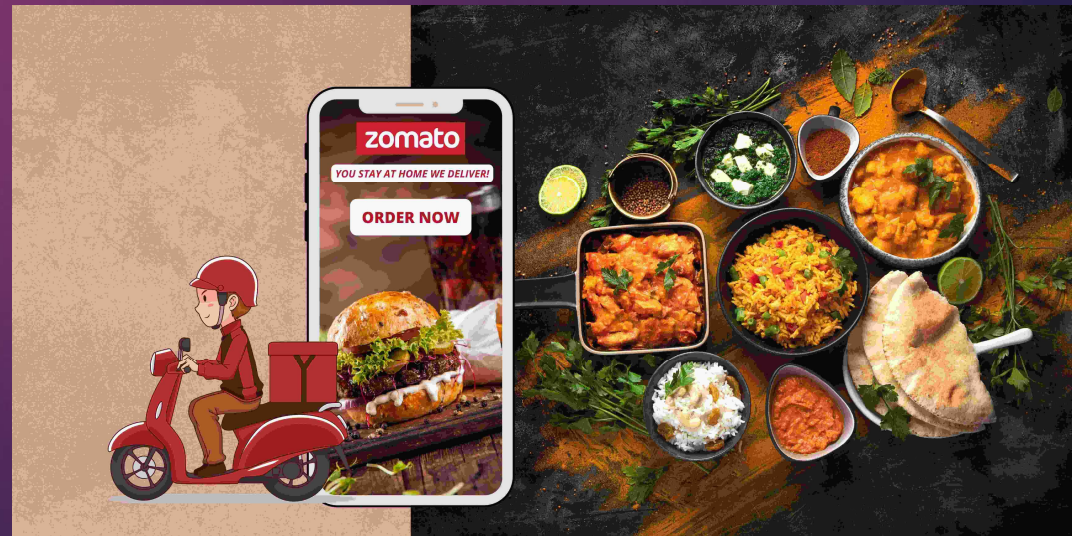


Comprehensive Data Analysis using SQL Zomato-A Food Delivery Platform



PROJECT OVERVIEW

- The goal of this project is to analyze Zomato's restaurant data to gain insights into food trends, customer preferences, and restaurant performance. By leveraging SQL, we aim to extract meaningful patterns from the dataset, which can guide business decisions, improve customer experiences, and optimize marketing strategies.
- This project demonstrates my SQL problem-solving skills through by analyzing of data for Zomato, a popular food delivery company in India. The project involves setting up the database, importing data, handling null values, and solving a variety of business problems using complex SQL queries.
- SQL offers a range of capabilities, such as filtering, joining, aggregating, and using analytical functions, that enable comprehensive data analysis. In this project I have gone from basics to Advanced SQL Queries which involve Data Cleaning, Data Filtering using CRUD, Aggregation & Grouping, Data Relationships, Sorting& Ranking, Date-Time Analysis & Advanced Analysis with CTEs.
- In this project I have worked on 5 different tables namely customers, restaurants, orders, deliveries, riders and it contains info on orders, restaurant names, order id, rider id, delivery details ...etc
- I will be explaining the steps I have done to create a database and create the nabove mentioned 5 tables and process of how data is imported from local file.

DATABASE CREATION

- Step1:After logging into the SQL create a new schema and give the name as Zomato Project.
- Step2:After that click on Apply and you will get the Zomato Project highlighted in the schemas tab
- Step3:After creation of Schema, Database needs to be created where we will be analysing all our business case studies and solutions using SQL Queries.
- Step4:Use Query **"use Zomato project"** to make sure that you are using the right database.

