**Retails Analytics Business Case Study**

**Contents**

[**Datasets Intro 2**](#_Toc201365838)

[**Approach for reading dataset: 3**](#_Toc201365839)

[**Business Problem: 3**](#_Toc201365840)

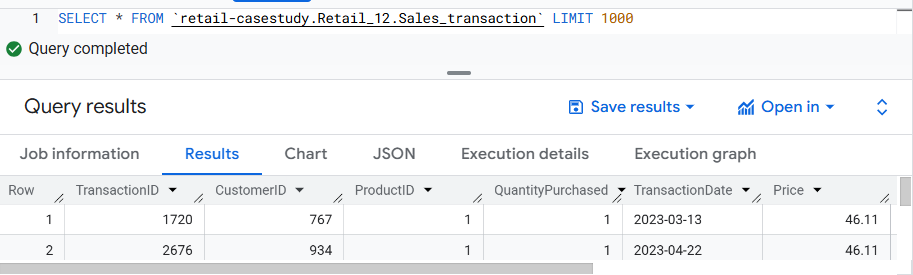
[**Approach to solve problem 4**](#_Toc201365841)

[**Exploratory Data Analysis (EDA): 13**](#_Toc201365842)

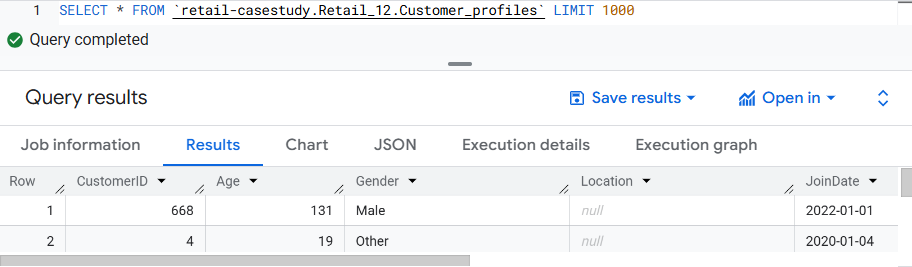
[**Recommandation to the problem: 45**](#_Toc201365843)

**Datasets Intro:**

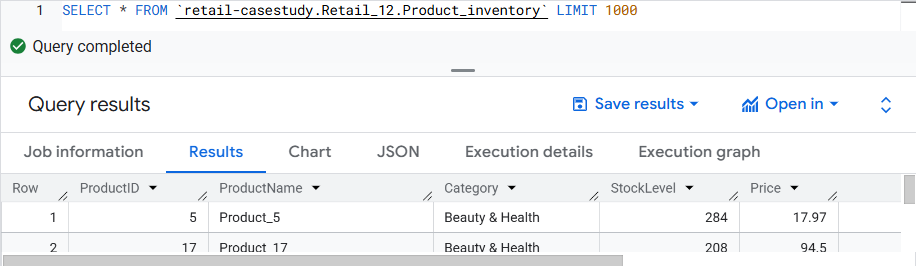
* **Sales Transactions Dataset**: Records of sales transactions, including transaction ID, customer ID, product ID, quantity purchased, transaction date, and price.



* **Customer Profiles Dataset**: Information on customers, including customer ID, age, gender, location, and join date.



* **Product Inventory Dataset**: Data on product inventory, including product ID, product name, category, stock level, and price.



# **Approach for reading dataset:**

1. Login to Bigquery sanbox application of free account (To complete this case study using bigquery platform)
2. Created Project pickers or project folder on bigquery sanbox ( retail-casestudy)
3. Created the database name : Retail\_12
4. Loaded all three files by creating tables with default schema check option (It automatically select datatype according to the dataset)

# **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 a lack of insights into customer purchasing behavior.

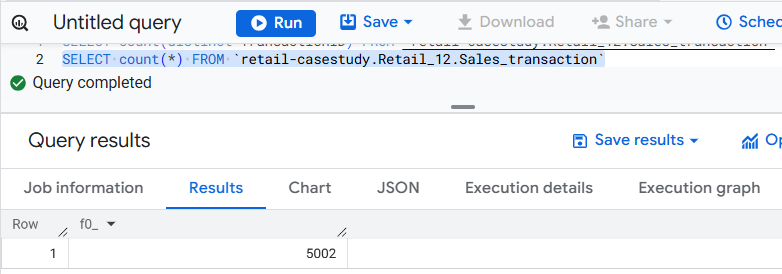
The company seeks to leverage its sales transaction data, customer profiles, and product inventory information to address the following key business problems:

* **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.
* **Customer Segmentation**: The company lacks a clear understanding of its customer base segmentation. Effective segmentation is essential for targeted marketing and enhancing customer satisfaction.
* **Customer Behavior Analysis**: Understanding patterns in customer behavior, including repeat purchases and loyalty indicators, is critical for tailoring customer engagement strategies and improving retention rates.

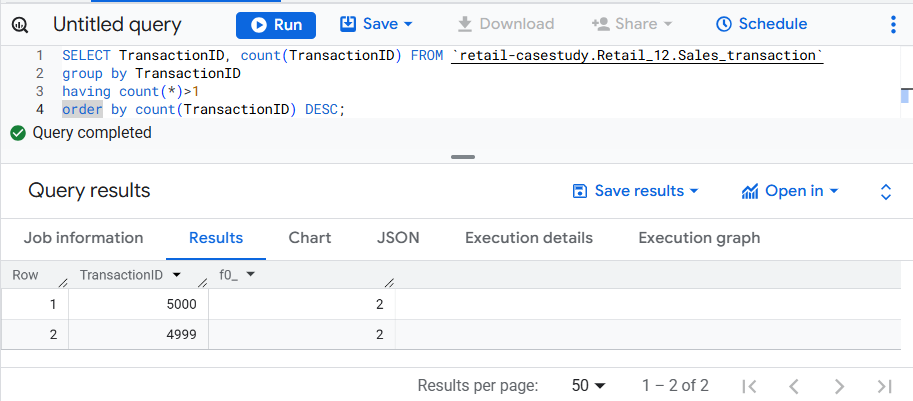
# **Approach to solve problem**

Data Cleaning:

Sales\_transaction table have 5002 total number of rows



This query represent the duplicate number of transaction with number of count in sales\_transaction table… It’s represent that table have two duplicate entry.



Query represent the unique records in sales\_transaction table

SELECT \* FROM `retail-casestudy.Retail\_12.Sales\_transaction` where TransactionID not in

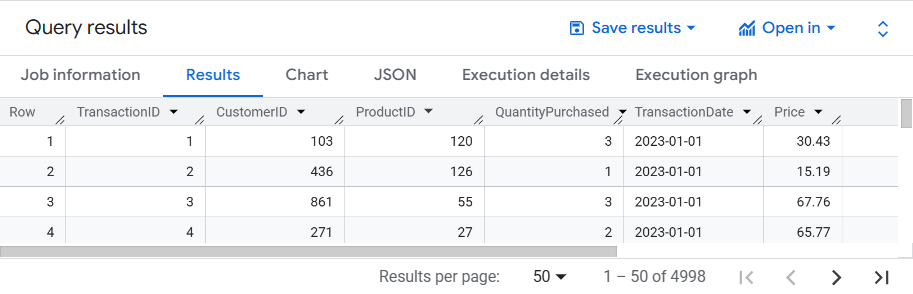
(SELECT TransactionID FROM `retail-casestudy.Retail\_12.Sales\_transaction`

group by TransactionID

having count(\*)>1

order by count(TransactionID) DESC)

order by TransactionID ASC, QuantityPurchased DESC;



This represent how many the duplicate number of records present in sales\_transaction

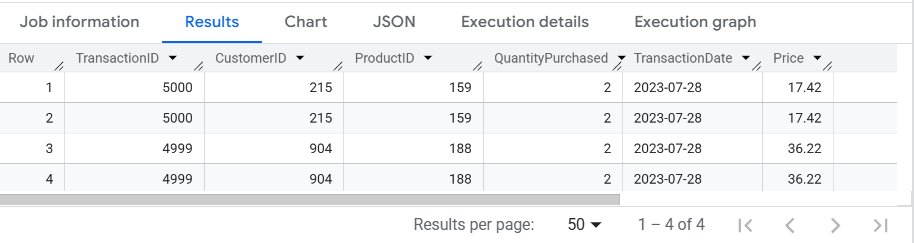
SELECT \* FROM `retail-casestudy.Retail\_12.Sales\_transaction` where TransactionID in

(SELECT TransactionID FROM `retail-casestudy.Retail\_12.Sales\_transaction`

group by TransactionID

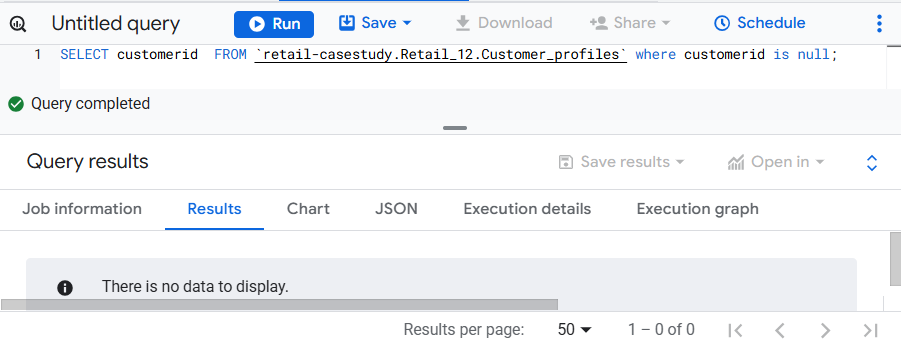
having count(\*)>1

order by count(TransactionID) DESC);



Checking any customerid column have null values.





Checking any duplicate customerid are present in customer\_profiles table or not

SELECT customerid, count(\*)  FROM `retail-casestudy.Retail\_12.Customer\_profiles`

group by customerid

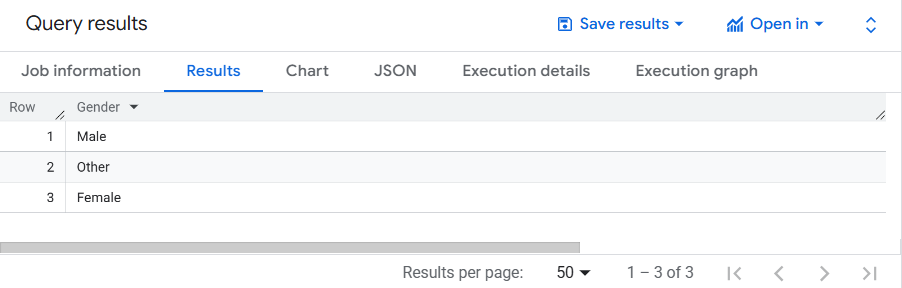
having count(\*)>1;

Checking any age column value are null

SELECT Age  FROM `retail-casestudy.Retail\_12.Customer\_profiles` where Age is null;

Number of distinct gender present in customer profiles table

SELECT distinct Gender FROM `retail-casestudy.Retail\_12.Customer\_profiles`;



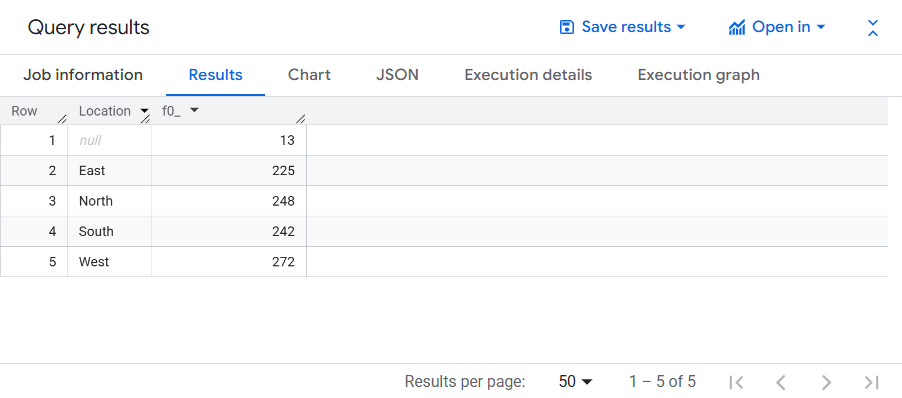
JoinDate column having any null entry or values

