

Ecommerce_Marketing_Insights_by_Diptyajit_Das

June 15, 2024

1 Marketing Insights for E-Commerce Company

1.1 Problem Statement:

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

1.2 Expectations:

Through this project, we expect to leverage a range of data analysis techniques to uncover actionable insights that propel the e-commerce company towards significant customer retention and revenue growth.

Key Metrics and Objectives:

1. Identifying Key Customer Segments and Behaviors:

- Utilize descriptive statistics and segmentation techniques to understand what drives customer acquisition and churn.

2. Evaluating Marketing Campaign Effectiveness:

- Employ hypothesis testing to assess the impact of online and offline marketing efforts on customer behavior and revenue.

3. Optimizing Discount Strategies:

- Analyze the influence of discounts and promotions on revenue and customer engagement to identify optimal pricing strategies.

4. Predicting Customer Lifetime Value:

- Implement data-driven models to anticipate future customer value and prioritize retention efforts.

5. Unveiling Cross-Selling Opportunities:

- Perform market basket analysis to discover frequently co-purchased products and inform product placement strategies.

6. Formulating Data-Driven Recommendations:

- Present clear and compelling visualizations and reports that translate insights into actionable marketing strategies for maximizing customer retention and revenue growth.

1.3 Dataset Description

Transaction data has been provided from 1st Jan 2019 to 31st Dec 2019.

1.3.1 Datasets:

1. Online_Sales.csv

- **CustomerID:** Customer unique ID
- **Transaction_ID:** Transaction Unique ID
- **Transaction_Date:** Date of Transaction
- **Product_SKU:** SKU ID – Unique Id for product
- **Product_Description:** Product Description
- **Product_Category:** Product Category
- **Quantity:** Number of items ordered
- **Avg_Price:** Price per one quantity
- **Delivery_Charges:** Charges for delivery
- **Coupon_Status:** Any discount coupon applied

2. Customers_Data.csv

- **CustomerID:** Customer Unique ID
- **Gender:** Gender of customer
- **Location:** Location of Customer
- **Tenure_Months:** Tenure in Months

3. Discount_Coupon.csv

- **Month:** Discount coupon applied in that month
- **Product_Category:** Product category
- **Coupon_Code:** Coupon Code for given Category and given month
- **Discount_pct:** Discount Percentage for given coupon

4. Marketing_Spend.csv

- **Date:** Date
- **Offline_Spend:** Marketing spend on offline channels like TV, Radio, Newspapers, hoardings etc.
- **Online_Spend:** Marketing spend on online channels like Google keywords, Facebook etc.

5. Tax_Amount.csv

- **Product_Category:** Product Category
- **GST:** Percentage of GST

```
[1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import ttest_ind, spearmanr, chi2_contingency, levene, shapiro
#!pip install pingouin
import pingouin as pg

#!pip install mlxtend
from mlxtend.frequent_patterns import apriori, association_rules
from operator import attrgetter
```

```

import pickle
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from category_encoders import TargetEncoder
from sklearn.metrics import silhouette_score, mean_squared_error as MSE
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.ensemble import StackingRegressor

import warnings
warnings.simplefilter('ignore')

```

```

[2]: dfs=pd.read_csv('data/Online_Sales.csv')
dfc=pd.read_csv('data/Customers.csv')
dfd=pd.read_csv('data/Discount_Coupon.csv')
dfm=pd.read_csv('data/Marketing_Spend.csv')
dft=pd.read_csv('data/Tax_amount.csv')

```

```

[3]: names=['sales', 'customers', 'discounts', 'marketing', 'taxes']
df_s=[dfs, dfc, dfd, dfm, dft]
for i in range(5):
    print(f'Shape of {names[i]} dataframe : ')
    print(df_s[i].shape)

    print()

    print(f'Number of missing values in {names[i]} dataframe : ')
    print(df_s[i].isna().sum().sum())
    print()

```

Shape of sales dataframe :
(52924, 10)

Number of missing values in sales dataframe :
0

Shape of customers dataframe :
(1468, 4)

Number of missing values in customers dataframe :
0

Shape of discounts dataframe :
(204, 4)

Number of missing values in discounts dataframe :
0

```
Shape of marketing dataframe :  
(365, 3)
```

```
Number of missing values in marketing dataframe :  
0
```

```
Shape of taxes dataframe :  
(20, 2)
```

```
Number of missing values in taxes dataframe :  
0
```

1.4 All datasets have no null values and the following shapes: sales (shape: 52924, 10), customers (shape: 1468, 4), discounts (shape: 204, 4), marketing (shape: 365, 3), and taxes (shape: 20, 2).

1.5 Preprocessing and Cleaning

1.5.1 Merging with taxes dataframe on Product_Category.

```
[4]: df=dfs.merge(dft,on='Product_Category',how='left')  
df.CustomerID=df.CustomerID.astype('object')  
df.Transaction_ID=df.Transaction_ID.astype('object')  
df.dtypes
```

```
[4]: CustomerID          object  
Transaction_ID         object  
Transaction_Date        object  
Product_SKU            object  
Product_Description     object  
Product_Category       object  
Quantity               int64  
Avg_Price               float64  
Delivery_Charges        float64  
Coupon_Status          object  
GST                    object  
dtype: object
```

1.5.2 Converting Transaction_Date to datetime and extracting month.

```
[5]: df['Transaction_Date'] = pd.to_datetime(df['Transaction_Date'])  
df['Month'] = df['Transaction_Date'].dt.strftime('%b')
```

1.5.3 Merging with discounts dataframe on Month and Product_Category.

1.5.4 Applying coupon if Coupon_Status is 'Used'.

```
[6]: df=df.merge(dfd,on=['Month','Product_Category'],how='left')
df['Coupon']=df['Coupon_Status'].apply(lambda x:1 if x=='Used' else 0)
df=df.rename(columns={'Coupon_Code_x':'Coupon_Code','Discount_pct_x':
↳'Discount_pct'}).drop(columns=['Coupon_Status'])
```

1.5.5 Converting GST to integer and calculating total Invoice Value.

```
[7]: df['GST']=df['GST'].str.replace('%','').astype(int)
df['Invoice']=(df['Quantity']*df['Avg_Price'])*(df['Coupon']*(1-df['Discount_pct'])/
↳100))*(1+df['GST']/100)+df['Delivery_Charges']
df.head()
```

```
[7]: CustomerID Transaction_ID Transaction_Date Product_SKU \
0 17850 16679 2019-01-01 GGOENEBJ079499
1 17850 16680 2019-01-01 GGOENEBJ079499
2 17850 16681 2019-01-01 GGOEGFKQ020399
3 17850 16682 2019-01-01 GGOEGAAB010516
4 17850 16682 2019-01-01 GGOEGBJL013999
```

```
Product_Description Product_Category \
0 Nest Learning Thermostat 3rd Gen-USA - Stainle... Nest-USA
1 Nest Learning Thermostat 3rd Gen-USA - Stainle... Nest-USA
2 Google Laptop and Cell Phone Stickers Office
3 Google Men's 100% Cotton Short Sleeve Hero Tee... Apparel
4 Google Canvas Tote Natural/Navy Bags
```

```
Quantity Avg_Price Delivery_Charges GST Month Coupon_Code Discount_pct \
0 1 153.71 6.5 10 Jan ELEC10 10.0
1 1 153.71 6.5 10 Jan ELEC10 10.0
2 1 2.05 6.5 10 Jan OFF10 10.0
3 5 17.53 6.5 18 Jan SALE10 10.0
4 1 16.50 6.5 18 Jan AI010 10.0
```

```
Coupon Invoice
0 1 158.6729
1 1 158.6729
2 1 8.5295
3 0 6.5000
4 1 24.0230
```

```
[8]: df=df[['CustomerID','Transaction_ID','Transaction_Date','Product_SKU','Product_Description','I
```

```
[9]: df.isna().sum()
```

```
[9]: CustomerID          0
      Transaction_ID     0
      Transaction_Date   0
      Product_SKU        0
      Product_Description 0
      Invoice            400
      Quantity           0
      Product_Category   0
      Month              0
      Coupon_Code        400
      Coupon             0
      Discount_pct       400
      dtype: int64
```

1.5.6 Imputing Invoice with the median value for that specific CustomerID.

1.5.7 Imputing Coupon_Code with 'No_coupon'

1.5.8 Imputing Discount_pct with 0

```
[10]: df['Invoice'] = df.groupby('CustomerID')['Invoice'].transform(lambda x: x.
      ↪fillna(x.median()))
      df['Coupon_Code'] = df.Coupon_Code.fillna('No_coupon')
      df['Discount_pct'] = df.Discount_pct.fillna(0)
      df.isna().sum()
```

```
[10]: CustomerID          0
      Transaction_ID     0
      Transaction_Date   0
      Product_SKU        0
      Product_Description 0
      Invoice            0
      Quantity           0
      Product_Category   0
      Month              0
      Coupon_Code        0
      Coupon             0
      Discount_pct       0
      dtype: int64
```

```
[11]: for col in df.columns:
      print(f'Number of unique values in {col} is : {df[col].nunique()}')
```

```
Number of unique values in CustomerID is : 1468
Number of unique values in Transaction_ID is : 25061
Number of unique values in Transaction_Date is : 365
Number of unique values in Product_SKU is : 1145
Number of unique values in Product_Description is : 404
Number of unique values in Invoice is : 5648
```

Number of unique values in Quantity is : 151
 Number of unique values in Product_Category is : 20
 Number of unique values in Month is : 12
 Number of unique values in Coupon_Code is : 46
 Number of unique values in Coupon is : 2
 Number of unique values in Discount_pct is : 4

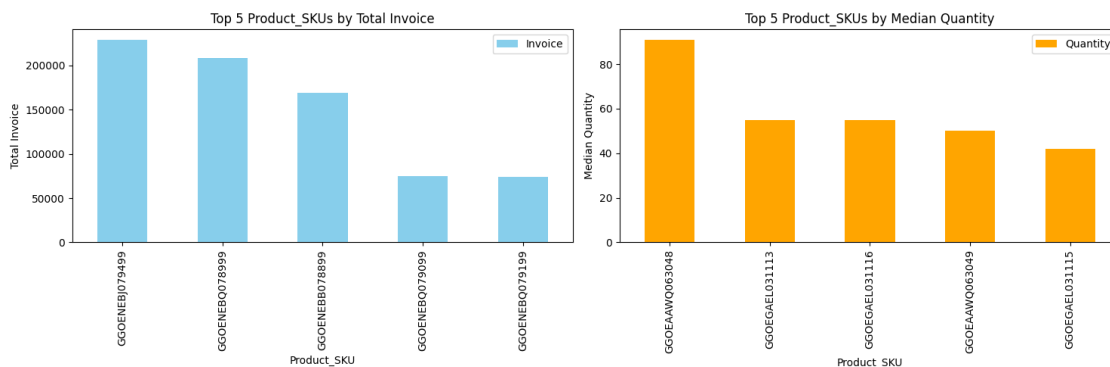
1.5.9 Top 5 Product_SKUs in terms of revenue

```
[12]: sku_grouped = df.groupby('Product_SKU', as_index=False).agg(Invoice=('Invoice',
    ↳ 'sum'), Quantity=('Quantity', 'median'))

sku_grouped_by_invoice = sku_grouped.sort_values('Invoice', ascending=False).
    ↳ head(5)
sku_grouped_by_quantity = sku_grouped.sort_values('Quantity', ascending=False).
    ↳ head(5)

fig, axes = plt.subplots(1, 2, figsize=(15, 5))
sku_grouped_by_invoice.plot(kind='bar', x='Product_SKU', y='Invoice',
    ↳ color='skyblue', ax=axes[0])
axes[0].set_title('Top 5 Product_SKUs by Total Invoice')
axes[0].set_xlabel('Product_SKU')
axes[0].set_ylabel('Total Invoice')

sku_grouped_by_quantity.plot(kind='bar', x='Product_SKU', y='Quantity',
    ↳ color='orange', ax=axes[1])
axes[1].set_title('Top 5 Product_SKUs by Median Quantity')
axes[1].set_xlabel('Product_SKU')
axes[1].set_ylabel('Median Quantity')
plt.tight_layout()
plt.show()
```



```
[13]: print("Top 5 Product_SKUs by Total Invoice:")
print(sku_grouped_by_invoice)

print("\nTop 5 Product_SKUs by Median Quantity:")
print(sku_grouped_by_quantity)
```

Top 5 Product_SKUs by Total Invoice:

	Product_SKU	Invoice	Quantity
981	GGOENEBJ079499	229191.1732	1.0
983	GGOENEBQ078999	208812.3695	1.0
976	GGOENEBO78899	168999.2536	1.0
984	GGOENEBQ079099	74881.1215	2.0
985	GGOENEBQ079199	74133.9858	2.0

Top 5 Product_SKUs by Median Quantity:

	Product_SKU	Invoice	Quantity
146	GGOEAAWQ063048	6.0	91.0
474	GGOEGAEL031113	6.5	55.0
477	GGOEGAEL031116	6.5	55.0
147	GGOEAAWQ063049	6.0	50.0
476	GGOEGAEL031115	6.5	42.0

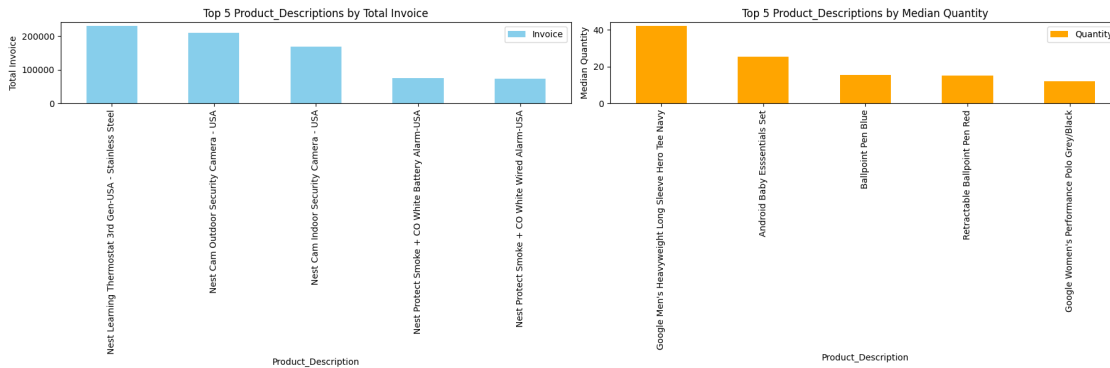
1.5.10 Top 5 Product_Descriptions in terms of revenue

```
[14]: description_grouped = df.groupby('Product_Description', as_index=False).
      ↪agg(Invoice=('Invoice', 'sum'), Quantity=('Quantity', 'median'))

description_grouped_by_invoice = description_grouped.sort_values('Invoice',
      ↪ascending=False)
description_grouped_by_quantity = description_grouped.sort_values('Quantity',
      ↪ascending=False)

fig, axes = plt.subplots(1, 2, figsize=(18, 6))
description_grouped_by_invoice.head(5).plot(kind='bar',
      ↪x='Product_Description', y='Invoice', color='skyblue', ax=axes[0])
axes[0].set_title('Top 5 Product_Descriptions by Total Invoice')
axes[0].set_xlabel('Product_Description')
axes[0].set_ylabel('Total Invoice')

description_grouped_by_quantity.head(5).plot(kind='bar',
      ↪x='Product_Description', y='Quantity', color='orange', ax=axes[1])
axes[1].set_title('Top 5 Product_Descriptions by Median Quantity')
axes[1].set_xlabel('Product_Description')
axes[1].set_ylabel('Median Quantity')
plt.tight_layout()
plt.show()
```

```
[15]: print("Top 5 Product_Descriptions by Total Invoice:")
      print(description_grouped_by_invoice.head(5))

      print("\nTop 5 Product_Descriptions by Median Quantity:")
      print(description_grouped_by_quantity.head(5))
```

Top 5 Product_Descriptions by Total Invoice:

	Product_Description	Invoice	Quantity
316	Nest Learning Thermostat 3rd Gen-USA - Stainle...	229191.1732	1.0
312	Nest Cam Outdoor Security Camera - USA	208812.3695	1.0
310	Nest Cam Indoor Security Camera - USA	168999.2536	1.0
321	Nest Protect Smoke + CO White Battery Alarm-USA	74881.1215	2.0
323	Nest Protect Smoke + CO White Wired Alarm-USA	74133.9858	2.0