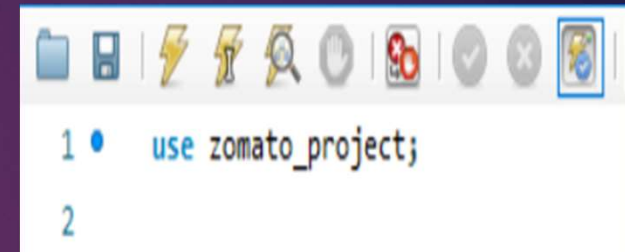
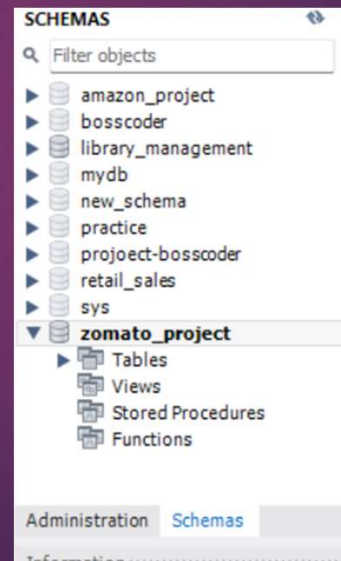
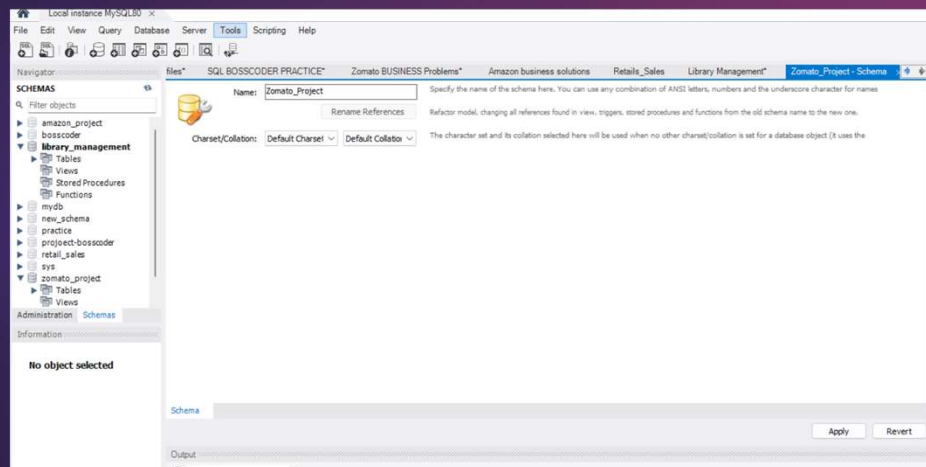


TABLE CREATION

- As mentioned earlier I have worked with 5 different tables and below are the queries used to create those tables.

- Customer Table:

```
create table customers  
(customer_id int primary key,  
customer_name varchar(25),  
reg_date date);
```

- Deliveries Table:

```
create table deliveries(delivery_id int primary key,  
order_id int ,  
delivery_status varchar(35),  
delivery_time time,  
rider_id int );
```

- Orders Table:

```
create table orders(order_id int primary key,  
customer_id int ,  
restaurant_id int,  
order_item varchar(45),  
order_date date,order_time time,  
order_status varchar(20),  
total_amount float);
```

TABLE CREATION

- Restaurant Table:

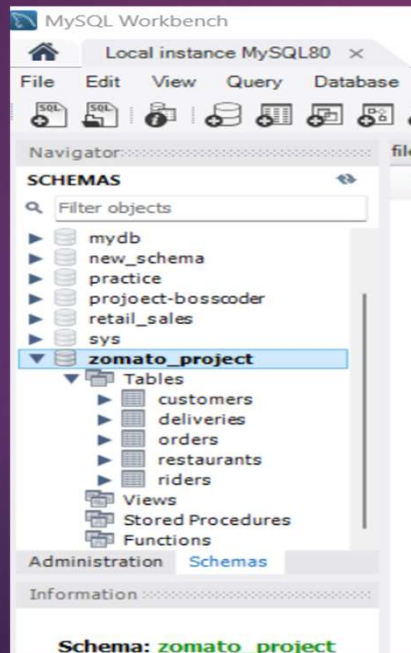
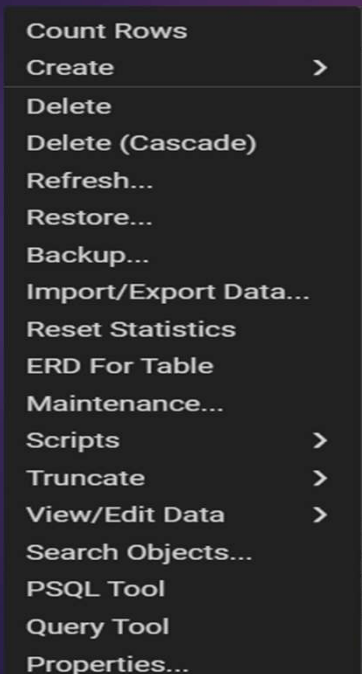
```
create table restaurants  
(restaurant_id int primary key,  
restaurant_name varchar(35),  
city varchar(20),  
opening_hours varchar(55));
```

- Riders Table:

```
create table riders  
(rider_id int primary key,  
rider_name varchar(30),  
sign_up_date date);
```

DATA IMPORT/DATA CLEANING

- After creation of table, data should be imported using “Table data import Wizard Option” or “Import/Export Data”
- Once the tables are imported all the tables are visible under the schemas tab as shown below
- Data cleaning should be performed by checking any null values or irrelevant data types.
- If exists delete or replace the null values using **DROP or UPDATE** command
- Change the data type using **UPDATE** command



DATA RETRIEVAL

- After importing the data use **SELECT** command to retrieve the data.

```
select * from customers;  
select * from restaurants;  
select * from orders;  
select * from riders;  
select * from deliveries;
```

Data Output				Messages	Notifications
				SQL	
	customer_id [PK] integer	customer_name character varying (25)	reg_date date		
1	1	Arjun Mehta	2023-03-10		
2	2	Priya Sharma	2023-04-15		
3	3	Vikram Singh	2023-05-01		
4	4	Ritu Patel	2023-06-05		
5	5	Aman Gupta	2023-07-12		
6	6	Sneha Desai	2023-08-18		
7	7	Rahul Verma	2023-09-05		
8	8	Neha Joshi	2023-10-10		
9	9	Karan Kapoor	2023-11-15		
10	10	Divya Nair	2023-12-20		
11	11	Rohan Iyer	2024-01-02		
12	12	Anjali Saxena	2024-01-08		
13	13	Sameer Khan	2024-01-12		
14	14	Pooja Rao	2024-01-15		
15	15	Nikhil Jain	2024-01-18		
16	16	Aarti Yadav	2024-01-22		
17	17	Manish Kulkarni	2024-01-26		
18	18	Shreya Ghosh	2024-01-30		
19	19	Aakash Dubey	2024-02-02		
Total rows: 33 of 33				Query complete 00:00:00.147	Ln 39, Col 1

BUSINESS PROBLEMS















- Qn1: Write a query to find the top 5 most frequently ordered dishes by the customer "Arjun Mehta" in-- the last 1 year

QUERY

```
select customer_name,dishes,dishes_count from(
select c.customer_id,c.customer_name,
count(o.order_item) as dishes_count,o.order_item as dishes,
dense_rank() over (order by count(o.order_item) desc) as rank
from customers c
join orders o
on c.customer_id=o.customer_id
where o.order_date between '2023-09-01' and '2024-09-30' and c.customer_name='Arjun Mehta'
group by 2,1,4
order by 1,3 desc)
where rank<=5;
```

OUTPUT

Data Output Messages Notifications			
         SQL			
	customer_name character varying (25) 	dishes character varying (45) 	dishes_count bigint 
1	Arjun Mehta	Paneer Butter Masala	9
2	Arjun Mehta	Mutton Rogan Josh	8
3	Arjun Mehta	Masala Dosa	8
4	Arjun Mehta	Mutton Biryani	7
5	Arjun Mehta	Pasta Alfredo	7
6	Arjun Mehta	Chicken Biryani	6
7	Arjun Mehta	Lamb Kebab	5

- Qn:2 Identify the time slots during which the most orders are placed, based on 2-hour intervals.

QUERY

```
select count(order_id),  
       floor(extract(hour from order_time)/2)*2 as start_time,  
       floor(extract(hour from order_time)/2)*2 +2 as end_time  
from orders  
group by 2,3  
order by 1 desc;
```

OUTPUT













Data Output				Messages	Notifications
	count bigint	start_time numeric	end_time numeric		
1	1188	14	16		
2	1136	18	20		
3	1123	22	24		
4	1115	12	14		
5	1107	10	12		
6	1089	20	22		
7	1080	16	18		
8	1076	8	10		
9	1074	6	8		
10	12	0	2		

- Qn:3 Find the average order value (AOV) per customer who has placed more than 750 orders.
- Return: customer_name, aov (average order value). in DESC of AOV.