SELECT JoinDate FROM `retail-casestudy.Retail\_12.Customer\_profiles` where JoinDate is null;

Calculating the total number of customer present in particular or each location

SELECT Location, count(\*) FROM `retail-casestudy.Retail\_12.Customer\_profiles`

group by Location;

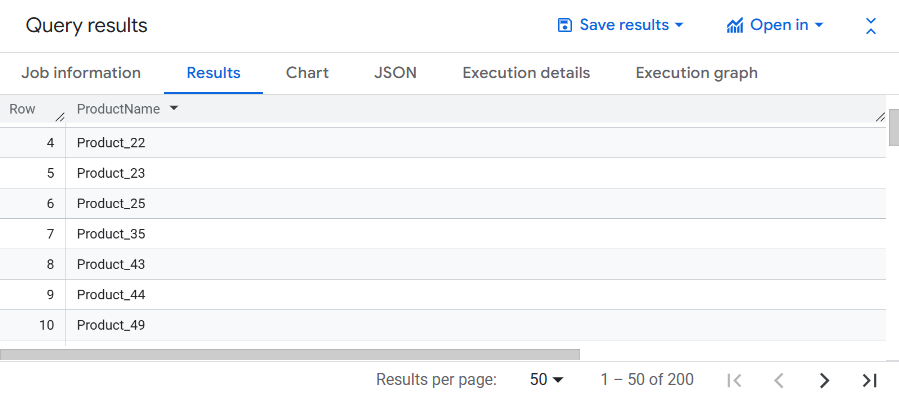


Just doing the sanity checks like all columns having proper values or null

SELECT productid FROM `retail-casestudy.Retail\_12.Product\_inventory` where productid is null;

Finding distinct number of produtname from profuct inventory table

SELECT distinct ProductName FROM `retail-casestudy.Retail\_12.Product\_inventory`;

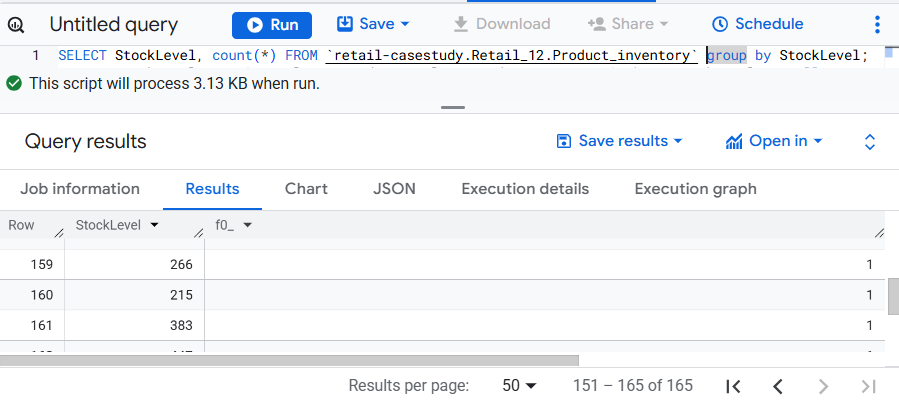


Total number of category with how many number of count’s are present in product inventory table

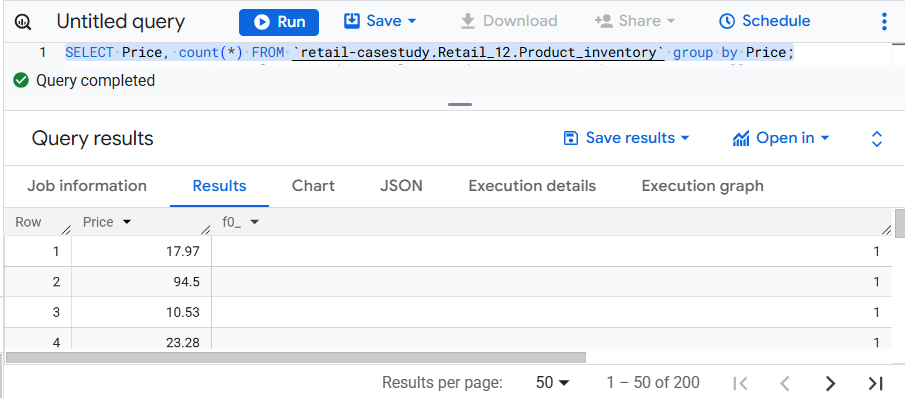
SELECT Category, count(\*) FROM `retail-casestudy.Retail\_12.Product\_inventory` group by Category;



Calculating the number of stocklevel with total number of records for each stocklevel



Calculating the number of price with total number of records for each price



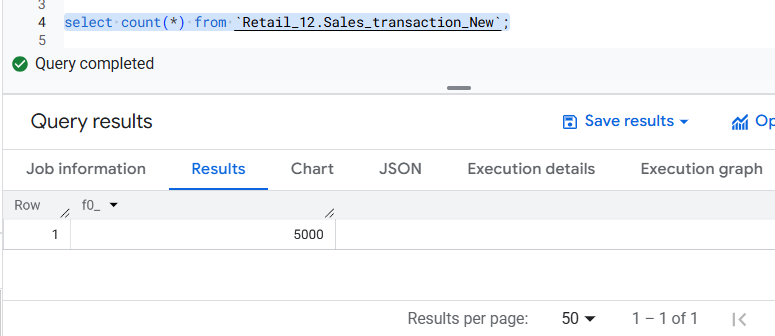
Creating the new table as **sales\_transaction\_new** for removal of duplicate records

CREATE TABLE `Retail\_12.Sales\_transaction\_New` AS select \* from (SELECT \*, row\_number() over(partition by TransactionID order by TransactionID) as cnt FROM `retail-casestudy.Retail\_12.Sales\_transaction`) as temp where cnt=1;

OR

CREATE TABLE `Retail\_12.Sales\_transaction\_New` AS

SELECT distinct \* FROM `retail-casestudy.Retail\_12.Sales\_transaction`;



Note: Using Bigquery of free account only permits the DDL and DQL statement. No update operation are allowed in free account. To use update operation or command you have to pay the money.

After this all the query are preformed on MySQL workbench :

Create database retail\_analysis\_case\_study if does not exists:

CREATE DATABASE if NOT EXISTS retail\_analysis\_case\_study;

To check the all present databases in you mysql workbench:

SHOW DATABASES;

To use the particular database:

USE retail\_analysis\_case\_study;

Table creation script for all three tables:

CREATE TABLE CUSTOMER\_PROFILES (

CustomerID INT NOT NULL,

Age INT NOT NULL,

Gender VARCHAR(10),

Location VARCHAR(255),

JoinDate DATE NOT NULL

);

CREATE TABLE Sales\_transaction (

TransactionID INT NOT NULL,

CustomerID INT NOT NULL,

ProductID INT NOT NULL,

QuantityPurchased INT NOT NULL,

TransactionDate DATE NOT NULL,

Price Decimal(38,5)

);

CREATE TABLE Product\_inventory (

ProductID INT NOT NULL,

ProductName VARCHAR(100) NOT NULL,

Category VARCHAR(50) NOT NULL,

StockLevel INT NOT NULL,

Price Decimal(38,5)

);

Finding the transaction or productid of price which does not match with the product\_inventory table price because the product\_inventory table is represent the details about the product

select \* from sales\_transaction\_new

where productid in (select productid from product\_inventory)

and price not in (select price from product\_inventory);

With out enabling the SAFE mode we can’t perform the update operation in mysql

SET SQL\_SAFE\_UPDATES = 0;

Updating the incorrect price of product in sales\_transaction table from product\_inventory table

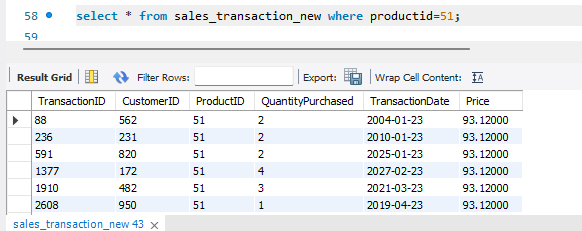
update sales\_transaction\_new st

JOIN product\_inventory as pi

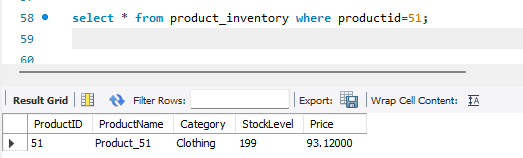
ON st.productid = pi.productid

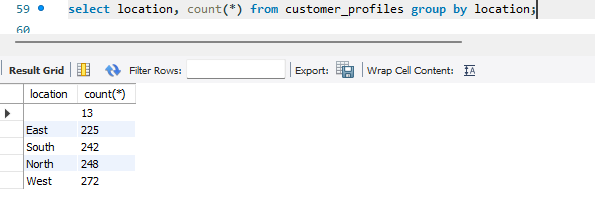
SET st.price = pi.price

where st.price <> pi.price;

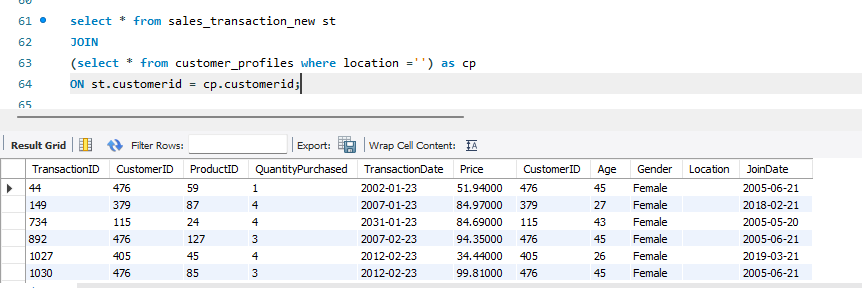


Product\_inventory table of productid=51 having the correct value and this value are update to sales\_transaction table of price





69 rows are found in output of join query. It’s present that 69 records having blank location.



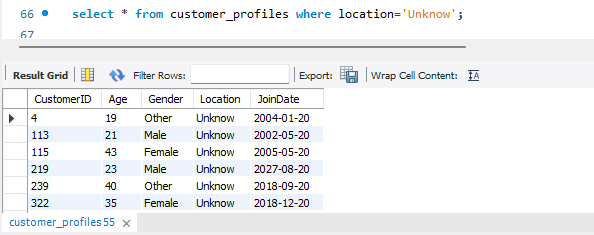
Using the update query location blank update to Unkow .

Reson for updating ‘Uknow’ value :

1. Because total number of blank location in customer profiles table having 13 records if I remove this record then it’s entry present to sales transaction table will be automatically went out.
2. Imputing with mode is also the option but I refer to the know the how many records are unknow.

This is update query to update the blank location to ‘Unknow’ value

update customer\_profiles set location='Unknow' where location='';

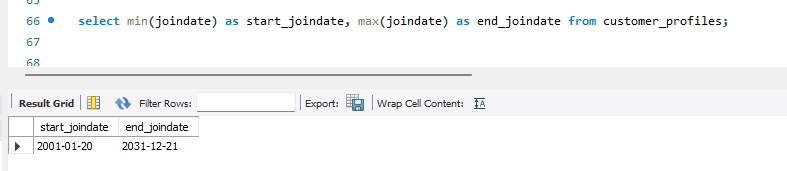


# **Exploratory Data Analysis (EDA):**

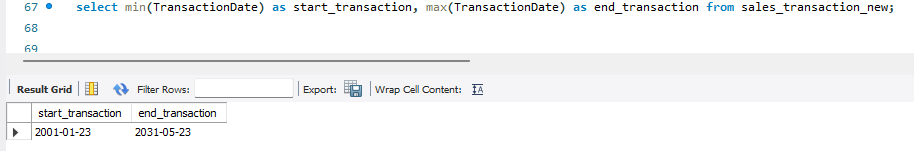
Perform basic product performance overview, customer purchase frequency analysis, and product categories performance evaluation.

