In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv("customer_shopping_behavior.csv")
```

In [3]:
```
df.head(5)
```

Out[3]:

	Customer ID	Age	Gender	Item Purchased	Category	Purchase Amount (USD)	Location	Size	Color	Season	Review Rating	Subscription Status	Shipping Type	Disc Apl
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	

In [4]:
```
df.describe(include='all')

#.describe() used to generate descriptive statistics only on Numerical Col so we need to write include='all' to get statistics
```

Out[4]:

	Customer ID	Age	Gender	Item Purchased	Category	Purchase Amount (USD)	Location	Size	Color	Season	Review Rating	Subscription Status
count	3900.000000	3900.000000	3900	3900	3900	3900.000000	3900	3900	3900	3900	3863.000000	3900
unique	NaN	NaN	2	25	4	NaN	50	4	25	4	NaN	2
top	NaN	NaN	Male	Blouse	Clothing	NaN	Montana	M	Olive	Spring	NaN	No
freq	NaN	NaN	2652	171	1737	NaN	96	1755	177	999	NaN	2847
mean	1950.500000	44.068462	NaN	NaN	NaN	59.764359	NaN	NaN	NaN	NaN	3.750065	NaN
std	1125.977353	15.207589	NaN	NaN	NaN	23.685392	NaN	NaN	NaN	NaN	0.716983	NaN
min	1.000000	18.000000	NaN	NaN	NaN	20.000000	NaN	NaN	NaN	NaN	2.500000	NaN
25%	975.750000	31.000000	NaN	NaN	NaN	39.000000	NaN	NaN	NaN	NaN	3.100000	NaN
50%	1950.500000	44.000000	NaN	NaN	NaN	60.000000	NaN	NaN	NaN	NaN	3.800000	NaN
75%	2925.250000	57.000000	NaN	NaN	NaN	81.000000	NaN	NaN	NaN	NaN	4.400000	NaN
max	3900.000000	70.000000	NaN	NaN	NaN	100.000000	NaN	NaN	NaN	NaN	5.000000	NaN

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
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
```

there are 37 null in Review Rating so we can take mean or median but taking mean may create problem so we'll take median (robust to outliers) but category wise . # why ? for an example if we take median for all category rating then it may gives us inaccurate result becoz rating is varry on category such as cloths # have different rating than footware category like shoes or some other product

In [6]:
```
df["Review Rating"] = df.groupby("Category")["Review Rating"].transform(lambda x: x.fillna(x.median()))
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
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 [8]:
```
df.columns
```

Out[8]:
```
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')
```

In [9]:
```
df.columns = df.columns.str.lower()

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

In [10]:
```
df = df.rename(columns={'purchase_amount_(usd)':'purchase_amount'})
```

In [11]:
```
df.columns
```

Out[11]:
```
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 [12]:
```
labels = ['Young Adult', 'Adult', 'Middle Aged', 'Senior']
df['age_group'] = pd.qcut(df['age'], q=4, labels = labels)
```

In [13]:
```
df[['age', 'age_group']].head(10)
```

Out[13]:

	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 [14]:
```
df['frequency_of_purchases']
```

Out[14]:
```
0       Fortnightly
1       Fortnightly
2            Weekly
3            Weekly
4          Annually
           ...
3895         Weekly
3896      Bi-Weekly
3897      Quarterly
3898         Weekly
3899      Quarterly
Name: frequency_of_purchases, Length: 3900, dtype: object
```

In [15]:
```
# creating anthoer feature (col) purchase_frequency_days


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

In [16]:
```
df[['purchase_frequency_days', 'frequency_of_purchases']].head(10)
```

Out[16]:

	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 [17]:
```
df.columns
```

Out[17]:
```
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', 'age_group',
       'purchase_frequency_days'],
      dtype='object')
```

In [18]:
```
df[['discount_applied', 'promo_code_used']].head(10)
```

Out[18]:

	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 [19]:
```
# both are looking exact same and if promo code is applied then discount is also applied but in many cases discount will apply
# expectional so we need to check both col are same or not, if both are same then we can drop one of them coz we dont need 2 c

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

Out[19]:
```
np.True_
```

In [20]:
```
# so we are getting True that mean both are same col so we can drop onr of them
df = df.drop('promo_code_used', axis=1)
```

In [21]:
```
df.columns
```

Out[21]:
```
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')
```

In [22]:
```
pip install psycopg2-binary sqlalchemy
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: psycopg2-binary in c:\users\ashwin\appdata\roaming\python\python313\site-packages (2.9.11)
Requirement already satisfied: sqlalchemy in c:\programdata\anaconda3\lib\site-packages (2.0.39)
Requirement already satisfied: greenlet!=0.4.17 in c:\programdata\anaconda3\lib\site-packages (from sqlalchemy) (3.1.1)
Requirement already satisfied: typing-extensions>=4.6.0 in c:\programdata\anaconda3\lib\site-packages (from sqlalchemy) (4.12.
2)
Note: you may need to restart the kernel to use updated packages.
```

Step I: Connect to PostgreSQL # Replace placeholders with your actual details from sqlalchemy import create_engine username = "postgres" # default user password = "0000" # the password you set during installation host = "localhost" # if running Locally port = "5432" # default PostgreSQL port database = "customer_behavior" # the database you created in pgAdmin engine = create_engine(f"postgresql+psycopg2://{username}:{password}@{host}:{port}/{database}") # Step 2: Load DataFrame into PostgreSQL table_name = "customer" # choose any 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}'.")

In [ ]: