**CS689**

**FINAL REPORT**

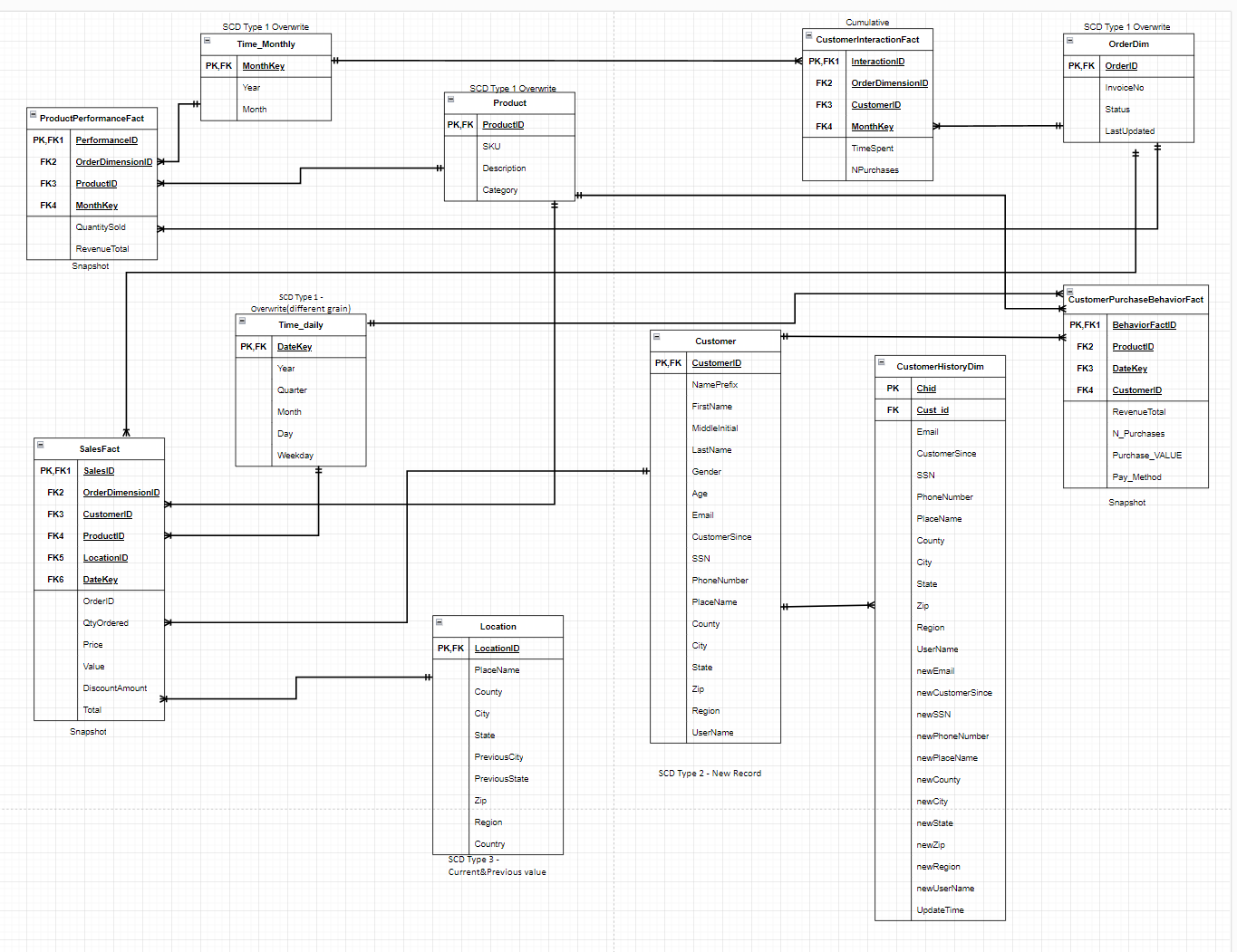
| Weilin Lu | U88797308  luweilin@bu.edu |
| --- | --- |
| Yuhan Xu | U05289286  yuhan09@bu.edu |

# 

# 

# 

# **Design**



# 

# **ETL**

### SQL: Table Create

CREATE TABLE Product (

Product\_id NVARCHAR(4000) PRIMARY KEY,

SKU NVARCHAR(4000),

Description NVARCHAR(4000),

Category NVARCHAR(4000)

);

CREATE TABLE Customer (

Cust\_id NVARCHAR(4000) PRIMARY KEY,

Name\_Prefix NVARCHAR(4000),

First\_Name NVARCHAR(4000),

Middle\_Initial NVARCHAR(4000),

Last\_Name NVARCHAR(4000),

Gender NVARCHAR(4000),

Age INT,

full\_name NVARCHAR(4000),

E\_mail NVARCHAR(4000),

Customer\_Since NVARCHAR(4000),

SSN NVARCHAR(4000),

Phone\_No NVARCHAR(4000),

PlaceName NVARCHAR(4000),

County NVARCHAR(4000),

City NVARCHAR(4000),

State NVARCHAR(4000),

Zip NVARCHAR(4000),

Region NVARCHAR(4000),

User\_Name NVARCHAR(4000)

);

CREATE TABLE Location (

Lid INT PRIMARY KEY IDENTITY(1,1),

LocationID NVARCHAR(4000),

Place\_Name NVARCHAR(4000),

County NVARCHAR(4000),

City NVARCHAR(4000),

State NVARCHAR(4000),

PreviousCity NVARCHAR(4000),

PreviousState NVARCHAR(4000),

Zip VARCHAR(4000),

Region NVARCHAR(4000),

Country NVARCHAR(4000)

);

CREATE TABLE OrderDim (

Oid INT PRIMARY KEY,

OrderID NVARCHAR(4000),

InvoiceNo NVARCHAR(4000),

Status NVARCHAR(4000),

LastUpdated NVARCHAR(4000)

);

CREATE TABLE Time\_Monthly (

MonthKey NVARCHAR(4000) PRIMARY KEY,

year NVARCHAR(4000),

month NVARCHAR(4000)

);

CREATE TABLE Time\_daily (

DateKey NVARCHAR(4000) PRIMARY KEY,

Year NVARCHAR(4000),

Quarter NVARCHAR(4000),

Month NVARCHAR(4000),

Day NVARCHAR(4000),

);

CREATE TABLE SalesFact (

SalesID INT PRIMARY KEY,

OrderID NVARCHAR(4000),

Oid INT,

Cust\_id NVARCHAR(4000),

Product\_id NVARCHAR(4000),

LocationID NVARCHAR(4000),

Lid INT,

DateKey NVARCHAR(4000),

qty\_ordered INT,

Price DECIMAL(18,2),

Value DECIMAL(18,2),

Discount\_Amount DECIMAL(18,2),

Total DECIMAL(18,2),

FOREIGN KEY (Oid) REFERENCES OrderDim(Oid),

FOREIGN KEY (Cust\_id) REFERENCES Customer(Cust\_id),

FOREIGN KEY (Product\_id) REFERENCES Product(Product\_id),

FOREIGN KEY (Lid) REFERENCES Location(Lid),

FOREIGN KEY (DateKey) REFERENCES Time\_daily(DateKey)

);

CREATE TABLE ProductPerformanceFact (

PerformanceID INT PRIMARY KEY,

OrderID NVARCHAR(4000),

Oid INT,

Product\_id NVARCHAR(4000),

MonthKey NVARCHAR(4000),

qty\_ordered INT,

Revenue\_Total DECIMAL(18,2),

FOREIGN KEY (Oid) REFERENCES OrderDim(Oid),

FOREIGN KEY (Product\_id) REFERENCES Product(Product\_id),

FOREIGN KEY (MonthKey) REFERENCES Time\_Monthly(MonthKey)

);

CREATE TABLE CustomerInteractionFact (

InteractionID INT PRIMARY KEY,

OrderID NVARCHAR(4000),

Oid INT,

Cust\_id NVARCHAR(4000),

MonthKey NVARCHAR(4000),

TimeSpent INT,

N\_Purchases INT,

FOREIGN KEY (Oid) REFERENCES OrderDim(Oid),

FOREIGN KEY (Cust\_id) REFERENCES Customer(Cust\_id),

FOREIGN KEY (MonthKey) REFERENCES Time\_Monthly(MonthKey)

);

CREATE TABLE CustomerPurchaseBehaviorFact (

BehaviorFactID INT PRIMARY KEY,

Cust\_id NVARCHAR(4000),

Product\_id NVARCHAR(4000) ,

DateKey NVARCHAR(4000),

Revenue\_Total DECIMAL(19,4),

N\_Purchases INT,

Purchase\_VALUE DECIMAL(19,4),

Pay\_Method INT,

FOREIGN KEY (Cust\_id) REFERENCES Customer(Cust\_id),

FOREIGN KEY (Product\_id) REFERENCES Product(Product\_id),

FOREIGN KEY (DateKey) REFERENCES Time\_daily(DateKey)

);

### Python: Load into Dataframe

import pandas as pd

# load Customer.csv

print("======================= Load Customer.csv =======================")

Cust\_file = 'G:\BU\_STUDY\METCS689A1\FP\Customer.csv'

df\_cust = pd.read\_csv(Cust\_file)

print(df\_cust.head())

print()

# load Location.csv

print("======================= Load Location.csv =======================")

Loc\_file = 'G:\BU\_STUDY\METCS689A1\FP\Location.csv'

df\_Loc = pd.read\_csv(Loc\_file, keep\_default\_na=False)

print(df\_Loc.head())

print()

# load OrderDim.csv

print("======================= Load OrderDim.csv =======================")

od\_file = 'G:\BU\_STUDY\METCS689A1\FP\OrderDim.csv'

df\_od = pd.read\_csv(od\_file)

print(df\_od.head())

print()

# load product.csv

print("======================= Load product.csv =======================")

product\_file = 'G:\BU\_STUDY\METCS689A1\FP\product.csv'

df\_product = pd.read\_csv(product\_file)

print(df\_product.head())

print()

# load time\_monthly.csv

print("======================= Load time\_monthly.csv =======================")

tm\_file = r'G:\BU\_STUDY\METCS689A1\FP\time\_monthly.csv'

df\_tm = pd.read\_csv(tm\_file)

print(df\_tm.head())

print()

# load time\_daily.csv

print("======================= Load time\_daily.csv =======================")

td\_file = r'G:\BU\_STUDY\METCS689A1\FP\time\_daily.csv'

df\_td = pd.read\_csv(td\_file)

print(df\_td.head())

print()

# load SalesFact.csv

print("======================= Load SalesFact.csv =======================")

sf\_file = 'G:\BU\_STUDY\METCS689A1\FP\SalesFact.csv'

df\_sf = pd.read\_csv(sf\_file)

print(df\_sf.head())

print()

# load ProductPerformanceFact.csv

print("======================= Load ProductPerformanceFact.csv =======================")

ppf\_file = 'G:\BU\_STUDY\METCS689A1\FP\ProductPerformanceFact.csv'

df\_ppf = pd.read\_csv(ppf\_file)

print(df\_ppf.head())

print()

# load CustomerInteractionFact.csv

print("======================= Load CustomerInteractionFact.csv =======================")

CustomerInteractionFact\_file = 'G:\BU\_STUDY\METCS689A1\FP\CustomerInteractionFact.csv'

df\_CustomerInteractionFact = pd.read\_csv(CustomerInteractionFact\_file)

print(df\_CustomerInteractionFact.head())

print()

# load CustomerPurchaseBehaviorFact.csv

print("======================= Load CustomerPurchaseBehaviorFact.csv =======================")

cpbf\_file = 'G:\BU\_STUDY\METCS689A1\FP\CustomerPurchaseBehaviorFact.csv'

df\_cpbf = pd.read\_csv(cpbf\_file, keep\_default\_na=False)

print(df\_cpbf.head())

print()

### Python: Transformation

# Transformation for Customer.csv

print("--------------------------------------------- Transformation for Customer.csv --------------------------------------------")

age\_aggregates = df\_cust['age'].describe()

print(age\_aggregates)

df\_cust=df\_cust.drop\_duplicates()

print(df\_cust.head())

# Transformation for Location.csv

print("--------------------------------------------- Transformation for Location.csv --------------------------------------------")

df\_Location=df\_Loc.drop\_duplicates()

print(df\_Location.head())

# Transformation for time\_monthly.csv

print("--------------------------------------------- Transformation for time\_monthly.csv --------------------------------------------")

df\_tm=df\_tm.drop\_duplicates()

print(df\_tm.head())

# Transformation for time\_daily.csv

print("--------------------------------------------- Transformation for time\_daily.csv --------------------------------------------")

df\_td=df\_td.drop\_duplicates()

print(df\_td.head())

# Transformation for ProductPerformanceFact.csv

print("--------------------------------------------- Transformation for SalesFact.csv --------------------------------------------")

sf\_aggregates\_qtrorder = df\_sf['qty\_ordered'].describe()

print(sf\_aggregates\_qtrorder)

print()

sf\_aggregates\_p = df\_sf['price'].describe()

print(sf\_aggregates\_p)

print()

sf\_aggregates\_value = df\_sf['value'].describe()

print(sf\_aggregates\_value)

print()

sf\_aggregates\_dm = df\_sf['discount\_amount'].describe()

print(sf\_aggregates\_dm)

print()

sf\_aggregates\_total = df\_sf['total'].describe()

print(sf\_aggregates\_total)

# Transformation for ProductPerformanceFact.csv

print("--------------------------------------------- Transformation for ProductPerformanceFact.csv --------------------------------------------")

ppf\_aggregates\_qtrorder = df\_ppf['qty\_ordered'].describe()

print(ppf\_aggregates\_qtrorder)

print()

ppf\_aggregates\_totalRev = df\_ppf['Revenue\_Total'].describe()

print(ppf\_aggregates\_totalRev)

# Transformation for CustomerInteractionFact.csv

print("--------------------------------------------- Transformation for CustomerInteractionFact.csv --------------------------------------------")

CustomerInteractionFact\_aggregates\_ts = df\_CustomerInteractionFact['Time\_Spent'].describe()

print(CustomerInteractionFact\_aggregates\_ts)

print()

CustomerInteractionFact\_aggregates\_NumPurchase = df\_CustomerInteractionFact['N\_Purchases'].describe()

print(CustomerInteractionFact\_aggregates\_NumPurchase)

### 

### Python: Load data into Database

import pyodbc as db

conn = db.connect('Driver={SQL Server};'

'Server=XUYUHAN;'

'Server=SQLServer-PC;'

'Database=689fp;'

'Trusted\_Connection=yes;'

'Connection Timeout=3000')

cursor = conn.cursor()

print("...................................... Load data into Table Customer ......................................")

insert\_query = """

INSERT INTO Customer (Cust\_id,Name\_Prefix,First\_Name,Middle\_Initial,Last\_Name,

Gender,age,full\_name,E\_mail,Customer\_Since,

SSN,Phone\_No,PlaceName,County,City,

State,Zip,Region,User\_Name)

VALUES (?,?,?,?,?,

?,?,?,?,?,

?,?,?,?,?,

?,?,?,?)

"""

for index, row in df\_cust.iterrows():

Key = index + 1

cursor.execute(insert\_query, row['cust\_id'],row['Name\_Prefix'],row['First\_Name'],row['Middle\_Initial'],row['Last\_Name'],

row['Gender'],row['age'],row['full\_name'],row['E\_mail'],row['Customer\_Since'],

row['SSN'],row['Phone\_No'],row['PlaceName'],row['County'],row['City'],

row['State'],row['Zip'],row['Region'],row['User\_Name'])

conn.commit()

print("...................................... Load data into Table Time\_Monthly ......................................")

insert\_query1 = """

INSERT INTO Time\_Monthly (MonthKey,year,month)

VALUES (?,?,?)

"""

for index, row in df\_tm.iterrows():

Key = index + 1

cursor.execute(insert\_query1, row['MonthKey'],row['year'],row['month'])

conn.commit()

print("...................................... Load data into Table Time\_daily ......................................")

insert\_query2 = """

INSERT INTO Time\_daily (DateKey,Year,Quarter,Month,Day)

VALUES (?,?,?,?,?)

"""

for index, row in df\_td.iterrows():

Key = index + 1

cursor.execute(insert\_query2, row['DateKey'],row['Year'],row['Quarter'],row['Month'],row['Day'])

conn.commit()

print("...................................... Load data into Table Product ......................................")

insert\_query3 = """

INSERT INTO Product (Product\_id,SKU,Description,Category)

VALUES (?,?,?,?)

"""

for index, row in df\_product.iterrows():

Key = index + 1

cursor.execute(insert\_query3, row['Product\_id'],row['sku'],row['Description'],row['category'])

conn.commit()

print("...................................... Load data into Table OrderDim ......................................")

insert\_query4 = """

INSERT INTO OrderDim (Oid,OrderID,InvoiceNo,Status,LastUpdated)

VALUES (?,?,?,?,?)

"""

for index, row in df\_od.iterrows():

Key = index + 1

cursor.execute(insert\_query4, row['Oid'], row['OrderID'],row['InvoiceNo'],row['status'],row['LastUpdated'])

conn.commit()

print("...................................... Load data into Table Location ......................................")

insert\_query5 = """

INSERT INTO Location (LocationID,Place\_Name,County,City,State,

PreviousCity,PreviousState,Zip,Region,Country)

VALUES (?,?,?,?,?,

?,?,?,?,?)

"""

for index, row in df\_Location.iterrows():

Key = index + 1

cursor.execute(insert\_query5, row['LocationID'],row['Place Name'],row['County'],row['City'],row['State'],

row['PreviousCity'],row['PreviousState'],row['Zip'],row['Region'],row['Country'])

conn.commit()

print("...................................... Load data into Table ProductPerformanceFact ......................................")

insert\_query6 = """

INSERT INTO ProductPerformanceFact (PerformanceID,OrderID,Product\_id,MonthKey,qty\_ordered,Revenue\_Total)

VALUES (?,?,?,?,?,?)

"""

for index, row in df\_ppf.iterrows():

Key = index + 1

cursor.execute(insert\_query6, row['PerformanceID'],row['OrderID'],row['Product\_id'],row['MonthKey'],row['qty\_ordered'],row['Revenue\_Total'])

conn.commit()

print("...................................... Load data into Table SalesFact ......................................")

insert\_query7 = """

INSERT INTO SalesFact (SalesID,OrderID,cust\_id,Product\_id,LocationID,

DateKey,qty\_ordered,price,value,discount\_amount,total)

VALUES (?,?,?,?,?,

?,?,?,?,?,?)

