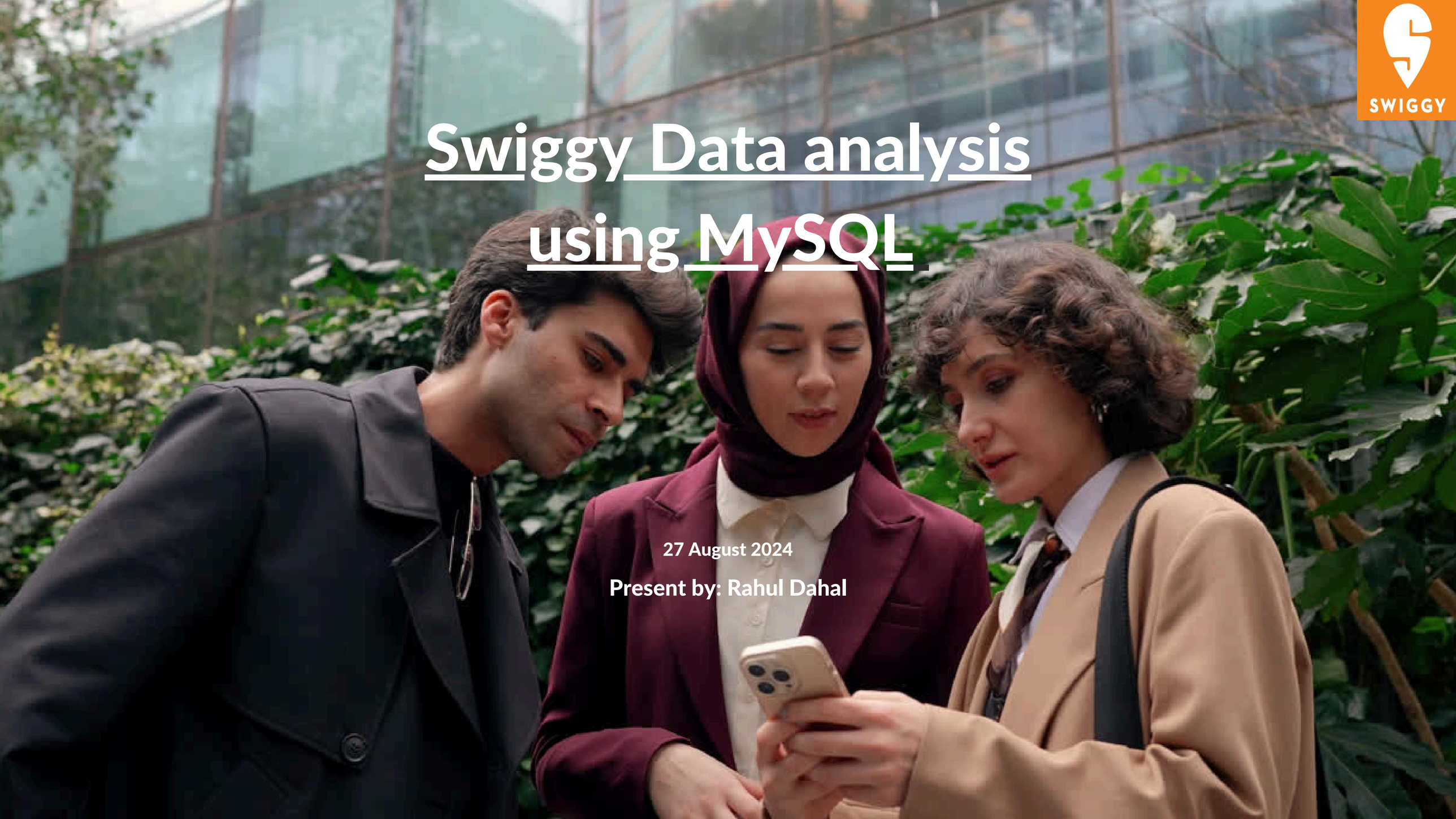


Swiggy Data analysis using MySQL

27 August 2024

Present by: Rahul Dahal


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

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

-- Inserting data into Customers table
INSERT INTO Customers (name, email, phone
VALUES
('Amit Sharma', 'amit.sharma@gmail.com',
('Rohini Verma', 'rohini.verma@yahoo.com'
('Rajesh Gupta', 'rajesh.gupta@gmail.com'
('Sneha Mehta', 'sneha.mehta@gmail.com',
('Manish Kumar', NULL, '9834567890', 'Del
```

Problem Statement:

In this project, Swiggy seeks to derive actionable insights from its extensive SQL dataset to drive strategic decision-making. The primary goal is to implement sophisticated SQL queries to analyze customer behavior, restaurant performance, and delivery partner efficiency. Specific objectives include identifying customer segments based on location, analyzing restaurant ratings, tracking order patterns, and evaluating delivery partner contributions. By uncovering these insights, Swiggy aims to enhance customer satisfaction, optimize restaurant partnerships, and streamline delivery operations.

Objective:

To extract key insights from Swiggy's SQL dataset by implementing advanced SQL queries, focusing on customer behavior, restaurant performance, and delivery efficiency to support strategic decision-making.

Swiggy

Swiggy seeks insights from its SQL dataset. Implement sophisticated SQ

Dataset Link: https://drive.google.com/file/d/1S32wPjwNUlhi2G5xiW3_w

Questions:

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.
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 differ
12. Find the delivery partner who has worked with the most different c
13. Identify customers who have the same city and have placed orders