Top 5 Product_Descriptions by Median Quantity:

	Product_Description	Invoice	Quantity
162	Google Men's Heavyweight Long Sleeve Hero Tee ...	957.38872	42.0
16	Android Baby Essentials Set	36.50000	25.5
76	Ballpoint Pen Blue	3091.96270	15.5
332	Retractable Ballpoint Pen Red	670.92760	15.0
251	Google Women's Performance Polo Grey/Black	1644.48912	12.0

1.5.11 Top 5 Product_Categorys in terms of revenue

```
[16]: category_grouped = df.groupby('Product_Category', as_index=False).
      ↪agg(Invoice=('Invoice', 'sum'), Quantity=('Quantity', 'median'))

      category_grouped_by_invoice = category_grouped.sort_values('Invoice',
      ↪ascending=False)
      category_grouped_by_quantity = category_grouped.sort_values('Quantity',
      ↪ascending=False)

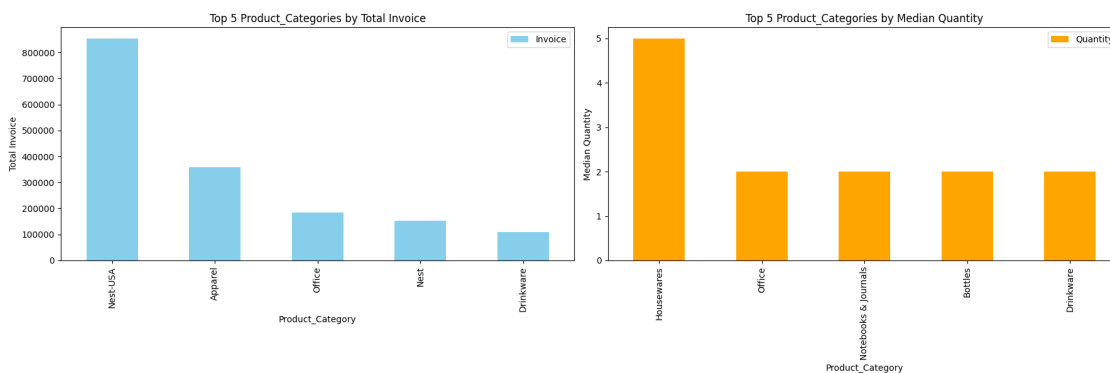
      fig, axes = plt.subplots(1, 2, figsize=(18, 6))
```

```

category_grouped_by_invoice.head(5).plot(kind='bar', x='Product_Category',
    ↪y='Invoice', color='skyblue', ax=axes[0])
axes[0].set_title('Top 5 Product_Categories by Total Invoice')
axes[0].set_xlabel('Product_Category')
axes[0].set_ylabel('Total Invoice')

category_grouped_by_quantity.head(5).plot(kind='bar', x='Product_Category',
    ↪y='Quantity', color='orange', ax=axes[1])
axes[1].set_title('Top 5 Product_Categories by Median Quantity')
axes[1].set_xlabel('Product_Category')
axes[1].set_ylabel('Median Quantity')
plt.tight_layout()
plt.show()

```



```

[17]: print("Top 5 Product_Categories by Total Invoice:")
print(category_grouped_by_invoice.head(5))

print("\nTop 5 Product_Categories by Median Quantity:")
print(category_grouped_by_quantity.head(5))

```

Top 5 Product_Categories by Total Invoice:

	Product_Category	Invoice	Quantity
16	Nest-USA	853645.00510	1.0
2	Apparel	359547.92298	1.0
18	Office	183604.07010	2.0
14	Nest	153509.13940	1.0
6	Drinkware	109896.88510	2.0

Top 5 Product_Categories by Median Quantity:

	Product_Category	Invoice	Quantity
11	Housewares	2934.2164	5.0
18	Office	183604.0701	2.0
17	Notebooks & Journals	43340.5317	2.0
5	Bottles	5893.2286	2.0

1.5.12 Top 5 Product SKUs by Total Invoice:

1. **GGOENEBJ079499:** This SKU corresponds to the Nest Learning Thermostat 3rd Gen-USA - Stainless Steel, which aligns with its top position in terms of total invoice amount.
2. **GGOENEBQ078999:** This SKU represents the Nest Cam Outdoor Security Camera - USA, confirming its popularity as the second-highest in total invoice amount.
3. **GGOENEBB078899:** This SKU corresponds to the Nest Cam Indoor Security Camera - USA, reflecting its strong sales performance as the third-highest in total invoice amount.
4. **GGOENEBQ079099:** Despite being ranked fourth, this SKU corresponds to the Nest Protect Smoke + CO White Battery Alarm-USA, indicating significant sales volume for this product variant.
5. **GGOENEBQ079199:** Similar to the previous SKU, this one corresponds to the wired variant of the Nest Protect Smoke + CO White Alarm-USA, indicating consistent demand for both battery and wired options.

1.5.13 Top 5 Product Descriptions by Total Invoice:

1. **Nest Learning Thermostat 3rd Gen-USA - Stainless Steel:** This product description tops the list in terms of total invoice amount, indicating high demand for this particular Nest product variant.
2. **Nest Cam Outdoor Security Camera - USA:** The outdoor security camera from Nest is the second highest in terms of total invoice amount, suggesting a strong interest in home security products.
3. **Nest Cam Indoor Security Camera - USA:** Following closely behind the outdoor camera, the indoor security camera variant also enjoys significant sales, reflecting a growing concern for home safety.
4. **Nest Protect Smoke + CO White Battery Alarm-USA:** This product description indicates a demand for smoke and CO detectors with battery functionality, as it ranks fourth in total invoice amount.
5. **Nest Protect Smoke + CO White Wired Alarm-USA:** Similar to the battery-powered variant, the wired smoke and CO detector also sees considerable sales, rounding up the top 5 product d

1.5.14 Top 5 Product Categories by Total Invoice:

1. **Nest-USA:** Despite having only one item per invoice, Nest-USA has the highest total invoice amount, indicating high-value purchases.
2. **Apparel:** Apparel follows closely behind Nest-USA in terms of total invoice amount, suggesting a strong demand for clothing products.
3. **Office:** Although ranking third, the Office category has a considerable total invoice amount, indicating a significant volume of purchases, likely for office supplies.
4. **Nest:** Similar to Nest-USA, the Nest category also has a high total invoice amount, indicating a strong demand for Nest products overall.
5. **Drinkware:** Despite ranking fifth, Drinkware has a noteworthy total invoice amount, indicating consistent sales in this product category.

These insights provide a deeper understanding of the top-performing product categories, descriptions, and SKUs based on their total invoice amounts. total invoice amounts.

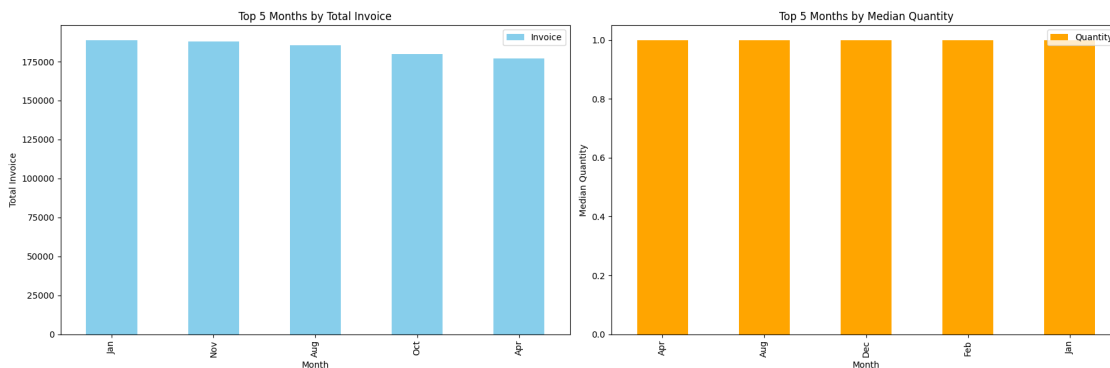
1.5.15 Top 5 Months in terms of revenue

```
[18]: month_grouped = df.groupby('Month', as_index=False).agg(Invoice=('Invoice',
    ↳ 'sum'), Quantity=('Quantity', 'median'))

month_grouped_by_invoice = month_grouped.sort_values('Invoice', ascending=False)
month_grouped_by_quantity = month_grouped.sort_values('Quantity',
    ↳ ascending=False)

fig, axes = plt.subplots(1, 2, figsize=(18, 6))
month_grouped_by_invoice.head(5).plot(kind='bar', x='Month', y='Invoice',
    ↳ color='skyblue', ax=axes[0])
axes[0].set_title('Top 5 Months by Total Invoice')
axes[0].set_xlabel('Month')
axes[0].set_ylabel('Total Invoice')

month_grouped_by_quantity.head(5).plot(kind='bar', x='Month', y='Quantity',
    ↳ color='orange', ax=axes[1])
axes[1].set_title('Top 5 Months by Median Quantity')
axes[1].set_xlabel('Month')
axes[1].set_ylabel('Median Quantity')
plt.tight_layout()
plt.show()
```



```
[19]: print("Top 5 Months by Total Invoice:")
print(month_grouped_by_invoice.head(5))

print("\nTop 5 Months by Median Quantity:")
print(month_grouped_by_quantity.head(5))
```

Top 5 Months by Total Invoice:

	Month	Invoice	Quantity
4	Jan	188859.89905	1.0
9	Nov	187969.78576	1.0
1	Aug	185528.76757	1.0
10	Oct	179983.71291	1.0
0	Apr	177094.95322	1.0

Top 5 Months by Median Quantity:

	Month	Invoice	Quantity
0	Apr	177094.95322	1.0
1	Aug	185528.76757	1.0
2	Dec	167504.75299	1.0
3	Feb	135630.25628	1.0
4	Jan	188859.89905	1.0

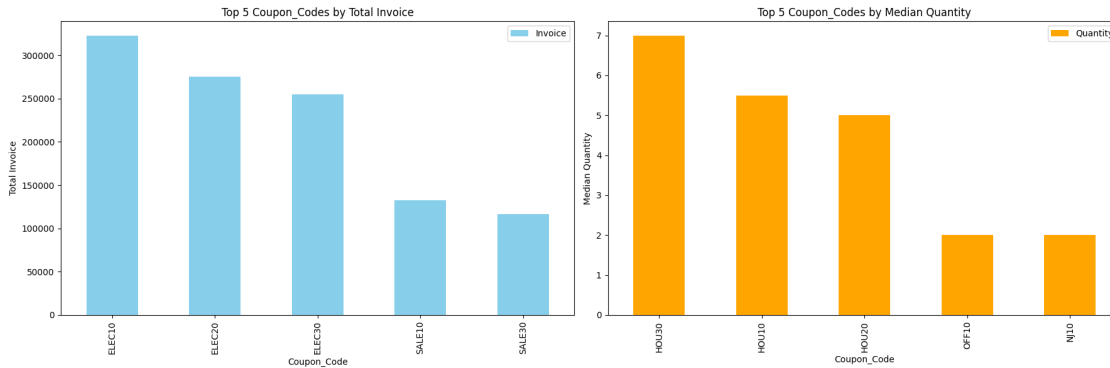
1.5.16 Top 5 Coupon_Codes in terms of revenue

```
[20]: coupon_grouped = df.groupby('Coupon_Code', as_index=False).
      ↪agg(Invoice=('Invoice', 'sum'), Quantity=('Quantity', 'median'))

coupon_grouped_by_invoice = coupon_grouped.sort_values('Invoice',
      ↪ascending=False)
coupon_grouped_by_quantity = coupon_grouped.sort_values('Quantity',
      ↪ascending=False)

fig, axes = plt.subplots(1, 2, figsize=(18, 6))
coupon_grouped_by_invoice.head(5).plot(kind='bar', x='Coupon_Code',
      ↪y='Invoice', color='skyblue', ax=axes[0])
axes[0].set_title('Top 5 Coupon_Codes by Total Invoice')
axes[0].set_xlabel('Coupon_Code')
axes[0].set_ylabel('Total Invoice')

coupon_grouped_by_quantity.head(5).plot(kind='bar', x='Coupon_Code',
      ↪y='Quantity', color='orange', ax=axes[1])
axes[1].set_title('Top 5 Coupon_Codes by Median Quantity')
axes[1].set_xlabel('Coupon_Code')
axes[1].set_ylabel('Median Quantity')
plt.tight_layout()
plt.show()
```



```
[21]: print("Top 5 Coupon_Codes by Total Invoice:")
      print(coupon_grouped_by_invoice.head(5))

      print("\nTop 5 Coupon_Codes by Median Quantity:")
      print(coupon_grouped_by_quantity.head(5))
```

Top 5 Coupon_Codes by Total Invoice:

	Coupon_Code	Invoice	Quantity
12	ELEC10	323126.20410	1.0
13	ELEC20	275706.28000	1.0
14	ELEC30	254812.52100	1.0
40	SALE10	132244.53118	1.0
42	SALE30	116555.15028	1.0

Top 5 Coupon_Codes by Median Quantity:

	Coupon_Code	Invoice	Quantity
26	HOU30	833.06800	7.0
24	HOU10	1289.22840	5.5
25	HOU20	811.92000	5.0
37	OFF10	70327.61470	2.0
33	NJ10	18531.91275	2.0

1.5.17 Top 5 Months by Total Invoice:

1. **January (Jan):** January ranks first in terms of total invoice amount, indicating strong sales at the beginning of the year, possibly due to New Year promotions or post-holiday shopping.
2. **November (Nov):** November closely follows January in total invoice amount, likely boosted by holiday shopping, Black Friday, and Cyber Monday sales.
3. **August (Aug):** August ranks third in total invoice amount, suggesting strong summer sales, possibly due to back-to-school promotions or end-of-summer clearance events.
4. **October (Oct):** October comes in fourth place in terms of total invoice amount, possibly benefiting from fall promotions or early holiday shopping.
5. **April (Apr):** April rounds up the top five months by total invoice amount, indicating solid spring sales, possibly driven by seasonal products or Easter promotions.

1.5.18 Top 5 Coupon Codes by Total Invoice:

1. **ELEC10**: This coupon code has the highest total invoice amount, suggesting that customers are taking advantage of a 10% discount on electronic products, driving significant sales volume.
2. **ELEC20**: The ELEC20 coupon code ranks second in terms of total invoice amount, indicating a strong response to a 20% discount on electronic items.
3. **ELEC30**: Despite being lower than ELEC10 and ELEC20, the ELEC30 coupon code still enjoys considerable usage, indicating a demand for products eligible for a 30% discount on electronics.
4. **SALE10**: This coupon code offers a 10% discount and ranks fourth in total invoice amount, indicating moderate usage compared to the electronics-focused codes.
5. **SALE30**: SALE30 ranks fifth in terms of total invoice amount, suggesting that customers are attracted to a 30% discount on a wide range of products, driving notable sales volume.

