```
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
```

```
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')
```

```
online_sales.head()
```

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Descripti
0	17850	16679	1/1/2019	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
1	17850	16680	1/1/2019	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
					Google Laptop a

Next steps: Generate code with online_sales View recommended plots

```
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
```

```
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.
```

```
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.
```

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

```
0       2019-01-01
1       2019-01-01
2       2019-01-01
3       2019-01-01
4       2019-01-01
           ...
52919   2019-12-31
52920   2019-12-31
52921   2019-12-31
52922   2019-12-31
52923   2019-12-31
Name: Transaction_Date, Length: 52924, dtype: datetime64[ns]
```

```
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
```

```
tax.head()
```

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

Next steps: Generate code with tax View recommended plots

```
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
```

```
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.
```

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

```
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

Next steps: Generate code with `tax` View recommended plots

```
tax.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Product_Category  20 non-null     object
 1   GST               20 non-null     object
 2   gst_pct           20 non-null     int64 
dtypes: int64(1), object(2)
memory usage: 608.0+ bytes
```

```
marketing_spend.head()
```

	Date	Offline_Spend	Online_Spend
0	1/1/2019	4500	2424.50
1	1/2/2019	4500	3480.36
2	1/3/2019	4500	1576.38
3	1/4/2019	4500	2928.55
4	1/5/2019	4500	4055.30

Next steps: Generate code with `marketing_spend` View recommended plots

```
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
```

```
marketing_spend.isnull().sum()
```

```
Date             0
Offline_Spend    0
Online_Spend     0
dtype: int64
```

```
marketing_spend.duplicated().sum()
```

```
0
```

```
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

```
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
```

```
coupons.head()
```

	Month	Product_Category	Coupon_Code	Discount_pct
0	Jan	Apparel	SALE10	10
1	Feb	Apparel	SALE20	20
2	Mar	Apparel	SALE30	30
3	Jan	Nest-USA	ELEC10	10
4	Feb	Nest-USA	ELEC20	20

Next steps: Generate code with coupons View recommended plots

```
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
```

```
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.
```

```
customers.head()
```

	CustomerID	Gender	Location	Tenure_Months
0	17850	M	Chicago	12
1	13047	M	California	43
2	12583	M	Chicago	33
3	13748	F	California	30
4	15100	M	California	49

Next steps: Generate code with customers | View recommended plots

```
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
```

```
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.
```

```
customers.duplicated().sum()
```

```
0
```

```
online_sales.head()
```

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Descripti
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
					Google Laptop a

Next steps: Generate code with online_sales | View recommended plots

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

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Descripti
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
					Google Laptop a

Next steps: Generate code with df | View recommended plots

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

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Descripti
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
2	17850	16681	2019-01-01	GGOEGFKQ020399	Google Laptop a Cell Phone Sticke

Next steps: Generate code with df | View recommended plots

```
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_Descripti
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
2	17850	16681	2019-01-01	GGOEGFKQ020399	Google Laptop a Cell Phone Sticke
3	17850	16682	2019-01-01	GGOEGAAB010516	Google Men's 10(Cotton Short Slee

Next steps: Generate code with df View recommended plots

```
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_Descripti
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
2	17850	16681	2019-01-01	GGOEGFKQ020399	Google Laptop a Cell Phone Sticke
3	17850	16682	2019-01-01	GGOEGAAB010516	Google Men's 10(Cotton Short Slee Hero Tee
4	17850	16682	2019-01-01	GGOEGBJL013999	Google Canvas Tc Natural/Na

Next steps: Generate code with df View recommended plots

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

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Descripti
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
2	17850	16681	2019-01-01	GGOEGFKQ020399	Google Laptop a Cell Phone Sticke
3	17850	16682	2019-01-01	GGOEGAAB010516	Google Men's 10(Cotton Short Slee Hero Tee
4	17850	16682	2019-01-01	GGOEGBJL013999	Google Canvas Tc Natural/Na

Next steps: Generate code with df View recommended plots

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

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Descripti
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
2	17850	16681	2019-01-01	GGOEGFKQ020399	Google Laptop a Cell Phone Sticke
3	17850	16682	2019-01-01	GGOEGAAB010516	Google Men's 10(Cotton Short Slee Hero Tee
4	17850	16682	2019-01-01	GGOEGBJL013999	Google Canvas To Natural/Na