**Product performance overview:**

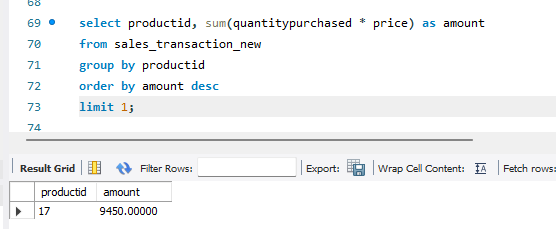
Mostly the customer joined or login the application to purchase the product from 2001 till 2031. Customer profile table contains data from 2001 till 2031



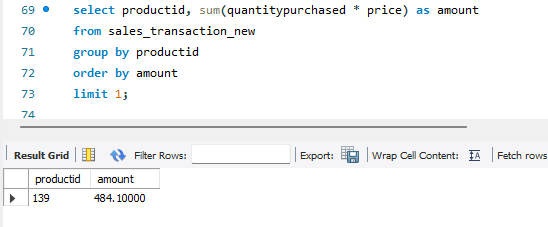
Transaction started from 2001 and last or till date transaction occurred 2031 year

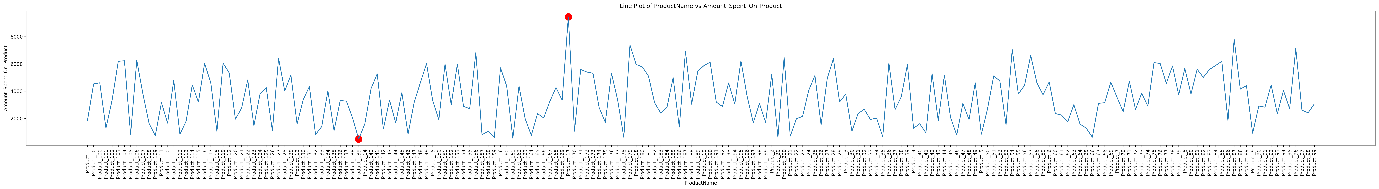


Product 17 are brought more and this product have the highest sales\_revenue

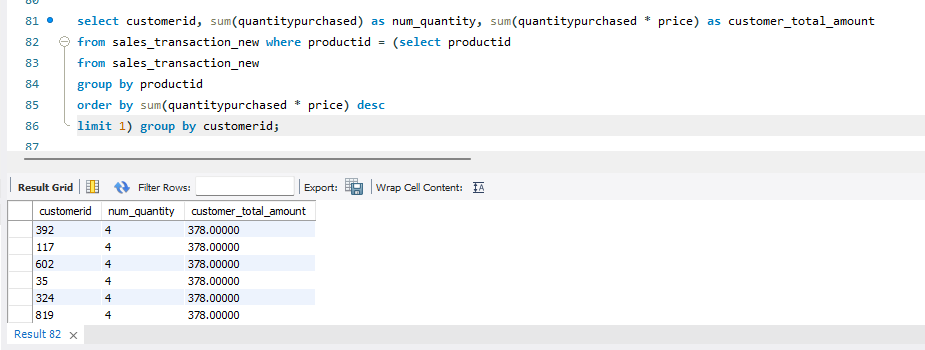


Productid – 139 having the lowest revenue or amount from all the product

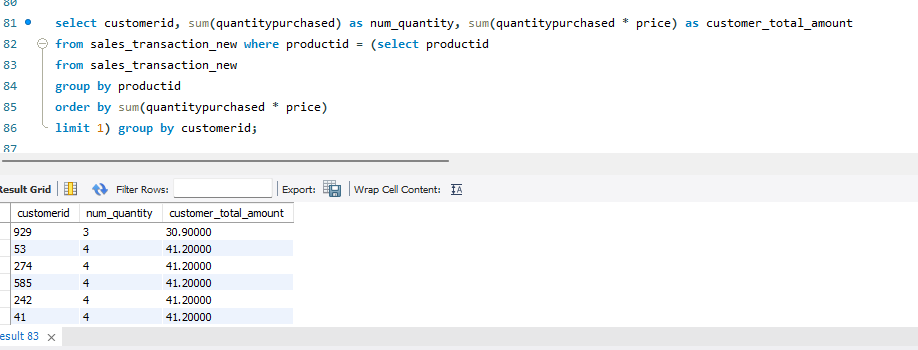




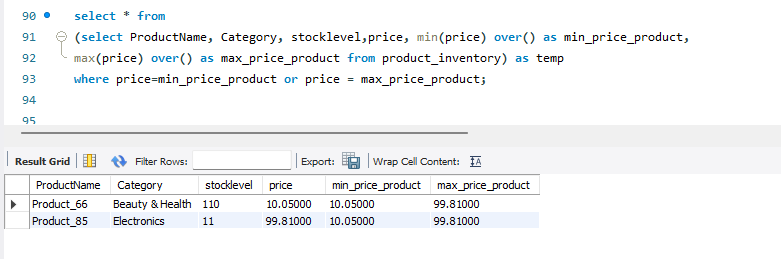
Product -17 was purchase by 39 customer



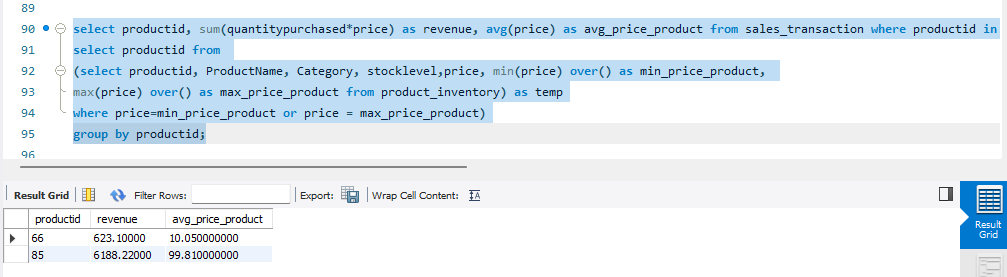
Product – 139 was purchase by 18 customer and cost of the product is 10.30



From the product inventory table finding the lowest and highest price of product with displaying the all details of product



Total revenue generated through the lowest or highest price product from sales\_transaction table and found that the product price which are highest or lowest in product table they are not highest or lowest in sales transaction table



WITH CTE AS

(select productid as max\_product\_sale\_id, round(sum(quantitypurchased \* price)) as max\_prod\_total\_revenue

from sales\_transaction\_new

group by productid

order by max\_prod\_total\_revenue desc

limit 1),

min\_prod\_price as

(select productid as min\_product\_sale\_id, round(sum(quantitypurchased \* price)) as min\_prod\_total\_revenue

from sales\_transaction\_new

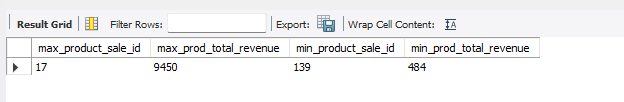
group by productid

order by min\_prod\_total\_revenue

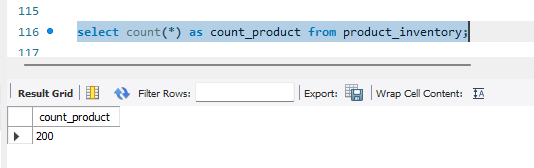
limit 1)

select max\_product\_sale\_id, max\_prod\_total\_revenue, min\_product\_sale\_id, min\_prod\_total\_revenue from CTE, min\_prod\_price;

This query represent the highest and lowest revenue generated by product from sale transaction new table



Total number of product or records present in product inventory table



WITH CTE AS

(select productid, concat(Year(TransactionDate),'-', Month(TransactionDate)) as Month\_Year\_Transaction,

SUM(QuantityPurchased \* Price) as total\_revenue, avg(price) as avg\_price

from sales\_transaction\_new

group by productid, Month\_Year\_Transaction),

product\_in as

(select productid, concat(productName,'\_',category) as productnamecategory from product\_inventory),

product\_summary as

(select pi.productnamecategory, cte.Month\_Year\_Transaction, cte.total\_revenue, cte.avg\_price from cte

JOIN product\_in as pi

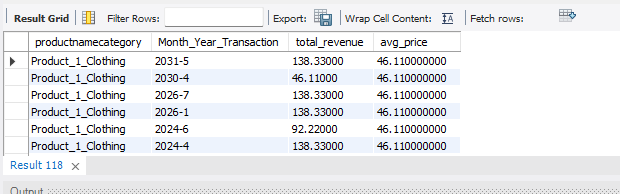
ON cte.productid = pi.productid

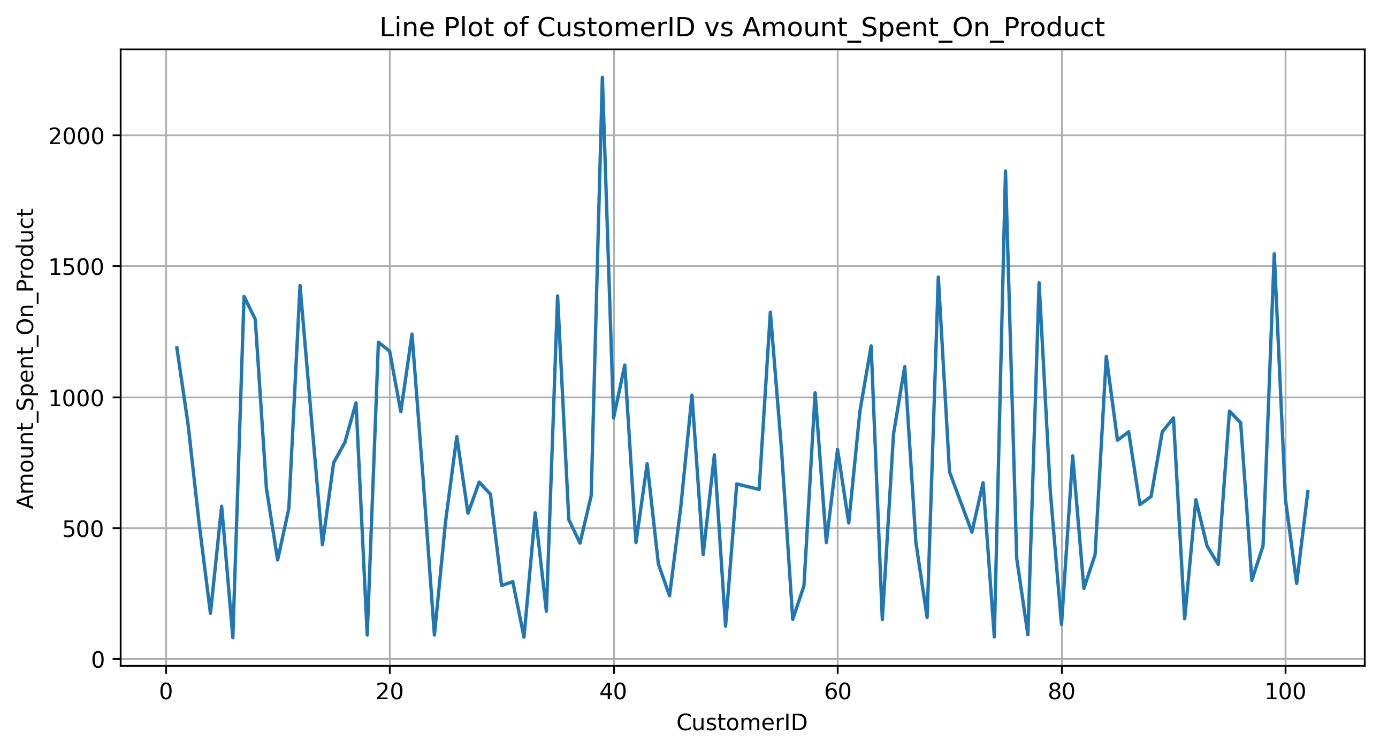
order by pi.productnamecategory asc, cte.Month\_Year\_Transaction desc)

select \* from product\_summary;

Calculating the category wise product item with there month and year wise total revenue as well as average price of product.

This represent the each product in each month of year generating how many revenue for retail company.





