A CASE STUDY ON ZOMATO

Introduction:

Zomato is a leading online food delivery and restaurant discovery platform that operates globally. Launched in 2008, Zomato has revolutionized the way people explore, order, and enjoy food. Users can browse through a vast database of restaurants, read reviews, view menus, and place orders for delivery or takeout through the website or mobile app. With its user-friendly interface and extensive coverage of restaurants, Zomato has become a go-to platform for food enthusiasts looking to discover new dining experiences or satisfy their cravings from the comfort of their homes.

Database Management:

This case study is performed using MySQL.

A database named “case_studies” has been created and used

```
create database case_studies;
use case_studies;
```

There are 4 tables that has been created and their subsequent values has been inserted in it.

```sql
drop table if exists sales;
CREATE TABLE sales(userid integer,created_date date,product_id integer);

INSERT INTO sales(userid,created_date,product_id)
 VALUES (1,'2017-04-19',2),
(3,'2019-12-18',1),
(2,'2020-07-20',3),
(1,'2019-10-23',2),
(1,'2018-03-19',3),
(3,'2016-12-20',2),
(1,'2016-11-09',1),
(1,'2016-05-20',3),
(2,'2017-09-24',1),
(1,'2017-03-11',2),
(1,'2016-03-11',1),
(3,'2016-11-10',1),
(3,'2017-12-07',2),
(3,'2016-12-15',2),
(2,'2017-11-08',2),
(2,'2018-09-10',3);


drop table if exists product;
CREATE TABLE product(product_id integer,product_name text,price integer);

INSERT INTO product(product_id,product_name,price)
 VALUES
(1,'p1',980),
(2,'p2',870),
(3,'p3',330);
```

```sql
drop table if exists goldusers_signup;
CREATE TABLE goldusers_signup(userid integer,gold_signup_date date);

INSERT INTO goldusers_signup(userid,gold_signup_date)
VALUES (1,'2017-09-22'),(3,'2017-04-21');

drop table if exists users;
CREATE TABLE users(userid integer,signup_date date);

INSERT INTO users(userid,signup_date)
 VALUES (1,'2014-09-02'),
(2,'2015-01-15'),
(3,'2014-04-11');
```

Now we input the command to view the complete records present in the tables using **Select** query

```
select * from sales;
select * from product;
select * from goldusers_signup;
select * from users;
```