/04

1. Display all customers who live in 'Delhi'.

Here to retrieve the data i used where clause, with condition.

```
1   #Display all customers who live in 'Delhi'.
2   select * from customers where city = "Delhi";
```

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

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

SWIGGY

/05

2. Find the average rating of all restaurants in 'Mumbai'.

Here to retrieve the data used again where clause with average function.

SQL File 8* | customers | restaurants

Don't Limit

```
1   #Find the average rating of all restaurants in "Mumbai".
2 • SELECT city, avg(rating) from restaurants where city = "Mumbai" group by city;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

city	avg(rating)
Mumbai	4.300000

SWIGGY

/06

3. List all customers who have placed at least one order.

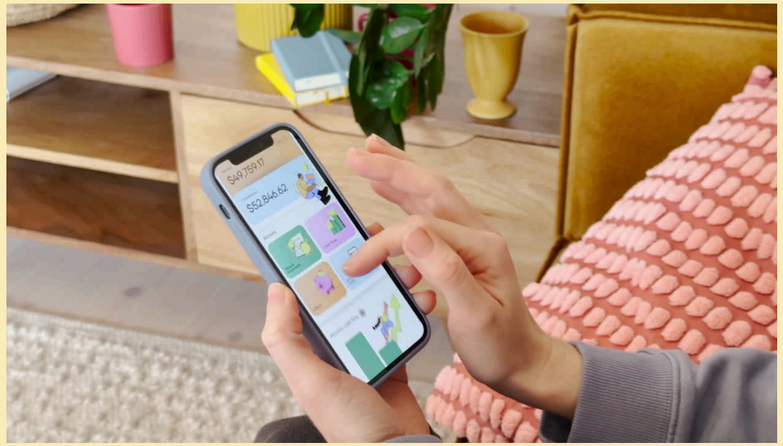
Here to retrieve the data having clause was used for the condition as the condition was based on aggregated column and join was utilized to retrieve the data from different tables.

```
1   #List all customers who have placed at least one order.
2
3 • select distinct customers.name, count(orders.order_id)from customers join orders
4   on customers.customer_id = orders.customer_id
5   group by customers.name having count(orders.order_id)>=1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	count(orders.order_id)
Suresh Nair	1
Karan Kapoor	1
Amit Sharma	2
Sneha Mehta	2
Kavita Deshmukh	2
Vivek Bhatt	2
Meera Joshi	2

/07



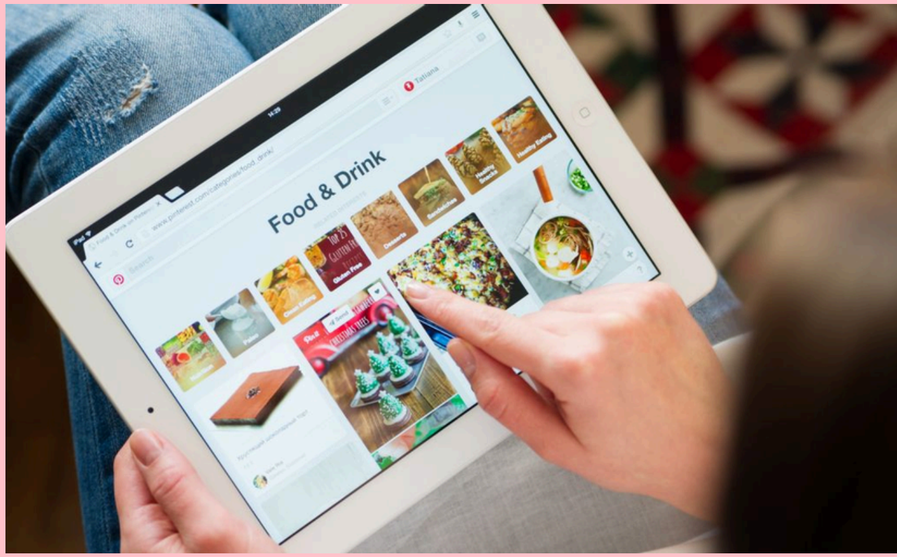
4. Display the total number of orders placed by each customer.

Similar to the previous one however utilized the left join to get all the data from left table which was customer and other common data from orders side

