Marketing insights for E-Commerce Company

Problem Statement:

A rapidly growing e-commerce company aims to transition from intuition-based marketing to a data-driven approach. By analyzing customer demographics, transaction data, marketing spend, and discount details from 2019, the company seeks to gain a comprehensive understanding of customer behavior. The objectives are to optimize marketing campaigns across various channels, leverage data insights to enhance customer retention, predict customer lifetime value, and ultimately drive sustainable revenue growth.

Dataset description: Transaction data has been provided from 1st Jan 2019 to 31st Dec 2019. The below datasets have been provided.

Online_Sales.csv: This file contains actual orders data (point of Sales data) at transaction level with the below variables.

1. CustomerID: Customer unique ID
2. Transaction_ID: Transaction Unique ID
3. Transaction_Date: Date of Transaction
4. Product_SKU: SKU ID – Unique Id for product
5. Product_Description: Product Description
6. Product_Cateogry: Product Category
7. Quantity: Number of items ordered
8. Avg_Price: Price per one quantity
9. Delivery_Charges: Charges for delivery
10. Coupon_Status: Any discount coupon applied

Customers_Data.csv: This file contains customer's demographics.

1. CustomerID: Customer Unique ID
2. Gender: Gender of customer
3. Location: Location of Customer
4. Tenure_Months: Tenure in Months

Discount_Coupon.csv: Discount coupons have been given for different categories in different months

1. Month: Discount coupon applied in that month
2. Product_Category: Product category
3. Coupon_Code: Coupon Code for given Category and given month
4. Discount_pct: Discount Percentage for given coupon

Marketing_Spend.csv: Marketing spend on both offline & online channels on day wise.

1. Date: Date
2. Offline_Spend: Marketing spend on offline channels like TV, Radio, NewsPapers, hoardings etc.
3. Online_Spend: Marketing spend on online channels like Google keywords, facebook etc.

Tax_Amount.csv: GST Details for given category

1. Product_Category: Product Category
2. GST: Percentage of GST

1. Data Cleaning and Preprocessing:

Step 01: Importing the libraries

```
Suggested code may be subject to a license | Shayl
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from scipy.stats import norm
from scipy import stats
import requests
import io
```

Step 02: Loading the dataset

```
[7] tax = pd.read_csv('/content/Tax_amount.csv')
    online_sales = pd.read_csv('/content/Online_Sales.csv')
    marketing_spend = pd.read_csv('/content/Marketing_Spend.csv')
    coupons = pd.read_csv('/content/Discount_Coupon.csv')
    customers = pd.read_csv('/content/Customers.csv')
```

Step 03: Null and duplicate check

a. Online Sales:

This file is a fact file which contains data for customers and their transactions related values.

```
[9] online_sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52924 entries, 0 to 52923
Data columns (total 10 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   CustomerID           52924 non-null  int64
 1   Transaction_ID       52924 non-null  int64
 2   Transaction_Date     52924 non-null  object
 3   Product_SKU          52924 non-null  object
 4   Product_Description  52924 non-null  object
 5   Product_Category     52924 non-null  object
 6   Quantity             52924 non-null  int64
 7   Avg_Price            52924 non-null  float64
 8   Delivery_Charges     52924 non-null  float64
 9   Coupon_Status        52924 non-null  object
dtypes: float64(2), int64(3), object(5)
memory usage: 4.0+ MB
```

This data set has 52924 records and 10 columns.

```
[17] online_sales['Transaction_Date'] = pd.to_datetime(online_sales['Transaction_Date'])
```

```
online_sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52924 entries, 0 to 52923
Data columns (total 10 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   CustomerID           52924 non-null  int64
 1   Transaction_ID       52924 non-null  int64
 2   Transaction_Date     52924 non-null  datetime64[ns]
 3   Product_SKU          52924 non-null  object
 4   Product_Description  52924 non-null  object
 5   Product_Category     52924 non-null  object
 6   Quantity             52924 non-null  int64
 7   Avg_Price            52924 non-null  float64
 8   Delivery_Charges     52924 non-null  float64
 9   Coupon_Status        52924 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(3), object(4)
memory usage: 4.0+ MB
```

Changed the data type of Transaction date from object to date time.

<u>Null Check:</u>

```
[13] for i in online_sales.columns:
       print(f'The column {i} has {sum(online_sales[i].isna())} null values.')
```

```
The column CustomerID has 0 null values.
The column Transaction_ID has 0 null values.
The column Transaction_Date has 0 null values.
The column Product_SKU has 0 null values.
The column Product_Description has 0 null values.
The column Product_Category has 0 null values.
The column Quantity has 0 null values.
The column Avg_Price has 0 null values.
The column Delivery_Charges has 0 null values.
The column Coupon_Status has 0 null values.
```

Duplicate Check:

```
[14] for i in online_sales.columns:
       print(f'The column {i} has {online_sales[i].nunique()} number of unique values.')
```

```
The column CustomerID has 1468 number of unique values.
The column Transaction_ID has 25061 number of unique values.
The column Transaction_Date has 365 number of unique values.
The column Product_SKU has 1145 number of unique values.
The column Product_Description has 404 number of unique values.
The column Product_Category has 20 number of unique values.
The column Quantity has 151 number of unique values.
The column Avg_Price has 546 number of unique values.
The column Delivery_Charges has 267 number of unique values.
The column Coupon_Status has 3 number of unique values.
```

b. Tax

This file has category wise GST values are present.

```
[20] tax.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 2 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Product_Category  20 non-null     object
 1   GST               20 non-null     object
dtypes: object(2)
memory usage: 448.0+ bytes
```

20 rows and 2 columns are present in this file.

```
[21] for i in tax.columns:
       print(f'The column {i} has {tax[i].nunique()} number of unique values.')
```

```
The column Product_Category has 20 number of unique values.
The column GST has 4 number of unique values.
```

```
[29] tax['gst_pct'] = pd.to_numeric(tax['GST'].str.replace('%', ''))
```

```
[30] tax.head()
```

	Product_Category	GST	gst_pct
0	Nest-USA	10%	10
1	Office	10%	10
2	Apparel	18%	18
3	Bags	18%	18
4	Drinkware	18%	18

Created a new column for calculation purpose.

c. marketing spends:

It has data for online and offline spends on a particular date.

```
[33] marketing_spend.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 365 entries, 0 to 364
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Date           365 non-null    object 
 1   Offline_Spend  365 non-null    int64  
 2   Online_Spend   365 non-null    float64
dtypes: float64(1), int64(1), object(1)
memory usage: 8.7+ KB
```

Let’s change the date column from object to date time.

```
[41] marketing_spend['Date'] = pd.to_datetime(marketing_spend['Date'])
     marketing_spend.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 365 entries, 0 to 364
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   Date           365 non-null    datetime64[ns]
 1   Offline_Spend  365 non-null    int64         
 2   Online_Spend   365 non-null    float64       
dtypes: datetime64[ns](1), float64(1), int64(1)
memory usage: 8.7 KB
```

Total 365 rows and 3 columns.

```
marketing_spend.describe()
```

	Offline_Spend	Online_Spend
count	365.000000	365.000000
mean	2843.561644	1905.880740
std	952.292448	808.856853
min	500.000000	320.250000
25%	2500.000000	1258.600000
50%	3000.000000	1881.940000
75%	3500.000000	2435.120000
max	5000.000000	4556.930000

Insights:

1. Mean offline spend is higher than online spend.

2. Min value of offline spend is 500 and maximum is 5000.

3. Min value of online spend is 320.25 and maximum is 4556.93.

d. coupons:

This data set has category wise monthly coupon codes and corresponding discount percentage.

```
[38] coupons.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 204 entries, 0 to 203
Data columns (total 4 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Month             204 non-null    object
 1   Product_Category  204 non-null    object
 2   Coupon_Code       204 non-null    object
 3   Discount_pct      204 non-null    int64
dtypes: int64(1), object(3)
memory usage: 6.5+ KB
```

Total 204 rows and 4 columns.

```
[39] for i in coupons.columns:
       print(f'The column {i} has {coupons[i].nunique()} number of unique values.')
```

```
The column Month has 12 number of unique values.
The column Product_Category has 17 number of unique values.
The column Coupon_Code has 48 number of unique values.
The column Discount_pct has 3 number of unique values.
```

No null values or duplicate values are present in the data set.

e. customers:

This data set has customer demographic data and tenure in months.

```
[43] customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1468 entries, 0 to 1467
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   CustomerID     1468 non-null   int64
 1   Gender         1468 non-null   object
 2   Location       1468 non-null   object
 3   Tenure_Months  1468 non-null   int64
dtypes: int64(2), object(2)
memory usage: 46.0+ KB
```

Total 1468 rows and 4 columns present in the data set with 0 null values.

```
[44] for i in customers.columns:
       print(f'The column {i} has {customers[i].nunique()} number of unique values.')
```

```
The column CustomerID has 1468 number of unique values.
The column Gender has 2 number of unique values.
The column Location has 5 number of unique values.
The column Tenure_Months has 49 number of unique values.
```

```
[46] customers.duplicated().sum()
```

```
0
```

This data set has no null and duplicate values present.

