Case Study - Retail Analytics - I

Background

Retail companies face challenges including stagnant growth, unclear customer segmentation, and inventory inefficiencies. This case study examines how SQL-based data analysis can identify product performance trends, segment customers, and reveal behavioural patterns to inform marketing and inventory decisions.

Business Problems

- Product Performance Variability: Identify best and worst-selling products.
- Customer Segmentation: Group customers by purchasing patterns.
- Customer Behaviour Analysis: Understand repeat purchase habits and loyalty signals.

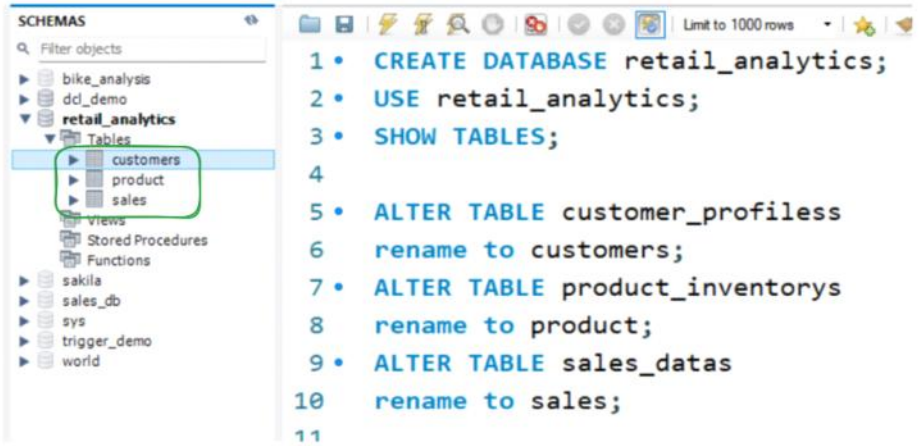
Objectives:

- *To utilize SQL queries for data cleaning and exploratory data analysis to ensure data quality and gain initial insights.*
- *To identify high and low sales products to optimize inventory and tailor marketing efforts.*
- *To segment customers based on their purchasing behavior for targeted marketing campaigns. Create Customer segments -*

Total Quantity of Products Purchased	Customer Segment
0	No Orders
1-10	Low
10-30	Mid
>30	High Value

- *To analyze customer behavior for insights on repeat purchases and loyalty, informing customer retention strategies.*

Reading the data



```
12 •  SELECT * FROM customers;
```

COUNT(*)
1000

ï»¿CustomerID	Age	Gender	Location	JoinDate
1	63	Other	East	01/01/20
2	63	Male	North	02/01/20
3	34	Other	North	03/01/20
4	19	Other		04/01/20
5	57	Male	North	05/01/20
6	22	Other	South	06/01/20
7	56	Other	East	07/01/20
8	65	Female	East	08/01/20
9	33	Male	West	09/01/20
10	34	Male	East	10/01/20
11	44	Other	North	11/01/20
12	24	Other	East	13/01/20
13	69	Male	East	14/01/20
14	25	Male	North	15/01/20
15	25	Male	North	16/01/20

Field	Type	Null	Key	Default	Extra
ï»¿CustomerID	int	YES		NULL	
Age	int	YES		NULL	
Gender	text	YES		NULL	
Location	text	YES		NULL	
JoinDate	text	YES		NULL	

```
13 •  SELECT * FROM product;
14 •  SELECT * FROM sales;
```

COUNT(*)
200

ï»¿ProductID	ProductName	Category	StockLevel	Price
1	Product_1	Clothing	22	46.11
2	Product_2	Home & Kitchen	140	81.6
3	Product_3	Home & Kitchen	473	78.72
4	Product_4	Clothing	386	22.06
5	Product_5	Beauty & Health	284	17.97
6	Product_6	Home & Kitchen	449	91.73
7	Product_7	Home & Kitchen	319	58.2
8	Product_8	Home & Kitchen	155	87.2
9	Product_9	Clothing	470	15.23
10	Product_10	Electronics	419	57.39
11	Product_11	Electronics	112	58.55
12	Product_12	Electronics	389	87.46
13	Product_13	Electronics	138	18.78
14	Product_14	Electronics	421	31.64
15	Product_15	Home & Kitchen	373	46.59

