SPARKLIFY | DATA DRIVEN MARKET INSIGHTS

In [1]:
```
# Importing required packages to be used

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats
from IPython.display import display
from scipy.stats.mstats import winsorize
from scipy.stats import chi2_contingency
```

In [2]:
```
# Importing and Reading top 5 data for Customers

df_customer=pd.read_excel('CustomersData.xlsx')
df_customer.head()
```

Out[2]:

	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

In [3]:
```
# Importing and Reading top 5 data for Marketing Spend

df_market=pd.read_csv('Marketing_Spend.csv')
df_market.head()
```

Out[3]:

	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

In [4]:
```
# Importing and Reading top 5 data for Tax amount

df_taxamt=pd.read_excel('Tax_amount.xlsx')
df_taxamt.head()
```

Out[4]:

	Product_Category	GST
0	Nest-USA	0.10
1	Office	0.10
2	Apparel	0.18
3	Bags	0.18
4	Drinkware	0.18

In [5]:
```
# Importing and Reading top 5 data for Discount Coupon

df_discount=pd.read_csv('Discount_Coupon.csv')
df_discount.head()
```

Out[5]:

	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

In [6]:
```
# Importing and Reading top 5 data for Online Sales

df_onlinesales=pd.read_csv('Online_Sales.csv')
df_onlinesales.head()
```

Out[6]:

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price	Delivery_Cl
0	17850	16679	1/1/2019	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	
1	17850	16680	1/1/2019	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	
2	17850	16681	1/1/2019	GGOEGFKQ020399	Google Laptop and Cell Phone Stickers	Office	1	2.05	
3	17850	16682	1/1/2019	GGOEGAAB010516	Google Men's 100% Cotton Short Sleeve Hero Tee...	Apparel	5	17.53	
4	17850	16682	1/1/2019	GGOEGBJL013999	Google Canvas Tote Natural/Navy	Bags	1	16.50	

General Analysis

In [7]:
```
# Defining the shape and dimension of Customer data

a = df_customer.shape
b = df_customer.ndim
print("Shape-->", a,'\n',"Dimension-->",b)
```

```
Shape--> (1468, 4)
 Dimension--> 2
```

In [8]:
```
# Defining the shape and dimension of Market data

a = df_market.shape
b = df_market.ndim
print("Shape-->", a,'\n',"Dimension-->",b)
```

```
Shape--> (365, 3)
 Dimension--> 2
```

In [9]:
```
# Defining the shape and dimension of Tax amount data

a = df_taxamt.shape
b = df_taxamt.ndim
print("Shape-->", a,'\n',"Dimension-->",b)
```

```
Shape--> (20, 2)
 Dimension--> 2
```

In [10]:
```
# Defining the shape and dimension of Discount data

a = df_discount.shape
b = df_discount.ndim
print("Shape-->", a,'\n',"Dimension-->",b)
```

```
Shape--> (204, 4)
 Dimension--> 2
```

In [11]:
```
# Defining the shape and dimension of Online sales data

a = df_onlinesales.shape
b = df_onlinesales.ndim
print("Shape-->", a,'\n',"Dimension-->",b)
```

```
Shape--> (52924, 10)
 Dimension--> 2
```

In [12]:
```
# Checking general info of cols {Customer}

df_customer.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
```

In [13]:
```
# Checking general info of cols {Market}

df_market.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
```

In [14]:
```
# Checking general info of cols {Tax}

df_taxamt.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     float64
dtypes: float64(1), object(1)
memory usage: 448.0+ bytes
```

In [15]:
```
# Checking general info of cols {Discount}

df_discount.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
```

In [16]:
```
# Checking general info of cols {Online sales}

df_onlinesales.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
```

Merging of all Dataframes into 1 Dataset

In [17]:
```
# Feature engg Online sales table to add a new column "Month" which could be combined with discount tables as comm

# Converting the 'Transaction_Date' column to datetime format
df_onlinesales['Transaction_Date'] = pd.to_datetime(df_onlinesales['Transaction_Date'])

# Extracting month in words and creation of new column
df_onlinesales['Month'] = df_onlinesales['Transaction_Date'].dt.strftime('%b')
df_onlinesales.head()
```

Out[17]:

rID	Transaction_ID	Transaction_Date	Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price	Delivery_Charges	Coup
350	16679	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	
350	16680	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	6.5	
350	16681	2019-01-01	GGOEGFKQ020399	Google Laptop and Cell Phone Stickers	Office	1	2.05	6.5	
350	16682	2019-01-01	GGOEGAAB010516	Google Men's 100% Cotton Short Sleeve Hero Tee...	Apparel	5	17.53	6.5	
350	16682	2019-01-01	GGOEGBJL013999	Google Canvas Tote Natural/Navy	Bags	1	16.50	6.5	

```
# Remarks
A New column Month is created which has month names
```

In [18]:
```
# Converting Date column into datetime format for Market dataset
df_market['Date'] = pd.to_datetime(df_market['Date'])
```

In [19]:
```
# Combining all of the 5 datasets into 1 single dataset for ease of analysis

df_merged = pd.merge(df_onlinesales, df_customer, on='CustomerID', how='left')
df_merged = pd.merge(df_merged, df_discount, on=['Month', 'Product_Category'], how='left')
df_merged = pd.merge(df_merged, df_market, left_on='Transaction_Date', right_on='Date', how='left')
df_merged = pd.merge(df_merged, df_taxamt, on='Product_Category', how='left')

# Changing the column name from Date to Marketing_Spend Data for distinguishing
df_merged.rename(columns={'Date': 'Marketing_Spend_Date'}, inplace=True)

df_merged.head()
```

Out[19]:

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price	Delivery_Ch
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	
2	17850	16681	2019-01-01	GGOEGFKQ020399	Google Laptop and Cell Phone Stickers	Office	1	2.05	
3	17850	16682	2019-01-01	GGOEGAAB010516	Google Men's 100% Cotton Short Sleeve Hero Tee...	Apparel	5	17.53	
4	17850	16682	2019-01-01	GGOEGBJL013999	Google Canvas Tote Natural/Navy	Bags	1	16.50	

```
# Remarks
Online sales is combined with Customer on Customer_id.
The dataset formed,is combined with discount on base of Month & product_category since discounts are given on
base of these 2
This dataset is again combined with market on base of date columns
The final dataset is obtained by combining above with tax table on base of product_category column.
```

In [334]:
```
# Converting Transaction_Date to timestamp dtype
df_merged['Transaction_Date'] = pd.to_datetime(df_merged['Transaction_Date'])

# Creating a new column "day name" which is extracted basically from transaction_date
df_merged['day_name'] = df_merged['Transaction_Date'].dt.day_name()

# Creating a new column "week name" which is extracted basically from transaction_date
df_merged['week_name'] = df_merged['Transaction_Date'].dt.strftime('%U')

# Rounding off all float values to 2 decimal places
df_merged = df_merged.round(2)

df_merged.head(2)
```

Out[334]:

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price	Delivery_Ch
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	

```
# Remarks
Since the data given is from 1st Jan 2019 to 31st Dec 2019, considering this range, 1st of jan 2019 was on
Tuesday and here
also same data is displayed.
Similary extracted week_name number and start is from 00 onwards
```

General Analysis and Data Cleaning

Duplicacy Check

In [335]:
```
# Finding duplicate rows based on all columns

duplicate_rows = df_merged[df_merged.duplicated()]

# Displaying the duplicate rows
print("Duplicate Rows:")
print(duplicate_rows)

# Counting the total number of duplicate rows
num_duplicates = len(duplicate_rows)

print(f"Total number of duplicate rows: {num_duplicates}")
```

```
Duplicate Rows:
Empty DataFrame
Columns: [CustomerID, Transaction_ID, Transaction_Date, Product_SKU, Product_Description, Product_Category, Quant
ity, Avg_Price, Delivery_Charges, Coupon_Status, Month, Gender, Location, Tenure_Months, Coupon_Code, Discount_pc
t, Marketing_Spend_Date, Offline_Spend, Online_Spend, GST, day_name, week_name, Invoice_Value]
Index: []
Total number of duplicate rows: 0
```

```
# Remarks
There are no exact duplicated rows present in the entire dataset
```

General Check

In [336]:
```
# Checking for value ranges (e.g., numeric columns should not have negative values)

numeric_columns = df_merged.select_dtypes(include=['int', 'float']).columns
value_range_issues = (df_merged[numeric_columns] < 0).any()

print("\nValue Range Issues:")
print(value_range_issues)
```

```
Value Range Issues:
CustomerID          False
Transaction_ID      False
Quantity            False
Avg_Price           False
Delivery_Charges    False
Tenure_Months       False
Discount_pct        False
Offline_Spend       False
Online_Spend        False
GST                 False
Invoice_Value       False
dtype: bool
```

Remarks: There are no negative values present in the dataset

In [337]:
```
# Overall stats of entire dataset

df_merged.describe(include="all")
```

```
C:\Users\Dell\AppData\Local\Temp\ipykernel_41356\1256549450.py:3: FutureWarning: Treating datetime data as categorical rather than numeric in `.describe` is deprecated and will be removed in a future version of pandas. Specify `datetime_is_numeric=True` to silence this warning and adopt the future behavior now.
  df_merged.describe(include="all")
C:\Users\Dell\AppData\Local\Temp\ipykernel_41356\1256549450.py:3: FutureWarning: Treating datetime data as categorical rather than numeric in `.describe` is deprecated and will be removed in a future version of pandas. Specify `datetime_is_numeric=True` to silence this warning and adopt the future behavior now.
  df_merged.describe(include="all")
```

Out[337]:

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price
count	52924.00000	52924.000000	52924	52924	52924	52924	52924.000000	52924.000000
unique	NaN	NaN	365	1145	404	20	NaN	NaN
top	NaN	NaN	2019-11-27 00:00:00	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Apparel	NaN	NaN
freq	NaN	NaN	335	3511	3511	18126	NaN	NaN
first	NaN	NaN	2019-01-01 00:00:00	NaN	NaN	NaN	NaN	NaN
last	NaN	NaN	2019-12-31 00:00:00	NaN	NaN	NaN	NaN	NaN
mean	15346.70981	32409.825675	NaN	NaN	NaN	NaN	4.497638	52.237646
std	1766.55602	8648.668977	NaN	NaN	NaN	NaN	20.104711	64.006882
min	12346.00000	16679.000000	NaN	NaN	NaN	NaN	1.000000	0.390000
25%	13869.00000	25384.000000	NaN	NaN	NaN	NaN	1.000000	5.700000
50%	15311.00000	32625.500000	NaN	NaN	NaN	NaN	1.000000	16.990000
75%	16996.25000	39126.250000	NaN	NaN	NaN	NaN	2.000000	102.130000
max	18283.00000	48497.000000	NaN	NaN	NaN	NaN	900.000000	355.740000

Remarks: The above data represents quite detailed data w.r.t various stats considering each column in the dataset

In [338]:
```
# Categorical Data Description

df_merged.describe(include= object).T
```

Out[338]:

	count	unique	top	freq
Product_SKU	52924	1145	GGOENEBJ079499	3511
Product_Description	52924	404	Nest Learning Thermostat 3rd Gen-USA - Stainle...	3511
Product_Category	52924	20	Apparel	18126
Coupon_Status	52924	3	Clicked	26926
Month	52924	12	Aug	6150
Gender	52924	2	F	33007
Location	52924	5	Chicago	18380
Coupon_Code	52924	46	SALE20	6373
day_name	52924	7	Friday	9266
week_name	52924	53	30	1515

Remarks: The above data displays all the stats for categorical data

Handling Null values

In [26]:
```
# Null/Empty values in dataset

df_merged.isna().sum().sort_values(ascending=False)
```

Out[26]:
```
Discount_pct            400
Coupon_Code             400
CustomerID                0
Transaction_ID            0
day_name                  0
GST                       0
Online_Spend              0
Offline_Spend             0
Marketing_Spend_Date      0
Tenure_Months             0
Location                  0
Gender                    0
Month                     0
Coupon_Status             0
Delivery_Charges          0
Avg_Price                 0
Quantity                  0
Product_Category          0
Product_Description       0
Product_SKU               0
Transaction_Date          0
week_name                 0
dtype: int64
```

Remarks: There are 400 null values for discount_pct and coupon_code because there are 4 product_categories which are not present in the discount_coupon dataset: - Google, Fun, Backpacks and More Bags

In [27]:
```
# Checking and cross-verifying the same (w.r.t null values)
df_merged[df_merged['Discount_pct'].isna()]
```

Out[27]:

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price	Deliver
62	17850	16704	2019-01-01	GGOEYOBR078599	YouTube Luggage Tag	Fun	4	9.27	
95	14688	16742	2019-01-02	GGOEGBRD079699	25L Classic Rucksack	Backpacks	1	103.15	
157	18074	16782	2019-01-02	GGOEGOBC078699	Google Luggage Tag	Fun	1	7.42	
178	16029	16800	2019-01-02	GGOEAOBH078799	Android Luggage Tag	Fun	2	7.42	
193	16250	16812	2019-01-02	GGOEGDHG082499	Google 25 oz Clear Stainless Steel Bottle	Google	1	11.54	
...	...	...	...	...	...	...	...	...	
44213	12472	42109	2019-10-30	GGOEGBRD079699	25L Classic Rucksack	Backpacks	1	79.99	
45167	14911	42756	2019-11-07	GGOEGBRD079699	25L Classic Rucksack	Backpacks	1	79.99	
45807	18125	43244	2019-11-12	GGOEGBRD079699	25L Classic Rucksack	Backpacks	1	99.99	
46239	17180	43537	2019-11-15	GGOEGBRD079699	25L Classic Rucksack	Backpacks	1	79.99	
46966	12377	44124	2019-11-21	GGOEGBRB079599	25L Classic Rucksack	Backpacks	1	99.99	

400 rows × 22 columns

In [340]:
```
# Handling these 400 NaN values by filling with value as N/A (for object type:-coupon_code) and 0 (for float type:

df_merged.replace(to_replace=pd.NA, value='N/A', inplace=True)
df_merged.fillna(value=0, inplace=True)
```

Adding of column "Invoice Value" to the final dataset

In [341]:
```
df_merged['Invoice_Value'] = round((((df_merged['Quantity'] * df_merged['Avg_Price'])) * (1 - (df_merged['Discount_
                                    * (1 + df_merged['GST'])) + df_merged['Delivery_Charges'],2)
df_merged.head()
```

Out[341]:

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price	Delivery_Cl
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	
2	17850	16681	2019-01-01	GGOEGFKQ020399	Google Laptop and Cell Phone Stickers	Office	1	2.05	
3	17850	16682	2019-01-01	GGOEGAAB010516	Google Men's 100% Cotton Short Sleeve Hero Tee...	Apparel	5	17.53	
4	17850	16682	2019-01-01	GGOEGBJL013999	Google Canvas Tote Natural/Navy	Bags	1	16.50	

Remarks This column is framed by the formula given in the dataset description. Invoice Value = ((Quantity Avg_price) (1 - Discount_pct) * (1 + GST)) + Delivery_Charges.

Also the discount percentage is a whole number and as per the formula, it wasn't converted into whole number, so did the conversion so as to avoid negative values in the entire column

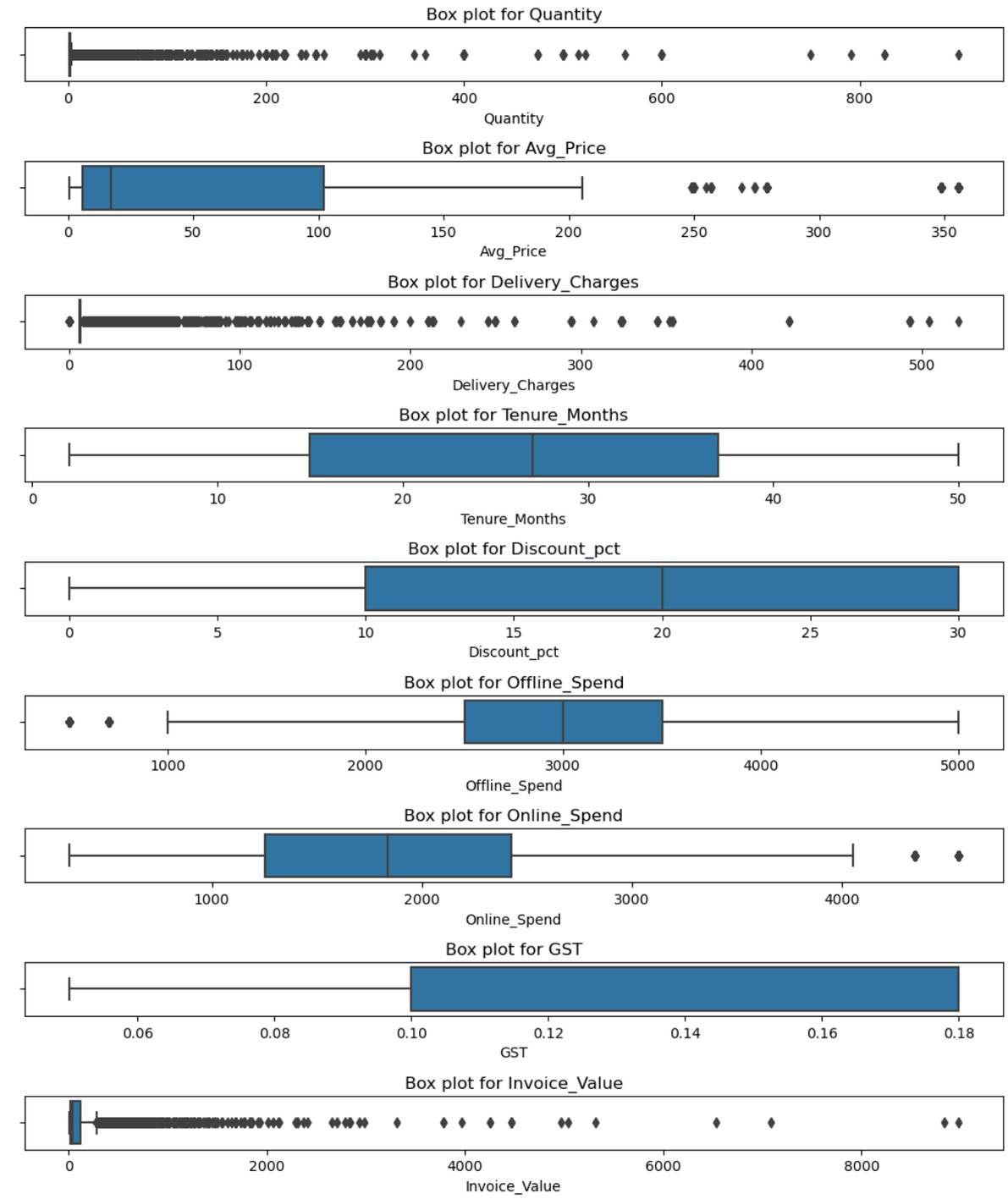
Outlier Checks and Treatment

In [343]:
```
# Creating a copy of original data

df_merged_copy=df_merged.copy()
df_merged_copy.head(2)
```

Out[343]:

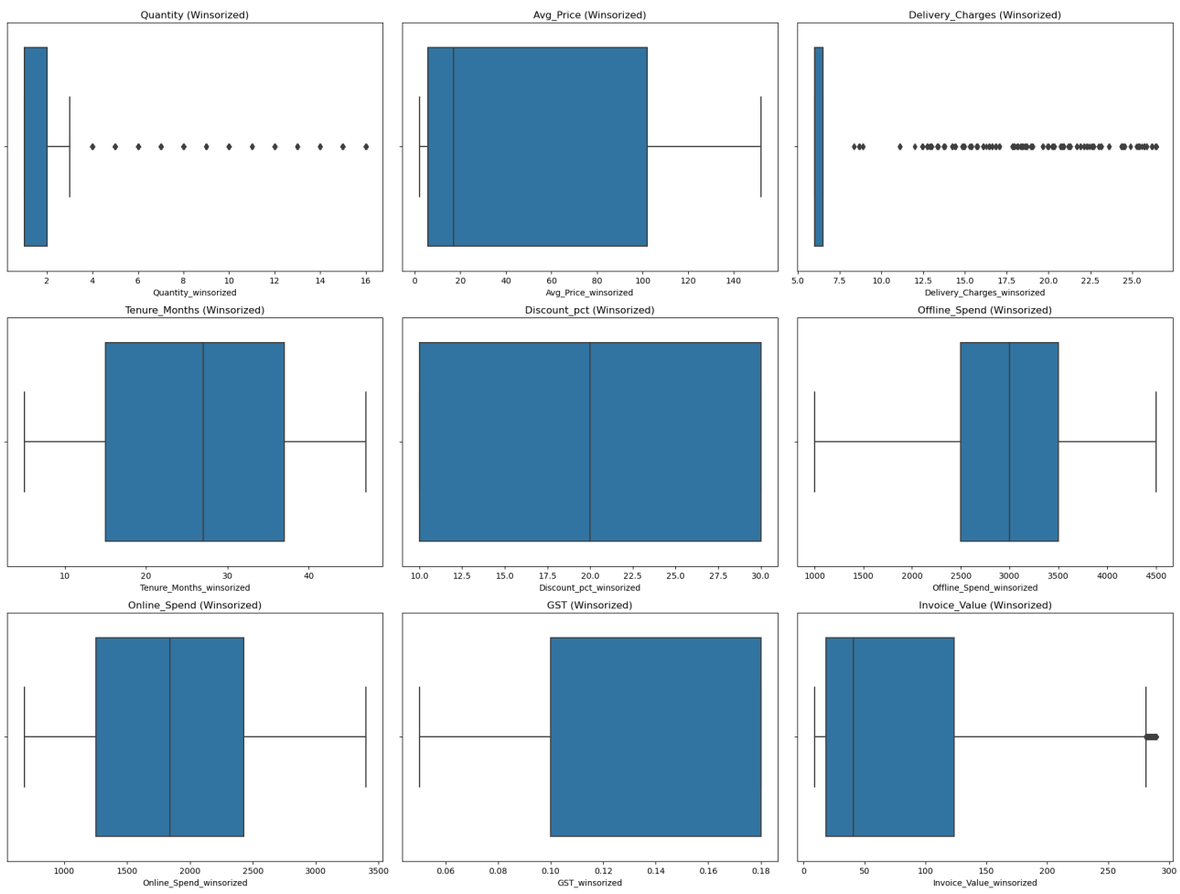
	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price	Delivery_Ch
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	

In [344]:
```
# Checking the outliers for all the numerical columns

numeric_columns = ['Quantity', 'Avg_Price', 'Delivery_Charges', 'Tenure_Months', 'Discount_pct',
                   'Offline_Spend', 'Online_Spend', 'GST', 'Invoice_Value']

fig, axes = plt.subplots(nrows=len(numeric_columns), ncols=1, figsize=(10, 12))

# Looping through each of the numeric columns and plotting graph for same
for i, column in enumerate(numeric_columns):
    sns.boxplot(x=df_merged_copy[column], ax=axes[i])
    axes[i].set_title(f'Box plot for {column}')

plt.tight_layout()
plt.show()
```



Remarks: There are lot of outliers present for Quantity, Delivery charges and invoice value

In [345]:
```
# Handling the above outliers using winsorization technique

columns_of_interest = ['Quantity', 'Avg_Price', 'Delivery_Charges', 'Tenure_Months', 'Discount_pct',
                       'Offline_Spend', 'Online_Spend', 'GST', 'Invoice_Value']

fig, axes = plt.subplots(3, 3, figsize=(20, 15))
axes = axes.flatten()

for i, column in enumerate(columns_of_interest):
# Calculating the 5th and 95th percentiles for winsorizing
    fifth_percentile = df_merged_copy[column].quantile(0.05)
    ninety_fifth_percentile = df_merged_copy[column].quantile(0.95)

# Applying winsorizing to the columns
    winsorized_column = f'{column}_winsorized'
    df_merged_copy[winsorized_column] = winsorize(df_merged_copy[column], limits=(0.05, 0.05))

    sns.boxplot(data=df_merged_copy, x=winsorized_column, orient='h', ax=axes[i])
    axes[i].set_title(f"{column} (Winsorized)")

plt.tight_layout()
plt.show()
```



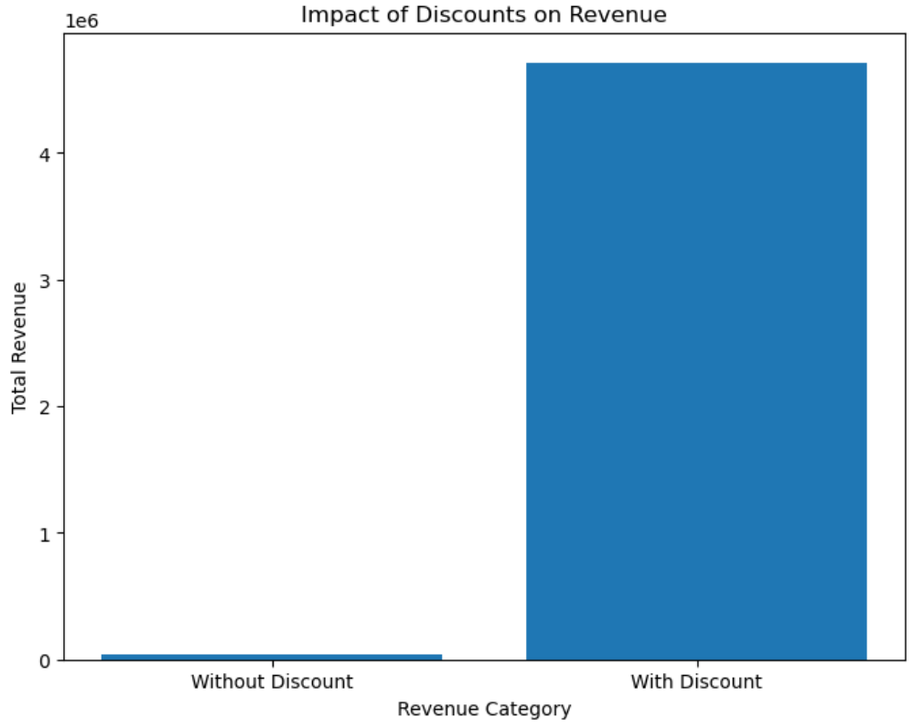
Remarks: The outliers are handled properly. Also removing the outliers would result in data loss, so I have handled it with nearest value

Graphical Analysis

```
Questions from the compulsory question bank
```

```
Question: - It is essential to delve into the impact of discounts on revenues, analyze key performance indicators
(KPIs) like revenue, number of orders, average order value, customer quantity, and more, across different
dimensions such as category, month, week, and day.
```