The result after query gets executed

User table

userid	signup_date
1	2014-09-02
2	2015-01-15
3	2014-04-11

Goldusers_signup table

userid	gold_signup_date
1	2017-09-22
3	2017-04-21

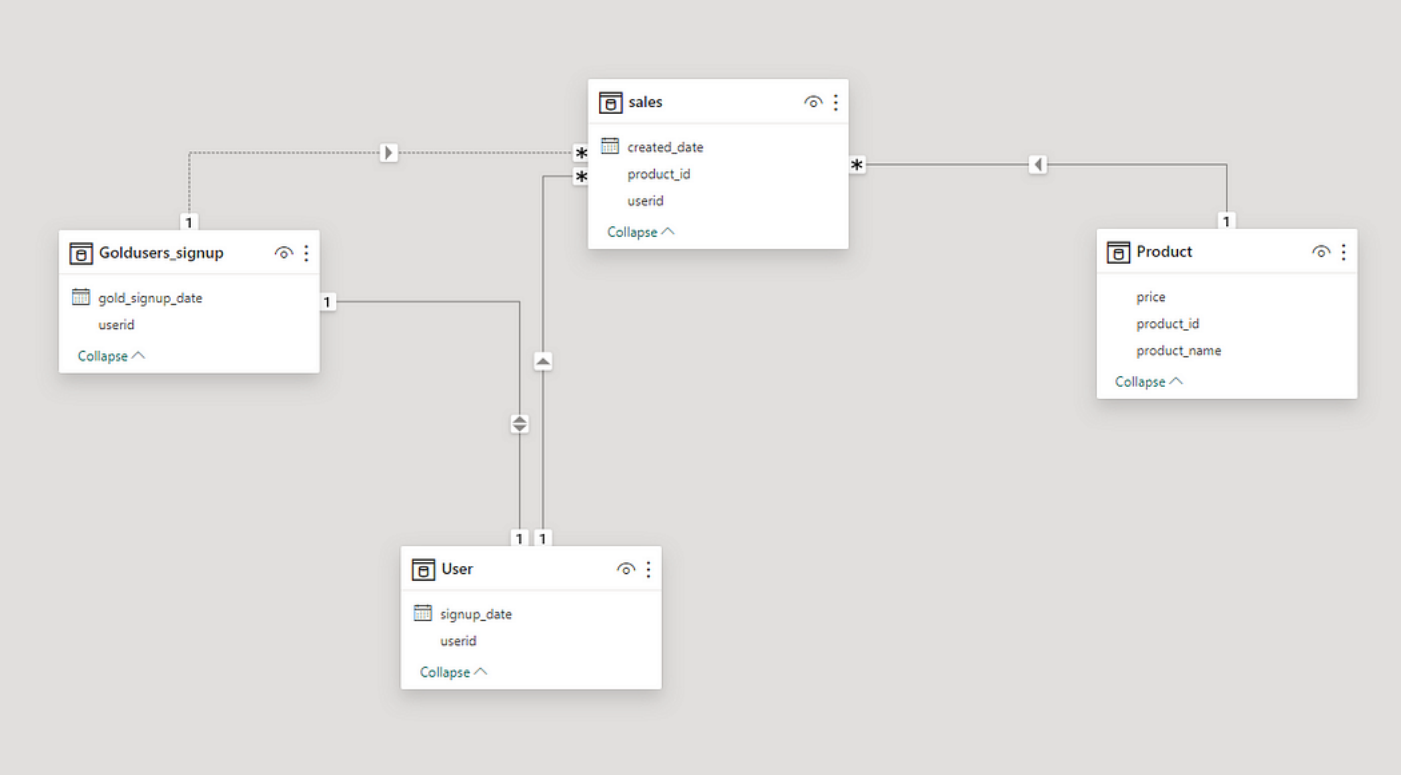
Product table

product_id	product_name	price
1	p1	980
2	p2	870
3	p3	330

Sales table

userid	created_date	product_id
1	2017-04-19	2
3	2019-12-18	1
2	2020-07-20	3
1	2019-10-23	2
1	2018-03-19	3
3	2016-12-20	2
1	2016-11-09	1
1	2016-05-20	3
2	2017-09-24	1
1	2017-03-11	2
1	2016-03-11	1
3	2016-11-10	1
3	2017-12-07	2
3	2016-12-15	2
2	2017-11-08	2
2	2018-09-10	3

The relationship diagram between the tables is described below.



This relationship diagram is generated using Microsoft PowerBI

Here, User, Products and Goldsignup_date are the dimension table where userid, goldusers_signup_date and userid are their primary keys respectively while sales table serves as the fact table.

Problem Statements:

Now the metrics/problems this case study will deal with:

1. What is the total amount each customer spent on zomato?

2. How many days has each customer visited zomato?

3. What was the first product purchased by each customer?

4. What is the most purchased item on the menu and how many times it was purchased by all the customers?

5. Which item is popular for each customer?

6. Which item was first purchased by the customer once they became the member?

7. Which item was purchased just before user became the gold member?

8. What is the total orders and amount spent for each member before they became the member?

9. If buying each product generates points, for eg. 5 Rs. = 2 Zomato points and each product have different purchasing points for eg. P1 5 Rs. = 1 Zomato point, P2 10 Rs. = 5 Zomato points and P3 5 Rs. = 1 Zomato point. So, calculate points collected by each customer and for which product most points have been collected till now?

10. In the first one year after the customer joins the gold program(including their join date)irrespective of what the customers has purchased they earn 5 zomato points for every 10 Rs. spent. Who

earned more 1 or 3 and what was their points earning in their first year?

11. Rank all the transactions of the customers based on price.

12. Rank all the transaction for each member whenever they are gold member. For every non gold member transaction marked as “na”.

Solutions:

Starting with the first problem.

Problem 1: What is the total amount each customer spent on zomato?

