CRM analysis

Customer Relationship Management (CRM) analysis involves the systematic examination and interpretation of data related to interactions between a business and its customers. Through CRM analysis, companies evaluate customer behavior, preferences, and feedback to gain valuable insights into their needs and expectations.

Importing required Libraries

In [ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import binom, norm, ttest_1samp, ttest_ind, ttest_rel, chi2, chisquare, chi2_contingency,stats
from scipy.stats import f_oneway, kruskal, levene, shapiro, pearsonr, spearmanr
from statsmodels.graphics.gofplots import qqplot
import statsmodels.api as sm
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:

```
df = pd.read_csv('/content/Ecom_CRM_analysis.csv',encoding='ISO-8859-1')
```

In [ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  object
 5   UnitPrice    541909 non-null  float64
```

```
 6   CustomerID    406829 non-null  float64
 7   Country       541909 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

Handling Null Values

In [ ]:

```
df.isna().sum()
```

Out[ ]:

	0
InvoiceNo	0
StockCode	0
Description	1454
Quantity	0
InvoiceDate	0
UnitPrice	0
CustomerID	135080
Country	0

dtype: int64

In [ ]:

```
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
df['Description'].fillna('Unknown',inplace=True)
df['CustomerID'].fillna('Unknown_Customer',inplace=True)
df.isna().sum()
```

Out[ ]:

	0
InvoiceNo	0
StockCode	0

Description	0
Quantity	0
InvoiceDate	0
UnitPrice	0
CustomerID	0
Country	0

dtype: int64

In [ ]:

```
df.head()
```

Out[ ]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

In [ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  541909 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  datetime64[ns]
```

```
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   541909 non-null  object
 7   Country      541909 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(1), object(5)
memory usage: 33.1+ MB
```

Creating Required new Features

In [ ]:

```
df['is_return'] = df['Quantity'] < 0
df['is_free_item'] = df['UnitPrice'] == 0
df['is_canceled'] = df['InvoiceNo'].str.startswith('C')
```

In [ ]:

```
df.head()
```

Out[ ]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	is_return	is_free_item	is_canceled
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	False	False	False
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	False	False	False
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	False	False	False
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	False	False	False
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	False	False	False

In [ ]:

```
df_filtered = df[~df['is_return'] & ~df['is_free_item'] & ~df['is_canceled']]
df_filtered.describe(include='all')
```

Out[ ]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	is_return	is_free_item	is_canceled

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	is_return	is_free_item	is_canceled
count	530106	530106	530106	530106.000000	530106	530106.000000	530106	530106	530106	530106	530106
unique	19962	3922	4026	NaN	NaN	NaN	4339	38	1	1	1
top	573585	85123A	WHITE HANGING HEART T-LIGHT HOLDER	NaN	NaN	NaN	Unknown_Customer	United Kingdom	False	False	False
freq	1114	2265	2323	NaN	NaN	NaN	132222	485125	530106	530106	530106
mean	NaN	NaN	NaN	10.542001	2011-07-04 20:16:17.864539392	3.865875	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	1.000000	2010-12-01 08:26:00	-11062.060000	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	1.000000	2011-03-28 12:22:00	1.250000	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	3.000000	2011-07-20 12:58:00	2.080000	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	10.000000	2011-10-19 12:39:00	4.130000	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	80995.000000	2011-12-09 12:50:00	13541.330000	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	155.523831	NaN	41.856120	NaN	NaN	NaN	NaN	NaN

In [ ]:

```
df['Year'] = df['InvoiceDate'].dt.year
df['Month'] = df['InvoiceDate'].dt.month
df['DayOfWeek'] = df['InvoiceDate'].dt.dayofweek
df['Hour'] = df['InvoiceDate'].dt.hour
df['Revenue'] = df['Quantity']*df['UnitPrice']
```

In [ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 16 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   InvoiceNo     541909 non-null  object
```

```
 1   StockCode     541909 non-null  object
 2   Description   541909 non-null  object
 3   Quantity      541909 non-null  int64
 4   InvoiceDate   541909 non-null  datetime64[ns]
 5   UnitPrice     541909 non-null  float64
 6   CustomerID    541909 non-null  object
 7   Country       541909 non-null  object
 8   is_return     541909 non-null  bool
 9   is_free_item  541909 non-null  bool
 10  is_canceled   541909 non-null  bool
 11  Year          541909 non-null  int32
 12  Month         541909 non-null  int32
 13  DayOfWeek     541909 non-null  int32
 14  Hour          541909 non-null  int32
 15  Revenue       541909 non-null  float64
dtypes: bool(3), datetime64[ns](1), float64(2), int32(4), int64(1), object(5)
memory usage: 47.0+ MB
```

