Zomato Performance Analysis

An Insightful Data Exploration Project Using SQL and Power BI & Microsoft Excel

SQL Query Documentation

📌 Project Summary

This project provides an in-depth analysis of Zomato's operational data, using over one million rows to uncover trends, patterns, and insights. The analysis is based on four core tables: Orders, Food, Restaurants, and Users. SQL queries were used externally for analysis, logic development, and validation of insights, which were then visualized using Power BI. The queries documented here represent the analytical foundation behind each dashboard visual.

📊 Key Insights Covered

- Total Sales & Orders
- Restaurant & Customer Network
- Monthly, Quarterly & Yearly Trends
- Year-over-Year Growth Metrics
- Food Type and City-based Sales Analysis
- User Demographics and Engagement

🛠️ Tools & Technologies Used

- **Microsoft Excel** – Quick data analysis and formatting
- **Power BI** – Data cleaning, modeling, and dashboard visualization
- **Microsoft SQL Server**– External queries for in-depth analysis and logic development
- **Microsoft Word** – Documentation of SQL queries and project summary

👤 Prepared by:

Prosenjit Majumder

www.linkedin.com/in/prosenjitmajumder

Date: April 18, 2025

Zomato Data Model Overview

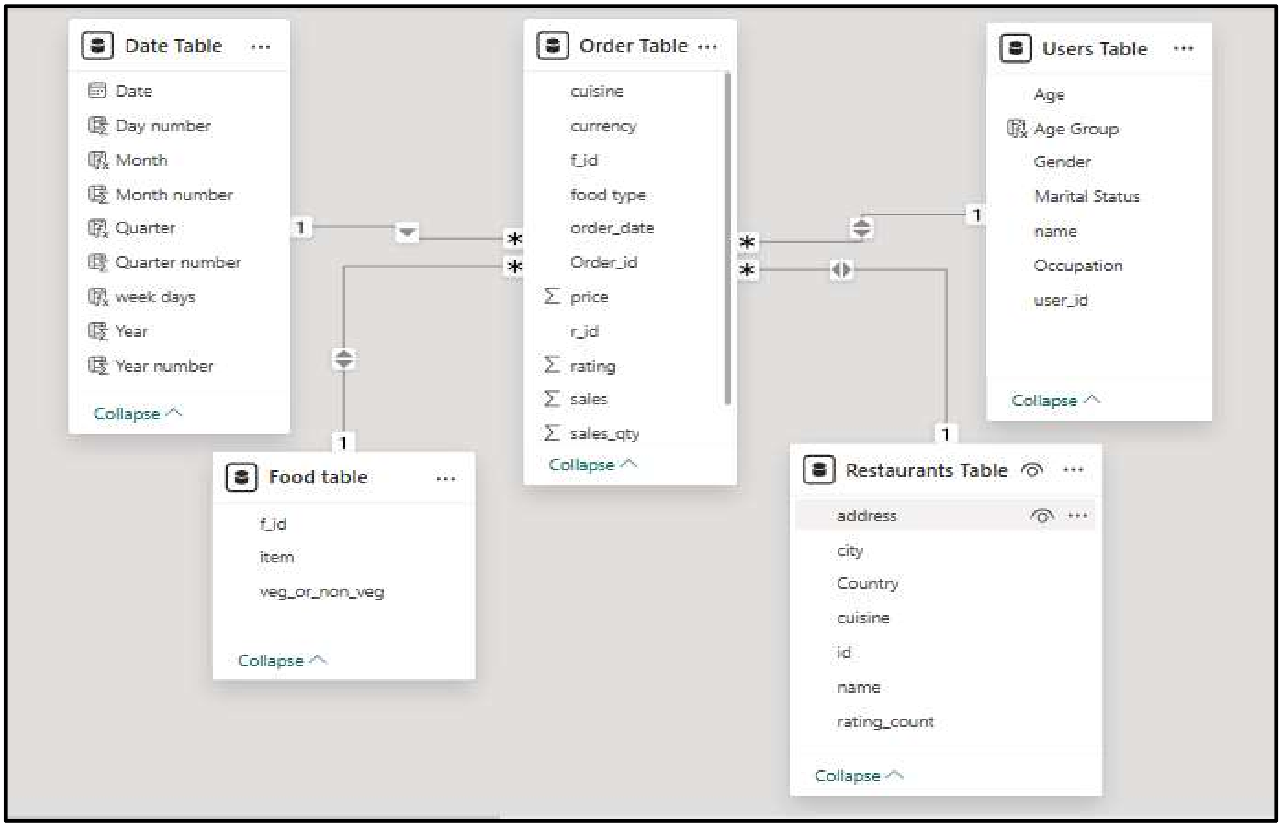
Before diving into SQL queries and analysis, it's essential to understand the structure of the database. Here's the **data model** I'm working with for this project.

