world business context. This project is ideal for stakeholders looking to understand how SQL can be used for business intelligence (BI), performance analytics, and operational optimization in the food tech industry.





AGENDA

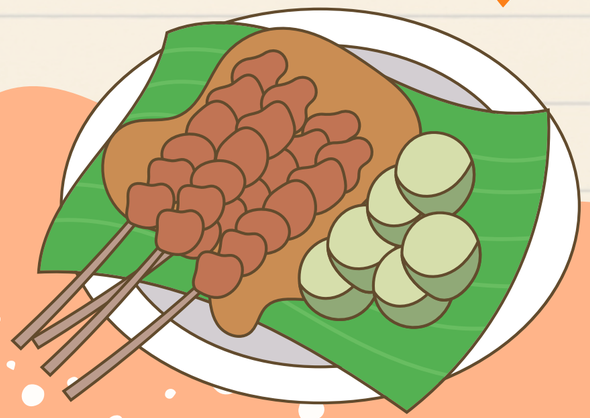
Customer Behaviour Analysis

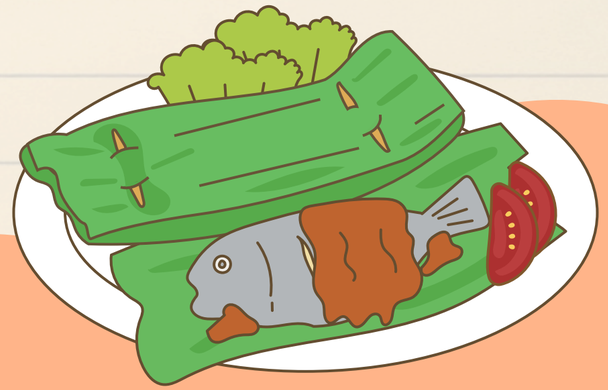
Restaurant Performance Metrics

Delivery Partner & Order Trends

Location Based Analysis

Order Timeline





TASK 01

Customer Based Analysis

Customers who have placed at least one order.

Query :

```
SELECT
    customers.customer_id,
    customers.name AS customer_name,
    COUNT(orders.order_id) AS total_orders
FROM
    customers
        JOIN
    orders ON customers.customer_id = orders.customer_id
GROUP BY 1 , 2
ORDER BY customers.customer_id;
```

Output:

customer_id	customer_name	total_orders
1	Amit Sharma	2
2	Rohini Verma	3
3	Rajesh Gupta	3
4	Sneha Mehta	2
5	Manish Kumar	4
6	Priya Singh	3
7	Vikas Reddy	3
8	Anjali Patel	3
9	Suresh Nair	1

TASK 02

Customer Based Analysis

Total number of orders placed by each customer.

Query :

```sql
SELECT
    customers.customer_id,
    customers.name AS customer_name,
    COUNT(orders.order_id) AS total_orders
FROM
    customers
        LEFT JOIN
    orders ON customers.customer_id = orders.customer_id
GROUP BY 1 , 2
ORDER BY count(orders.order_id) desc;
```

Output:

customer_id	customer_name	total_orders
5	Manish Kumar	4
2	Rohini Verma	3
3	Rajesh Gupta	3
6	Priya Singh	3
7	Vikas Reddy	3
8	Anjali Patel	3
14	Nidhi Saxena	3
15	Ashok Kumar	3
18	Sonali Mishra	3

TASK 03

Customer Based Analysis

Customers who have never placed an order.

