SWIGGY

Data Insights using MySQL.

-By Shivani Nagar

DATA ANALYSIS

OBJECTIVE

Swiggy aims to derive strategic insights from its SQL dataset by implementing advanced queries and complex joins. The objective is to conduct an in-depth analysis that will inform key business decisions, optimize operations, and enhance customer satisfaction.

DATA ANALYSIS

MySQL

Steps to Load the Database :

- In the Server Tab->Data Import->Import from self contained file->Choose location of the data from local file system->Start Import.
- Refresh the Schemas to see the swiggydb Database added.
- Then double tap on swiggydb to select the database.

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

```
3 •   select * from customers
4     where city = "Delhi";
```

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

```
3 •  select city,round(avg(rating),2) from restaurants
4    where city = "Mumbai" group by city;
```

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

Approach 1 :

```sql
select distinct customers.name from customers inner join orders
on customers.customer_id=orders.customer_id;
```

Approach 2 :

```sql
select customers.name,count(orders.customer_id)
from customers join orders
on customers.customer_id=orders.customer_id
group by customers.customer_id
having count(orders.order_id>=1);
```

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

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

5.Find the total revenue generated by each restaurant.

```
12 •  select restaurants.name,sum(orders.total_amount)
13    from restaurants left join orders
14    on restaurants.restaurant_id=orders.restaurant_id
15    group by restaurants.name;
```

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

```
11 •   select name,rating from restaurants
12     order by rating desc limit 5;
```

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

```sql
select distinct customers.name,orders.order_id
from customers left join orders
on customers.customer_id=orders.customer_id
where orders.order_id is null;
```

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

```
22 •  select customers.name,count(orders.customer_id)
23    from customers left join orders
24    on customers.customer_id=orders.customer_id
25    where city = "Mumbai"
26    group by customers.name;
```

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

```
28 •   select *,datediff(now(),order_date) as last_30_days from orders
29     where datediff(now(),order_date) <= 30;
```

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

```
3 •   select deliverypartners.name,count(orderdelivery.order_id)
4     from deliverypartners join orderdelivery
5     on deliverypartners.partner_id=orderdelivery.partner_id
6     group by deliverypartners.name
7     having count(orderdelivery.order_id) > 1;
```

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

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

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

```sql
select deliverypartners.name,deliverypartners.partner_id,
count(distinct orders.customer_id) as customer_ids
from deliverypartners join orderdelivery
on deliverypartners.partner_id = orderdelivery.partner_id
join orders
on orderdelivery.order_id=orders.order_id
group by deliverypartners.name,deliverypartners.partner_id
order by customer_ids desc limit 1;
```

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

```
4 •  select DISTINCT c1.name AS customer1,c2.name AS customer2, c1.city,
5    r.name AS restaurant, o1.order_date AS Order_dat1, o2.order_date AS Order_date2
6    FROM customers c1
7    JOIN Orders o1 ON c1.customer_id=o1.customer_id
8    JOIN Orders o2 ON o1.restaurant_id=o2.restaurant_id
9    JOIN Customers c2 ON c1.city = c2.city
10                    AND c1.customer_id <> c2.customer_id
11                    AND o2.customer_id = c2.customer_id
12   JOIN Restaurants r ON o1.restaurant_id = r.restaurant_id
13   WHERE o1.order_date <> o2.order_date
14   ORDER BY c1.city,r.name,o1.order_date;
```

THANKS FOR READING.

Follow for more :

https://www.linkedin.com/in/shivani-nagar-928028275/