"""

for index, row in df\_sf.iterrows():

Key = index + 1

cursor.execute(insert\_query7, row['SalesID'],row['OrderID'],row['cust\_id'],row['Product\_id'],row['LocationID'],

row['DateKey'],row['qty\_ordered'],row['price'],row['value'],row['discount\_amount'],

row['total'])

conn.commit()

print("...................................... Load data into Table CustomerInteractionFact ......................................")

insert\_query8 = """

INSERT INTO CustomerInteractionFact (InteractionID,OrderID,cust\_id,MonthKey,TimeSpent,N\_Purchases)

VALUES (?,?,?,?,?,?)

"""

for index, row in df\_CustomerInteractionFact.iterrows():

Key = index + 1

cursor.execute(insert\_query8, int(row['InteractionID']),int(row['OrderID']),int(row['cust\_id']),int(row['MonthKey']),int(row['Time\_Spent']),int(row['N\_Purchases']))

conn.commit()

print("...................................... Load data into Table CustomerPurchaseBehaviorFact ......................................")

insert\_query9 = """

INSERT INTO CustomerPurchaseBehaviorFact (BehaviorFactID,cust\_id,Product\_id,DateKey,Revenue\_Total,N\_Purchases,Purchase\_VALUE,Pay\_Method)

VALUES (?,?,?,?,

?,?,?,?)

"""

for index, row in df\_cpbf.iterrows():

Key = index + 1

cursor.execute(insert\_query9, row['BehaviorFactID'],row['cust\_id'],row['Product\_id'],row['DateKey'],row['Revenue\_Total'],row['N\_Purchases'],

row['Purchase\_VALUE'],row['Pay\_Method'])

conn.commit()

cursor.close()

conn.close()

### SQL: Maintain SCD 3

CREATE PROCEDURE UpdateLocationInfo

@Cust\_id VARCHAR(50),

@NewCity VARCHAR(50),

@NewState VARCHAR(50)

AS

BEGIN

-- Start the transaction

BEGIN TRANSACTION;

-- Declare a table variable

DECLARE @ToUpdate TABLE (LocationID INT);

-- Populate the table variable with the IDs to update

INSERT INTO @ToUpdate (LocationID)

SELECT loc.LocationID

FROM Location loc

INNER JOIN SalesFact sf ON loc.LocationID = sf.LocationID

WHERE sf.Cust\_id = @Cust\_id;

-- Perform the update using the table variable

UPDATE loc

SET

loc.PreviousCity = loc.City,

loc.PreviousState = loc.State,

loc.City = @NewCity,

loc.State = @NewState

FROM Location loc

INNER JOIN @ToUpdate ut ON loc.LocationID = ut.LocationID;

-- Commit the transaction

COMMIT;

### SQL: Maintain SCD 2

CREATE TABLE CustomerHistory (

Chid int PRIMARY KEY IDENTITY(1,1),

Cust\_id NVARCHAR(4000),

E\_mail NVARCHAR(4000),

Customer\_Since NVARCHAR(4000),

SSN NVARCHAR(4000),

Phone\_No NVARCHAR(4000),

PlaceName NVARCHAR(4000),

County NVARCHAR(4000),

City NVARCHAR(4000),

State NVARCHAR(4000),

Zip NVARCHAR(4000),

Region NVARCHAR(4000),

User\_Name NVARCHAR(4000),

newE\_mail NVARCHAR(4000),

newCustomer\_Since NVARCHAR(4000),

newSSN NVARCHAR(4000),

newPhone\_No NVARCHAR(4000),

newPlaceName NVARCHAR(4000),

newCounty NVARCHAR(4000),

newCity NVARCHAR(4000),

newState NVARCHAR(4000),

newZip NVARCHAR(4000),

newRegion NVARCHAR(4000),

newUser\_Name NVARCHAR(4000),

UpdateTime DATETIME

);

CREATE PROCEDURE UpdateAndInsertCustomerAddress

@Cust\_id INT,

@NewCity VARCHAR(50),

@NewState VARCHAR(50),

@NewZip VARCHAR(10)

AS BEGIN

-- Start the transaction

BEGIN TRANSACTION;

IF EXISTS (SELECT Cust\_id FROM Customer WHERE Cust\_id = @Cust\_id AND isCurrent = 1) BEGIN

IF (SELECT City FROM Customer WHERE Cust\_id = @Cust\_id AND isCurrent = 1) != @NewCity BEGIN

IF(SELECT City FROM Customer WHERE Cust\_id = @Cust\_id AND isCurrent = 1) != @NewState BEGIN

INSERT INTO CustomerHistory (

Cust\_id, City, State, Zip,

newCity, newState, newZip, UpdateTime

)

VALUES(

(SELECT Cust\_id FROM Customer WHERE Cust\_id = @Cust\_id AND isCurrent = 1),

(SELECT City FROM Customer WHERE Cust\_id = @Cust\_id AND isCurrent = 1),

(SELECT State FROM Customer WHERE Cust\_id = @Cust\_id AND isCurrent = 1),

(SELECT Zip FROM Customer WHERE Cust\_id = @Cust\_id AND isCurrent = 1),

@NewCity, @NewState,@NewZip, GETDATE()

)

-- Update the existing customer record

UPDATE Customer SET LastUpdated = GETDATE(), isCurrent = 2 WHERE Cust\_id = @Cust\_id AND isCurrent = 1;

-- Insert a new record for the customer with the updated information

INSERT INTO Customer (

Cust\_id, Name\_Prefix, First\_Name, Middle\_Initial, Last\_Name,

Gender, age, full\_name, E\_mail, Customer\_Since,

SSN, Phone\_No, PlaceName, County, City,

State, Zip, Region, User\_Name, LastUpdated, isCurrent

)

SELECT

Cust\_id, Name\_Prefix, First\_Name, Middle\_Initial, Last\_Name,

Gender, age, full\_name, E\_mail, Customer\_Since,

SSN, Phone\_No, PlaceName, County, @NewCity,

@NewState, @NewZip, Region, User\_Name, GETDATE(), 1

FROM Customer WHERE Cust\_id = @Cust\_id AND isCurrent = 2;

Update Customer SET isCurrent = 0 WHERE Cust\_id = @Cust\_id AND isCurrent = 2;

END;

ELSE BEGIN

INSERT INTO CustomerHistory (

Cust\_id, City, Zip,

newCity, newZip, UpdateTime

)

VALUES(

(SELECT Cust\_id FROM Customer WHERE Cust\_id = @Cust\_id AND isCurrent = 1),

(SELECT City FROM Customer WHERE Cust\_id = @Cust\_id AND isCurrent = 1),

(SELECT Zip FROM Customer WHERE Cust\_id = @Cust\_id AND isCurrent = 1),

@NewCity,@NewZip, GETDATE()

)

-- Update the existing customer record

UPDATE Customer SET LastUpdated = GETDATE(), isCurrent = 2 WHERE Cust\_id = @Cust\_id AND isCurrent = 1;

-- Insert a new record for the customer with the updated information

INSERT INTO Customer (

Cust\_id, Name\_Prefix, First\_Name, Middle\_Initial, Last\_Name,

Gender, age, full\_name, E\_mail, Customer\_Since,

SSN, Phone\_No, PlaceName, County, City,

State, Zip, Region, User\_Name, LastUpdated, isCurrent

)

SELECT

Cust\_id, Name\_Prefix, First\_Name, Middle\_Initial, Last\_Name,

Gender, age, full\_name, E\_mail, Customer\_Since,

SSN, Phone\_No, PlaceName, County, @NewCity,

State, @NewZip, Region, User\_Name, GETDATE(), 1

FROM Customer WHERE Cust\_id = @Cust\_id AND isCurrent = 2;

Update Customer SET isCurrent = 0 WHERE Cust\_id = @Cust\_id AND isCurrent = 2;

END;

END;

ELSE BEGIN

PRINT 'Your input Address is same as the Address in the Database.'

END;

END;

ELSE BEGIN

PRINT '@Customer ID Input DOES NOT EXIST IN DATABASE, Check again'

END;

-- Commit the transaction

COMMIT;

END;

EXEC UpdateAndInsertCustomerAddress @Cust\_id ='42485', @NewCity = 'Ames', @NewState = 'IA', @NewZip = '50011';

EXEC UpdateAndInsertCustomerAddress @Cust\_id ='42485', @NewCity = 'Boston', @NewState = 'MA', @NewZip = '02114';

SELECT \* FROM Customer WHERE Cust\_id = '42485'

SELECT \* FROM CustomerHistory WHERE Cust\_id = '42485'

### SQL: Maintain SCD 1

CREATE PROCEDURE UpdateProductInfo

@Product\_id NVARCHAR(4000),

@SKU NVARCHAR(4000),

@Description NVARCHAR(4000),

@Category NVARCHAR(4000)

AS BEGIN

BEGIN TRANSACTION

IF EXISTS (SELECT Product\_id FROM Product WHERE Product\_id = @Product\_id) BEGIN

IF (SELECT SKU FROM Product WHERE Product\_id = @Product\_id) != @SKU

UPDATE Product Set SKU = @SKU WHERE Product\_id = @Product\_id;

IF (SELECT Description FROM Product WHERE Product\_id = @Product\_id) != @Description

UPDATE Product Set Description = @Description WHERE Product\_id = @Product\_id;

IF (SELECT Description FROM Product WHERE Product\_id = @Product\_id) != @Description

UPDATE Product Set Category = @Category WHERE Product\_id = @Product\_id;

END;

ELSE BEGIN

INSERT INTO Product(Product\_id,SKU,Description,Category)

VALUES(@Product\_id,@SKU,@Description,@Category)

END;