In [352]:
```
# Impact of discounts on revenues

# Calculating total revenue with and without discount
without_discount = df_merged[df_merged['Discount_pct'] == 0]
df_merged_copy["Revenue_Without_Discount"] = round((((without_discount['Quantity'] * without_discount['Avg_Price']

df_merged_copy["Revenue_With_Discount"] = round((((df_merged['Quantity'] * df_merged['Avg_Price'])) * (1 - (df_mer

total_revenue_without_discount = df_merged_copy["Revenue_Without_Discount"].sum()
total_revenue_with_discount = df_merged_copy["Revenue_With_Discount"].sum()

# Calculating discount impact on revenue
discount_impact = round((total_revenue_with_discount - total_revenue_without_discount),2)
discount_impact_per = round(((total_revenue_with_discount - total_revenue_without_discount))/total_revenue_with_di

# Visualizing the impact
plt.figure(figsize=(8, 6))
plt.bar(["Without Discount", "With Discount"], [total_revenue_without_discount, total_revenue_with_discount])
plt.xlabel("Revenue Category")
plt.ylabel("Total Revenue")
plt.title("Impact of Discounts on Revenue")
plt.show()

print("Total Revenue Without Discount:", round((total_revenue_without_discount),2))
print("Total Revenue With Discount:", total_revenue_with_discount)
print("Discount Impact on Revenue:", discount_impact)
print("Discount Impact % on Revenue:", discount_impact_per)
```




```
Total Revenue Without Discount: 36949.65
Total Revenue With Discount: 4714507.83
Discount Impact on Revenue: 4677558.18
Discount Impact % on Revenue: 99.22
```

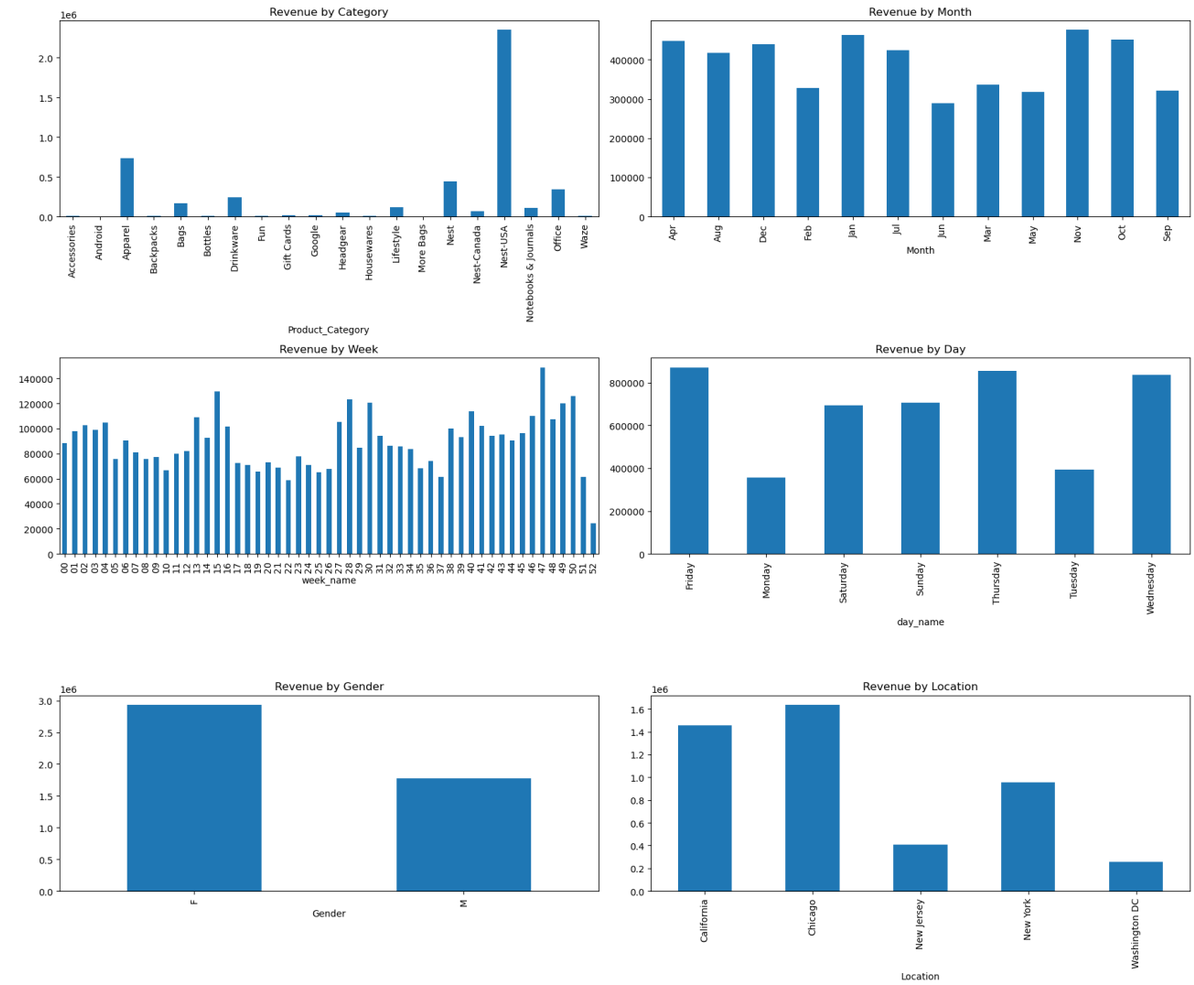
Remarks:- The positive high value for "Discount Impact on Revenue" indicates that the total revenue with discounts is far higher than the total revenue without discounts.

Feature Engg to calculate KPIs

In [40]:
```
# KPIs calculation

# Total Revenue Calculation
total_revenue_without_discount = df_merged_copy["Revenue_Without_Discount"].sum()
total_revenue_with_discount = df_merged_copy["Revenue_With_Discount"].sum()
revenue = total_revenue_without_discount+total_revenue_with_discount

# Other KPIs
num_orders = df_merged_copy['Transaction_ID'].nunique()
avg_order_value = round(df_merged_copy.groupby('Transaction_ID')['Invoice_Value'].mean().mean(),2)
customer_quantity = df_merged_copy['CustomerID'].nunique()

print(f"Total Revenue: {revenue}")
print(f"Number of Orders: {num_orders}")
print(f"Average Order Value: {avg_order_value}")
print(f"Number of Unique Customers: {customer_quantity}")
```

```
Total Revenue: 4751457.48
Number of Orders: 25061
Average Order Value: 115.24
Number of Unique Customers: 1468
```

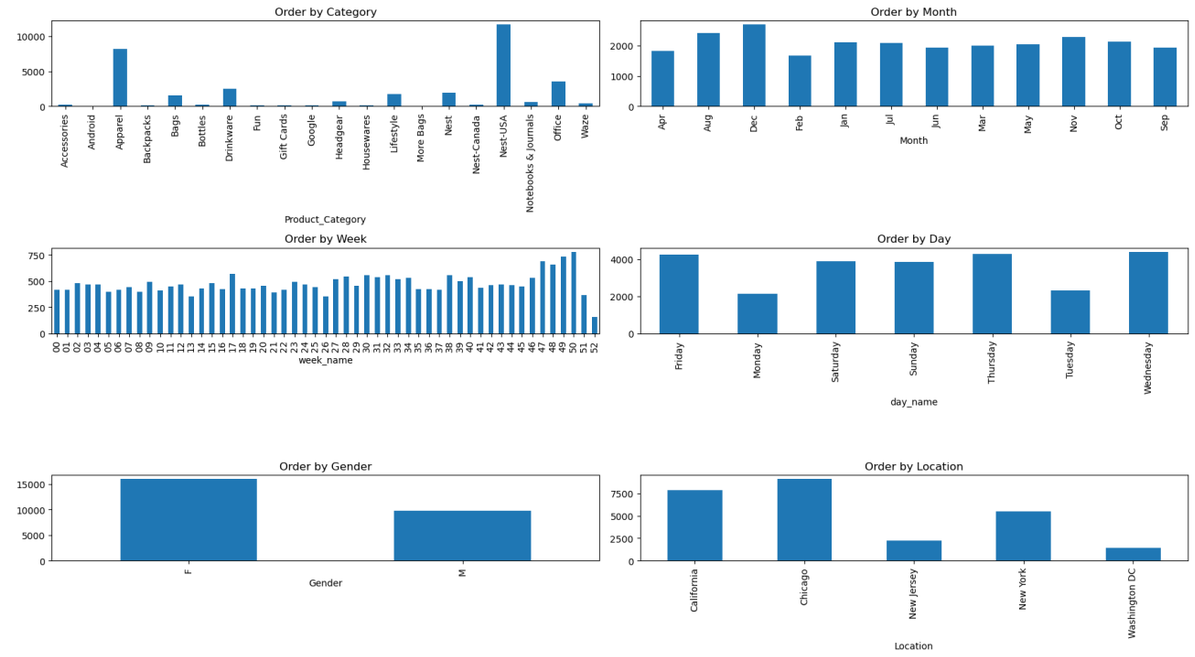
In [353]:
```
# Analysis of Revenue with various factors

revenue_by_category = df_merged_copy.groupby('Product_Category')['Invoice_Value'].sum()
revenue_by_month = df_merged_copy.groupby('Month')['Invoice_Value'].sum()
revenue_by_week = df_merged_copy.groupby('week_name')['Invoice_Value'].sum()
revenue_by_day = df_merged_copy.groupby('day_name')['Invoice_Value'].sum()
revenue_by_gender = df_merged_copy.groupby('Gender')['Invoice_Value'].sum()
revenue_by_location = df_merged_copy.groupby('Location')['Invoice_Value'].sum()

plt.figure(figsize=(18, 15))

# Revenue by Category
plt.subplot(3, 2, 1)
revenue_by_category.plot(kind='bar', title='Revenue by Category')

# Revenue by Month
plt.subplot(3, 2, 2)
revenue_by_month.plot(kind='bar', title='Revenue by Month')

# Revenue by Week
plt.subplot(3, 2, 3)
revenue_by_week.plot(kind='bar', title='Revenue by Week')

# Revenue by Day
plt.subplot(3, 2, 4)
revenue_by_day.plot(kind='bar', title='Revenue by Day')

# Revenue by Gender
plt.subplot(3, 2, 5)
revenue_by_gender.plot(kind='bar', title='Revenue by Gender')

# Revenue by Location
plt.subplot(3, 2, 6)
revenue_by_location.plot(kind='bar', title='Revenue by Location')

plt.tight_layout()
plt.show()
```



Remarks:

1. The highest revenue by category is from Nest_USA
2. November month has highest revenue generated
3. 47th week has highest revenue generated and lowest is on 52nd week number
4. Friday have more revenue generated and lowest is on Monday-Tuesday
5. Females contribute more towards the revenue generation compared to Males
6. The lowest revenue generated is from Washington Dc and highest from Chicago

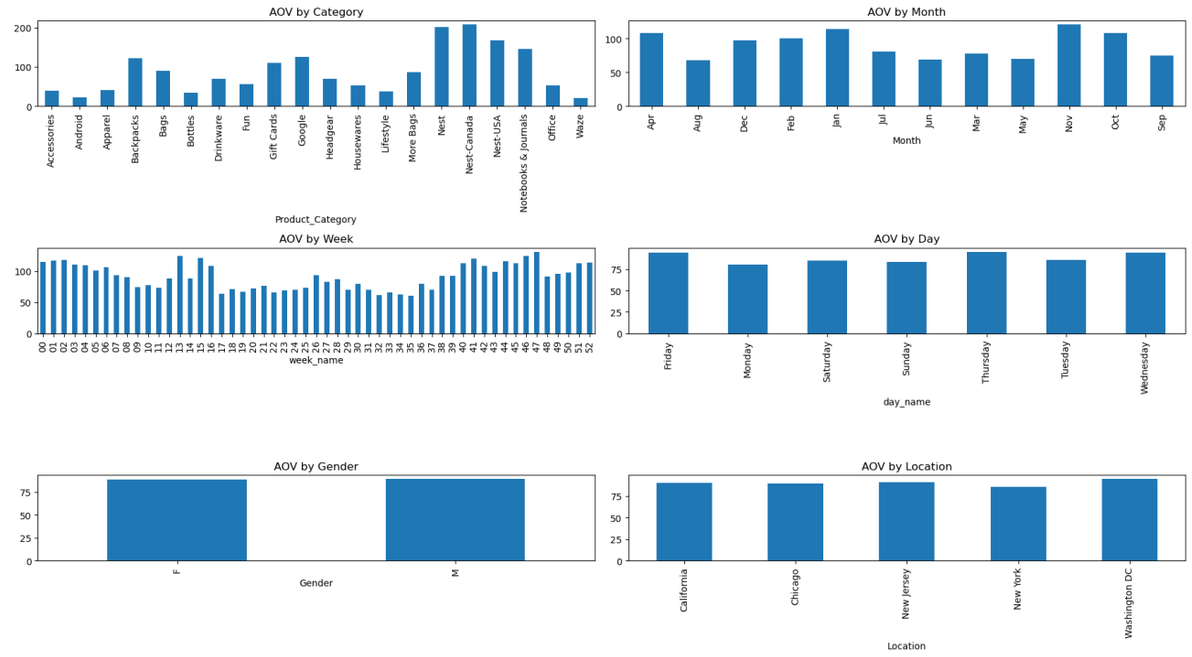
In [354]:
```
# Analysis of orders with various factors

orders_by_category = df_merged_copy.groupby('Product_Category')['Transaction_ID'].nunique()
orders_by_month = df_merged_copy.groupby('Month')['Transaction_ID'].nunique()
orders_by_week = df_merged_copy.groupby('week_name')['Transaction_ID'].nunique()
orders_by_day = df_merged_copy.groupby('day_name')['Transaction_ID'].nunique()
orders_by_gender = df_merged_copy.groupby('Gender')['Transaction_ID'].nunique()
orders_by_location = df_merged_copy.groupby('Location')['Transaction_ID'].nunique()

plt.figure(figsize=(18, 10))

# Order by Category
plt.subplot(3, 2, 1)
orders_by_category.plot(kind='bar', title='Order by Category')

# Order by Month
plt.subplot(3, 2, 2)
orders_by_month.plot(kind='bar', title='Order by Month')

# Order by Week
plt.subplot(3, 2, 3)
orders_by_week.plot(kind='bar', title='Order by Week')

# Order by Day
plt.subplot(3, 2, 4)
orders_by_day.plot(kind='bar', title='Order by Day')

# Order by Gender
plt.subplot(3, 2, 5)
orders_by_gender.plot(kind='bar', title='Order by Gender')

# Order by Location
plt.subplot(3, 2, 6)
orders_by_location.plot(kind='bar', title='Order by Location')

plt.tight_layout()
plt.show()
```



Remarks:

1. The highest order by category is from Nest_USA
2. December month has highest number of orders generated
3. 50th week has highest orders placed and lowest is on 52nd week number
4. Wednesday have more orders and lowest is on Monday
5. Females place more orders compared to Males
6. The lowest orders is from Washington Dc and highest from Chicago

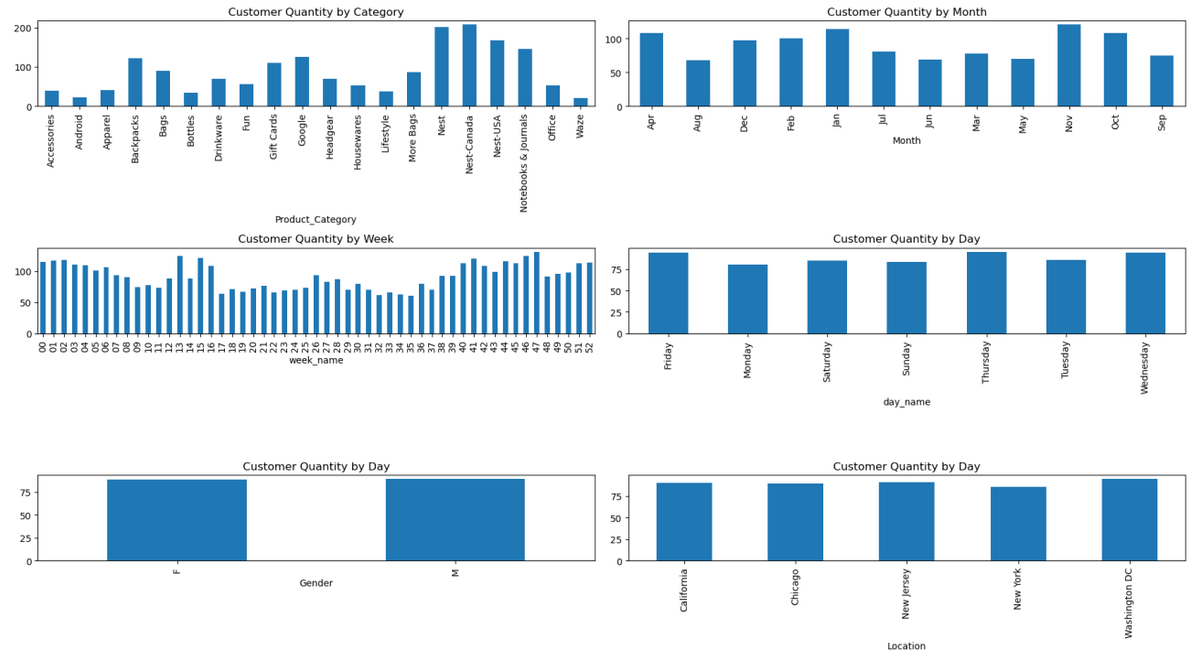
In [356]:
```
# Analysis of AOV with various factors

avg_order_value_by_category = df_merged_copy.groupby('Product_Category')['Invoice_Value'].mean()
avg_order_value_by_month = df_merged_copy.groupby('Month')['Invoice_Value'].mean()
avg_order_value_by_week = df_merged_copy.groupby('week_name')['Invoice_Value'].mean()
avg_order_value_by_day = df_merged_copy.groupby('day_name')['Invoice_Value'].mean()
avg_order_value_by_gender = df_merged_copy.groupby('Gender')['Invoice_Value'].mean()
avg_order_value_by_location = df_merged_copy.groupby('Location')['Invoice_Value'].mean()

plt.figure(figsize=(18, 10))

# AOV by Category
plt.subplot(3, 2, 1)
avg_order_value_by_category.plot(kind='bar', title='AOV by Category')

# AOV by Month
plt.subplot(3, 2, 2)
avg_order_value_by_month.plot(kind='bar', title='AOV by Month')

# AOV by Week
plt.subplot(3, 2, 3)
avg_order_value_by_week.plot(kind='bar', title='AOV by Week')

# AOV by Day
plt.subplot(3, 2, 4)
avg_order_value_by_day.plot(kind='bar', title='AOV by Day')

# AOV by Gender
plt.subplot(3, 2, 5)
avg_order_value_by_gender.plot(kind='bar', title='AOV by Gender')

# AOV by Location
plt.subplot(3, 2, 6)
avg_order_value_by_location.plot(kind='bar', title='AOV by Location')

plt.tight_layout()
plt.show()
```



Remarks:

1. The highest AOV by category is from Nest_USA
2. November month has highest AOV
3. 47th week has highest AOV and lowest is observed across various week numbers with constant rate
4. Similar AOV is observed as observed for orders chart
5. Females and Males have almost same AOV

In [44]:
```
# Analysis of Customer Quantity with various factors

customer_quantity_by_category = df_merged_copy.groupby('Product_Category')['CustomerID'].nunique()
customer_quantity_by_month = df_merged_copy.groupby('Month')['CustomerID'].nunique()
customer_quantity_by_week = df_merged_copy.groupby('week_name')['CustomerID'].nunique()
customer_quantity_by_day = df_merged_copy.groupby('day_name')['CustomerID'].nunique()
customer_quantity_by_gender = df_merged_copy.groupby('Gender')['CustomerID'].nunique()
customer_quantity_by_location = df_merged_copy.groupby('Location')['CustomerID'].nunique()

plt.figure(figsize=(18, 10))

# Customer Quantity by Category
plt.subplot(3, 2, 1)
avg_order_value_by_category.plot(kind='bar', title='Customer Quantity by Category')

# Customer Quantity by Month
plt.subplot(3, 2, 2)
avg_order_value_by_month.plot(kind='bar', title='Customer Quantity by Month')

# Customer Quantity by Week
plt.subplot(3, 2, 3)
avg_order_value_by_week.plot(kind='bar', title='Customer Quantity by Week')

# Customer Quantity by Day
plt.subplot(3, 2, 4)
avg_order_value_by_day.plot(kind='bar', title='Customer Quantity by Day')

# Customer Quantity by Gender
plt.subplot(3, 2, 5)
avg_order_value_by_gender.plot(kind='bar', title='Customer Quantity by Gender')

# Customer Quantity by Location
plt.subplot(3, 2, 6)
avg_order_value_by_location.plot(kind='bar', title='Customer Quantity by Location')

plt.tight_layout()
plt.show()
```



Remarks:

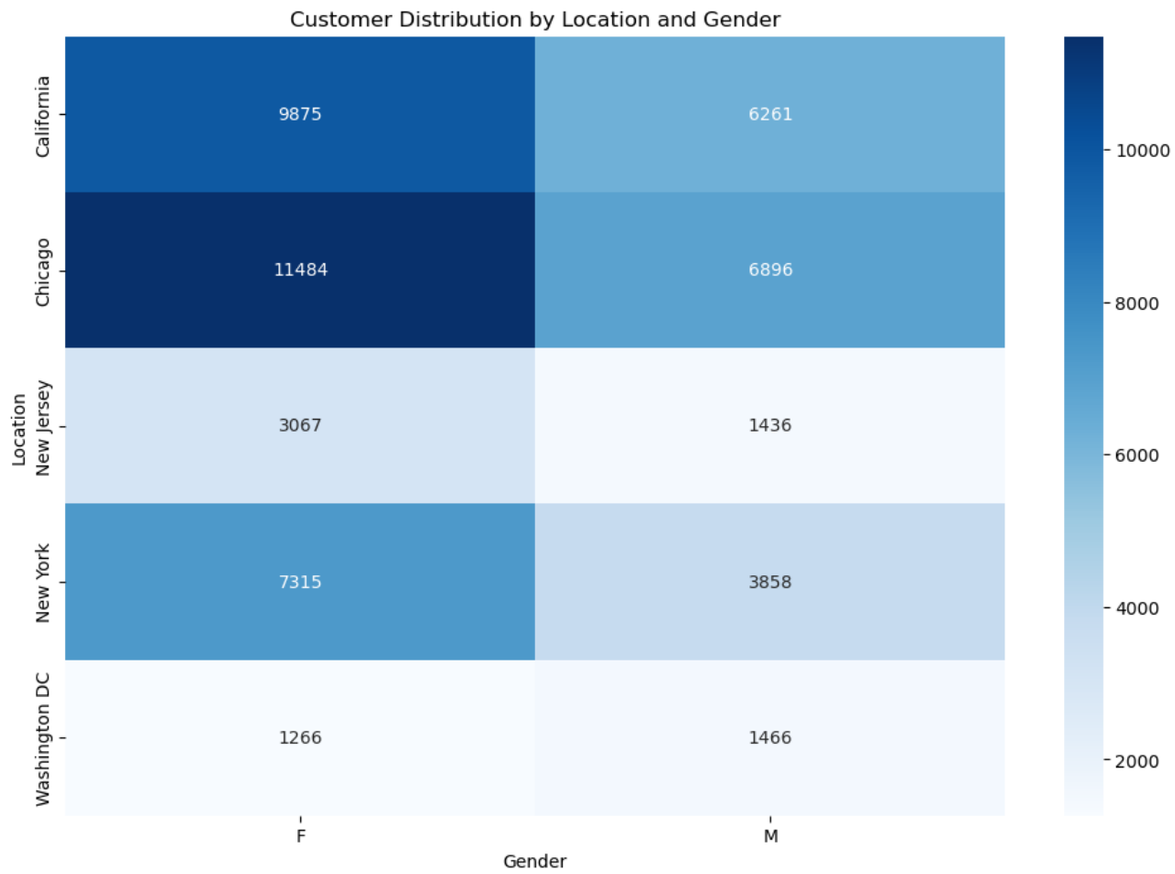
1. The highest is from Nest_USA
2. November month has highest number
3. 47th week has customer quantity
4. Wednesday,Thursday, Friday have more customer quantity

Additional KPIs Analysis

In [357]:
```
# Customer Acquisition Cost

# Total_Purchase_Value
total_purchase_value = df_merged_copy["Invoice_Value"]

# New_Customers_Acquired (new 1st transaction is within last 15 days from the latest date in dataset) [Assumption]
new_customers = df_merged_copy[(df_merged_copy['Transaction_Date'] >= latest_date - pd.DateOffset(days=15)) &
                               ~df_merged_copy['CustomerID'].isin(df_merged_copy[df_merged_copy['Transaction_Date
total_new_customers = new_customers['CustomerID'].nunique()
CAC = round((total_purchase_value.sum())/total_new_customers,2)

print("Total_Purchase_Value:", total_purchase_value.sum())
print("New_Customers_Acquired:", total_new_customers)
print("Customer Acquisition Cost:", CAC)
```

```
Total_Purchase_Value: 4714507.83
New_Customers_Acquired: 43
Customer Acquisition Cost: 109639.72
```

In [322]:
```
# CRM Analytics: Distribution of customers by location and gender

crm_analytics = df_merged_copy.groupby(['Location', 'Gender']).size().unstack(fill_value=0)

# Plotting the results
plt.figure(figsize=(12, 8))
sns.heatmap(crm_analytics, annot=True, fmt="d", cmap="Blues")
plt.title('Customer Distribution by Location and Gender')
plt.xlabel('Gender')
plt.ylabel('Location')
plt.show()
```