Key Tables:

- **Orders (Fact Table):** Contains transactional data including price, sales quantity, and order details.
- **Users (Dimension Table):** Stores information about customers.
- **Restaurant (Dimension Table):** Holds data related to restaurants.
- **Food Type (Dimension Table):** Categorizes the type of food ordered.

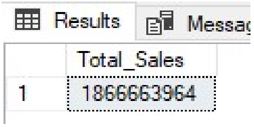
This model helps in structuring SQL queries efficiently and ensures data normalization.

Visual representation below to get a better understanding of table relationships and data flow:



Q1. Total Sales

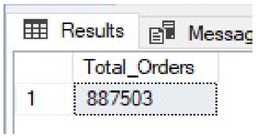
```sql
select sum(sales)as Total_Sales from Orders;
```



	Total_Sales
1	1866663964

Q2. Total Orders

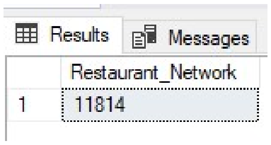
```sql
select count(distinct Order_id )as Total_Orders from Orders;
```



	Total_Orders
1	887503

Q3. Restaurant Network

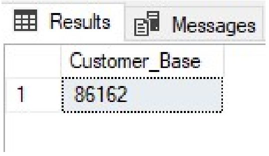
```sql
select count(distinct r_id)as Restaurant_Network from Orders;
```



	Restaurant_Network
1	11814

Q4. Customer Base

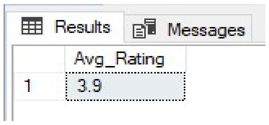
```sql
select count(distinct user_id)as Customer_Base from Orders;
```



	Customer_Base
1	86162

Q5. Avg Rating

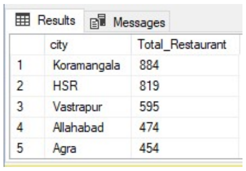
```sql
select cast(avg(rating*1.0 )as decimal(10,1))as Avg_Rating from Orders
where rating is not null;
```



	Avg_Rating
1	3.9

Q6. Total Restaurant by City

```sql
select r.city,count(distinct o.r_id) as Total_Restaurant from Orders as o
inner join Restaurants as r
on o.r_id=r.id
group by city
order by Total_Restaurant desc;
```

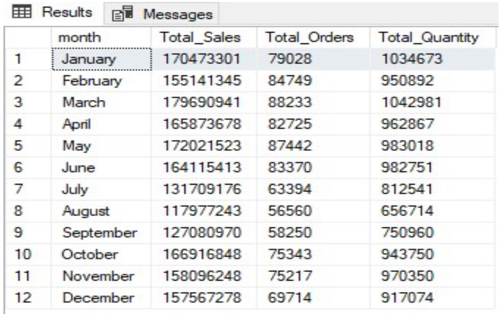


	city	Total_Restaurant
1	Koramangala	884
2	HSR	819
3	Vastrapur	595
4	Allahabad	474
5	Agra	454

Note: The screenshot above shows a partial output of the query for illustration purposes.

