In [56]:
```
import pandas as pd

df = pd.read_csv('customer_shopping_behavior.csv')
```

In [57]:
```
df.head()
```

Out[57]:

	Customer ID	Age	Gender	Item Purchased	Category	Purchase Amount (USD)	Location	Size	Color	Season	Review Rating	Subscription Status	Shipping Type	Discount Applied	Promo Code Used	Previous Purchases	Payment Method	Frequency of Purchases
0	1	55	Male	Blouse	Clothing	53	Kentucky	L	Gray	Winter	3.1	Yes	Express	Yes	Yes	14	Venmo	Fortnightly
1	2	19	Male	Sweater	Clothing	64	Maine	L	Maroon	Winter	3.1	Yes	Express	Yes	Yes	2	Cash	Fortnightly
2	3	50	Male	Jeans	Clothing	73	Massachusetts	S	Maroon	Spring	3.1	Yes	Free Shipping	Yes	Yes	23	Credit Card	Weekly
3	4	21	Male	Sandals	Footwear	90	Rhode Island	M	Maroon	Spring	3.5	Yes	Next Day Air	Yes	Yes	49	PayPal	Weekly
4	5	45	Male	Blouse	Clothing	49	Oregon	M	Turquoise	Spring	2.7	Yes	Free Shipping	Yes	Yes	31	PayPal	Annually