Step 04: Creating one complete data frame for analysis

```
df = pd.merge(online_sales, customers, on='CustomerID', how='left')
df.head()
```

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price	Delivery_Charges	Coupon_Status	Gender	Location	Tenure_Months
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	Used	M	Chicago	12
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	Used	M	Chicago	12
2	17850	16681	2019-01-01	GGOEGFKQ020399	Google Laptop and Cell Phone Stickers	Office	1	2.05	6.5	Used	M	Chicago	12
3	17850	16682	2019-01-01	GGOEGAAB010516	Google Men's 100% Cotton Short Sleeve Hero Tee...	Apparel	5	17.53	6.5	Not Used	M	Chicago	12
4	17850	16682	2019-01-01	GGOEGBJL013999	Google Canvas Tote Natural/Navy	Bags	1	16.50	6.5	Used	M	Chicago	12

```
[51] df = pd.merge(df, tax, on='Product_Category', how='left')
df.head()
```

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price	Delivery_Charges	Coupon_Status	Gender	Location	Tenure_Months	GST	gst_pct
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	Used	M	Chicago	12	10%	10
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	Used	M	Chicago	12	10%	10
2	17850	16681	2019-01-01	GGOEGFKQ020399	Google Laptop and Cell Phone Stickers	Office	1	2.05	6.5	Used	M	Chicago	12	10%	10
3	17850	16682	2019-01-01	GGOEGAAB010516	Google Men's 100% Cotton Short Sleeve Hero Tee...	Apparel	5	17.53	6.5	Not Used	M	Chicago	12	18%	18
4	17850	16682	2019-01-01	GGOEGBJL013999	Google Canvas Tote Natural/Navy	Bags	1	16.50	6.5	Used	M	Chicago	12	18%	18

```
df['Month_Value'] = pd.DatetimeIndex(df['Transaction_Date']).month_name()
df['Month_Value'] = df['Month_Value'].str[:3]
df.head()
```

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price	Delivery_Charges	Coupon_Status	Gender	Location	Tenure_Months	GST	gst_pct	Month_Value
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	Used	M	Chicago	12	10%	10	Jan
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	Used	M	Chicago	12	10%	10	Jan
2	17850	16681	2019-01-01	GGOEGFKQ020399	Google Laptop and Cell Phone Stickers	Office	1	2.05	6.5	Used	M	Chicago	12	10%	10	Jan
3	17850	16682	2019-01-01	GGOEGAAB010516	Google Men's 100% Cotton Short Sleeve Hero Tee...	Apparel	5	17.53	6.5	Not Used	M	Chicago	12	18%	18	Jan
4	17850	16682	2019-01-01	GGOEGBJL013999	Google Canvas Tote Natural/Navy	Bags	1	16.50	6.5	Used	M	Chicago	12	18%	18	Jan

```
[62] df = pd.merge(df, coupons, left_on=['Month_Value', 'Product_Category'], right_on = ['Month', 'Product_Category'], how='left')
df.head()
```

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price	Delivery_Charges	Coupon_Status	Gender	Location	Tenure_Months	GST	gst_pct	Month_Value	Month	Coupon
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	Used	M	Chicago	12	10%	10	Jan	Jan	EL
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	Used	M	Chicago	12	10%	10	Jan	Jan	EL
2	17850	16681	2019-01-01	GGOEGFKQ020399	Google Laptop and Cell Phone Stickers	Office	1	2.05	6.5	Used	M	Chicago	12	10%	10	Jan	Jan	(
3	17850	16682	2019-01-01	GGOEGAAB010516	Google Men's 100% Cotton Short Sleeve Hero Tee...	Apparel	5	17.53	6.5	Not Used	M	Chicago	12	18%	18	Jan	Jan	S/
4	17850	16682	2019-01-01	GGOEGBJL013999	Google Canvas Tote Natural/Navy	Bags	1	16.50	6.5	Used	M	Chicago	12	18%	18	Jan	Jan	.

```
[64] df.drop(['Month', 'GST'], axis=1, inplace=True)
df.head()
```

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price	Delivery_Charges	Coupon_Status	Gender	Location	Tenure_Months	gst_pct	Month_Value	Coupon_Code	Discoun
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	Used	M	Chicago	12	10	Jan	ELEC10	
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	Used	M	Chicago	12	10	Jan	ELEC10	
2	17850	16681	2019-01-01	GGOEGFKQ020399	Google Laptop and Cell Phone Stickers	Office	1	2.05	6.5	Used	M	Chicago	12	10	Jan	OFF10	
3	17850	16682	2019-01-01	GGOEGAAB010516	Google Men's 100% Cotton Short Sleeve Hero Tee...	Apparel	5	17.53	6.5	Not Used	M	Chicago	12	18	Jan	SALE10	
4	17850	16682	2019-01-01	GGOEGBJL013999	Google Canvas Tote Natural/Navy	Bags	1	16.50	6.5	Used	M	Chicago	12	18	Jan	AIO10	

Let's calculate one measure invoice value by given formula,

Invoice Value = ((Quantity * Avg_price) *(1 - Discount_pct) * (1 + GST)) + Delivery_Charges

```
[67] df['Invoice_Value'] = ((df['Quantity'] * df['Avg_Price']) *(1 - df['Discount_pct']/100) * (1 + df['gst_pct']/100)) + df['Delivery_Charges']
     df.head()
```

_ID	Transaction_Date	Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price	Delivery_Charges	Coupon_Status	Gender	Location	Tenure_Months	gst_pct	Month_Value	Coupon_Code	Discount_pct	Invoice_Value
5679	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	Used	M	Chicago	12	10	Jan	ELEC10	10.0	158.6729
5680	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	Used	M	Chicago	12	10	Jan	ELEC10	10.0	158.6729
5681	2019-01-01	GGOEGFKQ020399	Google Laptop and Cell Phone Stickers	Office	1	2.05	6.5	Used	M	Chicago	12	10	Jan	OFF10	10.0	8.5295
5682	2019-01-01	GGOEGAAB010516	Google Men's 100% Cotton Short Sleeve Hero Tee...	Apparel	5	17.53	6.5	Not Used	M	Chicago	12	18	Jan	SALE10	10.0	99.5843
5682	2019-01-01	GGOEGBJL013999	Google Canvas Tote Natural/Navy	Bags	1	16.50	6.5	Used	M	Chicago	12	18	Jan	AIO10	10.0	24.0230

After doing this merger we got some NaN value due to keys,

```
[115] df[df['Coupon_Code'].isna()]
```