In [ ]:

```
df[['InvoiceNo','StockCode','CustomerID']].nunique()
```

Out[ ]:

	0
InvoiceNo	25900
StockCode	4070
CustomerID	4373

dtype: int64

In [ ]:

```
df['is_return'].value_counts()
```

Out[ ]:

	count
is_return	
False	531285

	count
True	10624
is_return	

dtype: int64

In [ ]:

```
df['is_canceled'].value_counts()
```

Out[ ]:

	count
is_canceled	
False	532621
True	9288

dtype: int64

In [ ]:

```
df['is_free_item'].value_counts()
```

Out[ ]:

	count
is_free_item	
False	539394
True	2515

dtype: int64

Calculating customer lifetime value

In [ ]:

```
CLV = df.groupby('CustomerID')['Revenue'].sum().reset_index()
CLV.columns = ['CustomerID', 'CLV']
```

```
CLV.sort_values(by='CLV',ascending=False).head(10)
```

Out[ ]:

	CustomerID	CLV
4372	Unknown_Customer	1447682.12
1703	14646.0	279489.02
4233	18102.0	256438.49
3758	17450.0	187482.17
1895	14911.0	132572.62
55	12415.0	123725.45
1345	14156.0	113384.14
3801	17511.0	88125.38
3202	16684.0	65892.08
1005	13694.0	62653.10

In [ ]:

```
Customer_frequency = df.groupby('CustomerID')['InvoiceNo'].nunique().reset_index()
Customer_frequency.columns = ['CustomerID','Frequency']
Customer_frequency.sort_values(by='Frequency',ascending=False).head(10)
```

Out[ ]:

	CustomerID	Frequency
4372	Unknown_Customer	3710
1895	14911.0	248
330	12748.0	224
4042	17841.0	169
1674	14606.0	128
568	13089.0	118
2192	15311.0	118
487	12971.0	89

	CustomerID	Frequency
1615	14527.0	85
803	13408.0	81

Correlation Analyses

In [ ]:

```
corr_matrix = df[['Quantity', 'UnitPrice', 'Revenue', 'Year', 'Month', 'DayOfWeek', 'Hour']].corr()
corr_matrix
```

Out[ ]:

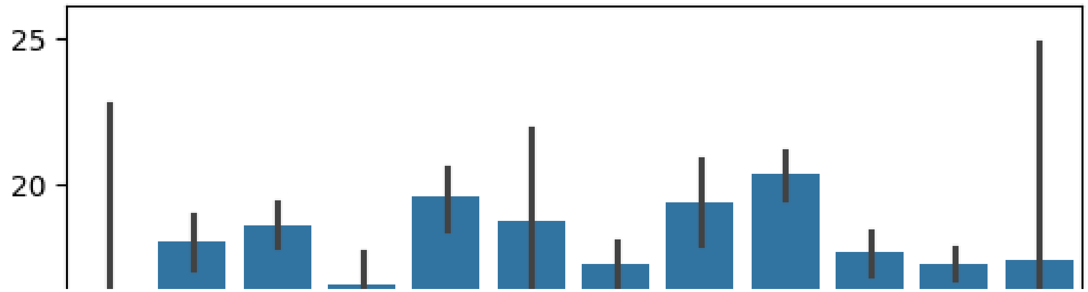
	Quantity	UnitPrice	Revenue	Year	Month	DayOfWeek	Hour
Quantity	1.000000	-0.001235	0.886681	0.002001	-0.001116	-0.000904	-0.011268
UnitPrice	-0.001235	1.000000	-0.162029	-0.004586	-0.000497	-0.007310	0.001268
Revenue	0.886681	-0.162029	1.000000	0.000275	0.000141	-0.002458	-0.009120
Year	0.002001	-0.004586	0.000275	1.000000	-0.369595	-0.007034	-0.010921
Month	-0.001116	-0.000497	0.000141	-0.369595	1.000000	0.040045	0.025649
DayOfWeek	-0.000904	-0.007310	-0.002458	-0.007034	0.040045	1.000000	-0.033176
Hour	-0.011268	0.001268	-0.009120	-0.010921	0.025649	-0.033176	1.000000