Q7. Total Sales,Total Orders & Total_Quantity by Month

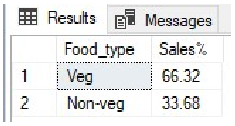
```
select month, Total_Sales,Total_Orders,Total_Quantity
from(select datename(month,order_date) as Month,
datepart(month,order_date)as Month_Number,
sum(sales)as Total_Sales,
count(distinct Order_id)as Total_Orders,
sum(sales_qty)as Total_Quantity
from Orders
group by datename(month,order_date),
datepart(month,order_date))as SQ
order by Month_Number asc;
```



	month	Total_Sales	Total_Orders	Total_Quantity
1	January	170473301	79028	1034673
2	February	155141345	84749	950892
3	March	179690941	88233	1042981
4	April	165873678	82725	962867
5	May	172021523	87442	983018
6	June	164115413	83370	982751
7	July	131709176	63394	812541
8	August	117977243	56560	656714
9	September	127080970	58250	750960
10	October	166916848	75343	943750
11	November	158096248	75217	970350
12	December	157567278	69714	917074

Q8. Total Sales% by Food type

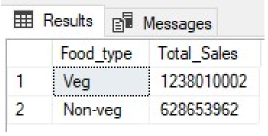
```
select Food_type,cast(sum(sales)*100.0/(select sum(sales) from Orders)as decimal(10,2))
as 'Sales%'
from Orders
group by food_type;
```



	Food_type	Sales%
1	Veg	66.32
2	Non-veg	33.68

Q9. Total Sales by Food type

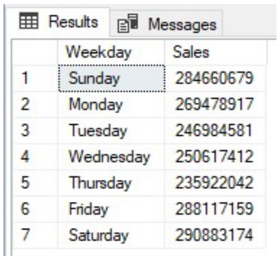
```
select Food_type,sum(sales)as Total_Sales from Orders
group by food_type;
```



	Food_type	Total_Sales
1	Veg	1238010002
2	Non-veg	628653962

Q10. Total Sales by Weekday

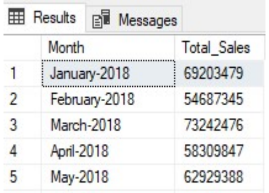
```
select datename(weekday,order_date)as Weekday, sum(sales)as Sales
from Orders
group by datename(weekday,order_date),datepart(weekday,order_date)
order by datepart(weekday,order_date) asc;
```



	Weekday	Sales
1	Sunday	284660679
2	Monday	269478917
3	Tuesday	246984581
4	Wednesday	250617412
5	Thursday	235922042
6	Friday	288117159
7	Saturday	290883174

Q11. Total Sales by Month & Year

```
select concat(datename(month,order_date),'-',year(order_date))as Month,
sum(sales)as Total_Sales from Orders
group by datename(month,order_date),datepart(month,order_date),year(order_date)
order by year(order_date),datepart(month,order_date);
```

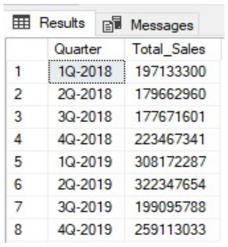


	Month	Total_Sales
1	January-2018	69203479
2	February-2018	54687345
3	March-2018	73242476
4	April-2018	58309847
5	May-2018	62929388

Note: The screenshot above shows a partial output of the query for illustration purposes.