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price	Delivery_Charges	Coupon_Status	Gender	Location	Tenure_Months	gst_pct	Month_Value	Coupon_Code	D
62	17850	16704	2019-01-01	GGOEYOBR078599	YouTube Luggage Tag	Fun	4	9.27	6.50	Used	M	Chicago	12	18	Jan	NaN	
95	14688	16742	2019-01-02	GGOEGBRD079699	25L Classic Rucksack	Backpacks	1	103.15	6.50	Clicked	F	New York	46	10	Jan	NaN	
157	18074	16782	2019-01-02	GGOEGOBC078699	Google Luggage Tag	Fun	1	7.42	6.50	Used	F	California	10	18	Jan	NaN	
178	16029	16800	2019-01-02	GGOEAOBH078799	Android Luggage Tag	Fun	2	7.42	6.50	Not Used	F	Washington DC	40	18	Jan	NaN	
193	16250	16812	2019-01-02	GGOEGDHG082499	Google 25 oz Clear Stainless Steel Bottle	Google	1	11.54	17.96	Clicked	F	California	30	10	Jan	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
44213	12472	42109	2019-10-30	GGOEGBRD079699	25L Classic Rucksack	Backpacks	1	79.99	6.00	Clicked	F	New Jersey	2	10	Oct	NaN	
45167	14911	42756	2019-11-07	GGOEGBRD079699	25L Classic Rucksack	Backpacks	1	79.99	6.00	Not Used	F	California	34	10	Nov	NaN	
45807	18125	43244	2019-11-12	GGOEGBRD079699	25L Classic Rucksack	Backpacks	1	99.99	6.00	Clicked	F	Chicago	3	10	Nov	NaN	
46239	17180	43537	2019-11-15	GGOEGBRD079699	25L Classic Rucksack	Backpacks	1	79.99	6.00	Used	F	Chicago	35	10	Nov	NaN	
46966	12377	44124	2019-11-21	GGOEGBRB079599	25L Classic Rucksack	Backpacks	1	99.99	6.00	Clicked	F	California	27	10	Nov	NaN	

400 rows × 18 columns

For this reason, invoice value is also coming as Nan for 400 rows. Let's treat this by replacing the Nans.

```
df['Discount_pct'].fillna(0, inplace=True)
df.head()
```

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price	Delivery_Charges	Coupon_Status	Gender	Location	Tenure_Months	gst_pct	Month_Value	Coupon_Code	Discoun
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	Used	M	Chicago	12	10	Jan	ELEC10	
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	Used	M	Chicago	12	10	Jan	ELEC10	
2	17850	16681	2019-01-01	GGOEGFKQ020399	Google Laptop and Cell Phone Stickers	Office	1	2.05	6.5	Used	M	Chicago	12	10	Jan	OFF10	
3	17850	16682	2019-01-01	GGOEGAAB010516	Google Men's 100% Cotton Short Sleeve Hero Tee...	Apparel	5	17.53	6.5	Not Used	M	Chicago	12	18	Jan	SALE10	
4	17850	16682	2019-01-01	GGOEGBJL013999	Google Canvas Tote Natural/Navy	Bags	1	16.50	6.5	Used	M	Chicago	12	18	Jan	AIO10	

Replacing discount percentage as 0 where ever there is no coupons available.

```
[117] df['Coupon_Code'].fillna('No Coupon', inplace=True)
      df[df['Coupon_Code'] == 'No Coupon'].head()
```

ID	Transaction_Date	Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price	Delivery_Charges	Coupon_Status	Gender	Location	Tenure_Months	gst_pct	Month_Value	Coupon_Code	Discount_pct	Invoice_Value
04	2019-01-01	GGOEYOBR078599	YouTube Luggage Tag	Fun	4	9.27	6.50	Used	M	Chicago	12	18	Jan	No Coupon	0.0	50.2544
42	2019-01-02	GGOEGBRD079699	25L Classic Rucksack	Backpacks	1	103.15	6.50	Clicked	F	New York	46	10	Jan	No Coupon	0.0	119.9650
82	2019-01-02	GGOEGOBC078699	Google Luggage Tag	Fun	1	7.42	6.50	Used	F	California	10	18	Jan	No Coupon	0.0	15.2556
00	2019-01-02	GGOEAOBH078799	Android Luggage Tag	Fun	2	7.42	6.50	Not Used	F	Washington DC	40	18	Jan	No Coupon	0.0	24.0112
12	2019-01-02	GGOEGDHG082499	Google 25 oz Clear Stainless Steel Bottle	Google	1	11.54	17.96	Clicked	F	California	30	10	Jan	No Coupon	0.0	30.6540

Change the coupon code as 'No Coupon'.

Step 05: Outlier Detection

Data Set 01: Tax

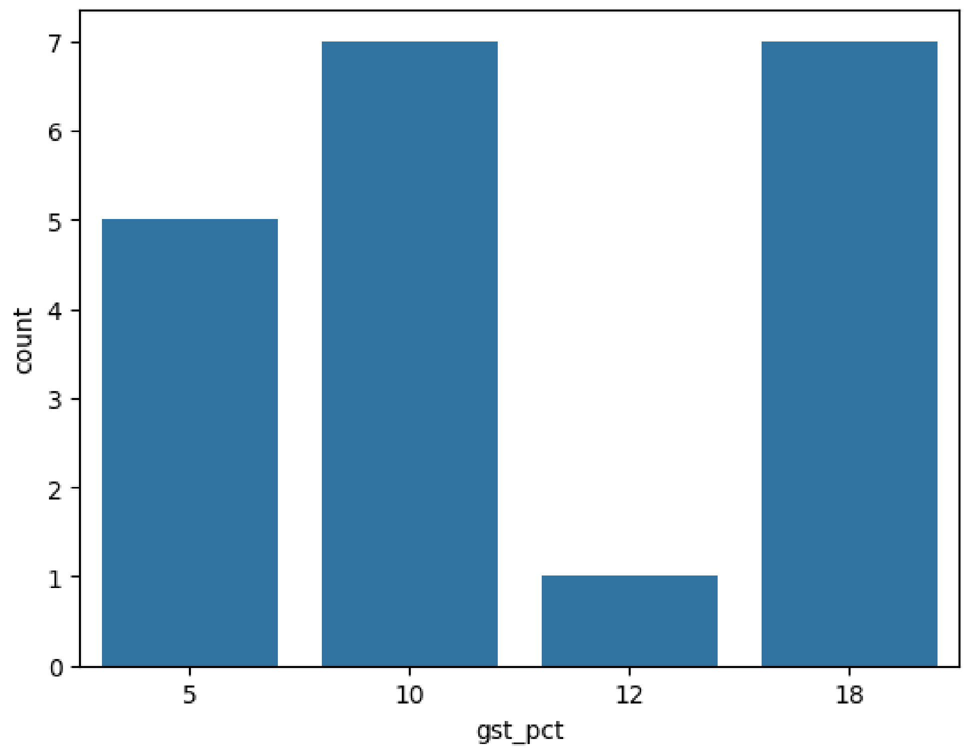
In this data set we have 20 unique categories and 4 GST allocated to those categories. No outlier is present in this data set.

```
tax['GST'].value_counts()
```

```
GST
10%    7
18%    7
5%     5
12%    1
Name: count, dtype: int64
```

```
[77] sns.countplot(x='gst_pct', data=tax)
```

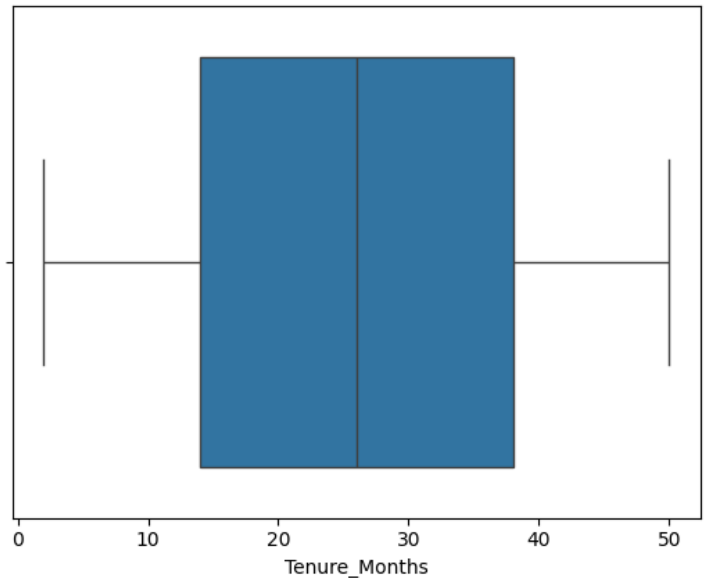
```
<Axes: xlabel='gst_pct', ylabel='count'>
```



Data Set 02: Customers

```
[79] sns.boxplot(x='Tenure_Months', data=customers)
```