```
3 •  select distinct customers.name, count(orders.order_id) as total_orders from customers left join orders
4    on customers.customer_id = orders.customer_id
5    group by customers.name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	total_orders
Gaurav Khanna	0
Suresh Nair	1
Karan Kapoor	1
Amit Sharma	2
Sneha Mehta	2
Kavita Deshmukh	2
Vivek Bhatt	2
Meera Joshi	2



/08

5. Find the total revenue generated by each restaurant.

Here to retrieve the data simply left join was utilized with sum and coalesce function to detect the total sum including those restaurants with null values too.

SQL File 8* | customers | restaurants | SQL File 12* | SQL File 13* | restaurants | orders

Don't Limit

```
1   # Find the total revenue generated by each restaurant
2
3 • select restaurants.name, coalesce(sum(orders.total_amount),0)
4   from restaurants left join orders
5   on restaurants.restaurant_id = orders.restaurant_id
6   group by restaurants.name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	coalesce(sum(orders.total_amount),0)
Thane Tadka	0.00
Lucknowi Nawabi	0.00
Taste of Punjab	600.00
Royal Biryani	650.00
Punjabi Tadka	900.00
Kerala Kitchen	950.00
Spice of India	1100.00
Tandoori Flames	1200.00

SWIGGY

6. Find the top 5 restaurants with the highest average rating.

Here to retrieve the highest data the order by operator was utilized by making data on descending order and then filtered the top 5 rows using limit.

SQL File 8* | customers | restaurants | SQL File 12* | SQL File 13* | SQL File 14* | restaurants

Don't Limit

```
1   #Find the top 5 restaurants with the highest average rating.
2 • select name, avg(rating) from restaurants
3   group by name
4   order by avg(rating) desc
5   limit 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	avg(rating)
Biryani House	4.800000
Paradise Biryani	4.800000
Lucknowi Nawabi	4.700000
Royal Biryani	4.700000
Flavours of Bengal	4.600000

SWIGGY

7. Display all customers who have never placed an order.

Here to retrieve the data again left join was utilized and where condition to get the desired values.

SQL File 8* | customers | restaurants | SQL File 12* | SQL File 13* | SQL File 14* | SQL File 15* | customers | orders

Don't Limit

```
1   #Display all customers who have never placed an order.
2 • select distinct customers.name, orders.order_id
3    from customers left join orders
4    on customers.customer_id = orders.customer_id
5    where orders.order_id = 0 or orders.order_id is null;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	order_id
Sonal Kaur	NULL
Vivek Malhotra	NULL
Divya Iyer	NULL
Rakesh Yadav	NULL
Mona Sharma	NULL
Sudha Pillai	NULL
Gaurav Khanna	NULL

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

Again left join utilization with the where condition

```sql
#Find the number of orders placed by each customer in 'Mumbai'

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, customers.city;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	name	city	count(orders.order_id)
1	Amit Sharma	Mumbai	2
3	Rajesh Gupta	Mumbai	3
19	Arjun Desai	Mumbai	2
23	Ravi Singh	Mumbai	2