Field	Type	Null	Key	Default	Extra
ï»¿ProductID	int	YES		NULL	
ProductName	text	YES		NULL	
Category	text	YES		NULL	
StockLevel	int	YES		NULL	
Price	double	YES		NULL	

```
14 •  SELECT * FROM sales;
```

COUNT(*)
5002

ï»¿TransactionID	CustomerID	ProductID	QuantityPurchased	TransactionDate	Price
1	103	120	3	01/01/23	30.43
2	436	126	1	01/01/23	15.19
3	861	55	3	01/01/23	67.76
4	271	27	2	01/01/23	65.77
5	107	118	1	01/01/23	14.55
6	72	53	1	01/01/23	26.27
7	701	39	2	01/01/23	95.92
8	21	65	4	01/01/23	17.19
9	615	145	4	01/01/23	66
10	122	158	2	01/01/23	22.27
11	467	181	2	01/01/23	69
12	215	13	3	01/01/23	18.78
13	331	21	1	01/01/23	14.29
14	459	147	3	01/01/23	53.98
15	88	53	2	01/01/23	26.27

Field	Type	Null	Key	Default	Extra
ï»¿TransactionID	int	YES		NULL	
CustomerID	int	YES		NULL	
ProductID	int	YES		NULL	
QuantityPurchased	int	YES		NULL	
TransactionDate	text	YES		NULL	
Price	double	YES		NULL	

```
+-------------------+--------+------+-----+---------+-------+
| Field             | Type   | Null | Key | Default | Extra |
+-------------------+--------+------+-----+---------+-------+
| TransactionID     | int    | YES  |     | NULL    |       |
| CustomerID        | int    | YES  |     | NULL    |       |
| ProductID         | int    | YES  |     | NULL    |       |
| QuantityPurchased | int    | YES  |     | NULL    |       |
| TransactionDate   | text   | YES  |     | NULL    |       |
| Price             | double | YES  |     | NULL    |       |
+-------------------+--------+------+-----+---------+-------+
```

```
-- ï»¿CustomerID -> CustomerID
-- ï»¿ProductID -> ProductID
-- ï»¿TransactionID -> TransactionID

ALTER TABLE Customers
CHANGE ï»¿CustomerID CustomerID INT;
-- OR
ALTER TABLE Customers
RENAME COLUMN ï»¿CustomerID TO CustomerID;

ALTER TABLE Product
CHANGE ï»¿ProductID ProductID INT;
-- OR
ALTER TABLE Product
RENAME COLUMN ï»¿ProductID TO ProductID;


ALTER TABLE sales
CHANGE ï»¿TransactionID TransactionID INT;
-- OR
ALTER TABLE sales
RENAME COLUMN ï»¿TransactionID TO TransactionID;
```

Challenge1:

create a separate table containing the unique values and remove the original table from the databases and replace the name of the new table with the original name.

Hint:
Use the "Sales_transaction" table.
There will be two resulting tables in the output. First, the table where the count of duplicates will be identified and in the second table we can check if the duplicates were removed or not by selecting the whole table.

```
+---------------+----------+
| TransactionID | count(*) |
+---------------+----------+
| Transaction 1 |    NUM   |
| Transaction 2 |    NUM   |
+---------------+----------+

+--------------+-------------+-----------+-------------------+-----------------+-------+
| TransactionID | CustomerID | ProductID | QuantityPurchased | TransactionDate | Price |
+--------------+-------------+-----------+-------------------+-----------------+-------+
| Transaction 1 | Customer 1 | Product 1 |        NUM        |       TEXT      |DECINUM|
| Transaction 2 | Customer 2 | Product 2 |        NUM        |       TEXT      |DECINUM|
| Transaction 3 | Customer 3 | Product 3 |        NUM        |       TEXT      |DECINUM|
| Transaction 4 | Customer 4 | Product 4 |        NUM        |       TEXT      |DECINUM|
+-------------+----------+----------+-------------------+-----------------+-------+
```

```
46 • SELECT
47       TransactionID,
48       COUNT(*)
49   FROM sales
50   GROUP BY TransactionID
51   HAVING COUNT(*) > 1;
```