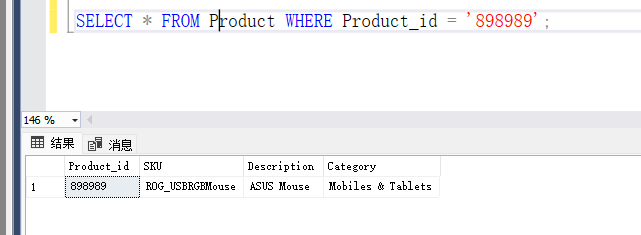
COMMIT;

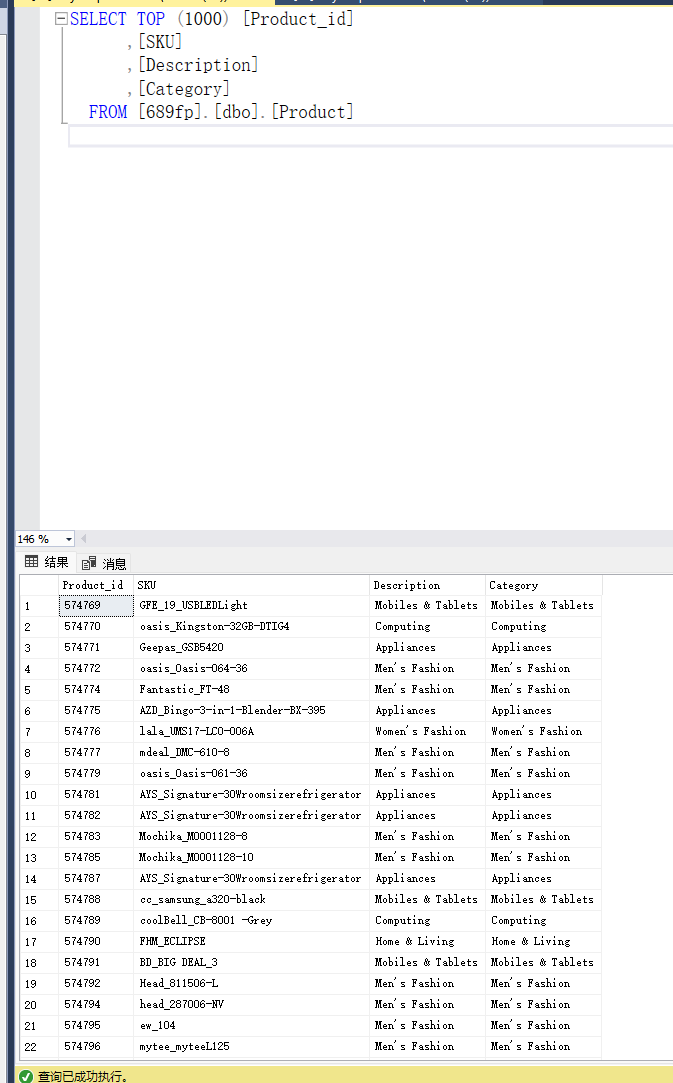
END;

DROP PROCEDURE UpdateProductInfo;

EXEC UpdateProductInfo @Product\_id = '574769', @SKU = 'GFE\_19\_USBLEDLight', @Description = 'Mobiles & Tablets',@Category = 'Mobiles & Tablets';

EXEC UpdateProductInfo @Product\_id = '898989', @SKU = 'ROG\_USBRGBMouse', @Description = 'ASUS Mouse',@Category = 'ELECTRONICS';





# **Business Questions**

1:Which product categories are consistently top performers across different international online retail platforms?

2: How does customer demographic information influence purchasing behavior in different e-commerce environments?

3: What are the most effective pricing strategies for maximizing sales volume and profitability in diverse online retail markets?

4: How can inventory turnover rates be optimized based on historical sales data across various product lines and geographical markets?

5: What is the effect of discount strategies on the average order value across different customer segments?

6: Which months or quarters demonstrate the highest sales volumes, and how do these trends vary across different product categories or geographical locations?

7: Can we identify any correlations between the amount of time spent by customers on the platform (TimeSpent) and the number of purchases they make, and how does this relationship differ across various product categories?

8: Are there specific product categories that are frequently purchased together, suggesting a potential for bundling strategies or cross-promotional marketing campaigns?

9: How do sales and customer interaction metrics such as 'TimeSpent' and 'N\_Purchases' vary across different cities or states, and what could this indicate about regional market preferences or potential areas for market expansion?

10: Which customer age groups are most responsive to our product offerings in terms of revenue generated, and how does this differ by location and time of year?

# **Answers**

### For Question 1(SQL)

(Use views and aggregates)

WITH InternationalSales AS (

SELECT

p.Category,

s.LocationID,

SUM(s.Total) AS CategorySales,

RANK() OVER (PARTITION BY s.LocationID ORDER BY SUM(s.Total) DESC) AS SalesRank

FROM Product p

JOIN SalesFact s ON p.Product\_id = s.Product\_id

GROUP BY p.Category, s.LocationID

)

SELECT

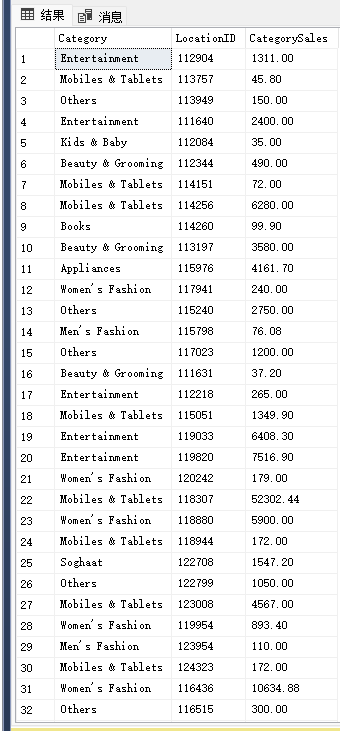
Category,

LocationID,

CategorySales

FROM InternationalSales

WHERE SalesRank = 1;



**Answer for Q1:**  
the query results indicate a varied consumer preference across different locations. Here’s a summary of the key insights:

**Category Performance:** The 'Entertainment' and 'Mobiles & Tablets' categories show high sales across several locations, suggesting these are universally popular categories.

**High Sales in Specific Locations:** Some categories like 'Women's Fashion' and 'Appliances' show exceptionally high sales in certain locations, which might be due to localized demand or successful promotions in those areas.

**Market Diversity:** The variety of categories with significant sales, such as 'Beauty & Grooming', 'Books', and 'Kids & Baby', indicates a market with diverse consumer interests.

**Opportunities for Retailers:** The data suggests potential for retailers to employ cross-selling techniques and to adjust their product offerings based on location-specific demand.

**Price Sensitivity and Willingness to Spend:** There is evidence of price sensitivity in some categories, while others show a willingness among consumers to spend more, pointing to opportunities to tailor pricing strategies accordingly.

**Market Saturation and Expansion:** Repeated appearance of categories like 'Mobiles & Tablets' across various locations with differing sales volumes suggests some markets may be saturated, while others could be ripe for expansion.

This information can aid retailers in making strategic decisions regarding inventory management, marketing, and customization of offerings to optimize sales and meet consumer demands effectively.

### For Question 2(SQL)

SELECT

Customer.Age,

Customer.Gender,

AVG(CustomerPurchaseBehaviorFact.Purchase\_VALUE) AS AvgPurchaseValue

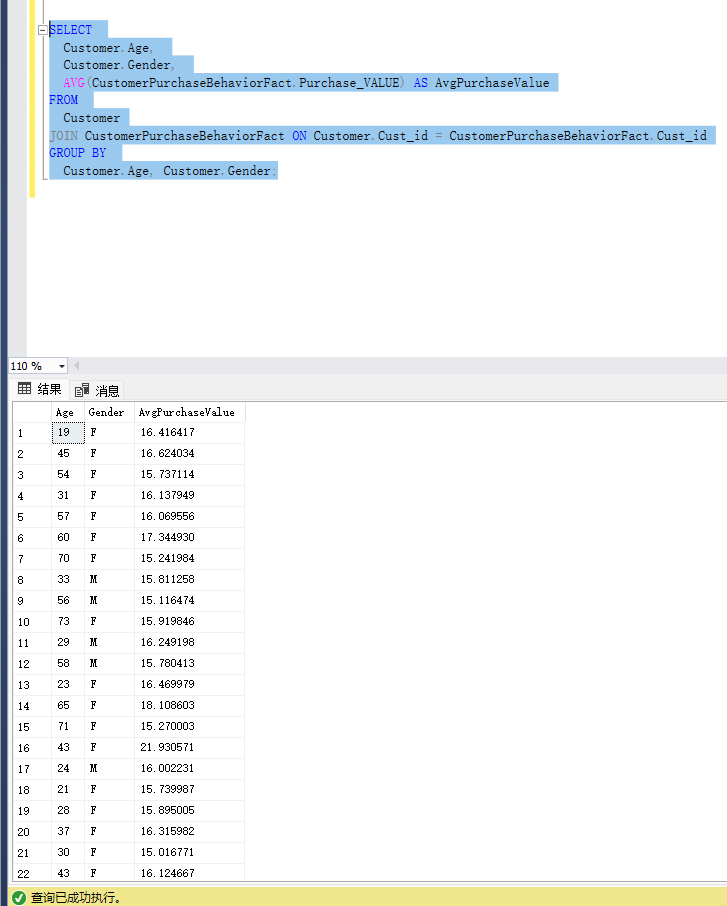
FROM

Customer

JOIN CustomerPurchaseBehaviorFact ON Customer.Cust\_id = CustomerPurchaseBehaviorFact.Cust\_id