Next steps: Generate code with df View recommended plots

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

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Descripti
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
2	17850	16681	2019-01-01	GGOEGFKQ020399	Google Laptop a Cell Phone Sticke
3	17850	16682	2019-01-01	GGOEGAAB010516	Google Men's 10(Cotton Short Slee Hero Tee
4	17850	16682	2019-01-01	GGOEGBJL013999	Google Canvas To Natural/Na

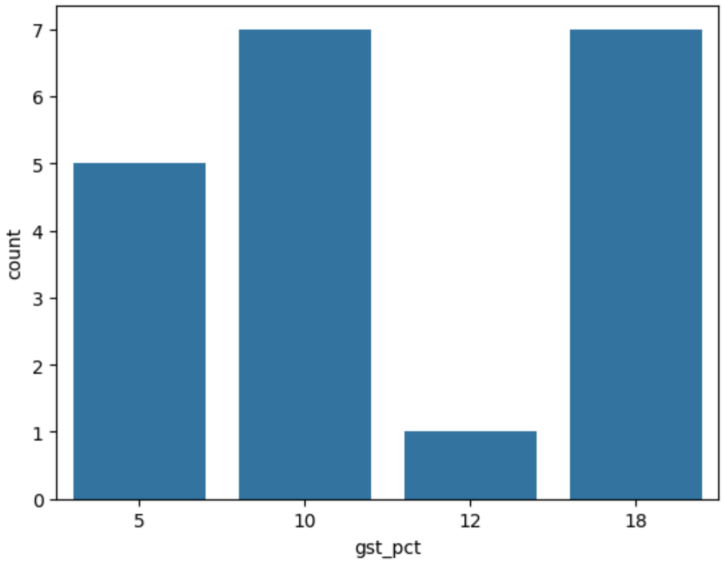
Next steps: Generate code with df View recommended plots

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

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

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

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



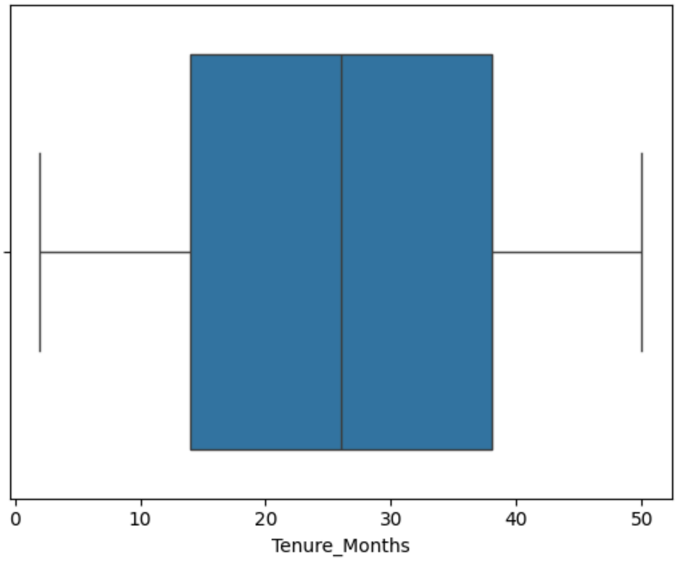
```
customers.head()
```

	CustomerID	Gender	Location	Tenure_Months
0	17850	M	Chicago	12
1	13047	M	California	43
2	12583	M	Chicago	33
3	13748	F	California	30
4	15100	M	California	49