Q12. Total Sales by Quarter & Year

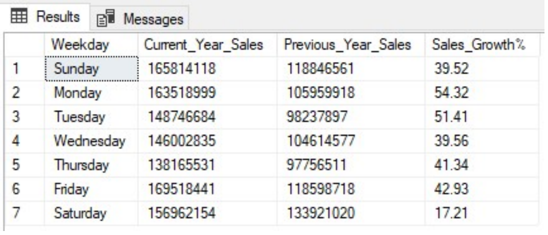
```
select concat(datename(quarter,order_date),'Q-',year(order_date))as Quarter,sum(sales)as
Total_Sales
from Orders
group by datename(quarter,order_date),year(order_date)
order by year(order_date),datename(quarter,order_date);
```



	Quarter	Total_Sales
1	1Q-2018	197133300
2	2Q-2018	179662960
3	3Q-2018	177671601
4	4Q-2018	223467341
5	1Q-2019	308172287
6	2Q-2019	322347654
7	3Q-2019	199095788
8	4Q-2019	259113033

Q13. Year over Year Sales by Weekday

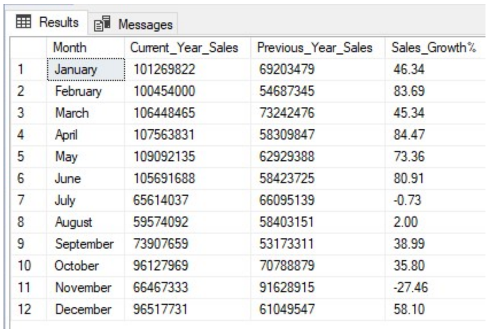
```
with Current_year as (select datename(weekday,order_date)as Weekday,
datepart(weekday,order_date)as Weekday_number,sum(sales)as Current_Year_Sales
from Orders where year(order_date)=2019
group by datename(weekday,order_date),datepart(weekday,order_date)),

Previous_year as (select datename(weekday,order_date)as Weekday,
datepart(weekday,order_date)as Weekday_number,sum(sales)as Previous_Year_Sales
from Orders where year(order_date)=2018
group by datename(weekday,order_date),datepart(weekday,order_date))

select cy.Weekday,cy.Current_Year_Sales,py.Previous_Year_Sales,,
cast((((cy.Current_Year_Sales-py.previous_year_sales)*100.0)/py.Previous_year_sales)as
decimal(10,2)) as 'Sales_Growth%'
from  current_year as cy
inner join previous_year as py
on cy.Weekday_number=py.Weekday_number
order by cy.Weekday_number;
```



	Weekday	Current_Year_Sales	Previous_Year_Sales	Sales_Growth%
1	Sunday	165814118	118846561	39.52
2	Monday	163518999	105959918	54.32
3	Tuesday	148746684	98237897	51.41
4	Wednesday	146002835	104614577	39.56
5	Thursday	138165531	97756511	41.34
6	Friday	169518441	118598718	42.93
7	Saturday	156962154	133921020	17.21

Q14. Year over Year Sales by Month

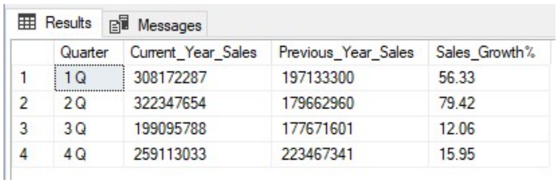
```
with Current_year as (select datename(month,order_date)as Month,
datepart(month,order_date)as Month_number,sum(sales)as Current_Year_Sales
from Orders where year(order_date)=2019
group by datename(month,order_date),datepart(month,order_date)),
Previous_year as (select datename(month,order_date)as Month,
datepart(month,order_date)as Month_number,sum(sales)as Previous_Year_Sales
from Orders where year(order_date)=2018
group by datename(month,order_date),datepart(month,order_date))

select cy.Month,cy.Current_Year_Sales,py.Previous_Year_Sales,
cast((((cy.Current_Year_Sales-py.previous_year_sales)*100.0)/py.Previous_year_sales)as
decimal(10,2)) as 'Sales_Growth%'
from  Current_year as cy
inner join Previous_year as py
on cy.Month_number=py.Month_number
order by cy.Month_number;
```