Remarks

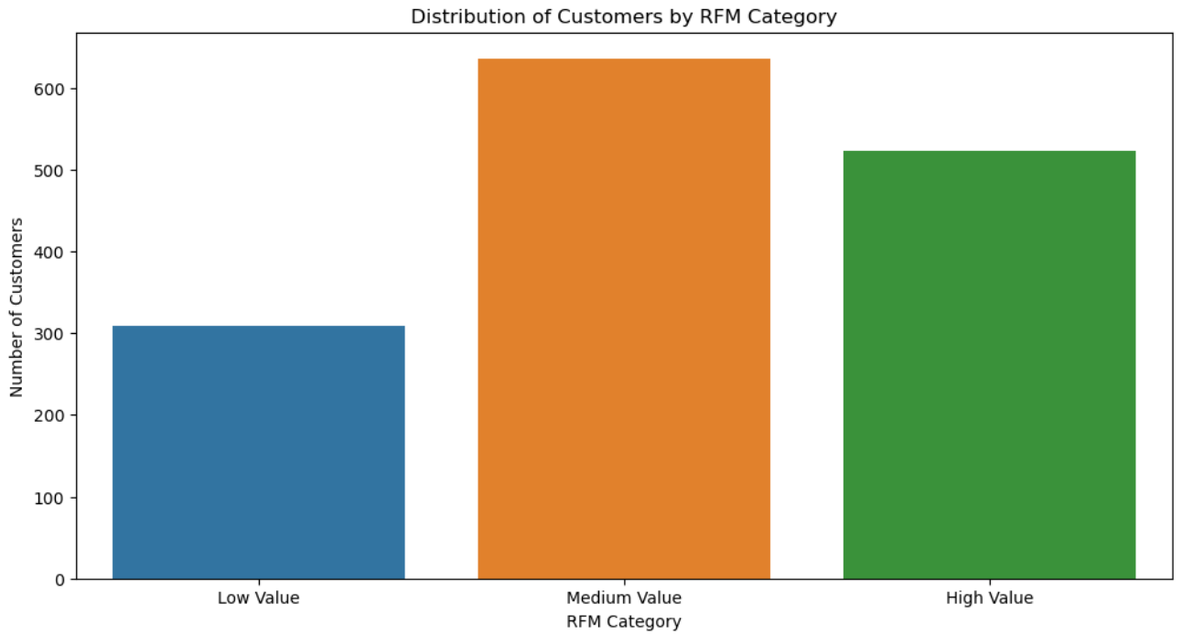
1. The highest is of Females from Chicago and lowest is of Females from Washington DC
2. Also the Females have a stronger hold over Males

In [358]:
```
# Calculating Recency, Frequency, and Monetary values (RFM Score)

current_date = df_merged_copy["Transaction_Date"].max()

recency = df_merged_copy.groupby('CustomerID')['Transaction_Date'].max()
recency = (current_date - recency).dt.days

frequency = df_merged_copy.groupby('CustomerID').size()

monetary = df_merged_copy.groupby('CustomerID')['Invoice_Value'].sum()

# Creating a new RFM DataFrame
rfm_df = pd.DataFrame({
    'Recency': recency,
    'Frequency': frequency,
    'Monetary': monetary
})

# Categorizing customers based on activity levels

# Defining quantiles for categorization
quantiles = rfm_df.quantile(q=[0.25, 0.5, 0.75])

# Assigning RFM scores
def rfm_score(x, parameter, quantiles):
    if x <= quantiles[parameter][0.25]:
        return 1
    elif x <= quantiles[parameter][0.5]:
        return 2
    elif x <= quantiles[parameter][0.75]:
        return 3
    else:
        return 4

rfm_df['Recency_Score'] = rfm_df['Recency'].apply(rfm_score, args=('Recency', quantiles))
rfm_df['Frequency_Score'] = rfm_df['Frequency'].apply(rfm_score, args=('Frequency', quantiles))
rfm_df['Monetary_Score'] = rfm_df['Monetary'].apply(rfm_score, args=('Monetary', quantiles))

# Combining the scores into a single RFM segment
rfm_df['RFM_Segment'] = rfm_df['Recency_Score'].astype(str) + rfm_df['Frequency_Score'].astype(str) + rfm_df['Mone

# Calculating RFM score as the sum of individual scores
rfm_df['RFM_Score'] = rfm_df['Recency_Score'] + rfm_df['Frequency_Score'] + rfm_df['Monetary_Score']

# Categorizing customers based on RFM score

# Defining segments based on RFM score
def categorize_rfm_score(score):
    if score >= 9:
        return 'High Value'
    elif score >= 6:
        return 'Medium Value'
    else:
        return 'Low Value'

# Applying the categorization function to create a new column 'RFM_Category'
rfm_df['RFM_Category'] = rfm_df['RFM_Score'].apply(categorize_rfm_score)
rfm_df.head()
```

Out[358]:

CustomerID	Recency	Frequency	Monetary	Recency_Score	Frequency_Score	Monetary_Score	RFM_Segment	RFM_Score	RFM_Category
12346	107	2	174.98	2	1	1	211	4	Low Value
12347	59	60	12090.30	2	4	4	244	10	High Value
12348	73	23	1501.90	2	3	2	232	7	Medium Value
12350	17	17	1183.72	1	2	2	122	5	Low Value
12356	107	36	1753.42	2	3	2	232	7	Medium Value

In [359]:
```
# Plotting the distribution of customers in each RFM category acheived above

plt.figure(figsize=(12, 6))
sns.countplot(x='RFM_Category', data=rfm_df, order=['Low Value', 'Medium Value', 'High Value'])
plt.title('Distribution of Customers by RFM Category')
plt.xlabel('RFM Category')
plt.ylabel('Number of Customers')
plt.show()
```



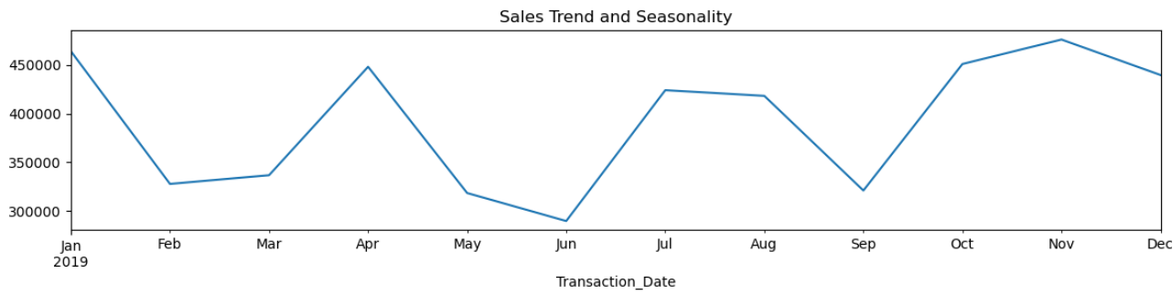
Remarks

Medium value customers (RFM score basis) are highest

Question: - Identifying trends and seasonality in sales, understanding variations in order numbers and sales on different days, and calculating various metrics like revenue, marketing spend, and delivery charges by month are critical components of the analysis.

In [362]:
```
# Identifying trends and seasonality in sales

df_merged_copy1=df_merged_copy.copy()

# Converting 'Transaction_Date' to datetime
df_merged_copy1['Transaction_Date'] = pd.to_datetime(df_merged_copy1['Transaction_Date'])
df_merged_copy1['Month'] = df_merged_copy1['Transaction_Date'].dt.to_period('M')

sales_trend_seasonality = df_merged_copy1.resample('M', on='Transaction_Date')['Invoice_Value'].sum()

# Resampling by day for variations in order numbers and sales
orders_by_day = df_merged_copy1.resample('D', on='Transaction_Date')['Transaction_ID'].nunique()
sales_by_day = df_merged_copy1.resample('D', on='Transaction_Date')['Invoice_Value'].sum()

# Calculating metrics by month
revenue_by_month = df_merged_copy1.groupby('Month')['Invoice_Value'].sum()
print(f"Total Revenue by Month:\n{revenue_by_month}")

plt.figure(figsize=(12, 8))
plt.subplot(3, 1, 1)
sales_trend_seasonality.plot(title='Sales Trend and Seasonality')

plt.tight_layout()
plt.show()
```

```
Total Revenue by Month:
Month
2019-01    463881.87
2019-02    327896.25
2019-03    336805.27
2019-04    447998.27
2019-05    318556.12
2019-06    289831.04
2019-07    423981.63
2019-08    418158.14
2019-09    321128.14
2019-10    450838.47
2019-11    475901.17
2019-12    439531.46
Freq: M, Name: Invoice_Value, dtype: float64
```

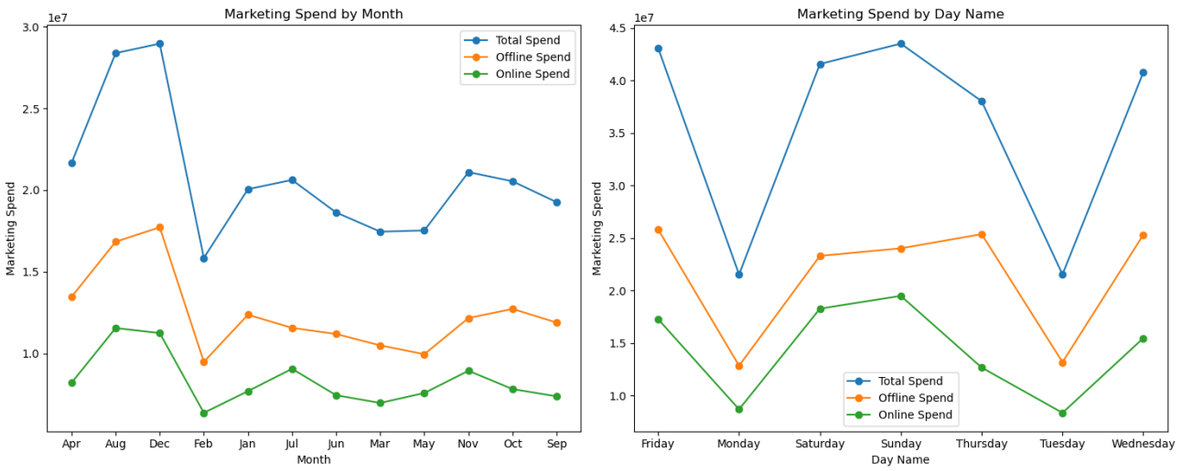


Remarks

The trend of sales is highest on December and it had a major dip in month of June. Also the revenue across each month is displayed above

In [367]:
```
# Analysis of Marketing spend with day and month

fig, axs = plt.subplots(1, 2, figsize=(15, 6))

# Marketing Spend by Month
monthly_spend_month = df_merged_copy.groupby('Month')[['Offline_Spend', 'Online_Spend']].sum()
print(f"Total Marketing Spend by Month:\n{monthly_spend_month}")

axs[0].plot(monthly_spend_month.index, monthly_spend_month['Offline_Spend'] + monthly_spend_month['Online_Spend'],
axs[0].plot(monthly_spend_month.index, monthly_spend_month['Offline_Spend'], marker='o', label='Offline Spend')
axs[0].plot(monthly_spend_month.index, monthly_spend_month['Online_Spend'], marker='o', label='Online Spend')
axs[0].set_xlabel('Month')
axs[0].set_ylabel('Marketing Spend')
axs[0].set_title('Marketing Spend by Month')
axs[0].legend()

# Marketing Spend by Day Name
monthly_spend_day = df_merged_copy.groupby('day_name')[['Offline_Spend', 'Online_Spend']].sum()
print(f"Total Marketing Spend by Day:\n{monthly_spend_day}")

axs[1].plot(monthly_spend_day.index, monthly_spend_day['Offline_Spend'] + monthly_spend_day['Online_Spend'], marke
axs[1].plot(monthly_spend_day.index, monthly_spend_day['Offline_Spend'], marker='o', label='Offline Spend')
axs[1].plot(monthly_spend_day.index, monthly_spend_day['Online_Spend'], marker='o', label='Online Spend')
axs[1].set_xlabel('Day Name')
axs[1].set_ylabel('Marketing Spend')
axs[1].set_title('Marketing Spend by Day Name')
axs[1].legend()

plt.tight_layout()
plt.show()
```

```
Total Marketing Spend by Month:
       Offline_Spend  Online_Spend
Month
Apr         13459000    8196922.13
Aug         16834500   11551233.77
Dec         17720000   11244402.01
Feb          9481400    6360136.05
Jan         12365900    7686875.17
Jul         11563000    9055934.41
Jun         11189000    7436403.73
Mar         10488500    6965280.31
May          9951500    7574021.02
Nov         12161500    8934799.69
Oct         12722500    7813772.39
Sep         11886500    7371126.34
Total Marketing Spend by Day:
            Offline_Spend  Online_Spend
day_name
Friday           25809400   17262619.35
Monday           12839900    8683607.11
Saturday         23298400   18272785.50
Sunday           24017700   19489691.47
Thursday         25366800   12663540.89
Tuesday          13178300    8352505.20
Wednesday        25312800   15466157.50
```

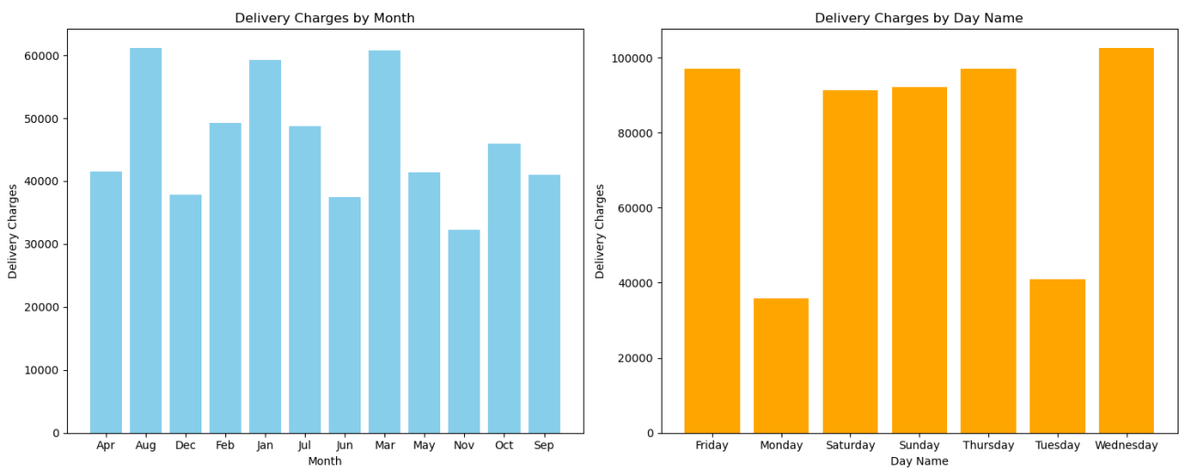


Remarks

1. The online spend amount is more than offline spend. Also the peak is seen in month of December
2. Same trend goes for day wise and also Sunday observed highest marketing spend

In [371]:
```
# Analysis of Delivery Charges with day and month

fig, axs = plt.subplots(1, 2, figsize=(15, 6))

# Delivery Charges by Month
monthly_delivery_charges = df_merged_copy.groupby('Month')['Delivery_Charges'].sum()
print(f"Total Delivery Charges by Month:\n{monthly_delivery_charges}")

axs[0].bar(monthly_delivery_charges.index, monthly_delivery_charges, color='skyblue')
axs[0].set_xlabel('Month')
axs[0].set_ylabel('Delivery Charges')
axs[0].set_title('Delivery Charges by Month')

# Delivery Charges by Day Name
daily_delivery_charges = df_merged_copy.groupby('day_name')['Delivery_Charges'].sum()
print(f"Total Delivery Charges by Day:\n{daily_delivery_charges}")

axs[1].bar(daily_delivery_charges.index, daily_delivery_charges, color='orange')
axs[1].set_xlabel('Day Name')
axs[1].set_ylabel('Delivery Charges')
axs[1].set_title('Delivery Charges by Day Name')

plt.tight_layout()
plt.show()
```

```
Total Delivery Charges by Month:
Month
Apr    41481.74
Aug    61099.57
Dec    37881.99
Feb    49216.60
Jan    59242.32
Jul    48723.93
Jun    37513.58
Mar    60799.94
May    41396.17
Nov    32311.93
Oct    45961.88
Sep    41005.42
Name: Delivery_Charges, dtype: float64
Total Delivery Charges by Day:
day_name
Friday        97066.61
Monday        35781.90
Saturday      91216.38
Sunday        92220.67
Thursday      96984.64
Tuesday       40907.07
Wednesday    102457.80
Name: Delivery_Charges, dtype: float64
```



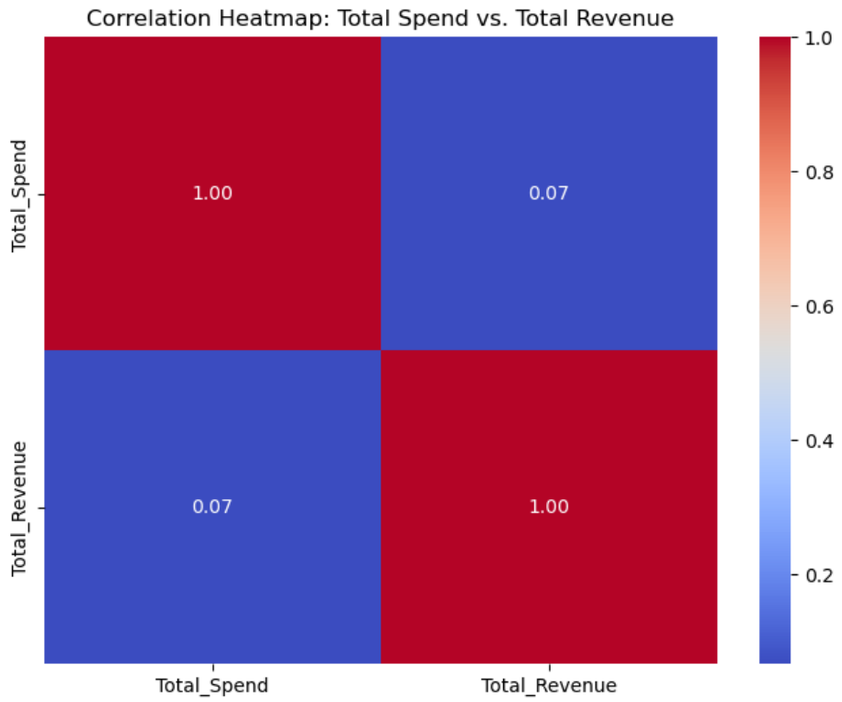
Remarks

1. The delivery charges are higher in month of AUgust and March.
2. The delivery charges are more in Wednesday and lowest on start of week days.

Question: - Assessing the influence of marketing spend on revenue, identifying products in transactions, and determining which products are frequently purchased together through exploratory and market basket analysis are also key steps.

In [374]:
```
# Correlation between Total Spend and Total Revenue

# Calculating Total_Spend (online + offline spend) and Total_Revenue
df_merged_copy1['Total_Spend'] = df_merged_copy1['Online_Spend'] + df_merged_copy1['Offline_Spend']
df_merged_copy1['Total_Revenue'] = df_merged_copy1['Invoice_Value_winsorized']

# Grouping by Month and sum the Total_Spend and Total_Revenue
monthly_data = df_merged_copy1.groupby('Month')[['Total_Spend', 'Total_Revenue']].sum()

# Calculating correlation between Total Spend and Total Revenue
correlation = monthly_data['Total_Spend'].corr(monthly_data['Total_Revenue'])
print(f"Correlation between Total Spend and Total Revenue: {correlation:.2f}")

correlation_matrix = df_merged_copy1[['Total_Spend', 'Total_Revenue']].corr()

plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap: Total Spend vs. Total Revenue')
plt.show()
```

```
Correlation between Total Spend and Total Revenue: 0.55
```

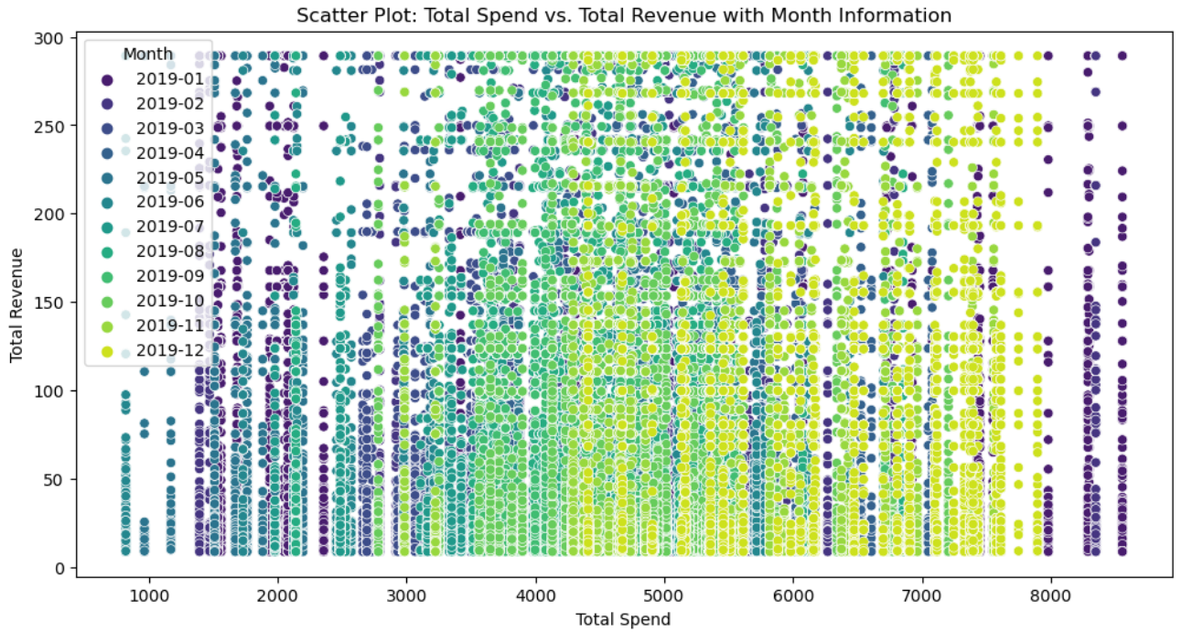


Remarks

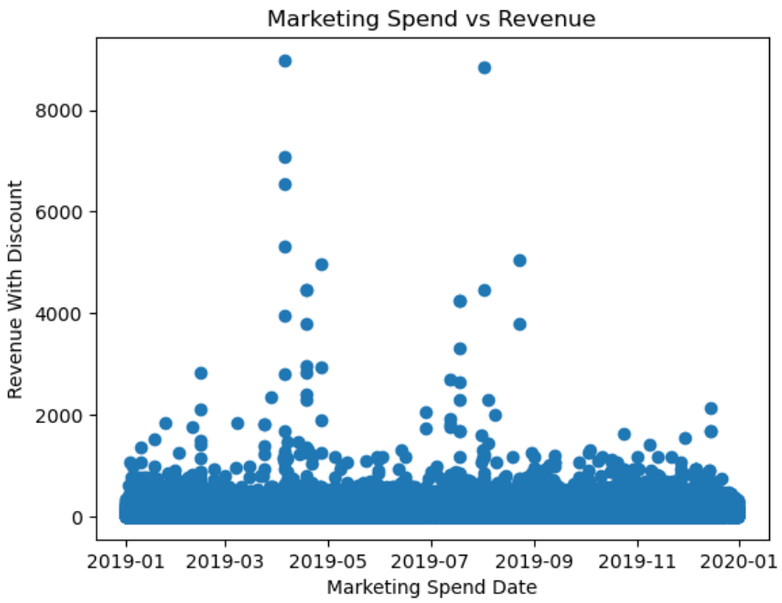
There is a strong correlation between both

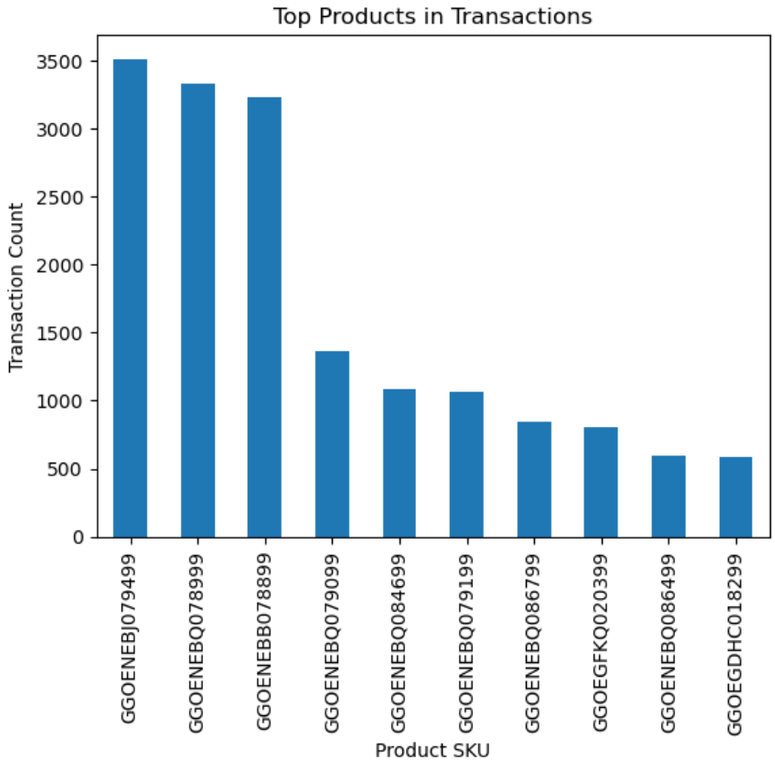
In [375]:
```
# Another way to represent above

df_merged_copy1['Month'] = df_merged_copy1['Marketing_Spend_Date'].dt.to_period('M')

# Grouping by month and calculate the sum for both Total Spend and Total Revenue
monthly_data = df_merged_copy1.groupby('Month')[['Total_Spend', 'Total_Revenue']].sum()

# Calculating correlation between total spend and total revenue
correlation = monthly_data['Total_Spend'].corr(monthly_data['Total_Revenue'])

print(f"Correlation between Total Spend and Total Revenue: {correlation:.2f}")

# Scatter plot to visualize the relationship with Month information
plt.figure(figsize=(12, 6))
sns.scatterplot(x='Total_Spend', y='Total_Revenue', hue='Month', data=df_merged_copy1, palette='viridis', legend='
plt.title('Scatter Plot: Total Spend vs. Total Revenue with Month Information')
plt.xlabel('Total Spend')
plt.ylabel('Total Revenue')
plt.show()
```

```
Correlation between Total Spend and Total Revenue: 0.55
```



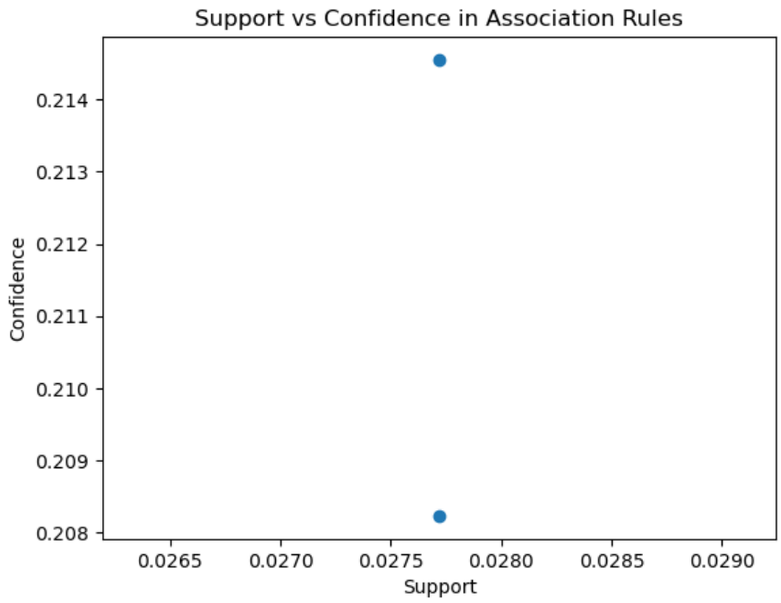
In [376]:
```
import pandas as pd
import matplotlib.pyplot as plt
from mlxtend.frequent_patterns import apriori, association_rules

# Influence of Marketing Spend on Revenue
plt.scatter(df_merged_copy1['Marketing_Spend_Date'], df_merged_copy1['Revenue_With_Discount'])
plt.title('Marketing Spend vs Revenue')
plt.xlabel('Marketing Spend Date')
plt.ylabel('Revenue With Discount')
plt.show()

# Identify products in transactions
product_counts = df_merged_copy1['Product_SKU'].value_counts()
top_products = product_counts.head(10)

# Plotting top products
top_products.plot(kind='bar', title='Top Products in Transactions')
plt.xlabel('Product SKU')
plt.ylabel('Transaction Count')
plt.show()

# Market Basket Analysis
discounted_transactions = df_merged_copy1[df_merged_copy1['Discount_pct'] > 0]

# Creating a binary matrix for market basket analysis
basket = discounted_transactions.groupby(['Transaction_ID', 'Product_SKU'])['Quantity'].sum().unstack().reset_inde

# Converting quantity values to binary for association rules
basket[basket > 0] = 1

# Using Apriori algorithm for frequent itemsets
frequent_itemsets = apriori(basket, min_support=0.02, use_colnames=True)

# Generating association rules
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)

# Displaying association rules
print("Association Rules:")
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])

# Plotting support vs confidence
plt.scatter(rules['support'], rules['confidence'])
plt.title('Support vs Confidence in Association Rules')
plt.xlabel('Support')
plt.ylabel('Confidence')
plt.show()
```



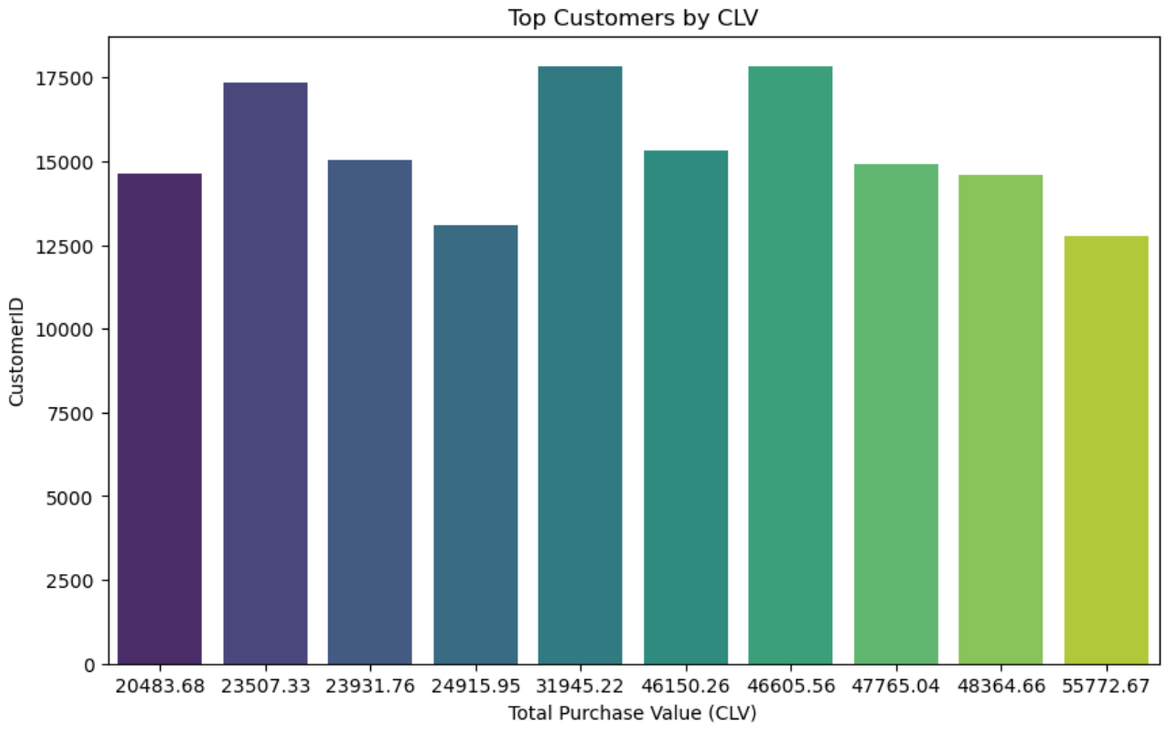



```
C:\Users\Dell\anaconda3\lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:109: DeprecationWarning: DataFram
es with non-bool types result in worse computationalperformance and their support might be discontinued in the fu
ture.Please use a DataFrame with bool type
  warnings.warn(
```

```
Association Rules:
        antecedents       consequents  support  confidence      lift
0  (GGOENEBB078899)  (GGOENEBQ078999)  0.02772    0.214551  1.611712
1  (GGOENEBQ078999)  (GGOENEBB078899)  0.02772    0.208233  1.611712
```



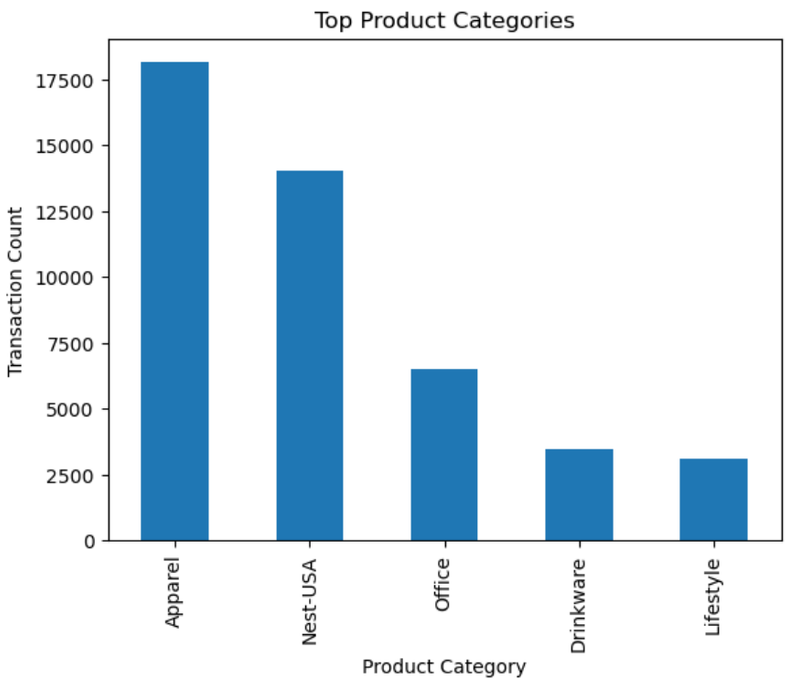
In [377]:
```
# Customer Lifetime Value (CLV) Calculation
df_merged_copy['Total_Purchase_Value'] = df_merged_copy['Invoice_Value_winsorized']  # Use appropriate column for p

clv_df = df_merged_copy.groupby('CustomerID')['Total_Purchase_Value'].sum().reset_index()
clv_df = clv_df.sort_values(by='Total_Purchase_Value', ascending=False)
top_customers = 10
plt.figure(figsize=(10, 6))
sns.barplot(x='Total_Purchase_Value', y='CustomerID', data=clv_df.head(top_customers), palette='viridis')
plt.title('Top Customers by CLV')
plt.xlabel('Total Purchase Value (CLV)')
plt.ylabel('CustomerID')
plt.show()
```



Remarlks

Highest CLV is observed across value 31k and 46k

In [378]:
```
# Identifying top 5 product categories

top_categories = df_merged_copy1['Product_Category'].value_counts().head(5)
top_categories.plot(kind='bar', title='Top Product Categories')
plt.xlabel('Product Category')
plt.ylabel('Transaction Count')
plt.show()
```

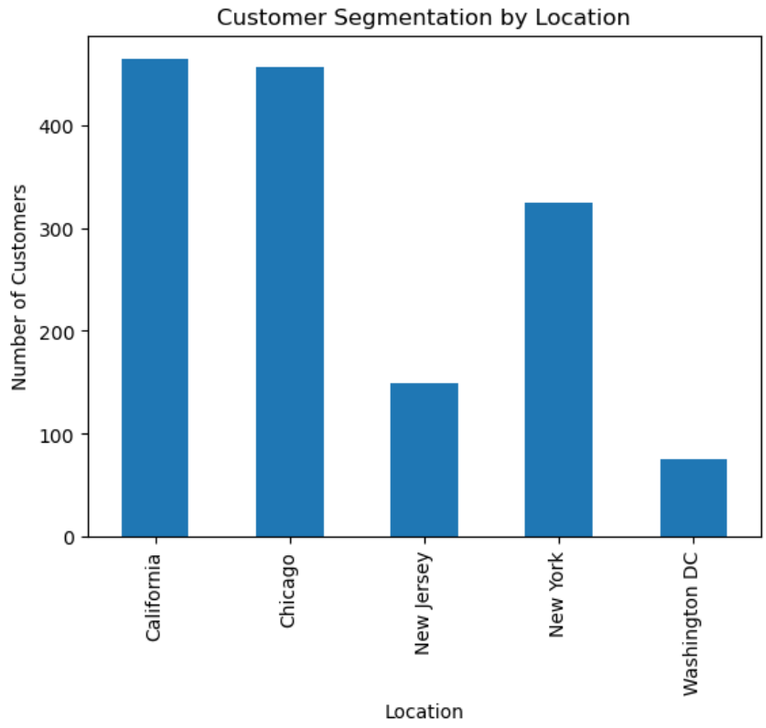


Remarks

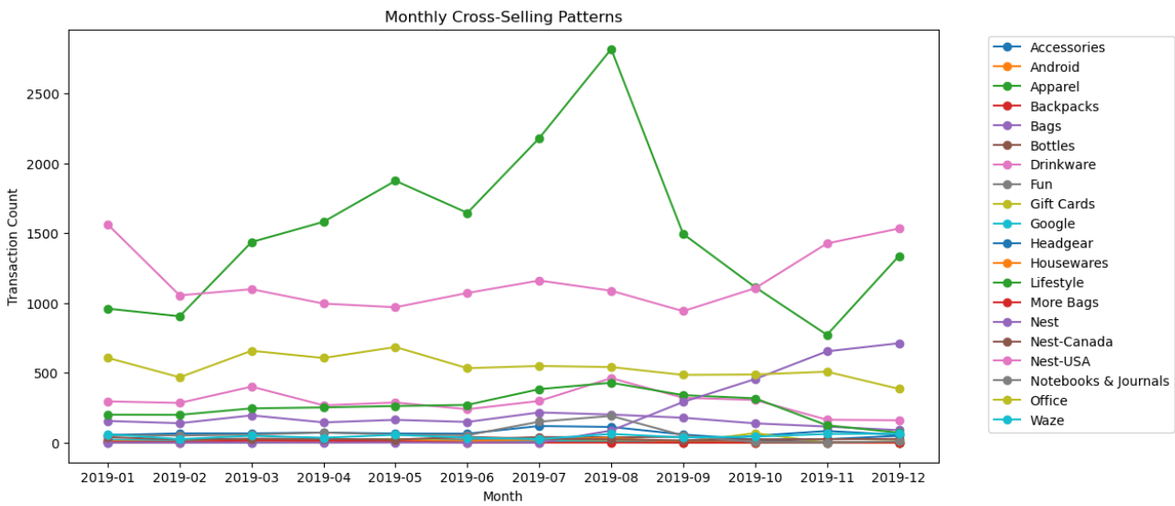
Apparel is famous product category in top 5 list.

Customer Segmentation

In [379]:
```
# Segmenting customers based on Location

location_segments = df_merged_copy1.groupby('Location')['CustomerID'].nunique()
location_segments.plot(kind='bar', title='Customer Segmentation by Location')
plt.xlabel('Location')
plt.ylabel('Number of Customers')
plt.show()
```



In [382]:
```
# Cross-selling patterns over months

df_merged_copy1['Month'] = df_merged_copy1['Month'].astype(str)

# Grouping by Month and Product_Category and count the occurrences
monthly_cross_sell = df_merged_copy1.groupby(['Month', 'Product_Category']).size().unstack(fill_value=0)

plt.figure(figsize=(12, 6))
for product_category in monthly_cross_sell.columns:
    plt.plot(monthly_cross_sell.index, monthly_cross_sell[product_category], marker='o', label=product_category)

plt.title('Monthly Cross-Selling Patterns')
plt.xlabel('Month')
plt.ylabel('Transaction Count')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')  # Adjust legend position
plt.show()
```



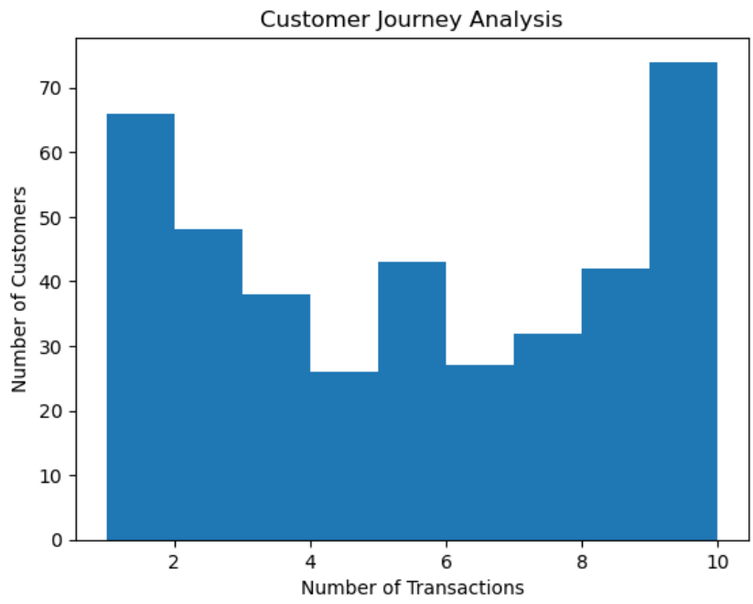
Remarks

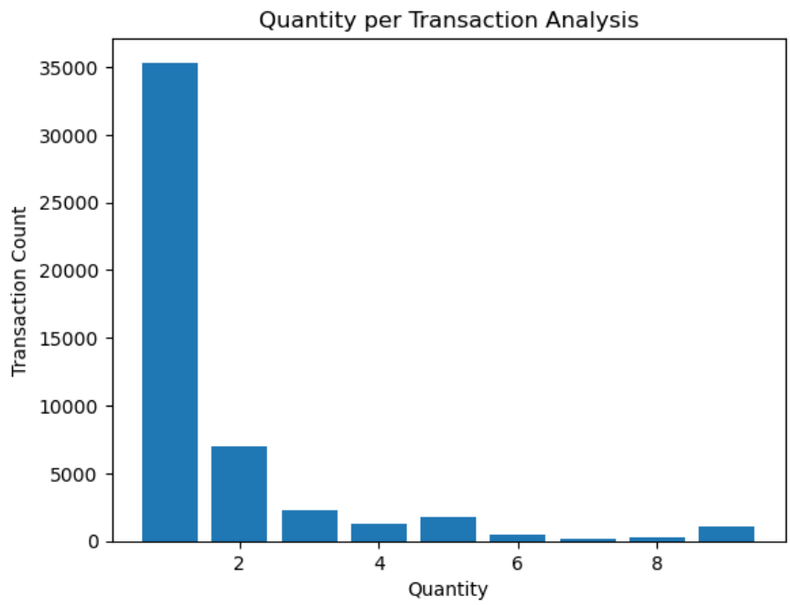
Apparel stands out as famous and it was highly preffered in month of August

In [383]:
```
# Generating Product association rules
frequent_itemsets = apriori(basket, min_support=0.02, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
print("Association Rules:")
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

```
Association Rules:
        antecedents       consequents  support  confidence      lift
0  (GGOENEBB078899)  (GGOENEBQ078999)  0.02772    0.214551  1.611712
1  (GGOENEBQ078999)  (GGOENEBB078899)  0.02772    0.208233  1.611712

C:\Users\Dell\anaconda3\lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:109: DeprecationWarning: DataFram
es with non-bool types result in worse computationalperformance and their support might be discontinued in the fu
ture.Please use a DataFrame with bool type
  warnings.warn(
```

In [384]:
```
# Analyzing the customer journey
customer_journey = df_merged_copy1.groupby('CustomerID')['Transaction_ID'].count()
customer_journey.plot(kind='hist', bins=range(1, 11), title='Customer Journey Analysis')
plt.xlabel('Number of Transactions')
plt.ylabel('Number of Customers')
plt.show()

# Analyzing the distribution of quantity per transaction
plt.hist(df_merged_copy1['Quantity'], bins=range(1, 11), align='left', rwidth=0.8)
plt.title('Quantity per Transaction Analysis')
plt.xlabel('Quantity')
plt.ylabel('Transaction Count')
plt.show()
```





Remarks

1. 1 is the highest quantity ordered
2. 10 is the highest cumulative transaction per customer

In [385]:
```
# Basket Analysis
average_basket_size = df_merged_copy1.groupby('Transaction_ID')['Product_SKU'].nunique().mean()
print(f"Average Basket Size: {average_basket_size}")
```

```
Average Basket Size: 2.111807190455289
```

In [386]:
```
# Product combos

from itertools import combinations

df_merged_copy1['Product_Description'] = df_merged_copy1['Product_Description'].astype(str)
combinations_list = df_merged_copy1.groupby('CustomerID')['Product_Description'].apply(lambda x: list(combinations
flat_combinations = [item for sublist in combinations_list for item in sublist]
combinations_df = pd.DataFrame(flat_combinations, columns=['Product_A', 'Product_B'])

result_df = combinations_df.groupby(['Product_A', 'Product_B']).size().reset_index(name='Total_Purchase_Frequency'

result_df = result_df[result_df['Product_A'] != result_df['Product_B']]

result_df = result_df.sort_values(by='Total_Purchase_Frequency', ascending=False)

top_10_combinations = result_df.head(10)

top_10_combinations
```

Out[386]:

	Product_A	Product_B	Total_Purchase_Frequency
87526	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest Cam Outdoor Security Camera - USA	12897
85621	Nest Cam Indoor Security Camera - USA	Nest Learning Thermostat 3rd Gen-USA - Stainle...	12080
86356	Nest Cam Outdoor Security Camera - USA	Nest Learning Thermostat 3rd Gen-USA - Stainle...	11800
87524	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest Cam Indoor Security Camera - USA	11717
85617	Nest Cam Indoor Security Camera - USA	Nest Cam Outdoor Security Camera - USA	11582
86350	Nest Cam Outdoor Security Camera - USA	Nest Cam Indoor Security Camera - USA	10358
87535	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest Protect Smoke + CO White Battery Alarm-USA	5627
87427	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Google Sunglasses	5560
85626	Nest Cam Indoor Security Camera - USA	Nest Protect Smoke + CO White Battery Alarm-USA	5023
60954	Google Sunglasses	Nest Cam Outdoor Security Camera - USA	4974

Remarks

The combos of products are displayed above and famous is Nest Learning Thermostat 3rd Gen-USA - Stainle... Nest Cam Outdoor Security Camera - USA

In [387]:
```
# Targeting customers which product frequent buy which month, accordingly can give them discounts

customer_month_product_counts = df_merged_copy1.groupby(['CustomerID', 'Product_Description', 'Month']).size().res

# Finding the most frequently purchased product for each customer in each month
top_products_per_month = customer_month_product_counts.groupby(['CustomerID', 'Month']).apply(lambda x: x.loc[x['P

# Seting a threshold for the minimum purchase count to consider a product as frequently purchased
threshold_purchase_count = 10
frequent_purchases = top_products_per_month[top_products_per_month['Purchase_Count'] > threshold_purchase_count]

frequent_purchases[['CustomerID', 'Month', 'Product_Description', 'Purchase_Count']]
```

Out[387]:

	CustomerID	Month	Product_Description	Purchase_Count
208	12748	2019-02	Nest Learning Thermostat 3rd Gen-USA - Stainle...	21
209	12748	2019-03	Nest Cam Indoor Security Camera - USA	12
210	12748	2019-04	Nest Cam Outdoor Security Camera - USA	12
417	13148	2019-11	Nest Cam Outdoor Security Camera - USA	13
426	13174	2019-03	Nest Cam Outdoor Security Camera - USA	17
672	13838	2019-02	Nest Learning Thermostat 3rd Gen-USA - Stainle...	12
748	14049	2019-10	Nest Learning Thermostat 3rd Gen-USA - Stainle...	11
807	14159	2019-05	BLM Sweatshirt	12
958	14505	2019-05	BLM Sweatshirt	21
993	14572	2019-09	Nest Thermostat E - USA	16
1008	14606	2019-07	Nest Cam Outdoor Security Camera - USA	13

Remarks

These are the top customers with highest purchase count over year. We can give them discounts to increase revenue

In [389]:
```
# Inactive customer | CHurn rate

# Convert 'Month' column to datetime format
df_merged_copy1['Month'] = pd.to_datetime(df_merged_copy1['Month'])

# Calculate the last activity month for each customer
last_activity_month = df_merged_copy1.groupby('CustomerID')['Month'].max().reset_index()
last_activity_month.rename(columns={'Month': 'Last_Activity_Month'}, inplace=True)

# Set the churn window (e.g., 6 months)
churn_window = pd.DateOffset(months=6)

# Identifying the customers who haven't made a purchase in the last 6 months
inactive_customers = last_activity_month[last_activity_month['Last_Activity_Month'] <= df_merged_copy1['Month'].ma

print("Inactive Customers (Haven't made a purchase in the last 6 months):")
print(inactive_customers[['CustomerID', 'Last_Activity_Month']].head())
```

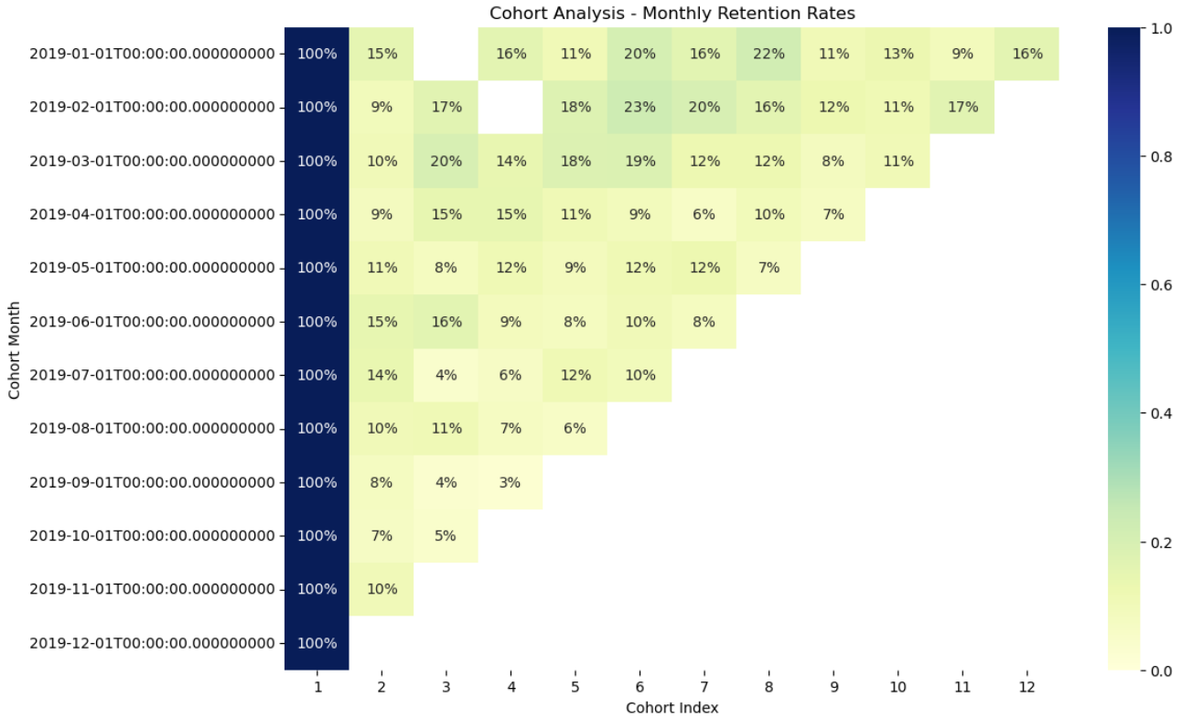
```
Inactive Customers (Haven't made a purchase in the last 6 months):
    CustomerID Last_Activity_Month
6        12370          2019-06-01
24       12429          2019-04-01
26       12433          2019-04-01
27       12434          2019-05-01
29       12441          2019-03-01
```

Remarks

Need to drive campaigns to bring back these customers who may have discontinued buying

Question: - Finally, performing cohort analysis by examining customer behavior based on cohorts starting in each month and identifying the month cohort with maximum retention is integral to gaining comprehensive insights into the business dynamics.

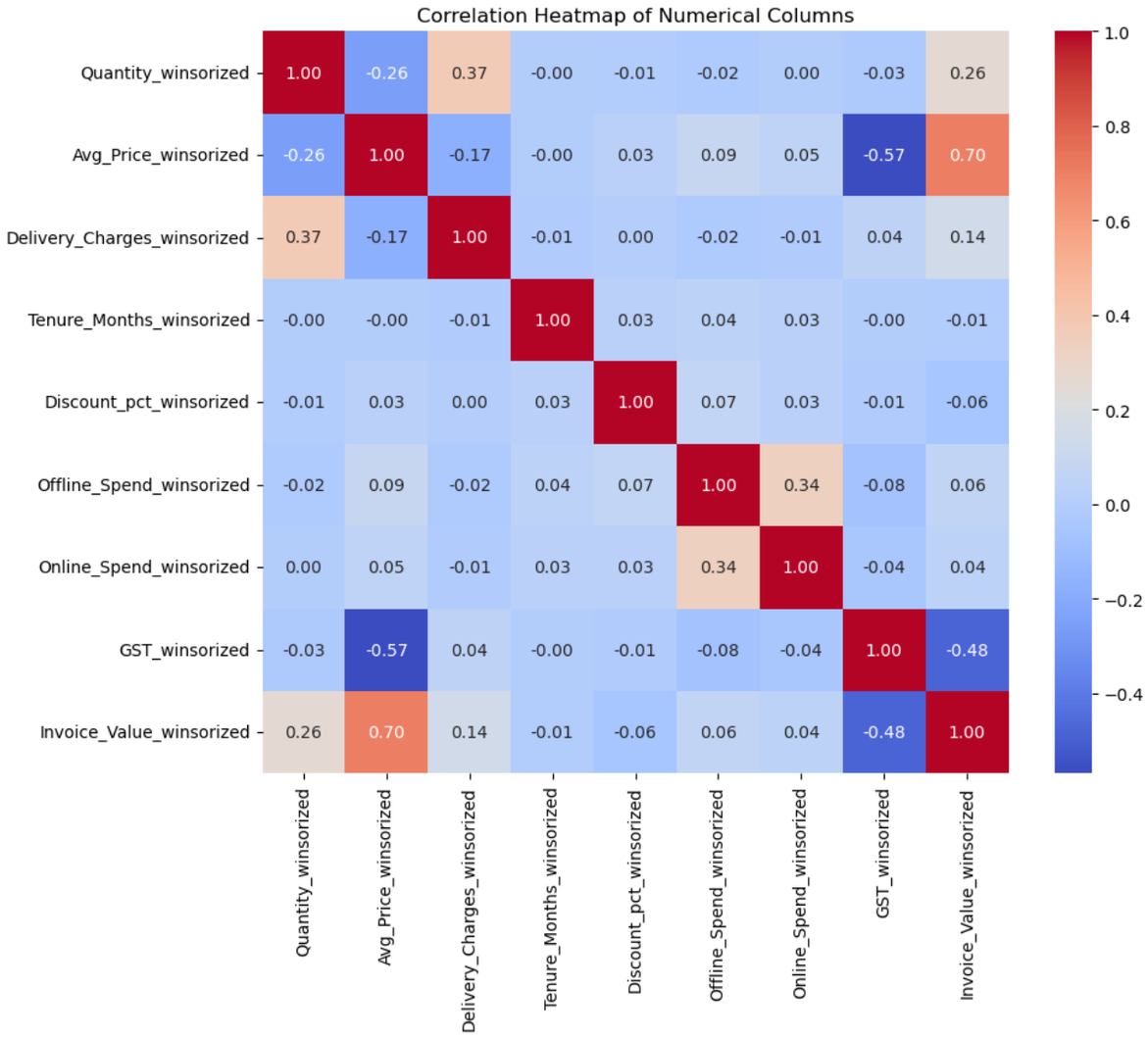
In [171]:
```
# Extracting the month and year from the 'Month' column
df_merged_copy1['OrderMonth'] = df_merged_copy1['Month'].dt.to_period('M')
df_merged_copy1['CohortMonth'] = df_merged_copy1.groupby('CustomerID')['Month'].transform('min').dt.to_period('M')