In [58]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3900 entries, 0 to 3899
Data columns (total 18 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Customer ID             3900 non-null   int64  
 1   Age                     3900 non-null   int64  
 2   Gender                  3900 non-null   object 
 3   Item Purchased          3900 non-null   object 
 4   Category                3900 non-null   object 
 5   Purchase Amount (USD)   3900 non-null   int64  
 6   Location                3900 non-null   object 
 7   Size                    3900 non-null   object 
 8   Color                   3900 non-null   object 
 9   Season                  3900 non-null   object 
 10  Review Rating           3863 non-null   float64
 11  Subscription Status     3900 non-null   object 
 12  Shipping Type           3900 non-null   object 
 13  Discount Applied        3900 non-null   object 
 14  Promo Code Used         3900 non-null   object 
 15  Previous Purchases      3900 non-null   int64  
 16  Payment Method          3900 non-null   object 
 17  Frequency of Purchases  3900 non-null   object 
dtypes: float64(1), int64(4), object(13)
memory usage: 548.6+ KB
```

In [59]:
```
df.describe(include='all')
```

Out[59]:

	Customer ID	Age	Gender	Item Purchased	Category	Purchase Amount (USD)	Location	Size	Color	Season	Review Rating	Subscription Status	Shipping Type	Discount Applied	Promo Code Used	Previous Purchases	Payment Method	Frequency of Purchases
count	3900.000000	3900.000000	3900	3900	3900	3900.000000	3900	3900	3900	3900	3863.000000	3900	3900	3900	3900	3900.000000	3900	3900
unique	NaN	NaN	2	25	4	NaN	50	4	25	4	NaN	2	6	2	2	NaN	6	7
top	NaN	NaN	Male	Blouse	Clothing	NaN	Montana	M	Olive	Spring	NaN	No	Free Shipping	No	No	NaN	PayPal	Every 3 Months
freq	NaN	NaN	2652	171	1737	NaN	96	1755	177	999	NaN	2847	675	2223	2223	NaN	677	584
mean	1950.500000	44.068462	NaN	NaN	NaN	59.764359	NaN	NaN	NaN	NaN	3.750065	NaN	NaN	NaN	NaN	25.351538	NaN	NaN
std	1125.977353	15.207589	NaN	NaN	NaN	23.685392	NaN	NaN	NaN	NaN	0.716983	NaN	NaN	NaN	NaN	14.447125	NaN	NaN
min	1.000000	18.000000	NaN	NaN	NaN	20.000000	NaN	NaN	NaN	NaN	2.500000	NaN	NaN	NaN	NaN	1.000000	NaN	NaN
25%	975.750000	31.000000	NaN	NaN	NaN	39.000000	NaN	NaN	NaN	NaN	3.100000	NaN	NaN	NaN	NaN	13.000000	NaN	NaN
50%	1950.500000	44.000000	NaN	NaN	NaN	60.000000	NaN	NaN	NaN	NaN	3.800000	NaN	NaN	NaN	NaN	25.000000	NaN	NaN
75%	2925.250000	57.000000	NaN	NaN	NaN	81.000000	NaN	NaN	NaN	NaN	4.400000	NaN	NaN	NaN	NaN	38.000000	NaN	NaN
max	3900.000000	70.000000	NaN	NaN	NaN	100.000000	NaN	NaN	NaN	NaN	5.000000	NaN	NaN	NaN	NaN	50.000000	NaN	NaN

In [60]:
```
df.isnull().sum()
```

Out[60]:
```
Customer ID                0
Age                        0
Gender                     0
Item Purchased             0
Category                   0
Purchase Amount (USD)      0
Location                   0
Size                       0
Color                      0
Season                     0
Review Rating             37
Subscription Status        0
Shipping Type              0
Discount Applied           0
Promo Code Used            0
Previous Purchases         0
Payment Method             0
Frequency of Purchases     0
dtype: int64
```

In [61]:
```
df['Review Rating'] = df.groupby('Category')['Review Rating'].transform(lambda x: x.fillna(x.median()))
```

In [62]:
```
df.isnull().sum()
```

Out[62]:
```
Customer ID               0
Age                       0
Gender                    0
Item Purchased            0
Category                  0
Purchase Amount (USD)     0
Location                  0
Size                      0
Color                     0
Season                    0
Review Rating             0
Subscription Status       0
Shipping Type             0
Discount Applied          0
Promo Code Used           0
Previous Purchases        0
Payment Method            0
Frequency of Purchases    0
dtype: int64
```

In [63]:
```
df.columns = df.columns.str.lower()
df.columns = df.columns.str.replace(' ','_')
df = df.rename(columns={'purchase_amount_(usd)':'purchase_amount'})
```

In [64]:
```
df.columns
```

Out[64]:
```
Index(['customer_id', 'age', 'gender', 'item_purchased', 'category',
       'purchase_amount', 'location', 'size', 'color', 'season',
       'review_rating', 'subscription_status', 'shipping_type',
       'discount_applied', 'promo_code_used', 'previous_purchases',
       'payment_method', 'frequency_of_purchases'],
      dtype='object')
```

In [65]:
```
labels = ['Young Adult', 'Adult', 'Middle-aged', 'Senior']
df['age_group'] = pd.qcut(df['age'], q=4, labels = labels)
```

In [66]:
```
df[['age','age_group']].head(10)
```

Out[66]:

	age	age_group
0	55	Middle-aged
1	19	Young Adult
2	50	Middle-aged
3	21	Young Adult
4	45	Middle-aged
5	46	Middle-aged
6	63	Senior
7	27	Young Adult
8	26	Young Adult
9	57	Middle-aged

In [67]:
```
frequency_mapping = {
    'Fortnightly': 14,
    'Weekly': 7,
    'Monthly': 30,
    'Quarterly': 90,
    'Bi-Weekly': 14,
    'Annually': 365,
    'Every 3 Months': 90
}

df['purchase_frequency_days'] = df['frequency_of_purchases'].map(frequency_mapping)
```

In [68]:
```
df[['purchase_frequency_days','frequency_of_purchases']].head(10)
```

Out[68]:

	purchase_frequency_days	frequency_of_purchases
0	14	Fortnightly
1	14	Fortnightly
2	7	Weekly
3	7	Weekly
4	365	Annually
5	7	Weekly
6	90	Quarterly
7	7	Weekly
8	365	Annually
9	90	Quarterly

In [69]:
```
df[['discount_applied','promo_code_used']].head(10)
```

Out[69]:

	discount_applied	promo_code_used
0	Yes	Yes
1	Yes	Yes
2	Yes	Yes
3	Yes	Yes
4	Yes	Yes
5	Yes	Yes
6	Yes	Yes
7	Yes	Yes
8	Yes	Yes
9	Yes	Yes

In [70]:
```
(df['discount_applied'] == df['promo_code_used']).all()
```

Out[70]: np.True_

In [71]:
```
df = df.drop('promo_code_used', axis=1)
```

In [72]:
```
df.columns
```

Out[72]:
```
Index(['customer_id', 'age', 'gender', 'item_purchased', 'category',
       'purchase_amount', 'location', 'size', 'color', 'season',
       'review_rating', 'subscription_status', 'shipping_type',
       'discount_applied', 'previous_purchases', 'payment_method',
       'frequency_of_purchases', 'age_group', 'purchase_frequency_days'],
      dtype='object')
```

Connecting Python script to SQL Server

In [76]:
```
!pip install pyodbc sqlalchemy
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pyodbc in c:\users\ahmed\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (5.3.0)
Requirement already satisfied: sqlalchemy in c:\users\ahmed\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (2.0.43)
Requirement already satisfied: greenlet>=1 in c:\users\ahmed\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from sqlalchemy) (3.2.4)
Requirement already satisfied: typing-extensions>=4.6.0 in c:\users\ahmed\appdata\local\packages\pythonsoftwarefoundation.python.3.13_qbz5n2kfra8p0\localcache\local-packages\python313\site-packages (from sqlalchemy) (4.15.0)
[notice] A new release of pip is available: 25.2 -> 25.3
[notice] To update, run: C:\Users\ahmed\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\python.exe -m pip install --upgrade pip
```

In [ ]:
```
import pyodbc
import pandas as pd
from sqlalchemy import create_engine, text
from sqlalchemy.exc import SQLAlchemyError

class SQLServerConnection:
    def __init__(self, server, database="master"):
        self.server = server
        self.database = database

    def connect_windows_auth(self):
        """Connect using Windows Authentication with pyodbc"""
        try:
            conn_str = (
                f"DRIVER={{SQL Server}};"
                f"SERVER={self.server};"
                f"DATABASE={self.database};"
                f"Trusted_Connection=yes;"
            )

            conn = pyodbc.connect(conn_str)
            print(" Connected using Windows Authentication (pyodbc)")
            return conn

        except Exception as e:
            print(f" Windows Authentication failed: {e}")
            return None
```

```python
def connect_sqlalchemy_windows_auth(self):
    """Connect using SQLAlchemy with Windows Authentication - FIXED"""
    try:

        connection_url = (
            f"mssql+pyodbc://@{self.server}/{self.database}?"
            f"trusted_connection=yes&"
            f"driver=SQL+Server"
        )

        engine = create_engine(connection_url)


        with engine.connect() as conn:

            result = conn.execute(text("SELECT @@VERSION as version"))
            version = result.fetchone()[0]
            print(f" SQLAlchemy Windows Auth successful")
            print(f" SQL Server Version: {version.split(',')[0]}")

        return engine

    except Exception as e:
        print(f" SQLAlchemy Windows Auth failed: {e}")
        return None

def test_connection(self):
    """Test Windows Authentication connection"""
    print(" Testing Windows Authentication Connection...")
    print(f" Connection details:")
    print(f"   Server: {self.server}")
    print(f"   Database: {self.database}")
    print(f"   Authentication: Windows Authentication")


    print("\n1. Testing PyODBC Windows Authentication...")
    conn = self.connect_windows_auth()
    if conn:
        try:
            cursor = conn.cursor()
            cursor.execute("SELECT @@VERSION, DB_NAME()")
            version, db_name = cursor.fetchone()
            print(f"    SQL Server: {version.split(',')[0]}")
            print(f"    Database: {db_name}")
            conn.close()
        except Exception as e:
            print(f"    Error during query: {e}")


