SWIGGY

ANALYZING SWIGGY DATA

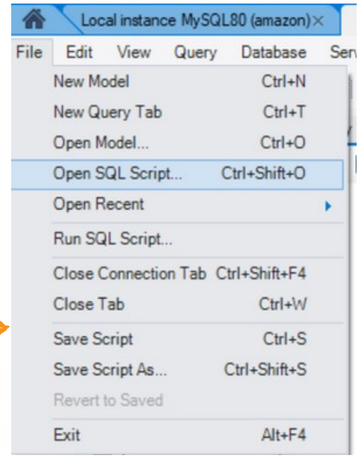
SQL Queries for Key Insight

Rahul Nagra

GET STARTED WITH THE PROJECT

```
-- Create Database
CREATE DATABASE IF NOT EXISTS SwiggyDB;
USE SwiggyDB;
```

IN WORKBENCH, CREATE A NEW DATABASE NAMED `SwiggyDB` AND THEN USE IT BY EXECUTING THE COMMAND `USE SwiggyDB;`



SINCE THE TABLES ARE EMPTY, IMPORT THE DATA BY CLICKING ON "FILE" AND SELECTING "OPEN SQL SCRIPT." UPLOAD YOUR DATASET FILE.

AFTER THE DATASET IS UPLOADED, YOU'LL HAVE ACCESS TO THE SWIGGY DATA, ALLOWING YOU TO START YOUR PROJECT TASKS.

```
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY AUTO_INCREMENT
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100),
    phone_number VARCHAR(15),
    city VARCHAR(50) NOT NULL,
    address VARCHAR(255)
);
```

TO LOAD THE DATA

Rahul Nagra

QUERIES

1	-- Display all customers who live in 'Delhi'.
2	-- Find the average rating of all restaurants in 'Mumbai'.
3	-- List all customers who have placed at least one order.
4	-- Display the total number of orders placed by each customer.
5	-- Find the total revenue generated by each restaurant.
6	-- Find the top 5 restaurants with the highest average rating.
7	-- Display all customers who have never placed an order.

QUERIES

8	-- Find the number of orders placed by each customer in 'Mumbai'.
9	-- Display all orders placed in the last 30 days.
10	-- List all delivery partners who have completed more than 1 delivery.
11	-- Find the customers who have placed orders on exactly three different days.
12	-- Find the delivery partner who has worked with the most different customers.
13	-- Identify customers who have the same city and have placed orders at the same restaurants, but on different dates.

Q1:-

```
-- Display all customers who live in 'Delhi'.
```

Syntax

```
SELECT
    customer_id, name, city
FROM
    customers
WHERE
    city = 'Delhi';
```

Purpose

This tool helps find all the customers who live in ‘Delhi’.

customer_id	name	city
2	Rohini Verma	Delhi
5	Manish Kumar	Delhi
18	Sonali Mishra	Delhi
NULL	NULL	NULL

Q2:-

```
-- Find the average rating of all restaurants in 'Mumbai'.
```

Syntax

```
SELECT
    city, round(AVG(rating),2) AS avg_rating
FROM
    restaurants
WHERE
    city = 'Mumbai';
```

Purpose

This query highlights the average rating of all restaurants that are from ‘Mumbai’.

city	avg_rating
Mumbai	4.30

Q3:-

```
-- List all customers who have placed at least one order.
```

Syntax

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

Purpose

This query finds all customers who have placed at least one order.

customer_id	name
1	Amit Sharma
2	Rohini Verma
3	Rajesh Gupta
4	Sneha Mehta
5	Manish Kumar
6	Priya Singh
7	Vikas Reddy

Q4:-

```
-- Display the total number of orders placed by each customer.
```

Syntax

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

This query displays the total number of orders placed by each customer.

customer_id	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

Q5:-

```
-- Find the total revenue generated by each restaurant.
```

Syntax

```
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 restaurant_id , restaurants.name;
```

Purpose

This query finds the total revenue generated by each restaurant.

restaurant_id	name	Revenue
1	Spice of India	1100.00
2	Tandoori Flames	1200.00
3	Biryani House	5300.00
4	Curry Pot	3200.00
5	Taste of Punjab	600.00
6	Royal Biryani	650.00
7	Coastal Delight	2100.00

Q6:-

```
-- Find the top 5 restaurants with the highest average rating.
```

Syntax

```
SELECT
    restaurants.restaurant_id,
    restaurants.name,
    ROUND((rating), 2) AS Avg_rating
FROM
    restaurants
GROUP BY restaurants.restaurant_id , restaurants.name
ORDER BY Avg_rating DESC
LIMIT 5;
```

Purpose

This query is used to find the top 5 restaurants with the highest average rating.

restaurant_id	name	Avg_rating
3	Biryani House	4.80
22	Paradise Biryani	4.80
30	Lucknowi Nawabi	4.70
6	Royal Biryani	4.70
12	Flavours of Bengal	4.60

Q7:-

```
-- Display all customers who have never placed an order.
```

Syntax

```
SELECT
    customers.customer_id,
    customers.name,
    COUNT(orders.order_id) AS count_order
FROM
    customers
        LEFT JOIN
    orders ON orders.customer_id = customers.customer_id
GROUP BY customers.customer_id , customers.name
HAVING COUNT(orders.order_id) = 0;
```

Purpose

This query displays all customers who have never placed an order.

customer_id	name	count_order
24	Sonal Kaur	0
25	Vivek Malhotra	0
26	Divya Iyer	0
27	Rakesh Yadav	0
28	Mona Sharma	0
29	Sudha Pillai	0
30	Gaurav Khanna	0

Q8:-

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

Syntax

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

Purpose

This query finds the number of orders placed by each customer in ‘Mumbai’.

