Customer Shopping Behavior Analysis

---

1. Project Overview & Data Profile

The analysis covers 3,900 individual records across 18 distinct data categories, including demographics, purchase specifics, and behavioral metrics.

- **Demographics:** Tracks age, gender, geographic location, and whether the customer is a subscriber.
- **Transactional Data:** Details include items purchased, product categories, total cost per item, and seasonal timing.
- **Behavioral Metrics:** Includes the use of discounts/promo codes, purchase frequency, review scores, and shipping methods.
- **Data Integrity:** The initial review identified 37 missing entries within the "Review Rating" column.

---

2. Data Preparation (Python)

The data was cleaned and structured using the Python pandas library to ensure accuracy for subsequent analysis.

- **Handling Nulls:** Missing review scores were addressed by imputing the median rating specific to each product category.
- **Standardization:** All column headers were converted to "snake_case" for improved coding efficiency and readability.
- **Feature Engineering:** * Binned ages into specific age_group categories.
    - Converted shopping frequency data into a numerical purchase_frequency_days format.
- **Redundancy Check:** Analysis revealed that "discount_applied" and "promo_code_used" were redundant; the latter was removed from the dataset.

3. Strategic Insights (SQL)

The cleaned data was migrated to a PostgreSQL database to query specific business performance metrics.

Business Question	Key Findings
Revenue by Gender	Male customers contributed significantly more revenue ($157,890) compared to female customers ($75,191). +1
Shipping Performance	Express shipping users spend slightly more on average ($60.48) than those using standard shipping ($58.46).
Subscription Value	While there are fewer subscribers (1,053), their average spend ($59.49) is nearly equal to non-subscribers ($59.87).
Customer Segments	The vast majority of the customer base (3,116) is classified as "Loyal," with 701 "Returning" and 83 "New" customers.
Product Popularity	Top products within their categories include Jewelry (Accessories), Blouses (Clothing), and Sandals (Footwear).

4. Visual Intelligence (Power BI)

An interactive dashboard was created to synthesize these findings into a single visual interface.

- **KPI Tracking:** Displays a global average purchase amount of $59.76 and a mean review rating of 3.75.

- **Subscription Breakdown:** 27% of customers are currently enrolled in a subscription plan, while 73% are not.

- **Demographic Contribution:** "Young Adults" represent the highest-earning age segment, contributing $62,143 in total revenue.

---

5. Strategic Recommendations

To drive growth, the following initiatives are suggested:

- **Enhance Subscriptions:** Develop and market exclusive incentives to convert the 73% of non-subscribers.

- **Loyalty Incentives:** Create rewards specifically targeted at "Returning" customers to transition them into the "Loyal" segment.

- **Precision Marketing:** Direct advertising efforts toward the "Young Adult" demographic and promote express shipping options to increase transaction values.

Open in Colab

Data Exploration

```
import pandas as pd

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

```
df.head()
```

	Customer ID	Age	Gender	Item Purchased	Category	Purchase Amount (USD)	Location	Size	Color	Season	Review Rating	Subscription Status	Shipping Type
0	1	55	Male	Blouse	Clothing	53	Kentucky	L	Gray	Winter	3.1	Yes	Express
1	2	19	Male	Sweater	Clothing	64	Maine	L	Maroon	Winter	3.1	Yes	Express
2	3	50	Male	Jeans	Clothing	73	Massachusetts	S	Maroon	Spring	3.1	Yes	Free Shipping
3	4	21	Male	Sandals	Footwear	90	Rhode Island	M	Maroon	Spring	3.5	Yes	Next Day Air
4	5	45	Male	Blouse	Clothing	49	Oregon	M	Turquoise	Spring	2.7	Yes	Free Shipping

Next steps: Generate code with df | New interactive sheet

```
df.describe(include='all')
```

	Customer ID	Age	Gender	Item Purchased	Category	Purchase Amount (USD)	Location	Size	Color	Season	Review Rating	Subscr
count	3900.000000	3900.000000	3900	3900	3900	3900.000000	3900	3900	3900	3900	3863.000000	
unique	NaN	NaN	2	25	4	NaN	50	4	25	4	NaN	
top	NaN	NaN	Male	Blouse	Clothing	NaN	Montana	M	Olive	Spring	NaN	
freq	NaN	NaN	2652	171	1737	NaN	96	1755	177	999	NaN	
mean	1950.500000	44.068462	NaN	NaN	NaN	59.764359	NaN	NaN	NaN	NaN	3.750065	
std	1125.977353	15.207589	NaN	NaN	NaN	23.685392	NaN	NaN	NaN	NaN	0.716983	
min	1.000000	18.000000	NaN	NaN	NaN	20.000000	NaN	NaN	NaN	NaN	2.500000	
25%	975.750000	31.000000	NaN	NaN	NaN	39.000000	NaN	NaN	NaN	NaN	3.100000	
50%	1950.500000	44.000000	NaN	NaN	NaN	60.000000	NaN	NaN	NaN	NaN	3.800000	
75%	2925.250000	57.000000	NaN	NaN	NaN	81.000000	NaN	NaN	NaN	NaN	4.400000	
max	3900.000000	70.000000	NaN	NaN	NaN	100.000000	NaN	NaN	NaN	NaN	5.000000	

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

	0
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()))
```

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

	0
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

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

```
# creating a new column age_group

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

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

	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

```
# creating column purchase_frequency_days

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)
```

```
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
4	365	Annually
5	7	Weekly
6	90	Quarterly
7	7	Weekly
8	365	Annually
9	90	Quarterly

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

	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

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

```
np.True_
```

```
# removing promo_code_used as it is irrelevant
df = df.drop('promo_code_used', axis = 1)
```

```
df.columns
```

```
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')
```

```
# pip install psycopg2-binary sqlalchemy
```

Connecting to PostgreSQL

```
""" from sqlalchemy import create_engine

# Step 1: Connect to PostgreSQL

username = " " # default user
password = " " # password set during installation
host = " " # if running locally
port = " " # default PostgreSQL port
database = " " # the database created for this project

engine = create_engine(f"postgresql+psycopg2://{username}:{password}@{host}:{port}/{database}")

# Step 2: Load Dataframe into PostgreSQL
table_name = " "
df.to_sql(table_name, engine, if_exists="replace", index=False)

print(f"Data successfully loaded into table '{table_name}' in database '{database}'.")

"""
```

```
' from sqlalchemy import create_engine\n\n# Step 1: Connect to PostgreSQL\n\nusername = " " # default user\npassword = " "
# password set during installation\nhost = " " # if running locally\nport = " " # default PostgreSQL port\ndatabase = " " #
the database created for this project\n\nengine = create_engine(f"postgresql+psycopg2://{username}:{password}@{host}:{por
t}/{database}")\n\n# Step 2: Load Dataframe into PostgreSQL\ntable_name = " "\ndf.to_sql(table_name, engine, if_exists="rep
```

Connecting to MySQL

```
#!pip install pymysql sqlalchemy
```

```
'''

from sqlalchemy import create_engine

# MySQL connection
username = " "
password = " "
host = " "
```

```
port = " "
database = " "

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

# Writing DataFrame to MySQL
table_name = "mytable"
df.to_sql(table_name, engine, if_exists="replace", index=False)
```