These insights provide a glimpse into the top-performing months and coupon codes based on their total invoice amounts, indicating peak sales periods and popular discount offerings.

```
[22]: print('Range of Dates')
dfm['Date']=pd.to_datetime(dfm['Date'])
print((df['Transaction_Date'].min(), df['Transaction_Date'].max(),
      ↪(df['Transaction_Date'].max() - df['Transaction_Date'].min()).days))
print((dfm['Date'].min(), dfm['Date'].max(), (dfm['Date'].max() - dfm['Date'].
      ↪min()).days))
```

Range of Dates

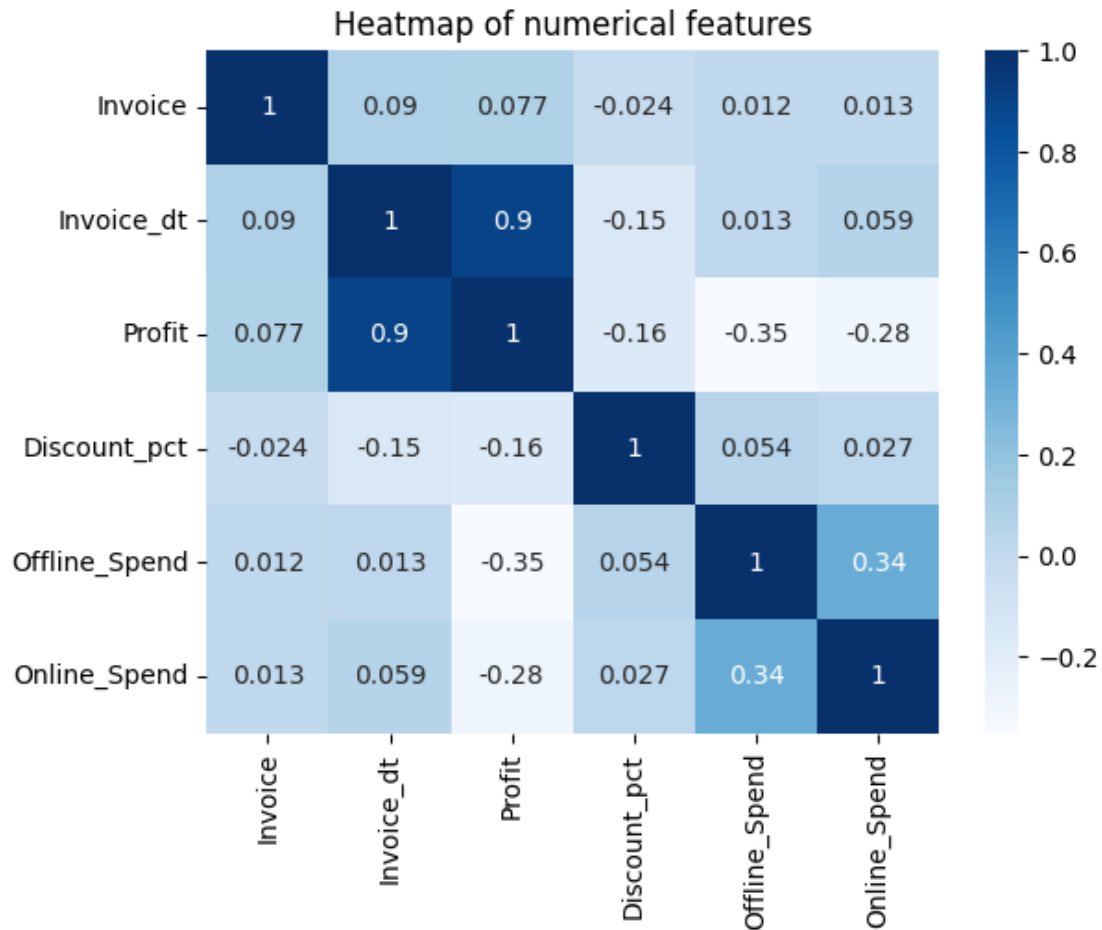
```
(Timestamp('2019-01-01 00:00:00'), Timestamp('2019-12-31 00:00:00'), 364)
(Timestamp('2019-01-01 00:00:00'), Timestamp('2019-12-31 00:00:00'), 364)
```

1.5.19 The data has records from 1st Jan 2019 to 31st December 2019 over a span of 365 days.

1.5.20 Merging with marketing dataframe on Transaction_Date.

```
[23]: df=df.merge(dfm,left_on='Transaction_Date',right_on='Date')
df['Invoice_dt']=df.groupby('Date')['Invoice'].transform('sum')
df['Profit']=df['Invoice_dt']-df['Offline_Spend']-df['Online_Spend']
df=df.drop(columns='Date').rename(columns={'Transaction_Date':'Date'})
```

```
[24]: columns=['Invoice','Invoice_dt','Profit','Discount_pct','Offline_Spend','Online_Spend']
plt.title('Heatmap of numerical features')
sns.heatmap(df[columns].corr(),annot=True,cmap='Blues')
plt.show()
```



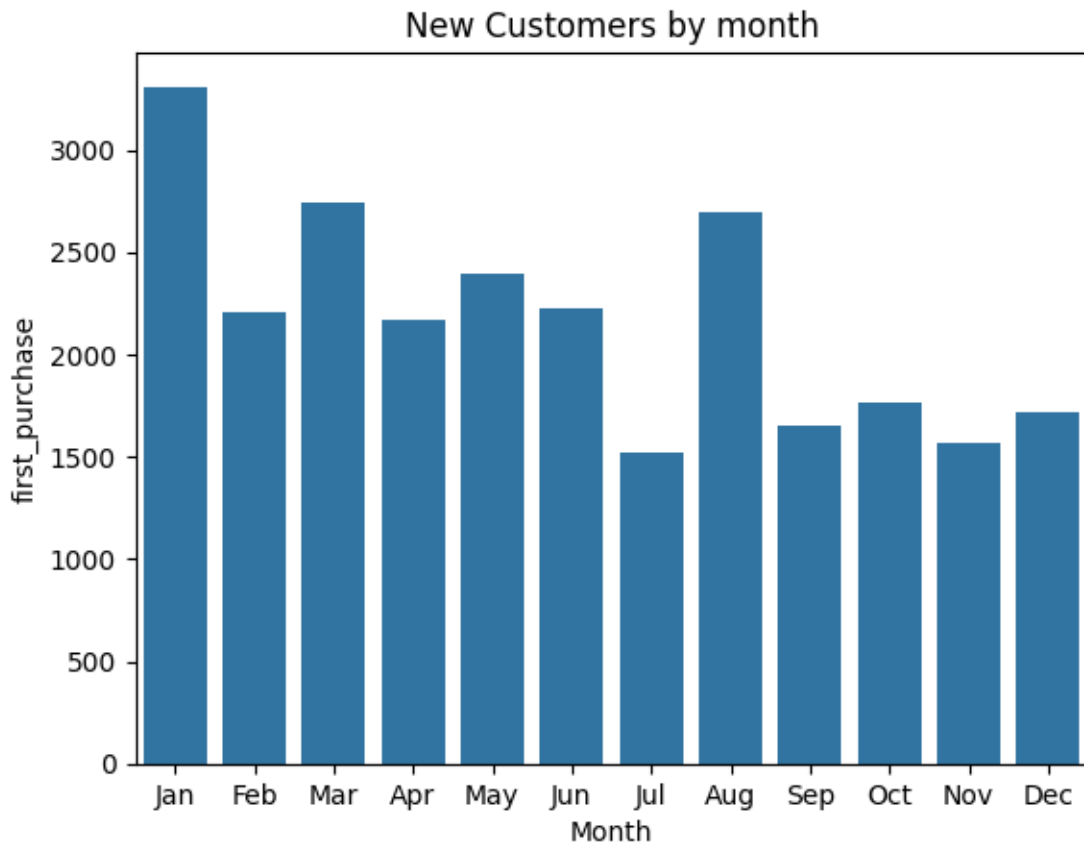
1.6 Profit and total Invoice per date is strongly correlated(.9) which is expected and Offline_Spend and Online_Spend is mildly correlated(.34).

1.6.1 Merging with customers dataframe on CustomerID.

```
[25]: df=df.merge(dfc,on='CustomerID')
```

2 Customer Acquisition

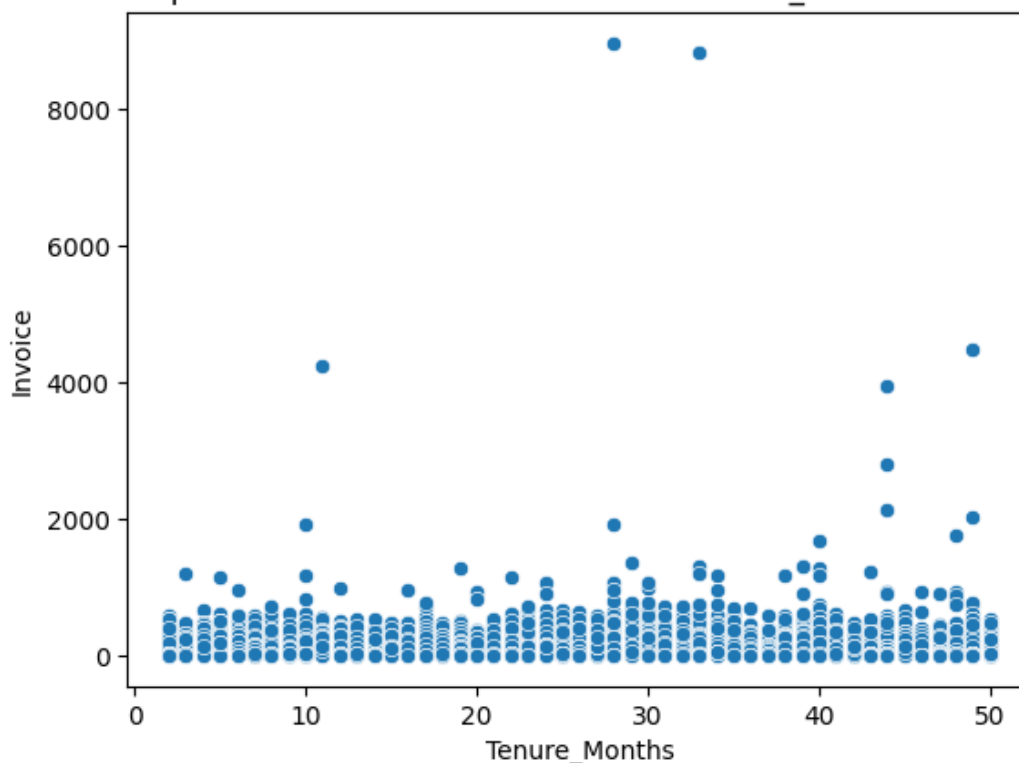
```
[26]: df['first_date']=df.groupby(['CustomerID'])['Date'].transform('min')
df['first_purchase']=np.where(df.Date==df.first_date,1,0)
acq_grouped=df.groupby('Month',as_index=False)['first_purchase'].sum()
sns.
    ↳ barplot(data=acq_grouped,x='Month',y='first_purchase',order=['Jan','Feb','Mar','Apr','May'],
plt.title('New Customers by month')
plt.show()
```

2.1 Jan and Mar have the most new users, with time passing lesser new customers are purchasing which calls for better marketing.

```
[27]: sns.scatterplot(data=df,x='Tenure_Months',y='Invoice')
plt.title('Scatterplot to check correlation between Tenure_Months and Invoice')
plt.show()
print('Spearman rank correlation')
print(spearmanr(df['Tenure_Months'],df['Invoice']))
```

Scatterplot to check correlation between Tenure_Months and Invoice



Spearman rank correlation

SignificanceResult(statistic=-0.006958459528620117, pvalue=0.10942337964383664)

2.1.1 Mostly Invoice is equally distributed withh respect to different Tenure_Months with 2 huge outliers above 8000.

```
[28]: df[df['Invoice']>8000]
```

```
[28]:
```

	CustomerID	Transaction_ID	Date	Product_SKU	\
3284	12748	24860	2019-04-05	GGOEGHPJ080110	
20589	15194	34429	2019-08-02	GGOEGHPJ080310	

	Product_Description	Invoice	Quantity	Product_Category	Month	\
3284	Google 5-Panel Cap	8979.2750	500	Headgear	Apr	
20589	Google Blackout Cap	8836.4076	791	Headgear	Aug	

	Coupon_Code	...	Discount_pct	Offline_Spend	Online_Spend	\
3284	HGEAR10	...	10.0	2500	2342.68	
20589	HGEAR20	...	20.0	1500	2155.96	

	Invoice_dt	Profit	Gender	Location	Tenure_Months	first_date	\
--	------------	--------	--------	----------	---------------	------------	---

3284	25367.74380	20525.06380	F	Chicago	28	2019-01-08
20589	23545.09169	19889.13169	M	Chicago	33	2019-03-16

```

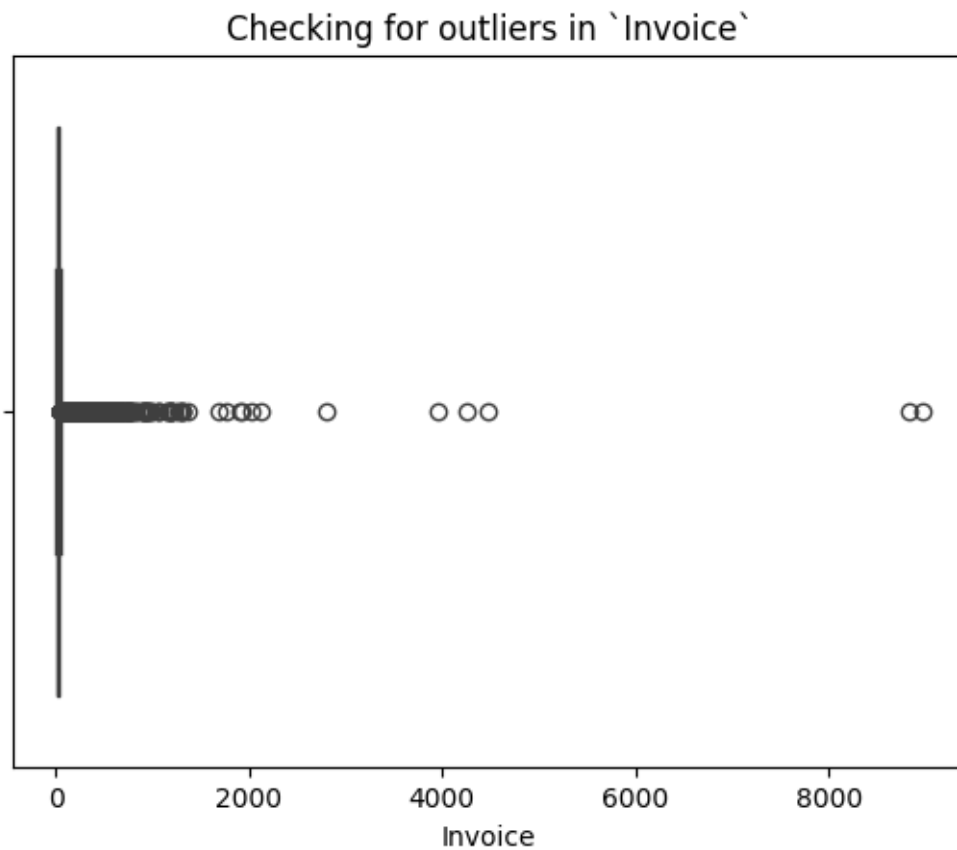
first_purchase
3284          0
20589          0

```

[2 rows x 21 columns]

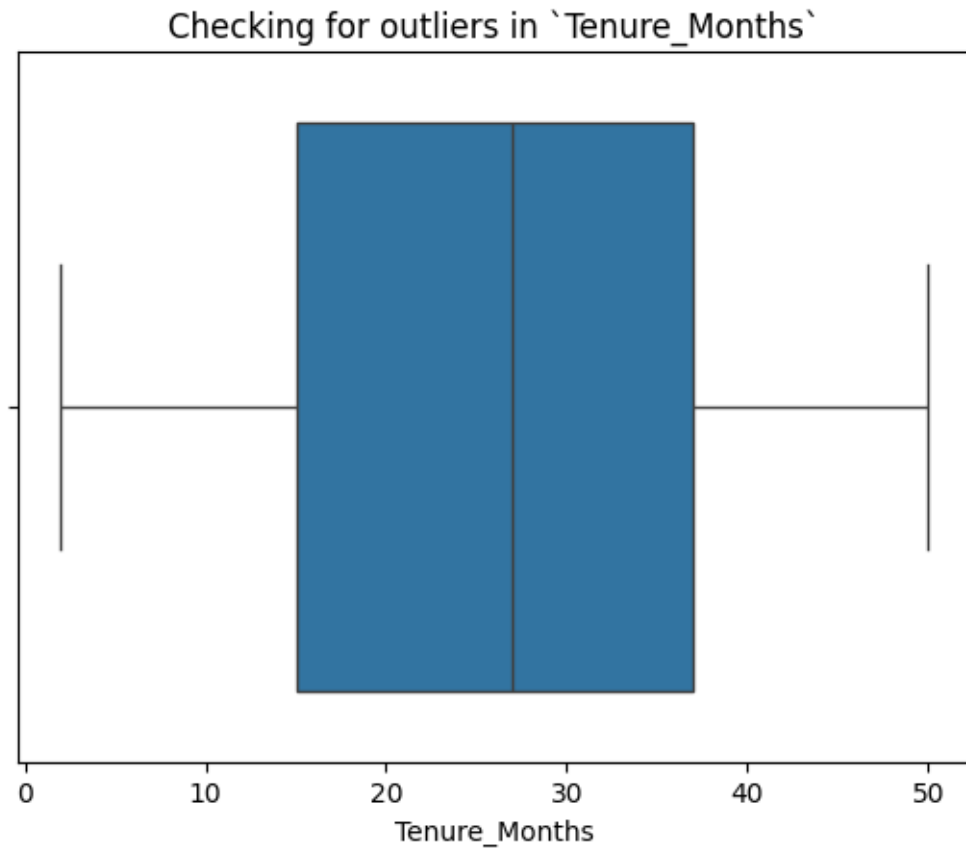
2.1.2 Outliers in Invoice column

```
[29]: sns.boxplot(data=df,x='Invoice')
plt.title('Checking for outliers in `Invoice`')
plt.show()
```



2.1.3 No outliers in Tenure_Months column

```
[30]: sns.boxplot(data=df,x='Tenure_Months')  
plt.title('Checking for outliers in `Tenure_Months`')  
plt.show()
```



2.2 Outliers in invoice target column ignored and will use Ttest which is more robust and Linear models for prediction which does not require normality of the target variable and is simple.