WITH CTE AS

(select productid, concat(Year(TransactionDate),'-', Month(TransactionDate)) as Month\_Year\_Transaction,

SUM(QuantityPurchased \* Price) as total\_revenue, avg(price) as avg\_price

from sales\_transaction\_new

group by productid, Month\_Year\_Transaction),

product\_in as

(select productid, concat(productName,'\_',category) as productnamecategory from product\_inventory),

product\_summary as

(select pi.productnamecategory, cte.Month\_Year\_Transaction, cte.total\_revenue, cte.avg\_price from cte

JOIN product\_in as pi

ON cte.productid = pi.productid

order by pi.productnamecategory asc, cte.Month\_Year\_Transaction desc)

select Month\_Year\_Transaction,

sum(CASE WHEN productnamecategory like '%1%' THEN total\_revenue END) AS product\_1,

sum(CASE WHEN productnamecategory like '%2%' THEN total\_revenue END) AS product\_2,

sum(CASE WHEN productnamecategory like '%3%' THEN total\_revenue END) AS product\_3,

sum(CASE WHEN productnamecategory like '%4%' THEN total\_revenue END) AS product\_4,

sum(CASE WHEN productnamecategory like '%5%' THEN total\_revenue END) AS product\_5,

sum(CASE WHEN productnamecategory like '%6%' THEN total\_revenue END) AS product\_6,

sum(CASE WHEN productnamecategory like '%7%' THEN total\_revenue END) AS product\_7,

sum(CASE WHEN productnamecategory like '%8%' THEN total\_revenue END) AS product\_8,

sum(CASE WHEN productnamecategory like '%9%' THEN total\_revenue END) AS product\_9,

sum(CASE WHEN productnamecategory like '%10%' THEN total\_revenue END) AS product\_10,

sum(CASE WHEN productnamecategory like '%11%' THEN total\_revenue END) AS product\_11,

sum(CASE WHEN productnamecategory like '%12%' THEN total\_revenue END) AS product\_12,

sum(CASE WHEN productnamecategory like '%13%' THEN total\_revenue END) AS product\_13,

sum(CASE WHEN productnamecategory like '%14%' THEN total\_revenue END) AS product\_14,

sum(CASE WHEN productnamecategory like '%15%' THEN total\_revenue END) AS product\_15,

sum(CASE WHEN productnamecategory like '%16%' THEN total\_revenue END) AS product\_16,

sum(CASE WHEN productnamecategory like '%17%' THEN total\_revenue END) AS product\_17,

sum(CASE WHEN productnamecategory like '%18%' THEN total\_revenue END) AS product\_18,

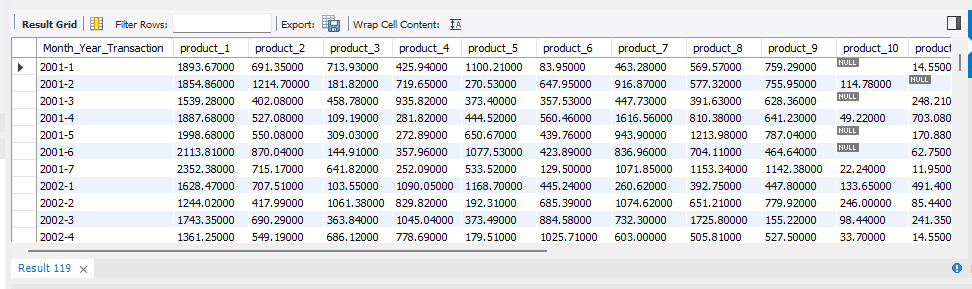
sum(CASE WHEN productnamecategory like '%19%' THEN total\_revenue END) AS product\_19,

sum(CASE WHEN productnamecategory like '%20%' THEN total\_revenue END) AS product\_20

from product\_summary

group by Month\_Year\_Transaction;

It’s just a tabular format of revenue generated by each product in each month of years.



WITH CTE AS

(select productid, Month(TransactionDate) as Month\_Transaction,

SUM(QuantityPurchased \* Price) as total\_revenue, avg(price) as avg\_price

from sales\_transaction\_new

group by productid, Month\_Transaction),

product\_in as

(select productid, concat(productName,'\_',category) as productnamecategory from product\_inventory),

product\_summary as

(select pi.productnamecategory, cte.Month\_Transaction, cte.total\_revenue, cte.avg\_price from cte

JOIN product\_in as pi

ON cte.productid = pi.productid

order by pi.productnamecategory asc, cte.Month\_Transaction desc)

select productnamecategory,

SUM(CASE WHEN Month\_Transaction=1 THEN total\_revenue ELSE 0 END) AS 'JAN',

SUM(CASE WHEN Month\_Transaction=2 THEN total\_revenue ELSE 0 END) AS 'FEB',

SUM(CASE WHEN Month\_Transaction=3 THEN total\_revenue ELSE 0 END) AS 'MAR',

SUM(CASE WHEN Month\_Transaction=4 THEN total\_revenue ELSE 0 END) AS 'APR',

SUM(CASE WHEN Month\_Transaction=5 THEN total\_revenue ELSE 0 END) AS 'MAY',

SUM(CASE WHEN Month\_Transaction=6 THEN total\_revenue ELSE 0 END) AS 'JUN',

SUM(CASE WHEN Month\_Transaction=7 THEN total\_revenue ELSE 0 END) AS 'JUL',

SUM(CASE WHEN Month\_Transaction=8 THEN total\_revenue ELSE 0 END) AS 'AUG',

SUM(CASE WHEN Month\_Transaction=9 THEN total\_revenue ELSE 0 END) AS 'SEP',

SUM(CASE WHEN Month\_Transaction=10 THEN total\_revenue ELSE 0 END) AS 'OCT',

SUM(CASE WHEN Month\_Transaction=11 THEN total\_revenue ELSE 0 END) AS 'NOV',

SUM(CASE WHEN Month\_Transaction=12 THEN total\_revenue ELSE 0 END) AS 'DEC',

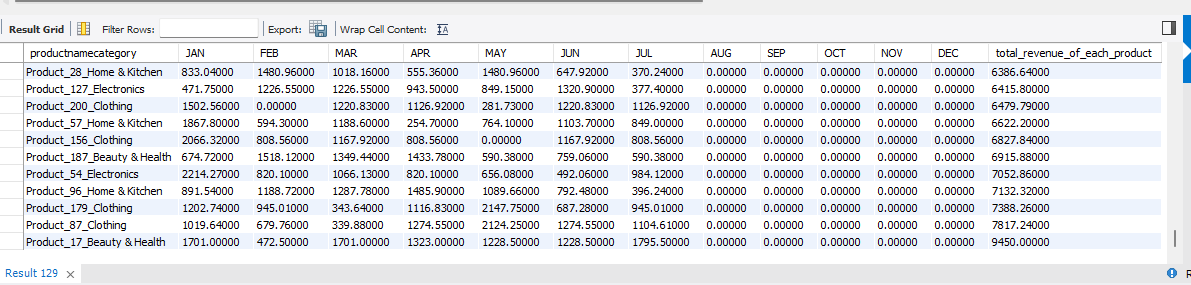
sum(total\_revenue) as total\_revenue\_of\_each\_product

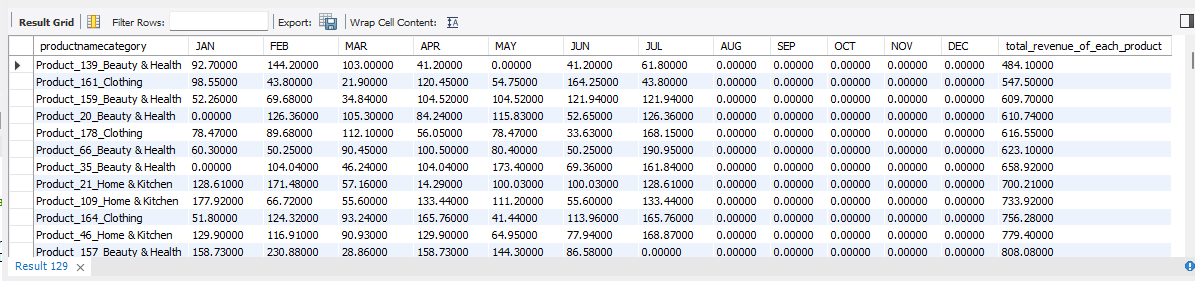
from product\_summary

group by productnamecategory

order by productnamecategory;

Representing the month wise product purchase amount. It’s show that transaction occurred in between Jan till July but after this no sales for four months.





WITH CTE AS

(select productid, Month(TransactionDate) as Month\_Transaction,

SUM(QuantityPurchased \* Price) as total\_revenue, avg(price) as avg\_price

from sales\_transaction\_new

group by productid, Month\_Transaction),

product\_in as

(select productid, category as productcategory from product\_inventory),

product\_summary as

(select pi.productcategory, cte.Month\_Transaction, cte.total\_revenue, cte.avg\_price from cte

JOIN product\_in as pi

ON cte.productid = pi.productid

order by pi.productcategory asc, cte.Month\_Transaction desc)

select productcategory,

SUM(CASE WHEN Month\_Transaction=1 THEN total\_revenue ELSE 0 END) AS 'JAN',

SUM(CASE WHEN Month\_Transaction=2 THEN total\_revenue ELSE 0 END) AS 'FEB',

SUM(CASE WHEN Month\_Transaction=3 THEN total\_revenue ELSE 0 END) AS 'MAR',

SUM(CASE WHEN Month\_Transaction=4 THEN total\_revenue ELSE 0 END) AS 'APR',

SUM(CASE WHEN Month\_Transaction=5 THEN total\_revenue ELSE 0 END) AS 'MAY',

SUM(CASE WHEN Month\_Transaction=6 THEN total\_revenue ELSE 0 END) AS 'JUN',

SUM(CASE WHEN Month\_Transaction=7 THEN total\_revenue ELSE 0 END) AS 'JUL',

SUM(CASE WHEN Month\_Transaction=8 THEN total\_revenue ELSE 0 END) AS 'AUG',

SUM(CASE WHEN Month\_Transaction=9 THEN total\_revenue ELSE 0 END) AS 'SEP',

SUM(CASE WHEN Month\_Transaction=10 THEN total\_revenue ELSE 0 END) AS 'OCT',

SUM(CASE WHEN Month\_Transaction=11 THEN total\_revenue ELSE 0 END) AS 'NOV',

SUM(CASE WHEN Month\_Transaction=12 THEN total\_revenue ELSE 0 END) AS 'DEC',

sum(total\_revenue) as total\_revenue\_of\_each\_category

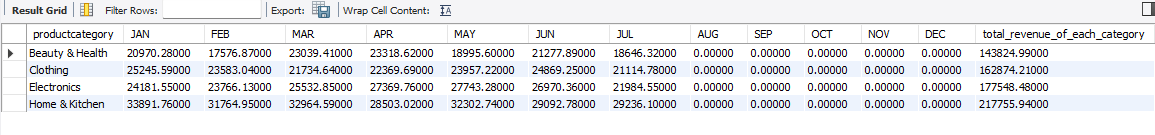
from product\_summary

group by productcategory

order by productcategory;

Product category wise purchase pattern for each month.

It’s represent that Home & Kitchen have the highest revenue from all the category of product and lowest one is Beauty & Health category.