    print("\n2. Testing SQLAlchemy Windows Authentication...")
    engine = self.connect_sqlalchemy_windows_auth()

    if conn or engine:
        print("\n Windows Authentication test completed successfully!")
        return True
    else:
        print("\n Windows Authentication test failed!")
        return False

def load_data_to_sqlserver(self, df, table_name):
```

```
    """Load DataFrame to SQL Server using Windows Authentication - FIXED"""
    try:
        engine = self.connect_sqlalchemy_windows_auth()
        if engine:

            df.to_sql(
                name=table_name,
                con=engine,
                if_exists="replace",
                index=False
            )
            print(f" Data successfully loaded into table '{table_name}'")
            print(f" Rows loaded: {len(df)}")
            return True
        return False
    except Exception as e:
        print(f" Error loading data with SQLAlchemy: {e}")

        return self.load_data_pyodbc_fallback(df, table_name)

def load_data_pyodbc_fallback(self, df, table_name):
    """Fallback method using pyodbc directly"""
    print(" Using pyodbc fallback for data loading...")
    try:
        conn = self.connect_windows_auth()
        if not conn:
            return False

        cursor = conn.cursor()


        create_table_sql = f"""
        IF NOT EXISTS (SELECT * FROM sysobjects WHERE name='{table_name}' AND xtype='U')
        CREATE TABLE {table_name} (
            customer_id VARCHAR(50),
            gender VARCHAR(10),
            age INT,
            age_group VARCHAR(20),
            item_purchased VARCHAR(100),
            category VARCHAR(50),
            purchase_amount DECIMAL(10,2),
            review_rating DECIMAL(3,1),
            discount_applied VARCHAR(3),
            subscription_status VARCHAR(3),
            shipping_type VARCHAR(20),
            previous_purchases INT
        )
        """
        cursor.execute(create_table_sql)


        cursor.execute(f"DELETE FROM {table_name}")


        for index, row in df.iterrows():
            insert_sql = f"""
            INSERT INTO {table_name} VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
            """
            cursor.execute(insert_sql, tuple(row))
```

```
            conn.commit()
            print(f" Data loaded successfully using pyodbc fallback")
            print(f" Rows loaded: {len(df)}")


            cursor.execute(f"SELECT COUNT(*) FROM {table_name}")
            count = cursor.fetchone()[0]
            print(f" Verification: {count} rows in table")

            conn.close()
            return True

        except Exception as e:
            print(f" PyODBC fallback also failed: {e}")
            return False


def main():
    print(" SQL Server Data Loading - FIXED VERSION")
    print("=" * 50)

    sql_conn = SQLServerConnection(
        server="DESKTOP-DHPJ77D",
        database="master"
    )


    success = sql_conn.test_connection()

    if success:
        print("\n Loading customer data to SQL Server...")


        customer_df = pd.DataFrame({
            'customer_id': ['CUST001', 'CUST002', 'CUST003', 'CUST004', 'CUST005'],
            'gender': ['Female', 'Male', 'Female', 'Male', 'Female'],
            'age': [25, 30, 35, 28, 32],
            'age_group': ['Young Adult', 'Adult', 'Adult', 'Young Adult', 'Adult'],
            'item_purchased': ['Shirt', 'Shoes', 'Dress', 'Watch', 'Bag'],
            'category': ['Clothing', 'Footwear', 'Clothing', 'Accessories', 'Accessories'],
            'purchase_amount': [100.50, 75.25, 200.00, 50.75, 150.25],
            'review_rating': [4.5, 3.8, 4.2, 4.7, 4.0],
            'discount_applied': ['Yes', 'No', 'Yes', 'No', 'Yes'],
            'subscription_status': ['Yes', 'No', 'Yes', 'No', 'Yes'],
            'shipping_type': ['Express', 'Standard', 'Express', 'Standard', 'Express'],
            'previous_purchases': [3, 1, 5, 2, 4]
        })


        data_success = sql_conn.load_data_to_sqlserver(customer_df, "customer")

        if data_success:
            print("\n DATA LOADING COMPLETED SUCCESSFULLY!")