2.3 Tenure_Months have no outliers with normal distribution with mean around 28 months.

2.3.1 Binning Tenure.

```
[31]: bin_edges = [0, 10, 20, 30, 40, float('inf')]  
bin_labels = ['0-10', '10-20', '20-30', '30-40', '>40']  
df['Tenurebin'] = pd.cut(df['Tenure_Months'], bins=bin_edges, labels=bin_labels)
```

```
[32]: df_profit=df.loc[:
      ↪,['Date','Offline_Spend','Online_Spend','Profit','Invoice_dt']].
      ↪drop_duplicates()
df=df[['CustomerID','Transaction_ID','first_purchase','Date','Product_SKU','Product_Description',
      ↪'Coupon_Code']]
df.head()
```

```
[32]: CustomerID Transaction_ID first_purchase Date Product_SKU \
0      17850      16679      1 2019-01-01 GGOENEBJ079499
1      17850      16680      1 2019-01-01 GGOENEBJ079499
2      17850      16681      1 2019-01-01 GGOEGFKQ020399
3      17850      16682      1 2019-01-01 GGOEGAAB010516
4      17850      16682      1 2019-01-01 GGOEGBJL013999

      Product_Description Invoice Quantity \
0 Nest Learning Thermostat 3rd Gen-USA - Stainle... 158.6729      1
1 Nest Learning Thermostat 3rd Gen-USA - Stainle... 158.6729      1
2      Google Laptop and Cell Phone Stickers      8.5295      1
3 Google Men's 100% Cotton Short Sleeve Hero Tee... 6.5000      5
4      Google Canvas Tote Natural/Navy      24.0230      1

      Product_Category Coupon Discount_pct Gender Location Tenure_Months \
0      Nest-USA      1      10.0      M Chicago      12
1      Nest-USA      1      10.0      M Chicago      12
2      Office      1      10.0      M Chicago      12
3      Apparel      0      10.0      M Chicago      12
4      Bags      1      10.0      M Chicago      12

      Tenurebin Month Coupon_Code
0      10-20 Jan      ELEC10
1      10-20 Jan      ELEC10
2      10-20 Jan      OFF10
3      10-20 Jan      SALE10
4      10-20 Jan      AI010
```

3 New vs Existing user sales

```
[33]: new_users = df[df['first_purchase'] == 1]
existing_users = df[df['first_purchase'] == 0]

new_users_monthly = new_users.groupby('Month')['Invoice'].sum().reset_index()
existing_users_monthly = existing_users.groupby('Month')['Invoice'].sum().
      ↪reset_index()

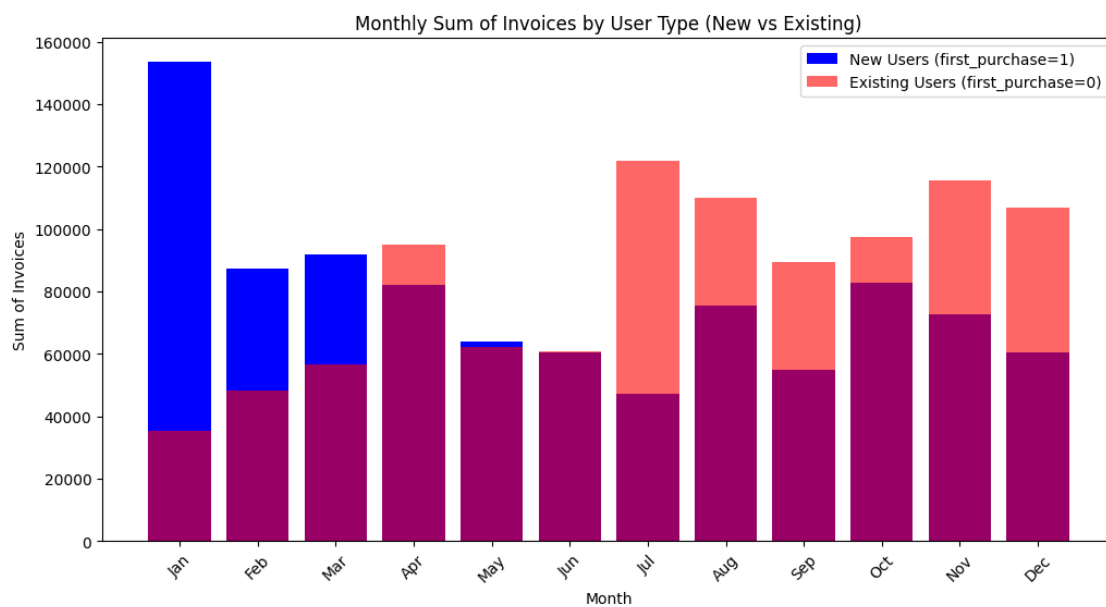
month_order = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep',
      ↪'Oct', 'Nov', 'Dec']
```

```

new_users_monthly['Month'] = pd.Categorical(new_users_monthly['Month'],
↳categories=month_order, ordered=True)
existing_users_monthly['Month'] = pd.
↳Categorical(existing_users_monthly['Month'], categories=month_order,
↳ordered=True)
new_users_monthly = new_users_monthly.sort_values('Month')
existing_users_monthly = existing_users_monthly.sort_values('Month')

fig, ax = plt.subplots(figsize=(12, 6))
ax.bar(new_users_monthly['Month'], new_users_monthly['Invoice'], color='blue',
↳label='New Users (first_purchase=1)')
ax.bar(existing_users_monthly['Month'], existing_users_monthly['Invoice'],
↳color='red', alpha=0.6, label='Existing Users (first_purchase=0)')
plt.xlabel('Month')
plt.ylabel('Sum of Invoices')
plt.title('Monthly Sum of Invoices by User Type (New vs Existing)')
plt.xticks(rotation=45)
plt.legend()
plt.show()

```

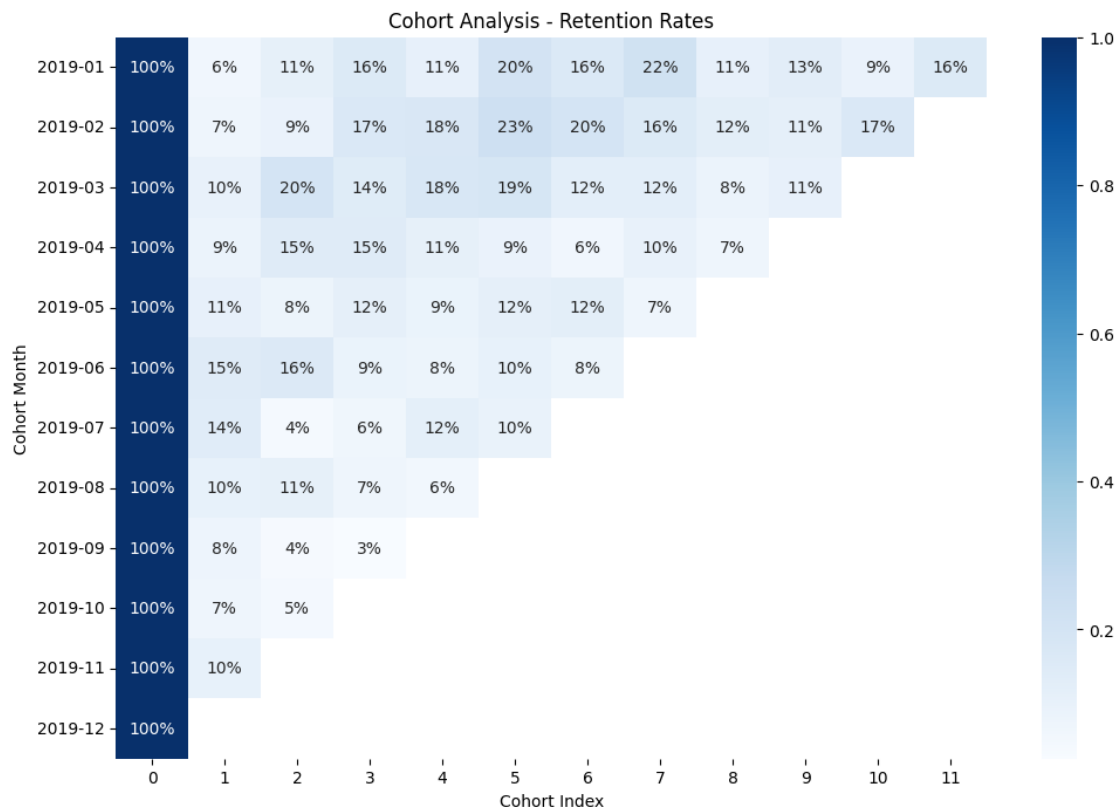


3.1 From Jan to Jun new users contribute to more sale however from Jun exisiting users contribute to more sales.

4 Cohort Analysis

```
[34]: df['CohortMonth'] = df.groupby('CustomerID')['Date'].transform('min').dt.\
    ↪to_period('M')
df['TransactionMonth'] = df['Date'].dt.to_period('M')
df['CohortIndex'] = (df['TransactionMonth'] - df['CohortMonth']).\
    ↪apply(attrgetter('n'))
cohort_data = df.groupby(['CohortMonth', 'CohortIndex'])['CustomerID'].\
    ↪nunique().reset_index()
cohort_counts = cohort_data.pivot(index='CohortMonth', columns='CohortIndex',\
    ↪values='CustomerID')
cohort_sizes = cohort_counts.iloc[:,0]
retention = cohort_counts.divide(cohort_sizes, axis=0)

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



4.1 Cohorts ‘2019-01’ and ‘2019-02’ are slightly outperforming in terms of retention with other cohorts.

5 Organic vs Marketing Sales

```
[35]: org = df[df['Coupon'] == 0]
      mark = df[df['Coupon'] == 1]

      org_monthly = org.groupby('Month')['Invoice'].sum().reset_index()
      mark_monthly = mark.groupby('Month')['Invoice'].sum().reset_index()
      total_monthly = df.groupby('Month')['Invoice'].sum().reset_index()

      org_monthly = pd.merge(org_monthly, total_monthly, on='Month', suffixes=('',
      ↪ '_Total'))
      mark_monthly = pd.merge(mark_monthly, total_monthly, on='Month', suffixes=('',
      ↪ '_Total'))
      org_monthly['Percentage'] = (org_monthly['Invoice'] /
      ↪ org_monthly['Invoice_Total']) * 100
      mark_monthly['Percentage'] = (mark_monthly['Invoice'] /
      ↪ mark_monthly['Invoice_Total']) * 100

      month_order = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep',
      ↪ 'Oct', 'Nov', 'Dec']
      org_monthly['Month'] = pd.Categorical(org_monthly['Month'],
      ↪ categories=month_order, ordered=True)
      mark_monthly['Month'] = pd.Categorical(mark_monthly['Month'],
      ↪ categories=month_order, ordered=True)
      org_monthly = org_monthly.sort_values('Month')
      mark_monthly = mark_monthly.sort_values('Month')

      fig, ax = plt.subplots(figsize=(12, 6))
      ax.bar(org_monthly['Month'], org_monthly['Invoice'], color='blue',
      ↪ label='Organic (Coupon=0)')
      ax.bar(mark_monthly['Month'], mark_monthly['Invoice'],
      ↪ bottom=org_monthly['Invoice'], color='red', alpha=0.6, label='Marketing
      ↪ (Coupon=1)')

      for i in range(len(org_monthly)):
          ax.text(x=i, y=org_monthly['Invoice'][i] / 2,
          ↪ s=f"{org_monthly['Percentage'][i]:.1f}%",
                  color='white', ha='center', va='center', fontsize=10)
          ax.text(x=i, y=org_monthly['Invoice'][i] + mark_monthly['Invoice'][i] / 2,
          ↪ s=f"{mark_monthly['Percentage'][i]:.1f}%",
```

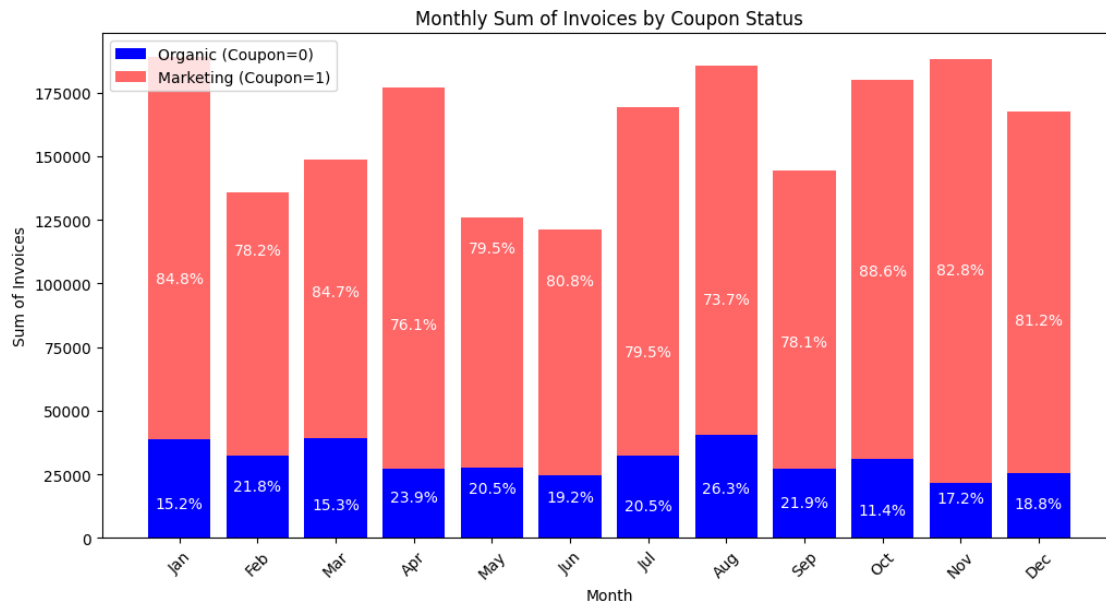


```

        color='white', ha='center', va='center', fontsize=10)

plt.xlabel('Month')
plt.ylabel('Sum of Invoices')
plt.title('Monthly Sum of Invoices by Coupon Status')
plt.xticks(rotation=45)
plt.legend()
plt.show()

```



5.1 Jan has the highest overall sales, Oct has the highest % marketing sales and August has the highest % organic sales.

6 Temporal Trends due to Marketing