WITH CTE AS

(select productid, YEAR(TransactionDate) as year\_Transaction,

SUM(QuantityPurchased \* Price) as total\_revenue, avg(price) as avg\_price

from sales\_transaction\_new

group by productid, year\_Transaction),

product\_in as

(select productid, category as productcategory from product\_inventory),

product\_summary as

(select pi.productcategory, cte.year\_Transaction, cte.total\_revenue, cte.avg\_price from cte

JOIN product\_in as pi

ON cte.productid = pi.productid

order by pi.productcategory asc, cte.year\_Transaction desc)

select year\_Transaction,

SUM(CASE WHEN productcategory='Beauty & Health' THEN total\_revenue ELSE 0 END) AS 'Beauty & Health',

SUM(CASE WHEN productcategory='Clothing' THEN total\_revenue ELSE 0 END) AS 'Clothing',

SUM(CASE WHEN productcategory='Electronics' THEN total\_revenue ELSE 0 END) AS 'Electronics',

SUM(CASE WHEN productcategory='Home & Kitchen' THEN total\_revenue ELSE 0 END) AS 'Home & Kitchen',

GREATEST(

SUM(CASE WHEN productcategory='Beauty & Health' THEN total\_revenue ELSE 0 END),

SUM(CASE WHEN productcategory='Clothing' THEN total\_revenue ELSE 0 END),

SUM(CASE WHEN productcategory='Electronics' THEN total\_revenue ELSE 0 END),

SUM(CASE WHEN productcategory='Home & Kitchen' THEN total\_revenue ELSE 0 END)

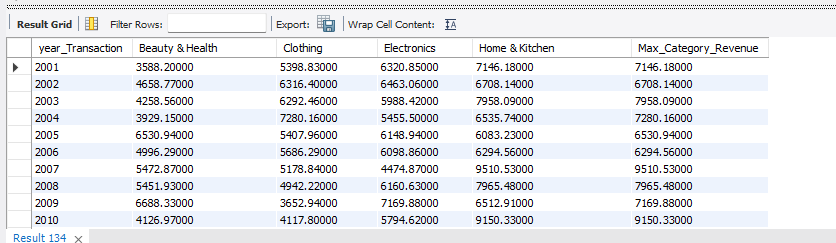
) AS Max\_Category\_Revenue

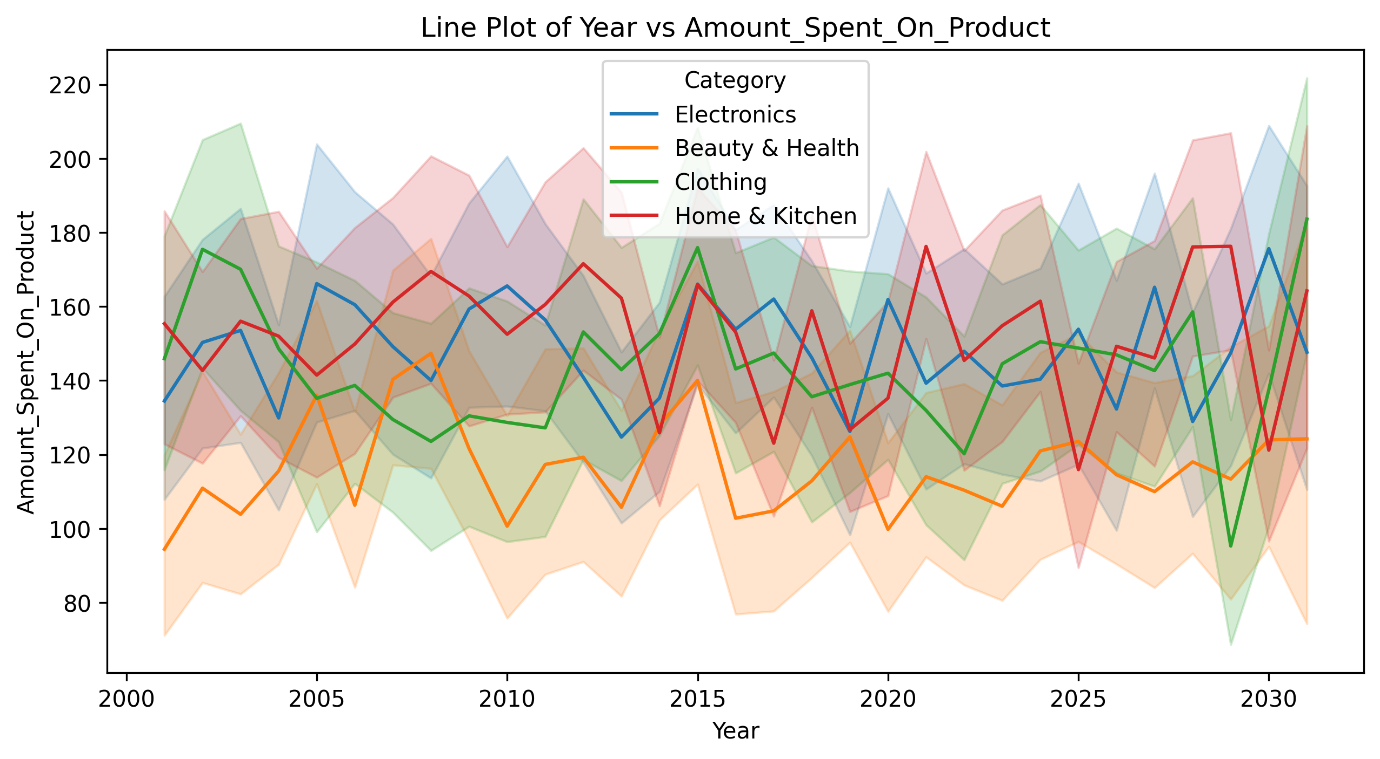
from product\_summary

group by year\_Transaction

order by year\_Transaction;

Category wise per year revenue generated





WITH CTE AS

(select productid, YEAR(TransactionDate) as year\_Transaction,

SUM(QuantityPurchased \* Price) as total\_revenue, avg(price) as avg\_price

from sales\_transaction\_new

group by productid, year\_Transaction),

product\_in as

(select productid, category as productcategory from product\_inventory),

product\_summary as

(select pi.productcategory, cte.year\_Transaction, cte.total\_revenue, cte.avg\_price from cte

JOIN product\_in as pi

ON cte.productid = pi.productid

order by pi.productcategory asc, cte.year\_Transaction desc),

year\_wise\_category as

(select year\_Transaction,

SUM(CASE WHEN productcategory='Beauty & Health' THEN total\_revenue ELSE 0 END) AS Beauty\_Health,

SUM(CASE WHEN productcategory='Clothing' THEN total\_revenue ELSE 0 END) AS Clothing,

SUM(CASE WHEN productcategory='Electronics' THEN total\_revenue ELSE 0 END) AS Electronics,

SUM(CASE WHEN productcategory='Home & Kitchen' THEN total\_revenue ELSE 0 END) AS Home\_Kitchen

from product\_summary

group by year\_Transaction

order by year\_Transaction)

select \*, CASE

WHEN Beauty\_Health >= Clothing AND Beauty\_Health >= Electronics AND Beauty\_Health >= Home\_Kitchen THEN 'Beauty & Health'

WHEN Clothing >= Beauty\_Health AND Clothing >= Electronics AND Clothing >= Home\_Kitchen THEN 'Clothing'

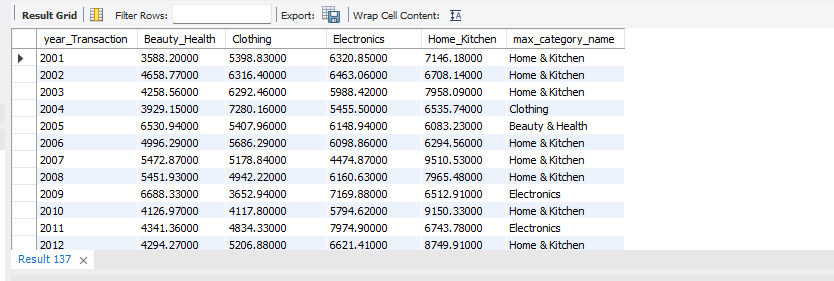
WHEN Electronics >= Beauty\_Health AND Electronics >= Clothing AND Electronics >= Home\_Kitchen THEN 'Electronics'

WHEN Home\_Kitchen >= Beauty\_Health AND Home\_Kitchen >= Clothing AND Home\_Kitchen >= Electronics THEN 'Home & Kitchen'

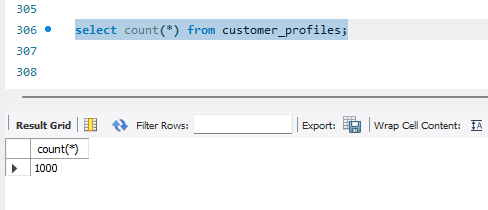
END AS max\_category\_name

from year\_wise\_category;

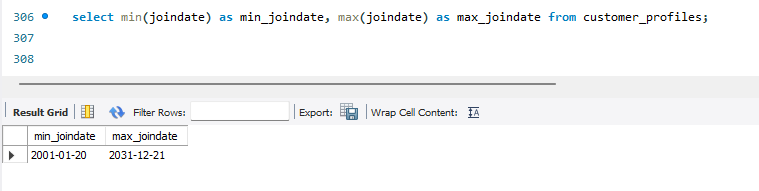
This query represent that each year which category having the highest revenue generated



Total number of records present in customer profiles table



Min joindate and max join date of cuatomer in cutomer profiles table

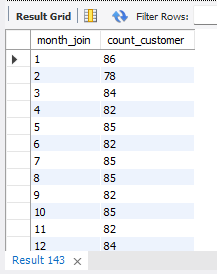


select MONTH(JoinDate) as month\_join, count(\*) as count\_customer from customer\_profiles

group by month\_join

order by month\_join;

Each month how many customers are join in this retail application

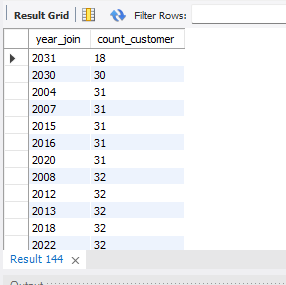


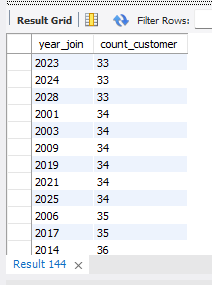
select YEAR(JoinDate) as year\_join, count(\*) as count\_customer from customer\_profiles

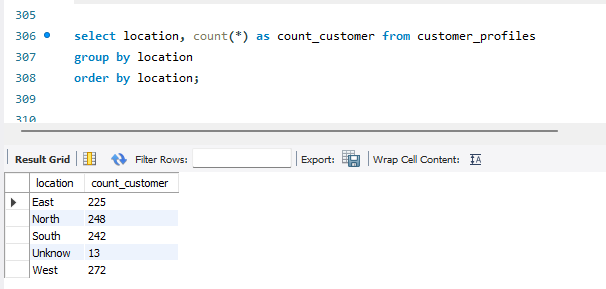
group by year\_join

order by year\_join;

Each year how many customers are join







select

COUNT(CASE WHEN Age between 1 and 10 THEN customerid END) AS 1\_to\_10,

COUNT(CASE WHEN Age between 11 and 20 THEN customerid END) AS 11\_to\_20,

COUNT(CASE WHEN Age between 21 and 30 THEN customerid END) AS 21\_to\_30,

COUNT(CASE WHEN Age between 31 and 40 THEN customerid END) AS 31\_to\_40,

COUNT(CASE WHEN Age between 41 and 50 THEN customerid END) AS 41\_to\_50,

COUNT(CASE WHEN Age between 51 and 60 THEN customerid END) AS 51\_to\_60,

COUNT(CASE WHEN Age between 61 and 70 THEN customerid END) AS 61\_to\_70,

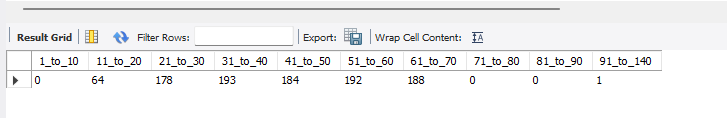
COUNT(CASE WHEN Age between 71 and 80 THEN customerid END) AS 71\_to\_80,

COUNT(CASE WHEN Age between 81 and 90 THEN customerid END) AS 81\_to\_90,

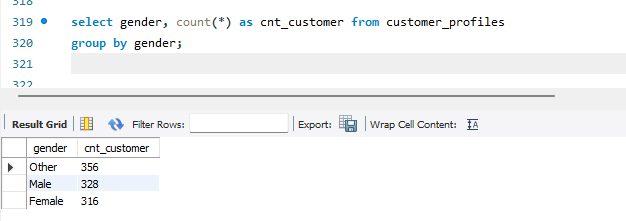
COUNT(CASE WHEN Age between 91 and 140 THEN customerid END) AS 91\_to\_140

from customer\_profiles;

Age wise representation of count of customer present in customer profiles table. Maximum 31 to 40 group age people are purchasing more. 21 to 30 age people are growing in there career and after 30 people become stable in finance so, after that purchase of product are increasing.



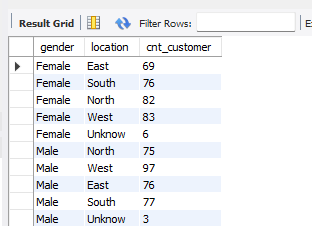
Other group gender are purchasing more than male and female.