Result Grid | Filter Rows: | Export:

TransactionID	COUNT(*)
4999	2
5000	2

```
56 • SELECT
57       TransactionID,
58       COUNT(*)
59   FROM sales_unique
60   GROUP BY TransactionID
61   HAVING COUNT(*) > 1;
```

Result Grid | Filter Rows: | Export:

TransactionID	COUNT(*)

```sql
-- Challenge 1
SELECT
    TransactionID,
    COUNT(*)
FROM sales
GROUP BY TransactionID
HAVING COUNT(*) > 1;

CREATE TABLE sales_unique AS
SELECT DISTINCT * FROM sales;

DROP TABLE sales;
ALTER TABLE sales_unique RENAME TO sales;
```

Challenge2:

Problem statement Send feedback

Write a query to identify the discrepancies in the price of the same product in "sales_transaction" and "product_inventory" tables. Also, update those discrepancies to match the price in both the tables.

Hint:

- *Use the "sales_transaction" and the "product_inventory" tables.*
- *There will be two resulting tables in the output. First, the table where the discrepancies will be identified and in the second table we can check if the discrepancies were updated or not.*

Output format:

```
+---------------+------------------+----------------+
| TransactionID | TransactionPrice | InventoryPrice |
+---------------+------------------+----------------+
| Transaction 1 |        NUM       |        DECINUM |
| Transaction 2 |        NUM       |        DECINUM |
| Transaction 3 |        NUM       |        DECINUM |
| Transaction 4 |        NUM       |        DECINUM |
+---------------+------------------+----------------+

+---------------+------------+-----------+-------------------+-----------------+-------+
| TransactionID | CustomerID | ProductID | QuantityPurchased | TransactionDate | Price |
+---------------+------------+-----------+-------------------+-----------------+-------+
| Transaction 1| Customer 1 | Product 1 |        NUM        |       TEXT      |DECINUM|
| Transaction 2| Customer 2 | Product 2 |        NUM        |       TEXT      |DECINUM|
| Transaction 3| Customer 3 | Product 3 |        NUM        |       TEXT      |DECINUM|
| Transaction 4| Customer 4 | Product 4 |        NUM        |       TEXT      |DECINUM|
| Transaction 5| Customer 5 | Product 5 |        NUM        |       TEXT      |DECINUM|
+---------------+------------+-----------+-------------------+-----------------+-------+
```

Note: *The NUM in the output format denotes a numerical values, DECINUM denotes a decimal values, Transaction 1 denotes to transaction number 1, Customer 1 also means the customer with unique ID 1 and TransactionDate is in text format thus, cannot be considered as date.*

TransactionID	CustomerID	ProductID	QuantityPurchased	TransactionDate	Price
1	103	120	3	01/01/23	30.43
2	436	126	1	01/01/23	15.19
3	861	55	3	01/01/23	67.76
4	271	27	2	01/01/23	65.77
5	107	118	1	01/01/23	14.55
6	72	53	1	01/01/23	26.27
7	701	39	2	01/01/23	95.92
8	21	65	4	01/01/23	17.19
9	615	145	4	01/01/23	66
10	122	158	2	01/01/23	22.27
11	467	181	2	01/01/23	69
12	215	13	3	01/01/23	18.78
13	331	21	1	01/01/23	14.29
14	459	147	3	01/01/23	53.98
15	88	53	2	01/01/23	26.27

ProductID	ProductName	Category	StockLevel	Price
1	Product_1	Clothing	22	46.11
2	Product_2	Home & Kitchen	140	81.6
3	Product_3	Home & Kitchen	473	78.72
4	Product_4	Clothing	386	22.06
5	Product_5	Beauty & Health	284	17.97
6	Product_6	Home & Kitchen	449	91.73
7	Product_7	Home & Kitchen	319	58.2
8	Product_8	Home & Kitchen	155	87.2
9	Product_9	Clothing	470	15.23
10	Product_10	Electronics	419	57.39
11	Product_11	Electronics	112	58.55
12	Product_12	Electronics	389	87.46
13	Product_13	Electronics	138	18.78
14	Product_14	Electronics	421	31.64
15	Product_15	Home & Kitchen	373	46.59