	Month	Current_Year_Sales	Previous_Year_Sales	Sales_Growth%
1	January	101269822	69203479	46.34
2	February	100454000	54687345	83.69
3	March	106448465	73242476	45.34
4	April	107563831	58309847	84.47
5	May	109092135	62929388	73.36
6	June	105691688	58423725	80.91
7	July	65614037	66095139	-0.73
8	August	59574092	58403151	2.00
9	September	73907659	53173311	38.99
10	October	96127969	70788879	35.80
11	November	66467333	91628915	-27.46
12	December	96517731	61049547	58.10

Q15. Year over Year Sales by Quarter

```
with Current_year as (select datename(quarter,order_date)as Quarter,sum(sales)as
Current_Year_Sales
from Orders where year(order_date)=2019
group by datename(quarter,order_date)),

Previous_year as (select datename(quarter,order_date)as Quarter,sum(sales)as
Previous_Year_Sales
from Orders where year(order_date)=2018
group by datename(quarter,order_date))

select concat(cy.quarter,' Q') as Quarter,
cy.Current_Year_Sales,
py.Previous_Year_Sales,
cast((((cy.Current_Year_Sales-py.previous_year_sales)*100.0)/py.Previous_year_sales)as
decimal(10,2)) as 'Sales_Growth%'
from  Current_year as cy
inner join Previous_year as py
on cy.Quarter=py.Quarter
order by cy.Quarter;
```

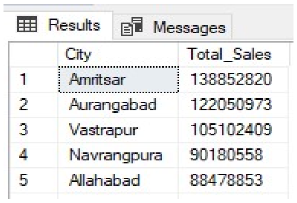


	Quarter	Current_Year_Sales	Previous_Year_Sales	Sales_Growth%
1	1 Q	308172287	197133300	56.33
2	2 Q	322347654	179662960	79.42
3	3 Q	199095788	177671601	12.06
4	4 Q	259113033	223467341	15.95

Q16. Total Sales by City

```
select r.city as City,sum(o.sales)as Total_Sales from Orders as o
```

```sql
inner join Restaurants as r
on o.r_id=r.id
group by city
Order by Total_Sales desc;
```

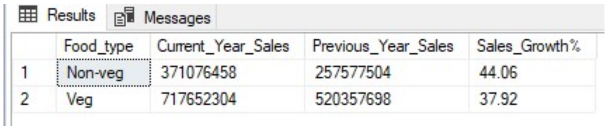


	City	Total_Sales
1	Amritsar	138852820
2	Aurangabad	122050973
3	Vastrapur	105102409
4	Navrangpura	90180558
5	Allahabad	88478853

Note: The screenshot above shows a partial output of the query for illustration purposes.

Q17. Year over Year Sales by Food Type

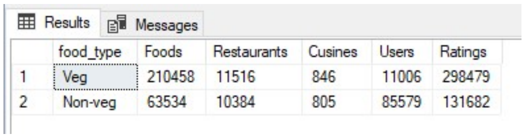
```sql
with Current_Year as (select food_type,sum(sales)as Current_Year_Sales
from Orders where year(order_date)=2019
group by food_type),

Previous_year as (select food_type,sum(sales)as Previous_Year_Sales
from Orders where year(order_date)=2018
group by food_type)

select cy.food_type,cy.Current_Year_Sales,py.Previous_Year_Sales,
cast((((cy.Current_Year_Sales-py.previous_year_sales)*100.0)/py.Previous_year_sales)as
decimal(10,2)) as 'Sales_Growth%'
from  Current_Year as cy
inner join Previous_year as py
on cy.food_type=py.food_type
order by cy.food_type;
```



	Food_type	Current_Year_Sales	Previous_Year_Sales	Sales_Growth%
1	Non-veg	371076458	257577504	44.06
2	Veg	717652304	520357698	37.92