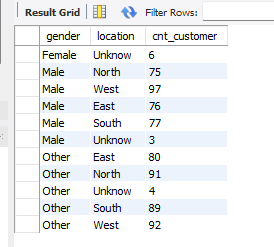


select gender, location, count(\*) as cnt\_customer from customer\_profiles

group by gender, location;

It’s present that gender wise location wise number of customers are present in customer profiles table





select gender,

COUNT(CASE WHEN Age between 1 and 10 THEN customerid END) AS 1\_to\_10,

COUNT(CASE WHEN Age between 11 and 20 THEN customerid END) AS 11\_to\_20,

COUNT(CASE WHEN Age between 21 and 30 THEN customerid END) AS 21\_to\_30,

COUNT(CASE WHEN Age between 31 and 40 THEN customerid END) AS 31\_to\_40,

COUNT(CASE WHEN Age between 41 and 50 THEN customerid END) AS 41\_to\_50,

COUNT(CASE WHEN Age between 51 and 60 THEN customerid END) AS 51\_to\_60,

COUNT(CASE WHEN Age between 61 and 70 THEN customerid END) AS 61\_to\_70,

COUNT(CASE WHEN Age between 71 and 80 THEN customerid END) AS 71\_to\_80,

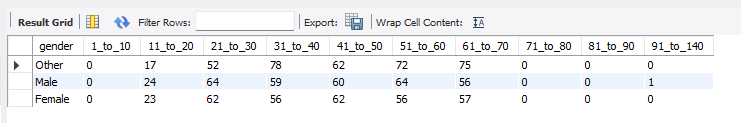
COUNT(CASE WHEN Age between 81 and 90 THEN customerid END) AS 81\_to\_90,

COUNT(CASE WHEN Age between 91 and 140 THEN customerid END) AS 91\_to\_140

from customer\_profiles

group by gender;

Gender and age wise number of customer present in each group category



WITH CTE AS

(select cp.Customerid, cp.age, cp.gender, cp.location, st.productid, st.QuantityPurchased, st.transactiondate, st.price, (st.QuantityPurchased \* st.price) as amount

from customer\_profiles as cp

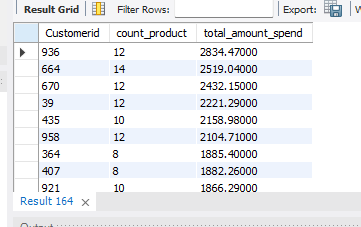
JOIN (select customerid, productid, QuantityPurchased, transactiondate, price from sales\_transaction\_new) as st

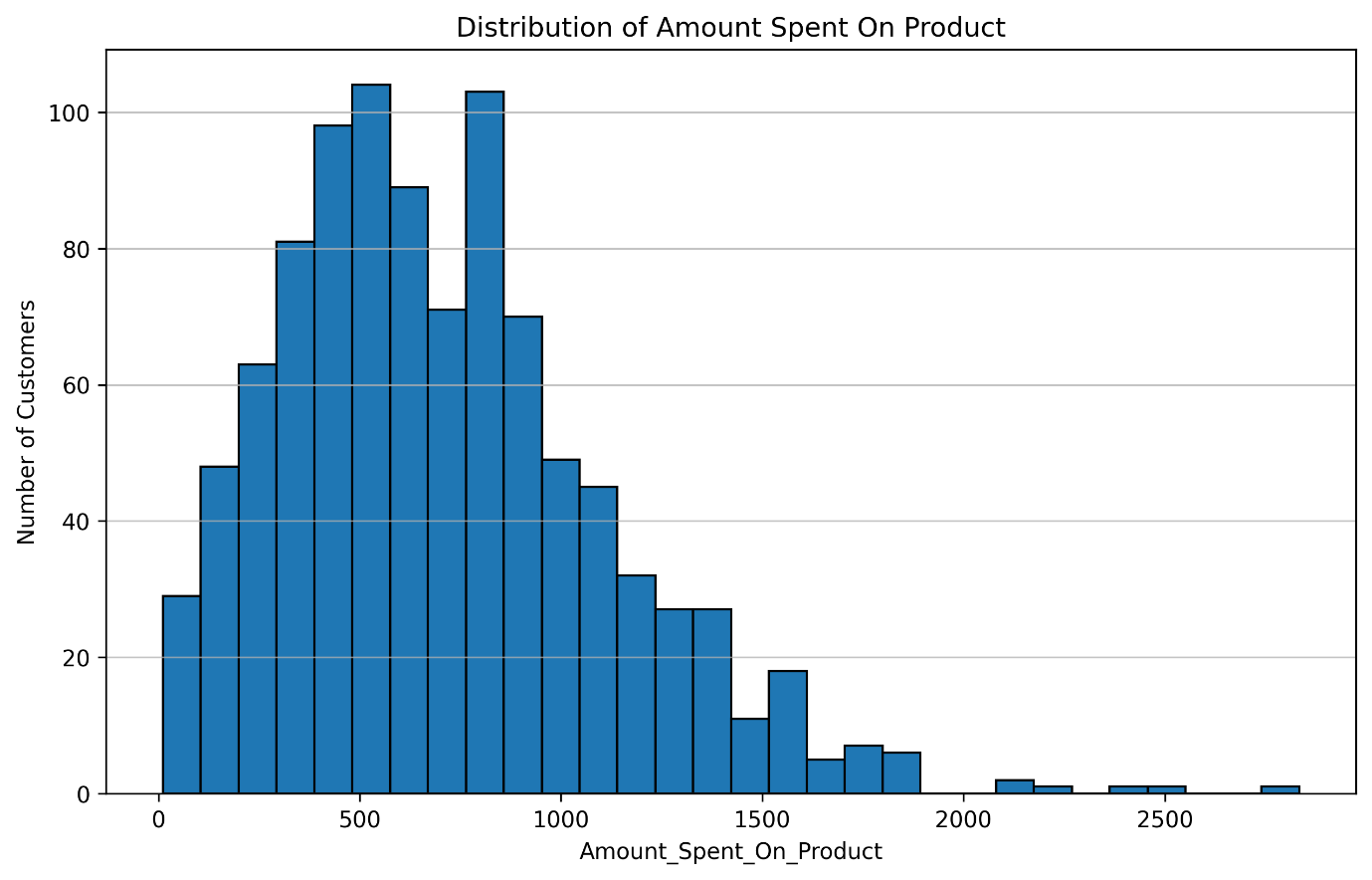
ON cp.customerid = st.customerid)

select customerid, count(productid) as count\_product, sum(amount) as total\_amount\_spend from cte

group by customerid;

It’s represent the each customer how many product purchase with total amount spend





WITH CTE AS

(select cp.Customerid, cp.age, cp.gender, cp.location, st.productid, st.QuantityPurchased, st.transactiondate, st.price, (st.QuantityPurchased \* st.price) as amount

from customer\_profiles as cp

JOIN (select customerid, productid, QuantityPurchased, transactiondate, price from sales\_transaction\_new) as st

ON cp.customerid = st.customerid),

customer\_bev as

(select customerid, count(productid) as count\_product, sum(amount) as total\_amount\_spend from cte

group by customerid)

select \*,

CASE WHEN count\_product=0 THEN 'No Order'

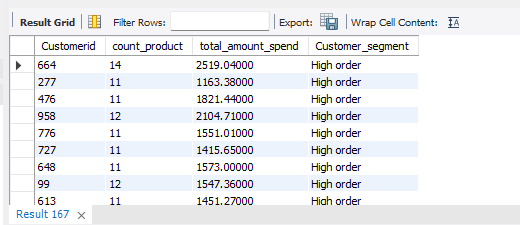
WHEN count\_product between 1 and 4 THEN 'Low Order'

WHEN count\_product between 5 and 10 THEN 'Mid Order'

WHEN count\_product > 10 THEN 'High order' END AS Customer\_segment

from customer\_bev;

According to the purchasing behaviour customer are categories into ‘No order’, ‘Low Order’, ‘Mid order’ and ‘High order’. It’s shows that customer purchasing behaviour.



WITH CTE AS

(select cp.Customerid, cp.age, cp.gender, cp.location, st.productid, st.QuantityPurchased, st.transactiondate, st.price, (st.QuantityPurchased \* st.price) as amount

from customer\_profiles as cp

JOIN (select customerid, productid, QuantityPurchased, transactiondate, price from sales\_transaction\_new) as st

ON cp.customerid = st.customerid),

customer\_bev as

(select customerid, count(productid) as count\_product, sum(amount) as total\_amount\_spend from cte

group by customerid),

customer\_seg as

(select \*,

CASE WHEN round(total\_amount\_spend) between 0 and 500 THEN 'very low order'

WHEN round(total\_amount\_spend) between 501 and 1000 THEN 'low order'

WHEN round(total\_amount\_spend) between 1001 and 1800 THEN 'Mid order'

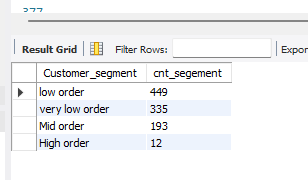
WHEN round(total\_amount\_spend) > 1800 THEN 'High order' END AS Customer\_segment

from customer\_bev)

select Customer\_segment, count(\*) as cnt\_segement from customer\_seg

group by Customer\_segment;

This segregation is based on the total amount spent and low order purchase customer are higher. Highest order customer are less.



WITH CTE AS

(select cp.Customerid, cp.age, cp.gender, cp.location, st.productid, st.QuantityPurchased, st.transactiondate, st.price,

(st.QuantityPurchased \* st.price) as amount, Month(st.transactiondate) as month\_transactiondate, YEAR(st.transactiondate) as year\_transactiondate

from customer\_profiles as cp

JOIN (select customerid, productid, QuantityPurchased, transactiondate, price from sales\_transaction\_new) as st

ON cp.customerid = st.customerid)

select

COUNT(CASE WHEN month\_transactiondate=1 THEN customerid END) AS 'JAN',

COUNT(CASE WHEN month\_transactiondate=2 THEN customerid END) AS 'FEB',

COUNT(CASE WHEN month\_transactiondate=3 THEN customerid END) AS 'MAR',

COUNT(CASE WHEN month\_transactiondate=4 THEN customerid END) AS 'APR',

COUNT(CASE WHEN month\_transactiondate=5 THEN customerid END) AS 'MAY',

COUNT(CASE WHEN month\_transactiondate=6 THEN customerid END) AS 'JUN',

COUNT(CASE WHEN month\_transactiondate=7 THEN customerid END) AS 'JUL',

COUNT(CASE WHEN month\_transactiondate=8 THEN customerid END) AS 'AUG',

COUNT(CASE WHEN month\_transactiondate=9 THEN customerid END) AS 'SEP',

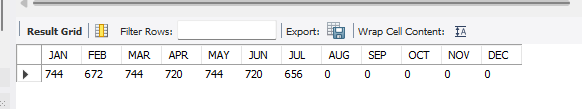
COUNT(CASE WHEN month\_transactiondate=10 THEN customerid END) AS 'OCT',

COUNT(CASE WHEN month\_transactiondate=11 THEN customerid END) AS 'NOV',

COUNT(CASE WHEN month\_transactiondate=12 THEN customerid END) AS 'DEC'

from cte;

This query represent that total number of customer purchase each month. No sales from AUG till DEC.