ProductID	TransactionID	TransactionPrice	InventoryPrice
51	88	9312	93.12
51	236	9312	93.12
51	51	9312	93.12
51	1377	9312	93.12
51	1910	9312	93.12
51	2608	9312	93.12
51	2939	9312	93.12
51	3377	9312	93.12
51	3635	9312	93.12
51	3839	9312	93.12
51	3918	9312	93.12
51	3959	9312	93.12
51	3962	9312	93.12
51	4148	9312	93.12
51	4158	9312	93.12
51	4221	9312	93.12
51	4408	9312	93.12
51	4532	9312	93.12
51	4754	9312	93.12
51	4968	9312	93.12

```
- Discrepancies in the price
SELECT
    p.ProductID,
    TransactionID,
    s.Price AS TransactionPrice,
    p.Price AS InventoryPrice
FROM sales s
JOIN product p
ON s.ProductID = p.ProductID
WHERE p.price <> s.price;
```

```
-- UPDATE Price of ProductID - 51
-- UPDATE sales
-- SET Price = 93.12
-- WHERE ProductID = 51; IN (51,2,343,54,12,42,644,334)

SET SQL_SAFE_UPDATES = 0;
UPDATE sales s
SET Price = (
    SELECT p.price FROM product p
    WHERE s.ProductID = p.ProductID
)
WHERE s.ProductID IN (
    SELECT ProductID FROM product p
    WHERE p.price <> s.price
);
```

```
64 •  SELECT
65        p.ProductID,
66        TransactionID,
67        s.Price AS TransactionPrice,
68        p.Price AS InventoryPrice
69    FROM sales s
70    JOIN product p
71    ON s.ProductID = p.ProductID
72    WHERE p.price <> s.price;
```

Result Grid | Filter Rows: | Export: | Wrap

ProductID	TransactionID	TransactionPrice	InventoryPrice

Challenge3:

Fixing Null Values

Easy • Score 0/40 • Average time to solve is 10m

Problem statement Send feedback

Write a SQL query to identify the null values in the dataset and replace those by "Unknown".

Hint:

- *Use the customer_profiles table.*
- *Identify the columns which contains null values and count the number of cells containing null values. Update those values with "unknown" and showcase the changes that the query has created.*

Output format:

```
+----------+
| count(*) |
+----------+
|  NUM     |
+----------+
+------------+------+--------+----------+----------+
| CustomerID | Age  | Gender | Location | JoinDate |
+------------+------+--------+----------+----------+
| Customer 1 |  NUM | Other  | East     | DD/MM/YY |
| Customer 2 |  NUM | Male   | North    | DD/MM/YY |
| Customer 3 |  NUM | Other  | North    | DD/MM/YY |
| Customer 4 |  NUM | Other  | Unknown  | DD/MM/YY |
| Customer 5 |  NUM | Male   | North    | DD/MM/YY |
+------------+------+--------+----------+----------+
```

Note: *NUM in the output format denotes a numerical value and joinDate is in Date format (DD/MM/YY)*

```
-- Challenge 3
SELECT * FROM Customers;
SELECT * FROM Customers WHERE Location LIKE "";
SELECT COUNT(*) FROM Customers WHERE Location LIKE ""; 13
-- SELECT COUNT(*) FROM Customers WHERE Location IS NULL;

UPDATE Customers
SET Location = "Unknown"
WHERE Location LIKE "";

UPDATE Customers
SET Location = "Unknown"
WHERE Location IS NULL;
```

CustomerID	Age	Gender	Location	JoinDate
1	63	Other	East	01/01/20
2	63	Male	North	02/01/20
3	34	Other	North	03/01/20
4	19	Other	Unknown	04/01/20
5	57	Male	North	05/01/20
6	22	Other	South	06/01/20
7	56	Other	East	07/01/20
8	65	Female	East	08/01/20
9	33	Male	West	09/01/20
10	34	Male	East	10/01/20
11	44	Other	North	11/01/20
12	24	Other	East	13/01/20
13	69	Male	East	14/01/20
14	25	Male	North	15/01/20
15	25	Male	North	16/01/20
16	40	Other	East	17/01/20