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

Q9:-

-- Display all orders placed in the last 30 days.

Syntax

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

Purpose

This query displays all orders placed in the last 30 days.

order_id	customer_id	restaurant_id	order_date	total_amount	status
NULL	NULL	NULL	NULL	NULL	NULL

Q10:-

```
-- List all delivery partners who have completed more than 1 delivery.
```

Syntax

```
SELECT DISTINCT
    deliverypartners.partner_id, deliverypartners.name
FROM
    deliverypartners
        JOIN
    orderdelivery ON deliverypartners.partner_id = orderdelivery.partner_id
        JOIN
    deliveryupdates ON orderdelivery.order_delivery_id = deliveryupdates.delivery_id
WHERE
    deliveryupdates.status = 'Delivered';
```

Purpose

This query lists all delivery partners who have completed more than 1 delivery.

partner_id	name
4	Suresh Reddy
3	Priya Patel
5	Anita Desai
7	Sonia Agarwal
2	Ravi Kumar
12	Reena Rao
18	Meera Gupta

Q11:-

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

Syntax

```
SELECT
    customers.customer_id,
    customers.name,
    COUNT(orders.order_date)
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;
```

Purpose

This query finds the customers who have placed orders on exactly three different days.

customer_id	name	COUNT(orders.order_date)
2	Rohini Verma	3
6	Priya Singh	3
8	Anjali Patel	3
14	Nidhi Saxena	3
15	Ashok Kumar	3
18	Sonali Mishra	3

Q12:-

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

Syntax

```
SELECT
    deliverypartners.partner_id,
    deliverypartners.name,
    COUNT(DISTINCT orders.customer_id) as 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;
```

Purpose

This query finds the delivery partner who has worked with the most different customers.

partner_id	name	customer_count
4	Suresh Reddy	6

Q13:-

```
-- Identify customers who have the same city and have placed orders at the same restaurants,
-- but on different dates.select * from complaints;
```

Syntax

```
SELECT
    c1.name AS customer1,
    c2.name AS customer2,
    c1.city AS city1,
    c2.city AS c2,
    restaurants.name
FROM
    customers AS c1
        JOIN
    orders AS o1 ON c1.customer_id = o1.customer_id
        JOIN
    orders AS o2 ON o2.restaurant_id = o1.restaurant_id
        JOIN
    customers AS c2 ON c1.city = c2.city AND c1.name <> c2.name
        AND o2.customer_id = c2.customer_id
        JOIN
    restaurants ON o1.restaurant_id = restaurants.restaurant_id
WHERE
    o1.order_date <> o2.order_date;
```

Purpose

This query identifies customers who have the same city and have placed orders at the same restaurants but on different dates.select* from complaints;

customer1	customer2	city1	c2	name
Manish Kumar	Sonali Mishra	Delhi	Delhi	Biryani House
Sonali Mishra	Manish Kumar	Delhi	Delhi	Biryani House
Sonali Mishra	Manish Kumar	Delhi	Delhi	Biryani House
Arjun Desai	Ravi Singh	Mumbai	Mumbai	Veggie Delight
Manish Kumar	Sonali Mishra	Delhi	Delhi	Biryani House
Ravi Singh	Arjun Desai	Mumbai	Mumbai	Veggie Delight

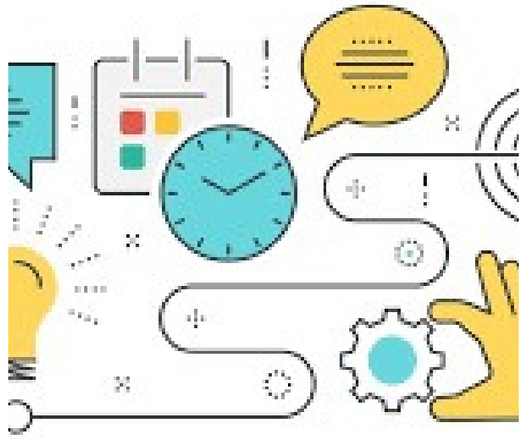
MEET WHAT WE HAVE LEARNED

In this project, we advanced our SQL skills with complex filtering and aggregate functions, sharpening our ability to analyze product attributes and customer reviews. We applied these skills to real-world data, improving our data-driven decision-making and optimizing queries to uncover valuable insights.

Advanced SQL Query Techniques

Practical Applications

Data Analysis Skills



Enhanced Query Optimization

Find this useful? like and share this post with your friends.

Connect with me.

Email
rahul.nagra.001@gmail.com

LinkedIn
Rahul Nagra

Call
7973193965