Next steps: Generate code with customers View recommended plots

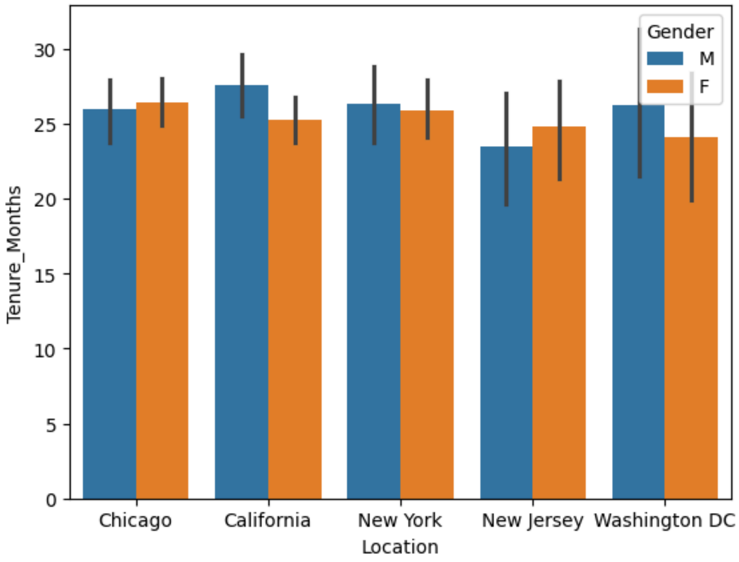
```
s.boxplot(x='Tenure_Months', data=customers)
```

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



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

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



```
marketing_spend.head()
```

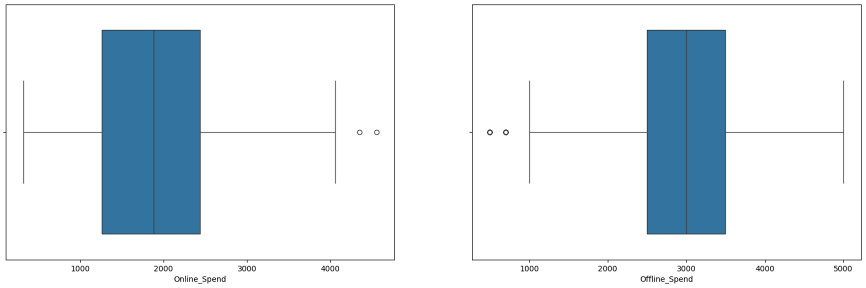
	Date	Offline_Spend	Online_Spend
0	2019-01-01	4500	2424.50
1	2019-01-02	4500	3480.36
2	2019-01-03	4500	1576.38
3	2019-01-04	4500	2928.55
4	2019-01-05	4500	4055.30

Next steps: Generate code with marketing_spend View recommended plots

```
r 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
--------------------------------------------------
```

```
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()
```



```
coupons.head()
```

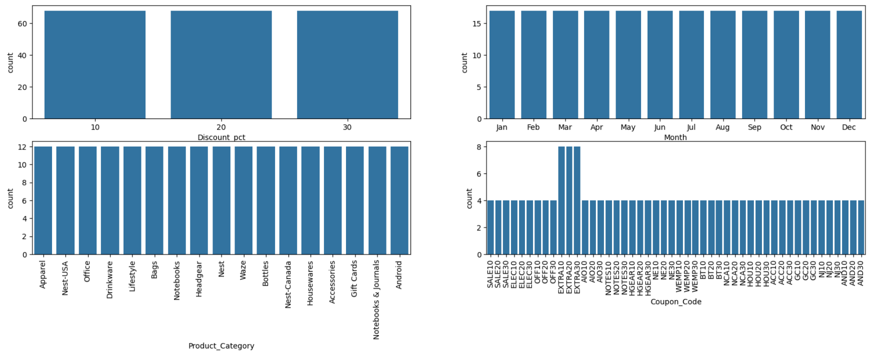
	Month	Product_Category	Coupon_Code	Discount_pct
0	Jan	Apparel	SALE10	10
1	Feb	Apparel	SALE20	20
2	Mar	Apparel	SALE30	30
3	Jan	Nest-USA	ELEC10	10
4	Feb	Nest-USA	ELEC20	20

Next steps: Generate code with coupons View recommended plots

```
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()
```



```
online_sales.head()
```

