**SWIGGY CUSTOMER-ORDER DATA ANALYSIS WITH SQL**

This project helps in analyzing the swiggy order data of a customer for 5 years. Swiggy in India is a food ordering application (similar to uber eats) which has gained prominence in recent years among the youth and working class of the society and has seen extensive customer orderings in recent years. I have tried to use SQL concepts like joins, regex, CTE, datetime, group by etc. to find out insights which when focused can help in understanding customer sentiment analysis. This can surely help the company to boost their sales by proper advertising and inventory maintenance, ultimately having a win-win situation for both parties.

I am using two csv data files items and orders. Using the import data wizard, I directly imported the data after creating the required tables and their column values.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

**Finding distinct items ordered by the customer.**

This helps us to understand what the customer is interested and ordering. So, accordingly we can promote relevant dishes to him.

SQL: select distinct name from items;

A screenshot of a computer

Description automatically generated

SQL: select count( distinct name) as distinct\_items\_ordered\_sofar from items;

A screenshot of a computer

Description automatically generated

**2. Veg dishes ordered so far by the customer.**

**This helped me to understand the interest of the customer in vegetarian and non-vegetarian dishes.**

SQL:

select name from items where is\_veg=1;

select count(name) from items where is\_veg=1;

A screenshot of a computer

Description automatically generated

**3. Orders with items characterized by Veg/Non-Veg ordered so far**

**This query helps us to understand that the customer prefers vegetarian food as compared to no-veg, so swiggy company should focus more on advertising and giving offers on vegetarian dishes, so that the customer feel happy and continue to order more and more.**

select count(name) as count, is\_veg from items group by is\_veg order by count(name) desc;

A screenshot of a computer

Description automatically generated

**4. Understanding the value of is\_veg=2.**

In our last query we found one product which is not falling in veg and non-veg both categories, so queried to find that it’s a brownie, looks like it a vegan dish, so they have kept it separately.

SQL: select name from items where is\_veg=2;

A screenshot of a computer

Description automatically generated

**5. Orders containing chicken**

The various chicken dishes that the customer has ordered so far.

SQL: select \* from items where name like '%chicken%';

A screenshot of a computer

Description automatically generated

**6. Average no. of items in each order.**

This query helped me to understand that the customer is always ordering in a combination of 2 items in each order, so if we promote more of combo packs to him/her, that would retain them for more time.

select count(name)/count(distinct order\_id) as avg\_items\_perorder from items;

A screenshot of a computer

Description automatically generated

**7. Number of times a particular product was ordered.**

Getting to know what a customer is ordering frequently is very crucial as that could be the retain the customer for longer time. Identifying them and getting it to the reach of the customer is very important.

Here it looks like Mac & cheese and certain other dairy related veg food is the customer’s favorite. It also clearly shows that the customer is trying new and new dishes, so maybe swiggy can display different dishes from various restaurants so that customer can keep trying new dishes and increase the order purchases, benefitting the swiggy.

SQL: select name, count(\*) as count from items group by name order by count(\*) desc;

A screenshot of a computer

Description automatically generated

**8. Number of restaurants the customer ordered from so far.**

The customer have ordered from around 49 different restaurants so far, this clearly shows that customer likes trying new and different restaurants, so maybe swiggy can reduce the delivery charges which will allow the scope of the customer to increase his/her purchasing capacity.

SQL: select count(distinct restaurant\_name) as distinct\_restaurants\_orderedsofar from orders;

A screenshot of a computer

Description automatically generated

**9. Number of Orders ordered from a restaurant.**

This query is important as it allowed me to understand that even though the customer might be interested in exploring different restaurants, “The Bowl Company” is the customer’s favorite restaurant in all his choices. So, swiggy can inform him about the membership options to avail him of better deals while ordering from that store and provide him timely offers and notifications from that store, this will help swiggy to make better profits.

SQL: select restaurant\_name, count(\*) as no\_of\_orders from orders group by restaurant\_name order by count(\*) desc;

A screenshot of a computer

Description automatically generated

**10. Monthwise number of Orders.**

This helps us to understand the trend of the order purchase by the customer, and this query tells us that the customer looks really happy and has been consistently increasing the no of orders as compared to previous months. So, swiggy company can continue its promotion to him and can offer him some membership option so that he will feel recognized.