QUERY

```
select c.customer_name, floor(avg(o.total_amount)) as avg_order_value,
count(o.order_id) as total_orders
from orders o
join customers c
on o.customer_id=c.customer_id
group by 1
having count(o.order_id) >=750
order by 2,3 desc;
```

OUTPUT

Data Output Messages Notifications			
         SQL			
	customer_name character varying (25) 	avg_order_value double precision 	total_orders bigint 
1	Sneha Desai	333	807
2	Aman Gupta	333	772
3	Rahul Verma	339	773

- Qn:4 List the customers who have spent more than 100K in total on food orders.
- Return: customer_name, customer_id,total_amount

QUERY

```
select o.customer_id,c.customer_name,sum(o.total_amount) as total_spent_food
from orders o
join customers c
on o.customer_id=c.customer_id
group by 1,2
having sum(o.total_amount)>100000
order by 3 desc;
```

OUTPUT

	customer_id integer	customer_name character varying (25)	total_spent_food double precision
1	6	Sneha Desai	269197
2	7	Rahul Verma	262094
3	5	Aman Gupta	257322
4	9	Karan Kapoor	244287
5	8	Neha Joshi	243223
6	4	Ritu Patel	242681
7	15	Nikhil Jain	168782
8	17	Manish Kulkarni	162552
9	22	Kavita Malhotra	154737
10	20	Bhavna Agarwal	154368
11	19	Aakash Dubey	150866
12	18	Shreya Ghosh	150807
13	16	Aarti Yadav	146145
14	21	Ramesh Chandra	143571

- Qn:5 Write a query to find orders that were placed but not delivered.
- Return: restaurant_name, city, and the number of not delivered orders.

QUERY

```
select r.restaurant_name,r.city,count(o.order_id)
from orders as o
left join restaurants as r
on r.restaurant_id=o.restaurant_id
left join deliveries as d
on d.order_id=o.order_id
where d.delivery_id is null
group by 1,2
order by 3 desc;
```

OUTPUT

Data Output Messages Notifications			
SQL			
	restaurant_name character varying (35)	city character varying (20)	count bigint
1	Britannia & Co.	Mumbai	16
2	Indigo	Mumbai	13
3	Bademiya	Mumbai	13
4	The Bombay Canteen	Mumbai	12
5	Mahesh Lunch Home	Mumbai	12
6	Masala Library	Mumbai	12
7	Ziya	Mumbai	11
8	Leopold Cafe	Mumbai	10
9	Gajalee	Mumbai	9
10	Yauatcha	Mumbai	8
11	Nagarjuna	Bengaluru	7
12	The Spicy Venue	Hvderabad	6
Total rows: 51 of 51 Query complete 00:00:00.111 Ln 134, Col 66			

- Qn6: Rank restaurants by their total revenue from the last year.
- Return: restaurant_name, total_revenue, and their rank within their city.

QUERY

```
with cte
as
(select r.restaurant_name,r.city,sum(o.total_amount) as total_reveune,
rank() over (order by sum(o.total_amount) desc) as rank
from restaurants r
join orders o
on r.restaurant_id=o.restaurant_id
group by 1,2)
select * from cte
```

OUTPUT

	restaurant_name character varying (35)	city character varying (20)	total_reveune double precision	rank bigint
1	Bademiya	Mumbai	157583	1
2	Gajalee	Mumbai	157162	2
3	Indigo	Mumbai	156467	3
4	Masala Library	Mumbai	155478	4
5	Britannia & Co.	Mumbai	152570	5
6	Yauatcha	Mumbai	151899	6
7	The Bombay Canteen	Mumbai	151472	7
8	Leopold Cafe	Mumbai	148033	8
9	Mahesh Lunch Home	Mumbai	146355	9
10	Ziya	Mumbai	145102	10

- Qn7: Identify the most popular dish in each city based on the number of orders.

QUERY

```
with cte as
(select r.city, count(o.order_id) as number_of_orders, o.order_item as dishes,
rank() over (partition by r.city order by count(o.order_id) desc) as rank
from orders o
join restaurants r
using(restaurant_id)
group by 1,3
order by 2 desc)
select * from cte
where rank=1;
```

OUTPUT

Data Output

Messages

Notifications

- Q8. Customer Churn
- Question: Find customers who haven't placed an order in 2024 but did in 2023

QUERY

```
select distinct(c.customer_id),c.customer_name
from orders o
join customers c
using (customer_id)
where order_date between '2023-01-01' and '2023-12-31'
and
customer_id not in(select distinct customer_id from orders
where order_date between '2024-01-01' and '2024-12-31')
order by 1;
```

OUTPUT

	customer_id [PK] integer	customer_name character varying (25)
1	5	Aman Gupta
2	6	Sneha Desai
3	9	Karan Kapoor
4	11	Rohan Iyer
5	18	Shreya Ghosh
6	21	Ramesh Chandra
7	22	Kavita Malhotra
8	23	Ashish Mishra
9	24	Megha Sinha

- Q9. Cancellation Rate Comparison
- Question: Calculate and compare the order cancellation rate for each restaurant between the current year and the previous year

