SQL CASE STUDY

TINY SHOP SALES

Samiksha Pun

INTRODUCTION

This case study uses MySQL. I have been exposed to the following areas of SQL.

- Basic Aggregations
- Joins
- CASE WHEN statements
- CTEs
- Subquery
- Date Time Functions
- Window Functions

ABOUT THE DATASET

The dataset represents a tiny shop that sells various products to its customers.

It consists of four main tables: "customers," "orders," "products," and "order_items".

Each table contains specific information related to customers, orders, products, and the items sold.

Customer Table

customer_id	firstname	lastname	email
1	John	Doe	johndoe@gmail.com
2	Jane	Smith	janesmith@gmail.com
3	Bob	Johnson	bobjohnson@gmail.com
4	Alice	Brown	alicebrown@gmail.com
5	Charlie	Davis	charliedavis@gmail.com
6	Eva	Fisher	evafisher@gmail.com
7	George	Harris	georgeharris@gmail.com
8	Ivy	Jones	ivyjones@gmail.com
9	Kevin	Miller	kevinmiller@gmail.com
10	Lily	Nelson	lilynelson@gmail.com
11	Oliver	Patterson	oliverpatterson@gmail.com
12	Quinn	Robert	quinnrobert@gmail.com
13	Sophia	Thomas	sophiathomas@gmail.com

order_id	customer_id	order_date
1	1	2023-05-01
2	2	2023-05-02
3	3	2023-05-03
4	1	2023-05-04
5	2	2023-05-05
6	3	2023-05-06
7	4	2023-05-07
8	5	2023-05-08
9	6	2023-05-09
10	7	2023-05-10
11	8	2023-05-11
12	9	2023-05-12
13	10	2023-05-13
14	11	2023-05-14
15	12	2023-05-15
16	13	2023-05-16

