Customer Shopping Behavior Analysis

1. Project Overview:

This project analyzes customer shopping behavior using transactional data from 3,900 purchases across various product categories. The goal is to uncover insights into spending patterns, customer segments, product preferences, and subscription behavior to guide strategic business decisions.

2. Dataset Summary:

-Rows: 3,900
- Columns: 18
- Key Features:
- Customer demographics (Age, Gender, Location, Subscription Status)
- Purchase details (Item Purchased, Category, Purchase Amount, Season, Size, Color)
- Shopping behavior (Discount Applied, Promo Code Used, Previous Purchases, Frequency of Purchases, Review Rating, Shipping Type)
- Missing Data: 37 values in Review Rating column

3. Exploratory Data Analysis using Python (EDA):

We began with data preparation and cleaning in Python:

- Data Loading: Imported the dataset using pandas.

- Initial Exploration: Used df.info() to check structure and .describe() for summary statistics.

```
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
```

```
df.describe()
```

	Customer ID	Age	Purchase Amount (USD)	Review Rating	Previous Purchases
count	3900.000000	3900.000000	3900.000000	3863.000000	3900.000000
mean	1950.500000	44.068462	59.764359	3.750065	25.351538
std	1125.977353	15.207589	23.685392	0.716983	14.447125
min	1.000000	18.000000	20.000000	2.500000	1.000000
25%	975.750000	31.000000	39.000000	3.100000	13.000000
50%	1950.500000	44.000000	60.000000	3.800000	25.000000
75%	2925.250000	57.000000	81.000000	4.400000	38.000000
max	3900.000000	70.000000	100.000000	5.000000	50.000000

4. Missing Data Handling:

Checked for null values and imputed missing values in the Review Rating column using the median rating of each product category.

```
df.isnull().sum()
```

```
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
```

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

5. Column Standardization:

Renamed columns to snake case for better readability and documentation.

```
df.columns = df.columns.str.lower()
df.columns = df.columns.str.replace(' ','_')
df.columns
```

```
Index(['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'],
      dtype='object')
```

```
df = df.rename(columns={'purchase_amount_(usd)': 'purchase_amount'})
df.columns
```

```
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')
```

6. Feature Engineering:

- Created age_group column by binning customer ages.
- Created purchase_frequency_days column from purchase data.

```
lable = ['Young Adult','Adult','Middle-Age','Senior']
df['age_group'] = pd.qcut(df['age'], q=4 , labels = lable)
df[['age','age_group']].head(10)
```

	age	age_group
0	55	Middle-Age
1	19	Young Adult
2	50	Middle-Age
3	21	Young Adult

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

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

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

	purchase_frequency_days	frequency_of_purchases
0	14	Fortnightly
1	14	Fortnightly
2	7	Weekly
3	7	Weekly

7. Data Consistency Check:

Verified if discount_applied and promo_code_used were redundant; dropped promo_code_used.

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

```
np.True_
```

```
df = df.drop('promo_code_used', axis = 1)
```

8. Database Integration:

Connected Python script to MySQL and loaded the cleaned DataFrame into the database for SQL analysis.

```
pip install pymysql sqlalchemy
```

```
from sqlalchemy import create_engine

# MySQL connection
username = "root"
password = "121212"
host = "localhost"
port = "3306"
database = "customer_behavior"

engine = create_engine(f"mysql+pymysql://{username}:{password}@{host}:{port}/{database}")

# Write DataFrame to MySQL
table_name = "customer"   # choose any table name
df.to_sql(table_name, engine, if_exists="replace", index=False)

# Read back sample
pd.read_sql("SELECT * FROM customer LIMIT 5;", engine)
```