SQL: select date\_format(order\_time,'%Y-%m') as ordered\_date, count( distinct order\_id) as no\_of\_orders from orders group by date\_format(order\_time,'%Y-%m') order by count(distinct order\_id) desc;

A screenshot of a computer

Description automatically generated

**11. Last order datetime**

This query tells us know when was the last time the customer ordered food from swiggy app. Since the data is of 2022, it shows 5th feb 2022.

SQL: select max(order\_time) from orders;

A screenshot of a computer

Description automatically generated

**12. Monthwise money spent on orders.**

This shows what part of the year is the customer spending most of his money. The customer is spending most of the money during vacation time which in India is October (because of festivals). Also, the summer-monsoon season(may-aug) also have a good record of orders, so Swiggy should advertise during that part of the time, so that they can make good profits.

SQL: select date\_format(order\_time,'%Y-%m') as ordered\_date, sum(order\_total) as no\_of\_orders from orders group by date\_format(order\_time,'%Y-%m') order by sum(order\_total) desc;

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**13. Weekday wise sales analysis.**

This sales analysis lets us know that customers are ordering heavily on Saturday, Thursday and Sunday. So better combo deals can be introduced so that the customer can buy more along with his regular purchases. Also, Monday records lowest order purchase so maybe swiggy can promote dishes with more discount on that day so that customer can increase his purchase on that day.

A screenshot of a computer

Description automatically generated

**14. Average value of order purchase.**

The average order value that the customer usually purchasing is around 293 rupees, so it would be better if swiggy can bundle some good deals under 300 so that it helps that customer to easily pick a bundle of his choice.

SQL: select sum(order\_total)/count(distinct order\_id) as avg\_order\_value from orders;

A screenshot of a computer

Description automatically generated

**15. Yearly money spent on orders.**

Year-wise analysis of the order purchase data of the customer. The customer looks like enjoying the swiggy service as he has consistently increased the purchase rate from previous years.

SQL**:** select date\_format(order\_time,'%Y') as year\_order, sum(order\_total) as no\_of\_orders from orders group by date\_format(order\_time,'%Y') order by sum(order\_total) desc; A screenshot of a computer

Description automatically generated

**16. Yearly money spend on orders with rankings**

I used the concept of CTE(common table expression) here to create a temporary data set that I used to find ranking on top of it. We can clearly see that 2021 is so far best in terms of the sales observed.

with year\_data as(

select date\_format(order\_time,'%Y') as year\_order, sum(order\_total) as total\_revenue from orders

group by date\_format(order\_time,'%Y'))

select year\_order, total\_revenue , rank() over(order by total\_revenue desc) as rankings from year\_data;

A screenshot of a computer

Description automatically generated

**17. Restaurant rankings based on money spent**

Here I used CTE to find the customer’s favorite restaurant along with the money spent on it.

SQL: with restaurant\_revenue as(

select restaurant\_name, sum(order\_total) as total\_money\_spent from orders group by restaurant\_name)

select restaurant\_name, total\_money\_spent,rank() over(order by total\_money\_spent desc) as ranking from restaurant\_revenue;

A screenshot of a computer

Description automatically generated

**18. Using Joins to fetch details about each order**

I executed join statements here to establish the relationship between orders and items table to fetch details about each order.

SQL: select a.name, a.is\_veg,b.order\_total,b.restaurant\_name,b.order\_time from items a

join orders b on a.order\_id=b.order\_id;

A screenshot of a computer

Description automatically generated

**19. Identifying combinations of items ordered**

I used the concept of self joins to identify the items which the customer is ordering in each order, Swiggy can use this data to bundle these items together to make it easy for the customer to pick the combos, making swiggy to make more benefits.

SQL: select a.order\_id,a.name as name1,b.name as name2,concat(a.name," - ",b.name) as combo\_ordered

from items a join items b

on a.order\_id=b.order\_id

where a.name!=b.name;

A screenshot of a computer

Description automatically generated