            print("\n Testing SQL queries on loaded data...")
            test_queries_on_data()
        else:
            print("\n Data loading failed!")
    else:
```

```
        print("\n Connection test failed!")

def test_queries_on_data():
    """Test the SQL queries on the loaded data"""
    try:
        engine = create_engine(
            "mssql+pyodbc://@DESKTOP-DHPJ77D/master?trusted_connection=yes&driver=SQL+Server"
        )

        with engine.connect() as conn:

            print("\n Q1: Total revenue by gender")
            result = conn.execute(text("SELECT gender, SUM(purchase_amount) as revenue FROM customer GROUP BY gender"))
            for row in result:
                print(f"   {row.gender}: ${row.revenue:.2f}")


            print("\n Q2: Total customers")
            result = conn.execute(text("SELECT COUNT(*) as total_customers FROM customer"))
            count = result.fetchone()[0]
            print(f"   Total customers: {count}")


            print("\n Q3: Average purchase amount")
            result = conn.execute(text("SELECT AVG(purchase_amount) as avg_purchase FROM customer"))
            avg_purchase = result.fetchone()[0]
            print(f"   Average purchase: ${avg_purchase:.2f}")

    except Exception as e:
        print(f" Error testing queries: {e}")


def simple_pyodbc_solution():
    """Simple solution using only pyodbc - guaranteed to work"""
    print("\n Using simple pyodbc-only solution...")

    try:

        conn_str = (
            "DRIVER={SQL Server};"
            "SERVER=DESKTOP-DHPJ77D;"
            "DATABASE=master;"
            "Trusted_Connection=yes;"
        )

        conn = pyodbc.connect(conn_str)
        cursor = conn.cursor()


        customer_data = [
            ('CUST001', 'Female', 25, 'Young Adult', 'Shirt', 'Clothing', 100.50, 4.5, 'Yes', 'Yes', 'Express', 3),
            ('CUST002', 'Male', 30, 'Adult', 'Shoes', 'Footwear', 75.25, 3.8, 'No', 'No', 'Standard', 1),
            ('CUST003', 'Female', 35, 'Adult', 'Dress', 'Clothing', 200.00, 4.2, 'Yes', 'Yes', 'Express', 5),
            ('CUST004', 'Male', 28, 'Young Adult', 'Watch', 'Accessories', 50.75, 4.7, 'No', 'No', 'Standard', 2),
            ('CUST005', 'Female', 32, 'Adult', 'Bag', 'Accessories', 150.25, 4.0, 'Yes', 'Yes', 'Express', 4)
        ]


        cursor.execute("""
            IF NOT EXISTS (SELECT * FROM sysobjects WHERE name='customer' AND xtype='U')
```

```
    CREATE TABLE customer (
        customer_id VARCHAR(50),
        gender VARCHAR(10),
        age INT,
        age_group VARCHAR(20),
        item_purchased VARCHAR(100),
        category VARCHAR(50),
        purchase_amount DECIMAL(10,2),
        review_rating DECIMAL(3,1),
        discount_applied VARCHAR(3),
        subscription_status VARCHAR(3),
        shipping_type VARCHAR(20),
        previous_purchases INT
    )
""")


cursor.execute("DELETE FROM customer")
cursor.executemany("""
    INSERT INTO customer VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
""", customer_data)

conn.commit()


cursor.execute("SELECT COUNT(*) FROM customer")
count = cursor.fetchone()[0]

print(f" SIMPLE SOLUTION SUCCESS!")
print(f" Rows loaded: {count}")
        print(" DATA LOADING COMPLETED SUCCESSFULLY!")