Q18. Performance Overview by Food Type

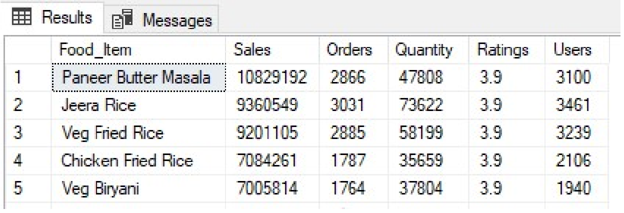
```sql
select food_type,
count(distinct f_id) as Foods,
count(distinct user_id) as Users,
count(distinct r_id) as Restaurants,
count(distinct cuisine) as Cusines,
count(distinct user_id) as Users,
count(rating) as Ratings
from Orders
group by food_type
Order by food_type desc;
```



	food_type	Foods	Restaurants	Cusines	Users	Ratings
1	Veg	210458	11516	846	11006	298479
2	Non-veg	63534	10384	805	85579	131682

Q19. Sales,Orders,Quantity,Avg Rating & Users by Food Item

```sql
select f.item as Food_Item,
sum(o.sales) as Sales,
count(distinct o.Order_id) as Orders,
sum(o.sales_qty) as Quantity,
cast(avg(o.rating*1.0)as decimal(10,1)) as Ratings,
count(distinct o.user_id)as Users
from Orders as o
inner join food as f
on o.f_id=f.f_id
group by f.item
order by Sales desc;
```

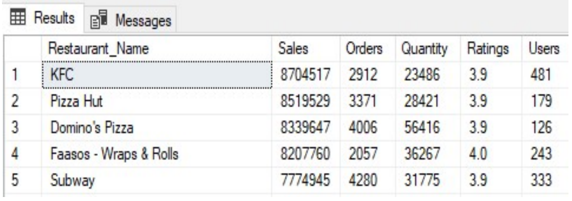


	Food_Item	Sales	Orders	Quantity	Ratings	Users
1	Paneer Butter Masala	10829192	2866	47808	3.9	3100
2	Jeera Rice	9360549	3031	73622	3.9	3461
3	Veg Fried Rice	9201105	2885	58199	3.9	3239
4	Chicken Fried Rice	7084261	1787	35659	3.9	2106
5	Veg Biryani	7005814	1764	37804	3.9	1940

Note: The screenshot above shows a partial output of the query for illustration purposes.

Q20. Sales,Orders,Quantity,Avg Rating & Users by Restaurant_Name

```
select r.name as Restaurant_Name,
sum(o.sales) as Sales,
count(distinct o.Order_id) as Orders,
sum(o.sales_qty) as Quantity,
cast(avg(o.rating*1.0)as decimal(10,1)) as Ratings,
count(distinct o.user_id)as Users
from Orders as o
inner join Restaurants as r
on o.r_id=r.id
group by r.name
order by Sales desc;
```

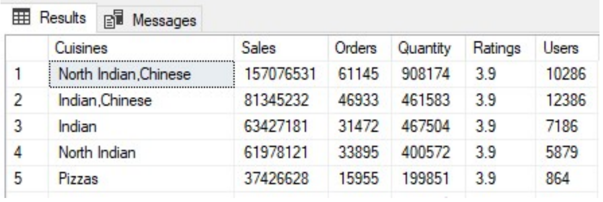


	Restaurant_Name	Sales	Orders	Quantity	Ratings	Users
1	KFC	8704517	2912	23486	3.9	481
2	Pizza Hut	8519529	3371	28421	3.9	179
3	Domino's Pizza	8339647	4006	56416	3.9	126
4	Faasos - Wraps & Rolls	8207760	2057	36267	4.0	243
5	Subway	7774945	4280	31775	3.9	333

Note: The screenshot above shows a partial output of the query for illustration purposes.