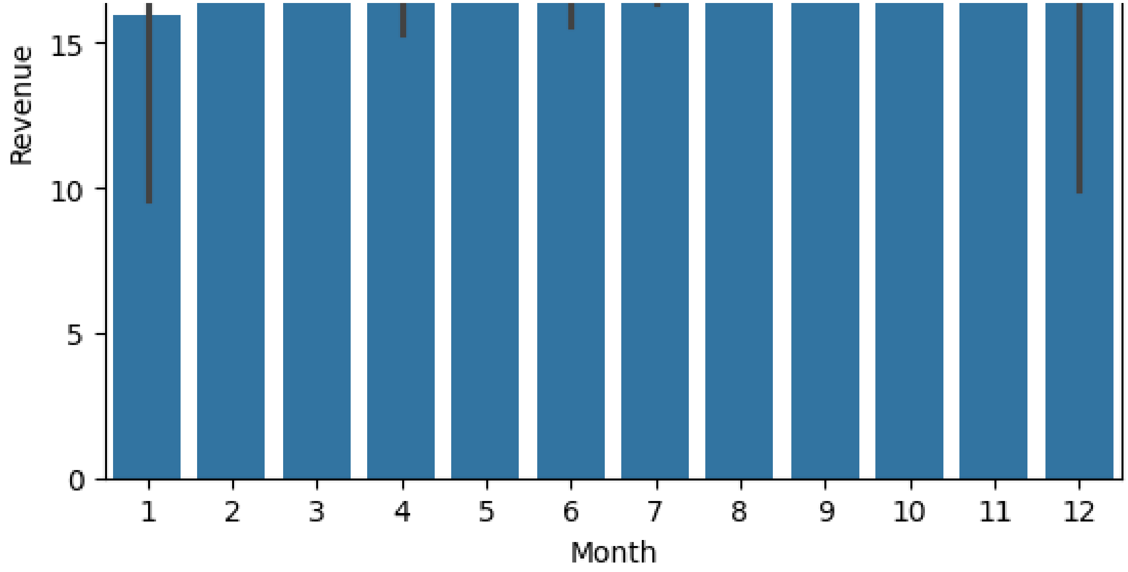
In [ ]:

```
monthly_sales = df.groupby('Month')['Revenue'].sum().reset_index()
sns.barplot(data=df,x='Month',y='Revenue')
```

Out[ ]:

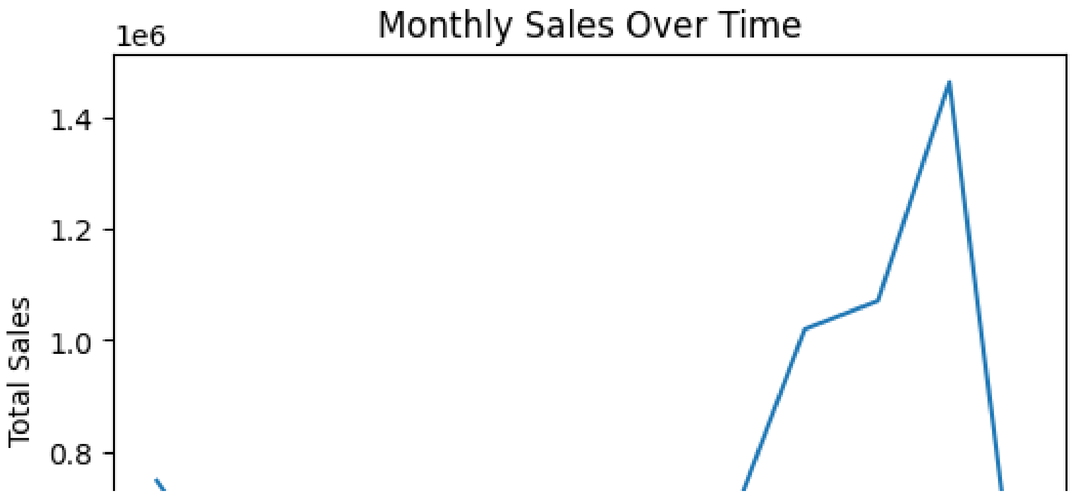
<Axes: xlabel='Month', ylabel='Revenue'>

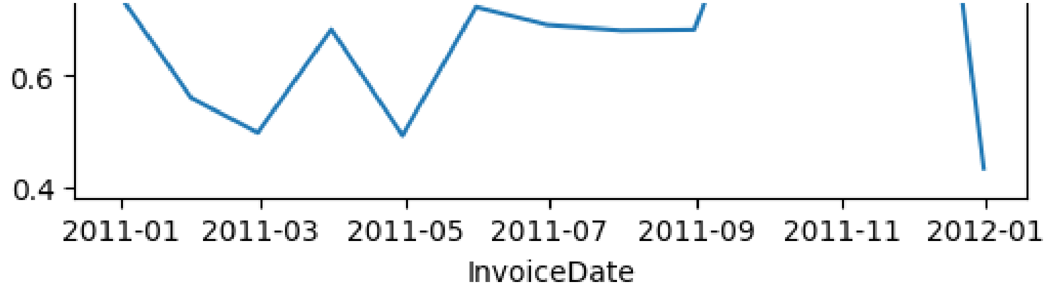




In [ ]:

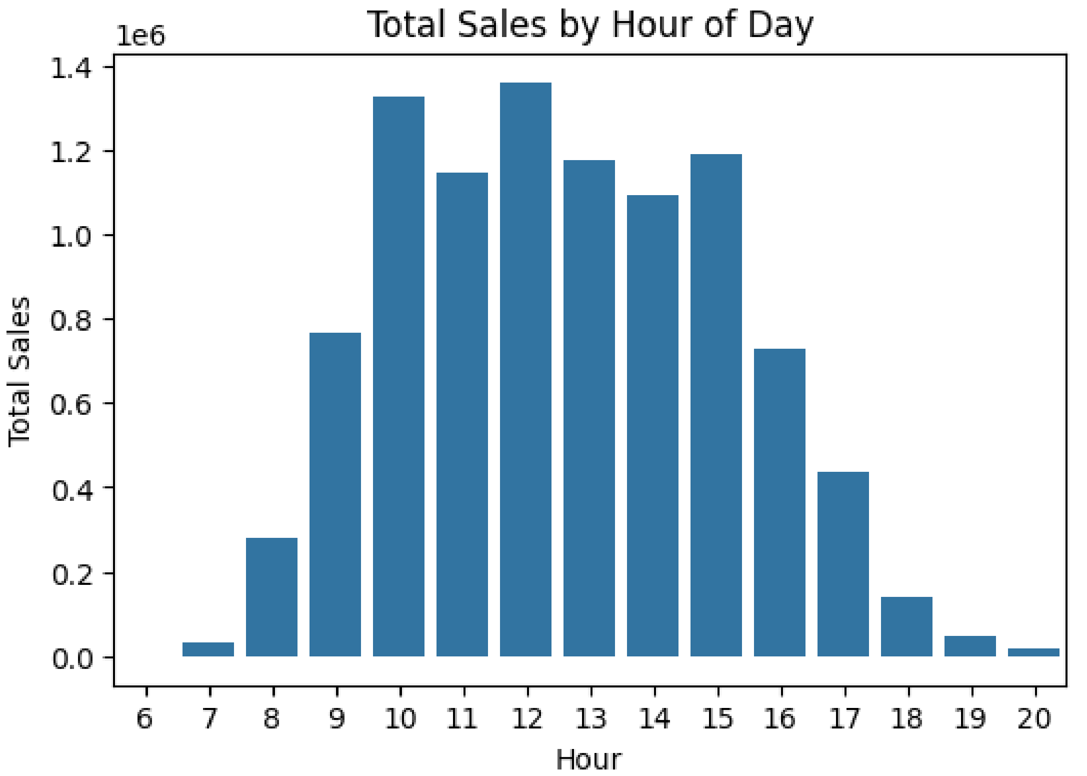
```
df_x = df.copy()
df_x.set_index('InvoiceDate', inplace=True)
Monthly_Sales = df_x['Revenue'].resample('M').sum()

plt.figure(figsize=(6,4))
#Monthly_Sales.plot()
sns.lineplot(data=Monthly_Sales)
plt.title('Monthly Sales Over Time')
plt.ylabel('Total Sales')
plt.show()
```





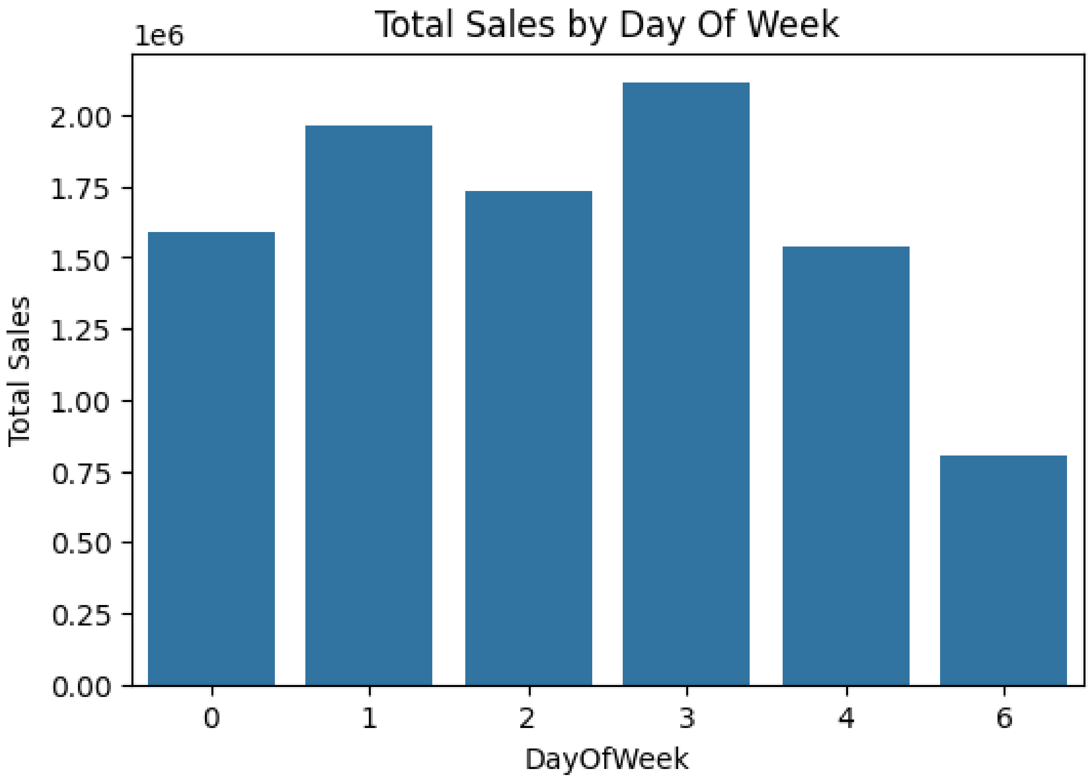
In [ ]:

```
hourly_sales = df.groupby('Hour')['Revenue'].sum()

plt.figure(figsize=(6,4))
sns.barplot(data=hourly_sales)
plt.title('Total Sales by Hour of Day')
plt.ylabel('Total Sales')
plt.show()
```



