SQL PROJECT

RETAIL ANALYTICS

By Shubham Singh

INTRODUCTION

In today's competitive retail landscape, understanding customer behaviour, optimising inventory, and improving sales performance are essential for business success. This project focuses on a retail company facing challenges such as stagnant growth and declining customer engagement. Using SQL and data analytics, we aim to uncover insights from sales transactions, customer profiles, and product inventory data to identify high and low-performing products, segment customers for targeted marketing, and analyse customer behaviour to enhance retention strategies. Through this analysis, we seek to develop actionable recommendations to drive growth, improve customer satisfaction, and streamline inventory management.

BUSINESS PROBLEM

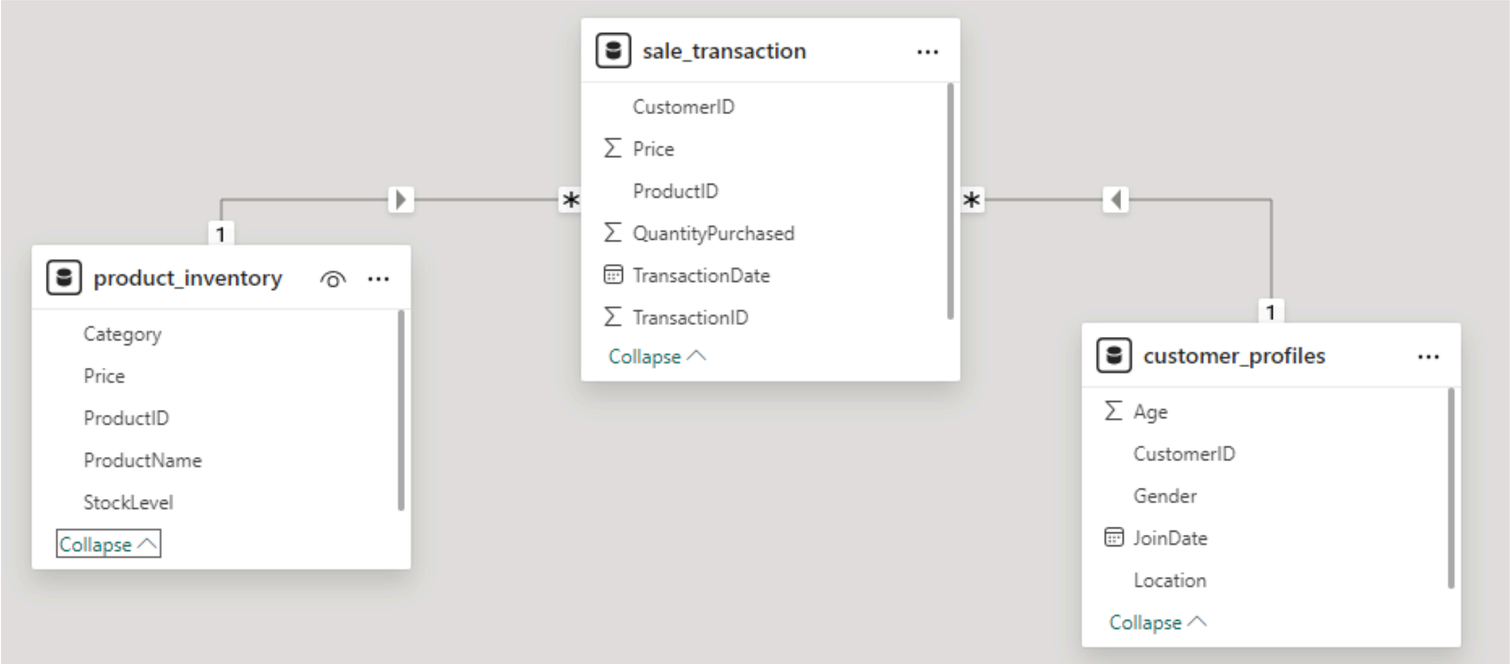
The retail company has observed stagnant growth and declining customer engagement metrics over the past quarters. Initial assessments indicate potential issues in product performance variability, ineffective customer segmentation, and lack of insights into customer purchasing behavior. The company seeks to leverage its sales transaction data, customer profiles, and product inventory information to address the following key business problems:

Product Performance Variability: Identifying which products are performing well in terms of sales and which are not. This insight is crucial for inventory management and marketing focus.