QUERY

```
with cancel_ratio_23 as
(select o.restaurant_id,count(o.order_id) total_orders,
count(case when d.delivery_id is null then 1 end) not_delivered
from orders o
left join deliveries d
on o.order_id=d.order_id
where extract(year from order_date)=2023
group by 1),
cancel_ratio_24
as
(select o.restaurant_id,count(o.order_id )as total_orders,
count(case when d.delivery_id is null then 1 end) as not_delivered
from orders o
left join deliveries d
on o.order_id=d.order_id
where extract (year from order_date)=2024
group by 1),
```

QUERY

```
as(
select restaurant_id,total_orders,not_delivered,
round(not_delivered::numeric/total_orders::numeric*100,2) as cancellation_ratio
from cancel_ratio_23),
current_year_data
as(
select restaurant_id,total_orders,not_delivered,
round(not_delivered::numeric/total_orders::numeric*100,2) as cancellation_ratio
from cancel_ratio_24)
select cd.restaurant_id,cd.cancellation_ratio,ld.cancellation_ratio
from current_year_data as cd
join last_year_data as ld
using(restaurant_id)
group by 1,2,3;
```

OUTPUT

	restaurant_id integer	current_year numeric	prev_year numeric
1	2	0.00	2.10
2	3	0.00	2.69
3	4	0.00	2.45
4	5	0.00	1.89
5	6	0.00	2.47
6	7	0.00	2.63
7	8	0.00	1.72
8	9	0.00	3.35
9	10	0.00	2.81
10	11	0.00	1.59
11	16	0.00	1.35
12	21	0.00	6.67
13	22	0.00	0.00
14	24	0.00	3.95
15	32	0.00	1.41
16	35	0.00	2.26
17	49	0.00	3.20
18	54	0.00	2.82

- Q10. Rider Average Delivery Time
- Question: Determine each rider's average delivery time.

QUERY

```
select distinct(rider_id),
round(avg(extract(minutes from time_taken_to_deliver)),2) as avg_delivery_time
from
(select d.rider_id,
o.order_time,d.delivery_time,d.delivery_status,d.delivery_time-o.order_time as time_taken_to_del
from orders o
join deliveries d
on o.order_id=d.order_id
group by 1,2,3,4) as x
where delivery_status='Delivered'
group by 1
order by 2 desc;
```

OUTPUT

Data Output		Messages	Notifications
	rider_id integer	avg_delivery_time numeric	
1	6	31.09	
2	5	30.33	
3	11	30.19	
4	8	30.18	
5	15	30.02	
6	7	29.31	
7	12	29.26	
8	10	29.11	
9	3	28.95	
10	9	28.70	
11	14	28.25	
12	13	28.24	
13	4	28.19	
14	2	26.62	
15	1	26.59	

- Q11: Calculate each restaurant's growth ratio based on the total number of delivered orders since its joining.

QUERY

```
with cte1 as
(select r.restaurant_name,count(o.order_id) as total_delivered_orders from
restaurants r
join orders o
on r.restaurant_id=o.restaurant_id
join deliveries d
on d.order_id=o.order_id
where d.delivery_status='Delivered'
group by 1
order by 2 desc),
cte2 as
(select r.restaurant_name,count(o.order_id) as total_received_orders from
restaurants r
join orders o
on r.restaurant_id=o.restaurant_id
join deliveries d
on d.order_id=o.order_id
group by 1
order by 2 desc)
select restaurant_name,cte1.total_delivered_orders,cte2.total_received_orders,
round(cte1.total_delivered_orders::numeric/cte2.total_received_orders::numeric *100,2)
as growth_ratio
from cte1
join cte2
using(restaurant_name)
```

- Q11: Calculate each restaurant's growth ratio based on the total number of delivered orders since its joining.

OUTPUT

Data Output Messages Notifications					
<div> <div>≡</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> <div>SQL</div> </div>					
	restaurant_name character varying (35)	total_delivered_orders bigint	total_received_orders bigint	growth_ratio numeric	
1	Moti Mahal	60	60	100.00	
2	MTR	73	76	96.05	
3	Paradise Biryani	123	129	95.35	
4	Almond House	132	140	94.29	
5	Ohri's Jiva Imperia	132	140	94.29	
6	Windmills Craftworks	123	131	93.89	
7	Peshawri	120	128	93.75	
8	Sagar Ratna	60	64	93.75	
9	Brahmin's Coffee Bar	119	127	93.70	
10	The Spicy Venue	118	126	93.65	
11	Saravana Bhavan	73	78	93.59	
12	Annalakshmi	131	140	93.57	
13	Diggin	72	77	93.51	
14	Cafe Bahar	127	136	93.38	
15	Leopold Cafe	436	467	93.36	
16	Cafe Delhi Heights	55	59	93.22	
17	Britannia & Co.	431	463	93.09	
18	Yauatcha	428	460	93.04	
Total rows: 61 of 61 Query complete 00:00:00.118 Ln 202, Col 1					

- Question13: Calculate each rider's total monthly earnings, assuming they earn 8% of the order amount.