```

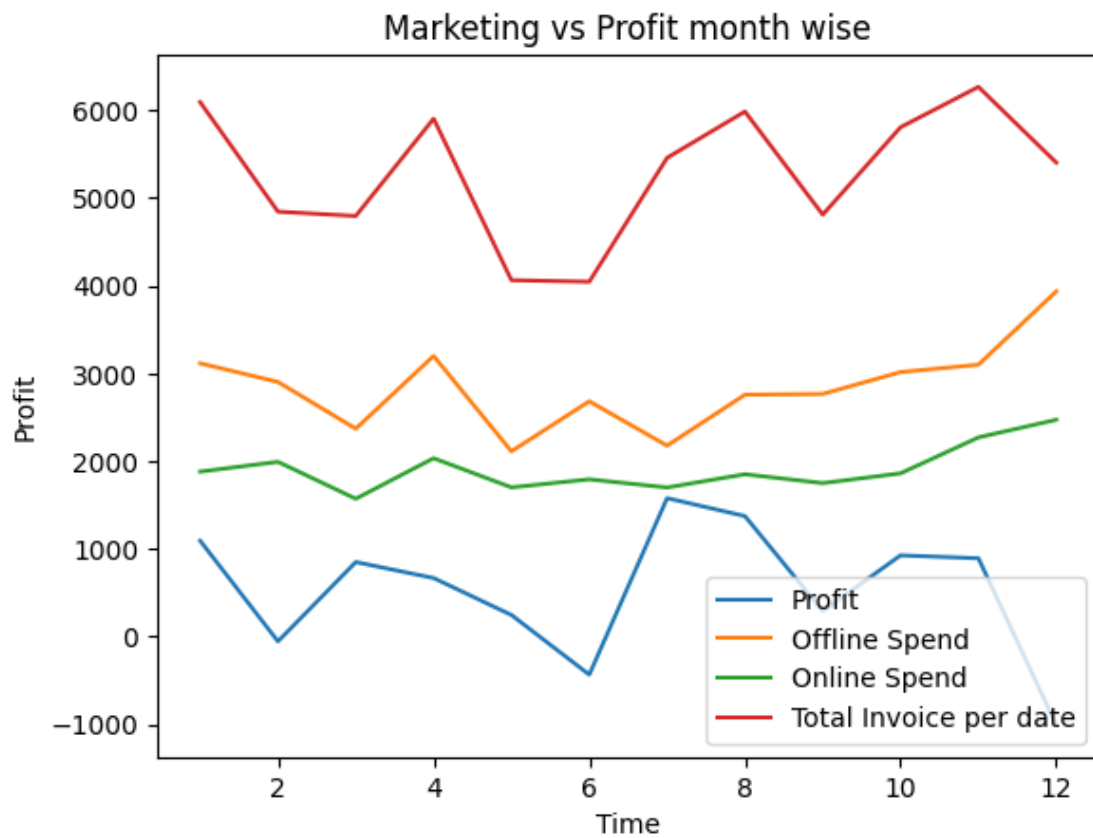
[36]: df_profit['Time'] = df_profit.Date.dt.month

sns.lineplot(data=df_profit, x='Time', y='Profit', ci=None, label='Profit')
sns.lineplot(data=df_profit, x='Time', y='Offline_Spend', ci=None,
             ↪label='Offline Spend')
sns.lineplot(data=df_profit, x='Time', y='Online_Spend', ci=None, label='Online_
             ↪Spend')
sns.lineplot(data=df_profit, x='Time', y='Invoice_dt', ci=None, label='Total_
             ↪Invoice per date')

plt.title('Marketing vs Profit month wise')
plt.legend()

```

```
plt.show()
```

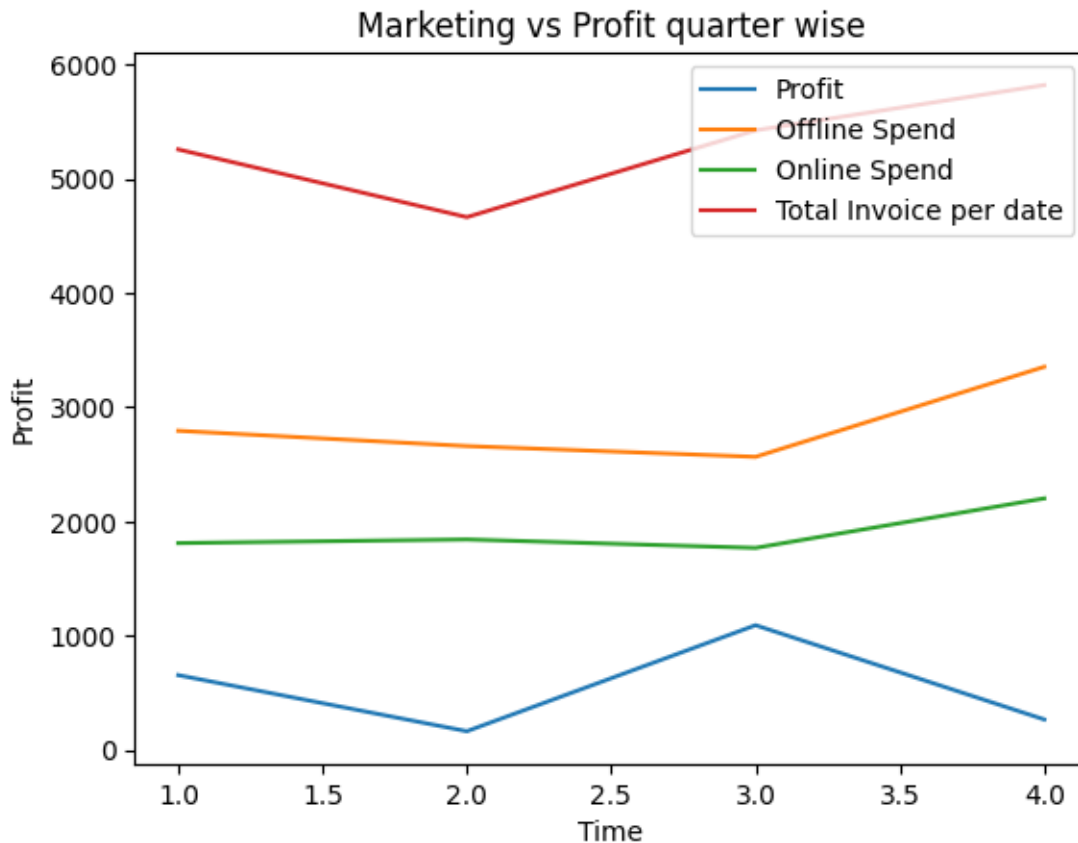


```
[37]: df_profit['Time'] = df_profit.Date.dt.quarter

sns.lineplot(data=df_profit, x='Time', y='Profit', ci=None, label='Profit')
sns.lineplot(data=df_profit, x='Time', y='Offline_Spend', ci=None,
             label='Offline Spend')
sns.lineplot(data=df_profit, x='Time', y='Online_Spend', ci=None, label='Online_
             Spend')
sns.lineplot(data=df_profit, x='Time', y='Invoice_dt', ci=None, label='Total_
             Invoice per date')

plt.title('Marketing vs Profit quarter wise')
plt.legend()

plt.show()
```

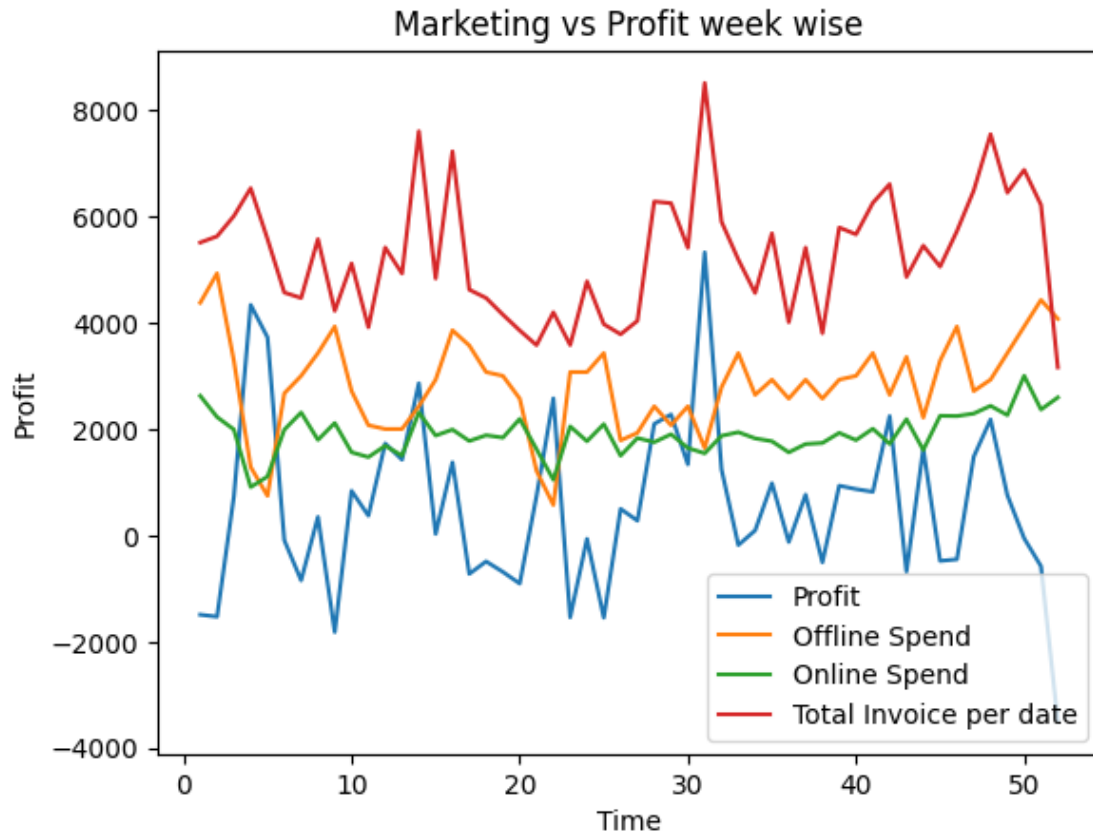


```
[38]: df_profit['Time'] = df_profit['Date'].dt.isocalendar().week

sns.lineplot(data=df_profit, x='Time', y='Profit', ci=None, label='Profit')
sns.lineplot(data=df_profit, x='Time', y='Offline_Spend', ci=None,
             label='Offline Spend')
sns.lineplot(data=df_profit, x='Time', y='Online_Spend', ci=None, label='Online_
             Spend')
sns.lineplot(data=df_profit, x='Time', y='Invoice_dt', ci=None, label='Total_
             Invoice per date')

plt.title('Marketing vs Profit week wise')
plt.legend()

plt.show()
```



6.1 Middle of the year shows lowest spendings and maximum profit typically in Q02-Q03 and week number 32 in month 7-8. Lowest profit is seen during end of the year when marketing efforts require more spendings and also volume of sales is going down.

7 RFM Analysis

```
[39]: df=df.drop_duplicates(subset=['CustomerID','Transaction_ID','Product_SKU'])
df.shape
```

```
[39]: (52924, 20)
```

```
[40]: df['last']=df.groupby('CustomerID')['Date'].transform('max')
df['days']=(df['last']-df['Date'].min()).dt.days
df['count']=df.groupby('CustomerID')['Transaction_ID'].transform('nunique')
df['frequency_']=df['count']/(1+df['days'])
df['monetary_']=df.groupby('CustomerID')['Invoice'].transform('median') #Taking_
↳median since heavily skewed
```

```
df['recency_'] = -(df['Date'].max() - df['last']).dt.days #Taking minus since  
↪ reverse
```

```
[41]: df1=df  
df=df[['CustomerID','recency_','frequency_','monetary_']].drop_duplicates()  
num_quantiles = 5  
df['recency_'] = pd.qcut(df['recency_'], num_quantiles, labels=False,  
    ↪ duplicates='drop')  
df['frequency_'] = pd.qcut(df['frequency_'], num_quantiles, labels=False,  
    ↪ duplicates='drop')  
df['monetary_'] = pd.qcut(df['monetary_'], num_quantiles, labels=False,  
    ↪ duplicates='drop')  
  
# To compensate the dropped ones  
df['recency_'] += 1  
df['frequency_'] += 1  
df['monetary_'] += 1  
  
df['FM'] = np.round((df['frequency_'].astype(int) + df['monetary_'].astype(int)) /  
    ↪ 2)
```

```
[42]: df['recency_'].value_counts()
```

```
[42]: recency  
3      296  
1      294  
5      293  
2      293  
4      292  
Name: count, dtype: int64
```

```
[43]: df['frequency_'].value_counts()
```

```
[43]: frequency  
1      296  
3      295  
5      293  
4      293  
2      291  
Name: count, dtype: int64
```

```
[44]: df['monetary_'].value_counts()
```

```
[44]: monetary  
1      869  
3      320  
4      267
```

```
2      12
Name: count, dtype: int64
```

```
[45]: def assign_rfm_segment(row):
        r_score = row['recency']
        fm_score = row['FM']

        if (r_score == 5 and fm_score == 5) or (r_score == 5 and fm_score == 4) or
        (r_score == 4 and fm_score == 5):
            return 'Champions'
        elif (r_score == 5 and fm_score == 3) or (r_score == 4 and fm_score == 4) or
        (r_score == 3 and fm_score == 5) or (r_score == 3 and fm_score == 4):
            return 'Loyal Customers'
        elif (r_score == 5 and fm_score == 2) or (r_score == 4 and fm_score == 2) or
        (r_score == 3 and fm_score == 3) or (r_score == 4 and fm_score == 3):
            return 'Potential Loyalists'
        elif r_score == 5 and fm_score == 1:
            return 'Recent Customers'
        elif (r_score == 4 and fm_score == 1) or (r_score == 3 and fm_score == 1):
            return 'Promising'
        elif (r_score == 3 and fm_score == 2) or (r_score == 2 and fm_score == 3) or
        (r_score == 2 and fm_score == 2):
            return 'Customers Needing Attention'
        elif r_score == 2 and fm_score == 1:
            return 'About to Sleep'
        elif (r_score == 2 and fm_score == 5) or (r_score == 2 and fm_score == 4) or
        (r_score == 1 and fm_score == 3):
            return 'At Risk'
        elif (r_score == 1 and fm_score == 5) or (r_score == 1 and fm_score == 4):
            return 'Cant Lose Them'
        elif r_score == 1 and fm_score == 2:
            return 'Hibernating'
        elif r_score == 1 and fm_score == 1:
            return 'Lost'

df['rfm_segment'] = df.apply(assign_rfm_segment, axis=1)
df.head()
```

```
[45]: CustomerID  recency_  frequency_  monetary_  recency  frequency  monetary  \
0      17850      -339      6.807692      6.50000      1          5          1
297    13047       -13      0.073864      6.50000      5          3          1
341    12583      -151      0.070093      6.50000      3          3          1
383    13748      -364      1.000000      6.50000      1          5          1
384    15100      -123      0.024793     11.16576      3          2          3

      FM      rfm_segment
0    3.0      At Risk
```

```

297 2.0          Potential Loyalists
341 2.0  Customers Needing Attention
383 3.0          At Risk
384 2.0  Customers Needing Attention

```

7.1 Defining recency score of 1,2 and FM score of 1,2 as churned customer.

7.1.1 There is no such fixed rule so I have picked a suitable condition to label churn.