Segmentation: The company lacks a clear understanding of its customer base segmentation. Effective segmentation is essential for targeted marketing and enhancing customer satisfaction.

Customer Behaviour Analysis: Understanding patterns in customer behavior. including repeat purchases and loyalty indicators, is critical for tailoring customer engagement strategies and improving retention rates.

ENTITY RELATIONSHIP DIAGRAM



QUESTION 1

Write a query to identify the number of duplicates in "sales_transaction"

```
select ï»¿TransactionID, count(*)
from sales_transaction
group by ï»¿TransactionID
having count(*) >1;
```

Result Grid | Filter Rows:

ï»¿TransactionID	count(*)
4999	2
5000	2

QUESTION 2

Create a separate table containing the unique values

```
CREATE TABLE Sales_transaction_dupes as(
Select  distinct * from sale_transaction
);
```

Remove the the original table from the databases

```
DROP TABLE sale_transaction;
```

QUESTION 3

Replace the name of the new table with the original name

```
Alter table Sales_transaction_dupes
rename to Sales_transaction;
```

QUESTION 4

identify the discrepancies in the price of the same product in "sales_transaction" And "product_inventory" tables

```
Select ï»¿TransactionID, st.Price as TransactionPrice, p.Price as InventoryPrice
from sales_transaction as st
join product_inventory as p on st.ProductID = p.ï»¿ProductID
where st.Price <> p.Price;
```

Result Grid | Filter Rows:

ï»¿TransactionID	TransactionPrice	InventoryPrice
88	9312	93.12
236	9312	93.12
591	9312	93.12
1377	9312	93.12
1910	9312	93.12
2608	9312	93.12
2939	9312	93.12
3377	9312	93.12
3635	9312	93.12
3839	9312	93.12
3918	9312	93.12
3959	9312	93.12
3962	9312	93.12
4148	9312	93.12

QUESTION 5

update those discrepancies to match the price in both the tables

```
UPDATE Sales_transaction st
set Price = (Select Price FROM  product_inventory as pi
where pi.ï»¿ProductID= st.ProductID)
where ProductID IN
(Select ï»¿ProductID FROM product_inventory as pi1 where pi1.Price <> st.Price);
```

QUESTION 6

Write a SQL query to summarize the total sales and quantities sold per product by the company

```sql
Select ProductID, SUM(QuantityPurchased) as TotalUnitsSold,
sum(QuantityPurchased*Price) as TotalSales
from Sales_transaction
group by ProductID
order by TotalSales DESC;
```

Result Grid | Filter Rows:

ProductID	TotalUnitsSold	TotalSales
17	100	9450
87	92	7817.239999999998
179	86	7388.259999999998
96	72	7132.3200000000015
54	86	7052.8600000000015
187	82	6915.880000000003
156	76	6827.840000000002
57	78	6622.199999999999
200	69	6479.790000000001
127	68	6415.799999999999
28	69	6386.640000000001
106	63	6262.829999999999
104	72	6230.160000000001
195	87	6229.199999999999

QUESTION 7

Write a SQL query to evaluate the performance of the product categories based on the total sales which help us understand the product categories which needs to be promoted in the marketing campaigns.

```sql
Select pi.Category, sum(st.QuantityPurchased) as TotalUnitsSold,
sum(st.QuantityPurchased*pi.Price) as TotalSales
from sales_transaction as st
join product_inventory as pi on st.ProductID = pi.ï»¿ProductID
group by pi.Category
order by TotalSales DESC;
```

Result Grid | Filter Rows: | Expo

Category	TotalUnitsSold	TotalSales
Home & Kitchen	3477	217755.94000000026
Electronics	3037	177548.48000000007
Clothing	2810	162874.21000000005
Beauty & Health	3001	143824.98999999947

QUESTION 8

Write a SQL query to find the top 10 products with the highest total sales revenue from the sales transactions. This will help the company to identify the High sales products which needs to be focused to increase the revenue of the company.

```
Select ProductID, sum(QuantityPurchased * Price) as TotalRevenue
from sales_transaction
group by ProductID
order by TotalRevenue DESC
limit 10;
```