GROUP BY

Customer.Age, Customer.Gender;



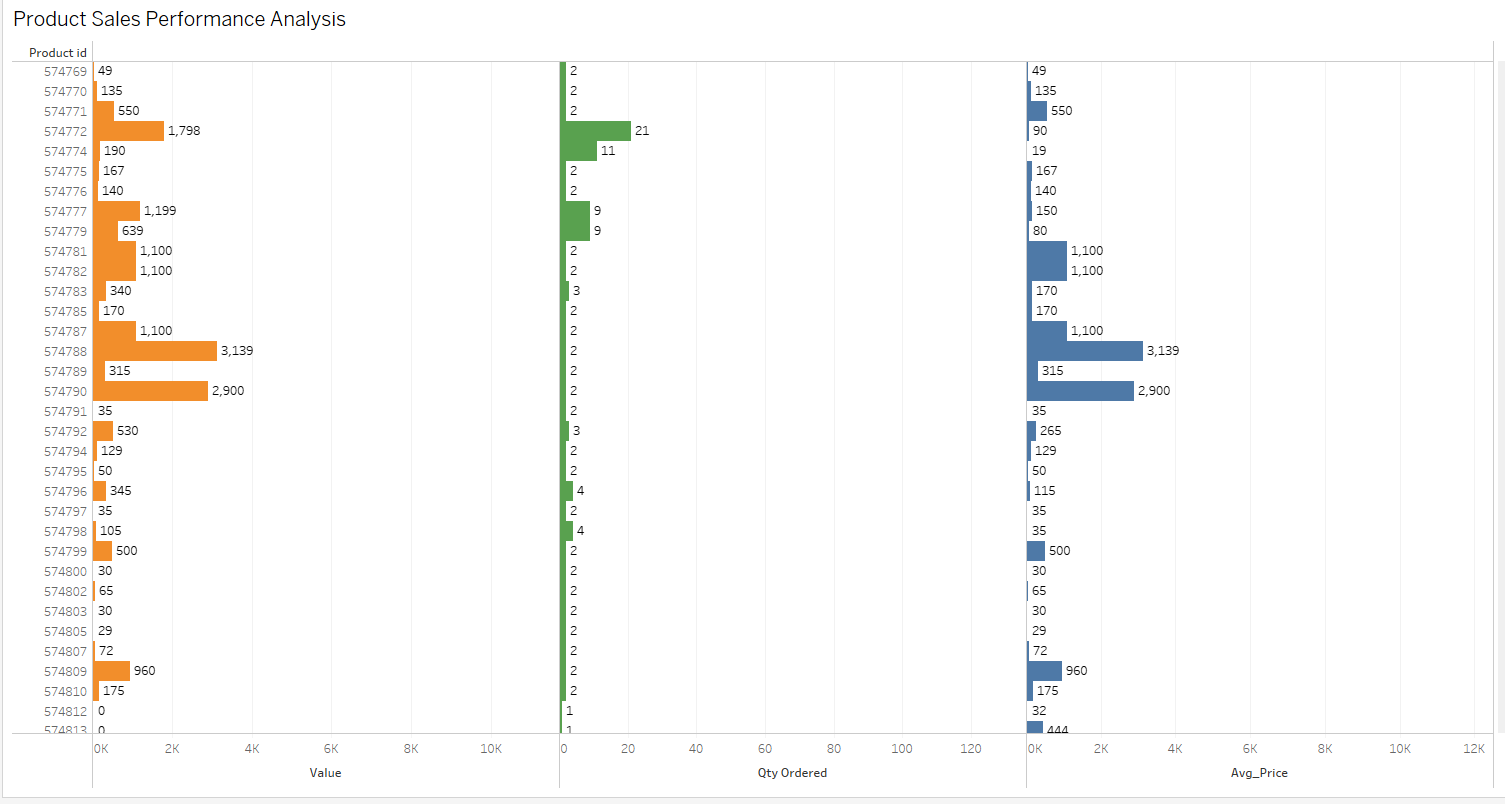
**Answer:**

The SQL query output shows the average purchase value broken down by customer age and gender. Based on the results, here are some observations:

* The purchasing value does not show a consistent trend with age. Customers of different ages seem to have varying average purchase values without a clear pattern indicating that age might not be a primary factor in determining the average purchase value.
* Female customers in the age groups of 45 and 65 have higher average purchase values compared to other age groups, with values of approximately 16.62 and 18.06 respectively.
* Male customers aged 33 have a notably lower average purchase value of around 15.11, whereas male customers aged 58 show a higher average purchase value close to 15.78.
* The dataset does not show a consistent difference in purchasing behavior between male and female customers across different age groups. For instance, a 19-year-old female has an average purchase value of approximately 16.46, while a 56-year-old female has a higher average purchase value of approximately 17.34.
* It's important to note that these observations are based solely on the data provided and do not account for other variables that might affect purchasing behavior. Additionally, the sample size for each age and gender category is not provided, which could significantly influence the interpretation of these results.

The analysis could be enriched by considering the number of transactions per age and gender group, which would give more context to the average purchase values observed.

### For Question 3(Tableau)



**Answer:**

Based on the Tableau output graph, which presents data for 'Value', 'Quantity Ordered', and 'Average Price' across different product IDs, here are some sentences that could describe the findings:

* Products with higher sales value do not necessarily have a higher quantity ordered, suggesting that some high-value sales come from items with a higher price point rather than a higher volume of sales.
* There is a product highlighted in orange with a particularly high sales value and a significant quantity ordered, indicating a popular item or one that may have been part of a promotional sale.
* Another product, also highlighted in orange, shows a high average price but a lower quantity ordered, which could suggest it is a premium or luxury item with less frequent purchases.
* The majority of products have a modest quantity ordered and corresponding sales value, which might imply a steady but average demand in the market.
* The product with the highest average price does not match the product with the highest sales value, suggesting that the most expensive items are not always the top revenue generators.

### For Question 4(SQL)

SELECT

p.Product\_id,

p.Description,

AVG(s.qty\_ordered) AS AvgQtySold,

COUNT(DISTINCT s.DateKey) AS SellingDays,

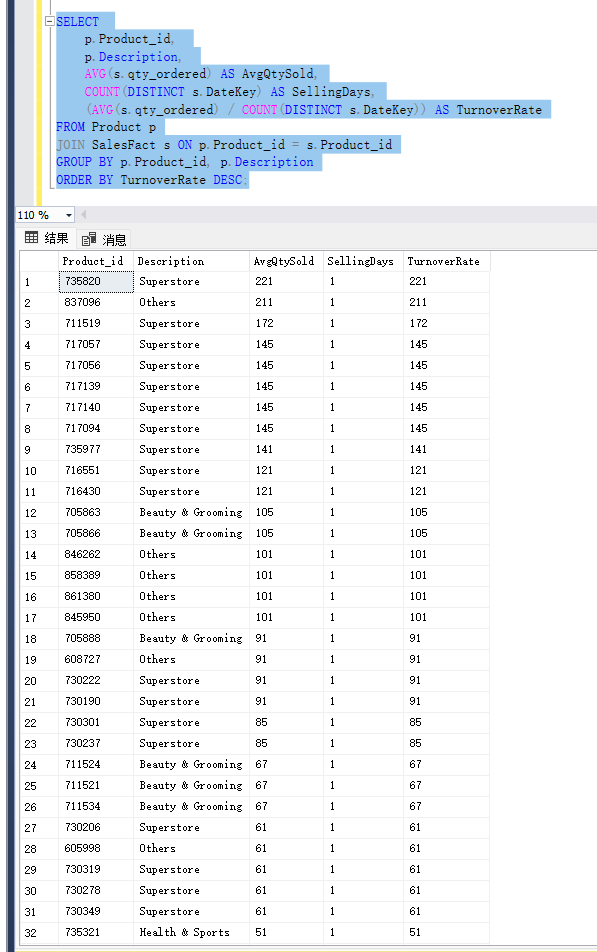
(AVG(s.qty\_ordered) / COUNT(DISTINCT s.DateKey)) AS TurnoverRate

FROM Product p

JOIN SalesFact s ON p.Product\_id = s.Product\_id

GROUP BY p.Product\_id, p.Description

ORDER BY TurnoverRate DESC;



**Answer for Q4:**

we can make the following observations about inventory turnover rates:

**High Turnover Rate Products:** Products classified under 'Superstore' and 'Others' show the highest turnover rates, with the top product achieving a turnover rate of 221. This indicates these products sell quickly and might require more frequent restocking.

**Turnover Rate Consistency:** Several products, especially in the 'Superstore' category, have a consistent turnover rate of 145 or 91, suggesting steady sales. It’s notable that these products have a selling period of 1 day, which means they were likely part of a one-day sale or a quick turnover event.

**Lower Turnover in Health & Sports:** The 'Health & Sports' category shows the lowest turnover rate at 51, which could indicate slower sales in this category or possibly a niche product that appeals to a specific market segment.

**Beauty & Grooming Category:** This category shows a range of turnover rates from 67 to 105, suggesting variability in sales velocity which could be influenced by factors like product type, promotions, or consumer trends.

I**mplications for Inventory Management:** Products with high turnover rates may require careful inventory management to avoid stockouts, while products with lower turnover rates may need strategies to increase sales or to review stock levels to prevent overstocking.