Query :

```
SELECT
    *
FROM
    customers
WHERE
    customer_id NOT IN (SELECT
            customer_id
        FROM
            orders);
```

Output:

customer_id	name	email	phone_number	city	address
24	Sonal Kaur	sonal.kaur@gmail.com	NULL	Amritsar	S-19, Ranjit Avenue
25	Vivek Malhotra	vivek.malhotra@hotmail.com	9812789012	Thane	NULL
26	Divya Iyer	NULL	9823890123	Bangalore	T-20, Indiranagar
27	Rakesh Yadav	rakesh.yadav@gmail.com	9834901234	Varanasi	U-31, Lanka
28	Mona Sharma	mona.sharma@yahoo.com	9845012345	Ranchi	NULL
29	Sudha Pillai	sudha.pillai@gmail.com	9856123789	Kozhikode	V-42, Mavoor Road
30	Gaurav Khanna	NULL	9867238901	Gwalior	W-53, City Centre

TASK 04

Customer Based Analysis

Customers who have placed orders on exactly 03 different days.

Query :

```
SELECT
    c.customer_id, c.name, count(distinct o.order_date) as Unique_Days
FROM
    customers c
        JOIN
    orders o ON c.customer_id = o.customer_id
GROUP BY c.name , c.customer_id
HAVING Unique_Days = 3
order by c.customer_id;
```

Output:

customer_id	name	Unique_Days
2	Rohini Verma	3
6	Priya Singh	3
8	Anjali Patel	3
14	Nidhi Saxena	3
15	Ashok Kumar	3
18	Sonali Mishra	3

TASK 05

Customer Based Analysis

Customers who have same city and have placed orders at the same restaurants, but on different dates.

Query :

```
SELECT
    c1.customer_id AS c1_id,
    c2.customer_id AS c2_id,
    c1.name AS c1_full_name,
    c2.name AS c2_full_name,
    c1.city,
    o1.restaurant_id,
    o1.order_date AS c1_order_date,
    o2.order_date AS c2_order_date
FROM
    customers c1
    JOIN customers c2 ON c1.city = c2.city
                     AND c1.customer_id < c2.customer_id
    JOIN orders o1 ON o1.customer_id = c1.customer_id
    JOIN orders o2 ON o2.customer_id = c2.customer_id
                  AND o1.restaurant_id = o2.restaurant_id
                  AND DATE(o1.order_date) <> DATE(o2.order_date);
```

Output:

c1_id	c2_id	c1_full_name	c2_full_name	city	restaurant_id	c1_order_date	c2_order_date
5	18	Manish Kumar	Sonali Mishra	Delhi	3	2024-08-04 00:00:00	2024-08-05 00:00:00
19	23	Arjun Desai	Ravi Singh	Mumbai	8	2024-08-03 00:00:00	2024-08-09 00:00:00
5	18	Manish Kumar	Sonali Mishra	Delhi	3	2024-08-07 00:00:00	2024-08-05 00:00:00

OBSERVATIONS AND SUGGESTIONS

OBSERVATIONS	SUGGESTIONS
A majority of customers have placed at least one order.	🎯 Launch retention campaigns to boost order frequency among active users.
📊 Order frequency differs significantly across customers.	🔁 Introduce tiered loyalty programs to incentivize frequent ordering.
⛔ Some customers have never placed an order.	📩 Engage these users through personalized onboarding and targeted offers.
📅 Certain users place orders on exactly three unique days.	🧠 Analyze behavioral patterns within this group to test timing-based engagement strategies.
🌆 Customers from the same city order from the same restaurants on different dates.	👥 Encourage group ordering, referral programs, or social nudges like “Your friend ordered here.”

TASK 06

Restaurant Performance Metrics

Total revenue generated by each restaurant.

Query :

```
SELECT
    restaurants.name as Restaurant_Name,
    COALESCE(SUM(orders.total_amount), 0) AS Total_Revenue
FROM
    restaurants
        LEFT JOIN
    orders ON restaurants.restaurant_id = orders.restaurant_id
GROUP BY restaurants.name;
```

Output:

Restaurant_Name	Total_Revenue
Spice of India	1100.00
Tandoori Flames	1200.00
Biryani House	5300.00
Curry Pot	3200.00
Taste of Punjab	600.00
Royal Biryani	650.00
Coastal Delight	2100.00
Veggie Delight	1600.00
Gujarat Express	2550.00
Andhra Spice	4050.00
Punjabi Tadka	900.00

TASK 07

Restaurant Performance Metrics

Average rating of all restaurants in 'Mumbai'.