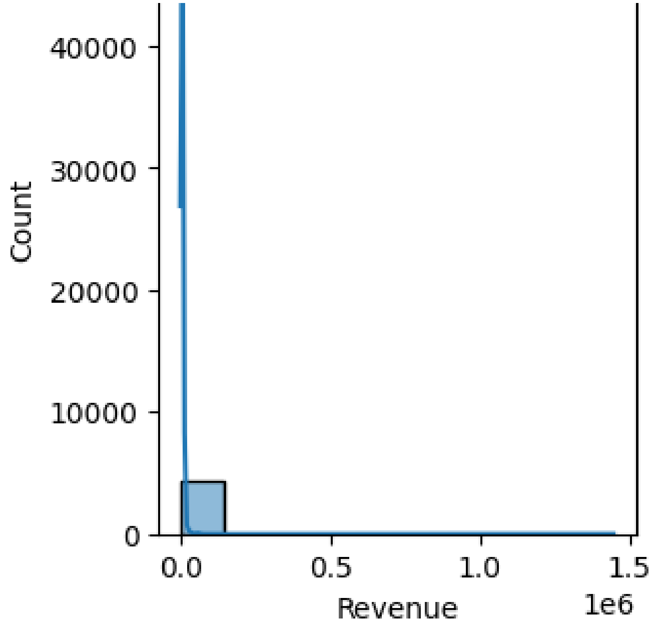
In [ ]:

```
Weekly_Sales = df.groupby('DayOfWeek')['Revenue'].sum()

plt.figure(figsize=(6,4))
sns.barplot(data=Weekly_Sales)
plt.title('Total Sales by Day Of Week')
plt.ylabel('Total Sales')
plt.show()
```



In [ ]:

```
customer_revenue = df.groupby('CustomerID')['Revenue'].sum().reset_index()
plt.figure(figsize=(3,4))
sns.histplot(customer_revenue['Revenue'], bins=10,kde=True)
plt.title('Customer Lifetime Value Distribution')
plt.show()
```



In [ ]:

```
df.groupby('Description')['Revenue'].sum().reset_index().sort_values(by='Revenue',ascending= False)
```

Out[ ]:

	Description	Revenue
1098	DOTCOM POSTAGE	206245.480
2915	REGENCY CAKESTAND 3 TIER	164762.190
3919	WHITE HANGING HEART T-LIGHT HOLDER	99668.470
2471	PARTY BUNTING	98302.980
1866	JUMBO BAG RED RETROSPOT	92356.030
...	...	...
615	Bank Charges	-7175.639
934	CRUK Commission	-7933.430
281	Adjust bad debt	-11062.060
2246	Manual	-68671.640
171	AMAZON FEE	-221520.500

	Description	Revenue

4224 rows × 2 columns

Calculating RFM metrics

In [ ]:

```
reference_date = df['InvoiceDate'].max()+ pd.DateOffset(days=1)
rfm = df.groupby('CustomerID').agg({
    'InvoiceDate': lambda x: (reference_date - x.max()).days,   # Recency
    'InvoiceNo': 'count',   # Frequency
    'Revenue': 'sum'     # Monetary
}).reset_index()
rfm.columns = ['CustomerID', 'Recency', 'Frequency', 'Monetary']
rfm
```

Out[ ]:

	CustomerID	Recency	Frequency	Monetary
0	12346.0	326	2	0.00
1	12347.0	2	182	4310.00
2	12348.0	75	31	1797.24
3	12349.0	19	73	1757.55
4	12350.0	310	17	334.40
...	...	...	...	...
4368	18281.0	181	7	80.82
4369	18282.0	8	13	176.60
4370	18283.0	4	756	2094.88
4371	18287.0	43	70	1837.28
4372	Unknown_Customer	1	135080	1447682.12

4373 rows × 4 columns

In [ ]:

```
rfm['R_Score'] = pd.qcut(rfm['Recency'], 3, labels=[3, 2, 1])
```

```
rfm['F_Score'] = pd.qcut(rfm['Frequency'], 3, labels=[3, 2, 1])
rfm['M_Score'] = pd.qcut(rfm['Monetary'], 3, labels=[3, 2, 1])
rfm['rfm_score'] = rfm['R_Score'].astype(str)+rfm['F_Score'].astype(str)+rfm['M_Score'].astype(str)
def segment_customer(row):
    if row['rfm_score'] == '331':
        return 'Best Customers'
    elif row['rfm_score'] == '321':
        return 'Loyal Customers'
    elif row['rfm_score'].startswith('3'):
        return 'Promising Customers'
    elif row['rfm_score'].startswith('2'):
        return 'At Risk Customers'
    else:
        return 'Lost Customers'

rfm['Segment'] = rfm.apply(segment_customer, axis=1)
rfm
```

Out[ ]:

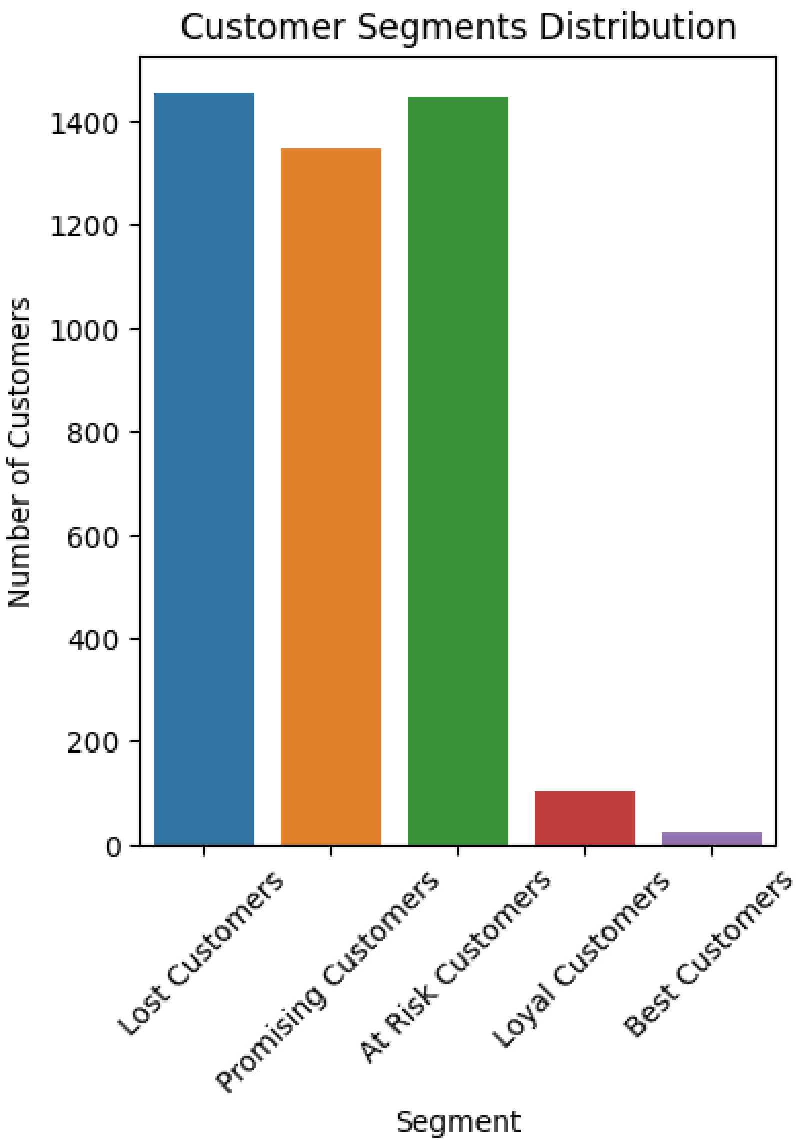
	CustomerID	Recency	Frequency	Monetary	R_Score	F_Score	M_Score	rfm_score	Segment
0	12346.0	326	2	0.00	1	3	3	133	Lost Customers
1	12347.0	2	182	4310.00	3	1	1	311	Promising Customers
2	12348.0	75	31	1797.24	2	2	1	221	At Risk Customers
3	12349.0	19	73	1757.55	3	2	1	321	Loyal Customers
4	12350.0	310	17	334.40	1	3	3	133	Lost Customers
...	...	...	...	...	...	...	...	...	...
4368	18281.0	181	7	80.82	1	3	3	133	Lost Customers
4369	18282.0	8	13	176.60	3	3	3	333	Promising Customers
4370	18283.0	4	756	2094.88	3	1	1	311	Promising Customers
4371	18287.0	43	70	1837.28	2	2	1	221	At Risk Customers
4372	Unknown_Customer	1	135080	1447682.12	3	1	1	311	Promising Customers