QUERY

```
select *,(t2.total_revenue*.08)as total_monthly_earnings
from (
select d.rider_id,r.rider_name,extract (year from o.order_date) as year,
extract (month from o.order_date) as month,
sum(total_amount) as total_revenue
from deliveries d
join riders r
on d.rider_id=r.rider_id
join orders o
on d.order_id=o.order_id
-- where d.delivery_status='Delivered'
group by 1,2,3,4
order by 1,2,3,4) as t2
group by 1,2,3,4,5,6
order by 1,2,3,4,5,6 desc;
```

OUTPUT

Data Output Messages Notifications						
	rider_id integer	rider_name character varying (30)	year numeric	month numeric	total_revenue double precision	total_monthly_earnings double precision
1	1	Ravi Kumar	2023	1	28277	2262.16
2	1	Ravi Kumar	2023	2	13593	1087.44
3	1	Ravi Kumar	2023	3	18456	1476.48
4	1	Ravi Kumar	2023	4	23080	1846.4
5	1	Ravi Kumar	2023	5	20306	1624.48
6	1	Ravi Kumar	2023	6	22267	1781.3600000000001
7	1	Ravi Kumar	2023	7	26043	2083.44
8	1	Ravi Kumar	2023	8	24639	1971.1200000000001
9	1	Ravi Kumar	2023	9	15531	1242.48
10	1	Ravi Kumar	2023	10	22530	1802.4
11	1	Ravi Kumar	2023	11	16987	1358.96
12	1	Ravi Kumar	2023	12	17905	1432.4
13	2	Anil Singh	2023	1	20234	1618.72

Total rows: 189 of 189 Query complete 00:00:00.139 Ln 266, Col 1

- Q14. Rider Ratings Analysis
- Question: Find the number of 5-star, 4-star, and 3-star ratings each rider has. Riders receive ratings based on delivery time:
- ● 5-star: Delivered in less than 15 minutes
- ● 4-star: Delivered between 15 and 20 minutes
- ● 3-star: Delivered after 20 minutes

QUERY

```
select rt.rider_id,rt.rating,count(rt.rating) as rating_count
from
(select o.order_id,d.delivery_id,d.rider_id,o.order_time,d.delivery_time,d.delivery_status,
(d.delivery_time-o.order_time) as time_taken_to_deliver,
extract(epoch from (d.delivery_time-o.order_time)/60) as delivery_time_in_minutes,
case when extract(epoch from (d.delivery_time-o.order_time)/60)<15 then '5-star'
      when extract(epoch from (d.delivery_time-o.order_time)/60) between 15 and 20 then '4-star'
      else '3-star' end as rating
from orders o
join deliveries d
on o.order_id=d.order_id
group by 1,2,3,4,5,6,9
order by 1,2,3,4,5,6) as rt
where rt.delivery_status='Delivered'
group by 1,2
order by 1,2,3;
```

OUTPUT

Data Output

Messages

Notifications

- Q15. Order Frequency by Day-
- Question: Analyze order frequency per day of the week and identify the peak day for each restaurant

QUERY

```
with cte as
(select r.restaurant_name,count(o.order_id) as total_no_of_orders_day,
to_char(o.order_date,'Day')as Day,
dense_rank() over (partition by r.restaurant_name order by count(o.order_id) desc) as rank
from orders o
join restaurants r
on o.restaurant_id=r.restaurant_id
group by 1,3
order by 1,2 desc)
select *
from cte
where rank=1;
```

OUTPUT


Data Output Messages Notifications				
	restaurant_name character varying (35)	total_no_of_orders_day bigint	day text	rank bigint
1	Almond House	28	Thursday	1
2	Almond House	28	Tuesday	1
3	Annalakshmi	25	Sunday	1
4	Bademiya	82	Wednes...	1
5	Bawarchi	22	Sunday	1
6	Bawarchi	22	Monday	1
7	Brahmin's Coffee Bar	26	Monday	1
8	Britannia & Co.	85	Saturday	1
9	Bukhara	15	Monday	1
10	Cafe Bahar	25	Monday	1
11	Cafe Delhi Heights	11	Friday	1
12	Corner House	24	Wednes...	1
13	Dakshin	28	Friday	1
14	Diggin	13	Thursday	1
15	Dindigul Thalappakatti	25	Monday	1
16	Dindigul Thalappakatti	25	Sunday	1
17	Ebony	27	Wednes...	1
18	Empire Restaurant	13	Sunday	1
19	Farzi Cafe	16	Thursday	1
Total rows: 69 of 69 Query complete 00:00:00.109 Ln 311, Col 1				

- Q16: Calculate the total revenue generated by each customer over all their orders

QUERY

```
select c.customer_name,c.customer_id,sum(o.total_amount) from orders o
join customers c
using(customer_id)
group by 1,2
order by 3 desc;
```

OUTPUT

Data Output Messages Notifications			
			
	customer_name character varying (25)	customer_id [PK] integer	sum double precision
1	Sneha Desai	6	269197
2	Rahul Verma	7	262094
3	Aman Gupta	5	257322
4	Karan Kapoor	9	244287
5	Neha Joshi	8	243223
6	Ritu Patel	4	242681
7	Nikhil Jain	15	168782
8	Manish Kulkarni	17	162552
9	Kavita Malhotra	22	154737
10	Bhavna Agarwal	20	154368
11	Aakash Dubey	19	150866
12	Shreya Ghosh	18	150807
13	Aarti Yadav	16	146145
14	Ramesh Chandra	21	143571
15	Arjun Mehta	1	66286
16	Anjali Saxena	12	64788
17	Vikram Singh	3	59750
18	Divya Nair	10	59235
Total rows: 24 of 24 Query complete 00:00:00.137 Ln 326, Col 1			

- Q17. Monthly Sales Trends-- Identify sales trends by comparing each month's total sales to the previous month.

QUERY

```
with cte as(
select
extract (year from order_date) as year,
extract (month from order_date) as month,
to_char (order_date,'Month') as name_of_month,
sum(total_amount) as total_sales,
lag(sum(total_amount), 1)
over (order by extract (year from order_date),
extract (month from order_date)) as prev_month_sales
from orders
group by 1,2,3)
select *,((total_sales-prev_month_sales)/total_sales)*100 as sales_trend
from cte;
```

OUTPUT

	year	month	name_of_month	total_sales	prev_month_sales	sales_trend
	numeric	numeric	text	double precision	double precision	double precision
1	2023	1	January	275656	[null]	[null]
2	2023	2	February	233439	275656	-18.08481016453977
3	2023	3	March	288309	233439	19.03166394389353
4	2023	4	April	271615	288309	-6.146199583969957
5	2023	5	May	276827	271615	1.8827643257341227
6	2023	6	June	247780	276827	-11.722899346194204
7	2023	7	July	284818	247780	13.004093842383558
8	2023	8	August	255234	284818	-11.590932242569565
9	2023	9	September	259743	255234	1.7359466857624652
10	2023	10	October	286519	259743	9.345279021635564
11	2023	11	November	264235	286519	-8.43340208526501
12	2023	12	December	276097	264235	4.296316149758962
13	2024	1	January	7944	276097	-3375.541289023162

- Q18. Rider Efficiency
- Question: Evaluate rider efficiency by determining average delivery times and identifying those with the lowest and highest averages.

QUERY

```
with cte as
(select o.order_id,d.rider_id,o.order_time,d.delivery_time,
(d.delivery_time-o.order_time) as time_difference,
extract(epoch from (d.delivery_time-o.order_time+
case when d.delivery_time<o.order_time then interval '1 day' else
interval '0 day' end))/60) as time_diff_inmin
from orders o
join deliveries d
on o.order_id=d.order_id
where d.delivery_status='Delivered'
group by 1,2,3,4
order by 1,2,3,4),
riders_time
as(
select rider_id,round(avg(time_diff_inmin),2) as avg_del_time
from cte
group by 1)
select min(avg_del_time),max(avg_del_time)
from riders_time;
```

OUTPUT

Data Output		Messages	Notifications
	min numeric	max numeric	
1	32.43	51.78	

- Question19: Track the popularity of specific order items over time
- and identify seasonal demand spikes.