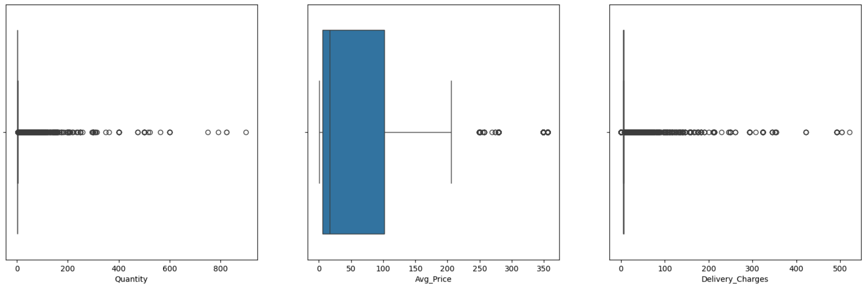
	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Descripti
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
					Google Laptop a

Next steps: Generate code with online_sales View recommended plots

```
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
--------------------------------------------------
```

```
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()
```



```
df.head()
```

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Descripti
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
2	17850	16681	2019-01-01	GGOEGFKQ020399	Google Laptop a Cell Phone Sticke
3	17850	16682	2019-01-01	GGOEGAAB010516	Google Men's 10(Cotton Short Slee Hero Tee
4	17850	16682	2019-01-01	GGOEGBJL013999	Google Canvas T(Natural/Na

Next steps: Generate code with df View recommended plots

```
for i in df.columns:
  print(f'Column {i} has {df[i].isna().sum()} number of null values.')
```

```
Column CustomerID has 0 number of null values.
Column Transaction_ID has 0 number of null values.
Column Transaction_Date has 0 number of null values.
Column Product_SKU has 0 number of null values.
Column Product_Description has 0 number of null values.
Column Product_Category has 0 number of null values.
Column Quantity has 0 number of null values.
Column Avg_Price has 0 number of null values.
Column Delivery_Charges has 0 number of null values.
Column Coupon_Status has 0 number of null values.
Column Gender has 0 number of null values.
Column Location has 0 number of null values.
Column Tenure_Months has 0 number of null values.
Column gst_pct has 0 number of null values.
Column Month_Value has 0 number of null values.
Column Coupon_Code has 400 number of null values.
Column Discount_pct has 0 number of null values.
Column Invoice_Value has 0 number of null values.
```

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

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Descr
62	17850	16704	2019-01-01	GGOEYOBR078599	YouTube Lugg
95	14688	16742	2019-01-02	GGOEGBRD079699	25L Classic R
157	18074	16782	2019-01-02	GGOEGOBC078699	Google Lugg
178	16029	16800	2019-01-02	GGOEAOBH078799	Android Lugg
193	16250	16812	2019-01-02	GGOEGDHG082499	Google 25 Stainless Ste
...	...	...	...	...	
44213	12472	42109	2019-10-30	GGOEGBRD079699	25L Classic R
45167	14911	42756	2019-11-07	GGOEGBRD079699	25L Classic R
45807	18125	43244	2019-11-12	GGOEGBRD079699	25L Classic R
46239	17180	43537	2019-11-15	GGOEGBRD079699	25L Classic R
46966	12377	44124	2019-11-21	GGOEGBRB079599	25L Classic R

400 rows × 18 columns

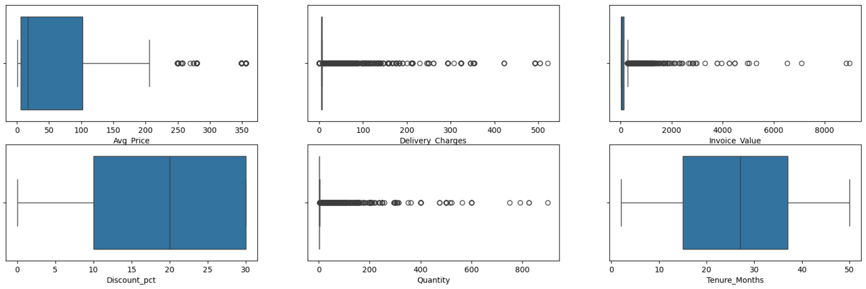
```
df['Coupon_Code'].fillna('No Coupon', inplace=True)
df[df['Coupon_Code'] == 'No Coupon'].head()
```