Result Grid | Filter Rows:

ProductID	TotalRevenue
17	9450
87	7817.239999999998
179	7388.259999999998
96	7132.3200000000015
54	7052.8600000000015
187	6915.880000000003
156	6827.840000000002
57	6622.199999999999
200	6479.790000000001
127	6415.799999999999

QUESTION 9

Write a SQL query to find the ten products with the least amount of units sold from the sales transactions, provided that at least one unit was sold for those products.

```
Select ProductID, SUM(Quantitypurchased) as TotalUnitsSold
from sales_transaction
group by ProductID
having SUM(QuantityPurchased) >0
order by TotalUnitsSold asc
limit 10;
```

Result Grid | Filter Rows:

ProductID	TotalUnitsSold
142	27
33	31
174	33
159	35
60	35
41	35
91	35
198	36
124	39
163	39

QUESTION 10

Write a SQL query to identify the sales trend to understand the revenue pattern of the company.

- The resulting table must have DATETRANS in date format, count the number of transaction on that particular date, total units sold and the total sales took place.
- Return the result table ordered by datetrans in descending order.

```sql
Select CAST(TransactionDate as DATE) AS DATETRANS,
COUNT(ï»¿TransactionID) AS Transaction_count,
SUM(QuantityPurchased) as TotalUnitsSold,
SUM(QuantityPurchased * Price) as TotalSales
from sales_transaction
GROUP BY cast(TransactionDate as date)
order by DATETRANS DESC;
```

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

DATETRANS	Transaction_count	TotalUnitsSold	TotalSales
2031-05-23	24	64	3569.0000000000005
2031-03-23	24	55	3468.1500000000005
2031-01-23	24	68	4089.9
2030-06-23	24	67	3908.7699999999995
2030-05-23	24	58	3528.6499999999996
2030-04-23	24	63	3451.67
2030-03-23	24	54	3249.25
2030-01-23	24	51	2614.3300000000004
2029-06-23	24	59	3471.2599999999998
2029-05-23	24	54	2840.61
2029-04-23	24	61	3908.6899999999996
2029-03-23	24	57	2859.2900000000004
2029-01-23	24	61	3885.9900000000002
2028-07-23	8	18	1158.8600000000001

QUESTION 11

Write a SQL query to understand the month on month growth rate of sales of the company which will help understand the growth trend of the company.

- **Return the result table ordering by month.**

```sql
with base_table as
(Select month(cast(TransactionDate as date)) as month,
sum(QuantityPurchased * Price) as total_sales
from sales_transaction
group by month(cast(TransactionDate as date))
)
SELECT
    month,
    Total_Sales,
    LAG(Total_Sales) OVER(ORDER BY month) AS previous_month_sales,
    ((Total_Sales - LAG(Total_Sales) OVER(ORDER BY month)) /
    LAG(Total_Sales) OVER(ORDER BY month)) * 100 AS mom_growth_percentage
FROM
    base_table
ORDER BY
    month;
```

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

month	Total_Sales	previous_month_sales	mom_growth_percentage
1	104289.17999999993	NULL	NULL
2	96690.98999999995	104289.17999999993	-7.285693491884769
3	103271.49	96690.98999999995	6.805701337839299
4	101561.09000000014	103271.49	-1.656217025628141
5	102998.83999999995	101561.09000000014	1.4156504228142972
6	102210.28	102998.83999999995	-0.7656008553105592
7	90981.75000000004	102210.28	-10.985714939827927

QUESTION 12

- Write a SQL query that segments customers based on the total quantity of products they have purchased. Also, count the number of customers in each segment which will help us target a particular segment for marketing.

```sql
with customer_segment as(
select CustomerID, SUM(QuantityPurchased) as Qty,
case
    when SUM(QuantityPurchased)<10 then "Low"
    when SUM(QuantityPurchased)<=30 then "Med"
    else "High"
end as CustomerSegment
from sales_transaction
group by CustomerID)
Select CustomerSegment, COUNT(*) from customer_segment
group by CustomerSegment;
```

Result Grid | Filter Rows:

CustomerSegment	COUNT(*)
Med	627
Low	355
High	7

THANK YOU