WITH CTE AS

(select cp.Customerid, cp.age, cp.gender, cp.location, st.productid, st.QuantityPurchased, st.transactiondate, st.price,

(st.QuantityPurchased \* st.price) as amount, Month(st.transactiondate) as month\_transactiondate, YEAR(st.transactiondate) as year\_transactiondate

from customer\_profiles as cp

JOIN (select customerid, productid, QuantityPurchased, transactiondate, price from sales\_transaction\_new) as st

ON cp.customerid = st.customerid)

select customerid,

SUM(CASE WHEN month\_transactiondate=1 THEN amount ELSE 0 END) AS 'JAN',

SUM(CASE WHEN month\_transactiondate=2 THEN amount ELSE 0 END) AS 'FEB',

SUM(CASE WHEN month\_transactiondate=3 THEN amount ELSE 0 END) AS 'MAR',

SUM(CASE WHEN month\_transactiondate=4 THEN amount ELSE 0 END) AS 'APR',

SUM(CASE WHEN month\_transactiondate=5 THEN amount ELSE 0 END) AS 'MAY',

SUM(CASE WHEN month\_transactiondate=6 THEN amount ELSE 0 END) AS 'JUN',

SUM(CASE WHEN month\_transactiondate=7 THEN amount ELSE 0 END) AS 'JUL',

SUM(CASE WHEN month\_transactiondate=8 THEN amount ELSE 0 END) AS 'AUG',

SUM(CASE WHEN month\_transactiondate=9 THEN amount ELSE 0 END) AS 'SEP',

SUM(CASE WHEN month\_transactiondate=10 THEN amount ELSE 0 END) AS 'OCT',

SUM(CASE WHEN month\_transactiondate=11 THEN amount ELSE 0 END) AS 'NOV',

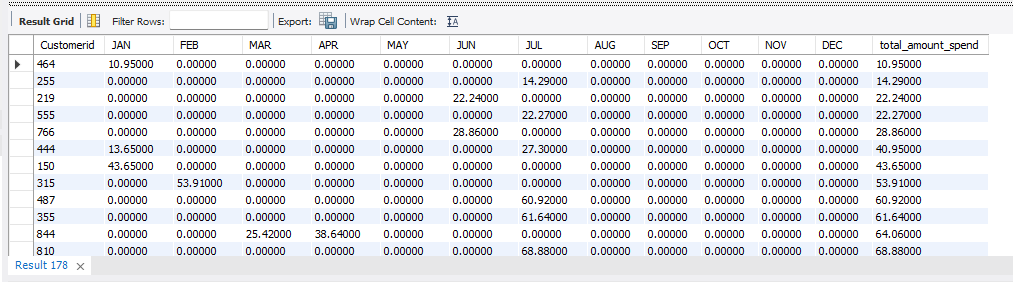
SUM(CASE WHEN month\_transactiondate=12 THEN amount ELSE 0 END) AS 'DEC',

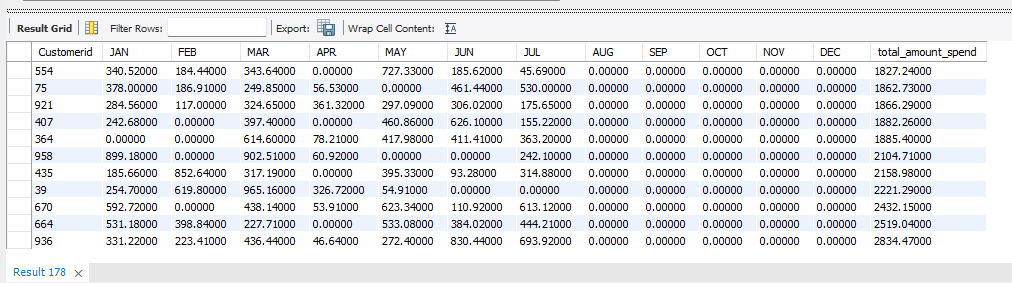
SUM(amount) as total\_amount\_spend

from cte

group by customerid;

Customer wise each month amount spent to any of the product. Customerid – 464 is the lowest amount spend or the customer and customer- 936 is the highest customer how purchase the amount with spenting more money





WITH CTE AS

(select cp.Customerid, cp.age, cp.gender, cp.location, st.productid, st.QuantityPurchased, st.transactiondate, st.price,

(st.QuantityPurchased \* st.price) as amount, Month(st.transactiondate) as month\_transactiondate, YEAR(st.transactiondate) as year\_transactiondate

from customer\_profiles as cp

JOIN (select customerid, productid, QuantityPurchased, transactiondate, price from sales\_transaction\_new) as st

ON cp.customerid = st.customerid)

select IFNULL(customerid, 'TOTAL') AS customerid,

SUM(CASE WHEN year\_transactiondate=2031 THEN amount ELSE 0 END) AS '2031',

SUM(CASE WHEN year\_transactiondate=2030 THEN amount ELSE 0 END) AS '2030',

SUM(CASE WHEN year\_transactiondate=2029 THEN amount ELSE 0 END) AS '2029',

SUM(CASE WHEN year\_transactiondate=2028 THEN amount ELSE 0 END) AS '2028',

SUM(CASE WHEN year\_transactiondate=2027 THEN amount ELSE 0 END) AS '2027',

SUM(CASE WHEN year\_transactiondate=2026 THEN amount ELSE 0 END) AS '2026',

SUM(CASE WHEN year\_transactiondate=2025 THEN amount ELSE 0 END) AS '2025',

SUM(CASE WHEN year\_transactiondate=2024 THEN amount ELSE 0 END) AS '2024',

SUM(CASE WHEN year\_transactiondate=2023 THEN amount ELSE 0 END) AS '2023',

SUM(CASE WHEN year\_transactiondate=2022 THEN amount ELSE 0 END) AS '2022',

SUM(CASE WHEN year\_transactiondate=2021 THEN amount ELSE 0 END) AS '2021',

SUM(CASE WHEN year\_transactiondate=2020 THEN amount ELSE 0 END) AS '2020',

SUM(CASE WHEN year\_transactiondate=2019 THEN amount ELSE 0 END) AS '2019',

SUM(CASE WHEN year\_transactiondate=2018 THEN amount ELSE 0 END) AS '2018',

SUM(CASE WHEN year\_transactiondate=2017 THEN amount ELSE 0 END) AS '2017',

SUM(CASE WHEN year\_transactiondate=2016 THEN amount ELSE 0 END) AS '2016',

SUM(CASE WHEN year\_transactiondate=2015 THEN amount ELSE 0 END) AS '2015',

SUM(CASE WHEN year\_transactiondate=2014 THEN amount ELSE 0 END) AS '2014',

SUM(CASE WHEN year\_transactiondate=2013 THEN amount ELSE 0 END) AS '2013',

SUM(CASE WHEN year\_transactiondate=2012 THEN amount ELSE 0 END) AS '2012',

SUM(CASE WHEN year\_transactiondate=2011 THEN amount ELSE 0 END) AS '2011',

SUM(CASE WHEN year\_transactiondate=2010 THEN amount ELSE 0 END) AS '2010',

SUM(CASE WHEN year\_transactiondate=2009 THEN amount ELSE 0 END) AS '2009',

SUM(CASE WHEN year\_transactiondate=2008 THEN amount ELSE 0 END) AS '2008',

SUM(CASE WHEN year\_transactiondate=2007 THEN amount ELSE 0 END) AS '2007',

SUM(CASE WHEN year\_transactiondate=2006 THEN amount ELSE 0 END) AS '2006',

SUM(CASE WHEN year\_transactiondate=2005 THEN amount ELSE 0 END) AS '2005',

SUM(CASE WHEN year\_transactiondate=2004 THEN amount ELSE 0 END) AS '2004',

SUM(CASE WHEN year\_transactiondate=2003 THEN amount ELSE 0 END) AS '2003',

SUM(CASE WHEN year\_transactiondate=2002 THEN amount ELSE 0 END) AS '2002',

SUM(CASE WHEN year\_transactiondate=2001 THEN amount ELSE 0 END) AS '2001',

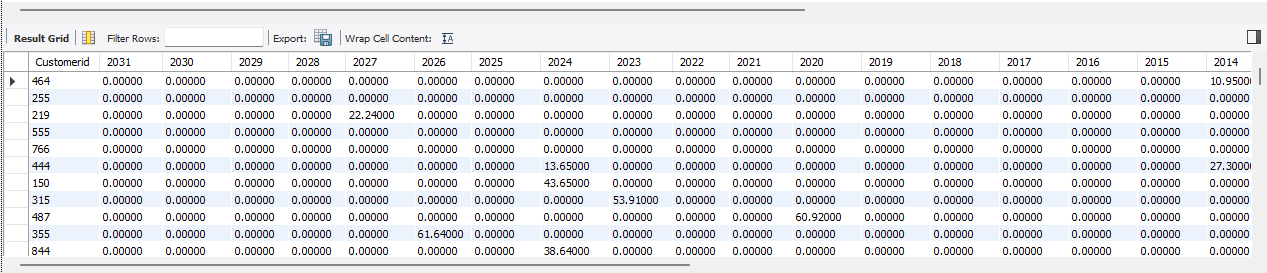
SUM(amount) as total\_amount\_spend,

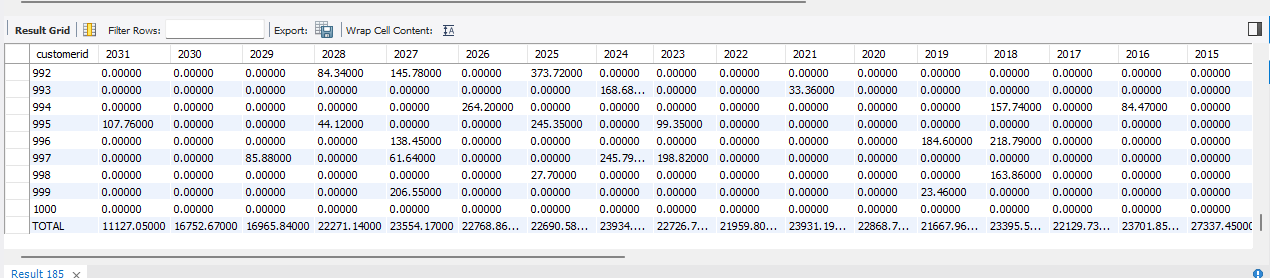
count(amount) as frequency\_buying\_yearly

from cte

group by customerid WITH ROLLUP;

Year wise customer purchase behaviour





WITH CTE AS

(select cp.Customerid, cp.age, cp.gender, cp.location, st.productid, st.QuantityPurchased, st.transactiondate, st.price,

(st.QuantityPurchased \* st.price) as amount, Month(st.transactiondate) as month\_transactiondate, YEAR(st.transactiondate) as year\_transactiondate

from customer\_profiles as cp

JOIN (select customerid, productid, QuantityPurchased, transactiondate, price from sales\_transaction\_new) as st

ON cp.customerid = st.customerid)

select year\_transactiondate,

SUM(CASE WHEN age between 11 and 20 THEN amount ELSE 0 END) AS 11\_20,

SUM(CASE WHEN age between 21 and 30 THEN amount ELSE 0 END) AS 21\_30,

SUM(CASE WHEN age between 31 and 40 THEN amount ELSE 0 END) AS 31\_40,

SUM(CASE WHEN age between 41 and 50 THEN amount ELSE 0 END) AS 41\_50,

SUM(CASE WHEN age between 51 and 60 THEN amount ELSE 0 END) AS 51\_60,

SUM(CASE WHEN age between 61 and 70 THEN amount ELSE 0 END) AS 61\_70,

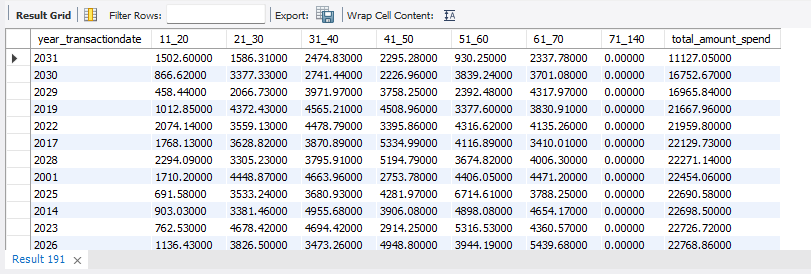
SUM(CASE WHEN age between 71 and 140 THEN amount ELSE 0 END) AS 71\_140,

sum(amount) as total\_amount\_spend

from cte

group by year\_transactiondate;

Age group wise amount spent each year



WITH CTE AS

(select cp.Customerid, cp.age, cp.gender, cp.location, st.productid, st.QuantityPurchased, st.transactiondate, st.price,

(st.QuantityPurchased \* st.price) as amount, Month(st.transactiondate) as month\_transactiondate, YEAR(st.transactiondate) as year\_transactiondate

from customer\_profiles as cp

JOIN (select customerid, productid, QuantityPurchased, transactiondate, price from sales\_transaction\_new) as st

ON cp.customerid = st.customerid),

cust\_prod as

(select customerid, productid, count(\*) as cnt from cte

group by customerid, productid

having count(\*)>1

order by customerid, productid),

cust as

(select customerid, productid, month\_transactiondate, year\_transactiondate

from cte where (customerid, productid) in(select customerid,productid from cust\_prod)

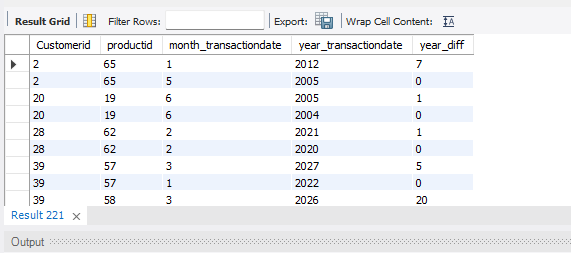
order by customerid, productid asc, year\_transactiondate desc)

select \*,

IFNULL(ABS(year\_transactiondate - LAG(year\_transactiondate) OVER (PARTITION BY customerid, productid ORDER BY year\_transactiondate)), 0) AS year\_diff

from cust;

This query represent that which product are purchase more with buying behaviour or year difference between same product purchase. Customer are buying same product in same year or having 1,2,3,4…etc gap for buying same product.