To calculate total amount each customer spent on Zomato, we have used **Inner join** with the sales and product table on product_id field and further used **Sum** and **Group by** function which calculates the amount spent by each customer.

```
-- Problem 1 : What is the total amount each customer spent on zomato?

select s.userid,sum(p.price) as Money_spent
from sales s
inner join product p on s.product_id = p.product_id
group by userid
order by userid;
```

Following the execution of the aforementioned query, the resultant output is provided below.

	userid	Money_spent
▶	1	5230
	2	2510
	3	4570

Conclusion: The total amount spent by Customer 1, Customer 2 and Customer 3 are Rs. 5230, Rs. 2510 and Rs. 4570 respectively with **Customer 1 being the highest paid customer** and **Customer 2 being the lowest paid customer.**

Problem 2: How many days has each customer visited zomato?

To find the number of customers who visited Zomato, we have used **Group by** function on the userid and **Count** along with **Distinct** function on the sales table.

```
-- Problem 2: How many days has each customer visited zomato?

select userid, count(distinct created_date) as No_of_days_visited_by_each_customer
from sales
group by userid;
```

The resulted output after the execution of the query is given below.

	userid	No_of_days_visited_by_each_customer
▶	1	7
	2	4
	3	5

Conclusion: Customer 1 have visited with the highest number of days ,i.e., 7 days on the Zomato app, Customer 3 have visited on 5

days while **Customer 2 have visited on 4 days which is a minimum number.**

Problem 3: What was the first product purchased by each customer?

A structured approach involving two steps is utilized to address the particular problem.

Step 1: In the first step, we have **Inner join** the sales and product table and further used the **Row_number window function partitioned by** userid and ordered by created_date. This step will provide the row_number on the basis of individual userid which is ordered by created_date.

```
-- Problem 3: What was the first product purchased by each customer?

-- 1st method

-- 1st step

select s.userid,s.created_date,p.product_id,p.product_name,p.price, row_number() over(partition by userid order by created_date) as rn
from sales s
inner join product p on s.product_id = p.product_id;
```

Below, you'll find the output generated upon executing the aforementioned query.

userid	created_date	product_id	product_name	price	rn
1	2016-03-11	1	p1	980	1
1	2016-05-20	3	p3	330	2
1	2016-11-09	1	p1	980	3
1	2017-03-11	2	p2	870	4
1	2017-04-19	2	p2	870	5
1	2018-03-19	3	p3	330	6
1	2019-10-23	2	p2	870	7
2	2017-09-24	1	p1	980	1
2	2017-11-08	2	p2	870	2
2	2018-09-10	3	p3	330	3
2	2020-07-20	3	p3	330	4
3	2016-11-10	1	p1	980	1
3	2016-12-15	2	p2	870	2
3	2016-12-20	2	p2	870	3
3	2017-12-07	2	p2	870	4
3	2019-12-18	1	p1	980	5

Step 2: Now, to find the first product of each customer, we simply put the result generated in step 1 within a **common table expression**, here named as cte and then further use the **where** clause (rnk=1) to filter the table generated in step 1 to get the desired result.

```
-- 2nd step

with cte as(
select s.userid,s.created_date,p.product_id,p.product_name,p.price, row_number() over(partition by userid order by created_date) as rn
from sales s
inner join product p on s.product_id = p.product_id
)
select * from cte
where rn =1;
```

Following the execution of the query stated above, the resultant output is illustrated below.

userid	created_date	product_id	product_name	price	rn
1	2016-03-11	1	p1	980	1
2	2017-09-24	1	p1	980	1
3	2016-11-10	1	p1	980	1

Another method: Another method/approach which can be used to obtain the desired result is given below.

```
-- Alternative method

with cte as(
select * from sales
where userid=1
order by userid,created_date
limit 1),
cte2 as(
select * from sales
where userid=2
order by userid,created_date
limit 1),
cte3 as(
select * from sales
where userid=3
order by userid,created_date
limit 1)
select * from
cte,cte2,cte3;
```

After executing the query mentioned above, the output is provided below for reference.

	userid	created_date	product_id	userid	created_date	product_id	userid	created_date	product_id
▶	1	2016-03-11	1	2	2017-09-24	1	3	2016-11-10	1

Conclusion: The **first product ordered by Customer 1 was P1** of Rs. 980 on 11–03–2016. **The first product ordered by Customer 2 was P1** of Rs. 980 on 24–09–2017 and the **first product ordered by Customer 3 was also P1** of Rs. 980 on 10–11–2016.