### For Question 5(SQL)

SELECT

C.Gender,

C.Age,

AVG(SF.Total) AS AverageOrderValue,

AVG(SF.Discount\_Amount) AS AverageDiscount

FROM

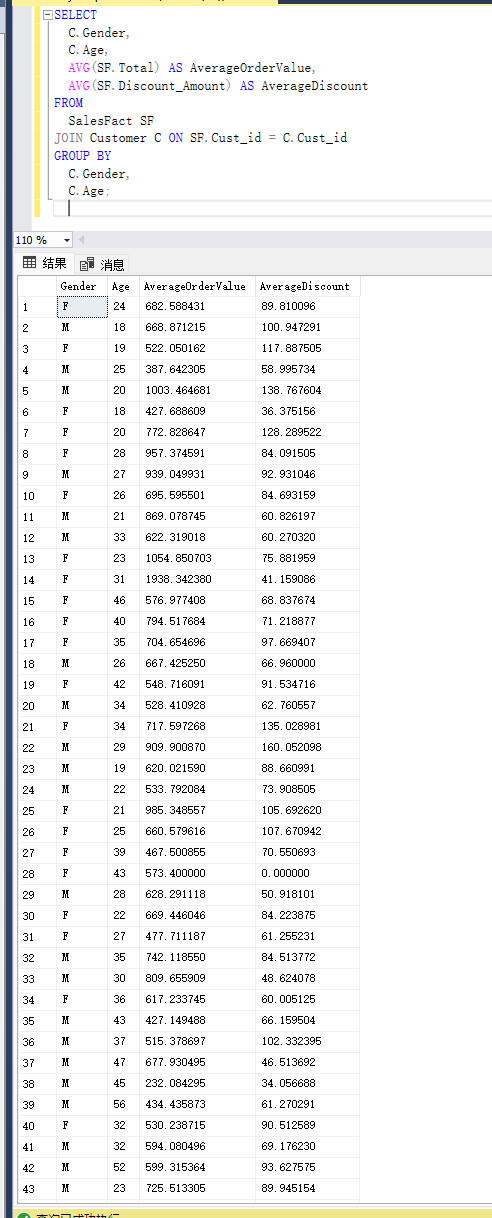
SalesFact SF

JOIN Customer C ON SF.Cust\_id = C.Cust\_id

GROUP BY

C.Gender,

C.Age;



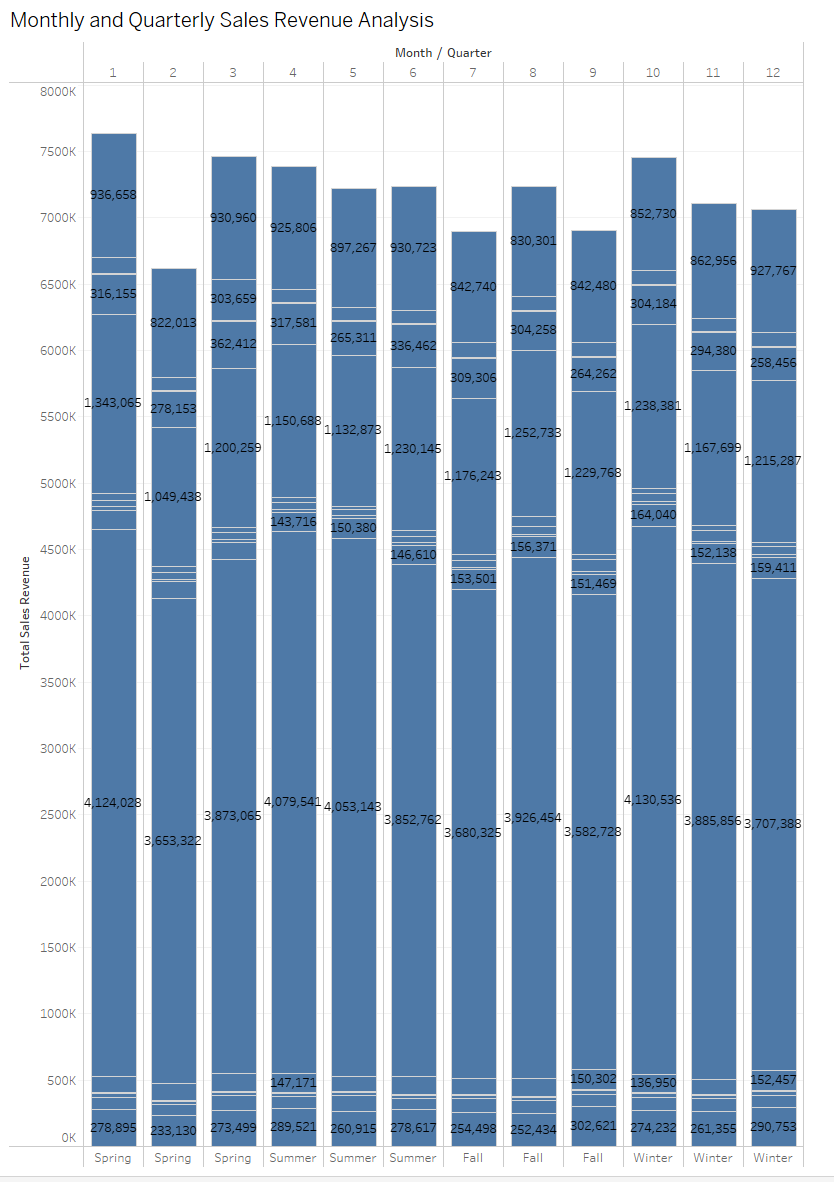
**Answer:**

**The SQL query output suggests an analysis of the average order value and average discount across various customer segments, categorized by gender and age. Here are some insights based on the data provided:**

* **The average order value varies significantly across different age and gender segments, indicating that purchasing power or preferences may differ with these demographics.**
* **For instance, females aged 31 have the highest average order value at approximately 1,938.34, which is substantially higher than other segments.**
* **The average discount received by customers also varies, with males aged 18 receiving the highest average discount of around 100.95.**
* **Interestingly, the segment with the highest average order value (females aged 31) does not receive the highest average discount, suggesting that a higher discount does not necessarily correlate with a higher order value.**
* **It is also noted that some segments, such as females aged 43, have a zero average discount, yet they still make purchases, which could indicate a segment that is less sensitive to pricing strategies.**
* **Younger segments, like males aged 25, show lower average order values but higher average discounts, which may reflect a strategic focus on attracting younger customers through discounts.**
* **Overall, the data could suggest that while discounts might play a role in influencing customer purchasing behavior, they are not the sole factor driving the average order value, as evidenced by the lack of a consistent relationship between the average discount and order value across segments.**

**This analysis would be vital for informing targeted discount strategies and understanding the purchasing behavior across different demographic segments.**

### For Question 6(Tableau)

  
**Answer:**  
The chart appears to show total sales revenue broken down by month and quarter. From this visualization, we can discern the following:

**1.Quarterly Trends:** There is a noticeable trend that aligns with typical retail sales cycles. Sales seem to peak in Q4, which encompasses the holiday season, a common period for increased consumer spending due to events like Black Friday, Cyber Monday, and Christmas.

**2. Monthly Fluctuations:** Within each quarter, there are fluctuations. Some months show significantly higher revenue than others. For example, month 12, which likely represents December, shows a high spike in sales, while other months like month 6 and month 9 have comparatively lower sales.