Orders Table

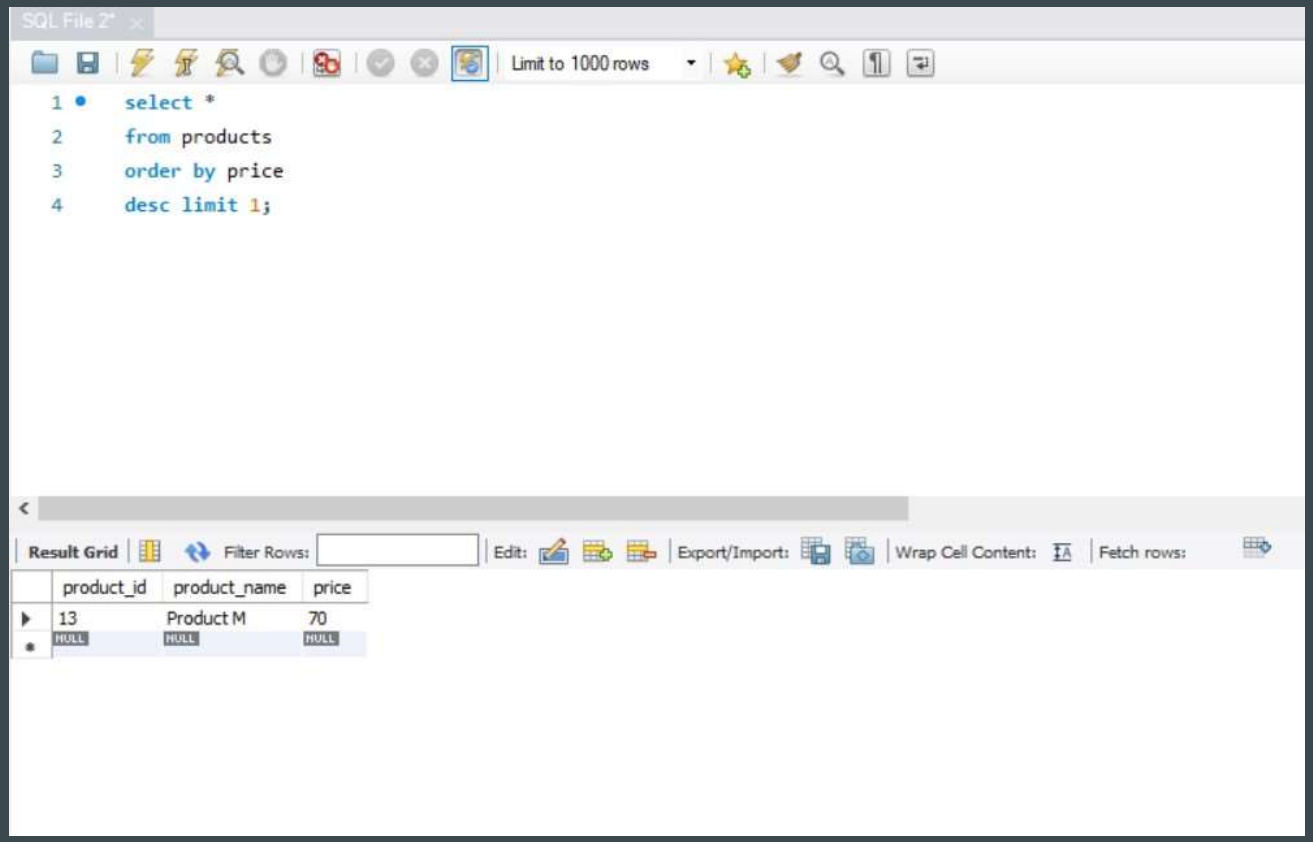
Products Table

product_id	product_name	price
1	Product A	10
2	Product B	15
3	Product C	20
4	Product D	25
5	Product E	30
6	Product F	35
7	Product G	40
8	Product H	45
9	Product I	50
10	Product J	55
11	Product K	60
12	Product L	65
13	Product M	70

order_id	product_id	quantity
1	1	2
1	2	1
2	2	1
2	3	3
3	1	1
3	3	2
4	2	4
4	3	1
5	1	1
5	3	2
6	1	1
6	2	3
7	4	1
7	5	2
8	6	3
8	7	1
9	8	2
9	9	1
10	10	3
10	11	2
11	12	1
11	13	3
12	4	2
12	5	1
13	6	3
13	7	2
14	8	1
14	9	2
15	10	3
15	11	1
16	12	2
16	13	3

Order Items Table

1. Which product has the highest price ? Only return a single row.



2. Which customer has made the most orders ?

SQL File 2* | SQL File 7* | customers | orders | products | order_items

Limit to 1000 rows

```sql
with customer_order_count as (
select c.customer_id, c.firstname, c.lastname,
count(*) as orders_count
from customers c join orders o on c.customer_id = o.customer_id
group by c.customer_id, c.firstname, c.lastname
)
, max_order_count as (
select
max(orders_count) as max_order_count
from customer_order_count
)
select a.*
from customer_order_count a join max_order_count b on a.orders_count = b.max_order_count
order by a.customer_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	firstname	lastname	orders_count
1	John	Doe	2
2	Jane	Smith	2
3	Bob	Johnson	2

3. What’s the total revenue per product ?

```sql
with quantity_sold_per_product as (
select product_id,
count(quantity) as quantity_sold
from order_items group by product_id
)
select p.product_id, p.product_name,
qp.quantity_sold, (qp.quantity_sold * p.price) as revenue
from quantity_sold_per_product qp
join products p on qp.product_id = p.product_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_id	product_name	quantity_sold	revenue
1	Product A	4	40
2	Product B	4	60
3	Product C	4	80
4	Product D	2	50
5	Product E	2	60
6	Product F	2	70
7	Product G	2	80
8	Product H	2	90
9	Product I	2	100
10	Product J	2	110
11	Product K	2	120
12	Product L	2	130
13	Product M	2	140

4. Find the day with the highest revenue ?

customers | orders | products | order_items | SQL File 9*

Limit to 1000 rows

```sql
SELECT o.order_date, SUM(p.price * oi.quantity) AS total_revenue
FROM orders o
JOIN order_items oi ON o.order_id = oi.order_id
JOIN products p ON oi.product_id = p.product_id
GROUP BY o.order_date
ORDER BY total_revenue DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

order_date	total_revenue
2023-05-16	340

5. Find the first order (by date) for each customer ?

SQL File 2* | customers | orders

Limit to 1000 rows

