



SWIGGY ADVANCE SQL PROJECT

PRESENTED BY: AMANJOT SINGH



PROJECT INTRODUCTION

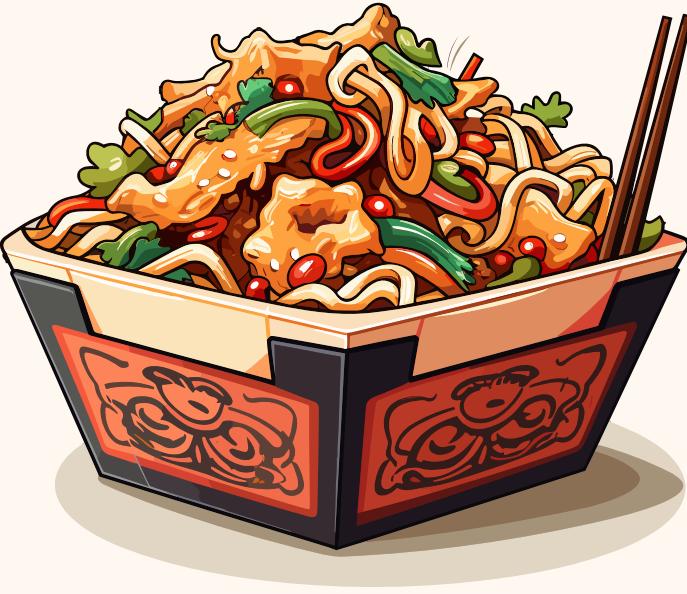
Swiggy, India's leading food delivery platform, strives to enhance customer satisfaction and streamline operations through data-driven insights. This project analyzes Swiggy's SQL dataset to uncover patterns in customer behavior, restaurant performance, and delivery partner efficiency. The findings will guide Swiggy in optimizing its services and boosting customer experience.





SWIGGY

SQL QUERIES USED FOR ANALYSIS



Display all customers who live in **ਦਿੱਲੀ**



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

Find the average rating of all
restaurants in मुंबई

```
SELECT  
    ROUND(AVG(rating), 1)  
FROM  
    restaurants  
WHERE  
    city = 'mumbai';
```

List all customers who have placed at least one order.

```
SELECT DISTINCT  
    customers.customer_id, customers.name  
FROM  
    customers  
        INNER JOIN  
    orders ON customers.customer_id = orders.customer_id;
```



Display the total number of orders placed by each customer.

```
SELECT  
    customers.customer_id,  
    customers.name,  
    COUNT(orders.order_id) AS no_of_orders  
FROM  
    customers  
        LEFT JOIN  
    orders ON customers.customer_id = orders.customer_id  
GROUP BY customers.customer_id , customers.name;
```



Find the total revenue generated by each restaurant.

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



Find the top 5 restaurants with the highest average rating.

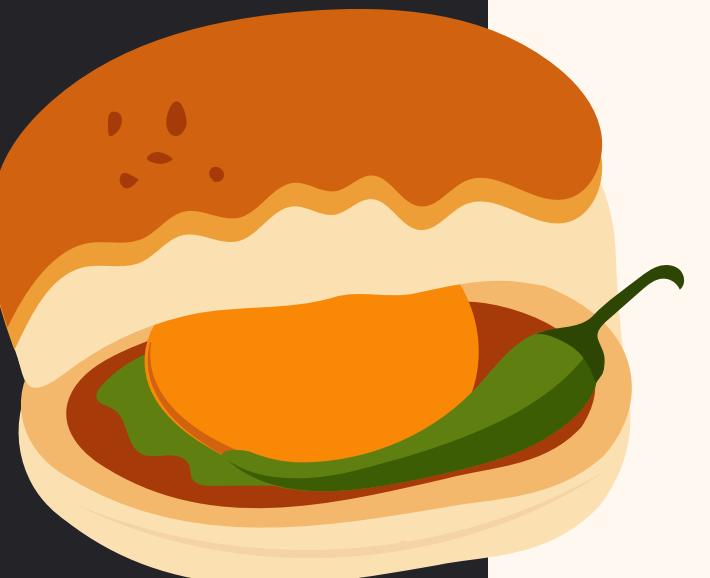
```
SELECT
    restaurant_id, name, city, ROUND(AVG(rating), 1) avg_rating
FROM
    restaurants
GROUP BY restaurant_id , name , city
ORDER BY AVG(rating) DESC
LIMIT 5;
```