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Descrip
62	17850	16704	2019-01-01	GGOEYOBR078599	YouTube Luggag
95	14688	16742	2019-01-02	GGOEGBRD079699	25L Classic Rucl
157	18074	16782	2019-01-02	GGOEGOBC078699	Google Luggag
178	16029	16800	2019-01-02	GGOEAOBH078799	Android Luggag
193	16250	16812	2019-01-02	GGOEGDHG082499	Google 25 oz Stainless Steel I

```
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
--------------------------------------------------
```

```
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)

'lt.subplot(2, 3, 4)
 s.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()
```



```
df.head()
```

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Descripti
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
2	17850	16681	2019-01-01	GGOEGFKQ020399	Google Laptop a Cell Phone Sticke
3	17850	16682	2019-01-01	GGOEGAAB010516	Google Men's 10(Cotton Short Slee Hero Tee
4	17850	16682	2019-01-01	GGOEGBJL013999	Google Canvas Tc Natural/Na

Next steps: Generate code with df View recommended plots

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52924 entries, 0 to 52923
Data columns (total 18 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
 10  Gender               52924 non-null  object
 11  Location             52924 non-null  object
 12  Tenure_Months        52924 non-null  int64
 13  gst_pct              52924 non-null  int64
 14  Month_Value          52924 non-null  object
 15  Coupon_Code          52924 non-null  object
 16  Discount_pct         52924 non-null  float64
 17  Invoice_Value        52924 non-null  float64
dtypes: datetime64[ns](1), float64(4), int64(5), object(8)
memory usage: 7.3+ MB
```

```
from operator import attrgetter

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

```
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

Next steps: Generate code with df_cohort View recommended plots

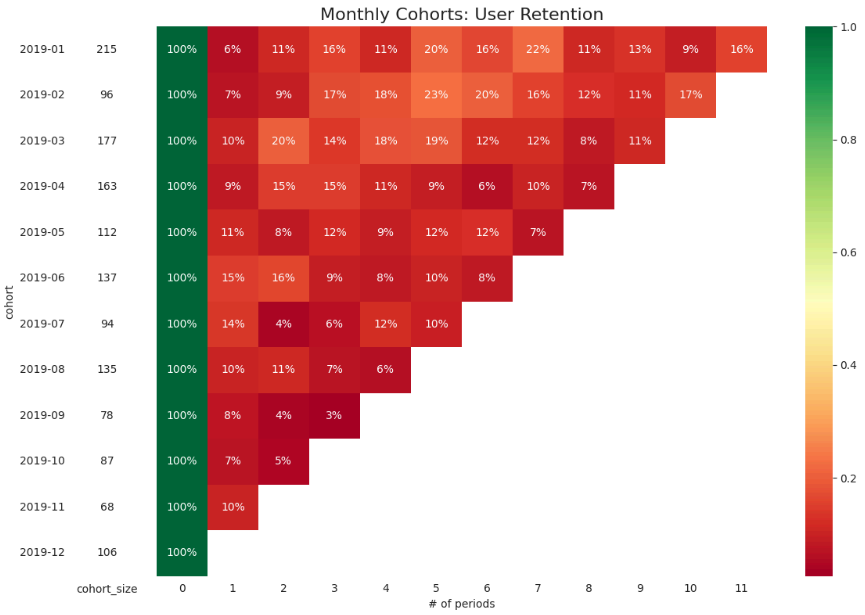
```
cohort_pivot = df_cohort.pivot_table(index='cohort', columns='period_number', values='n_customers')
```

```
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

Next steps: Generate code with cohort_pivot View recommended plots

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


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()
```



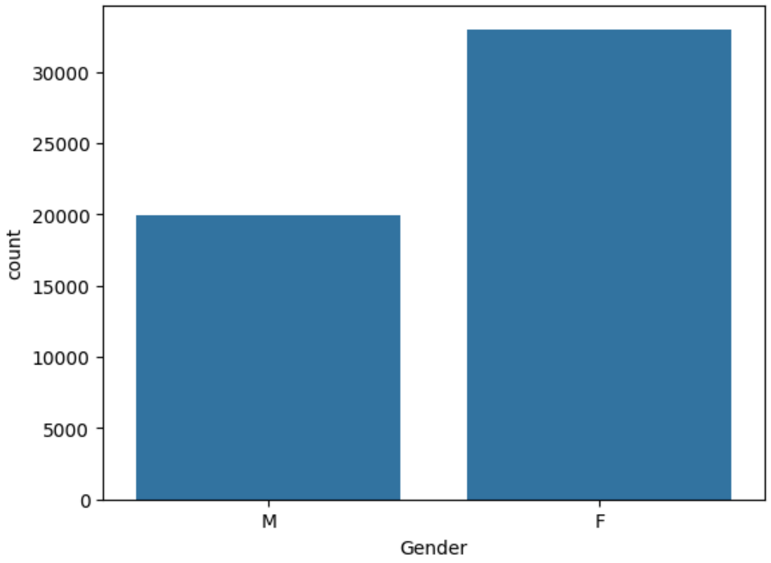
```
df.head()
```

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Descripti
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
2	17850	16681	2019-01-01	GGOEGFKQ020399	Google Laptop a Cell Phone Sticke
3	17850	16682	2019-01-01	GGOEGAAB010516	Google Men's 10(Cotton Short Slee Hero Tee
4	17850	16682	2019-01-01	GGOEGBJL013999	Google Canvas To Natural/Na

Next steps: Generate code with df View recommended plots

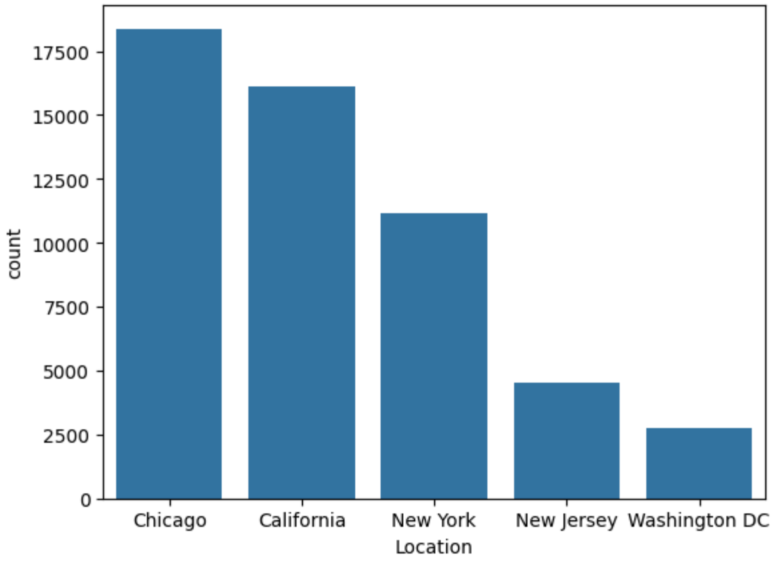
```
sns.countplot(x='Gender', data=df)
```

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

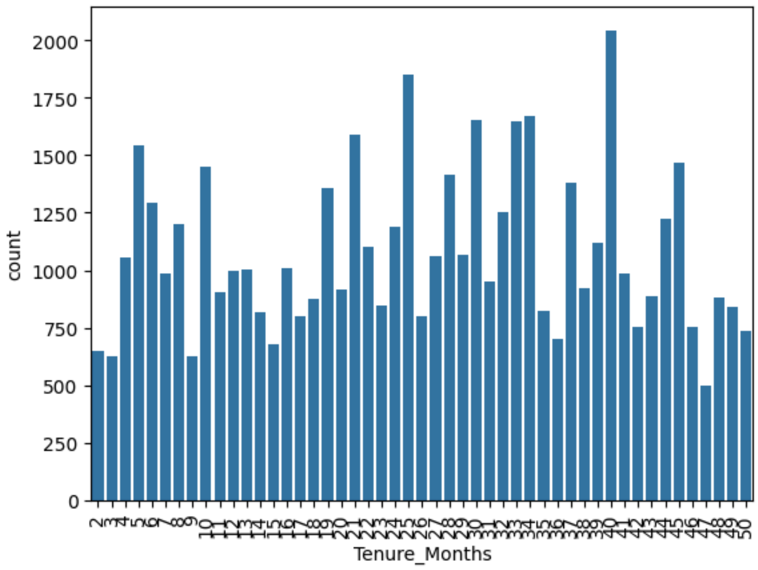


```
sns.countplot(x='Location', data=df)
```

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



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



```
marketing_spend.head()
```

	Date	Offline_Spend	Online_Spend
0	2019-01-01	4500	2424.50
1	2019-01-02	4500	3480.36
2	2019-01-03	4500	1576.38
3	2019-01-04	4500	2928.55
4	2019-01-05	4500	4055.30

Next steps: Generate code with `marketing_spend` View recommended plots

```
df.head()
```

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Descripti
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
2	17850	16681	2019-01-01	GGOEGFKQ020399	Google Laptop a Cell Phone Sticke
3	17850	16682	2019-01-01	GGOEGAAB010516	Google Men's 10(Cotton Short Slee Hero Tee
4	17850	16682	2019-01-01	GGOEGBJL013999	Google Canvas To Natural/Na

Next steps: Generate code with `df` View recommended plots

```
df_order = df[['Transaction_Date', 'Transaction_ID', 'Invoice_Value']].groupby('Transaction_Date').agg({'Invoice_Value': 'sum', 'Transac
 _order.rename(columns={'Transaction_ID': 'order_count', 'Invoice_Value' : 'revenue'}, inplace=True)
 _order.head()
```

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

Next steps: Generate code with `df_order` View recommended plots

```
df_spend = pd.merge(df_order, marketing_spend, right_on=['Date'], left_on = ['Transaction_Date'], how='left')
df_spend.head()
```

	Transaction_Date	revenue	order_count	Date	Offline_Spend	Online_Spend
0	2019-01-01	8489.73148	89	2019-01-01	4500	2424.50
1	2019-01-02	14244.70418	115	2019-01-02	4500	3480.36
2	2019-01-03	27379.80059	207	2019-01-03	4500	1576.38

Next steps: Generate code with `df_spend` View recommended plots

```
df_spend.corr()
```

	Transaction_Date	revenue	order_count	Date	Offline_Spend	Onl
Transaction_Date	1.000000	0.085899	0.092098	1.000000	0.179203	
revenue	0.085899	1.000000	0.696203	0.085899	0.081556	
order_count	0.092098	0.696203	1.000000	0.092098	-0.033937	
Date	1.000000	0.085899	0.092098	1.000000	0.179203	
Offline_Spend	0.179203	0.081556	-0.033937	0.179203	1.000000	
Online_Spend	0.144907	0.064874	-0.040347	0.144907	0.351122	

```
df.head()
```

e	Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price	Deliv
1	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	
1	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	
1	GGOEGFKQ020399	Google Laptop and Cell Phone Stickers	Office	1	2.05	
1	GGOEGAAB010516	Google Men's 100% Cotton Short Sleeve Hero Tee...	Apparel	5	17.53	
1	GGOEGBJL013999	Google Canvas Tote Natural/Navy	Bags	1	16.50	

Next steps: Generate code with df View recommended plots

```
online_sales.head()
```

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Descripti
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
					Google Laptop a

Next steps: Generate code with online_sales View recommended plots

```
df.head()
```

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Descripti
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learni Thermostat 3rd Ge USA - Stainle
2	17850	16681	2019-01-01	GGOEGFKQ020399	Google Laptop a Cell Phone Sticke
3	17850	16682	2019-01-01	GGOEGAAB010516	Google Men's 10(Cotton Short Slee Hero Tee
4	17850	16682	2019-01-01	GGOEGBJL013999	Google Canvas To Natural/Na

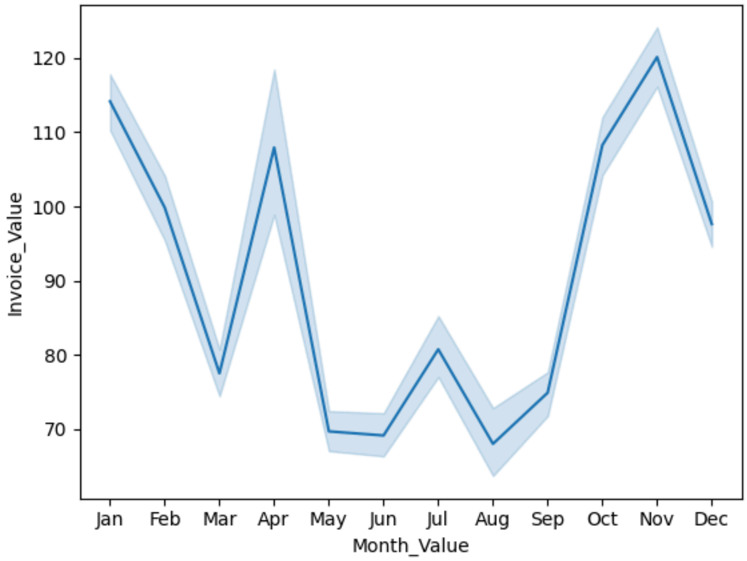
Next steps: Generate code with df View recommended plots