# Converting 'OrderMonth' and 'CohortMonth' to timestamps
df_merged_copy1['OrderMonth'] = df_merged_copy1['OrderMonth'].dt.to_timestamp()
df_merged_copy1['CohortMonth'] = df_merged_copy1['CohortMonth'].dt.to_timestamp()

# Calculating the time offset for each transaction relative to the cohort month
df_merged_copy1['CohortIndex'] = (df_merged_copy1['OrderMonth'] - df_merged_copy1['CohortMonth']).dt.days // 30 +

# Creating a cohort table
cohort_table = df_merged_copy1.groupby(['CohortMonth', 'CohortIndex'])['CustomerID'].nunique().reset_index()
cohort_table = cohort_table.pivot(index='CohortMonth', columns='CohortIndex', values='CustomerID')

# Calculating retention rates
cohort_size = cohort_table.iloc[:, 0]
retention = cohort_table.divide(cohort_size, axis=0)

plt.figure(figsize=(12, 8))
sns.heatmap(retention, annot=True, fmt='.0%', cmap='YlGnBu', vmin=0, vmax=1)
plt.title('Cohort Analysis - Monthly Retention Rates')
plt.xlabel('Cohort Index')
plt.ylabel('Cohort Month')
plt.show()

# Identifying the month cohort with maximum retention
max_retention_cohort = retention.mean(axis=1).idxmax()
max_retention_rate = retention.mean(axis=1).max()

print(f"The cohort with the maximum retention is {max_retention_cohort}.")
print(f"The maximum retention rate is {max_retention_rate:.2%}.")
```



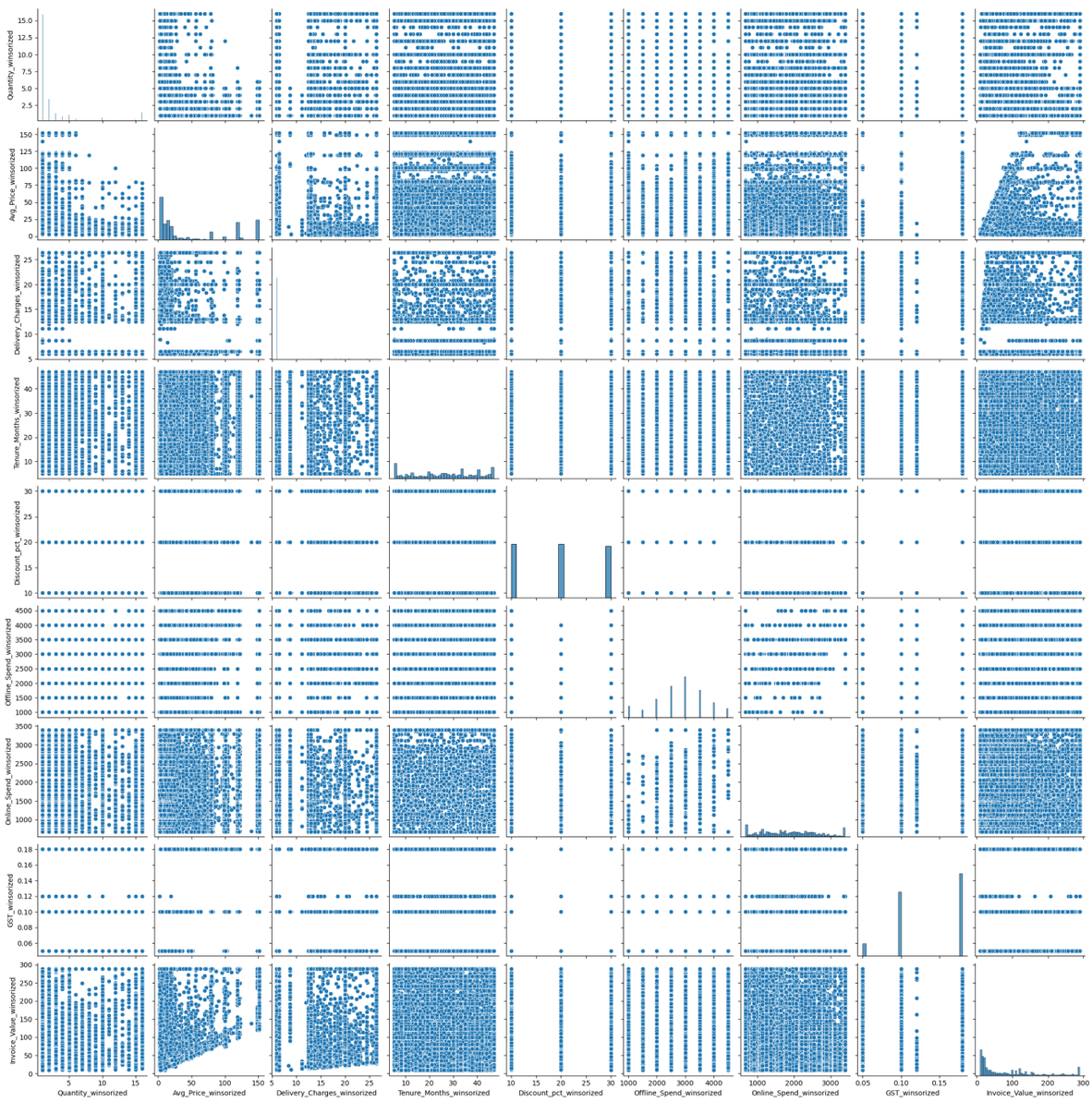

```
The cohort with the maximum retention is 2019-12-01 00:00:00.
The maximum retention rate is 100.00%.
```

Multivariate Analysis

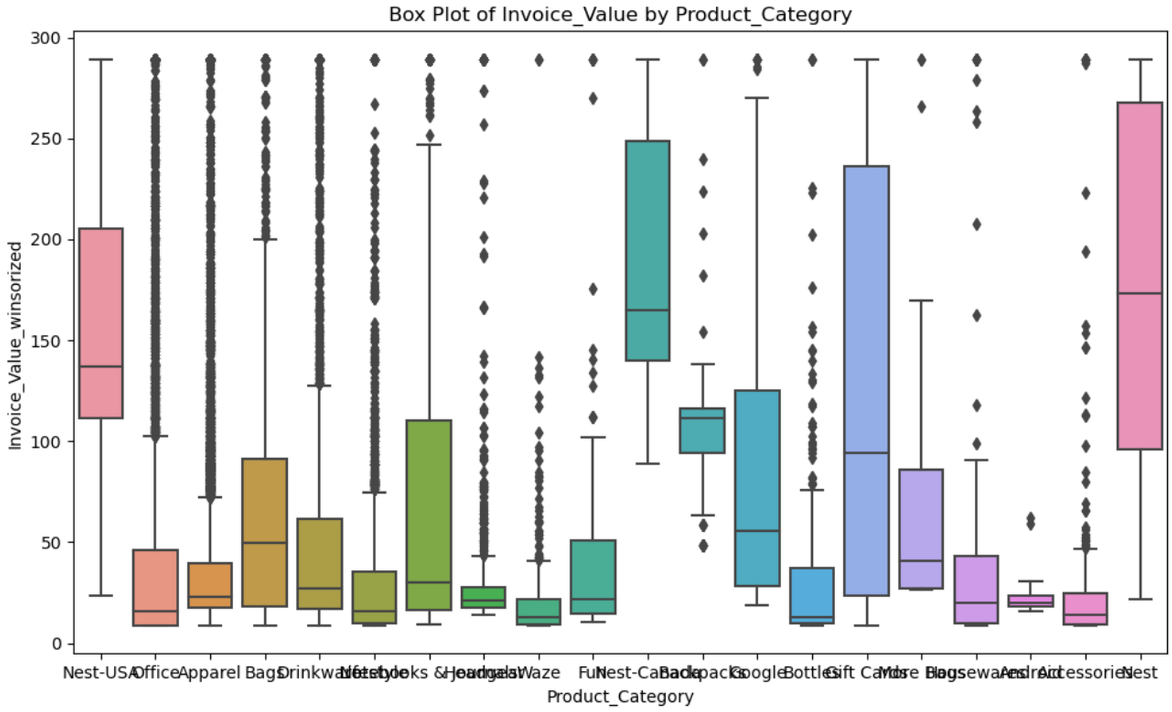
In [327]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming df_merged_copy is your DataFrame
# Modify the column names based on your actual DataFrame columns

# Selecting numerical columns for the heatmap
numerical_columns = [
    'Quantity_winsorized', 'Avg_Price_winsorized', 'Delivery_Charges_winsorized',
    'Tenure_Months_winsorized', 'Discount_pct_winsorized', 'Offline_Spend_winsorized',
    'Online_Spend_winsorized', 'GST_winsorized', 'Invoice_Value_winsorized'
]

# Calculating the correlation matrix
correlation_matrix = df_merged_copy[numerical_columns].corr()

# Creating a heatmap for the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap of Numerical Columns')
plt.show()
```

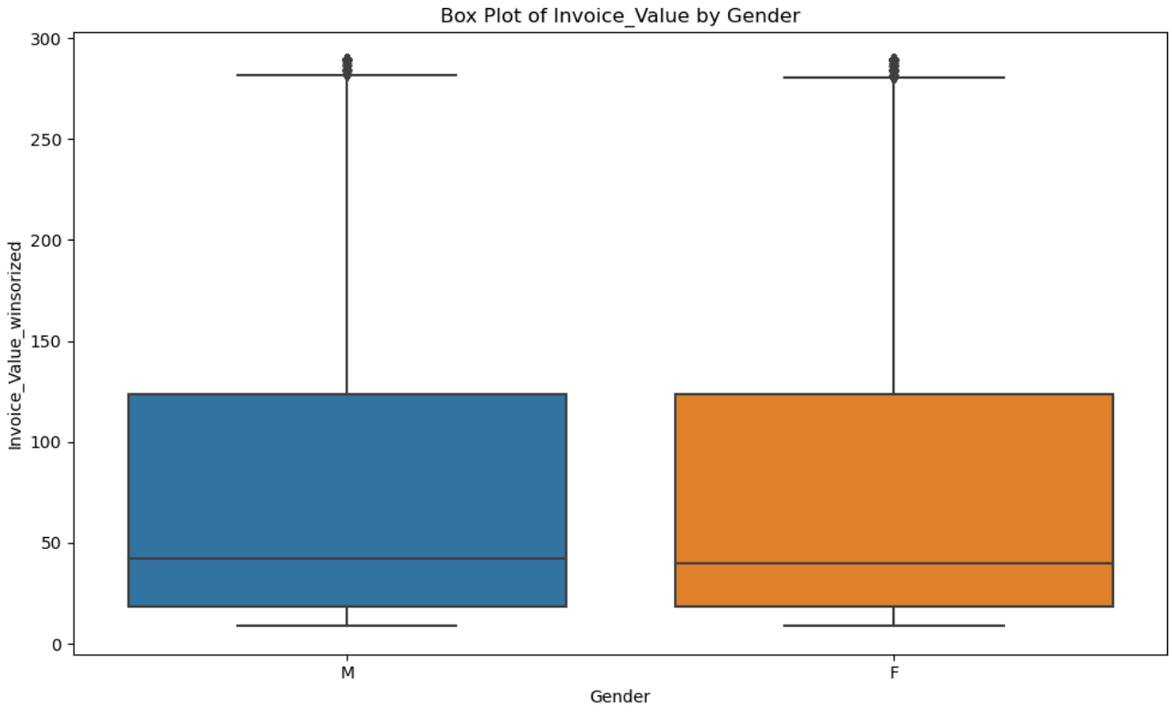


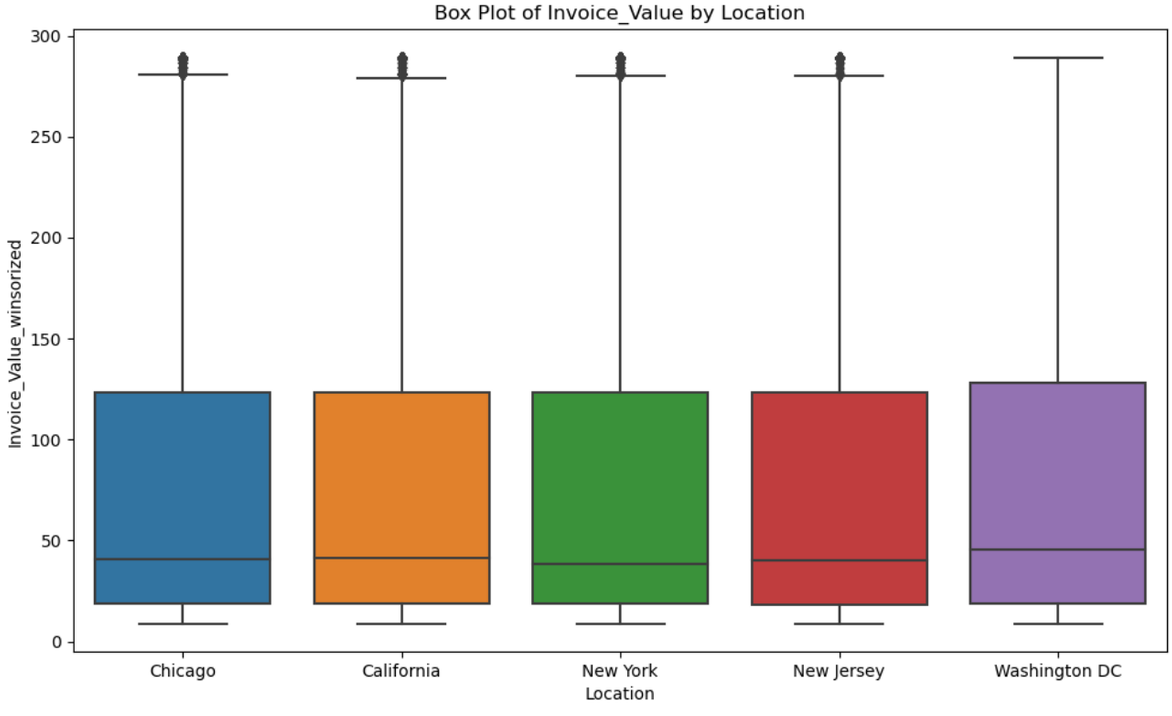
In [328]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming df_merged_copy is your DataFrame
# Modify the column names based on your actual DataFrame columns

# Selecting numerical columns for the pair plot
numerical_columns = [
    'Quantity_winsorized', 'Avg_Price_winsorized', 'Delivery_Charges_winsorized',
    'Tenure_Months_winsorized', 'Discount_pct_winsorized', 'Offline_Spend_winsorized',
    'Online_Spend_winsorized', 'GST_winsorized', 'Invoice_Value_winsorized'
]

# Creating a pair plot for multivariate analysis
sns.pairplot(df_merged_copy[numerical_columns])
plt.show()
```

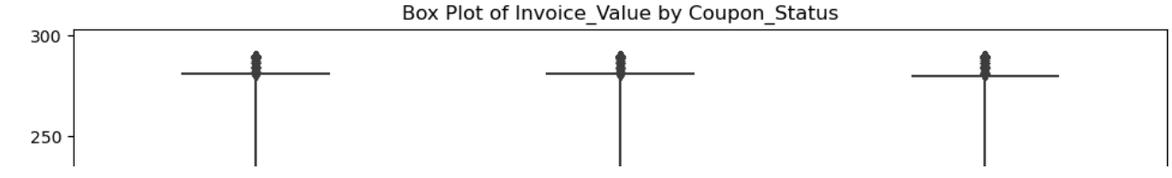


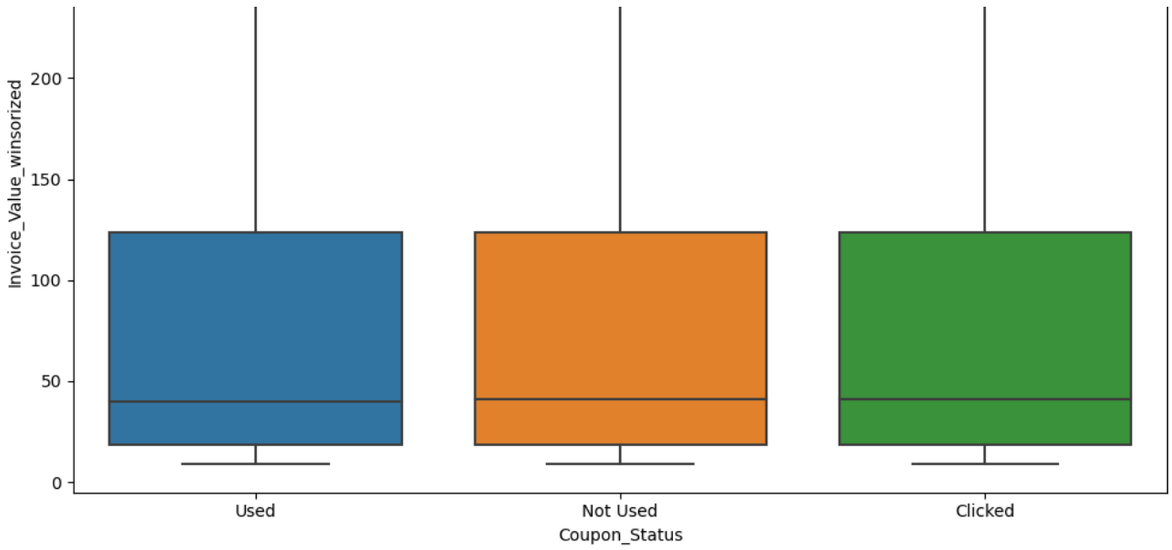
In [329]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming df_merged_copy is your DataFrame
# Modify the column names based on your actual DataFrame columns

# Selecting columns for multivariate analysis
columns_for_multivariate = [
    'Product_Category', 'Gender', 'Location', 'Coupon_Status', 'day_name', 'week_name'
]

# Create subplots
fig, axes = plt.subplots(nrows=len(columns_for_multivariate), ncols=1, figsize=(10, 6 * len(columns_for_multivaria

# Plotting box plots for each combination of numerical and object-type column
for i, column in enumerate(columns_for_multivariate):
    sns.boxplot(x=column, y='Invoice_Value_winsorized', data=df_merged_copy, ax=axes[i])
    axes[i].set_title(f'Box Plot of Invoice_Value by {column}')
    axes[i].set_ylabel('Invoice_Value_winsorized')

# Adjust layout
plt.tight_layout()
plt.show()
```

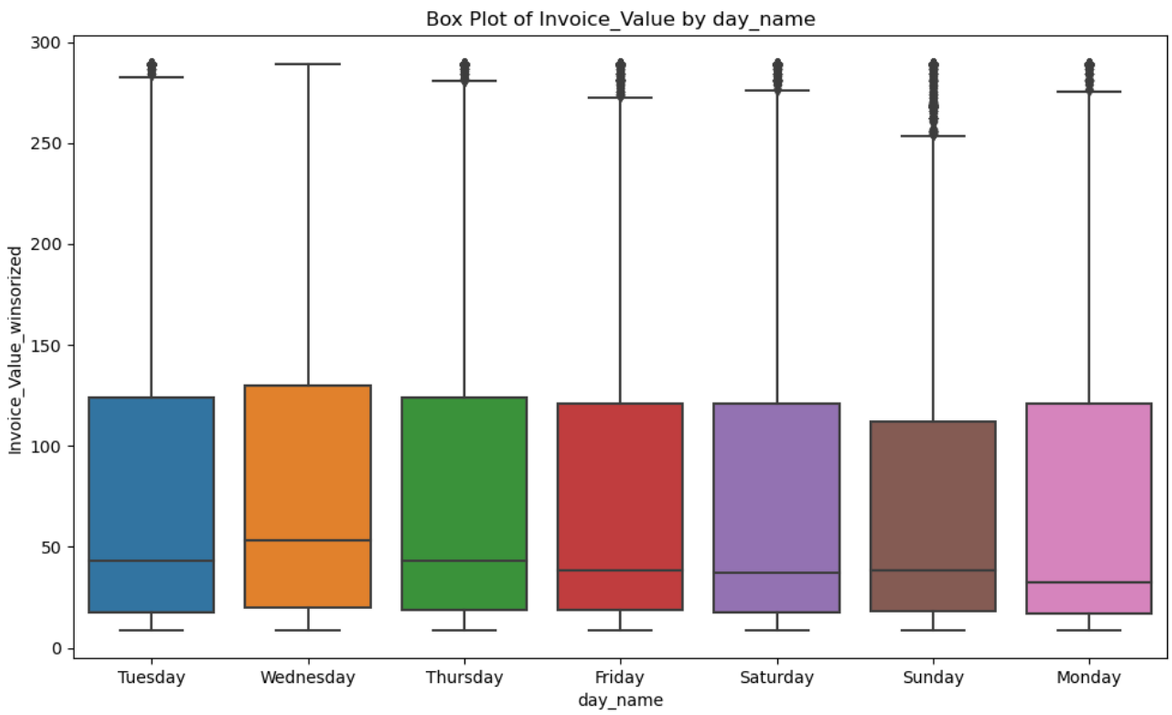

Box Plot of Invoice_Value by Product_Category
Invoice_Value_winsorized
Product_Category
Nest-USA
Office
Apparel
Bags
Drinkware
Waze
Nest-Canada
Backpacks
Google
Bottles
Gift Cards
Mugs
Housewares
Accessories
Nest

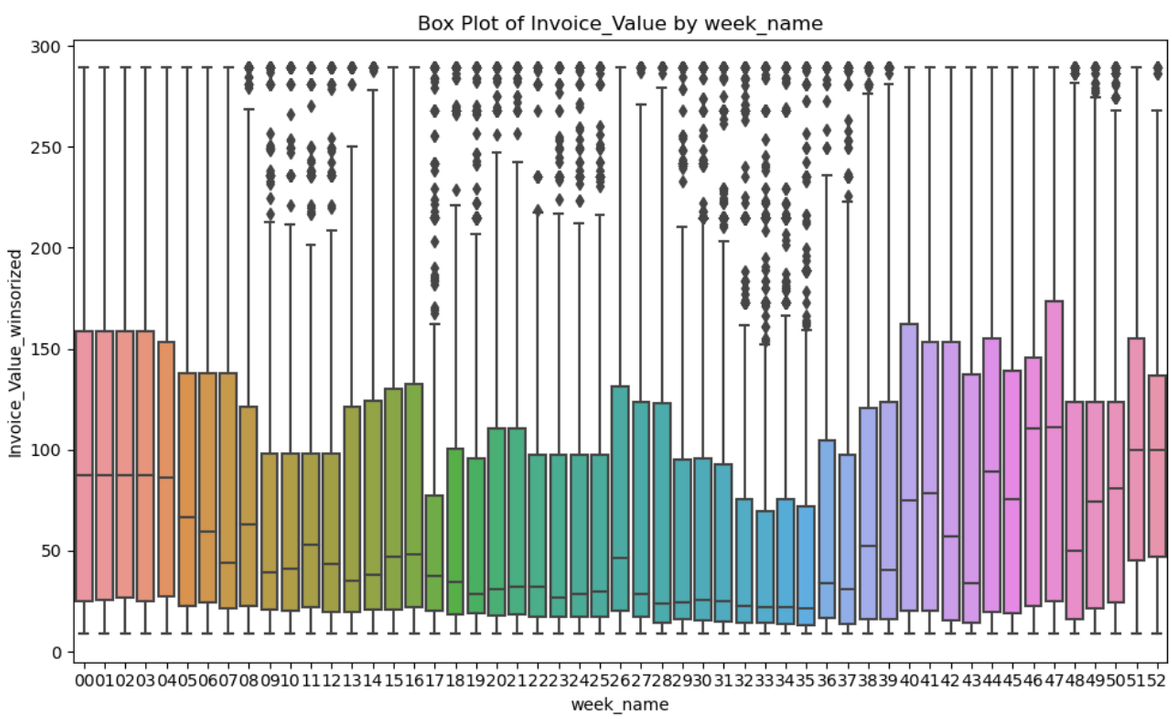

Box Plot of Invoice_Value by Gender
Invoice_Value_winsorized
Gender
M
F


Box Plot of Invoice_Value by Location
Invoice_Value_winsorized
Location
Chicago
California
New York
New Jersey
Washington DC


Box Plot of Invoice_Value by Coupon_Status


Invoice_Value_winsorized
0
50
100
150
200
Used
Not Used
Clicked
Coupon_Status


Box Plot of Invoice_Value by day_name
Invoice_Value_winsorized
0
50
100
150
200
250
300
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Monday
day_name


Box Plot of Invoice_Value by week_name
Invoice_Value_winsorized
0
50
100
150
200
250
300
week_name

Probability Front

In [390]:
```
cross_tab_result = pd.crosstab(index=df_merged_copy1['Gender'], columns=df_merged_copy1['Location'], \
                               margins=True, margins_name='Total')

# Display the cross-tab
print(cross_tab_result)
```

```
Location  California  Chicago  New Jersey  New York  Washington DC  Total
Gender
F               9875    11484        3067      7315           1266  33007
M               6261     6896        1436      3858           1466  19917
Total          16136    18380        4503     11173           2732  52924
```

In [117]:
```
cross_tab_gender_coupon = pd.crosstab(index=df_merged_copy1['Gender'], \
                                      columns=df_merged_copy1['Coupon_Status'], margins=True, margins_name='Total'
print(cross_tab_gender_coupon)
```

```
Coupon_Status  Clicked  Not Used   Used  Total
Gender
F                16788      5067  11152  33007
M                10138      3027   6752  19917
Total            26926      8094  17904  52924
```

In [118]:
```
cross_tab_product_coupon = pd.crosstab(index=df_merged_copy1['Product_Category'], \
                                       columns=df_merged_copy1['Coupon_Status'], margins=True, margins_name='Total
print(cross_tab_product_coupon)
```

```
Coupon_Status         Clicked  Not Used   Used  Total
Product_Category
Accessories               125        32     77    234
Android                    23        10     10     43
Apparel                  9223      2747   6156  18126
Backpacks                  47        15     27     89
Bags                      972       285    625   1882
Bottles                   138        48     82    268
Drinkware                1752       570   1161   3483
Fun                        83        25     52    160
Gift Cards                 81        22     56    159
Google                     51        25     29    105
Headgear                  401       114    256    771
Housewares                 68        18     36    122
Lifestyle                1529       462   1101   3092
More Bags                  25         3     18     46
Nest                     1127       351    720   2198
Nest-Canada               175        47     95    317
Nest-USA                 7141      2160   4712  14013
Notebooks & Journals      389       103    257    749
Office                   3295       968   2250   6513
Waze                      281        89    184    554
Total                   26926      8094  17904  52924
```

In [131]:
```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

cross_tab_product_coupon = pd.crosstab(index=df_merged_copy1['Product_Category'],\
                                       columns=df_merged_copy1['Gender'], margins=True, margins_name='Total')
print(cross_tab_product_coupon)
```

```
Gender                    F      M  Total
Product_Category
Accessories             143     91    234
Android                  26     17     43
Apparel               11355   6771  18126
Backpacks                46     43     89
Bags                   1182    700   1882
Bottles                 174     94    268
Drinkware              2181   1302   3483
Fun                      99     61    160
Gift Cards              119     40    159
Google                   63     42    105
Headgear                504    267    771
Housewares               73     49    122
Lifestyle              1908   1184   3092
More Bags                32     14     46
Nest                   1326    872   2198
Nest-Canada             197    120    317
Nest-USA               8642   5371  14013
Notebooks & Journals    432    317    749
Office                 4178   2335   6513
Waze                    327    227    554
Total                 33007  19917  52924
```

In [136]:
```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

cross_tab_product_coupon = pd.crosstab(index=df_merged_copy1['Product_Category'], \
                                       columns=df_merged_copy1['Location'], margins=True, margins_name='Total')
print(cross_tab_product_coupon)
```

```
Location              California  Chicago  New Jersey  New York  \
Product_Category
Accessories                   80       86          17        41
Android                       13       18           4         7
Apparel                     5491     6158        1627      3902
Backpacks                     29       34           8        14
Bags                         539      731         151       383
Bottles                       91       85          27        54
Drinkware                   1117     1252         284       675
Fun                           44       56          14        32
Gift Cards                    27       89          18        24
Google                        40       33           6        22
Headgear                     221      264          62       191
Housewares                    38       50           4        23
Lifestyle                    977     1086         251       624
More Bags                     15       17           5         8
Nest                         762      710         188       421
Nest-Canada                   91      120          23        63
Nest-USA                    4184     4855        1203      2975
Notebooks & Journals         238      260          41       181
Office                      1993     2273         522      1409
Waze                         146      203          48       124
Total                      16136    18380        4503     11173

Location              Washington DC  Total
Product_Category
Accessories                      10    234
Android                           1     43
Apparel                         948  18126
Backpacks                         4     89
Bags                             78   1882
Bottles                          11    268
Drinkware                       155   3483
Fun                              14    160
Gift Cards                        1    159
Google                            4    105
Headgear                         33    771
Housewares                        7    122
Lifestyle                       154   3092
More Bags                         1     46
Nest                            117   2198
Nest-Canada                      20    317
Nest-USA                        796  14013
Notebooks & Journals             29    749
Office                          316   6513
Waze                             33    554
Total                          2732  52924
```

In [124]:
```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

cross_tab_product_coupon = pd.crosstab(index=df_merged_copy1['Month'], \
                                       columns=df_merged_copy1['Product_Category'], margins=True, margins_name='To
print(cross_tab_product_coupon)
```

```
Product_Category     Accessories  Android  Apparel  Backpacks  Bags  Bottles  \
Month
2019-01-01 00:00:00            1        4      960          3   155       16
2019-02-01 00:00:00            0        1      905          6   139       19
2019-03-01 00:00:00            1        7     1437          7   195       24
2019-04-01 00:00:00            4        2     1582         11   145       20
2019-05-01 00:00:00            6        5     1876         21   163       26
2019-06-01 00:00:00            7        9     1646          4   148       22
2019-07-01 00:00:00            6        8     2182          8   216       40
2019-08-01 00:00:00            8        7     2820         11   201       38
2019-09-01 00:00:00           13        0     1495          6   178       41
2019-10-01 00:00:00           49        0     1114          8   138       22
2019-11-01 00:00:00           83        0      772          4   115        0
2019-12-01 00:00:00           56        0     1337          0    89        0
Total                        234       43    18126         89  1882      268

Product_Category     Drinkware  Fun  Gift Cards  Google  Headgear  Housewares  \
Month
2019-01-01 00:00:00        295   11           4      21        54          10
2019-02-01 00:00:00        284   14           6      15        66           5
2019-03-01 00:00:00        401   17           6      17        66          17
2019-04-01 00:00:00        267   18           7      10        72          10
2019-05-01 00:00:00        287   15          25      11        65          12
2019-06-01 00:00:00        239   17          10      13        64          10
2019-07-01 00:00:00        298   35          10      12       119          22
2019-08-01 00:00:00        462   21          10       6       112          33
2019-09-01 00:00:00        321    9           3       0        58           3
2019-10-01 00:00:00        305    3          68       0        22           0
2019-11-01 00:00:00        164    0           0       0        24           0
2019-12-01 00:00:00        160    0          10       0        49           0
Total                     3483  160         159     105       771         122

Product_Category     Lifestyle  More Bags  Nest  Nest-Canada  Nest-USA  \
Month
2019-01-01 00:00:00        200          5     0           39      1563
2019-02-01 00:00:00        199          4     0           22      1055
2019-03-01 00:00:00        245         15     0           28      1099
2019-04-01 00:00:00        252         13     0           27       996
2019-05-01 00:00:00        262          9     0           21       970
2019-06-01 00:00:00        270          0     0           41      1072
2019-07-01 00:00:00        383          0     0           27      1161
2019-08-01 00:00:00        429          0    86           24      1088
2019-09-01 00:00:00        340          0   291           15       942
2019-10-01 00:00:00        317          0   455           23      1106
2019-11-01 00:00:00        124          0   654           24      1427
2019-12-01 00:00:00         71          0   712           26      1534
Total                     3092         46  2198          317     14013

Product_Category     Notebooks & Journals  Office  Waze  Total
Month
2019-01-01 00:00:00                    54     607    61   4063
2019-02-01 00:00:00                    51     468    25   3284
2019-03-01 00:00:00                    57     658    49   4346
2019-04-01 00:00:00                    73     606    35   4150
2019-05-01 00:00:00                    57     685    56   4572
2019-06-01 00:00:00                    53     533    35   4193
2019-07-01 00:00:00                   150     549    25   5251
2019-08-01 00:00:00                   191     541    62   6150
2019-09-01 00:00:00                    51     485    37   4288
2019-10-01 00:00:00                     3     488    43   4164
2019-11-01 00:00:00                     0     508    62   3961
2019-12-01 00:00:00                     9     385    64   4502
Total                                 749    6513   554  52924
```

In [125]:
```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

cross_tab_product_coupon = pd.crosstab(index=df_merged_copy1['Quantity'],\
                                       columns=df_merged_copy1['Product_Category'], margins=True, margins_name='To
print(cross_tab_product_coupon)
```

```
81                      0         0    0           0        0
82                      0         0    0           0        0
83                      1         0    0           0        0
84                      0         0    0           0        0
85                      0         0    0           0        0
88                      0         0    0           0        0
90                      0         0    0           0        0
91                      0         0    0           0        0
93                      0         0    0           0        0
95                      0         0    0           0        0
96                      0         0    0           0        0
98                      0         0    0           0        0
100                    16         0    0           0        0
101                     0         0    0           0        0
102                     4         0    0           0        0
103                     1         0    0           0        0
104                     0         0    0           0        0
105                     0         0    0           0        0
108                     0         0    0           0        0
109                     0         0    0           0        0
```

In [134]:
```
# Calculate conditional probability for each combination of variables
conditional_probabilities = round(pd.crosstab(index=df_merged_copy1['Gender'], \
                                              columns=df_merged_copy1['Location'], normalize='index') * 100,2)
# Display the conditional probabilities
print(conditional_probabilities)
```

```
Location  California  Chicago  New Jersey  New York  Washington DC
Gender
F              29.92    34.79        9.29     22.16           3.84
M              31.44    34.62        7.21     19.37           7.36
```

In [133]:
```
# Calculate conditional probability for each combination of variables
conditional_probabilities = round(pd.crosstab(index=df_merged_copy1['Product_Category'],\
                                              columns=df_merged_copy1['Gender'], normalize='index') * 100,2)
# Display the conditional probabilities
print(conditional_probabilities)
```

```
Gender                   F      M
Product_Category
Accessories          61.11  38.89
Android              60.47  39.53
Apparel              62.64  37.36
Backpacks            51.69  48.31
Bags                 62.81  37.19
Bottles              64.93  35.07
Drinkware            62.62  37.38
Fun                  61.88  38.12
Gift Cards           74.84  25.16
Google               60.00  40.00
Headgear             65.37  34.63
Housewares           59.84  40.16
Lifestyle            61.71  38.29
More Bags            69.57  30.43
Nest                 60.33  39.67
Nest-Canada          62.15  37.85
Nest-USA             61.67  38.33
Notebooks & Journals 57.68  42.32
Office               64.15  35.85
Waze                 59.03  40.97
```

In [135]:
```
# Calculate conditional probability for each combination of variables
conditional_probabilities = round(pd.crosstab(index=df_merged_copy1['Month'], \
                                              columns=df_merged_copy1['Product_Category'], normalize='index') * 10
# Display the conditional probabilities
print(conditional_probabilities)
```

```
Product_Category  Accessories  Android  Apparel  Backpacks  Bags  Bottles  \
Month
2019-01-01               0.02     0.10    23.63       0.07  3.81     0.39
2019-02-01               0.00     0.03    27.56       0.18  4.23     0.58
2019-03-01               0.02     0.16    33.06       0.16  4.49     0.55
2019-04-01               0.10     0.05    38.12       0.27  3.49     0.48
2019-05-01               0.13     0.11    41.03       0.46  3.57     0.57
2019-06-01               0.17     0.21    39.26       0.10  3.53     0.52
2019-07-01               0.11     0.15    41.55       0.15  4.11     0.76
2019-08-01               0.13     0.11    45.85       0.18  3.27     0.62
2019-09-01               0.30     0.00    34.86       0.14  4.15     0.96
2019-10-01               1.18     0.00    26.75       0.19  3.31     0.53
2019-11-01               2.10     0.00    19.49       0.10  2.90     0.00
2019-12-01               1.24     0.00    29.70       0.00  1.98     0.00

Product_Category  Drinkware   Fun  Gift Cards  Google  Headgear  Housewares  \
Month
2019-01-01             7.26  0.27        0.10    0.52      1.33        0.25
2019-02-01             8.65  0.43        0.18    0.46      2.01        0.15
2019-03-01             9.23  0.39        0.14    0.39      1.52        0.39
2019-04-01             6.43  0.43        0.17    0.24      1.73        0.24
2019-05-01             6.28  0.33        0.55    0.24      1.42        0.26
2019-06-01             5.70  0.41        0.24    0.31      1.53        0.24
2019-07-01             5.68  0.67        0.19    0.23      2.27        0.42
2019-08-01             7.51  0.34        0.16    0.10      1.82        0.54
2019-09-01             7.49  0.21        0.07    0.00      1.35        0.07
2019-10-01             7.32  0.07        1.63    0.00      0.53        0.00
2019-11-01             4.14  0.00        0.00    0.00      0.61        0.00
2019-12-01             3.55  0.00        0.22    0.00      1.09        0.00

Product_Category  Lifestyle  More Bags   Nest  Nest-Canada  Nest-USA  \
Month
2019-01-01             4.92       0.12   0.00         0.96     38.47
2019-02-01             6.06       0.12   0.00         0.67     32.13
2019-03-01             5.64       0.35   0.00         0.64     25.29
2019-04-01             6.07       0.31   0.00         0.65     24.00
2019-05-01             5.73       0.20   0.00         0.46     21.22
2019-06-01             6.44       0.00   0.00         0.98     25.57
2019-07-01             7.29       0.00   0.00         0.51     22.11
2019-08-01             6.98       0.00   1.40         0.39     17.69
2019-09-01             7.93       0.00   6.79         0.35     21.97
2019-10-01             7.61       0.00  10.93         0.55     26.56
2019-11-01             3.13       0.00  16.51         0.61     36.03
2019-12-01             1.58       0.00  15.82         0.58     34.07

Product_Category  Notebooks & Journals  Office  Waze
Month
2019-01-01                        1.33   14.94  1.50
2019-02-01                        1.55   14.25  0.76
2019-03-01                        1.31   15.14  1.13
2019-04-01                        1.76   14.60  0.84
2019-05-01                        1.25   14.98  1.22
2019-06-01                        1.26   12.71  0.83
2019-07-01                        2.86   10.46  0.48
2019-08-01                        3.11    8.80  1.01
2019-09-01                        1.19   11.31  0.86
2019-10-01                        0.07   11.72  1.03
2019-11-01                        0.00   12.83  1.57
2019-12-01                        0.20    8.55  1.42
```

In [137]:
```
# Calculate conditional probability for each combination of variables
conditional_probabilities = round(pd.crosstab(index=df_merged_copy1['Product_Category'],\
                                              columns=df_merged_copy1['Location'], normalize='index') * 100,2)
# Display the conditional probabilities
print(conditional_probabilities)
```

```
Location             California  Chicago  New Jersey  New York  Washington DC
Product_Category
Accessories               34.19    36.75        7.26     17.52           4.27
Android                   30.23    41.86        9.30     16.28           2.33
Apparel                   30.29    33.97        8.98     21.53           5.23
Backpacks                 32.58    38.20        8.99     15.73           4.49
Bags                      28.64    38.84        8.02     20.35           4.14
Bottles                   33.96    31.72       10.07     20.15           4.10
Drinkware                 32.07    35.95        8.15     19.38           4.45
Fun                       27.50    35.00        8.75     20.00           8.75
Gift Cards                16.98    55.97       11.32     15.09           0.63
Google                    38.10    31.43        5.71     20.95           3.81
Headgear                  28.66    34.24        8.04     24.77           4.28
Housewares                31.15    40.98        3.28     18.85           5.74
Lifestyle                 31.60    35.12        8.12     20.18           4.98
More Bags                 32.61    36.96       10.87     17.39           2.17
Nest                      34.67    32.30        8.55     19.15           5.32
Nest-Canada               28.71    37.85        7.26     19.87           6.31
Nest-USA                  29.86    34.65        8.58     21.23           5.68
Notebooks & Journals      31.78    34.71        5.47     24.17           3.87
Office                    30.60    34.90        8.01     21.63           4.85
Waze                      26.35    36.64        8.66     22.38           5.96
```

In [141]:
```
# Calculate conditional probability for each combination of variables
conditional_probabilities = round(pd.crosstab(index=df_merged_copy1['Gender'],\
                                              columns=df_merged_copy1['Location'], normalize='index') * 100,2)
# Display the conditional probabilities
print(conditional_probabilities)
```

Location	California	Chicago	New Jersey	New York	Washington DC
Gender					
F	29.92	34.79	9.29	22.16	3.84
M	31.44	34.62	7.21	19.37	7.36

Hypothesis Testing

In [148]:
```
# Is there a statistically significant difference in the average purchase amount between different customer segment
# Hypothesis Statement: The average purchase amount differs significantly between different customer segments.

import scipy.stats as stats

# Assuming 'YourDataFrame' is the name of your DataFrame
result_anova = stats.f_oneway(df_merged_copy1['Avg_Price'][df_merged_copy1['Gender'] == 'M'],
                              df_merged_copy1['Avg_Price'][df_merged_copy1['Gender'] == 'F'])

print(result_anova)

# Display results
print("ANOVA p-value:", result_anova.pvalue)

if result_anova.pvalue < 0.05:
    print("Conclusion: Reject the null hypothesis. There is a significant difference in average prices between mal
else:
    print("Conclusion: Fail to reject the null hypothesis. There is no significant difference in average prices be
```

```
F_onewayResult(statistic=8.339949524711047, pvalue=0.0038798386725696107)
ANOVA p-value: 0.0038798386725696107
Conclusion: Reject the null hypothesis. There is a significant difference in average prices between male and fema
le customers.
```

In [152]:
```
# Null Hypothesis (H0): There is no significant difference in average purchase amount between genders.
# Alternative Hypothesis (H1): There is a significant difference in average purchase amount between genders.

from scipy.stats import ttest_ind

# Split data by gender
male_data = df_merged_copy1[df_merged_copy1['Gender'] == 'M']['Invoice_Value']
female_data = df_merged_copy1[df_merged_copy1['Gender'] == 'F']['Invoice_Value']

# Perform t-test
t_stat, p_value = ttest_ind(male_data, female_data, equal_var=False)

# Print results
print("t-statistic:", t_stat)
print("p-value:", p_value)

# Conclusion
if p_value < 0.05:
    print("Reject the null hypothesis. There is a significant difference in average purchase amount between gender
else:
    print("Fail to reject the null hypothesis. There is no significant difference in average purchase amount betwe
```

```
t-statistic: 0.17356375237673308
p-value: 0.8622091346725185
Fail to reject the null hypothesis. There is no significant difference in average purchase amount between gender
s.
```

In [154]:
```
# Null Hypothesis (H0): There is no significant difference in average purchase amount between customers with and wi
# Alternative Hypothesis (H1): There is a significant difference in average purchase amount between customers with

from scipy.stats import ttest_ind

# Split data by coupon status
with_coupon = df_merged_copy1[df_merged_copy1['Coupon_Status'] == 'Used']['Invoice_Value']
without_coupon = df_merged_copy1[df_merged_copy1['Coupon_Status'] == 'Not Used']['Invoice_Value']

# Perform t-test
t_stat, p_value = ttest_ind(with_coupon, without_coupon, equal_var=False)

# Print results
print("t-statistic:", t_stat)
print("p-value:", p_value)

# Conclusion
if p_value < 0.05:
    print("Reject the null hypothesis. There is a significant difference in average purchase amount between custome
else:
    print("Fail to reject the null hypothesis. There is no significant difference in average purchase amount betwe
```

```
t-statistic: -1.2940287556715202
p-value: 0.19567493052486554
Fail to reject the null hypothesis. There is no significant difference in average purchase amount between custome
rs with and without coupons.
```

In [391]:
```
# Null Hypothesis (H0): There is no significant difference in the quantity sold across different product categories
# Alternative Hypothesis (H1): There is a significant difference in the quantity sold across different product cate

from scipy.stats import f_oneway

# Perform one-way ANOVA
category_groups = [df_merged_copy1[df_merged_copy1['Product_Category'] == category]['Quantity'] for category \
                   in df_merged_copy1['Product_Category'].unique()]
f_stat, p_value = f_oneway(*category_groups)

# Print results
print("F-statistic:", f_stat)
print("p-value:", p_value)

# Conclusion
if p_value < 0.05:
    print("Reject the null hypothesis. There is a significant difference in quantity sold across different product
else:
    print("Fail to reject the null hypothesis. There is no significant difference in quantity sold across differen
```

```
F-statistic: 141.73228416810764
p-value: 0.0
Reject the null hypothesis. There is a significant difference in quantity sold across different product categorie
s.
```

In [156]:
```
# Null Hypothesis (H0): There is no significant difference in average revenue across different locations.
# Alternative Hypothesis (H1): There is a significant difference in average revenue across different locations.

from scipy.stats import f_oneway

# Perform one-way ANOVA
location_groups = [df_merged_copy1[df_merged_copy1['Location'] == location]['Revenue_With_Discount'].dropna() \
                   for location in df_merged_copy1['Location'].unique()]
f_stat, p_value = f_oneway(*location_groups)

# Print results
print("F-statistic:", f_stat)
print("p-value:", p_value)

# Conclusion
if p_value < 0.05:
    print("Reject the null hypothesis. There is a significant difference in average revenue across different locat
else:
    print("Fail to reject the null hypothesis. There is no significant difference in average revenue across differ
```

```
F-statistic: 2.701953854090806
p-value: 0.02882189631217692
Reject the null hypothesis. There is a significant difference in average revenue across different locations.
```

In [161]:
```
# Null Hypothesis (H0): There is no significant difference in average purchase amount between weekends and weekdays
# Alternative Hypothesis (H1): There is a significant difference in average purchase amount between weekends and we

from scipy.stats import ttest_ind

# Split data by day type
weekends = df_merged_copy1[df_merged_copy1['day_name'].isin(['Saturday', 'Sunday'])]['Invoice_Value']
weekdays = df_merged_copy1[~df_merged_copy1['day_name'].isin(['Saturday', 'Sunday'])]['Invoice_Value']

# Perform t-test
t_stat, p_value = ttest_ind(weekends, weekdays, equal_var=False)

# Print results
print("t-statistic:", t_stat)
print("p-value:", p_value)

# Conclusion
if p_value < 0.05:
    print("Reject the null hypothesis. There is a significant difference in average purchase amount between weeken
else:
    print("Fail to reject the null hypothesis. There is no significant difference in average purchase amount betwe
```

```
t-statistic: -5.883667684414874
p-value: 4.043035481772227e-09
Reject the null hypothesis. There is a significant difference in average purchase amount between weekends and wee
kdays.
```

In [159]:
```
# Null Hypothesis (H0): There is no significant difference in average invoice value across different days.
# Alternative Hypothesis (H1): There is a significant difference in average invoice value across different days.

from scipy.stats import f_oneway

# Perform one-way ANOVA
day_groups = [df_merged_copy1[df_merged_copy1['day_name'] == day]['Invoice_Value'] for day in df_merged_copy1['day
f_stat, p_value = f_oneway(*day_groups)

# Print results
print("F-statistic:", f_stat)
print("p-value:", p_value)

# Conclusion
if p_value < 0.05:
    print("Reject the null hypothesis. There is a significant difference in average invoice value across different
else:
    print("Fail to reject the null hypothesis. There is no significant difference in average invoice value across
```

```
F-statistic: 12.07459347138538
p-value: 1.3144845984664848e-13
Reject the null hypothesis. There is a significant difference in average invoice value across different days.
```

In [168]:
```
# Null Hypothesis (H0): There is no significant difference in marketing spend across different product categories.
# Alternative Hypothesis (H1): There is a significant difference in marketing spend across different product catego

from scipy.stats import f_oneway

# Convert 'Marketing_Spend_Month' to ordinal representation
df_merged_copy1['Marketing_Spend_Month'] = df_merged_copy1['Marketing_Spend_Month'].dt.to_timestamp().apply(lambda

# Perform one-way ANOVA
category_groups = [df_merged_copy1[df_merged_copy1['Product_Category'] == category]['Marketing_Spend_Month'] for ca
f_stat, p_value = f_oneway(*category_groups)

# Print results
print("F-statistic:", f_stat)
print("p-value:", p_value)

# Conclusion
if p_value < 0.05:
    print("Reject the null hypothesis. There is a significant difference in marketing spend across different produc
else:
    print("Fail to reject the null hypothesis. There is no significant difference in marketing spend across differe
```

```
F-statistic: 234.20641818025675
p-value: 0.0
Reject the null hypothesis. There is a significant difference in marketing spend across different product categor
ies.
```

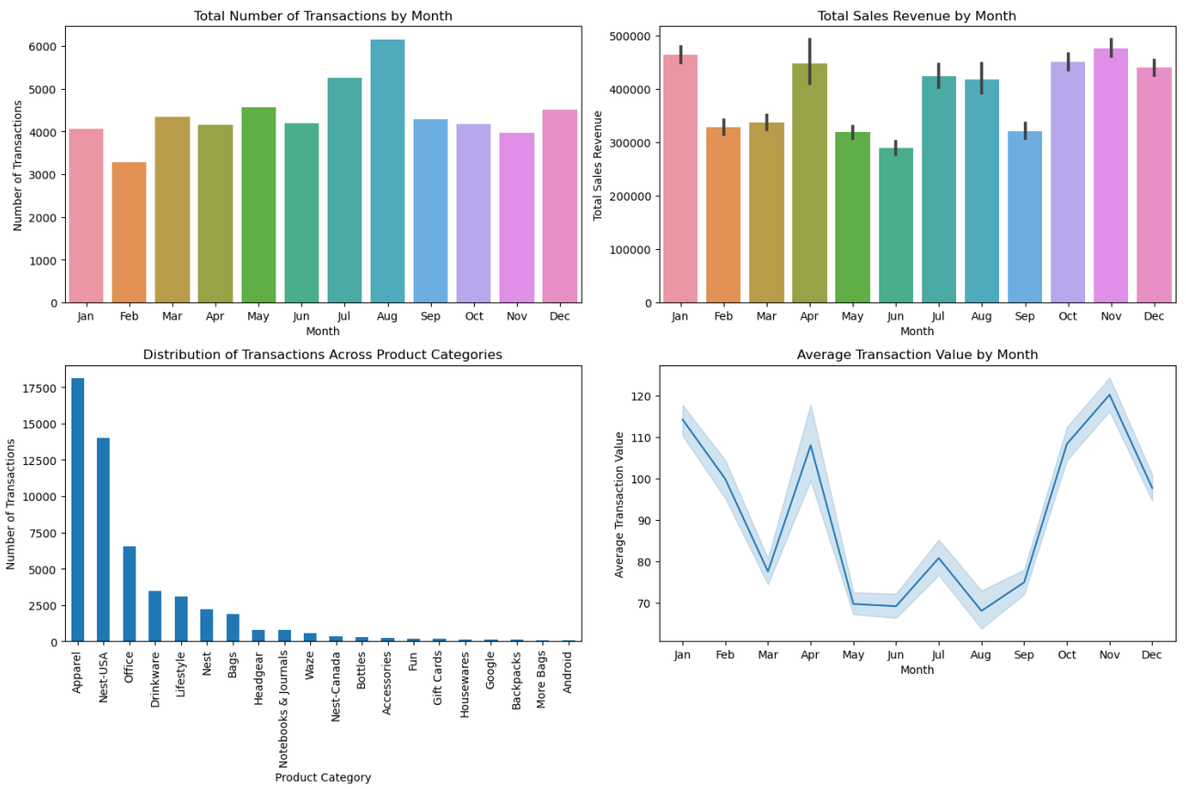
Additional Pointers | Presentation Front

In [236]:
```
df_merged_copy2=df_merged_copy.copy()
df_merged_copy2.head(2)
```

Out[236]:

	CustomerID	Transaction_ID	Transaction_Date	Product_SKU	Product_Description	Product_Category	Quantity	Avg_Price	Delivery_Ch
0	17850	16679	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	
1	17850	16680	2019-01-01	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	Nest-USA	1	153.71	

In [238]:
```
'''Overview of Online Sales:
1.Total number of transactions.
2.Total sales revenue.
3.Distribution of transactions across different product categories.
4.Average transaction value.'''

total_transactions = df_merged_copy2['Transaction_ID'].nunique()
total_sales_revenue = df_merged_copy2['Invoice_Value'].sum()
category_distribution = df_merged_copy2['Product_Category'].value_counts()
average_transaction_value = df_merged_copy2['Invoice_Value'].mean()

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))

# 1. Total number of transactions
sns.countplot(x='Month', data=df_merged_copy2, ax=axes[0, 0])
axes[0, 0].set_title('Total Number of Transactions by Month')
axes[0, 0].set_xlabel('Month')
axes[0, 0].set_ylabel('Number of Transactions')

# 2. Total sales revenue
sns.barplot(x='Month', y='Invoice_Value', data=df_merged_copy2, estimator=sum, ax=axes[0, 1])
axes[0, 1].set_title('Total Sales Revenue by Month')
axes[0, 1].set_xlabel('Month')
axes[0, 1].set_ylabel('Total Sales Revenue')

# 3. Distribution of transactions across different product categories
category_distribution.plot(kind='bar', ax=axes[1, 0])
axes[1, 0].set_title('Distribution of Transactions Across Product Categories')
axes[1, 0].set_xlabel('Product Category')
axes[1, 0].set_ylabel('Number of Transactions')

# 4. Average transaction value
sns.lineplot(x='Month', y='Invoice_Value', data=df_merged_copy2, estimator='mean', ax=axes[1, 1])
axes[1, 1].set_title('Average Transaction Value by Month')
axes[1, 1].set_xlabel('Month')
axes[1, 1].set_ylabel('Average Transaction Value')

plt.tight_layout()

plt.show()
```



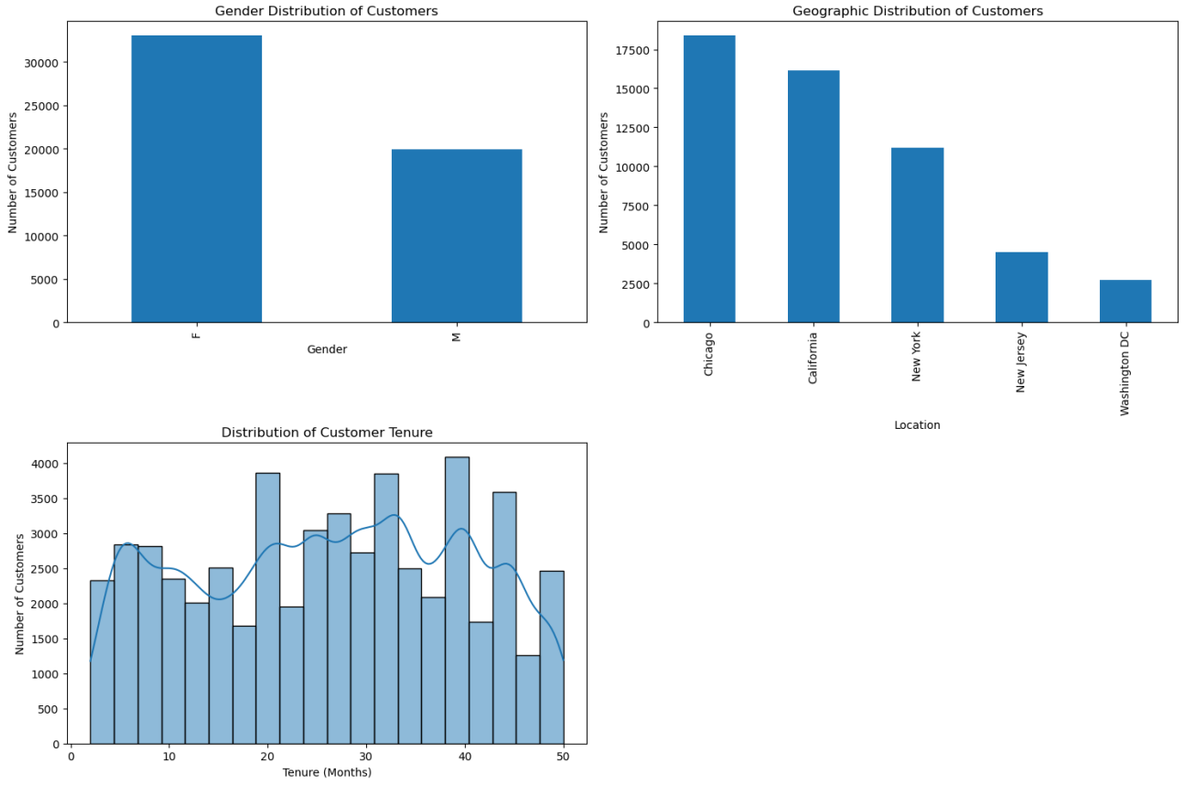
Remarks

1. Total Number of Orders/ Transactions: Transactions are max in the month of Aug (above 6000), followed by July (around 5200), May (around 4300)

2. Total Sales Revenue: Sales is max in the month of Jan & Nov (above 4,50,000), followed by Apr & Oct (around 4,40,000)

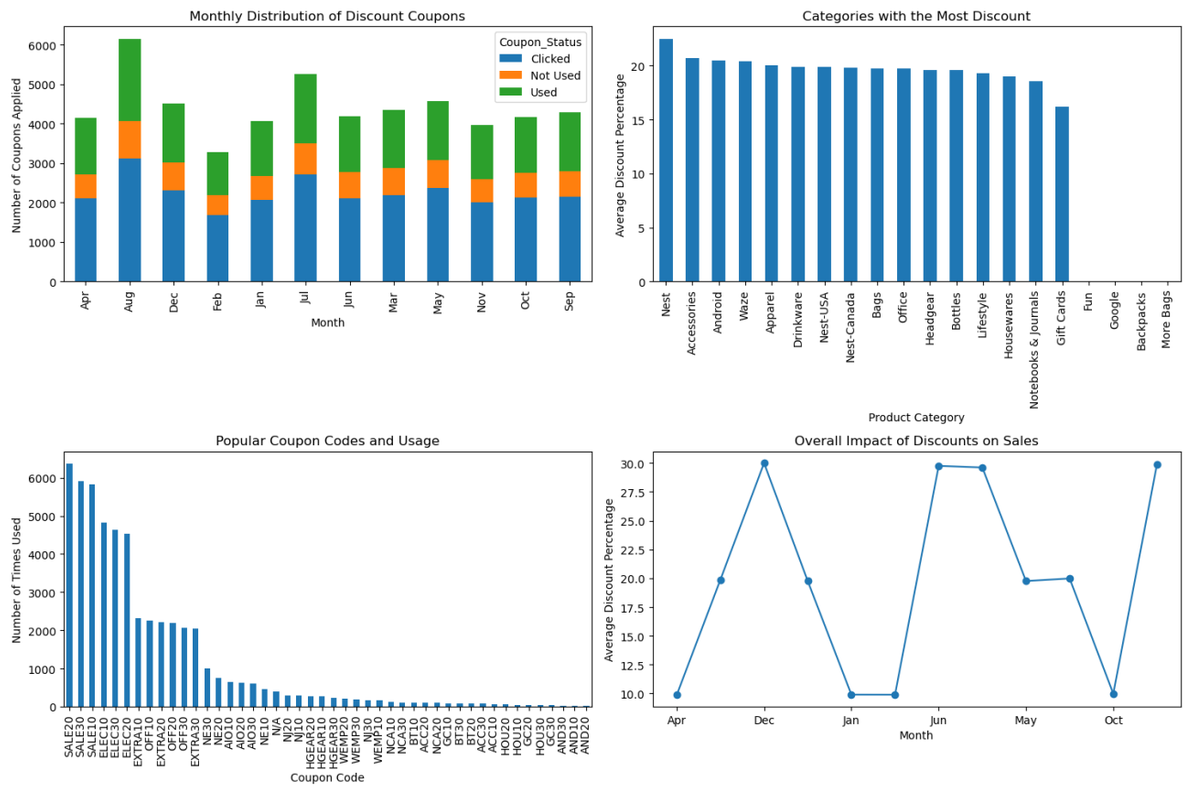
3. Distribution of Orders across Product Category : There are max transactions for Product Category – Apparel (above 17500) followed by Nest-USA(around 13000) & Office (around 6250)
4. Average Order Value/ Transaction Value : Average Transaction value is max in the month of Nov (above 120) followed by Jan (around 115) & Apr (around 109)

In [243]:
```
'''Customer Demographics:
1.Gender distribution of customers.
2.Geographic distribution of customers.
3.Average tenure of customers.'''

# 1. Gender distribution of customers
gender_distribution = df_merged_copy2['Gender'].value_counts()

# 2. Geographic distribution of customers
location_distribution = df_merged_copy2['Location'].value_counts()

# 3. Average tenure of customers
average_tenure = df_merged_copy2['Tenure_Months'].mean()

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))

# 1. Gender distribution of customers
gender_distribution.plot(kind='bar', ax=axes[0, 0])
axes[0, 0].set_title('Gender Distribution of Customers')
axes[0, 0].set_xlabel('Gender')
axes[0, 0].set_ylabel('Number of Customers')

# 2. Geographic distribution of customers
location_distribution.plot(kind='bar', ax=axes[0, 1])
axes[0, 1].set_title('Geographic Distribution of Customers')
axes[0, 1].set_xlabel('Location')
axes[0, 1].set_ylabel('Number of Customers')

# 3. Average tenure of customers
sns.histplot(df_merged_copy2['Tenure_Months'], bins=20, kde=True, ax=axes[1, 0])
axes[1, 0].set_title('Distribution of Customer Tenure')
axes[1, 0].set_xlabel('Tenure (Months)')
axes[1, 0].set_ylabel('Number of Customers')

# Removing the last empty subplot
fig.delaxes(axes[1, 1])

plt.tight_layout()

plt.show()
```



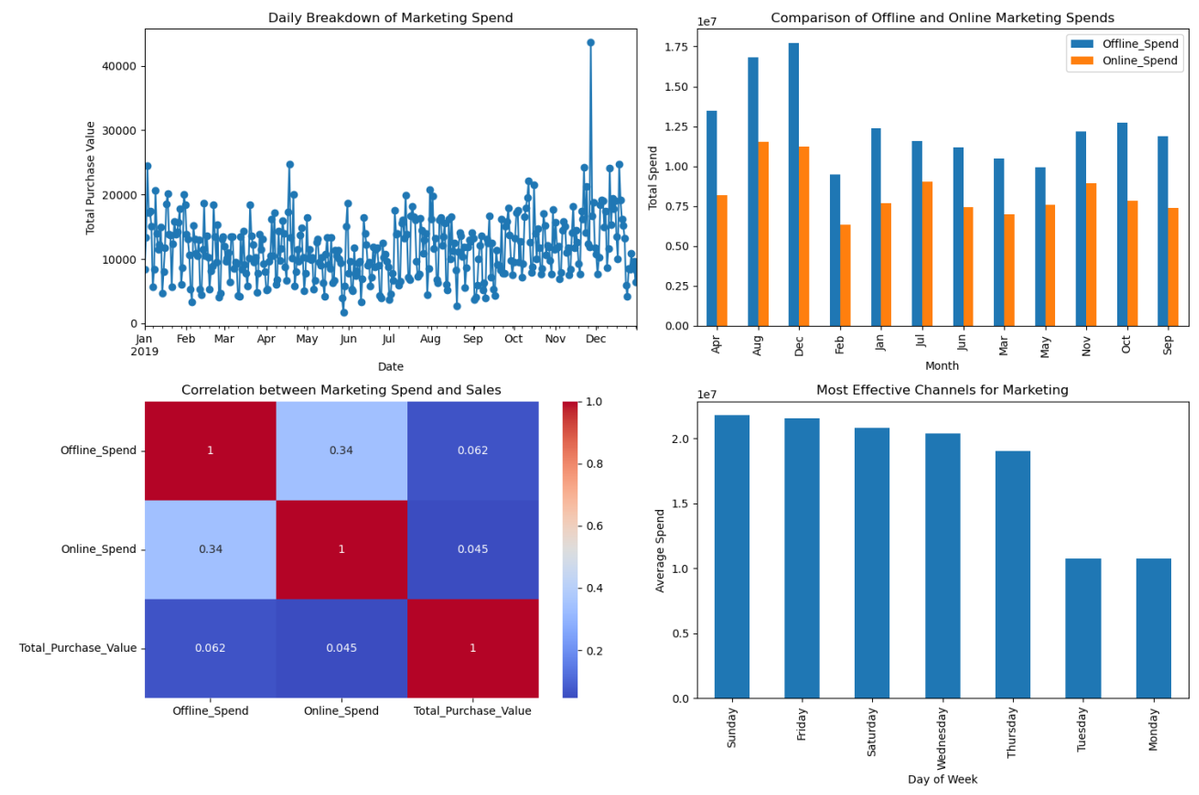
Remarks
1. Gender distribution of customers : Female Customers are more (around 35000) compared to Male Customers (around 20000)
2. Geographic distribution of customers : Max number of customers are from Chicago (around 18000)
    followed by California (around 16500) & New York (around 11500)

In [244]:
```
'''Discount Coupon Analysis:
1.Monthly distribution of discount coupons applied.
2.Categories that received the most discount.
3.Popular coupon codes and their usage.
4.Overall impact of discounts on sales.'''

# 1. Monthly distribution of discount coupons applied
monthly_coupon_distribution = df_merged_copy2.groupby('Month')['Coupon_Status'].value_counts().unstack().fillna(0)

# 2. Categories that received the most discount
category_discount = df_merged_copy2.groupby('Product_Category')['Discount_pct'].mean().sort_values(ascending=False

# 3. Popular coupon codes and their usage
popular_coupons = df_merged_copy2['Coupon_Code'].value_counts()

# 4. Overall impact of discounts on sales
discount_impact = df_merged_copy2.groupby('Month')['Discount_pct'].mean()

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))

# 1. Monthly distribution of discount coupons applied
monthly_coupon_distribution.plot(kind='bar', stacked=True, ax=axes[0, 0])
axes[0, 0].set_title('Monthly Distribution of Discount Coupons')
axes[0, 0].set_xlabel('Month')
axes[0, 0].set_ylabel('Number of Coupons Applied')

# 2. Categories that received the most discount
category_discount.plot(kind='bar', ax=axes[0, 1])
axes[0, 1].set_title('Categories with the Most Discount')
axes[0, 1].set_xlabel('Product Category')
axes[0, 1].set_ylabel('Average Discount Percentage')

# 3. Popular coupon codes and their usage
popular_coupons.plot(kind='bar', ax=axes[1, 0])
axes[1, 0].set_title('Popular Coupon Codes and Usage')
axes[1, 0].set_xlabel('Coupon Code')
axes[1, 0].set_ylabel('Number of Times Used')

# 4. Overall impact of discounts on sales
discount_impact.plot(kind='line', marker='o', ax=axes[1, 1])
axes[1, 1].set_title('Overall Impact of Discounts on Sales')
axes[1, 1].set_xlabel('Month')
axes[1, 1].set_ylabel('Average Discount Percentage')

plt.tight_layout()
plt.show()
```




```
#### Remarks
1. Monthly distribution of discount coupons applied : There is maximum Discount coupons used in the month of Aug,
    followed by Jul & May
2. Categories that received the most discount : Products under Category Nest offered the most discounts
    followed by Accessories & Android
3. Popular coupon codes and their usage : Coupon code SALE20 (above 6300) was used the max followed by
    SALE30 (around 5800) & SALE10(around 5700)
4. Overall impact of discounts on sales: Peak is noticed in the month of Dec
```

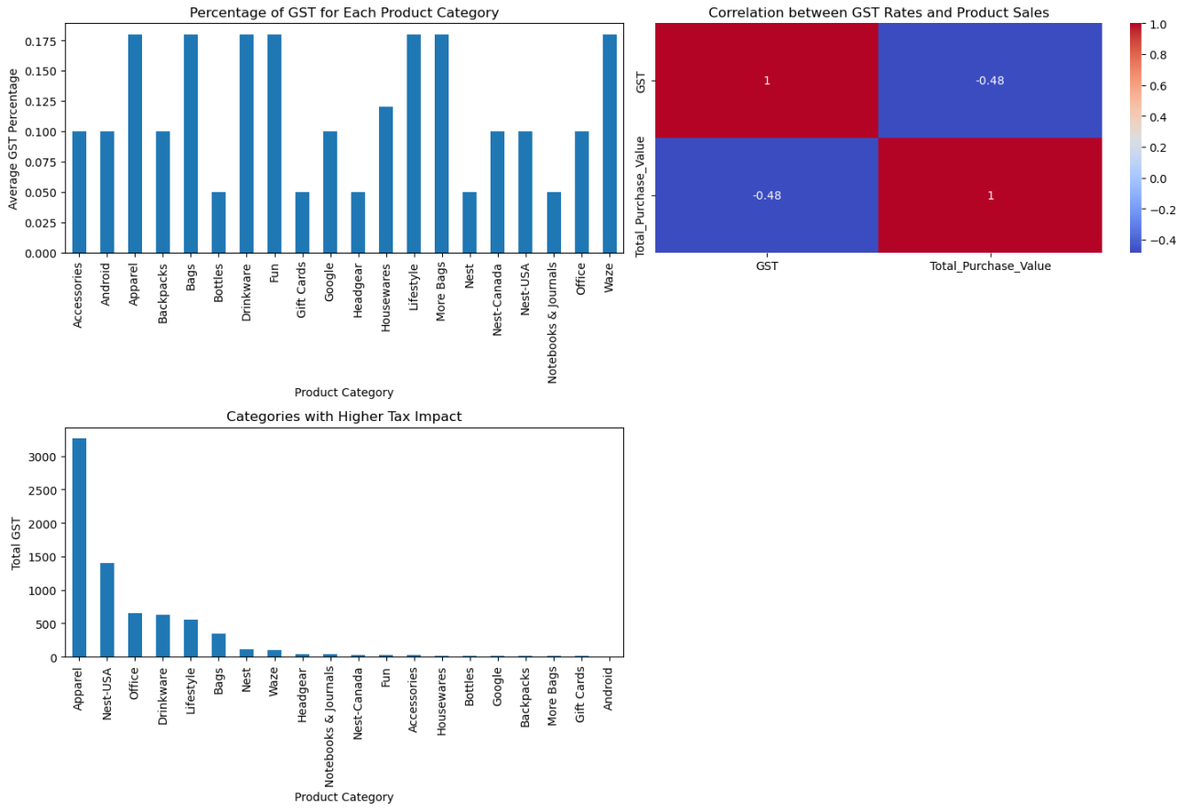
In [245]:
```
'''Marketing Spend:
1.Daily breakdown of marketing spend.
2.Comparison of offline and online marketing spends.
3.Correlation between marketing spend and sales.
4.Identifying the most effective channels for marketing.'''

# 1. Daily breakdown of marketing spend
daily_marketing_spend = df_merged_copy2.groupby('Marketing_Spend_Date')['Total_Purchase_Value'].sum()

# 2. Comparison of offline and online marketing spends
offline_online_comparison = df_merged_copy2.groupby('Month')[['Offline_Spend', 'Online_Spend']].sum()

# 3. Correlation between marketing spend and sales
correlation_spend_sales = df_merged_copy2[['Offline_Spend', 'Online_Spend', 'Total_Purchase_Value']].corr()

# 4. Identifying the most effective channels for marketing
channels_effectiveness = df_merged_copy2.groupby('day_name')[['Offline_Spend', 'Online_Spend']].sum().mean(axis=1)

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))

# 1. Daily breakdown of marketing spend
daily_marketing_spend.plot(kind='line', marker='o', ax=axes[0, 0])
axes[0, 0].set_title('Daily Breakdown of Marketing Spend')
axes[0, 0].set_xlabel('Date')
axes[0, 0].set_ylabel('Total Purchase Value')

# 2. Comparison of offline and online marketing spends
offline_online_comparison.plot(kind='bar', ax=axes[0, 1])
axes[0, 1].set_title('Comparison of Offline and Online Marketing Spends')
axes[0, 1].set_xlabel('Month')
axes[0, 1].set_ylabel('Total Spend')

# 3. Correlation between marketing spend and sales
sns.heatmap(correlation_spend_sales, annot=True, cmap='coolwarm', ax=axes[1, 0])
axes[1, 0].set_title('Correlation between Marketing Spend and Sales')

# 4. Identifying the most effective channels for marketing
channels_effectiveness.plot(kind='bar', ax=axes[1, 1])
axes[1, 1].set_title('Most Effective Channels for Marketing')
axes[1, 1].set_xlabel('Day of Week')
axes[1, 1].set_ylabel('Average Spend')

plt.tight_layout()

plt.show()
```



Remarks

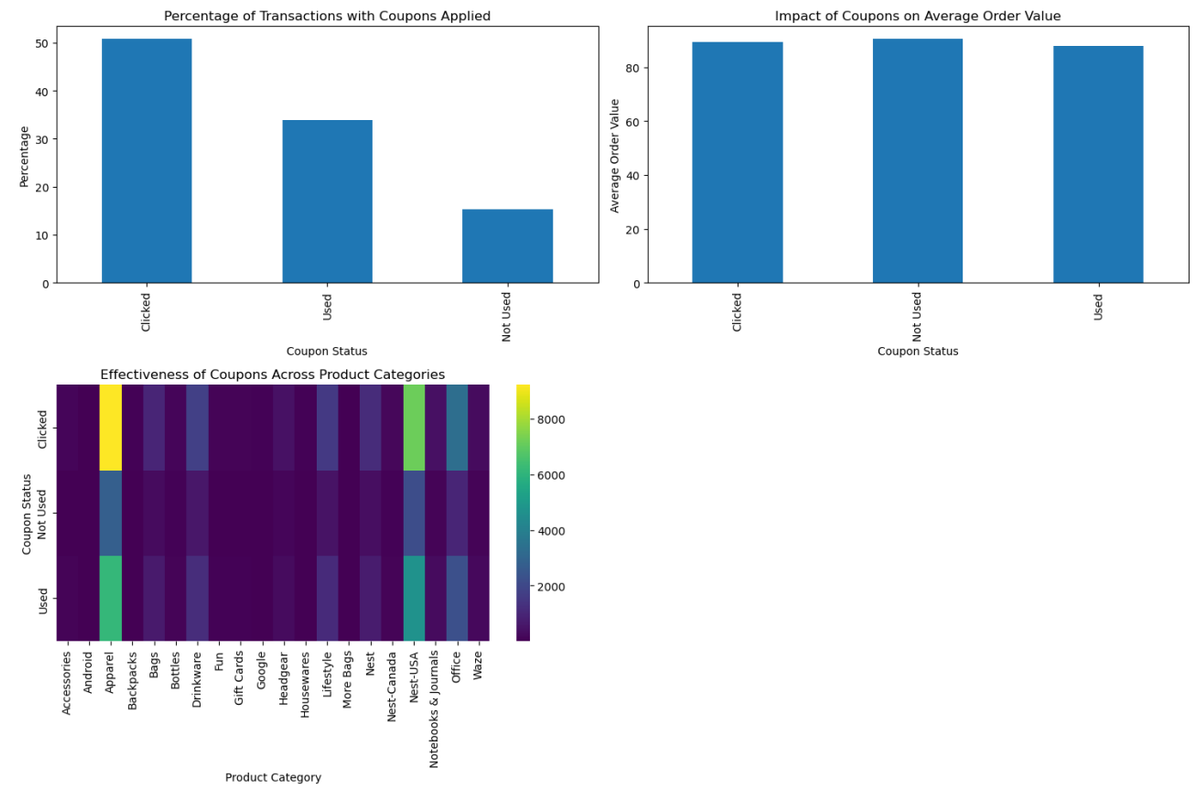
Daily breakdown of marketing spend : Total Purchase Value is noticed to peak in the month around Dec Comparison of offline and online marketing spends Offline spend is max in the month of Dec followed by Aug & Apr Online spend is max in the month of Aug, followed by Dec & Jul Correlation between marketing spend and sales. Total Purchase Value is more related with Offline_Spend as compared to Online Spend Average Marketing Spend : Is more on Sundays followed by Friday & Saturday.

In [246]:
```
'''Tax Impact on Sales:
1.Percentage of GST for each product category.
2.Correlation between GST rates and product sales.
3.Identifying categories with higher tax impact.'''

# 1. Percentage of GST for each product category
gst_percentage_category = df_merged_copy2.groupby('Product_Category')['GST'].mean()

# 2. Correlation between GST rates and product sales
correlation_gst_sales = df_merged_copy2[['GST', 'Total_Purchase_Value']].corr()

# 3. Identifying categories with higher tax impact
tax_impact_categories = df_merged_copy2.groupby('Product_Category')['GST'].sum().sort_values(ascending=False)

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))

# 1. Percentage of GST for each product category
gst_percentage_category.plot(kind='bar', ax=axes[0, 0])
axes[0, 0].set_title('Percentage of GST for Each Product Category')
axes[0, 0].set_xlabel('Product Category')
axes[0, 0].set_ylabel('Average GST Percentage')

# 2. Correlation between GST rates and product sales
sns.heatmap(correlation_gst_sales, annot=True, cmap='coolwarm', ax=axes[0, 1])
axes[0, 1].set_title('Correlation between GST Rates and Product Sales')

# 3. Identifying categories with higher tax impact
tax_impact_categories.plot(kind='bar', ax=axes[1, 0])
axes[1, 0].set_title('Categories with Higher Tax Impact')
axes[1, 0].set_xlabel('Product Category')
axes[1, 0].set_ylabel('Total GST')

fig.delaxes(axes[1, 1])

plt.tight_layout()

plt.show()
```



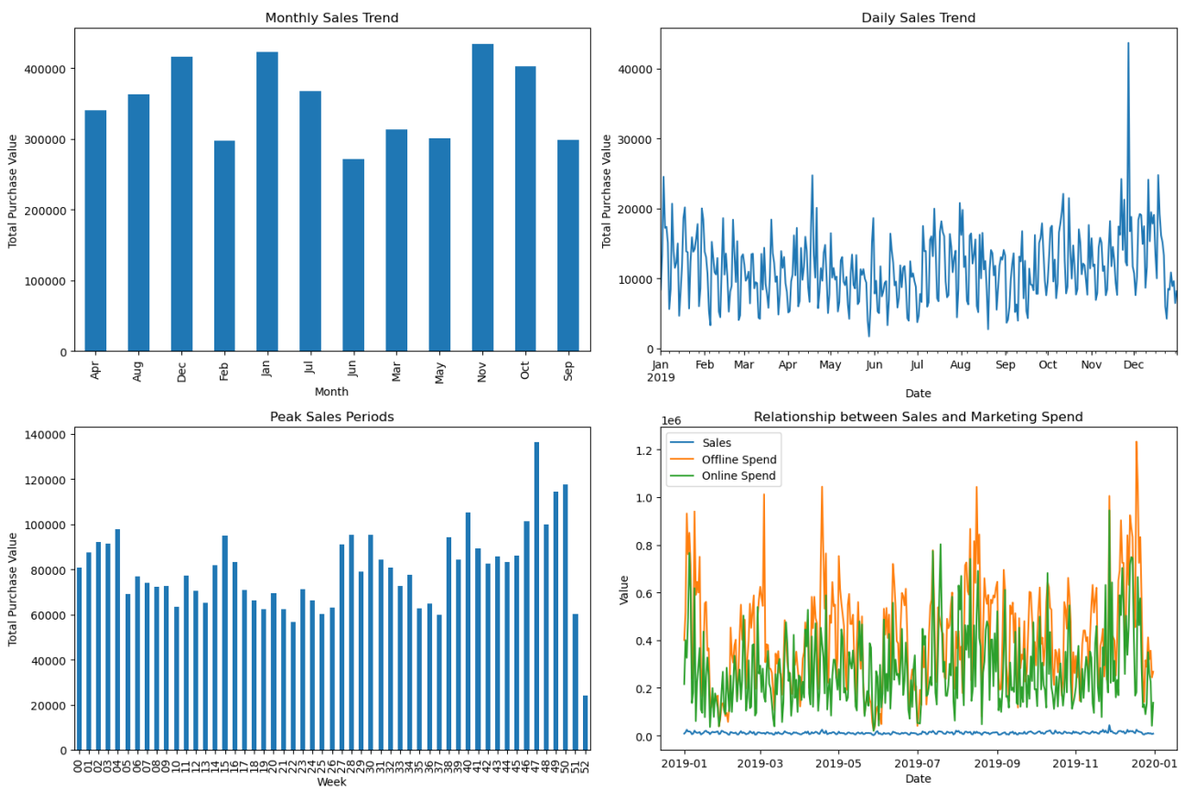

```
### Remarks

Percentage of GST for each product category. GST is almost the same for categories Apparel, Bags, Drinkware, Fun,
Lifestyle, More Bags. GST is least for categories like Bottles, Gift Cards, Headgear, Nest & Notebook & Journals
Correlation between GST rates and product sales : Negative correlation Identifying categories with higher tax
impact :
        Apparel as a category has max GST followed by Nest-USA & Office
```

In [247]:
```
'''Coupon Redemption Analysis:
1.Percentage of transactions where coupons were applied.
2.Impact of coupons on average order value.
3.Effectiveness of coupons across different product categories.'''

# 1. Percentage of transactions where coupons were applied
coupon_redemption_percentage = (df_merged_copy2['Coupon_Status'].value_counts() / df_merged_copy2.shape[0]) * 100

# 2. Impact of coupons on average order value
coupon_impact_avg_order = df_merged_copy2.groupby('Coupon_Status')['Invoice_Value'].mean()

# 3. Effectiveness of coupons across different product categories
coupon_effectiveness_category = df_merged_copy2.groupby(['Coupon_Status', 'Product_Category']).size().unstack(fill_

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))

# 1. Percentage of transactions where coupons were applied
coupon_redemption_percentage.plot(kind='bar', ax=axes[0, 0])
axes[0, 0].set_title('Percentage of Transactions with Coupons Applied')
axes[0, 0].set_xlabel('Coupon Status')
axes[0, 0].set_ylabel('Percentage')

# 2. Impact of coupons on average order value
coupon_impact_avg_order.plot(kind='bar', ax=axes[0, 1])
axes[0, 1].set_title('Impact of Coupons on Average Order Value')
axes[0, 1].set_xlabel('Coupon Status')
axes[0, 1].set_ylabel('Average Order Value')

# 3. Effectiveness of coupons across different product categories
sns.heatmap(coupon_effectiveness_category, cmap='viridis', ax=axes[1, 0])
axes[1, 0].set_title('Effectiveness of Coupons Across Product Categories')
axes[1, 0].set_xlabel('Product Category')
axes[1, 0].set_ylabel('Coupon Status')

fig.delaxes(axes[1, 1])

plt.tight_layout()

plt.show()
```



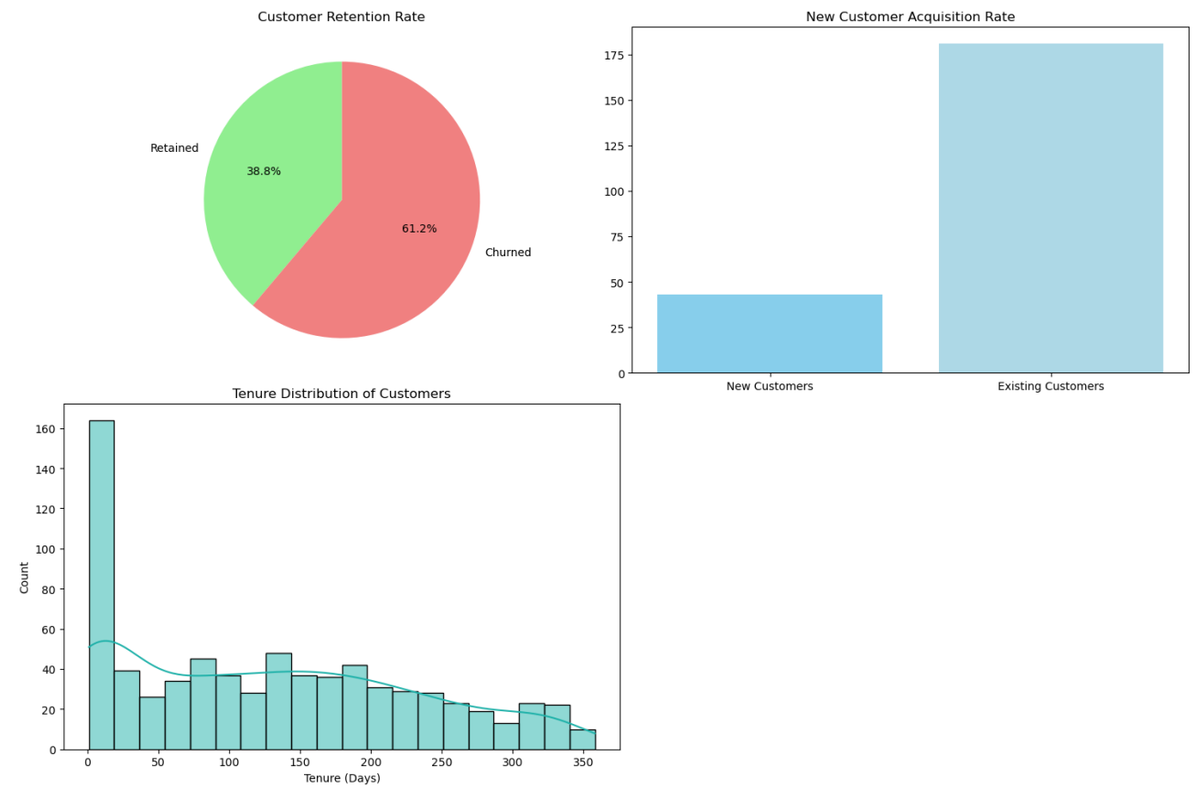

```
### remarks
Percentage of transactions where coupons were applied : above 30% Impact of coupons on average order value :
above 80
Effectiveness of coupons across different product categories : Most for Apparel & Nest-USA Sales Trend Over Time:
        Monthly and daily trends in sales. Max in the month of Nov followed by Jan & Dec
Identifying peak sales periods : in 47th week Relationship between sales and external factors (e.g., marketing
spend, discounts) :
        Offline spends is more compared to Online spend – for the sales trend shown
```

In [248]:
```
'''Sales Trend Over Time:
1.Monthly and daily trends in sales.
2.Identifying peak sales periods.
3.Relationship between sales and external factors (e.g., marketing spend, discounts).'''

# 1. Monthly and daily trends in sales
monthly_sales_trend = df_merged_copy2.groupby('Month')['Total_Purchase_Value'].sum()
daily_sales_trend = df_merged_copy2.groupby('Transaction_Date')['Total_Purchase_Value'].sum()

# 2. Identifying peak sales periods
peak_sales_periods = df_merged_copy2.groupby('week_name')['Total_Purchase_Value'].sum()

# 3. Relationship between sales and external factors (e.g., marketing spend, discounts)
sales_vs_marketing_spend = df_merged_copy2.groupby('Marketing_Spend_Date')[['Total_Purchase_Value', 'Offline_Spend
sales_vs_discount = df_merged_copy2.groupby('Marketing_Spend_Date')[['Total_Purchase_Value', 'Discount_pct']].mean

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))

# 1. Monthly and daily trends in sales
monthly_sales_trend.plot(kind='bar', ax=axes[0, 0])
axes[0, 0].set_title('Monthly Sales Trend')
axes[0, 0].set_xlabel('Month')
axes[0, 0].set_ylabel('Total Purchase Value')

daily_sales_trend.plot(kind='line', ax=axes[0, 1])
axes[0, 1].set_title('Daily Sales Trend')
axes[0, 1].set_xlabel('Date')
axes[0, 1].set_ylabel('Total Purchase Value')

# 2. Identifying peak sales periods
peak_sales_periods.plot(kind='bar', ax=axes[1, 0])
axes[1, 0].set_title('Peak Sales Periods')
axes[1, 0].set_xlabel('Week')
axes[1, 0].set_ylabel('Total Purchase Value')

# 3. Relationship between sales and external factors
sns.lineplot(x='Marketing_Spend_Date', y='Total_Purchase_Value', data=sales_vs_marketing_spend, ax=axes[1, 1], lab
sns.lineplot(x='Marketing_Spend_Date', y='Offline_Spend', data=sales_vs_marketing_spend, ax=axes[1, 1], label='Off
sns.lineplot(x='Marketing_Spend_Date', y='Online_Spend', data=sales_vs_marketing_spend, ax=axes[1, 1], label='Onli

axes[1, 1].set_title('Relationship between Sales and Marketing Spend')
axes[1, 1].set_xlabel('Date')
axes[1, 1].set_ylabel('Value')

plt.tight_layout()
plt.show()
```




```
## Reamrka
Customer retention rate : 38.8%
New customer acquisition rate : around 40
Tenure distribution of customers : Max count for the 10-12 days followed by 130-140 days
```

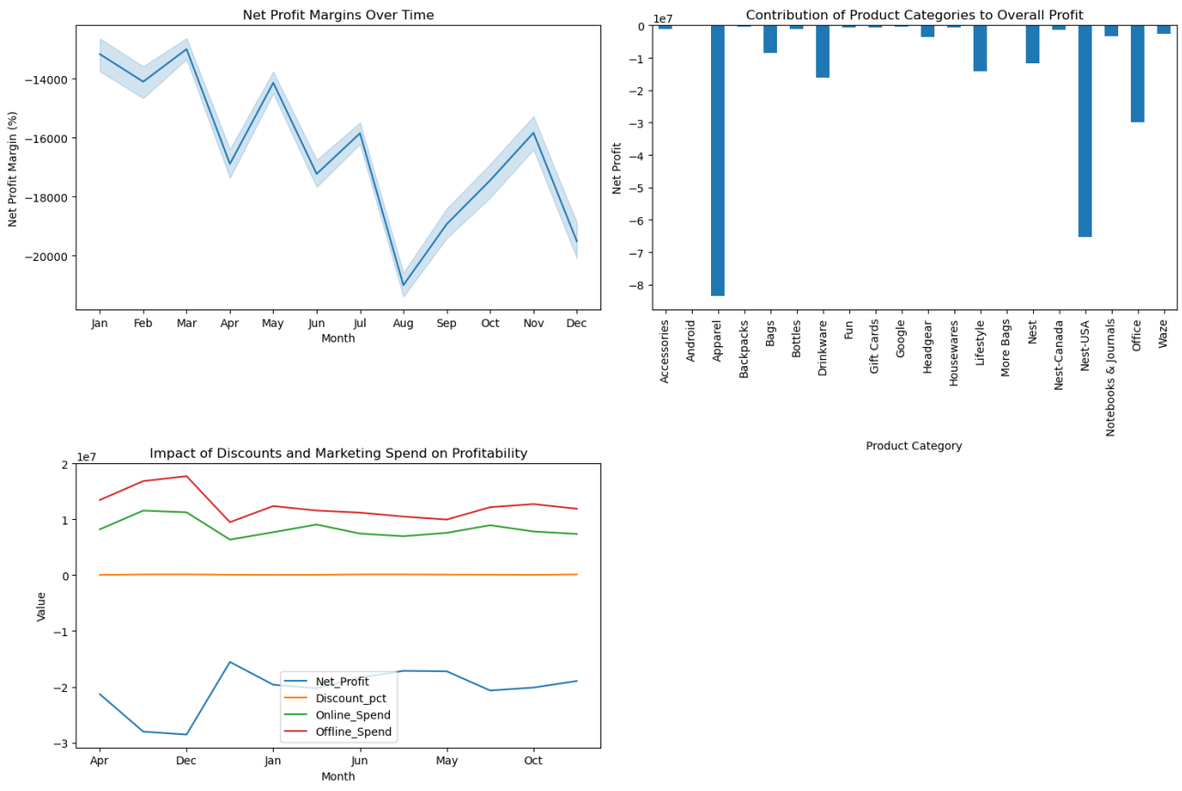
In [260]:
```
'''Customer Retention and Acquisition:
1.Customer retention rate.
2.New customer acquisition rate.
3.Tenure distribution of customers.'''

# Find the latest date in the dataset
latest_date = df_merged_copy2['Transaction_Date'].max()

# Identify new customers as those who made a transaction in the last 15 days
# and have no transactions before that period
new_customers = df_merged_copy2[(df_merged_copy2['Transaction_Date'] >= latest_date - pd.DateOffset(days=15)) &
                                ~df_merged_copy2['CustomerID'].isin(df_merged_copy2[df_merged_copy2['Transaction_Da

# Count the number of new customers
total_new_customers = new_customers['CustomerID'].nunique()

# Count the total number of customers at the start
total_customers_at_start = df_merged_copy2[df_merged_copy2['Month'] == df_merged_copy2['Month'].min()]['CustomerID

# Calculate the New Customer Acquisition Rate
customer_acquisition_rate = (total_new_customers / total_customers_at_start) * 100

# 1. Customer retention rate
retention_rate = df_merged_copy2.groupby('CustomerID')['Transaction_Date'].agg(['min', 'max'])
retention_rate['Tenure'] = (retention_rate['max'] - retention_rate['min']).dt.days
total_customers = retention_rate.shape[0]
retained_customers = retention_rate[~retention_rate.index.isin(churned_customers)]['Tenure'].count()
customer_retention_rate = (retained_customers / total_customers) * 100

# 2. New customer acquisition rate [Last 15 days in dec month made 1st transac]
# Adjusted calculation based on the new criteria
new_customers = df_merged_copy2[(df_merged_copy2['Transaction_Date'] >= latest_date - pd.DateOffset(days=15)) &
                                ~df_merged_copy2['CustomerID'].isin(df_merged_copy2[df_merged_copy2['Transaction_Da
total_new_customers = new_customers['CustomerID'].nunique()
customer_acquisition_rate = (total_new_customers / total_customers_at_start) * 100

# 3. Tenure distribution of customers
tenure_distribution = retention_rate[retention_rate['Tenure'] > 0]['Tenure']

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))

# 1. Customer retention rate
axes[0, 0].pie([retained_customers, total_customers - retained_customers], labels=['Retained', 'Churned'], autopct
axes[0, 0].set_title('Customer Retention Rate')

# 2. New customer acquisition rate
axes[0, 1].bar(['New Customers', 'Existing Customers'], [total_new_customers, total_customers_at_start - total_new
axes[0, 1].set_title('New Customer Acquisition Rate')

# 3. Tenure distribution of customers
sns.histplot(tenure_distribution, bins=20, kde=True, color='lightseagreen', ax=axes[1, 0])
axes[1, 0].set_title('Tenure Distribution of Customers')
axes[1, 0].set_xlabel('Tenure (Days)')
fig.delaxes(axes[1, 1])


plt.tight_layout()

plt.show()
```



Remrks

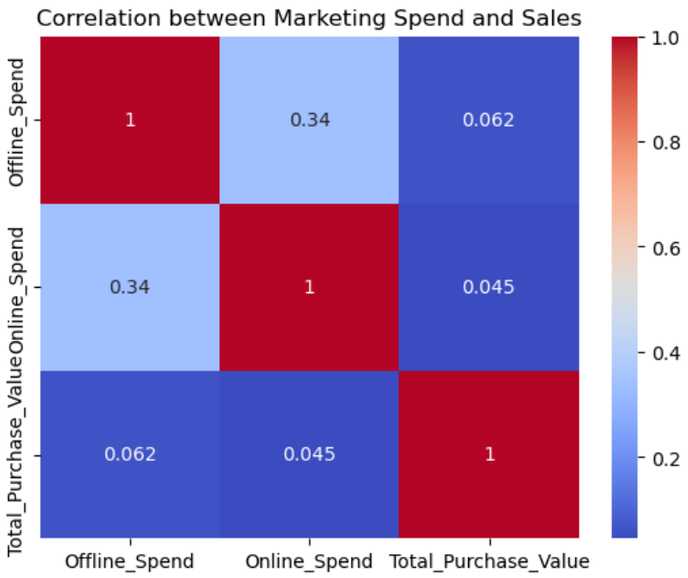
Net profit margins : is decreasing over time with peaks in Mar, followed by May, Jul & Nov
Contribution of different product categories to overall profit : Least in category Apparel followed by Nest-USA & Office;
Most in category More Bags, Backpacks, Fun, Google, Android, Gift Cards, Housewares
Impact of discounts and marketing spend on profitability : Profitability is noticed to peak when when discount_pct is maintained at zero, Offline spend is around 1 & online spend around 0.8

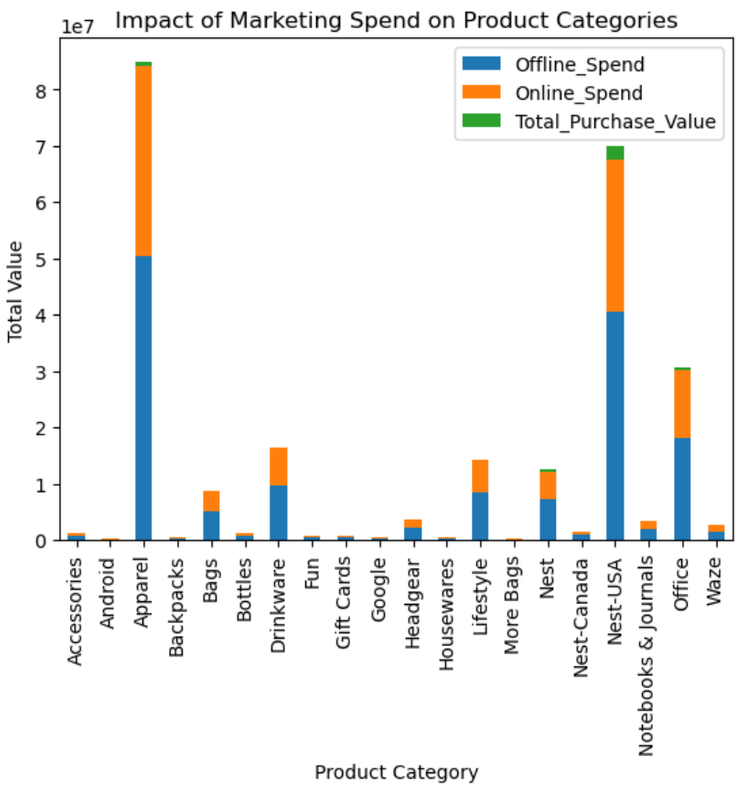
In [255]:
```
'''Profitability Analysis:
1.Net profit margins.
2.Contribution of different product categories to overall profit.
3.Impact of discounts and marketing spend on profitability.'''

# 1. Net profit margins
df_merged_copy2['Net_Profit'] = df_merged_copy2['Total_Purchase_Value'] - df_merged_copy2['Online_Spend'] - df_mer
df_merged_copy2['Net_Profit_Margin'] = (df_merged_copy2['Net_Profit'] / df_merged_copy2['Total_Purchase_Value']) *

# 2. Contribution of different product categories to overall profit
profit_contribution_category = df_merged_copy2.groupby('Product_Category')['Net_Profit'].sum()

# 3. Impact of discounts and marketing spend on profitability
profitability_discount_marketing = df_merged_copy2.groupby('Month')[['Net_Profit', 'Discount_pct', 'Online_Spend',

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))

# 1. Net profit margins
sns.lineplot(x='Month', y='Net_Profit_Margin', data=df_merged_copy2, ax=axes[0, 0])
axes[0, 0].set_title('Net Profit Margins Over Time')
axes[0, 0].set_xlabel('Month')
axes[0, 0].set_ylabel('Net Profit Margin (%)')

# 2. Contribution of different product categories to overall profit
profit_contribution_category.plot(kind='bar', ax=axes[0, 1])
axes[0, 1].set_title('Contribution of Product Categories to Overall Profit')
axes[0, 1].set_xlabel('Product Category')
axes[0, 1].set_ylabel('Net Profit')

# 3. Impact of discounts and marketing spend on profitability
profitability_discount_marketing.plot(kind='line', ax=axes[1, 0])
axes[1, 0].set_title('Impact of Discounts and Marketing Spend on Profitability')
axes[1, 0].set_xlabel('Month')
axes[1, 0].set_ylabel('Value')

fig.delaxes(axes[1, 1])

plt.tight_layout()
plt.show()
```



Another Graph shows Offline Marketing spend more for categories Apparel followed by Nest-USA & Office;
Same is true for Online spend ;

In [256]:
```
'''
1.Suggestions for optimizing marketing spend.
2.Recommendations for improving customer retention.
3.Strategies for maximizing the impact of discount coupons.'''

# Explore the correlation between marketing spend and sales
correlation_spend_sales = df_merged_copy2[['Offline_Spend', 'Online_Spend', 'Total_Purchase_Value']].corr()

# Visualizing the correlation
sns.heatmap(correlation_spend_sales, annot=True, cmap='coolwarm')
plt.title('Correlation between Marketing Spend and Sales')
plt.show()

# Exploring the impact of marketing spend on a specific product category
category_spend_sales = df_merged_copy2.groupby('Product_Category')[['Offline_Spend', 'Online_Spend', 'Total_Purcha

# Visualizing the impact
category_spend_sales.plot(kind='bar', stacked=True)
plt.title('Impact of Marketing Spend on Product Categories')
plt.xlabel('Product Category')
plt.ylabel('Total Value')
plt.show()
```





Recommendations

```
1.Strategic Discounting:
Leveraging the success of coupon code SALE20 by continuing to offer targeted discounts on popular categories like
Apparel
and Nest-USA.
Exploring creating exclusive discounts for high-impact months such as August and July.

2.Optimized Marketing Spend:
Focusing on increasing marketing efforts during peak sales periods, particularly in August and December, to
capitalize on higher
```

consumer engagement.
Exploring diversifying marketing channels and allocation of resources based on their historical effectiveness.

3.Targeted Geographic Marketing:
Chicago has the highest customer base, considering targeted marketing campaigns in this region to further boost sales.
Exploring opportunities to increase brand visibility in California and New York.

4.Enhance Customer Retention:
Developing targeted retention strategies for customers with a tenure of 10-12 days and 130-140 days to strengthen loyalty.
Introducing loyalty programs or special offers for returning customers.

5.Product Category Optimization:
Focusing on promoting and expanding product lines in high-performing categories like Apparel, Nest-USA, and Office.
Considering bundling or cross-selling strategies to boost sales in less-performing categories.

6.Fine-tune GST Strategy:
Evaluating the impact of GST rates on sales and consider adjusting pricing or promotional strategies for categories
like Apparel with higher GST.
Exploring lobbying for reduced GST rates in categories with lower sales and potential for growth.

7.Maximize Online Marketing Impact:
Given the peak in online spend in August, tailoring online marketing campaigns during this period to maximize visibility
and customer engagement.
Optimize online marketing spend for high-impact categories identified in offline marketing.

8.Customer Education and Engagement:
Educating customers about the benefits of discounts and how they contribute to increased average order value.
Encouraging customers to explore a wider range of product categories through targeted marketing.

9.Dynamic Pricing Strategies:
Implementing dynamic pricing strategies based on historical data, considering peak sales periods and customer behavior.
Adjusting pricing to maintain competitiveness while maximizing profitability.

10.Investment in Profitable Categories:
Allocating resources towards promoting and expanding product lines in categories contributing significantly to overall profit, such as More Bags, Backpacks, Fun, Google, Android, and Gift Cards.
Evaluating the potential for introducing new products within these high-profit categories.