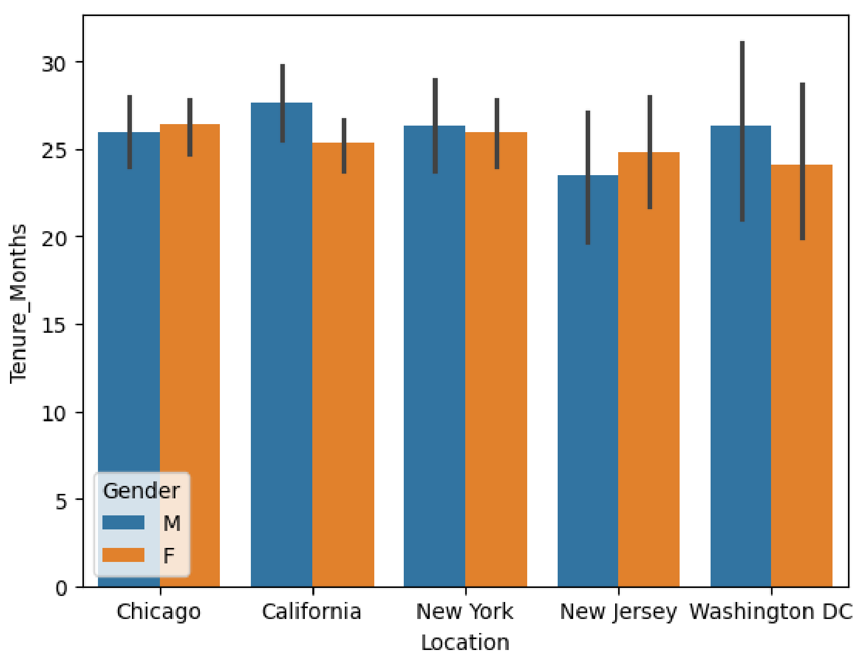
```
<Axes: xlabel='Tenure_Months'>
```



No outliers are there for Tenure Months.

```
sns.barplot(x='Location', y='Tenure_Months', hue = 'Gender', data=customers)
```

```
<Axes: xlabel='Location', ylabel='Tenure_Months'>
```



Location and gender wise customers are equally distributed for month tenure.

Data Set 03: Marketing Spends

```
[88] for i in ['Offline_Spend', 'Online_Spend']:
       q1 = np.percentile(marketing_spend[i], 25)
       q3 = np.percentile(marketing_spend[i], 75)
       iqr = q3 - q1
       lower_bound = q1 - 1.5 * iqr
       upper_bound = q3 + 1.5 * iqr
       outlier_count = len(marketing_spend[(marketing_spend[i] < lower_bound) | (marketing_spend[i] > upper_bound)])
       print(f'The 25%tile value for {i} is {q1}')
       print(f'The 75%tile value for {i} is {q3}')
       print(f'The IQR value for {i} is {iqr}')
       print(f'The lower bound for {i} is {lower_bound}')
       print(f'The upper bound for {i} is {upper_bound}')
       print(f'The number of outliers for {i} is {outlier_count}')
       print('-'*50)
```

```
The 25%tile value for Offline_Spend is 2500.0
The 75%tile value for Offline_Spend is 3500.0
The IQR value for Offline_Spend is 1000.0
The lower bound for Offline_Spend is 1000.0
The upper bound for Offline_Spend is 5000.0
The number of outliers for Offline_Spend is 14
--------------------------------------------------
The 25%tile value for Online_Spend is 1258.6
The 75%tile value for Online_Spend is 2435.12
The IQR value for Online_Spend is 1176.52
The lower bound for Online_Spend is -506.18000000000006
The upper bound for Online_Spend is 4199.9
The number of outliers for Online_Spend is 2
--------------------------------------------------
```

For offline spend there are 14 outliers and for online spend there are only 2 outliers present in this data set.

```
plt.figure(figsize=(20, 6))

plt.subplot(1, 2, 1)
sns.boxplot(x='Online_Spend', data=marketing_spend)

plt.subplot(1, 2, 2)
sns.boxplot(x='Offline_Spend', data=marketing_spend)

plt.show()
```

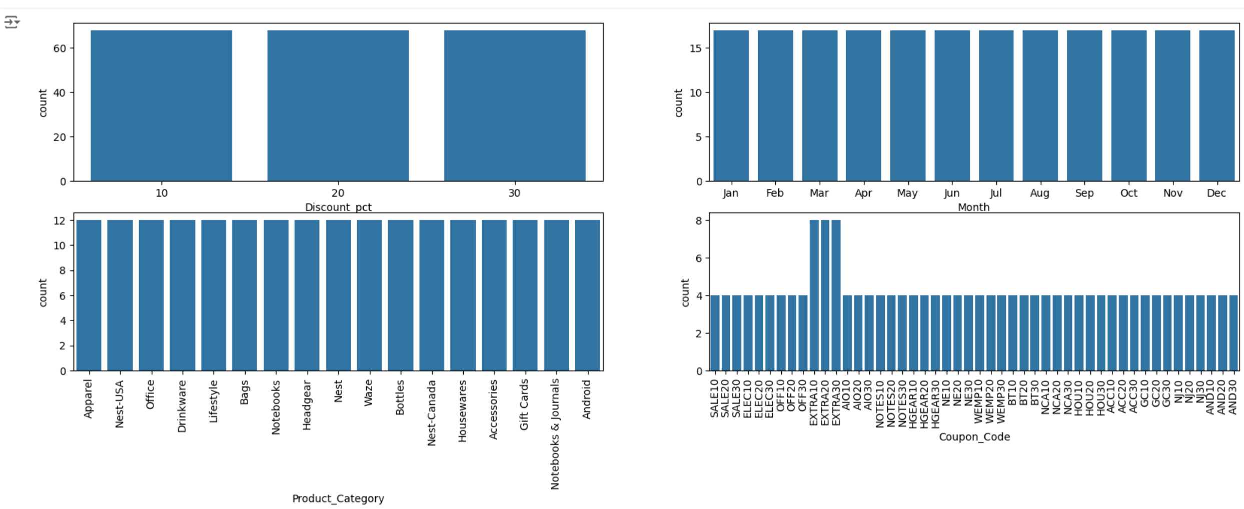
From box plot also we can see outliers are present in this data set.

Data Set 04: Coupons

```
[93] coupons['Discount_pct'].value_counts()
```

```
Discount_pct
10    68
20    68
30    68
Name: count, dtype: int64
```

```
plt.figure(figsize=(20, 6))

plt.subplot(2, 2, 1)
sns.countplot(x='Discount_pct', data=coupons)

plt.subplot(2, 2, 2)
sns.countplot(x='Month', data=coupons)

plt.subplot(2, 2, 3)
sns.countplot(x='Product_Category', data=coupons)
plt.xticks(rotation=90)

plt.subplot(2, 2, 4)
sns.countplot(x='Coupon_Code', data=coupons)
plt.xticks(rotation=90)

plt.show()
```



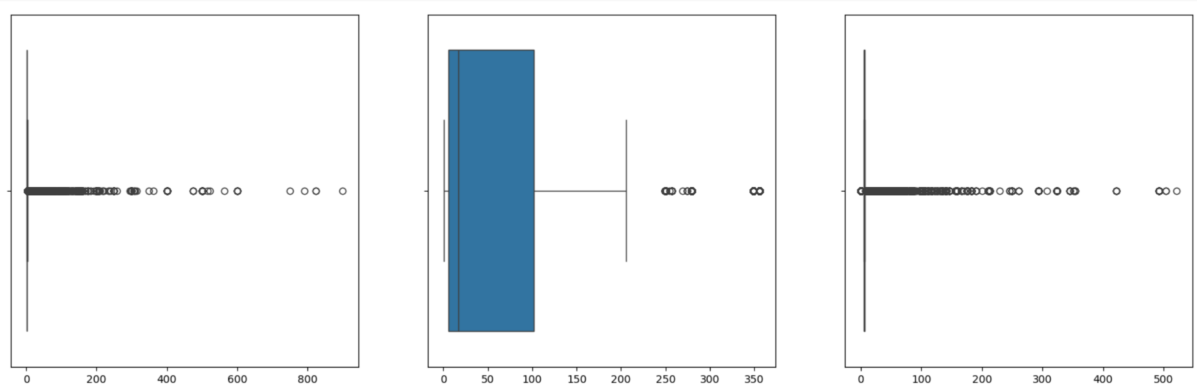
No outliers present in the data set.