```sql
select customer_id, min(order_date) as First_Order
from orders
group by customer_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	First_Order
1	2023-05-01
2	2023-05-02
3	2023-05-03
4	2023-05-07
5	2023-05-08
6	2023-05-09
7	2023-05-10
8	2023-05-11
9	2023-05-12
10	2023-05-13
11	2023-05-14
12	2023-05-15
13	2023-05-16

6. Find the top 3 customers who have ordered the most distinct products ?

SQL File 2* | SQL File 7* | customers | orders | products | order_items

Limit to 1000 rows

```sql
select c.customer_id, count(distinct oi.product_id) as distinct_orders
from customers c join orders o on c.customer_id = o.customer_id
join order_items oi on oi.order_id = o.order_id
group by c.customer_id
order by distinct_orders desc limit 3;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

customer_id	distinct_orders
1	3
2	3
3	3

7. Which product has been bought the least in terms of quantity ?

SQL File 2* | customers | orders | products | order_items

Limit to 1000 rows

```sql
select p.product_name, sum(oi.quantity) as Quantity_Bought
from order_items oi inner join products p on oi.product_id=p.product_id
group by p.product_name
order by sum(oi.quantity);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_name	Quantity_Bought
Product D	3
Product E	3
Product G	3
Product H	3
Product I	3
Product K	3
Product L	3
Product A	5
Product F	6
Product J	6
Product M	6
Product C	8
Product B	9

8. What is the median order total ?

```sql
SELECT avg(order_total) AS median_order_total
FROM (
  SELECT o.order_id, SUM(p.price * oi.quantity) AS order_total
  FROM orders o
  JOIN order_items oi ON o.order_id = oi.order_id
  JOIN products p ON oi.product_id = p.product_id
  GROUP BY o.order_id
) subquery;
```

Result Grid

median_order_total
140.6250

9. For each order, determine if it was ‘Expensive’ (total over 300), ‘Affordable’ (total over 100), or ‘Cheap’ ?

```sql
select o.order_id,
sum((oi.quantity * p.price)) as total_revenue,
case
    when sum((oi.quantity * p.price)) > 300 then 'Expensive'
    when sum((oi.quantity * p.price)) > 100 and sum((oi.quantity * p.price)) < 300 then 'Affordable'
    Else 'Cheap'
 End as status
 from orders o join order_items oi on o.order_id = oi.order_id
 join products p on p.product_id = oi.product_id
 group by o.order_id
 order by total_revenue desc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	total_revenue	status
16	340	Expensive
10	285	Affordable
11	275	Affordable
15	225	Affordable
13	185	Affordable
8	145	Affordable
14	145	Affordable
9	140	Affordable
7	85	Cheap
4	80	Cheap
12	80	Cheap
2	75	Cheap
6	55	Cheap
3	50	Cheap
5	50	Cheap
1	35	Cheap

10. Find customers who have ordered the product with the highest price ?

SQL File 2* | customers | orders | products | order_items

Limit to 1000 rows

```sql
select c.customer_id, c.firstname, p.product_name, p.price
from products p join order_items oi on p.product_id = oi.product_id
join orders o on oi.order_id = o.order_id
join customers c on c.customer_id = o.customer_id
where p.price = (select max(price) from products);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	firstname	product_name	price
8	Ivy	Product M	70
13	Sophia	Product M	70

CONCLUSION

- Product M shows the highest price item.
- John Doe, Jane Smith, and Bob Jhonson are the customers who have made the maximum orders overall, and distinct orders.
- Highest revenue made in the shop was on 16th May 2023.
- The median order total with overall sales is $140.6 that helps us to find out the transactions made by the shop.
- Ivy Jones and Sophia Thomas are the customers who have ordered products with highest price.

THANK YOU !

(This dataset is obtained from the website named Data in Motion)