```
df[['Month_Value', 'Invoice_Value']].head()
```

	Month_Value	Invoice_Value
0	Jan	158.6729
1	Jan	158.6729
2	Jan	8.5295
3	Jan	99.5843
4	Jan	24.0230

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

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

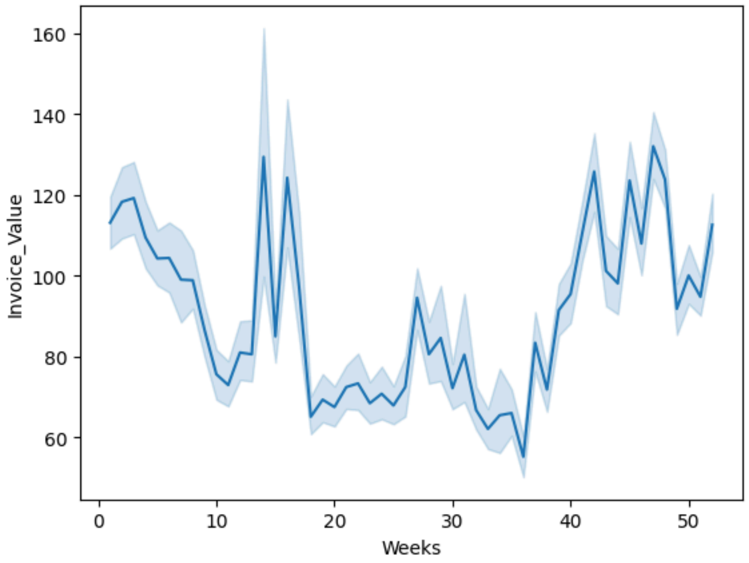


```
df['Weeks'] = df['Transaction_Date'].dt.isocalendar().week
df.head()
```

Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price	Delivery_
3GOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	
3GOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	
;GOEGFKQ020399	Google Laptop and Cell Phone Stickers	Office	1	2.05	
;GOEGAAB010516	Google Men's 100% Cotton Short Sleeve Hero Tee...	Apparel	5	17.53	
3GOEGBJL013999	Google Canvas Tote Natural/Navy	Bags	1	16.50	

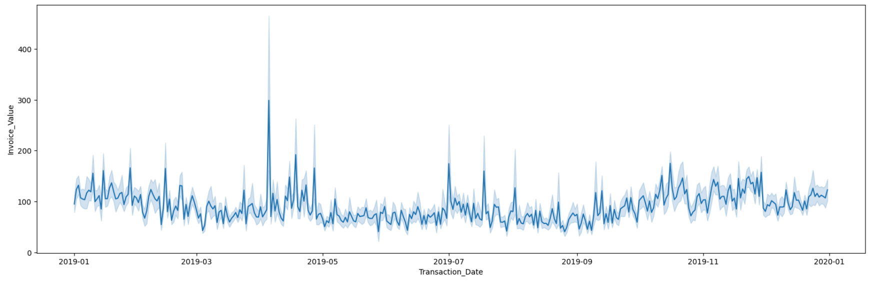
```
sns.lineplot(x='Weeks', y='Invoice_Value', data=df)
```

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



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

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

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



Suggested code may be subject to a license | tomdxb0004/Data-Science

```
df.head()
```

```
df_cat = df[['Product_Category', 'Invoice_Value', 'Transaction_ID', 'Avg_Price']].groupby('Product_Category').agg({'Invoice_Value': 'sur
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
```