Problem 4: What is the most purchased item on the menu and how many times it was purchased by all the customers?

There are two steps involved in addressing the specific problem.

Step 1: To answer the most purchased item on the menu, we have **group by** the product_id and used **ordered by** on the **count** of product_id and finally used the **limit** function on the product table.

```
-- Problem 4: What is the most purchased item on the menu and how many times it was purchased by all the customers?

-- 1st step

select product_id
from sales
group by product_id
order by count(product_id) desc
limit 1;
```

Result after the query gets executed is given below.

product_id
2

Step 2: So from step 1, we found out most purchased item on the menu was the product with product_id 2. Now to answer that how many times it was purchased by each customer, we had used the result obtained from step 1 as a **subquery** and used it with the **where** clause and **group by** on the userid which will finally bifurcate the product order which is maximum ordered.

```sql
-- 2nd Step

select userid,count(product_id) as order_count from sales
where product_id = (
select product_id
from sales
group by product_id
order by count(product_id) desc
limit 1)
group by userid
order by userid;
```

After the execution of the query mentioned previously, the resulting output is displayed below.

userid	order_count
1	3
2	1
3	3

Conclusion: Product with **product_id 2 was the most purchased item** on Zomato where **Customer 1, Customer 2 and Customer 3 have ordered the particular product 3,1,3 times respectively**.

Problem 5: Which item is popular for each customer?

To find the popular item for each customer, we have bifurcated the problem into 3 steps.

Step 1: In the first step, we group by the userid and product_id and count the product_id on the sales table.

```sql
-- Problem 5: Which item is popular for each customer?

-- 1st step

select userid,product_id,count(product_id) as cnt
from sales
group by userid,product_id
order by userid;
```

The above query will generate the following result.

userid	product_id	cnt
1	1	2
1	2	3
1	3	2
2	1	1
2	2	1
2	3	2
3	1	2
3	2	3

Step 2: The table obtained from result 1 is temporarily stored using **common table expression** named cte and then **Rank window function partitioned by** userid is further used to rank the result generated in step 1 grouped by userid.

```sql
-- 2nd step

with cte as(
select userid,product_id,count(product_id) as cnt
from sales
group by userid,product_id
order by userid)
select *, rank()over(partition by userid order by cnt desc) as rnk
from cte;
```

Subsequent to the execution of the above query, the resulting output is delineated below.

userid	product_id	cnt	rnk
1	2	3	1
1	1	2	2
1	3	2	2
2	3	2	1
2	1	1	2
2	2	1	2
3	2	3	1
3	1	2	2

Step 3: We have further used cte2 for the steps involved in step 2 and finally used the **where** clause for rnk=1 (from step 2) which generates the popular item for each customer.

```
-- 3rd step

with cte as(
select userid,product_id,count(product_id) as cnt
from sales
group by userid,product_id
order by userid),
cte2 as(
select *, rank()over(partition by userid order by cnt desc) as rnk
from cte)
select * from cte2
where rnk =1;
```

Below, you'll find the output generated upon executing the aforementioned query.

userid	product_id	cnt	rnk
1	2	3	1
2	3	2	1
3	2	3	1

Conclusion: For **Customer 1, product with product_id 2 is most popular with count of 3 times** of getting ordered.
For **Customer 2, product with product_id 3 is most popular with count of 2 times** of getting ordered while for **Customer 3, product with product_id 2 is most popular with a count of 3 times** of getting ordered.

Problem 6: Which item was first purchased by the customer once they became the member?

The specified issue is addressed through a series of four sequential steps.

Step 1: In the first step we perform **inner join** on the sales and product table and then **left join** on the goldusers_signup table and further select the particular fields along with using the **datediff function** with created_date and the gold_signup_date which will provide the number of days from the day user have ordered something to the day when user have taken the gold membership.

```
-- Problem 6: Which item was first purchased by the customer once they became the member?

-- Step 1

select s.userid,s.product_id,g.gold_signup_date,s.created_date,datediff(s.created_date,g.gold_signup_date) as dd
from sales s
inner join users u on s.userid=u.userid
left join goldusers_signup g on s.userid=g.userid
order by userid,dd;
```

The output presented below is derived from the execution of the query mentioned earlier.

userid	product_id	gold_signup_date	created_date	dd
1	1	2017-09-22	2016-03-11	-560
1	3	2017-09-22	2016-05-20	-490
1	1	2017-09-22	2016-11-09	-317
1	2	2017-09-22	2017-03-11	-195
1	2	2017-09-22	2017-04-19	-156
1	3	2017-09-22	2018-03-19	178
1	2	2017-09-22	2019-10-23	761
2	3	NULL	2020-07-20	NULL
2	1	NULL	2017-09-24	NULL
2	2	NULL	2017-11-08	NULL
2	3	NULL	2018-09-10	NULL
3	1	2017-04-21	2016-11-10	-162
3	2	2017-04-21	2016-12-15	-127
3	2	2017-04-21	2016-12-20	-122
3	2	2017-04-21	2017-12-07	230
3	1	2017-04-21	2019-12-18	971

Step 2: In the second step we have introduced the **where** clause on the datediff function, where we only select the records where dd(datediff) >0, since we have to know about the orders of the customers once they had taken the gold membership.

```
-- 2nd step

select s.userid,s.product_id,g.gold_signup_date,s.created_date,datediff(s.created_date,g.gold_signup_date) as dd
from sales s
inner join users u on s.userid=u.userid
left join goldusers_signup g on s.userid=g.userid
where datediff(s.created_date,g.gold_signup_date) >0
order by userid,dd;
```

After the execution of the query, the generated output is given below.

	userid	product_id	gold_signup_date	created_date	dd
▶	1	3	2017-09-22	2018-03-19	178
	1	2	2017-09-22	2019-10-23	761
	3	2	2017-04-21	2017-12-07	230
	3	1	2017-04-21	2019-12-18	971

Step 3: In this particular step, we had confined the table obtained from step 2 under cte which is a **common table expression** and further used the **Rank window function partitioned by** userid which will group the records based on the userid.

```
-- step 3

with cte as(
select s.userid,s.product_id,g.gold_signup_date,s.created_date,datediff(s.created_date,g.gold_signup_date) as dd
from sales s
inner join users u on s.userid=u.userid
left join goldusers_signup g on s.userid=g.userid
where datediff(s.created_date,g.gold_signup_date) >0
order by userid,dd)
select *, rank() over(partition by userid order by dd) as rnk
from cte;
```

The output displayed below is the result of executing the aforementioned query.

	userid	product_id	gold_signup_date	created_date	dd	rnk
▶	1	3	2017-09-22	2018-03-19	178	1
	1	2	2017-09-22	2019-10-23	761	2
	3	2	2017-04-21	2017-12-07	230	1
	3	1	2017-04-21	2019-12-18	971	2

Step 4: Finally, to get our desired output we had confined the records from step 2 under cte2 and then used the **where** clause for rnk(rank) =1 to get the first records of the customers once they had acquired the gold membership.

```sql
-- step 4

with cte as(
select s.userid,s.product_id,g.gold_signup_date,s.created_date,datediff(s.created_date,g.gold_signup_date) as dd
from sales s
inner join users u on s.userid=u.userid
left join goldusers_signup g on s.userid=g.userid
where datediff(s.created_date,g.gold_signup_date) >0
order by userid,dd),
cte2 as(
select *, rank() over(partition by userid order by dd) as rnk
from cte)
select * from cte2
where rnk=1;
```

After executing the query mentioned above, the output is provided below for reference.

userid	product_id	gold_signup_date	created_date	dd	rnk
1	3	2017-09-22	2018-03-19	178	1
3	2	2017-04-21	2017-12-07	230	1

Conclusion: So, **Customer 1 purchased the first product with product_id 3 on 19–03–2018** after he had taken the gold membership which is on 22–09–2017 while **Customer 3 purchased the first product with product_id 2 on 07–12–2017** after he had taken the gold membership on 21–04–2017. **Customer 2 have not taken** the gold membership.