

## Customer Shopping Behavior Analysis

```
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	Discount Applied	Promo Code Used
0	1	55	Male	Blouse	Clothing	53	Kentucky	L	Gray	Winter	3.1	Yes	Express	Yes	Yes
1	2	19	Male	Sweater	Clothing	64	Maine	L	Maroon	Winter	3.1	Yes	Express	Yes	Yes
2	3	50	Male	Jeans	Clothing	73	Massachusetts	S	Maroon	Spring	3.1	Yes	Free Shipping	Yes	Yes
3	4	21	Male	Sandals	Footwear	90	Rhode Island	M	Maroon	Spring	3.5	Yes	Next Day Air	Yes	Yes
4	5	45	Male	Blouse	Clothing	49	Oregon	M	Turquoise	Spring	2.7	Yes	Free Shipping	Yes	Yes

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(include='all')

	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
count	3900.000000	3900.000000	3900	3900	3900	3900.000000	3900	3900	3900	3900	3863.000000	3900	3900	3900	3900
unique	NaN	NaN	2	25	4	NaN	50	4	25	4	NaN	2	6	Free Shipping	Free Shipping
top	NaN	NaN	Male	Blouse	Clothing	NaN	Montana	M	Olive	Spring	NaN	No	NaN	NaN	NaN
freq	NaN	NaN	2652	171	1737	NaN	96	1755	177	999	NaN	2847	675	NaN	NaN
mean	1950.500000	44.068462	NaN	NaN	NaN	59.764359	NaN	NaN	NaN	NaN	3.750065	NaN	NaN	NaN	NaN
std	1125.977353	15.207589	NaN	NaN	NaN	23.685392	NaN	NaN	NaN	NaN	0.716983	NaN	NaN	NaN	NaN
min	1.000000	18.000000	NaN	NaN	NaN	20.000000	NaN	NaN	NaN	NaN	2.500000	NaN	NaN	NaN	NaN
25%	975.750000	31.000000	NaN	NaN	NaN	39.000000	NaN	NaN	NaN	NaN	3.100000	NaN	NaN	NaN	NaN
50%	1950.500000	44.000000	NaN	NaN	NaN	60.000000	NaN	NaN	NaN	NaN	3.800000	NaN	NaN	NaN	NaN
75%	2925.250000	57.000000	NaN	NaN	NaN	81.000000	NaN	NaN	NaN	NaN	4.400000	NaN	NaN	NaN	NaN
max	3900.000000	70.000000	NaN	NaN	NaN	100.000000	NaN	NaN	NaN	NaN	5.000000	NaN	NaN	NaN	NaN

```

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

```

- As we can see Review Rating has 37 null values.
- To deal with this we can replace it with either Mean or Median.
- But mean is affected by outliers and median is robust to outliers.
- We can impute null with Median but reviews are related to different product categories
- So it will be unfair if we impute clothing category null with footwear category, and that can introduce bias into dataset
- So we will replace the Null's with Median within each category
- We will find median review rating for each category and impute missing according to category.

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

```
## convert column names to snake case i.e all letters to lower case and underscores instead of spaces
for col in df.columns:
    #df[col] = df[col].replace(" ", "_").lower()
    df = df.rename(columns={col: col.replace(" ", "_").lower()})
```

```
## Rename Purchase amount column
df = df.rename(columns={'purchase_amount_(usd)': 'purchase_amount'})
```

```
## create new column age_group using qcut()
labels = ['Young Adult', 'Adult', 'Middle Aged', 'Senior' ]
df['age_group'] = pd.qcut(df['age'], q=4, labels=labels)
df[['age', 'age_group']].head()
```

	<b>age</b>	<b>age_group</b>
0	55	Middle Aged
1	19	Young Adult
2	50	Middle Aged
3	21	Young Adult
4	45	Middle Aged

```

## create column purchase_frequency_days

# df['frequency_of_purchases'].unique()

frequency_mapping = {
    'Fortnightly' : 14,
    'Weekly' : 7,
    'Monthly' : 30,
    'Every 3 Months':90,
    'Quarterly' : 90,
    'Annually' : 365,
    'Bi-Weekly' : 14
}
df['purchase_frequency_days'] = df['frequency_of_purchases'].map(frequency_mapping)
df[['frequency_of_purchases', 'purchase_frequency_days']]

```

	frequency_of_purchases	purchase_frequency_days
0	Fortnightly	14
1	Fortnightly	14
2	Weekly	7
3	Weekly	7
4	Annually	365
...	...	...
3895	Weekly	7
3896	Bi-Weekly	14
3897	Quarterly	90
3898	Weekly	7
3899	Quarterly	90
3900 rows × 2 columns		

```

## lets check for other columns
## discount_applied and promo_code_used
## whenever promo code is used, discount should be applied
## lets check if both column has same values, or is there any redundancy

```

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

```
## Now as it is confirmed that both columns have same values, we can drop one column
df = df.drop(columns=['promo_code_used'])
```

- Now lets load this dataset to PostgreSQL for further analysis
- For that first create database in PostgreSQL
- Then install psycopg-binary and sqlalchemy

Connect to PostgreSQL  
!pip install sqlalchemy psycopg2-binary  
from sqlalchemy import create\_engine

```
## step 1: connect to postgreSQL

username = "postgres"
password = "Password"
database = "customer_behavior"
host = "localhost"
port = "5432"

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

## step 2: Load datframe into postgreSQL
table_name = "customer"
df.to_sql(table_name, engine, if_exists='replace', index=False)

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