Data Set 05: Online Sales

```
[103] for i in ['Quantity', 'Avg_Price', 'Delivery_Charges']:
        q1 = np.percentile(online_sales[i], 25)
        q3 = np.percentile(online_sales[i], 75)
        iqr = q3 - q1
        lower_bound = q1 - 1.5 * iqr
        upper_bound = q3 + 1.5 * iqr
        outlier_count = len(online_sales[(online_sales[i] < lower_bound) | (online_sales[i] > upper_bound)])
        print(f'The 25%tile value for {i} is {q1}')
        print(f'The 75%tile value for {i} is {q3}')
        print(f'The IQR value for {i} is {iqr}')
        print(f'The lower bound for {i} is {lower_bound}')
        print(f'The upper bound for {i} is {upper_bound}')
        print(f'The number of outliers for {i} is {outlier_count}')
        print('-'*50)
```

```
The 25%tile value for Quantity is 1.0
The 75%tile value for Quantity is 2.0
The IQR value for Quantity is 1.0
The lower bound for Quantity is -0.5
The upper bound for Quantity is 3.5
The number of outliers for Quantity is 8284
--------------------------------------------------
The 25%tile value for Avg_Price is 5.7
The 75%tile value for Avg_Price is 102.13
The IQR value for Avg_Price is 96.42999999999999
The lower bound for Avg_Price is -138.945
The upper bound for Avg_Price is 246.77499999999998
The number of outliers for Avg_Price is 728
--------------------------------------------------
The 25%tile value for Delivery_Charges is 6.0
The 75%tile value for Delivery_Charges is 6.5
The IQR value for Delivery_Charges is 0.5
The lower bound for Delivery_Charges is 5.25
The upper bound for Delivery_Charges is 7.25
The number of outliers for Delivery_Charges is 10243
--------------------------------------------------
```

```
[104] plt.figure(figsize=(20, 6))

      plt.subplot(1, 3, 1)
      sns.boxplot(x='Quantity', data=online_sales)

      plt.subplot(1, 3, 2)
      sns.boxplot(x='Avg_Price', data=online_sales)

      plt.subplot(1, 3, 3)
      sns.boxplot(x='Delivery_Charges', data=online_sales)

      plt.show()
```

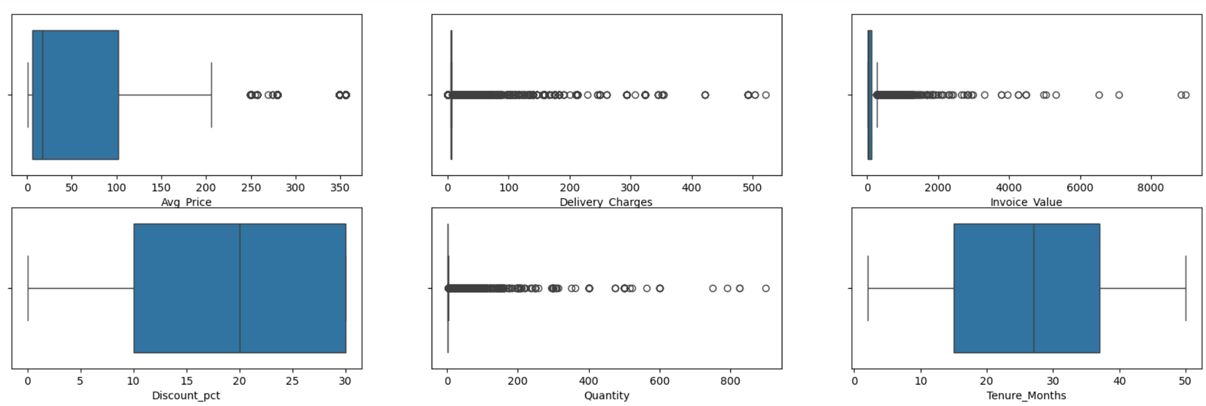


Data Frame: DF

```
for i in ['Avg_Price', 'Delivery_Charges', 'Invoice_Value', 'Discount_pct', 'Quantity', 'Tenure_Months']:
    q1 = np.percentile(df[i], 25)
    q3 = np.percentile(df[i], 75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    outlier_count = len(df[(df[i] < lower_bound) | (df[i] > upper_bound)])
    print(f'The 25%tile value for {i} is {q1}')
    print(f'The 75%tile value for {i} is {q3}')
    print(f'The IQR value for {i} is {iqr}')
    print(f'The lower bound for {i} is {lower_bound}')
    print(f'The upper bound for {i} is {upper_bound}')
    print(f'The number of outliers for {i} is {outlier_count}')
    print('-'*50)
```

```
The 25%tile value for Avg_Price is 5.7
The 75%tile value for Avg_Price is 102.13
The IQR value for Avg_Price is 96.42999999999999
The lower bound for Avg_Price is -138.945
The upper bound for Avg_Price is 246.77499999999998
The number of outliers for Avg_Price is 728
--------------------------------------------------
The 25%tile value for Delivery_Charges is 6.0
The 75%tile value for Delivery_Charges is 6.5
The IQR value for Delivery_Charges is 0.5
The lower bound for Delivery_Charges is 5.25
The upper bound for Delivery_Charges is 7.25
The number of outliers for Delivery_Charges is 10243
--------------------------------------------------
The 25%tile value for Invoice_Value is 18.54576
The 75%tile value for Invoice_Value is 123.4476
The IQR value for Invoice_Value is 104.90183999999999
The lower bound for Invoice_Value is -138.807
The upper bound for Invoice_Value is 280.80035999999996
The number of outliers for Invoice_Value is 2883
--------------------------------------------------
The 25%tile value for Discount_pct is 10.0
The 75%tile value for Discount_pct is 30.0
The IQR value for Discount_pct is 20.0
The lower bound for Discount_pct is -20.0
The upper bound for Discount_pct is 60.0
The number of outliers for Discount_pct is 0
--------------------------------------------------
The 25%tile value for Quantity is 1.0
The 75%tile value for Quantity is 2.0
The IQR value for Quantity is 1.0
The lower bound for Quantity is -0.5
The upper bound for Quantity is 3.5
The number of outliers for Quantity is 8284
--------------------------------------------------
The 25%tile value for Tenure_Months is 15.0
The 75%tile value for Tenure_Months is 37.0
The IQR value for Tenure_Months is 22.0
The lower bound for Tenure_Months is -18.0
The upper bound for Tenure_Months is 70.0
The number of outliers for Tenure_Months is 0
--------------------------------------------------
```

```python
plt.figure(figsize = (20, 6))

plt.subplot(2, 3, 1)
sns.boxplot(x='Avg_Price', data=df)

plt.subplot(2, 3, 2)
sns.boxplot(x='Delivery_Charges', data=df)

plt.subplot(2, 3, 3)
sns.boxplot(x='Invoice_Value', data=df)

plt.subplot(2, 3, 4)
sns.boxplot(x='Discount_pct', data=df)

plt.subplot(2, 3, 5)
sns.boxplot(x='Quantity', data=df)

plt.subplot(2, 3, 6)
sns.boxplot(x='Tenure_Months', data=df)

plt.show()
```



Insights:

1. Discount Pct and Tenure Month don't have any outliers.

2. Average price, delivery charges, invoice value, quantity all have outliers.

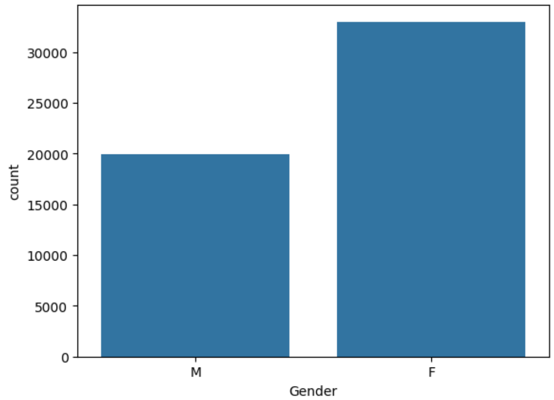
2. Exploratory Data Analysis (EDA):

Customer Acquisition & Retention: Analyze trends in customer acquisition and churn across different customer demographics (gender, location, tenure) and timeframes (monthly). Tools like time series analysis and segmentation can be helpful here.