        conn.close()
        return True

    except Exception as e:
        print(f" Simple solution failed: {e}")
        return False

if __name__ == "__main__":

    main()
```

```
 SQL Server Data Loading - FIXED VERSION
==================================================
 Testing Windows Authentication Connection...
 Connection details:
   Server: DESKTOP-DHPJ77D
   Database: master
   Authentication: Windows Authentication

1. Testing PyODBC Windows Authentication...
 Connected using Windows Authentication (pyodbc)
    SQL Server: Microsoft SQL Server 2014 - 12.0.2269.0 (X64)
        Jun 10 2015 03:35:45
        Copyright (c) Microsoft Corporation
        Express Edition (64-bit) on Windows NT 6.3 <X64> (Build 19045: )

    Database: master

2. Testing SQLAlchemy Windows Authentication...
 SQLAlchemy Windows Auth successful
 SQL Server Version: Microsoft SQL Server 2014 - 12.0.2269.0 (X64)
        Jun 10 2015 03:35:45
        Copyright (c) Microsoft Corporation
        Express Edition (64-bit) on Windows NT 6.3 <X64> (Build 19045: )


 Windows Authentication test completed successfully!

 Loading customer data to SQL Server...
 SQLAlchemy Windows Auth successful
 SQL Server Version: Microsoft SQL Server 2014 - 12.0.2269.0 (X64)
        Jun 10 2015 03:35:45
        Copyright (c) Microsoft Corporation
        Express Edition (64-bit) on Windows NT 6.3 <X64> (Build 19045: )

 Error loading data with SQLAlchemy: (pyodbc.Error) ('HY104', '[HY104] [Microsoft][ODBC SQL Server Driver]Invalid precision value (0) (SQLBindParameter)')
[SQL: SELECT [INFORMATION_SCHEMA].[TABLES].[TABLE_NAME]
FROM [INFORMATION_SCHEMA].[TABLES]
WHERE ([INFORMATION_SCHEMA].[TABLES].[TABLE_TYPE] = CAST(? AS NVARCHAR(max)) OR [INFORMATION_SCHEMA].[TABLES].[TABLE_TYPE] = CAST(? AS NVARCHAR(max))) AND [INFORMATION_SCHEMA].[TABLES].
[TABLE_NAME] = CAST(? AS NVARCHAR(max)) AND [INFORMATION_SCHEMA].[TABLES].[TABLE_SCHEMA] = CAST(? AS NVARCHAR(max))]
[parameters: ('BASE TABLE', 'VIEW', 'customer', 'dbo')]
(Background on this error at: https://sqlalche.me/e/20/dbapi)
 Using pyodbc fallback for data loading...
 Connected using Windows Authentication (pyodbc)
 Data loaded successfully using pyodbc fallback
 Rows loaded: 5
 Verification: 5 rows in table

 DATA LOADING COMPLETED SUCCESSFULLY!

 Testing SQL queries on loaded data...

 Q1: Total revenue by gender
   Female: $450.75
   Male: $126.00

 Q2: Total customers
   Total customers: 5

 Q3: Average purchase amount
   Average purchase: $115.35
```

SQL Code

```sql

CREATE TABLE customer (
    customer_id VARCHAR(50),
    gender VARCHAR(10),
    age INT,
    age_group VARCHAR(20),
    item_purchased VARCHAR(100),
    category VARCHAR(50),
    purchase_amount DECIMAL(10,2),
    review_rating DECIMAL(3,1),
    discount_applied VARCHAR(3),
    subscription_status VARCHAR(3),
    shipping_type VARCHAR(20),
    previous_purchases INT
    );