QUERY

```
with cte as(select *,extract(month from order_date) as month,
case when extract(month from order_date) between 4 and 6 then 'Spring'
when extract(month from order_date) > 6 and extract(month from order_date)< 9 then 'Summer'
else 'Winter' end as Season
from orders
group by 1,3,4
order by 2)
select order_item,season,count(order_id)
from cte
group by 1,2
order by 1,2 ;
```

OUTPUT

Data Output				Messages	Notifications
	order_item character varying (45)	season text	count bigint		
1	Burger	Spring	96		
2	Burger	Summer	69		
3	Burger	Winter	263		
4	Butter Chicken	Spring	84		
5	Butter Chicken	Summer	52		
6	Butter Chicken	Winter	197		
7	Chicken Biryani	Spring	196		
8	Chicken Biryani	Summer	128		
9	Chicken Biryani	Winter	430		
10	Chicken Shawarma	Spring	102		
11	Chicken Shawarma	Summer	71		
12	Chicken Shawarma	Winter	253		
13	Chicken Tikka	Spring	37		
14	Chicken Tikka	Summer	17		
15	Chicken Tikka	Winter	57		
16	Chole Bhature	Spring	113		
17	Chole Bhature	Summer	63		
18	Chole Bhature	Winter	234		
Total rows: 69 of 69				Query complete 00:00:00.108	Ln 378, Col 1

- Q20. City Revenue Ranking
- Question: Rank each city based on the total revenue for the last year (2023)

QUERY

```
with cte as(select r.city,sum(o.total_amount) as total_revenue,  
extract (year from o.order_date) as year,  
rank() over(order by sum(o.total_amount) desc) from orders o  
join restaurants r  
on o.restaurant_id=r.restaurant_id  
group by 1,3)  
select city,total_revenue,year,rank  
from cte  
where year='2023'  
group by 1,2,3,4  
order by 4;
```

OUTPUT

Data Output Messages Notifications				
	city character varying (20) 🔒	total_revenue double precision 🔒	year numeric 🔒	rank bigint 🔒
1	Mumbai	1517450	2023	1
2	Bengaluru	719062	2023	2
3	Delhi	393896	2023	3
4	Hyderabad	305467	2023	4
5	Chennai	284397	2023	5

KEY FINDINGS

- This project helped me analyze the data & find key insights like High Value Customers, Food sales city-wise & restaurant-wise, Most Popular dish, Customer churn rate, Average delivery time of orders, Rider efficiency in deliveries, Monthly sales trend, etc
- This project highlights my ability to handle complex SQL queries and provides solutions to real-world business problems in the context of a food delivery service like Zomato. The approach taken here demonstrates a structured problem-solving methodology, data manipulation skills, and the ability to derive actionable insights from data.
- Top-Rated Cities: Certain cities emerged as having a higher concentration of top-rated restaurants, especially in food-centric cities
- Customer Preference and Rating Patterns: Higher-rated restaurants were often in the mid to high price range, indicating a correlation between pricing and customer satisfaction.
- Seasonal and Time-Based Insights: Certain times of the year, such as festival seasons and holidays, showed a surge in reviews and higher engagement, suggesting peak dining periods. Analysis of time-of-day data revealed that most reviews and high ratings were given in the evening, aligning with peak dining hours.

KEY LEARNINGS

1.Importance of Data Quality and Cleaning:

Missing data, such as incomplete address details or unrated restaurants, impacted analysis accuracy. Cleaning and preprocessing were critical in handling inconsistencies and ensuring reliable results.

2.Correlation between Cost, Ratings, and Engagement:

A clear link was observed between restaurant pricing and ratings, suggesting that customers associate higher costs with quality, yet still prefer moderate-priced options for repeat visits.

3.Utility of SQL for Multi-Level Data Analysis:

SQL's grouping, joining, and aggregation functionalities proved invaluable for cross-referencing information, such as merging restaurant locations with ratings data or analyzing reviews by season.

4.Value of Trend and Time-Series Analysis:

Seasonal insights offered by SQL date functions provided a new perspective on dining behaviors, showing the importance of timing in restaurant performance and marketing strategies.

5.Business Insights for Strategic Decision-Making:

The analysis provided actionable insights, such as where to focus marketing efforts, potential areas for new restaurant openings, supporting strategic business decisions.

6.Usage of Advanced Queries:

This project paved the way of using basic to advanced queries including joins, ctes, aggregate functions, Sub-queries,rank()&dense-rank(),lead()&lag() functions