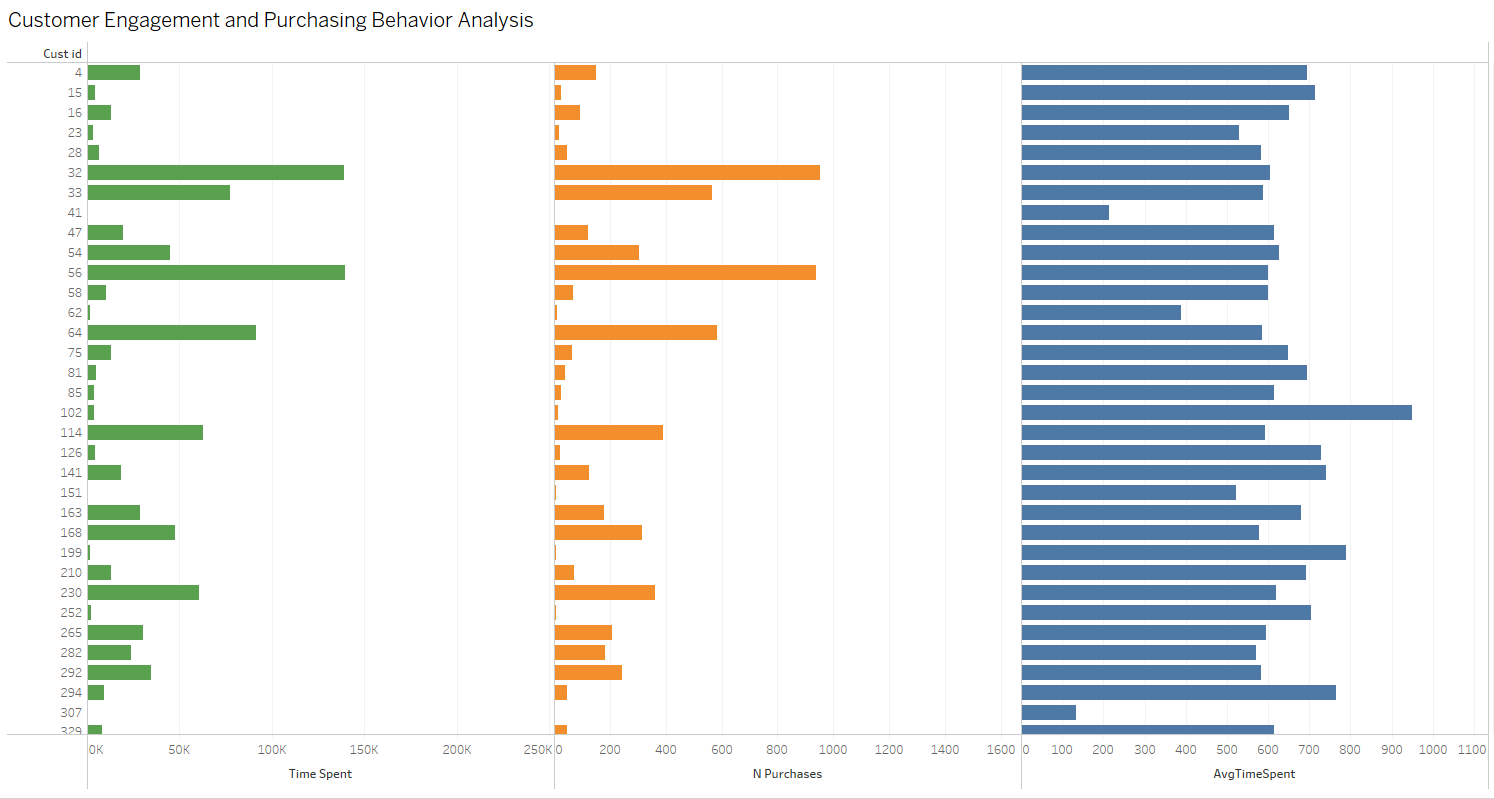
**3. Seasonal Analysis:** The chart provides a clear illustration of seasonality in sales. Spring shows a gradual increase in sales, which could be attributed to new inventory or seasonal promotions. Summer sales are relatively stable and consistent. Fall shows a slight uptick, which might be due to back-to-school shopping or early holiday promotions. Winter, and particularly the end of Q4, shows the highest sales revenue, which is consistent with the holiday shopping season.

**4. Revenue Range:** The revenue range across the months varies from as low as around 150k to as high as over 900k. This variance is crucial for planning inventory, staffing, and marketing throughout the year.

**5. Interpretation for Planning:** Businesses can use this data to plan for inventory stock-up ahead of the expected peaks, especially in Q4. It also helps in budget allocation for marketing activities to ensure they coincide with periods where consumers are more likely to make purchases.

**6. Potential Action Items:** For months with lower sales, strategies such as promotions, special events, or targeted marketing campaigns could be considered to boost revenue.

### For Question 7 (Tableau)



**Answer:**

**Based on the uploaded Tableau output graph, which shows customer IDs along with corresponding 'Time Spent', 'Number of Purchases', and 'Average Time Spent' on the platform, here are some sentences that could be used to describe the findings:**

* **The graph indicates that there is considerable variability in how much time customers spend on the platform and the number of purchases they make.**
* **Some customers with relatively lower time spent have a high number of purchases, suggesting efficiency in their shopping behavior or a possible indication of repeat purchases of familiar items.**
* **Conversely, a few customers show high time spent but fewer purchases, which could suggest more extensive browsing or comparison behavior before making a purchase decision.**
* **The 'Average Time Spent' section of the graph reveals that there are customers who spend significantly more time on average, which might correlate with more deliberate purchase decisions, potential engagement with content on the platform, or a preference for in-depth product research.**
* **The customer with the highest average time spent does not correspond to the highest number of purchases, indicating that time spent does not necessarily translate to a higher volume of transactions.**

**To accurately interpret this graph and draw more definitive conclusions, additional context about the customer journey and their interactions with the platform would be beneficial.**

### For Question 8(SQL)

SELECT

S1.Product\_id AS Product1,

S2.Product\_id AS Product2,

COUNT(\*) AS TimesPurchasedTogether

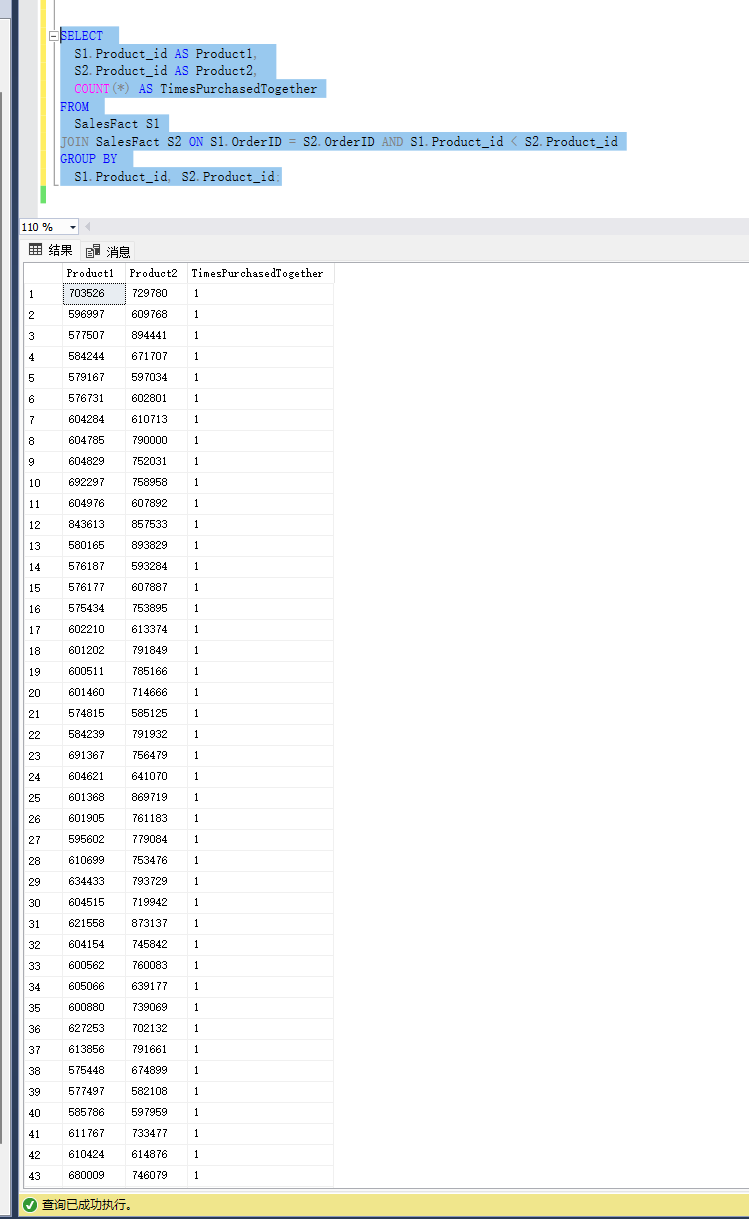
FROM

SalesFact S1

JOIN SalesFact S2 ON S1.OrderID = S2.OrderID AND S1.Product\_id < S2.Product\_id

GROUP BY

S1.Product\_id, S2.Product\_id;