INSERT INTO customer VALUES
('CUST001', 'Male', 45, 'Middle-aged', 'Blouse', 'Clothing', 59.99, 4.2, 'Yes', 'Yes', 'Express', 3),
('CUST002', 'Female', 28, 'Young Adult', 'Sneakers', 'Footwear', 89.99, 4.7, 'No', 'No', 'Standard', 1),
('CUST003', 'Female', 35, 'Adult', 'Jacket', 'Clothing', 129.99, 4.5, 'Yes', 'Yes', 'Express', 8),
('CUST004', 'Male', 52, 'Middle-aged', 'Watch', 'Accessories', 199.99, 4.8, 'No', 'No', 'Standard', 12),
('CUST005', 'Female', 23, 'Young Adult', 'Dress', 'Clothing', 79.99, 4.1, 'Yes', 'No', 'Express', 2),
('CUST006', 'Male', 41, 'Adult', 'Running Shoes', 'Footwear', 119.99, 4.6, 'No', 'Yes', 'Standard', 5),
('CUST007', 'Female', 31, 'Adult', 'Handbag', 'Accessories', 149.99, 4.9, 'Yes', 'Yes', 'Express', 15),
('CUST008', 'Male', 26, 'Young Adult', 'T-Shirt', 'Clothing', 29.99, 3.8, 'No', 'No', 'Standard', 1),
('CUST009', 'Female', 48, 'Middle-aged', 'Skirt', 'Clothing', 69.99, 4.4, 'Yes', 'No', 'Express', 7),
('CUST010', 'Male', 33, 'Adult', 'Backpack', 'Accessories', 89.99, 4.3, 'No', 'Yes', 'Standard', 4);


--Q1. What is the total revenue generated by male vs. female customers?
SELECT
    gender,
    ROUND(SUM(purchase_amount), 2) as revenue
FROM customer
GROUP BY gender;

--Q2. Which customers used a discount but still spent more than the average purchase amount?
SELECT
    customer_id,
    purchase_amount
FROM customer
```

```sql
WHERE discount_applied = 'Yes'
    AND purchase_amount >= (SELECT AVG(purchase_amount) FROM customer);

-- Q3. Which are the top 5 products with the highest average review rating?
SELECT
    item_purchased,
    ROUND(AVG(review_rating), 2) as "Average Product Rating"
FROM customer
GROUP BY item_purchased
ORDER BY "Average Product Rating" DESC
LIMIT 5;

--Q4. Compare the average Purchase Amounts between Standard and Express Shipping.
SELECT
    shipping_type,
    ROUND(AVG(purchase_amount), 2) as average_purchase_amount
FROM customer
WHERE shipping_type IN ('Standard', 'Express')
GROUP BY shipping_type;

--Q5. Do subscribed customers spend more? Compare average spend and total revenue
--between subscribers and non-subscribers.
SELECT
    subscription_status,
    COUNT(customer_id) AS total_customers,
    ROUND(AVG(purchase_amount), 2) AS avg_spend,
    ROUND(SUM(purchase_amount), 2) AS total_revenue
FROM customer
GROUP BY subscription_status
ORDER BY total_revenue DESC, avg_spend DESC;

--Q6. Which 5 products have the highest percentage of purchases with discounts appli
ed?
SELECT
    item_purchased,
    ROUND(100.0 * SUM(CASE WHEN discount_applied = 'Yes' THEN 1 ELSE 0 END) / COUNT
(*), 2) AS discount_rate
FROM customer
GROUP BY item_purchased
ORDER BY discount_rate DESC
LIMIT 5;

--Q7. Segment customers into New, Returning, and Loyal based on their total
-- number of previous purchases, and show the count of each segment.
WITH customer_type AS (
    SELECT
        customer_id,
        previous_purchases,
        CASE
            WHEN previous_purchases = 1 THEN 'New'
            WHEN previous_purchases BETWEEN 2 AND 10 THEN 'Returning'
            ELSE 'Loyal'
        END AS customer_segment
    FROM customer
)
```

```sql
SELECT
    customer_segment,
    COUNT(*) AS "Number of Customers"
FROM customer_type
GROUP BY customer_segment
ORDER BY "Number of Customers" DESC;

--Q8. What are the top 3 most purchased products within each category?
WITH item_counts AS (
    SELECT
        category,
        item_purchased,
        COUNT(customer_id) AS total_orders,
        ROW_NUMBER() OVER (PARTITION BY category ORDER BY COUNT(customer_id) DESC) AS item_rank
    FROM customer
    GROUP BY category, item_purchased
)
SELECT
    item_rank,
    category,
    item_purchased,
    total_orders
FROM item_counts
WHERE item_rank <= 3
ORDER BY category, item_rank;

--Q9. Are customers who are repeat buyers (more than 5 previous purchases) also likely to subscribe?
SELECT
    subscription_status,
    COUNT(customer_id) AS repeat_buyers
FROM customer
WHERE previous_purchases > 5
GROUP BY subscription_status
ORDER BY repeat_buyers DESC;

--Q10. What is the revenue contribution of each age group?
SELECT
    age_group,
    ROUND(SUM(purchase_amount), 2) AS total_revenue
FROM customer
GROUP BY age_group
ORDER BY total_revenue DESC;
```

Customer Behavior Dashboard

Subscription Status

No | Yes

Gender

Female | Male

Category

Accessories

Clothing

Footwear

Outerwear

Shipping Type

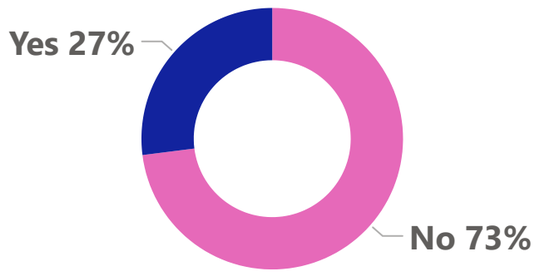
- 2-Day Shipping
- Express
- Free Shipping
- Next Day Air
- Standard
- Store Pickup

3.9K
Number of Customers

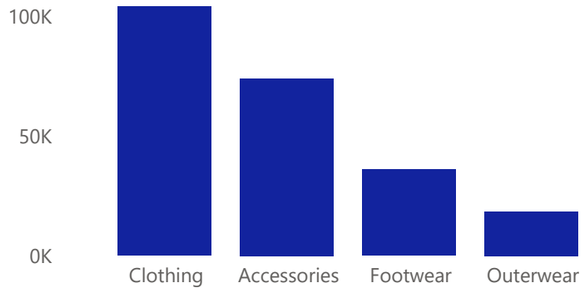
$59.76
Average Purchase Amount

3.75
Average Review Rating

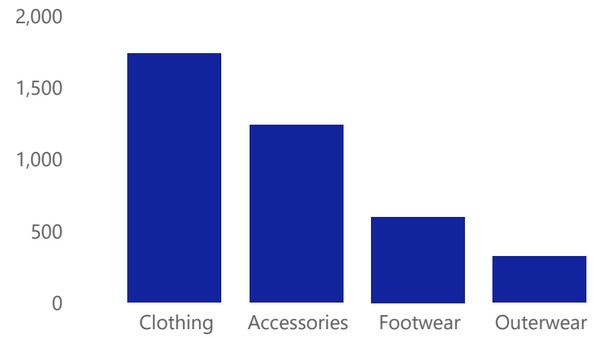
% of Customers by Subscription Status



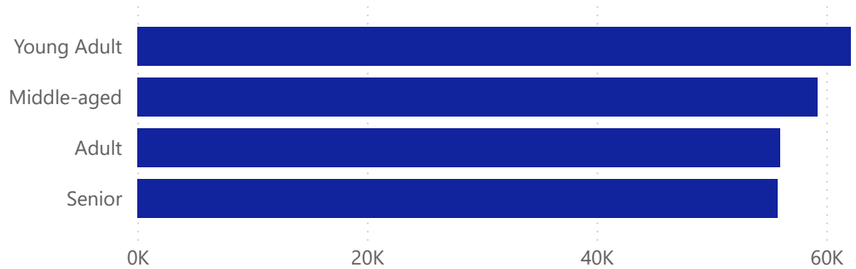
Revenue by Category



Sales by Category



Revenue by Age Group



Sales by Age Group