Univariate Analysis:

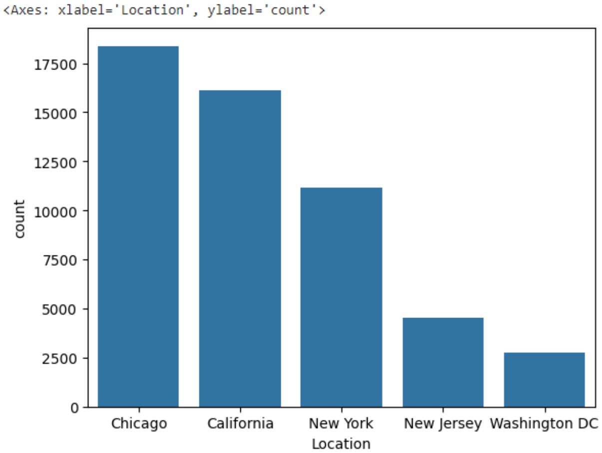
```
sns.countplot(x='Gender', data=df)
```

<Axes: xlabel='Gender', ylabel='count'>

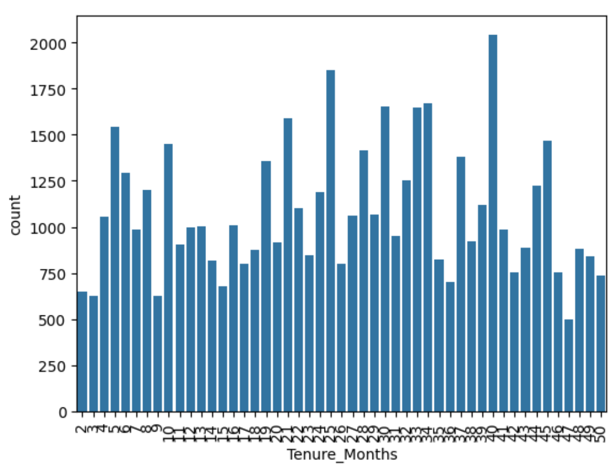



```
[77] sns.countplot(x='Location', data=df)
```

<Axes: xlabel='Location', ylabel='count'>




```
[80] sns.countplot(x='Tenure_Months', data=df)
     plt.xticks(rotation=90)
     plt.show()
```



```
[69] from operator import attrgetter

     # machine learning libraries
     from sklearn.preprocessing import StandardScaler
     from sklearn.cluster import KMeans
```

```
[70] df['order_month'] = df['Transaction_Date'].dt.to_period('M')
     df['cohort'] = df.groupby('CustomerID')['Transaction_Date'].transform('min').dt.to_period('M')
     df_cohort = df.groupby(['cohort', 'order_month']).agg(n_customers=('CustomerID', 'nunique')).reset_index(drop=False)
     df_cohort['period_number'] = (df_cohort.order_month - df_cohort.cohort).apply(attrgetter('n'))
     df_cohort.head()
```

	cohort	order_month	n_customers	period_number
0	2019-01	2019-01	215	0
1	2019-01	2019-02	13	1
2	2019-01	2019-03	24	2
3	2019-01	2019-04	34	3
4	2019-01	2019-05	23	4

```
cohort_pivot = df_cohort.pivot_table(index='cohort', columns='period_number', values='n_customers')
```

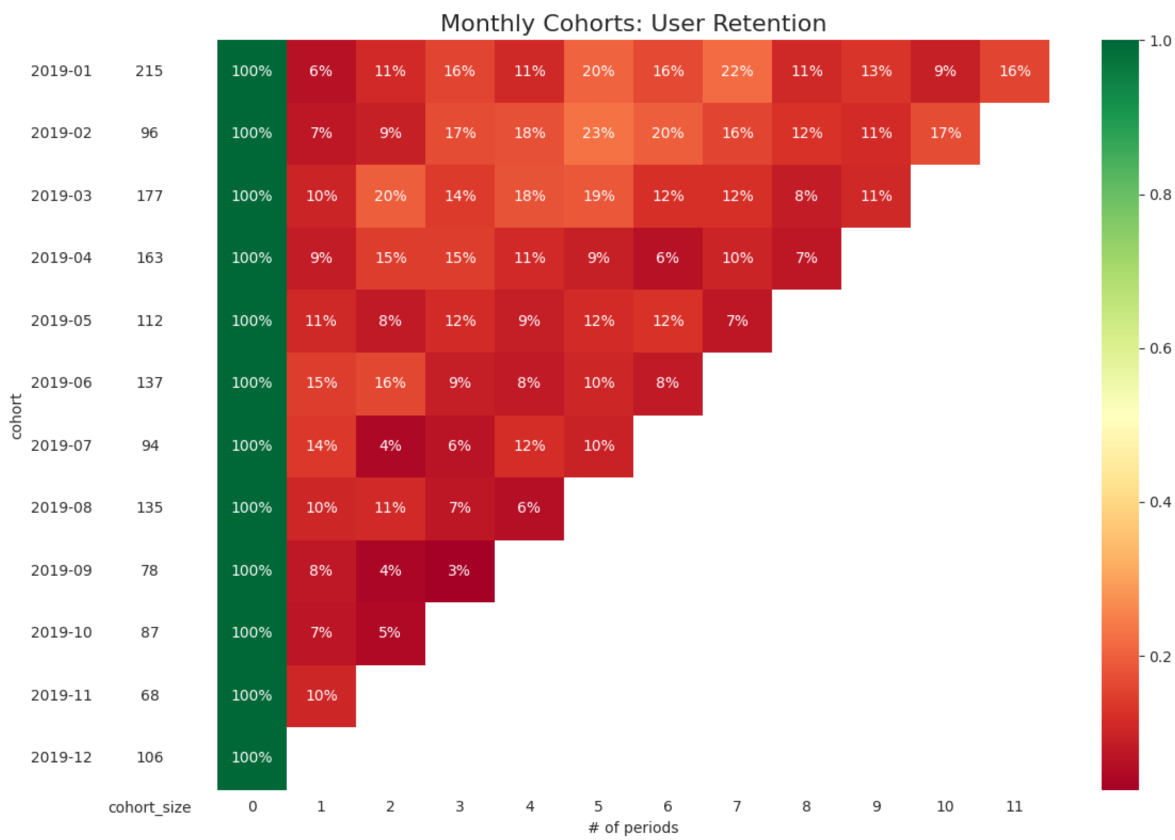
```
[72] cohort_pivot
```

period_number	0	1	2	3	4	5	6	7	8	9	10	11
cohort												
2019-01	215.0	13.0	24.0	34.0	23.0	44.0	35.0	47.0	23.0	28.0	20.0	34.0
2019-02	96.0	7.0	9.0	16.0	17.0	22.0	19.0	15.0	12.0	11.0	16.0	NaN
2019-03	177.0	18.0	35.0	25.0	32.0	33.0	22.0	22.0	15.0	19.0	NaN	NaN
2019-04	163.0	14.0	24.0	24.0	18.0	15.0	10.0	16.0	12.0	NaN	NaN	NaN
2019-05	112.0	12.0	9.0	13.0	10.0	13.0	14.0	8.0	NaN	NaN	NaN	NaN
2019-06	137.0	20.0	22.0	12.0	11.0	14.0	11.0	NaN	NaN	NaN	NaN	NaN
2019-07	94.0	13.0	4.0	6.0	11.0	9.0	NaN	NaN	NaN	NaN	NaN	NaN
2019-08	135.0	14.0	15.0	10.0	8.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2019-09	78.0	6.0	3.0	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2019-10	87.0	6.0	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2019-11	68.0	7.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2019-12	106.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
[73] cohort_size = cohort_pivot.iloc[:, 0]
     retention_matrix = cohort_pivot.divide(cohort_size, axis=0)
```

```
[74] import matplotlib.colors as mcolors
     with sns.axes_style("white"):
         fig, ax = plt.subplots(1, 2, figsize=(12, 8), sharey=True, gridspec_kw={'width_ratios': [1, 11]})

         # retention matrix
         sns.heatmap(retention_matrix,
                     mask=retention_matrix.isnull(),
                     annot=True,
                     fmt='.0%',
                     cmap='RdYlGn',
                     ax=ax[1])
         ax[1].set_title('Monthly Cohorts: User Retention', fontsize=16)
         ax[1].set(xlabel='# of periods',
                   ylabel='')