Q21. Sales,Orders,Quantity,Avg Rating & Users by Cuisines

```
select r.cuisine as Cuisines,
sum(o.sales) as Sales,
count(distinct o.Order_id) as Orders,
sum(o.sales_qty) as Quantity,
cast(avg(o.rating*1.0)as decimal(10,1)) as Ratings,
count(distinct o.user_id)as Users
from Orders as o
inner join Restaurants as r
on o.r_id=r.id
group by r.cuisine
order by Sales desc;
```

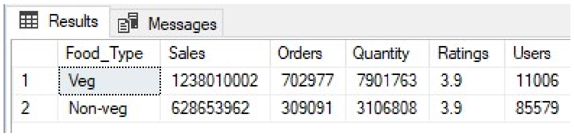


	Cuisines	Sales	Orders	Quantity	Ratings	Users
1	North Indian,Chinese	157076531	61145	908174	3.9	10286
2	Indian,Chinese	81345232	46933	461583	3.9	12386
3	Indian	63427181	31472	467504	3.9	7186
4	North Indian	61978121	33895	400572	3.9	5879
5	Pizzas	37426628	15955	199851	3.9	864

Note: The screenshot above shows a partial output of the query for illustration purposes.:s.

Q22. Sales,Orders,Quantity,Avg Rating & Users by Food Type

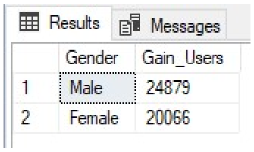
```
select food_type as Food_Type,
sum(sales) as Sales,
count(distinct Order_id) as Orders,
sum(sales_qty) as Quantity,
cast(avg(rating*1.0)as decimal(10,1)) as Ratings,
count(distinct user_id)as Users
from Orders
group by food_type
order by Sales desc;
```



	Food_Type	Sales	Orders	Quantity	Ratings	Users
1	Veg	1238010002	702977	7901763	3.9	11006
2	Non-veg	628653962	309091	3106808	3.9	85579

Q23. Year over Year Gain Users by Gender

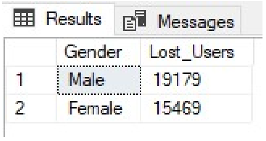
```
with Gain_User_2019 as (select distinct user_id from Orders
where datepart(year,order_date)=2019),
Gain_User_2018 as (select distinct user_id from Orders
where datepart(year,order_date)=2018)

select u.Gender,count(distinct new.user_id) as Gain_Users from Gain_User_2019 as new
left join Gain_User_2018 as old on new.user_id=old.user_id
inner join users as u on new.user_id=u.user_id
where old.user_id is null
group by u.Gender
order by Gain_Users desc;
```



	Gender	Gain_Users
1	Male	24879
2	Female	20066

Q24. Year over Year Lost Users by Gender

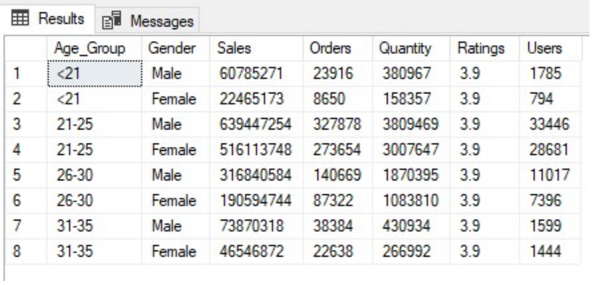
```
with User_2019 as (select distinct user_id from Orders
where datepart(year,order_date)=2019),

User_2018 as (select distinct user_id from Orders
where datepart(year,order_date)=2018)

select u.Gender,count(distinct old.user_id) as Lost_Users from User_2018 as old
left join User_2019 as new on old.user_id=new.user_id
inner join users as u on old.user_id=u.user_id
where new.user_id is null
group by u.Gender
order by Lost_Users desc;
```



	Gender	Lost_Users
1	Male	19179
2	Female	15469

Q25. Users by Age Group and Gender

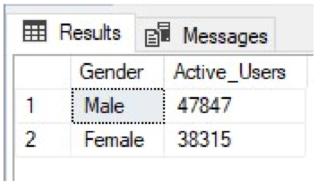
```
select case
when u.Age<=35 and u.Age>30 then '31-35'
when u.Age<31 and u.Age>25 then '26-30'
when u.Age<26 and u.Age>20 then '21-25'
when u.Age<21 then '<21'
end as Age_Group,u.Gender,
sum(o.sales) as Sales,
count(distinct o.Order_id) as Orders,
sum(o.sales_qty) as Quantity,
cast(avg(o.rating*1.0)as decimal(10,1)) as Ratings,
count(distinct o.user_id)as Users
from Orders as o
inner join users as u
on o.user_id=u.user_id
group by case when u.Age<=35 and u.Age>30 then '31-35'
when u.Age<31 and u.Age>25 then '26-30'
when u.Age<26 and u.Age>20 then '21-25'
when u.Age<21 then '<21'
end,u.Gender
Order by Age_Group asc,u.Gender desc;
```



	Age_Group	Gender	Sales	Orders	Quantity	Ratings	Users
1	<21	Male	60785271	23916	380967	3.9	1785
2	<21	Female	22465173	8650	158357	3.9	794
3	21-25	Male	639447254	327878	3809469	3.9	33446
4	21-25	Female	516113748	273654	3007647	3.9	28681
5	26-30	Male	316840584	140669	1870395	3.9	11017
6	26-30	Female	190594744	87322	1083810	3.9	7396
7	31-35	Male	73870318	38384	430934	3.9	1599
8	31-35	Female	46546872	22638	266992	3.9	1444

Q26. Active Users by Gender

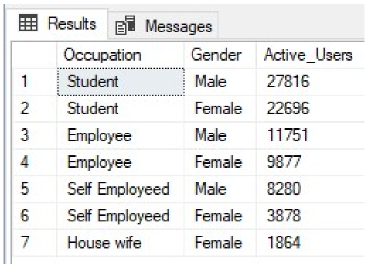
```
select u.Gender,count(distinct o.user_id) as Active_Users from Orders as o
inner join users as u
on o.user_id=u.user_id
group by u.Gender
order by Active_Users desc;
```



	Gender	Active_Users
1	Male	47847
2	Female	38315

Q27. Active Users by Occupation & Gender

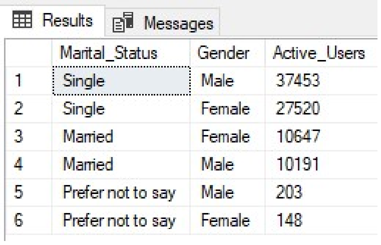
```
select u.Occupation, u.Gender,count(distinct o.user_id) as Active_Users from Orders as o
inner join users as u
on o.user_id=u.user_id
group by u.Occupation,u.Gender
order by Active_Users desc;
```



	Occupation	Gender	Active_Users
1	Student	Male	27816
2	Student	Female	22696
3	Employee	Male	11751
4	Employee	Female	9877
5	Self Employeed	Male	8280
6	Self Employeed	Female	3878
7	House wife	Female	1864

Q28. Active Users by Marital Status & Gender

```
select u.Marital_Status, u.Gender,count(distinct o.user_id) as Active_Users from Orders
as o
inner join users as u
on o.user_id=u.user_id
group by u.Marital_Status,u.Gender
order by Active_Users desc
```



	Marital_Status	Gender	Active_Users
1	Single	Male	37453
2	Single	Female	27520
3	Married	Female	10647
4	Married	Male	10191
5	Prefer not to say	Male	203
6	Prefer not to say	Female	148

🏁 Conclusion

This documentation provided a structured walkthrough of the Zomato database, starting from the data model to a series of targeted SQL queries. Each query was designed to address specific aspects of the business — including user activity, food preferences, sales trends, and restaurant performance.

By aligning SQL logic with the database structure, this project highlights how relational data can be transformed into meaningful insights.