4373 rows × 9 columns

In [ ]:

```
plt.figure(figsize=(4,5))
```

```
plt.figure(figsize=(4,5))
sns.countplot(data=rfm,x='Segment',hue='Segment')
plt.title('Customer Segments Distribution')
plt.xticks(rotation=45)
plt.ylabel('Number of Customers')
plt.show()
```



In [ ]:

```
df1 = df.copy()
df1 = df1.sort_values(by=['CustomerID', 'InvoiceDate'])
```

```
df1['DaysSinceLastPurchase'] = df1.groupby('CustomerID')['InvoiceDate'].diff().dt.days
avg_days_between_purchases = df1.groupby('CustomerID')['DaysSinceLastPurchase'].mean().reset_index()
avg_days_between_purchases.columns = ['CustomerID', 'AvgDaysBetweenPurchases']
avg_days_between_purchases
```

Out[ ]:

	CustomerID	AvgDaysBetweenPurchases
0	12346.0	0.000000
1	12347.0	2.000000
2	12348.0	9.400000
3	12349.0	0.000000
4	12350.0	0.000000
...	...	...
4368	18281.0	0.000000
4369	18282.0	9.833333
4370	18283.0	0.433113
4371	18287.0	2.275362
4372	Unknown_Customer	0.000763

4373 rows × 2 columns

In [ ]:

```
customer_preferred_day = df1.groupby('CustomerID')['DayOfWeek'].agg(lambda x: x.mode()[0]).reset_index()
customer_preferred_day.columns = ['CustomerID','Preferred_shopping_Day']
customer_preferred_day
```

Out[ ]:

	CustomerID	Preferred_shopping_Day
0	12346.0	1
1	12347.0	1
2	12348.0	3
3	12349.0	0

	CustomerID	Preferred_shopping_Day
4	12350.0	2
...	...	...
4368	18281.0	6
4369	18282.0	4
4370	18283.0	3
4371	18287.0	2
4372	Unknown_Customer	1

4373 rows × 2 columns

In [ ]:

```
customer_preferred_hour = df1.groupby('CustomerID')['Hour'].agg(lambda x: x.mode()[0]).reset_index()
customer_preferred_hour.columns = ['CustomerID','Preferred_shopping_Hour']
customer_preferred_hour
```

Out[ ]:

	CustomerID	Preferred_shopping_Hour
0	12346.0	10
1	12347.0	14
2	12348.0	19
3	12349.0	9
4	12350.0	16
...	...	...
4368	18281.0	10
4369	18282.0	13
4370	18283.0	14
4371	18287.0	10
4372	Unknown_Customer	15

4373 rows × 2 columns

In [ ]:

```
rfm = pd.merge(rfm, avg_days_between_purchases, on='CustomerID', how='left')
rfm = pd.merge(rfm, customer_preferred_day, on='CustomerID', how='left')
rfm = pd.merge(rfm, customer_preferred_hour, on='CustomerID', how='left')
rfm.head()
```

Out[ ]:

	CustomerID	Recency	Frequency	Monetary	R_Score	F_Score	M_Score	rfm_score	Segment	AvgDaysBetweenPurchases	Preferred_shopping_Day	Preferred_
0	12346.0	326	2	0.00	1	3	3	133	Lost Customers	0.0	1	
1	12347.0	2	182	4310.00	3	1	1	311	Promising Customers	2.0	1	
2	12348.0	75	31	1797.24	2	2	1	221	At Risk Customers	9.4	3	
3	12349.0	19	73	1757.55	3	2	1	321	Loyal Customers	0.0	0	
4	12350.0	310	17	334.40	1	3	3	133	Lost Customers	0.0	2	