SWIGGY

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

Use of date difference function with the desired condition

```
1    # Display all orders placed in the last 30 days.
2
3 •  select * from orders
4    where datediff(current_Date, order_date)<=30;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

order_id	customer_id	restaurant_id	order_date	total_amount	status
1	1	3	2024-08-01 00:00:00	750.00	Completed
2	2	5	2024-08-02 00:00:00	600.00	Completed
3	3	1	2024-08-04 00:00:00	0.00	Cancelled
4	4	7	2024-08-01 00:00:00	850.00	Completed
5	5	2	2024-08-03 00:00:00	1200.00	Completed
6	1	4	2024-08-06 00:00:00	500.00	Processing
7	6	8	2024-08-03 00:00:00	950.00	Completed
8	7	9	2024-08-08 00:00:00	700.00	Completed
9	8	6	2024-08-02 00:00:00	650.00	Completed

SWIGGY

10. List all delivery partners who have completed more than 1 delivery

Use of multiple joins as of the difference on common columns and utilization of having clause as of the aggerated column- based condition

SQL File 18* | deliverypartners | orderdelivery | deliveryupdates

Don't Limit

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

select deliverypartners. name, count(orderdelivery.order_id)
from deliverypartners join orderdelivery
on orderdelivery.partner_id = deliverypartners.partner_id
join deliveryupdates
on deliveryupdates.order_id = orderdelivery.order_id
where deliveryupdates.status<>"failed"
group by deliverypartners.name
having count(orderdelivery.order_id)>1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	count(orderdelivery.order_id)
Suresh Reddy	10
Anita Desai	6
Rajesh Gupta	6
Priya Patel	4
Sneha Iyer	2
Amit Sharma	4

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

As there was a condition for three different days so simply count function was used by making date distinct using having clause.

SQL File 15* | customers | orders

Don't Limit

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

select customers.name, count(orders.order_date)
from customers join orders
on customers.customer_id = orders.customer_id
group by customers.name
having count(distinct orders.order_date)=3;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	count(orders.order_date)
Anjali Patel	3
Ashok Kumar	3
Nidhi Saxena	3
Priya Singh	3
Rohini Verma	3
Sonali Mishra	3

SWIGGY

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

As similar to before as of the multiple common columns from different tables more then two joins were utilized, additionally to get the highest value order by was used and finally limited the row with one to achieve top value.

SQL File 16* | deliverypartners | orderdelivery | orders

Don't Limit

```
1   # Find the delivery partner who has worked with most different customers.
2
3   Select deliverypartners.name, count(distinct orders.customer_id)
4   as diff_customers
5   from orderdelivery join orders
6   on orderdelivery.order_id = orders.order_id
7   join deliverypartners
8   on deliverypartners.partner_id = orderdelivery.partner_id
9   group by deliverypartners.name
10  order by count(distinct orders.customer_id) desc
11  Limit 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	diff_customers
Suresh Reddy	6

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

This task was quite tricky, as here we wanted to utilize data from the same table by making duplicate tables using self-join. Here we distinct the name column as not to make it repeat. As of no common columns tried to put the condition at the same time while joining. at the same time used “on” & “and” operators as the “where” condition was being rejected while tried to make at the end. so “and” operator was utilized at the time of join.

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

SELECT DISTINCT
    c1.name,
    c2.name,
    c1.city,
    c2.city,
    restaurants.name,
    o1.order_date,
    o2.order_date
FROM
    customers AS c1
        JOIN
    customers AS c2 ON c1.city = c2.city
        AND c1.customer_id <> c2.customer_id
        JOIN
    orders AS o1 ON c1.customer_id = o1.customer_id
        JOIN
    orders AS o2 ON c2.customer_id = o2.customer_id
        AND o1.order_date <> o2.order_date
        JOIN
    restaurants ON restaurants.restaurant id = o1.restaurant id:
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name	name	city	city	name	order_date	order_date
Vikas Reddy	Ashok Kumar	Chennai	Chennai	Flavours of Bengal	2024-08-09 00:00:00	2024-08-06 00:00:00
Vikas Reddy	Ashok Kumar	Chennai	Chennai	Flavours of Bengal	2024-08-09 00:00:00	2024-08-12 00:00:00
Vikas Reddy	Ashok Kumar	Chennai	Chennai	Flavours of Bengal	2024-08-09 00:00:00	2024-08-15 00:00:00
Vikas Reddy	Ashok Kumar	Chennai	Chennai	Gujarat Express	2024-08-08 00:00:00	2024-08-06 00:00:00

Conclusion

The analysis of swiggy's data using SQL provided valuable insights into customer choices, product and restaurants performance, and customer sentiment. By querying the dataset, we identified key trends such as restaurants with good performance, high-value delivery partners and common customer themes. These findings can inform operational and marketing strategies, enhance product offerings, and improve overall customer satisfaction. The ability to derive actionable insights from data underscores the importance of data-driven decision-making in optimizing e-commerce operations.

SWIGGY

Thank You

rahuldahal2053@gmail.com

+61-413654026

Rahul Dahal

Daw park, Adelaide, 5041

August 2024

Wscube tech