Query :

```sql
SELECT
    ROUND(AVG(COALESCE(rating, 0)), 2) AS Average_Restaurant_Rating
FROM
    restaurants
WHERE
    city = 'Mumbai';
```

Output:

Average_Restaurant_Rating
3.23

TASK 08

Restaurant Performance Metrics

Top 5 restaurants with the highest average rating.

Query :

```
with avg_restaurant_rating as (
select restaurant_id, name, round(avg(coalesce(rating,0)),2) as Average_rating from restaurants
group by restaurant_id, name),

top_rating_restaurant as ( select *, dense_rank() over (order by Average_rating desc) as Ranking
from avg_restaurant_rating)

select * from top_rating_restaurant where ranking <=5;
```

Output:

restaurant_id	name	Average_rating	Ranking
3	Biryani House	4.80	1
22	Paradise Biryani	4.80	1
6	Royal Biryani	4.70	2
30	Lucknowi Nawabi	4.70	2
12	Flavours of Bengal	4.60	3
19	Awadhi Zaika	4.60	3
1	Spice of India	4.50	4
15	Rajasthani Rasoi	4.50	4
26	Shahi Dawat	4.50	4
10	Andhra Spice	4.40	5
14	The Great Indian ...	4.40	5
25	Udupi Palace	4.40	5

OBSERVATIONS AND SUGGESTIONS

OBSERVATIONS	SUGGESTIONS
• Mumbai restaurants show varied average ratings.	Promote top-rated restaurants; audit and improve those with poor ratings through feedback mechanisms or quality checks.
• Some restaurants earn significantly higher revenue than others.	Prioritize visibility for high-performing restaurants; investigate causes behind low revenue in others and take corrective action.
• Top 5 restaurants (by rating) clearly stand out.	Feature them in “Editor’s Pick” or boost their presence during peak hours to maximize impact.

TASK 09

Location Based Analysis

Total orders placed by each customer in 'Mumbai'.

Query :

```
SELECT
    customers.customer_id,
    customers.name,
    customers.city,
    COUNT(orders.order_id)
FROM
    customers
        LEFT JOIN
    orders ON customers.customer_id = orders.customer_id
WHERE
    customers.city = 'Mumbai'
GROUP BY customers.customer_id , customers.name;
```

Output:

customer_id	name	city	count(o
1	Amit Sharma	Mumbai	2
3	Rajesh Gupta	Mumbai	3
19	Arjun Desai	Mumbai	2
23	Ravi Singh	Mumbai	2

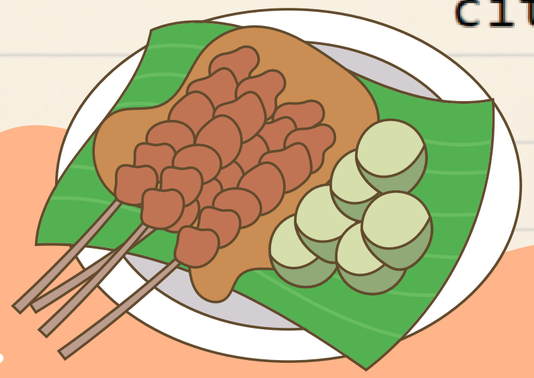
TASK 10

Location Based Analysis

Display all customers who live in 'Delhi'.

Query :

```
SELECT
    *
FROM
    customers
WHERE
    city = 'Delhi';
```



Output:

Result Grid | Filter Rows: | Edit: | Export/Import:

customer_id	name	email	phone_number	city	address
2	Rohini Verma	rohini.verma@yahoo.com	9823456789	Delhi	B-23, Saket
5	Manish Kumar	NULL	9834567890	Delhi	D-45, Lajpat Nagar
18	Sonali Mishra	NULL	9878345678	Delhi	N-54, Karol Bagh
NULL	NULL	NULL	NULL	NULL	NULL

OBSERVATIONS AND SUGGESTIONS

OBSERVATIONS	SUGGESTIONS
• Active user base is present in Delhi.	Launch localized marketing campaigns and events tailored to the Delhi audience.
• Mumbai customers show diverse ordering frequency.	Customize offers and incentives based on customer activity tiers (high/medium/low frequency).

TASK 11

Order Timeline

All orders placed in the last 10 days.

Query :

```
SELECT
    *
FROM
    orders
WHERE
    order_date >= '2024-08-15' - INTERVAL 10 DAY;
```

Output:

order_id	customer_id	restaurant_id	order_date	total_amount	status
13	3	14	2024-08-05 00:00:00	750.00	Completed
23	18	3	2024-08-05 00:00:00	1250.00	Completed
36	14	16	2024-08-05 00:00:00	950.00	Completed
40	3	14	2024-08-05 00:00:00	850.00	Completed
6	1	4	2024-08-06 00:00:00	500.00	Processing
14	11	4	2024-08-06 00:00:00	800.00	Processing
19	15	17	2024-08-06 00:00:00	1300.00	Completed
37	18	18	2024-08-07 00:00:00	1200.00	Processing

OBSERVATIONS AND SUGGESTIONS

OBSERVATIONS	SUGGESTIONS
• Orders have been placed in the last 10 days.	• Monitor peak traffic hours using time-based order analysis • Optimize delivery logistics by allocating more riders during peak hours • Launch flash deals during high-traffic periods to increase conversion • Implement personalized push notifications to re-engage recent customers • Analyze order patterns to forecast inventory needs more accurately • Introduce time-bound discounts to create urgency and drive more orders

TASK 12

Delivery Partners and Order Trends

Delivery Partners Who Have Completed More than 01 Completed Delivery

Query :

```
SELECT
    deliverypartners.partner_id,
    deliverypartners.name AS full_name,
    orders.status,
    COUNT(orders.order_id) AS delivery_count
FROM
    deliverypartners
        INNER JOIN
    orderdelivery ON deliverypartners.partner_id = orderdelivery.partner_id
        INNER JOIN
    orders ON orders.order_id = orderdelivery.order_id
WHERE
    orders.status =  completed'
GROUP BY ' , 2 , 3
HAVING delivery_count > 1;
```

Output:

partner_id	full_name	status	delivery_count
4	Suresh Reddy	Completed	4
5	Anita Desai	Completed	4
5	Rajesh Gupta	Completed	2
3	Priya Patel	Completed	3
1	Amit Sharma	Completed	2
7	Sonia Agarwal	Completed	3
2	Ravi Kumar	Completed	3
8	Vikram Singh	Completed	2
13	Mohit Saini	Completed	2

TASK 13

Delivery Partners and Order Trends

Delivery Partner who has worked with the most different customers

Query :

```sql
SELECT
    d1.name, COUNT(DISTINCT o1.order_id) AS Different_Customers
FROM
    deliverypartners d1
        JOIN
    orderdelivery o1 ON d1.partner_id = o1.partner_id
GROUP BY d1.name
ORDER BY Different_Customers DESC
LIMIT 1;
```

Output:

	name	Different_Customers
▶	Suresh Reddy	6

OBSERVATIONS AND SUGGESTIONS

OBSERVATIONS	SUGGESTIONS
Some delivery partners have multiple successful deliveries.	• Implement a tiered incentive program for top-performing partners. Use their delivery data to train or mentor new hires.
A few partners serve more unique customers.	• Analyze their delivery areas to identify underserved zones. Use them for pilot testing hyperlocal delivery campaigns.

THANK YOU!