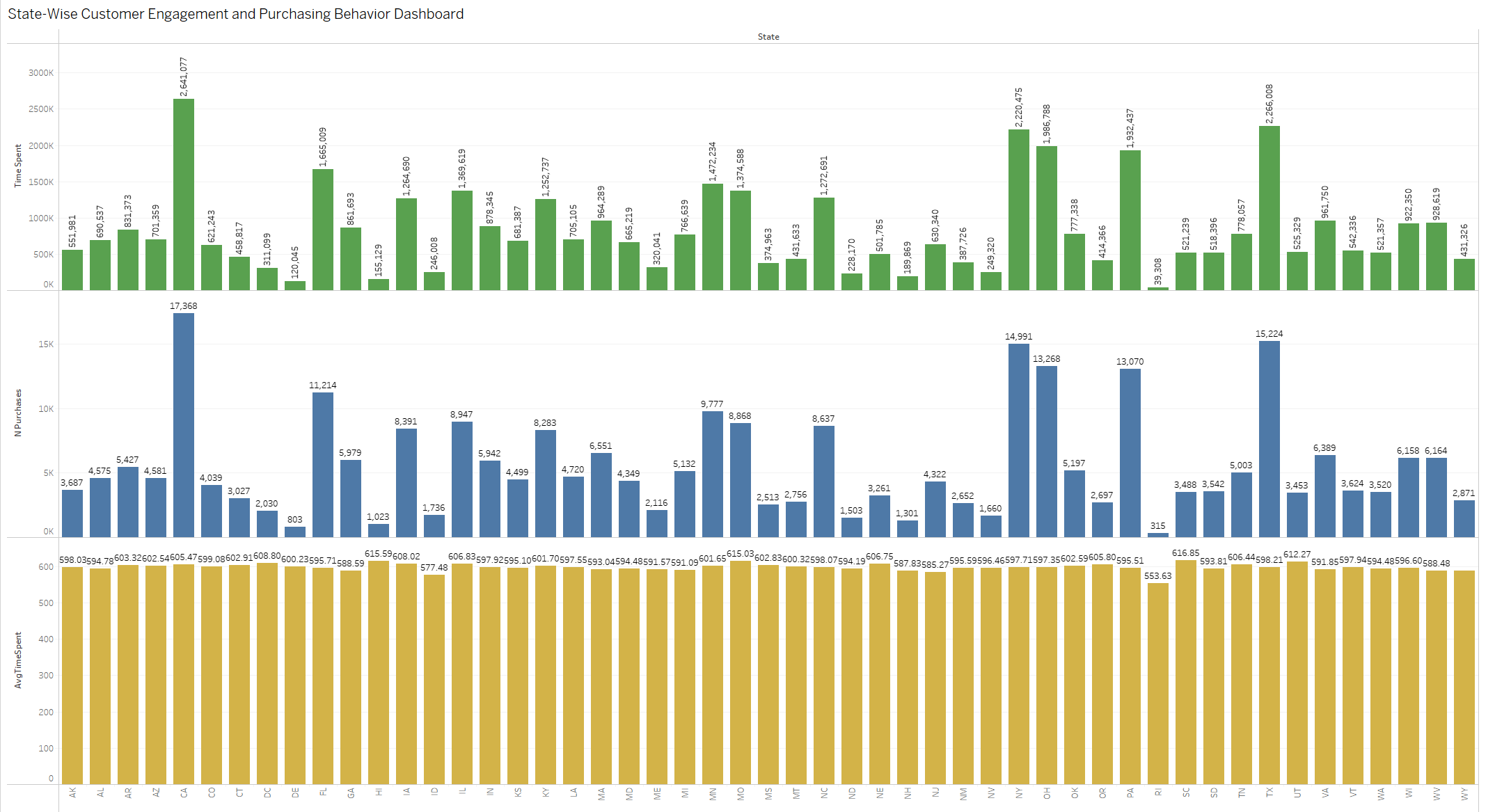


**Answer:**

**The output result from the SQL query shows pairs of product IDs and the count of times they were purchased together. Here are some sentences that describe the findings:**

* **Each pair of products listed in the result has been purchased together exactly once, as indicated by the 'TimesPurchasedTogether' count being** 1 **for all entries.**
* **This information can be used to identify product affinities, although the data suggests that there may not be a strong repeat pattern of products being purchased together since each combination has occurred only once.**
* **To find significant patterns for bundling strategies or cross-promotion, you might need to look for product pairs with a higher frequency of being purchased together.**
* **Given that all counts are** 1**, it may be necessary to run the query over a larger dataset or over a different time frame to capture more frequent combinations.**
* **This result set could serve as an initial step in identifying product relationships but further analysis with a more substantial dataset might be required to draw actionable insights for marketing strategies.**

### For Question 9(Tableau)



**Answer:  
First analysis, Top Chart (TimeSpent):** This chart seems to represent the total time spent by customers in each state. Some states show significantly higher total time spent, which could be due to a larger customer base, longer interaction times, or a combination of both.

**Middle Chart (N\_Purchases):** This chart likely displays the total number of purchases made by customers in each state. Similar to the time spent, there are noticeable variations, with some states having a higher number of purchases.

**Bottom Chart (AvgTimeSpent):** The bottom chart presumably represents the average time spent per interaction or purchase in each state. The distribution is more uniform compared to the total time spent, indicating that while the total time varies widely, the average time per interaction may not vary as significantly.

Then. Give the answer, There are clear regional differences in both the total time customers spend interacting (presumably with the website or service) and the total number of purchases.

States that show higher total time spent do not necessarily correspond to a higher number of purchases, suggesting that more time spent doesn't always translate to a higher transaction volume.

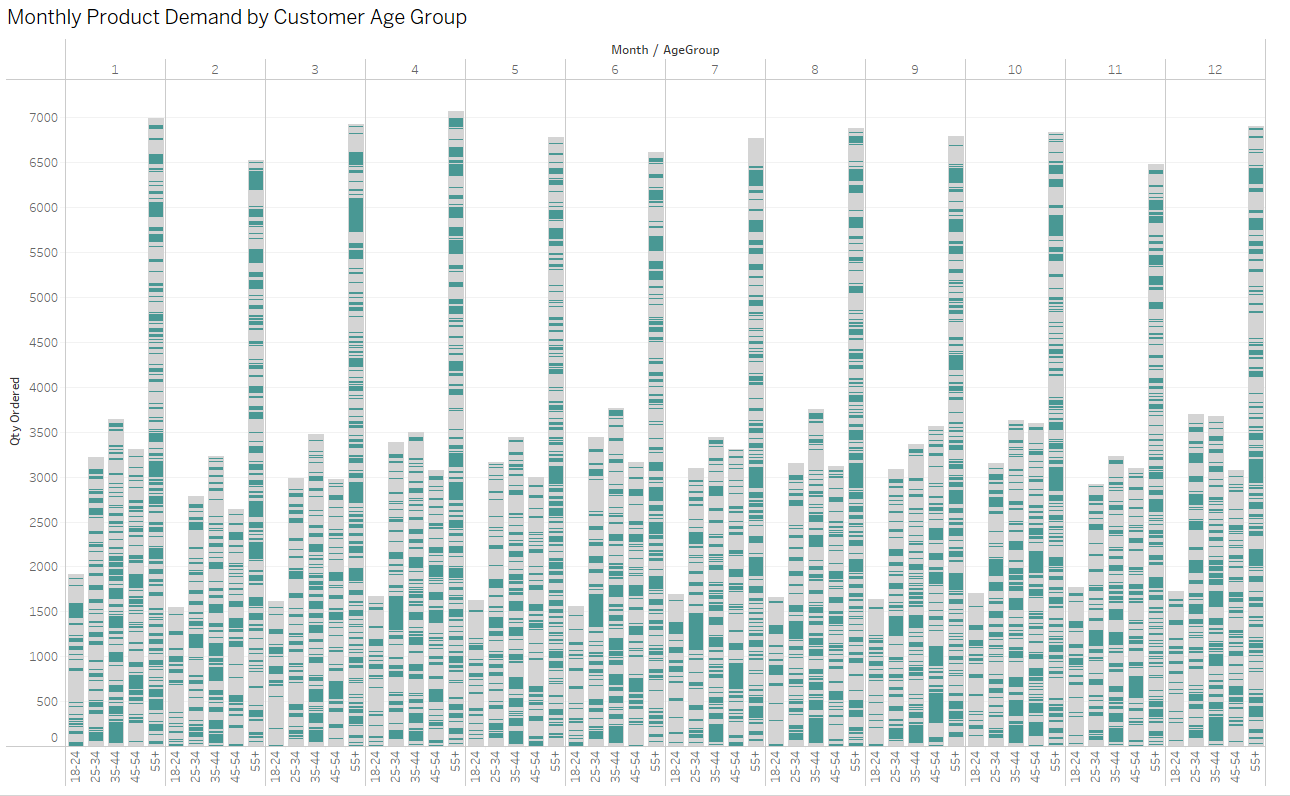
The average time spent is relatively consistent across states, which might suggest that while customer engagement levels in terms of time vary, the interaction quality or duration remains similar.

These insights can be used to inform regional marketing strategies, customer service resource allocation, and product distribution planning. For example, states with high time spent but lower purchase numbers might benefit from strategies aimed at converting browsing to buying.

The data indicates potential areas for market expansion or increased marketing efforts, especially in states with high interaction but relatively lower purchase numbers.

In summary, these charts provide a detailed breakdown of customer engagement metrics by state, which can be leveraged to optimize marketing and sales strategies in different regions.

### For Question 10 (Tableau)



**Answer:**

**The provided chart displays quantity ordered segmented by month and customer age groups, offering a visual representation of customer responsiveness to product offerings throughout the year. Here are some observations based on the chart:**

* **Customer age groups are differentiated by shades, with each age group having a consistent pattern across all months.**
* **The lighter shades, potentially representing younger age groups, appear to have higher quantities ordered in certain months, suggesting a possible increase in responsiveness or demand during those times.**
* **The bands of color seem to indicate that all age groups have a similar pattern of purchasing throughout the year, with spikes in certain months that could correspond to peak sales periods such as holidays or special promotions.**
* **There is a noticeable increase in the quantity ordered for most age groups around the same months, which could indicate seasonal trends affecting purchasing behavior across the board.**
* **The consistent patterns across different age groups suggest that while the quantity of products ordered might vary by age group, the times of the year when purchases increase or decrease are relatively uniform across demographics.**

**Given that the visualization does not include specific revenue data or location details, it is not possible to directly comment on revenue generation or regional differences. However, if the quantity ordered can be used as a proxy for sales performance, then the chart suggests that all age groups respond similarly to seasonal factors, with some age groups ordering in larger quantities.**

# **Data Source:**

**Data Source 1:**

URL: <https://www.kaggle.com/datasets/ytgangster/online-sales-in-usa>

Columns: 36, Rows: 286393

**Data Source 2:**

URL: <https://www.kaggle.com/datasets/ishanshrivastava28/tata-online-retail-dataset>

Columns: 8, Rows: 541909

**Data Source 3:**

URL: https://www.kaggle.com/datasets/onlineretailshop/online-shop-customer-sales-data/data

Columns: 12, Rows: 65797

The datasets "Online Sales in US," "TATA Online Retail Dataset," and "Online Shop Customer Sales Data" collectively offer a rich source of information for analyzing consumer purchasing patterns, product performance, and pricing strategies across different markets. They likely contain data on sales, customer demographics, product categories, and pricing, which are crucial for understanding market trends and consumer behavior. However, integrating these datasets may present challenges such as ensuring data consistency across different sources, handling missing or incomplete data, aligning data of varying granularity, and transforming categorical data for analysis. These issues will require careful data transformation and cleaning to ensure accurate and meaningful insights.