Display all customers who have never placed an order.

```
SELECT DISTINCT
    customers.customer_id, customers.name
FROM
    customers
        LEFT JOIN
    orders ON customers.customer_id = orders.customer_id
WHERE
    orders.customer_id IS NULL;
```

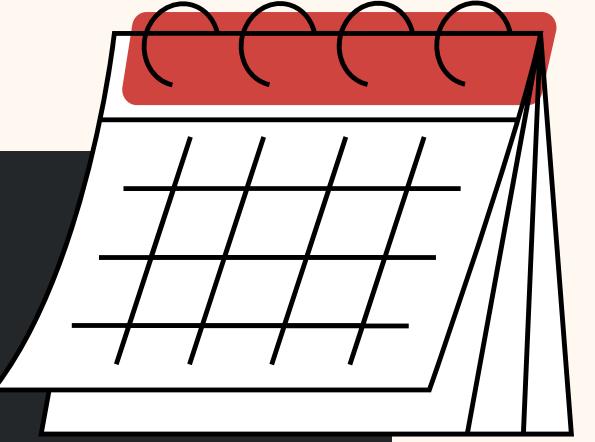
Find the number of orders placed by each customer in 'Mumbai'.

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



Display all orders placed in the last 30 days.

```
SELECT
  *
FROM
  orders
WHERE
  order_date >= CURDATE() - INTERVAL 30 DAY;
```



List all delivery partners who have completed more than 1 delivery



```
SELECT
    deliverypartners.partner_id,
    deliverypartners.name,
    COUNT(deliveryupdates.order_id) no_of_orders
FROM
    deliverypartners
        JOIN
    orderdelivery ON deliverypartners.partner_id = orderdelivery.partner_id
        JOIN
    deliveryupdates ON deliveryupdates.order_id = orderdelivery.order_id
WHERE
    deliveryupdates.status = 'Delivered'
GROUP BY deliverypartners.partner_id
HAVING no_of_orders > 1;
```

Find the customers who have placed orders on exactly three different days.

```
SELECT  
    customers.customer_id, customers.name  
FROM  
    customers  
    JOIN  
        orders ON customers.customer_id = orders.customer_id  
GROUP BY customers.customer_id , customers.name  
HAVING COUNT(DISTINCT orders.order_date) = 3;
```



Find the delivery partner who has worked with the most different customers.

```
SELECT
    deliverypartners.partner_id,
    deliverypartners.name,
    COUNT(DISTINCT orders.customer_id) customer_count
FROM
    deliverypartners
    JOIN
    orderdelivery ON deliverypartners.partner_id = orderdelivery.partner_id
    JOIN
    orders ON orderdelivery.order_id = orders.order_id
GROUP BY deliverypartners.partner_id , deliverypartners.name
ORDER BY customer_count DESC
LIMIT 1;
```



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

```
SELECT c1.customer_id AS customer1_id, c1.name AS customer1_name, c1.city,
       c2.customer_id AS customer2_id, c2.name AS customer2_name, o1.restaurant_id
  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;

SELECT c1.name AS customer1_name, c1.city,
       c2.name AS customer2_name, o1.restaurant_id
  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 o1.order_date <> o2.order_date;
```



THANK YOU

Like, Comment, Share, and Follow!



Connect with me on LinkedIn



Amanjot Singh