         # cohort size
         cohort_size_df = pd.DataFrame(cohort_size).rename(columns={0: 'cohort_size'})
         white_cmap = mcolors.ListedColormap(['white'])
         sns.heatmap(cohort_size_df,
                     annot=True,
                     cbar=False,
                     fmt='g',
                     cmap=white_cmap,
                     ax=ax[0])

         fig.tight_layout()
```

[74]

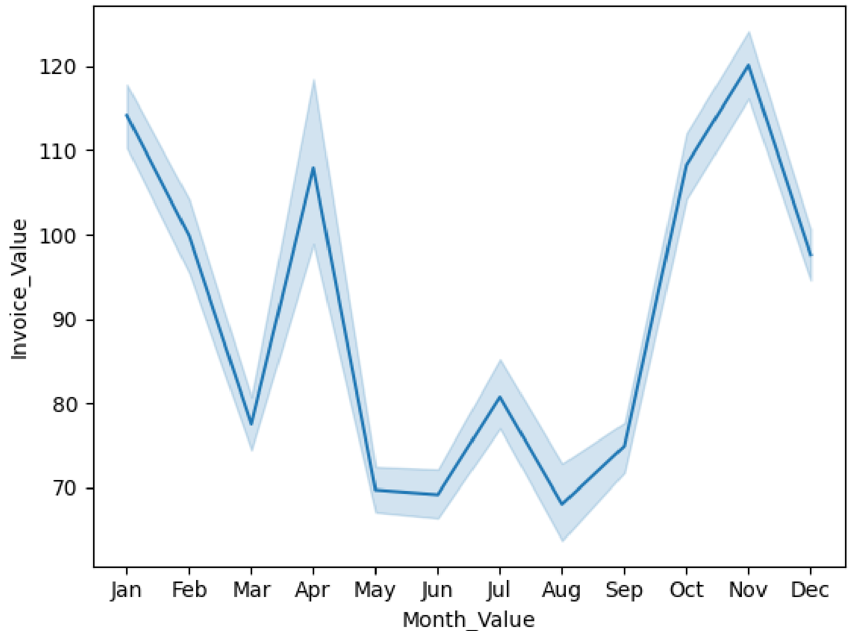


Seasonality & Trends: Identify seasonal trends and patterns in sales data across different timeframes (month, week, day) to inform future marketing strategies.

Monthly Sales:

```
[81] sns.lineplot(x='Month_Value', y='Invoice_Value', data=df)
```

```
<Axes: xlabel='Month_Value', ylabel='Invoice_Value'>
```



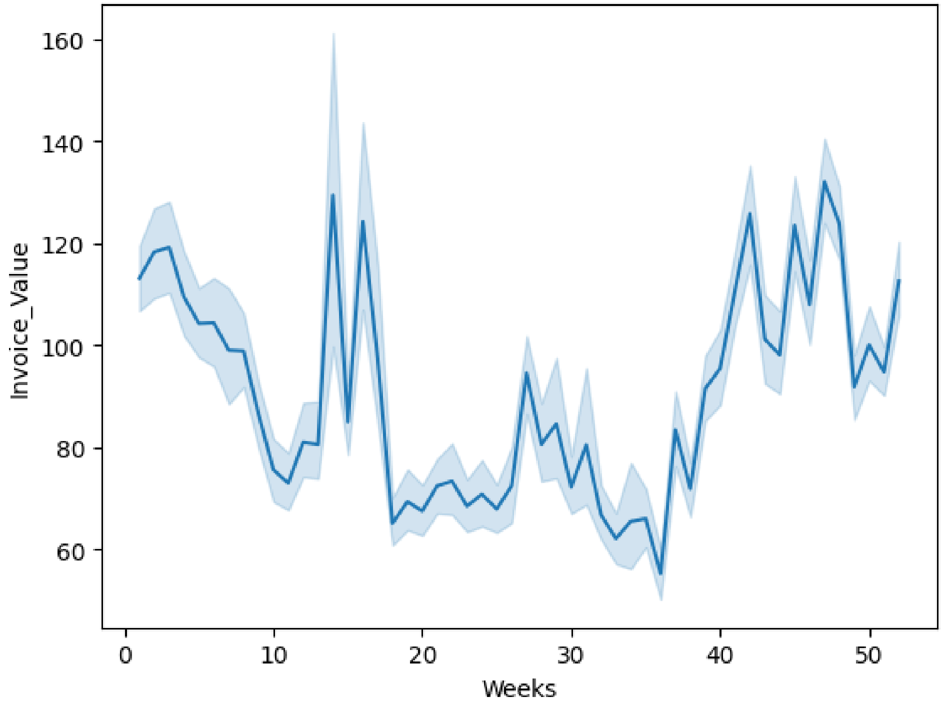
Weekly Sales:

```
[86] df['Weeks'] = df['Transaction_Date'].dt.isocalendar().week
     df.head()
```

Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price	Delivery_Charges	Coupon_Status	...	Location	Tenure_Months	gst_pct	Month_Value	Coupon_Code	Discount_pct	Invoice_Value	order_month	cohort	Weeks
GOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	Used	...	Chicago	12	10	Jan	ELEC10	10.0	158.6729	2019-01	2019-01	1
GOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	Used	...	Chicago	12	10	Jan	ELEC10	10.0	158.6729	2019-01	2019-01	1
GOEGFKQ020399	Google Laptop and Cell Phone Stickers	Office	1	2.05	6.5	Used	...	Chicago	12	10	Jan	OFF10	10.0	8.5295	2019-01	2019-01	1
GOEGAAB010516	Google Men's 100% Cotton Short Sleeve Hero Tee...	Apparel	5	17.53	6.5	Not Used	...	Chicago	12	18	Jan	SALE10	10.0	99.5843	2019-01	2019-01	1
GOEGBJL013999	Google Canvas Tote Natural/Navy	Bags	1	16.50	6.5	Used	...	Chicago	12	18	Jan	AIO10	10.0	24.0230	2019-01	2019-01	1

```
sns.lineplot(x='Weeks', y='Invoice_Value', data=df)
```

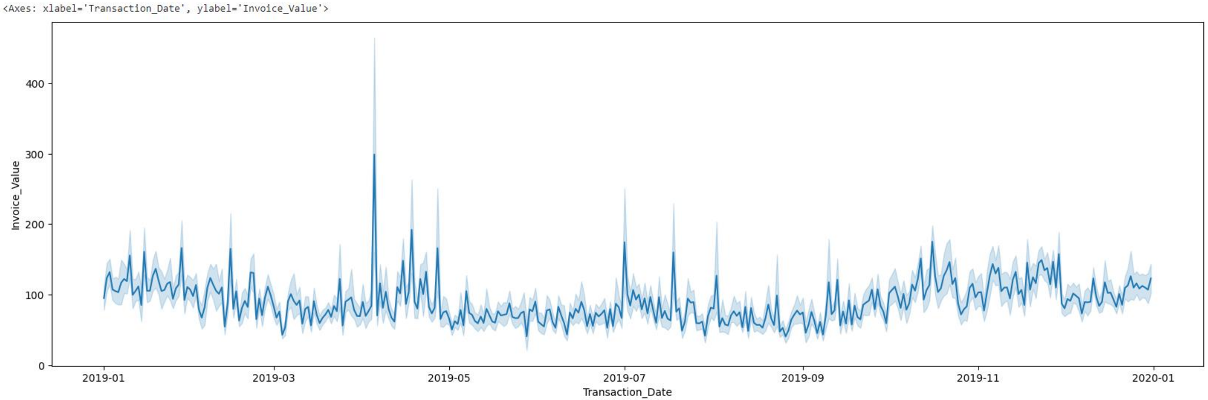
```
<Axes: xlabel='Weeks', ylabel='Invoice_Value'>
```



<u>Daily Sales:</u>

```
[90] plt.figure(figsize=(20, 6))

     sns.lineplot(x = 'Transaction_Date', y = 'Invoice_Value', data = df)
```

```
<Axes: xlabel='Transaction_Date', ylabel='Invoice_Value'>
```



Insights:

From above we can see the sales is high during Jan, Apr, Oct to Dec. Other months it is low.

Calculate key performance indicators (KPIs): like revenue, number of orders, and average order value across various dimensions (category, month, week, day).

1. Categorical Measures:

```
[97] df_cat = df[['Product_Category', 'Invoice_Value', 'Transaction_ID', 'Avg_Price']].groupby('Product_Category').agg({'Invoice_Value': 'sum', 'Transaction_ID' : 'count', 'Avg_Price': 'mean'}).reset_index()
     df_cat.rename(columns={'Transaction_ID': 'order_count', 'Invoice_Value' : 'revenue', 'Avg_Price' : 'Avg_order_value'}, inplace=True)
     df_cat
```

	Product_Category	revenue	order_count	Avg_order_value
0	Accessories	9.277126e+03	234	8.211068
1	Android	9.860494e+02	43	15.903488
2	Apparel	7.354504e+05	18126	19.788995
3	Backpacks	1.081288e+04	89	80.046404
4	Bags	1.688531e+05	1882	29.830797
5	Bottles	9.309917e+03	268	3.437201
6	Drinkware	2.402678e+05	3483	10.696893
7	Fun	8.994542e+03	160	6.743812
8	Gift Cards	1.757481e+04	159	111.363270
9	Google	1.316881e+04	105	16.446190
10	Headgear	5.345419e+04	771	15.879624
11	Housewares	6.372834e+03	122	2.060574
12	Lifestyle	1.145590e+05	3092	3.860078
13	More Bags	3.973113e+03	46	19.776957
14	Nest	4.399770e+05	2198	194.221074
15	Nest-Canada	6.554575e+04	317	157.243249
16	Nest-USA	2.351316e+06	14013	124.331850
17	Notebooks & Journals	1.093681e+05	749	11.758505
18	Office	3.440001e+05	6513	3.770012
19	Waze	1.125057e+04	554	6.607852

2. Monthly Measures:

```
[98] df_mnth = df[['Month_Value', 'Invoice_Value', 'Transaction_ID', 'Avg_Price']].groupby('Month_Value').agg({'Invoice_Value': 'sum', 'Transaction_ID' : 'count', 'Avg_Price': 'mean'}).reset_index()
     df_mnth.rename(columns={'Transaction_ID': 'order_count', 'Invoice_Value' : 'revenue', 'Avg_Price' : 'Avg_order_value'}, inplace=True)
     df_mnth
```

	Month_Value	revenue	order_count	Avg_order_value
0	Apr	447999.19523	4150	42.660465
1	Aug	418160.56704	6150	34.743348
2	Dec	439530.03015	4502	80.763678
3	Feb	327896.56020	3284	53.171443
4	Jan	463883.05705	4063	61.756055
5	Jul	423982.34361	5251	38.078315
6	Jun	289830.32931	4193	44.192690
7	Mar	336805.20383	4346	45.055541
8	May	318556.30056	4572	39.122417
9	Nov	475902.15336	3961	86.006503
10	Oct	450837.46255	4164	64.757985
11	Sep	321128.35638	4288	50.030893

3. Weekly Measures:

```
[100] df_wk = df[['Weeks', 'Invoice_Value', 'Transaction_ID', 'Avg_Price']].groupby('Weeks').agg({'Invoice_Value': 'sum', 'Transaction_ID' : 'count', 'Avg_Price': 'mean'}).reset_index()
      df_wk.rename(columns={'Transaction_ID': 'order_count', 'Invoice_Value' : 'revenue', 'Avg_Price' : 'Avg_order_value'}, inplace=True)
      df_wk.head()
```

	Weeks	revenue	order_count	Avg_order_value
0	1	119476.36561	1056	67.205549
1	2	98081.41564	829	59.399686
2	3	100403.60143	842	66.164715
3	4	103231.57870	943	59.787434
4	5	96555.27441	926	57.693110

4. Daily Measures:

```
[101] df_daily = df[['Transaction_Date', 'Invoice_Value', 'Transaction_ID', 'Avg_Price']].groupby('Transaction_Date').agg({'Invoice_Value': 'sum', 'Transaction_ID' : 'count', 'Avg_Price': 'mean'}).reset_index()
      df_daily.rename(columns={'Transaction_ID': 'order_count', 'Invoice_Value' : 'revenue', 'Avg_Price' : 'Avg_order_value'}, inplace=True)
      df_daily.head()
```

	Transaction_Date	revenue	order_count	Avg_order_value
0	2019-01-01	8489.73148	89	58.237753
1	2019-01-02	14244.70418	115	78.179478
2	2019-01-03	27379.80059	207	74.534638
3	2019-01-04	18185.88125	169	65.115325
4	2019-01-05	19884.09018	189	49.702116

Marketing Spend & Revenue: Calculate revenue, marketing spends, and delivery charges by month to understand their correlation.

```
df_mon = df[['Transaction_Date', 'Delivery_Charges', 'Invoice_Value']].groupby('Transaction_Date').agg({'Invoice_Value': 'sum', 'Delivery_Charges' : 'sum'}).reset_index()
df_mon.rename(columns={'Invoice_Value' : 'revenue'}, inplace=True)
df_spend = pd.merge(df_mon, marketing_spend, right_on=['Date'], left_on = ['Transaction_Date'], how='left')
df_spend.drop('Date', axis=1, inplace=True)
df_spend['Month_Value'] = df_spend['Transaction_Date'].dt.month_name().str[:3]
df_spend.head()
```

	Transaction_Date	revenue	Delivery_Charges	Offline_Spend	Online_Spend	Month_Value
0	2019-01-01	8489.73148	1082.23	4500	2424.50	Jan
1	2019-01-02	14244.70418	872.00	4500	3480.36	Jan
2	2019-01-03	27379.80059	3650.24	4500	1576.38	Jan
3	2019-01-04	18185.88125	1501.94	4500	2928.55	Jan
4	2019-01-05	19884.09018	2411.29	4500	4055.30	Jan

```
df_monthly_spend = df_spend.groupby('Month_Value').agg({'revenue': 'sum', 'Delivery_Charges': 'sum', 'Offline_Spend' : 'sum', 'Online_Spend' : 'sum'}).reset_index()
df_monthly_spend.head()
```

	Month_Value	revenue	Delivery_Charges	Offline_Spend	Online_Spend
0	Apr	447999.19523	41481.74	96000	61026.83
1	Aug	418160.56704	61099.57	85500	57404.15
2	Dec	439530.03015	37881.99	122000	76648.75
3	Feb	327896.56020	49216.60	81300	55807.92
4	Jan	463883.05705	59242.32	96600	58328.95

```
[108] df_monthly_spend[['revenue', 'Delivery_Charges', 'Offline_Spend', 'Online_Spend']].corr()
```

	revenue	Delivery_Charges	Offline_Spend	Online_Spend
revenue	1.000000	0.024401	0.586158	0.621804
Delivery_Charges	0.024401	1.000000	-0.243276	-0.462752
Offline_Spend	0.586158	-0.243276	1.000000	0.879841
Online_Spend	0.621804	-0.462752	0.879841	1.000000

Recommendations:

1. Develop targeted marketing campaigns tailored to each segment. For example, offer discounts on products frequently purchased by a particular segment or send personalized emails highlighting new arrivals in their favorite categories.

2. Allocate budget to high-performing campaigns predicted to yield the highest ROI. Test different marketing messages and channels to identify the most effective combinations.

3. Invest more in acquiring and retaining high CLV customers. Provide loyalty programs, special offers, and exclusive deals to these customers to enhance their experience and encourage repeat purchases.

4. Implement personalized product recommendations on the website and in email marketing. Use dynamic content to show relevant products and promotions to individual customers.

5. Address common pain points and improve product offerings based on feedback. Respond to negative reviews and social media comments promptly to demonstrate commitment to customer satisfaction.

6. Implement retention strategies for at-risk customers, such as personalized offers, re-engagement emails, and loyalty rewards. Monitor the effectiveness of these strategies and adjust based on performance data.

7. Use machine learning algorithms to identify products frequently bought together and suggest these combinations to customers. Display complementary products during the checkout process and in post-purchase emails.

8. Adjust prices in real-time to reflect changes in demand and maximize profit margins. Test different pricing strategies to find the optimal balance between volume and margin.

9. Use demand forecasting models to predict future sales and adjust inventory accordingly. Implement just-in-time inventory practices to reduce holding costs and improve cash flow.