```

[46]: df['churn']=df.apply(lambda x: 1 if (x['recency']<=2) and (x['FM']<=2) else 0,axis=1)

```

8 KMeans segmentation

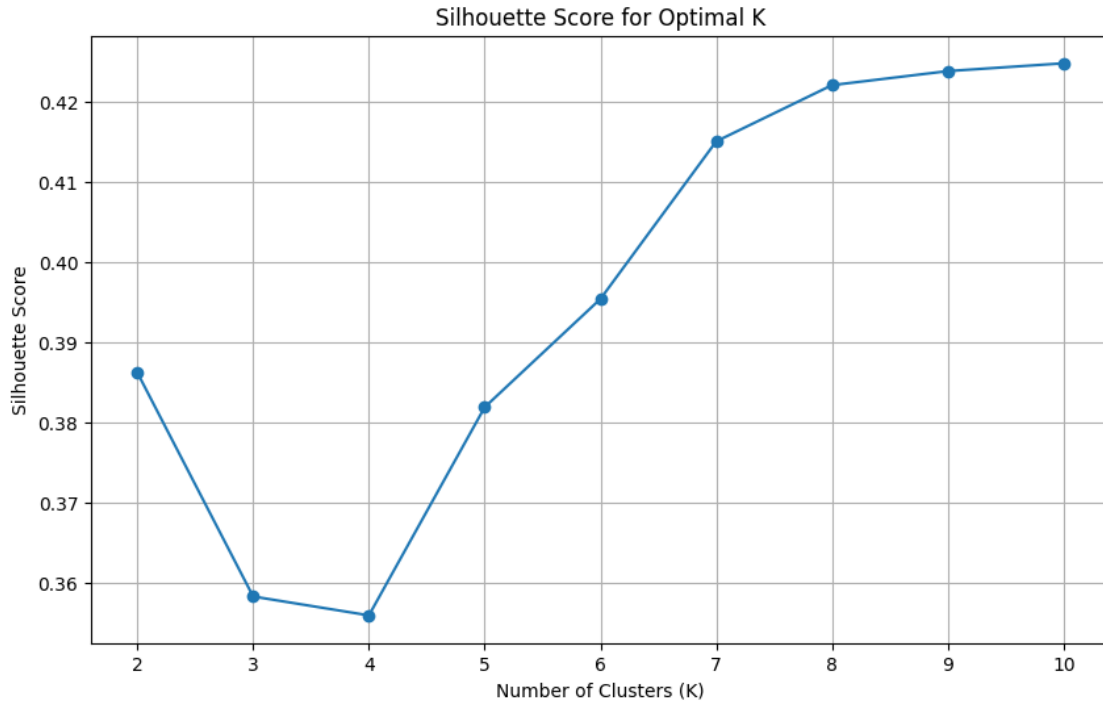
```

[47]: data = df[['recency', 'frequency', 'monetary']]
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data)

silhouette_scores = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=95)
    kmeans.fit(scaled_data)
    silhouette_scores.append(silhouette_score(scaled_data, kmeans.labels_))

plt.figure(figsize=(10, 6))
plt.plot(range(2, 11), silhouette_scores, marker='o', linestyle='-')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Score for Optimal K')
plt.xticks(range(2, 11))
plt.grid(True)
plt.show()

```



```
[48]: optimal_k = 4
kmeans = KMeans(n_clusters=optimal_k, random_state=95)
kmeans.fit(scaled_data)
cluster_labels = kmeans.labels_
df['Cluster'] = cluster_labels

[49]: cluster_summary = df.groupby('Cluster')[['recency_', 'frequency_',
↪ 'monetary_']].describe()
cluster_summary.columns = [f"{col[0]}_{col[1]}" for col in cluster_summary.
↪ columns]
cluster_summary = cluster_summary.round(2)
cluster_summary = cluster_summary.T
print(cluster_summary)
```

Cluster	0	1	2	3
recency__count	306.00	533.00	281.00	348.00
recency__mean	-70.99	-84.62	-226.78	-233.53
recency__std	48.16	67.38	65.09	88.36
recency__min	-158.00	-302.00	-362.00	-364.00
recency__25%	-112.50	-131.00	-270.00	-311.25
recency__50%	-69.00	-65.00	-221.00	-241.50
recency__75%	-23.00	-28.00	-178.00	-167.00
recency__max	0.00	0.00	-99.00	-40.00
frequency__count	306.00	533.00	281.00	348.00

frequency__mean	0.05	0.06	0.13	0.43
frequency__std	0.08	0.07	0.22	1.16
frequency__min	0.00	0.00	0.00	0.02
frequency__25%	0.02	0.02	0.03	0.08
frequency__50%	0.03	0.04	0.08	0.15
frequency__75%	0.06	0.07	0.15	0.29
frequency__max	0.79	0.82	2.46	13.33
monetary__count	306.00	533.00	281.00	348.00
monetary__mean	20.79	6.25	25.94	6.33
monetary__std	29.56	0.25	41.88	0.24
monetary__min	7.05	6.00	7.13	6.00
monetary__25%	10.68	6.00	12.48	6.00
monetary__50%	12.99	6.00	13.75	6.50
monetary__75%	17.60	6.50	20.86	6.50
monetary__max	324.00	6.99	541.15	6.79

```
[50]: df_segment=df
df=df1
df=df.merge(df_segment,on='CustomerID')
```

```
[51]: df=df[['CustomerID', 'Transaction_ID', 'Date', 'Product_SKU', 'Product_Description',
            'Invoice', 'Quantity', 'Product_Category', 'Month', 'Coupon_Code',
            'Coupon', 'Discount_pct', 'Tenurebin', 'Tenure_Months', 'Location',
            'Gender', 'rfm_segment', 'churn']]
```

9 Market Basket Analysis

```
[52]: basket = (df
            .groupby(['Transaction_ID', 'Product_Description'])['Quantity']
            .sum().unstack().reset_index().fillna(0)
            .set_index('Transaction_ID'))

def encode_units(x):
    return 0 if x <= 0 else 1

basket = basket.applymap(encode_units)
frequent_itemsets = apriori(basket, min_support=0.01, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
rules.sort_values('lift', ascending=False)
```

```
[52]:
```

	antecedents	\
0	(Nest Cam Outdoor Security Camera - USA)	
1	(Nest Cam Indoor Security Camera - USA)	

	consequents	antecedent support	\
0	(Nest Cam Indoor Security Camera - USA)	0.132796	

1	(Nest Cam Outdoor Security Camera - USA)	0.128886
---	--	----------

	consequent support	support	confidence	lift	leverage	conviction \
0	0.128886	0.027653	0.208233	1.615644	0.010537	1.100216
1	0.132796	0.027653	0.214551	1.615644	0.010537	1.104087

	zhangs_metric
0	0.439403
1	0.437430

```
[53]: basket = (df
        .groupby(['Transaction_ID', 'Product_SKU'])['Quantity']
        .sum().unstack().reset_index().fillna(0)
        .set_index('Transaction_ID'))

def encode_units(x):
    return 0 if x <= 0 else 1

basket = basket.applymap(encode_units)
frequent_itemsets = apriori(basket, min_support=0.01, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
rules.sort_values('lift', ascending=False)
```

	antecedents	consequents	antecedent support	consequent support \
0	(GGOEGHGR019499)	(GGOEGHGC019799)	0.014644	0.017677
1	(GGOEGHGC019799)	(GGOEGHGR019499)	0.017677	0.014644
2	(GGOENEBQ078999)	(GGOENEBB078899)	0.132796	0.128886
3	(GGOENEBB078899)	(GGOENEBQ078999)	0.128886	0.132796

	support	confidence	lift	leverage	conviction	zhangs_metric
0	0.010654	0.727520	41.156636	0.010395	3.605126	0.990203
1	0.010654	0.602709	41.156636	0.010395	2.480185	0.993260
2	0.027653	0.208233	1.615644	0.010537	1.100216	0.439403
3	0.027653	0.214551	1.615644	0.010537	1.104087	0.437430

```
[54]: basket = (df
        .groupby(['Transaction_ID', 'Product_Category'])['Quantity']
        .sum().unstack().reset_index().fillna(0)
        .set_index('Transaction_ID'))

def encode_units(x):
    return 0 if x <= 0 else 1

basket = basket.applymap(encode_units)
frequent_itemsets = apriori(basket, min_support=0.01, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
```

```
rules[rules['zhangs_metric']>=.85].sort_values('lift', ascending=False).
↪reset_index(drop=True)
```

```
[54]:
```

	antecedents	consequents	antecedent support	\
0	(Lifestyle)	(Office, Bags)	0.068313	
1	(Drinkware)	(Office, Bags)	0.100714	
2	(Office, Drinkware)	(Lifestyle)	0.046287	
3	(Lifestyle)	(Office, Drinkware)	0.068313	
4	(Bags)	(Office, Drinkware)	0.061650	
5	(Office)	(Bags, Lifestyle)	0.140697	
6	(Drinkware)	(Office, Lifestyle)	0.100714	
7	(Office)	(Drinkware, Bags)	0.140697	
8	(Office)	(Drinkware, Lifestyle)	0.140697	
9	(Office)	(Notebooks & Journals)	0.140697	

	consequent support	support	confidence	lift	leverage	conviction	\
0	0.026336	0.010175	0.148949	5.655759	0.008376	1.144072	
1	0.026336	0.014285	0.141838	5.385774	0.011633	1.134593	
2	0.068313	0.016719	0.361207	5.287504	0.013557	1.458511	
3	0.046287	0.016719	0.244743	5.287504	0.013557	1.262766	
4	0.046287	0.014285	0.231715	5.006047	0.011432	1.241353	
5	0.014963	0.010175	0.072320	4.833091	0.008070	1.061828	
6	0.035114	0.016719	0.166006	4.727596	0.013183	1.156946	
7	0.021707	0.014285	0.101531	4.677354	0.011231	1.088845	
8	0.025857	0.016719	0.118832	4.595736	0.013081	1.105513	
9	0.024740	0.013846	0.098412	3.977900	0.010365	1.081714	

	zhangs_metric
0	0.883547
1	0.905525
2	0.850229
3	0.870330
4	0.852817
5	0.922949
6	0.876780
7	0.914932
8	0.910513
9	0.871184

9.0.1 Single Product Association:

1. Association between Specific Products:

- There is a significant association between the Nest Cam Indoor Security Camera - USA and the Nest Cam Outdoor Security Camera - USA. This association is bidirectional, indicating that customers who purchase one camera are likely to purchase the other as well.
- Similarly, there is a strong association between product SKUs GGOEGHGC019799 and GGOEGHGR019499, suggesting that customers who buy one SKU are highly likely to

purchase the other.

9.0.2 Product Combination and Cross-Category Associations:

2. Association between Product Combinations and Cross-Category Behavior:

- This analysis identifies associations not only between specific product combinations but also across different categories. For instance, it observes a notable association between lifestyle products and the purchase of office and bags items together, indicating that customers interested in lifestyle products tend to also buy office and bags items. Additionally, it uncovers associations between drinkware and office items purchased together, suggesting that customers purchasing drinkware are likely to buy office supplies. Moreover, it recognizes that office items have associations with various other categories such as bags, lifestyle, and drinkware, indicating common purchasing patterns across different product categories. These findings provide insights into customer preferences and behaviors, facilitating opportunities for cross-selling and marketing strategies across a diverse range of product categories.

10 Descriptive Statistics

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

```
[55]:
```

	CustomerID	Transaction_ID	Date	\
count	52924.0	52924.0	52924	
unique	1468.0	25061.0	NaN	
top	12748.0	32526.0	NaN	
freq	695.0	35.0	NaN	
mean	NaN	NaN	2019-07-05 19:16:09.450532864	
min	NaN	NaN	2019-01-01 00:00:00	
25%	NaN	NaN	2019-04-12 00:00:00	
50%	NaN	NaN	2019-07-13 00:00:00	
75%	NaN	NaN	2019-09-27 00:00:00	
max	NaN	NaN	2019-12-31 00:00:00	
std	NaN	NaN	NaN	

	Product_SKU	Product_Description	\
count	52924	52924	
unique	1145	404	
top	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen-USA - Stainle...	
freq	3511	3511	
mean	NaN	NaN	
min	NaN	NaN	
25%	NaN	NaN	
50%	NaN	NaN	
75%	NaN	NaN	
max	NaN	NaN	
std	NaN	NaN	

	Invoice	Quantity	Product_Category	Month	Coupon_Code	\
count	52924.000000	52924.000000	52924	52924	52924	
unique	NaN	NaN	20	12	46	
top	NaN	NaN	Apparel	Aug	SALE20	
freq	NaN	NaN	18126	6150	6373	
mean	36.505044	4.497638	NaN	NaN	NaN	
min	0.000000	1.000000	NaN	NaN	NaN	
25%	6.000000	1.000000	NaN	NaN	NaN	
50%	6.500000	1.000000	NaN	NaN	NaN	
75%	23.444437	2.000000	NaN	NaN	NaN	
max	8979.275000	900.000000	NaN	NaN	NaN	
std	99.082101	20.104711	NaN	NaN	NaN	

	Coupon	Discount_pct	Tenurebin	Tenure_Months	Location	Gender	\
count	52924.000000	52924.000000	52924	52924.000000	52924	52924	
unique	NaN	NaN	5	NaN	5	2	
top	NaN	NaN	20-30	NaN	Chicago	F	
freq	NaN	NaN	12588	NaN	18380	33007	
mean	0.338296	19.802358	NaN	26.127995	NaN	NaN	
min	0.000000	0.000000	NaN	2.000000	NaN	NaN	
25%	0.000000	10.000000	NaN	15.000000	NaN	NaN	
50%	0.000000	20.000000	NaN	27.000000	NaN	NaN	
75%	1.000000	30.000000	NaN	37.000000	NaN	NaN	
max	1.000000	30.000000	NaN	50.000000	NaN	NaN	
std	0.473134	8.278878	NaN	13.478285	NaN	NaN	

	rfm_segment	churn
count	52924	52924.000000
unique	11	NaN
top	Potential Loyalists	NaN
freq	18250	NaN
mean	NaN	0.068324
min	NaN	0.000000
25%	NaN	0.000000
50%	NaN	0.000000
75%	NaN	0.000000
max	NaN	1.000000
std	NaN	0.252304

10.1 Descriptive Statistics Insight:

- **Customer Count:** There are 1468 unique customers in the dataset.
- **Transaction Count:** There are 25061 unique transactions in the dataset.
- **Date:** Transactions span from January 1, 2019, to December 31, 2019, with an average transaction date of July 5, 2019.
- **Invoice Amount:** The average invoice amount is \$36.51, with a minimum of \$0 and a maximum of \$8,979.28.
– Std: \$99.08

- Median: \$6.50
- **Quantity:** The average quantity per transaction is 4.50, with a minimum of 1 and a maximum of 900.
 - Std: 20.10
 - Median: 1.00
- **Product Category:** The most frequent product category is Apparel, accounting for 18,126 transactions.
- **Month:** Transactions are spread across 12 months, with August being the most frequent month (6,150 transactions).
- **Coupon Code:** The most frequently used coupon code is SALE20, used in 6,373 transactions.
- **Discount Percentage:** Coupon is applied 33.83% times with mean percentage 19.8% and minimum of 0% and maximum of 30%.
 - Std: 8.29%
 - Median 20%
- **Tenure Months:** The average tenure of customers is approximately 26.13 months, with a range from 2 to 50 months.
 - Std: 13.48 months
 - Median: 27.00 months
- **Location:** The majority of transactions (18380) originate from Chicago.
- **Gender:** Transactions are primarily from female customers, with a frequency of 33,007.
- **RFM Segment:** The most common RFM segment is Potential Loyalists, identified in 18,250 transactions.
- **Churn Rate:** The overall churn rate is approximately 6.83%.

11 Multivariate Analysis

11.0.1 Getting the mode Product purchased by each groups.