WITH CTE AS

(select cp.Customerid, cp.age, cp.gender, cp.location, st.productid, st.QuantityPurchased, st.transactiondate, st.price,

(st.QuantityPurchased \* st.price) as amount, Month(st.transactiondate) as month\_transactiondate, YEAR(st.transactiondate) as year\_transactiondate

from customer\_profiles as cp

JOIN (select customerid, productid, QuantityPurchased, transactiondate, price from sales\_transaction\_new) as st

ON cp.customerid = st.customerid),

cust\_prod as

(select customerid, productid, count(\*) as cnt from cte

group by customerid, productid

having count(\*)>1

order by customerid, productid),

cust as

(select customerid, productid, month\_transactiondate, year\_transactiondate

from cte where (customerid, productid) in(select customerid,productid from cust\_prod)

order by customerid, productid asc, year\_transactiondate desc),

cust\_purchase as

(select \*,

IFNULL(ABS(year\_transactiondate - LAG(year\_transactiondate) OVER (PARTITION BY customerid, productid ORDER BY year\_transactiondate)), 0) AS year\_diff

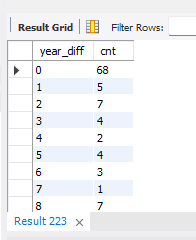
from cust)

select year\_diff , count(distinct customerid) as cnt from cust\_purchase

group by year\_diff

order by year\_diff;

Zero presenet that same year customer purchased more same product after that pattern are increasing or decreasing order



SELECT

customerid,

YEAR(transactiondate) AS purchase\_year,

MONTH(transactiondate) AS purchase\_month,

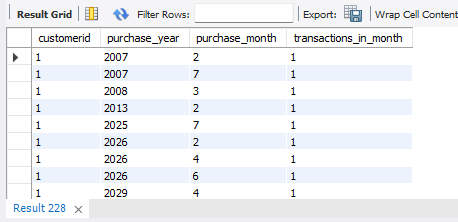
COUNT(\*) AS transactions\_in\_month

FROM sales\_transaction\_new

GROUP BY customerid, purchase\_year, purchase\_month

ORDER BY customerid, purchase\_year, purchase\_month;

Purchases per customer happen each year/month



WITH CTE AS (

SELECT

cp.Customerid,

st.transactiondate

FROM customer\_profiles cp

JOIN sales\_transaction\_new st ON cp.customerid = st.customerid

),

frequency\_calc AS (

SELECT

customerid,

transactiondate,

LAG(transactiondate) OVER (PARTITION BY customerid ORDER BY transactiondate) AS prev\_transaction

FROM CTE

)

SELECT

customerid,

COUNT(\*) AS total\_transactions,

ROUND(AVG(DATEDIFF(transactiondate, prev\_transaction)), 2) AS avg\_days\_between\_purchases

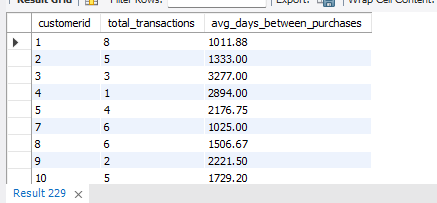
FROM frequency\_calc

WHERE prev\_transaction IS NOT NULL

GROUP BY customerid

ORDER BY customerid;

Customer did how many transaction with avearge days purchasing pattern



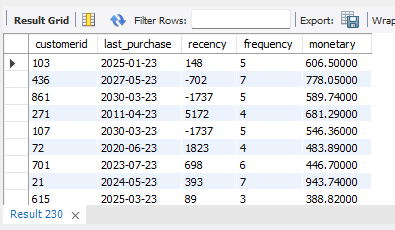
SELECT customerid, MAX(transactiondate) AS last\_purchase, DATEDIFF(CURDATE(), MAX(transactiondate)) AS recency,

COUNT(\*) AS frequency, SUM(QuantityPurchased \* price) AS monetary

FROM sales\_transaction\_new

GROUP BY customerid;

This query represent the last purchase date of customer with difference between the number of days happend from today’s date that is recency and number of time customer purchased the product that is frequency of customer with total amount spent is monetary



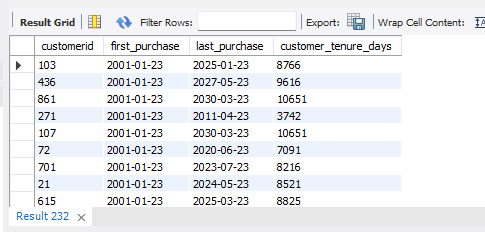
SELECT customerid, MIN(transactiondate) AS first\_purchase, MAX(transactiondate) AS last\_purchase,

DATEDIFF(MAX(transactiondate), MIN(transactiondate)) AS customer\_tenure\_days

FROM sales\_transaction\_new

GROUP BY customerid;

Customer first and last day of purchase with total days spent on this retail application



WITH rfm\_raw AS (

SELECT

customerid,

MAX(transactiondate) AS last\_purchase,

DATEDIFF(CURDATE(), MAX(transactiondate)) AS recency,

COUNT(\*) AS frequency,

round(SUM(QuantityPurchased \* price)) AS monetary

FROM sales\_transaction\_new

GROUP BY customerid

),

rfm\_scored AS (

SELECT \*,CASE WHEN recency <= 30 THEN 3

WHEN recency <= 90 THEN 2

ELSE 1 END AS r\_score,

CASE WHEN frequency >= 10 THEN 3

WHEN frequency >= 5 THEN 2

ELSE 1 END AS f\_score,

CASE WHEN monetary >= 10000 THEN 3

WHEN monetary >= 5000 THEN 2

ELSE 1 END AS m\_score

FROM rfm\_raw

),

rfm\_final AS (

SELECT \*,

CONCAT(r\_score, f\_score, m\_score) AS rfm\_score,

CASE

WHEN r\_score = 3 AND f\_score = 3 AND m\_score = 3 THEN 'Champion'

WHEN r\_score = 3 AND f\_score = 2 THEN 'Loyal'

WHEN r\_score = 2 AND f\_score >= 2 THEN 'Potential Loyalist'

WHEN r\_score = 1 AND f\_score = 3 THEN 'At Risk'

WHEN r\_score = 1 AND f\_score = 1 AND m\_score = 1 THEN 'Lost'

ELSE 'Others'

END AS rfm\_category

FROM rfm\_scored

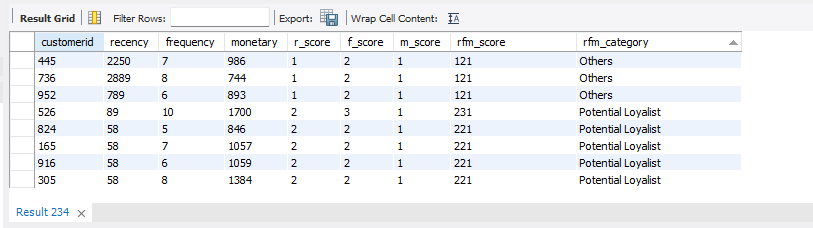
)

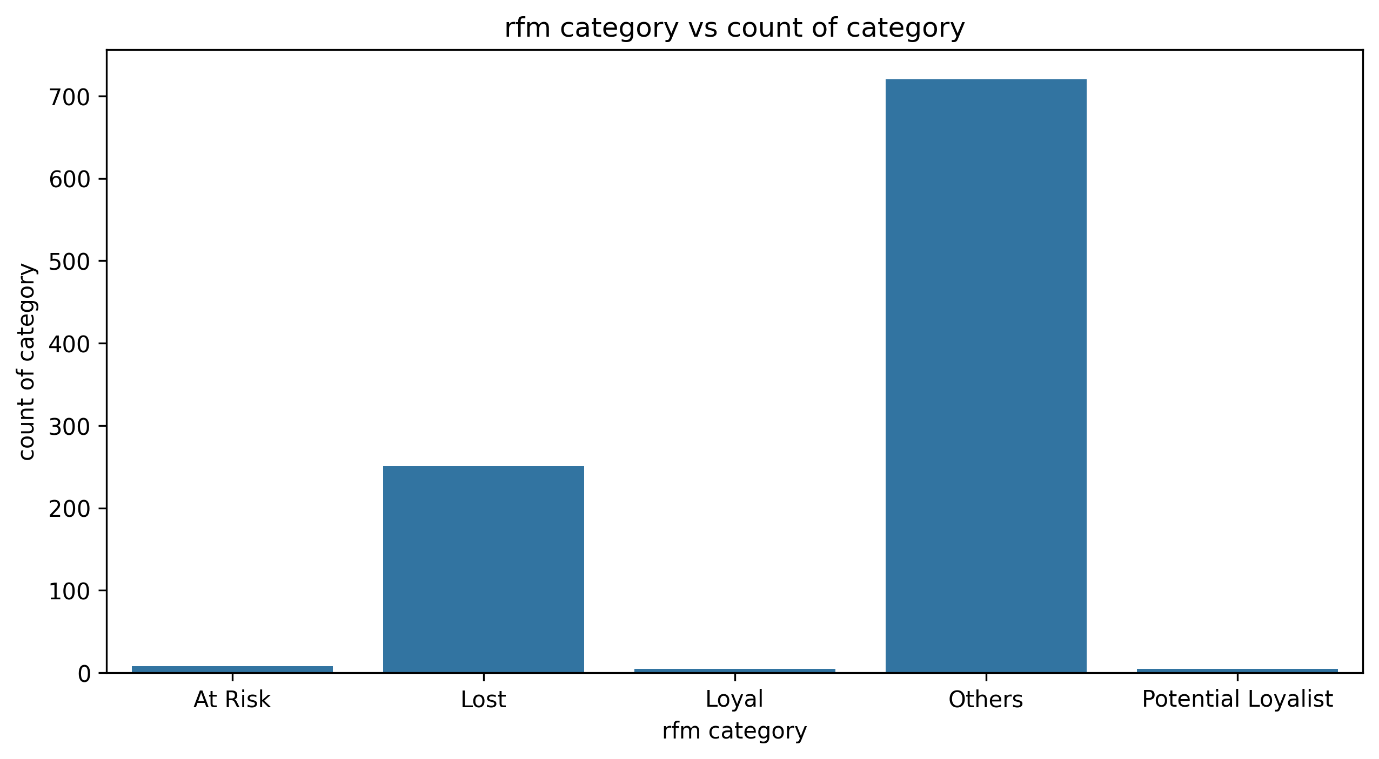
SELECT

customerid, recency, frequency, monetary, r\_score, f\_score, m\_score, rfm\_score, rfm\_category

FROM rfm\_final ORDER BY rfm\_score DESC;

This query represent that customer behaviour on the basis of recency, frequency and monentay. Calculating the score of each behaviour and on the basis of rfm category are decided like, champions, loyal, potential loyalist, at risk, lost and others.





# **Recommandation to the problem:**

1. People have low awareness about beauty and health, so they are buying fewer products. The data covers the years 2001 to 2031, and only in 2005 did the revenue from health and beauty products reach its highest point.
2. There are no "champion" customers in this data. The focus should be on retaining **loyal customers** and turning **potential loyalists** into champions by offering new deals or discounts.
3. **Loyal Customers**: Offer coupons or bigger discounts if they buy products worth a certain amount.
4. **Potential Loyalists**: Give them bonus loyality point for 2 or 3 puchase to they have buy more product with this point or redeem them.
5. **At-Risk Customers**: Offer exchange deals (e.g., trade in old products and pay a small amount for new ones) or run big sales like Amazon’s Big Billion Days to bring them back.

To view the same analysis in pandas refer the link :-

https://colab.research.google.com/drive/1UgCrF6BhB1VPbKQyjjYISk-VuAMPSuA0#scrollTo=pNksgM\_0Gy0L