SELECT \* FROM newschema.online\_retail;

-- Define metadata in MySQL Workbench or any SQL tool

-- Metadata refers to data about data. In MySQL Workbench or any SQL tool, metadata includes the structure of the database, tables, fields, types, indexes, relationships, and other constraints.

-- Distribution of order values across all customers

use newschema;

SELECT CustomerID, SUM(Quantity \* UnitPrice) AS total\_order\_value

FROM online\_retail

GROUP BY CustomerID

ORDER BY total\_order\_value DESC;

-- How many unique products has each customer purchased?

-- count the distinct products (based on StockCode) each customer has bought.

SELECT CustomerID, COUNT(DISTINCT StockCode) AS unique\_products

FROM online\_retail

GROUP BY CustomerID;

/\* Which customers have only made a single purchase?

find customers who have only one record in the online\_retail table.\*/

SELECT CustomerID

FROM online\_retail

GROUP BY CustomerID

HAVING COUNT(InvoiceNo) = 1;

-- Which products are most commonly purchased together by customers?

SELECT a.StockCode AS Product1, b.StockCode AS Product2, COUNT(\*) AS Frequency

FROM online\_retail a

JOIN online\_retail b ON a.InvoiceNo = b.InvoiceNo AND a.StockCode != b.StockCode

GROUP BY Product1, Product2

ORDER BY Frequency DESC

LIMIT 10;

/\*1. Customer Segmentation by Purchase Frequency

Group customers into segments based on their purchase frequency, such as high, medium, and low frequency customers.

This can help you identify your most loyal customers and those who need more attention. \*/

SELECT CustomerID, purchase\_frequency,

CASE

WHEN purchase\_frequency > 50 THEN 'High'

WHEN purchase\_frequency BETWEEN 20 AND 50 THEN 'Medium'

ELSE 'Low'

END AS frequency\_segment

FROM (

SELECT CustomerID, COUNT(InvoiceNo) AS purchase\_frequency

FROM online\_retail

GROUP BY CustomerID

) AS freq;

/\* Yeh query customers ko unki purchase frequency ke hisaab se 3 groups mein daalti hai:

High Frequency: Jo customers bohot zyada shopping karte hain, unka purchase count 50 se zyada hota hai.

Medium Frequency: Jo customers moderate shopping karte hain, unka count 20 aur 50 ke darmiyan hota hai.

Low Frequency: Jo customers kam shopping karte hain, unka purchase count 20 se kam hota hai.

Maqsad:

Isse tumhein pata chalega ke kaunse customers loyal hain (High Frequency),

aur kaunse customers ko zyada tawajjo dene ki zaroorat hai (Low Frequency).

Yeh analysis tumhein marketing aur customer engagement strategies ko better banane mein madad karega. \*/

/\* 2. Average Order Value by Country

Calculate the average order value for each country to identify

where your most valuable customers are located. \*/

SELECT Country, AVG(total\_order\_value) AS average\_order\_value

FROM (

SELECT Country, (Quantity \* UnitPrice) AS total\_order\_value

FROM online\_retail

) AS order\_values

GROUP BY Country

ORDER BY average\_order\_value DESC;

/\* Yeh query har country ka average order value nikalti hai. Pehle subquery mein hum har

order ki total value calculate kar rahay hain using Quantity \* UnitPrice. Phir hum outer query mein har

country ka average nikal rahay hain.

Maqsad:

Yeh tumhein yeh identify karne mein madad dega ke kis country ke customers sabse

zyada valuable hain, taake tum un countries mein apni marketing aur efforts zyada focus kar sako. \*/

SELECT CustomerID

FROM (

SELECT CustomerID, MAX(InvoiceDate) AS last\_purchase\_date

FROM online\_retail

GROUP BY CustomerID

) AS last\_purchase

WHERE last\_purchase\_date < DATE\_SUB(CURDATE(), INTERVAL 6 MONTH);

/\* MAX(InvoiceDate): Yeh function har customer ke liye sabse recent purchase date ko nikal raha hai.

MAX function ki wajah se humhein har customer ki last (yaani sabse latest) purchase date mil jaati hai.

GROUP BY CustomerID: Yeh customers ko unke CustomerID ke base par group karta hai,

taake har customer ki alag se calculation ho sake.

CURDATE(): Yeh function current date ko return karta hai, for example, agar aaj ki date 2024-09-05 hai,

toh CURDATE() yeh value return karega.

DATE\_SUB(CURDATE(), INTERVAL 6 MONTH): Yeh function current date se 6 months peechay jaa kar ek date generate karta hai.

For example, agar aaj ki date 2024-09-05 hai, toh yeh 2024-03-05 ki date nikalay ga.