```
[56]: df.Product_Description=df.Product_Description.str[:32]
cat_col1 = ['Product_SKU', 'Product_Description']
cat_col2 = ['Gender', 'churn', 'Tenurebin', 'rfm_segment', 'Location',
↪, 'Coupon_Code']
```

```
[57]: print(f'MODE Product_SKU and Product_Description by Month:')
print(df.groupby('Month', as_index=False)[cat_col1].agg(lambda x: pd.Series.
↪mode(x)[0]))
print()
```

MODE Product_SKU and Product_Description by Month:

	Month	Product_SKU	Product_Description
0	Apr	GGOENEBB078899	Nest Learning Thermostat 3rd Gen
1	Aug	GGOENEBQ078999	Nest Learning Thermostat 3rd Gen
2	Dec	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen
3	Feb	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen
4	Jan	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen
5	Jul	GGOENEBQ078999	Nest Learning Thermostat 3rd Gen
6	Jun	GGOENEBQ078999	Nest Learning Thermostat 3rd Gen

7	Mar	GGOENEBQ078999	Nest Learning Thermostat 3rd Gen
8	May	GGOENEBO078899	Nest Learning Thermostat 3rd Gen
9	Nov	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen
10	Oct	GGOENEBQ078999	Nest Learning Thermostat 3rd Gen
11	Sep	GGOENEBO078899	Nest Learning Thermostat 3rd Gen

```
[58]: print(f'Top 5 Product_SKU by Month and total Invoice:')
print(df.groupby(['Month', 'Product_SKU'], as_index=False)['Invoice'].sum().
      ↪sort_values('Invoice', ascending=False).head(5).reset_index(drop=True))
print()
```

Top 5 Product_SKU by Month and total Invoice:

	Month	Product_SKU	Invoice
0	Jan	GGOENEBJ079499	40767.5780
1	Jan	GGOENEBQ078999	26076.3675
2	Feb	GGOENEBJ079499	21766.1400
3	Nov	GGOENEBJ079499	21572.8000
4	Dec	GGOENEBJ079499	20807.1352

```
[59]: print(f'Top 5 Product_Category by Month and total Invoice:')
print(df.groupby(['Month', 'Product_Category'], as_index=False)['Invoice'].sum().
      ↪sort_values('Invoice', ascending=False).head(5).reset_index(drop=True))
print()
```

Top 5 Product_Category by Month and total Invoice:

	Month	Product_Category	Invoice
0	Jan	Nest-USA	103309.1541
1	Nov	Nest-USA	91249.8100
2	Dec	Nest-USA	82770.5110
3	Jul	Nest-USA	77164.6700
4	Oct	Nest-USA	76008.7300

```
[60]: for col in cat_col2:
      print(f'MODE Product_SKU and Product_Description by {col} :')
      print(df.groupby(col, as_index=False)[cat_col1].agg(lambda x: pd.Series.
      ↪mode(x)[0]))
      print()
```

MODE Product_SKU and Product_Description by Gender :

	Gender	Product_SKU	Product_Description
0	F	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen
1	M	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen

MODE Product_SKU and Product_Description by churn :

	churn	Product_SKU	Product_Description
0	0	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen

1 1 GGOENEBQ078999 Nest Learning Thermostat 3rd Gen

MODE Product_SKU and Product_Description by Tenurebin :

	Tenurebin	Product_SKU	Product_Description
0	0-10	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen
1	10-20	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen
2	20-30	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen
3	30-40	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen
4	>40	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen

MODE Product_SKU and Product_Description by rfm_segment :

	rfm_segment	Product_SKU \
0	About to Sleep	GGOENEBJ079499
1	At Risk	GGOENEBJ079499
2	Cant Lose Them	GGOENEBJ079499
3	Champions	GGOENEBJ079499
4	Customers Needing Attention	GGOENEBQ078999
5	Hibernating	GGOENEBB078899
6	Lost	GGOEGBJC019999
7	Loyal Customers	GGOENEBQ078999
8	Potential Loyalists	GGOENEBJ079499
9	Promising	GGOENEBJ079499
10	Recent Customers	GGOENEBQ078999

	Product_Description
0	Android Toddler Short Sleeve T-s
1	Nest Learning Thermostat 3rd Gen
2	Nest Learning Thermostat 3rd Gen
3	Nest Learning Thermostat 3rd Gen
4	Nest Learning Thermostat 3rd Gen
5	Nest Cam Indoor Security Camera
6	Google Sunglasses
7	Nest Learning Thermostat 3rd Gen
8	Nest Learning Thermostat 3rd Gen
9	Nest Learning Thermostat 3rd Gen
10	Nest Learning Thermostat 3rd Gen

MODE Product_SKU and Product_Description by Location :

	Location	Product_SKU	Product_Description
0	California	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen
1	Chicago	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen
2	New Jersey	GGOENEBB078899	Nest Learning Thermostat 3rd Gen
3	New York	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen
4	Washington DC	GGOENEBQ078999	Nest Learning Thermostat 3rd Gen

MODE Product_SKU and Product_Description by Coupon_Code :

	Coupon_Code	Product_SKU	Product_Description
0	ACC10	GGOEGCKQ084999	Emoji Sticker Sheet

1	ACC20	GGOEAFKA087499	Android Small Removable Sticker
2	ACC30	GGOEGFKA086699	Google Emoji Sticker Pack
3	AIO10	GGOEGBMJ013399	Sport Bag
4	AIO20	GGOEGBMJ013399	Sport Bag
5	AIO30	GGOEGBMJ013399	Sport Bag
6	AND10	GGOEAAAH083314	Android Men's Paradise Short Sle
7	AND20	GGOEAAAH083313	Android Men's Paradise Short Sle
8	AND30	GGOEAAAH083315	Android Men's Paradise Short Sle
9	BT10	GGOEYDHJ056099	22 oz YouTube Bottle Infuser
10	BT20	GGOEADHH055999	22 oz Android Bottle
11	BT30	GGOEADHH055999	22 oz Android Bottle
12	ELEC10	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen
13	ELEC20	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen
14	ELEC30	GGOENEBJ079499	Nest Learning Thermostat 3rd Gen
15	EXTRA10	GGOEGDHC018299	Google Sunglasses
16	EXTRA20	GGOEGDHC018299	Google Sunglasses
17	EXTRA30	GGOEGDHC018299	Google Sunglasses
18	GC10	GGOEGGCX056399	Gift Card - \$250.00
19	GC20	GGOEGGCX056299	Gift Card - \$25.00
20	GC30	GGOEGGCX056299	Gift Card - \$25.00
21	HGEAR10	GGOEGHPJ080310	Google Blackout Cap
22	HGEAR20	GGOEGHPJ080310	Google Blackout Cap
23	HGEAR30	GGOEGHPJ080310	Google Blackout Cap
24	HOU10	GGOEGCBQ016499	SPF-15 Slim & Slender Lip Balm
25	HOU20	GGOEGCBQ016499	SPF-15 Slim & Slender Lip Balm
26	HOU30	GGOEGCBQ016499	SPF-15 Slim & Slender Lip Balm
27	NCA10	GGOENEBJ081899	Nest Learning Thermostat 3rd Gen
28	NCA20	GGOENEBJ081899	Nest Learning Thermostat 3rd Gen
29	NCA30	GGOENEBJ081899	Nest Learning Thermostat 3rd Gen
30	NE10	GGOENEBQ086799	Nest Thermostat E - USA
31	NE20	GGOENEBQ086799	Nest Thermostat E - USA
32	NE30	GGOENEBQ086799	Nest Thermostat E - USA
33	NJ10	GGOEGOCC077299	Google RFID Journal
34	NJ20	GGOEGOCC077299	Google RFID Journal
35	NJ30	GGOEGOCL077699	Google Hard Cover Journal
36	No_coupon	GGOEGOBC078699	Google Luggage Tag
37	OFF10	GGOEGFKQ020399	Google Laptop and Cell Phone Sti
38	OFF20	GGOEGFKQ020399	Google Laptop and Cell Phone Sti
39	OFF30	GGOEGFKQ020399	Google Laptop and Cell Phone Sti
40	SALE10	GGOEGHPB071610	Google Men's 100% Cotton Short S
41	SALE20	GGOEGHPB071610	Google Men's 100% Cotton Short S
42	SALE30	GGOEGHPB071610	Google Men's 100% Cotton Short S
43	WEMP10	GGOEWEBB082699	Waze Mobile Phone Vent Mount
44	WEMP20	GGOEWEBB082699	Waze Mobile Phone Vent Mount
45	WEMP30	GGOEWEBB082699	Waze Mobile Phone Vent Mount

11.1 GGOENEBJ079499 is the most popular.

11.2 For Coupon_Code there are variety of most frequent products based on the specific code.

12 Hypothesis Testing

12.0.1 Significance level (alpha) is set to .05 if not mentioned otherwise.

12.0.2 Independent ttest to check difference in mean invoice across Gender and churn

H0 : The mean Invoice among the 2 subgroups of each category is same.

H1 : The mean Invoice among the 2 subgroups of each category is significantly difference.

Significance level(alpha) is set to .05 .

```
[61]: M,f=df[df['Gender']=='M']['Invoice'],df[df['Gender']=='F']['Invoice']
ttest_gender = ttest_ind(f,M)
if ttest_gender.pvalue <= 0.05:
    print(f"There is a statistically significant difference in mean invoice_
    ↳between genders. pvalue : {ttest_gender.pvalue}")
else:
    print(f"There is NO statistically significant difference in mean invoice_
    ↳between genders. pvalue : {ttest_gender.pvalue}")

Nc,c=df[df['churn']==0]['Invoice'],df[df['churn']==1]['Invoice']
ttest_churn = ttest_ind(Nc,c)
if ttest_churn.pvalue <= 0.05:
    print(f"There is a statistically significant difference in mean invoice_
    ↳between churned and non-churned customers. pvalue : {ttest_churn.pvalue}")
else:
    print(f"There is NO statistically significant difference in mean invoice_
    ↳between churned and non-churned customers. pvalue : {ttest_churn.pvalue}")
```

There is NO statistically significant difference in mean invoice between genders. pvalue : 0.2813480064152183

There is a statistically significant difference in mean invoice between churned and non-churned customers. pvalue : 4.8909588067553136e-11

12.0.3 ANOVA and Kruskal-Walis for Tenurebin and rf_m_segment and Location and Coupon_Code.

H0 : The mean Invoice among the subgroups of each category is same.

H1 : The mean Invoice among the subgroups of each category is significantly difference.

Significance level(alpha) is set to .05 .

```
[62]: pg.normality(df['Invoice'], method='shapiro')
```

```
[62]:           W    pval  normal
Invoice  0.276398    0.0   False
```

Tenurebin

```
[63]: pg.homoscedasticity(df, dv='Invoice', group='Tenurebin')
```

```
[63]:           W    pval  equal_var
levene  0.60963  0.655679        True
```

```
[64]: pg.anova(data=df, dv='Invoice', between='Tenurebin')
```

```
[64]:      Source  ddof1  ddof2      F    p-unc    np2
0  Tenurebin      4  52919  0.64025  0.63375  0.000048
```

```
[65]: pg.kruskal(data=df, dv='Invoice', between='Tenurebin')
```

```
[65]:      Source  ddof1      H    p-unc
Kruskal  Tenurebin      4  31.071518  0.000003
```

rfm_segment

```
[66]: pg.homoscedasticity(df, dv='Invoice', group='rfm_segment')
```

```
[66]:           W    pval  equal_var
levene  14.702842  1.667344e-26   False
```

```
[67]: pg.anova(data=df, dv='Invoice', between='rfm_segment')
```

```
[67]:      Source  ddof1  ddof2      F    p-unc    np2
0  rfm_segment    10  52913  15.624663  2.137679e-28  0.002944
```

```
[68]: pg.kruskal(data=df, dv='Invoice', between='rfm_segment')
```

```
[68]:      Source  ddof1      H    p-unc
Kruskal  rfm_segment    10  1811.366036    0.0
```

Location

```
[69]: pg.homoscedasticity(df, dv='Invoice', group='Location')
```

```
[69]:           W    pval  equal_var
levene  0.308458  0.872496        True
```

```
[70]: pg.anova(data=df, dv='Invoice', between='Location')
```

```
[70]:      Source  ddof1  ddof2      F    p-unc    np2
0  Location      4  52919  0.294788  0.881518  0.000022
```

```
[71]: pg.kruskal(data=df, dv='Invoice', between='Location')
```

```
[71]:
```

	Source	ddof1	H	p-unc
Kruskal	Location	4	7.535014	0.110175

Coupon_Code

```
[72]: pg.homoscedasticity(df, dv='Invoice', group='Coupon_Code')
```

```
[72]:
```

	W	pval	equal_var
levene	46.232491	0.0	False

```
[73]: pg.anova(data=df, dv='Invoice', between='Coupon_Code')
```

```
[73]:
```

	Source	ddof1	ddof2	F	p-unc	np2
0	Coupon_Code	45	52878	46.106051	0.0	0.037756

```
[74]: pg.kruskal(data=df, dv='Invoice', between='Coupon_Code')
```

```
[74]:
```

	Source	ddof1	H	p-unc
Kruskal	Coupon_Code	45	967.448797	1.917051e-173

12.0.4 Statistical Test Results:

1. Gender Invoice Comparison:

- There is NO statistically significant difference in mean invoice between genders (p-value: 0.281).

2. Churn Invoice Comparison:

- There is a statistically significant difference in mean invoice between churned and non-churned customers (p-value: 4.89e-11).

3. Assessment of Normality:

- Invoice data is not normally distributed.

4. Tenurebin Kruskal-Wallis Test:

- Levene's test indicates homogeneity of variance (p-value: 0.61).
- Kruskal results suggest a statistically significant difference in mean invoice across tenure bins (p-value: 3e-6).

5. rfm_segment Kruskal-Wallis Test:

- Levene's test indicates heterogeneity of variance (p-value: <0.05).
- Kruskal results suggest a statistically significant difference in mean invoice across RFM segments (p-value: 0.0).

6. Location Kruskal-Wallis Test:

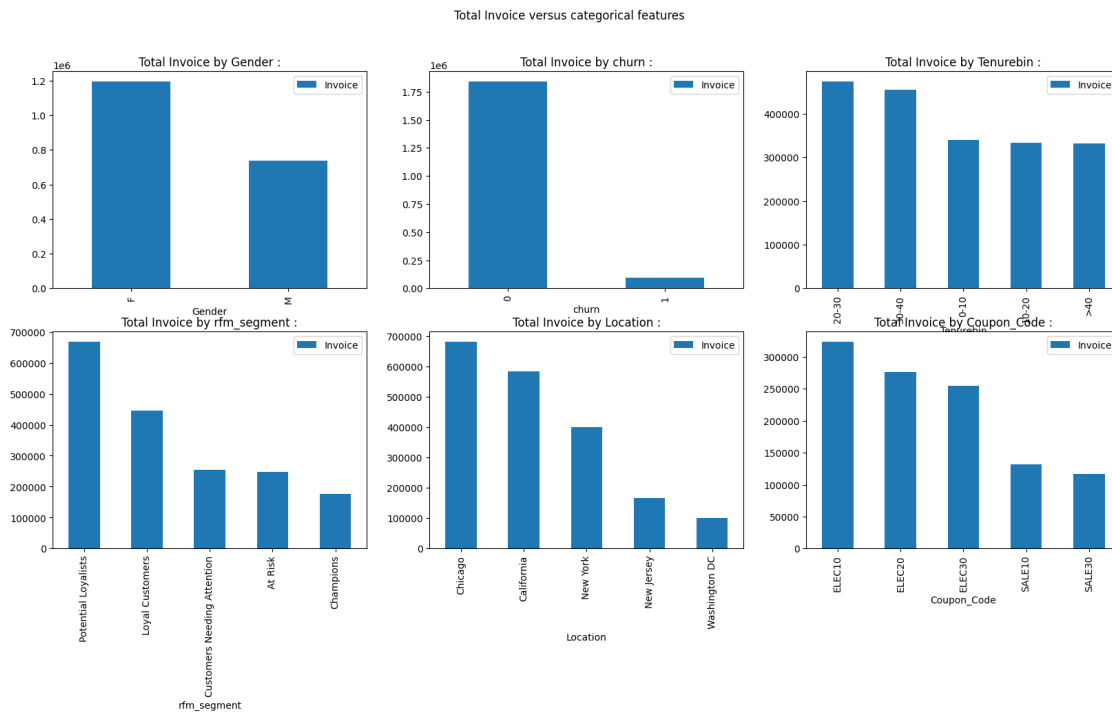
- Levene's test indicates homogeneity of variance (p-value: 0.31).
- Kruskal-Wallis results suggest no statistically significant difference in mean invoice across locations (p-value: 0.11).

7. Coupon_Code Kruskal-Wallis Test:

- Levene's test indicates heterogeneity of variance (p-value: 0.0).
- Kruskal-Wallis results suggest statistically significant difference in mean invoice across coupon codes. (p-value: 1.92e-173).

These results provide insights into the differences in mean invoice across different groups, as well as the normality and variance assumptions of the tests performed.