Relation between average days between purchases and monetary value

In [ ]:

```
rfm_clean = rfm.copy()
rfm_clean['AvgDaysBetweenPurchases_x'].fillna(rfm['AvgDaysBetweenPurchases_x'].median(), inplace=True)
rfm_clean['Monetary'].fillna(rfm['Monetary'].median(), inplace=True)
Ho = 'There is no significant relationship between average days between purchases and monetary value'
Ha = 'There is significant relationship between average days between purchases and monetary value'
alpha = 0.05
statistic, pvalue = stats.pearsonr(rfm_clean['AvgDaysBetweenPurchases_x'],rfm_clean['Monetary'])
if pvalue < alpha:
  print(pvalue)
  print('Reject Null Hypotheses')
  print(Ha)
else:
  print(pvalue)
  print('Fail to Reject Null Hypotheses')
```

```
    print(Ho)
```

```
0.40571214377229947
Fail to Reject Null Hypotheses
There is no significant relationship between average days between purchases and monetary value
```

```
<ipython-input-59-5e1f4d066506>:7: DeprecationWarning: Please import `pearsonr` from the `scipy.stats` namespace; the `s
cipy.stats.stats` namespace is deprecated and will be removed in SciPy 2.0.0.
  statistic, pvalue = stats.pearsonr(rfm_clean['AvgDaysBetweenPurchases_x'],rfm_clean['Monetary'])
```

In [ ]:

```
rfm_clean[['Preferred_shopping_Day_x','Monetary']]
```

Out[ ]:

	Preferred_shopping_Day_x	Monetary
0	1	0.00
1	1	4310.00
2	3	1797.24
3	0	1757.55
4	2	334.40
...	...	...
4368	6	80.82
4369	4	176.60
4370	3	2094.88
4371	2	1837.28
4372	1	1447682.12

4373 rows × 2 columns

impact of preferred shopping day on spending

In [ ]:

```
f_oneway(rfm['Monetary'],rfm['Preferred_shopping_Day_x'])
```

Out[ ]:

```
F_onewayResult(statistic=39.73862573702288, pvalue=3.043757623227029e-10)
```

In [ ]:

```
Ho = 'There is no significant impact of preferred shopping day and amount spent'
Ha = 'There is a significant impact of preferred shopping day and amount spent'
alpha = 0.05
statistic , pvalue = f_oneway(rfm['Monetary'],rfm['Preferred_shopping_Day_x'])
if pvalue < alpha:
  print(pvalue)
  print('Reject Null Hypotheses')
  print(Ha)
else:
  print(pvalue)
  print('Fail to Reject Null Hypotheses')
  print(Ho)
```

```
3.043757623227029e-10
Reject Null Hypotheses
There is a significant impact of preferred shopping day and amount spent
```

Impact of peak shopping hours and amount spent

In [ ]:

```
Ho = 'There is no significant impact of preferred shopping hours and amount spent'
Ha = 'There is a significant impact of preferred shopping hours and amount spent'
alpha = 0.05
statistic , pvalue = f_oneway(rfm['Monetary'],rfm['Preferred_shopping_Hour_x'])
if pvalue < alpha:
  print(pvalue)
  print('Reject Null Hypotheses')
  print(Ha)
else:
  print(pvalue)
  print('Fail to Reject Null Hypotheses')
  print(Ho)
```

```
3.6465264061610397e-10
Reject Null Hypotheses
There is a significant impact of preferred shopping hours and amount spent
```

Insights and Recommendations

Conclusions:

Preferred Shopping Hours Impact Spending:

- Customers tend to spend significantly more during specific hours of the day.
- This indicates that shopping behavior is influenced by the time of day, likely related to convenience, routine, or promotional activities.

Preferred Shopping Days Impact Spending:

- There is a significant difference in spending based on the day of the week.
- Certain days might see higher spending due to factors like promotions, paydays, or typical consumer behavior patterns (e.g., weekend shopping)

No Significant Relationship Between Average Days Between Purchases and Monetary Value:

- The frequency of purchases (as measured by the average days between purchases) does not have a significant impact on how much customers spend in total.
- This suggests that customers who buy more frequently do not necessarily spend more over time, and high-value customers may have diverse shopping frequencies.

Recommendations:

- Optimize Marketing Campaigns Around Peak Shopping Hours
- Leverage Day-Specific Promotions
- Personalize Marketing Based on Shopping Patterns
- Do Not Overemphasize Purchase Frequency in Customer Segmentation
- Experiment with Pricing Strategies on Low-Traffic Days and Hours Analyze Non-Significant Factors Further
- valuate Shipping Costs: Since shipping fees contribute significantly to overall revenue, consider offering free shipping for high-value orders to encourage larger purchases or offering tiered shipping discounts.
- Focus on Customer Retention: Using the RFM segmentation, target high-value and at-risk customers with personalized marketing efforts. For example, offering exclusive discounts to 'At-Risk' customers could help bring them back to the platform.
- Monitor Operational Costs: Finally, regularly monitor expenses such as bank charges and commissions to reduce unnecessary financial drains on the business.”