WHERE last\_purchase\_date < DATE\_SUB(CURDATE(), INTERVAL 6 MONTH): Yeh condition check kar rahi hai

ke kaunse customers ki last purchase date 6 months se purani hai. Agar last\_purchase\_date 6 months peechay ki date ho,

toh us customer ko result mein include kiya jaayega. \*/

/\* 4. Product Affinity Analysis

Determine which products are often purchased together by

calculating the correlation between product purchases. \*/

SELECT Product1, Product2, COUNT(\*) AS Frequency

FROM (

SELECT a.StockCode AS Product1, b.StockCode AS Product2

FROM online\_retail a

JOIN online\_retail b ON a.InvoiceNo = b.InvoiceNo AND a.StockCode != b.StockCode

) AS product\_pairs

GROUP BY Product1, Product2

ORDER BY Frequency DESC

LIMIT 10;

/\* Is query mein hum un product pairs ko identify karte hain jo aksar customers saath mein purchase karte hain.

Subquery mein hum invoice number ke zariye products ko pair karte hain jo ek hi order mein liye gaye hain (JOIN).

Phir outer query se yeh check hota hai ke kaunse products frequently saath mein purchase hote hain.

Yeh product affinity analysis humhein cross-selling aur product bundling ke ideas dene mein madad karega.

hum dekh sakte hai ke kaunse products ko log ek saath khareedte hain, aur accordingly combo deals ya bundles bana sakte ho. \*/

SET SQL\_SAFE\_UPDATES = 0;

UPDATE online\_retail

SET InvoiceDate = STR\_TO\_DATE(InvoiceDate, '%m/%d/%Y %H:%i')

WHERE InvoiceDate IS NOT NULL;

ALTER TABLE online\_retail

MODIFY COLUMN InvoiceDate DATETIME;

SELECT year, month\_name, SUM(total\_sales) AS total\_sales

FROM (

SELECT YEAR(InvoiceDate) AS year,

MONTHNAME(InvoiceDate) AS month\_name,

(Quantity \* UnitPrice) AS total\_sales

FROM online\_retail

WHERE InvoiceDate IS NOT NULL -- Ensure we only consider non-null dates

) AS monthly\_sales

GROUP BY year, month\_name

ORDER BY year, month\_name;

/\* SET SQL\_SAFE\_UPDATES = 0;

Yeh line SQL safe updates ko disable karti hai. Default setting mein MySQL aap ko bina WHERE clause ke koi UPDATE ya DELETE query chalane se rokti hai.

Is line ka matlab hai ke aap is safety ko hata rahe ho taake aap apni query freely chala sako.

UPDATE online\_retail

SET InvoiceDate = STR\_TO\_DATE(InvoiceDate, '%m/%d/%Y %H:%i') WHERE InvoiceDate IS NOT NULL;

Is query ka kaam yeh hai ke online\_retail table ke andar jo bhi InvoiceDate hai usay update kiya jaye.

STR\_TO\_DATE(InvoiceDate, '%m/%d/%Y %H:%i') ek string format (mm/dd/yyyy hh:mm) ko MySQL ke proper date-time format mein convert karta hai.

WHERE InvoiceDate IS NOT NULL yeh condition ensure karti hai ke sirf woh rows update hon jisme InvoiceDate null (khaali) nahi hai.

ALTER TABLE online\_retail MODIFY COLUMN InvoiceDate DATETIME;

Is query ka kaam yeh hai ke InvoiceDate column ko DATETIME type mein modify kiya jaye.

Iska matlab hai ke ab is column ke sare data ko date-time ke tor par treat kiya jayega.

SELECT year, month\_name, SUM(total\_sales) AS total\_sales

FROM ( SELECT YEAR(InvoiceDate) AS year, MONTHNAME(InvoiceDate) AS month\_name, (Quantity \* UnitPrice) AS total\_sales

FROM online\_retail WHERE InvoiceDate IS NOT NULL ) AS monthly\_sales

GROUP BY year, month\_name ORDER BY year, month\_name;

Is query ka kaam yeh hai ke total sales ko year aur month ke hisaab se nikaala jaye.

Inner query ke andar:

YEAR(InvoiceDate) se InvoiceDate ka year nikaala jaata hai.

MONTHNAME(InvoiceDate) se InvoiceDate ka month ka naam (jaise "January", "February") liya jata hai.

Quantity \* UnitPrice total sales ko har row ke liye calculate karta hai.

Outer query mein:

Yeh total sales ko year aur month ke hisaab se group karti hai (GROUP BY year, month\_name).

Fir yeh total sales ko har group ke liye sum karti hai (SUM(total\_sales)).

Aur aakhir mein result ko year aur month name ke hisaab se order kiya jata hai. \*/

-- the end of project