```
[75]: fig,axes=plt.subplots(2,3,figsize=(20,9))
      ax=axes.flatten()
      Store=[]
      plt.suptitle('Total Invoice versus categorical features')
      for i,col in enumerate(cat_col2):
          group=df.groupby(col,as_index=False)['Invoice'].sum().
          ↪sort_values('Invoice',ascending=False).head(5)
          Store.append(group)
          ax[i].set_title(f'Total Invoice by {col} :')
          group.head(5).plot(kind='bar',x=col,y='Invoice',ax=ax[i])
      plt.show()
```



```
[76]: for i in Store:
      print(i)
      print()
```

	Gender	Invoice
0	F	1.193025e+06
1	M	7.389675e+05

	churn	Invoice
0	0	1.837792e+06

1 1 9.420123e+04

	Tenurebin	Invoice
2	20-30	472881.24307
3	30-40	454830.86494
0	0-10	340153.12611
1	10-20	332629.13539
4	>40	331498.55667

	rfm_segment	Invoice
8	Potential Loyalists	666863.19447
7	Loyal Customers	444621.39499
4	Customers Needing Attention	253967.35675
1	At Risk	246907.04605
3	Champions	176769.01709

	Location	Invoice
1	Chicago	679791.55891
0	California	584489.25898
3	New York	400631.41154
2	New Jersey	166720.07400
4	Washington DC	100360.62275

	Coupon_Code	Invoice
12	ELEC10	323126.20410
13	ELEC20	275706.28000
14	ELEC30	254812.52100
40	SALE10	132244.53118
42	SALE30	116555.15028

12.0.5 Gender Invoice Insights:

- Female customers have a higher total invoice amount (\$1,193,025) compared to male customers (\$738,967.50).

12.0.6 Churn Invoice Insights:

- Customers who did not churn have a significantly higher total invoice amount (\$1,837,792) compared to churned customers (\$94,201.23).

12.0.7 Tenurebin Invoice Insights:

- Customers in the 20-30 tenure months category have the highest total invoice amount (\$472,881.24), followed by customers in the 30-40 tenure months category (\$454,830.86).

12.0.8 RFM Segment Invoice Insights:

- Potential Loyalists, identified as a valuable segment, have the highest total invoice amount (\$666,863.19), while the Lost segment has the lowest total invoice amount (\$1,905.32).

12.0.9 Location Invoice Insights:

- Transactions from Chicago contribute the highest total invoice amount (\$679,791.56), followed by California (\$584,489.26), and New York (\$400,631.41).

12.0.10 Coupon_Code Insights:

- ELEC 10,20,30 and SALE 10,20,30 contributes the most to total Invoice.

13 Churn Analysis

13.1 Q. Is there significant relationship between categorical columns and churn?

13.1.1 Applying chisquare test of independence with significance value alpha set to .05.

H0 : The categorical column and churn is not dependent on each other.

H1 : There is significant dependence of churn on the categorical column.

Significance level(alpha) is set to .05

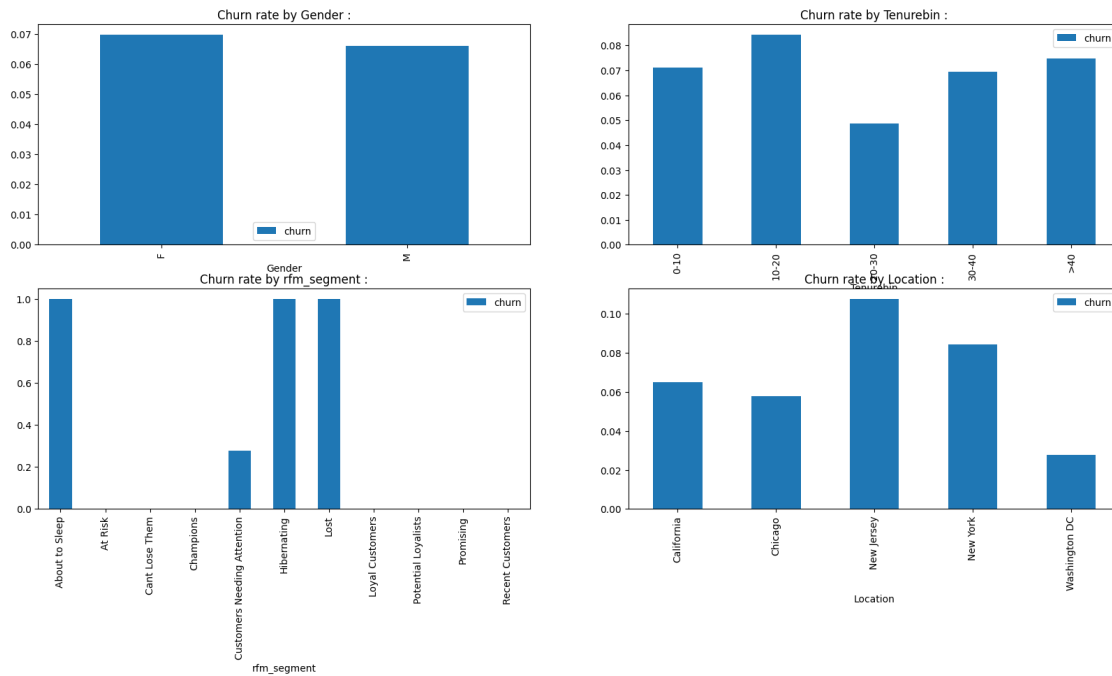
```
[77]: categorical=['Gender', 'Tenurebin', 'rfm_segment', 'Location' ]
for col in categorical:
    ct=pd.crosstab(df[col],df['churn'])
    pval=chi2_contingency(ct).pvalue
    if pval<=.05:
        print(f'As pvalue({pval})<=alpha(.05) we reject null hypothesis, churn
↪is significantly dependent on {col}')
```

```
As pvalue(0.11502249180753568)>alpha(.05) we fail to reject null hypothesis,
churn is NOT significantly dependent on Gender
As pvalue(4.341700473181041e-25)<=alpha(.05) we reject null hypothesis, churn
is significantly dependent on Tenurebin
As pvalue(0.0)<=alpha(.05) we reject null hypothesis, churn is significantly
dependent on rfm_segment
As pvalue(8.669623094961317e-55)<=alpha(.05) we reject null hypothesis, churn
is significantly dependent on Location
```

```
[78]: fig,axes=plt.subplots(2,2,figsize=(20,9))
ax=axes.flatten()
Store=[]
```

```
plt.suptitle('Churn rate across categorical features')
for i,col in enumerate(categorical):
    group=df.groupby(col,as_index=False)['churn'].mean()
    Store.append(group)
    ax[i].set_title(f'Churn rate by {col} :')
    group.plot(kind='bar',x=col,y='churn',ax=ax[i])
plt.show()
```

Churn rate across categorical features



```
[79]: for i in Store:
        print(i)
        print()
```

	Gender	churn
0	F	0.069682
1	M	0.066074

	Tenurebin	churn
0	0-10	0.071057
1	10-20	0.084206
2	20-30	0.048697
3	30-40	0.069528
4	>40	0.074701

	rfm_segment	churn
0	About to Sleep	1.0
1	At Risk	0.0
2	Can't Lose Them	0.0
3	Champions	0.0
4	Customers Needing Attention	0.28
5	Hibernating	1.0
6	Lost	1.0
7	Loyal Customers	0.0
8	Potential Loyalists	0.0
9	Promising	0.0
10	Recent Customers	0.0

0	About to Sleep	1.000000
1	At Risk	0.000000
2	Cant Lose Them	0.000000
3	Champions	0.000000
4	Customers Needing Attention	0.279327
5	Hibernating	1.000000
6	Lost	1.000000
7	Loyal Customers	0.000000
8	Potential Loyalists	0.000000
9	Promising	0.000000
10	Recent Customers	0.000000

	Location	churn
0	California	0.065072
1	Chicago	0.057835
2	New Jersey	0.107484
3	New York	0.084400
4	Washington DC	0.027818

13.1.2 Churn Dependence Insights:

1. Gender:

- Churn is NOT significantly dependent on gender (p-value: 0.115).

2. Tenurebin:

- Churn is significantly dependent on tenurebin (p-value: 4.34e-25).
- Customers with tenure between 20-30 months have the lowest churn rate (4.87%), while those with tenure between 10-20 months have the highest churn rate (8.42%).

3. RFM Segment:

- Churn is significantly dependent on RFM segment (p-value: 0.0).
- Customers categorized as 'At Risk', 'Cant Lose Them', 'Champions', 'Loyal Customers', and 'Potential Loyalists' have the lowest churn rates (0.0%), indicating high loyalty.

4. Location:

- Churn is significantly dependent on location (p-value: 8.67e-55).
- Customers from Washington DC exhibit the lowest churn rate (2.78%), while those from New Jersey have the highest churn rate (10.75%).

These insights highlight the factors influencing churn rates, including tenure, RFM segment, and location. Understanding these dependencies can help in devising targeted retention strategies and improving customer loyalty.

13.1.3 Crosschecking by checking if mean Invoice and mean Tenure is significantly different for churn and not churn.

H0 : Not churned customers have mean invoice less than or equal to that of churned customer.

H1 : Not churned customers have mean invoice greater than that of churned customer.

Significance level(α)=.05

```
[80]: C,Nc=df[df['churn']==1]['Invoice'],df[df['churn']==0]['Invoice']  
      levene(Nc,C)
```

```
[80]: LeveneResult(statistic=42.93837475167171, pvalue=5.70083792022353e-11)
```

As Levene test $pvalue < .05$ equal_var is set to False

```
[81]: ttest_ind(Nc,C,alternative='greater',equal_var=False)
```

```
[81]: TtestResult(statistic=10.502516703639804, pvalue=7.411181740803589e-26,  
df=5391.8653543032015)
```

13.1.4 As $pvalue < .05$ we reject null hypothesis and can conclude that not churned customers have higher mean Invoice value which is expected by definition.

H0 : Not churned customers have mean tenure greater than or equal to that of churned customer.

H1 : Not churned customers have mean tenure less than that of churned customer.

Significance level(α) is set to .05.

```
[82]: C,Nc=df[df['churn']==1]['Tenure_Months'],df[df['churn']==0]['Tenure_Months']  
      levene(Nc,C)
```

```
[82]: LeveneResult(statistic=58.40757862523135, pvalue=2.166461537104734e-14)
```

As Levene test $pvalue < .05$ equal_var is set to False

```
[83]: ttest_ind(Nc,C,alternative='less',equal_var=False)
```

```
[83]: TtestResult(statistic=0.2037658181614431, pvalue=0.580726699025027,  
df=4106.604928531195)
```

13.1.5 As $pvalue > .05$ we fail to reject null hypothesis and cannot conclude that not churned customers have lower mean Tenure_Months value.

- **Invoice Value Analysis:**
 - The statistical test indicates that non-churned customers have a significantly higher mean invoice value compared to churned customers ($p < 0.05$). This aligns with expectations, as loyal customers tend to make more larger purchases over time.
- **Tenure_Months Analysis:**
 - The analysis reveals that non-churned customers do not have lower mean tenure value.

14 Customer Lifetime Value (CLTV)

14.0.1 Feature Engineering

```
[84]: df.Coupon=df.Discount_pct*df.Coupon

#Encoding
encoder = TargetEncoder()
df['Location_enc'] = encoder.fit_transform(df['Location'], df['Invoice'])

#Grouping
customer_df = df.groupby('CustomerID').agg({'Invoice': 'sum', 'Transaction_ID': 'nunique', 'Location_enc': 'mean',
      'Quantity': 'median', 'Tenure_Months': 'median', 'Coupon': 'mean', 'churn': 'mean',
}) .reset_index().rename(columns={'Transaction_ID': 'Total_Transactions'})

#Scaling
columns_to_normalize = ['Total_Transactions', 'Quantity', 'Tenure_Months', 'Coupon', 'churn', 'Location_enc']
scaler = MinMaxScaler()
customer_df[columns_to_normalize] = scaler.fit_transform(customer_df[columns_to_normalize])
```

14.0.2 Splitting and Tuning and Stacking

```
[85]: # Data preparation
X = customer_df[['Total_Transactions', 'Quantity', 'Tenure_Months', 'Coupon', 'churn', 'Location_enc']]
y = customer_df['Invoice']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=95)

# Linear Regression
param_grid_lr = {'fit_intercept': [True, False]}
model_lr = LinearRegression()
grid_search_lr = GridSearchCV(estimator=model_lr, param_grid=param_grid_lr, cv=3, scoring='r2', n_jobs=-1)
grid_search_lr.fit(X_train, y_train)
best_model_lr = grid_search_lr.best_estimator_

# Lasso
param_grid_lasso = {'alpha': [0.001, 0.01, 0.1, 1, 10, 100]}
model_lasso = Lasso(random_state=95)
grid_search_lasso = GridSearchCV(estimator=model_lasso, param_grid=param_grid_lasso, cv=3, scoring='r2', n_jobs=-1)
grid_search_lasso.fit(X_train, y_train)
```

```

best_model_lasso = grid_search_lasso.best_estimator_

# Ridge
param_grid_ridge = {'alpha': [0.001, 0.01, 0.1, 1, 10, 100]}
model_ridge = Ridge(random_state=95)
grid_search_ridge = GridSearchCV(estimator=model_ridge,
    ↪param_grid=param_grid_ridge, cv=3, scoring='r2', n_jobs=-1)
grid_search_ridge.fit(X_train, y_train)
best_model_ridge = grid_search_ridge.best_estimator_

# Stacking
stacked_estimators = [
    ('linear', best_model_lr),
    ('lasso', best_model_lasso),
    ('ridge', best_model_ridge)]
stacked_model =
    ↪StackingRegressor(estimators=stacked_estimators, final_estimator=LinearRegression())
stacked_model.fit(X_train, y_train)

```

```

[85]: StackingRegressor(estimators=[('linear', LinearRegression(fit_intercept=False)),
    ('lasso', Lasso(alpha=1, random_state=95)),
    ('ridge', Ridge(alpha=0.1, random_state=95))],
    final_estimator=LinearRegression())

```

14.0.3 Evaluation

```

[86]: # Evaluating the stacked model
y_pred_stacked = stacked_model.predict(X_test)

rmse_stacked = np.sqrt(MSE(y_test, y_pred_stacked))
r2_stacked = stacked_model.score(X_test, y_test)

print(f"Stacked Model RMSE: {rmse_stacked}")
print(f"Stacked Model R2 score: {r2_stacked}")

```

Stacked Model RMSE: 744.0948124033317

Stacked Model R² score: 0.8469546623725761

14.1 Through stacking and hyperparameter tuning a regression model is built with decent .85 r^2 value and 744 RMSE which predicts total revenue that a customer generates based on the features Location, count of transactions, median Quantity bought, Tenure, median Coupon discount availed and churn.

15 Recommendations Based on Insights

1. **Targeted Marketing for Top Products:** Focus marketing campaigns on top-performing products such as the Nest Learning Thermostat 3rd Gen-USA and Nest Cam Outdoor Security

Camera. Highlight their features and benefits to capitalize on their high demand.

2. **Leverage Peak Sales Months:** Increase promotional activities and special offers during January, November, and August, as these months show the highest total invoice amounts. Utilize events like New Year sales, Black Friday, and back-to-school promotions to maximize revenue.
3. **Optimize Coupon Strategies:** Promote and potentially expand successful coupon codes like ELEC10, ELEC20, and ELEC30. These codes drive significant sales volume and should be a focal point in discount strategies.
4. **Enhance Customer Retention Programs:** Develop loyalty programs targeting customers with tenure between 20-30 months, who exhibit the lowest churn rates and maximum revenue. Personalized offers and engagement strategies can help maintain their loyalty and reduce churn.
5. **Address High Churn Regions:** Implement targeted retention strategies for regions with high churn rates, particularly New Jersey. Tailor marketing efforts and customer service improvements to address specific needs and reduce churn in these areas.
6. **Promote Product Bundles:** Highlight product combinations that show significant associations, such as the Nest Cam Indoor and Outdoor Security Cameras. Cross-sell these products to customers to increase average transaction values.
7. **Improve Customer Experience for High-Value Segments:** Focus on enhancing the customer experience for high-value RFM segments like Loyal Customers and Potential Loyalists. Provide exclusive benefits and personalized services to keep them engaged and loyal.
8. **Expand Successful Product Categories:** Increase the variety and visibility of high-demand categories such as Nest-USA, Apparel and Office supplies. Tailor marketing campaigns to showcase the range and quality of products in these categories.
9. **Monitor Seasonal Spending Patterns:** Develop strategies and discounts to boost sales during typically lower profit periods such as from Q1-Q2 and Q3-Q4. Also utilize high profit during Q2-Q3 by further increasing sales volume through discounts and other strategies.
10. **Leverage High-Retention Cohorts:** Focus retention efforts on high-performing cohorts like '2019-01' and '2019-02'. Analyze what contributed to their higher retention rates and replicate successful strategies across other cohorts.
11. **Utilize CLTV predictions:** Use the predictive model's CLTV estimates to prioritize retention efforts, personalize marketing strategies, and optimize resource allocation for maximum long-term profitability.
12. **Target High Revenue Segments for Enhanced Profitability:** Focus retention efforts on female customers and those in the Potential Loyalists segment. Prioritize high-invoice regions like Chicago, California, and New York.

[]: