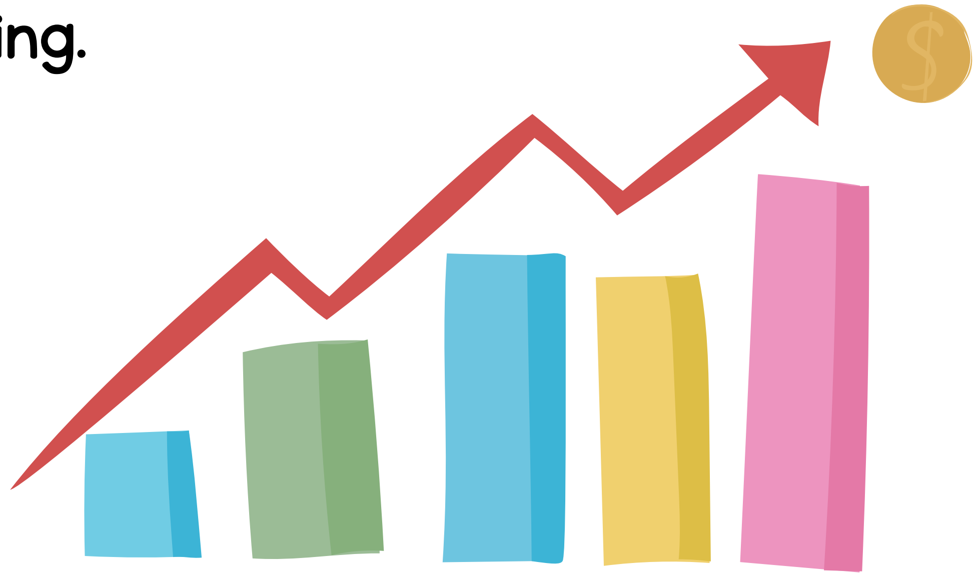
DATA ANALYSIS ON RETAIL STORE

DATA

We possess two datasets from a Retail establishment: one holds Customer transaction specifics, while the other records Customer response. By employing data analysis and visualization tools like Python, MySQL, and MS-Excel, we intend to derive meaningful insights from these datasets. Our examination will encompass diverse areas such as sales figures, customer actions, and product trends, offering a thorough grasp of the store's functions and aiding in strategic decision-making.



DATA ANALYSIS USING PYTHON

We are conducting Data analysis on a Retail store using Python in Jupyter Notebook. The dataset includes Sales Transactions and Sales Responses. The initial steps are Data Collection and Data Setup. This involves gathering data from appropriate sources and setting up or uploading the correct path in Jupyter Notebook to ensure the dataset can be read properly for subsequent Data Analysis and Visualization.

DATA COLLECTION AND SETUP

Data Collection: Download the data from Kaggle as a csv file and place it on the proper path

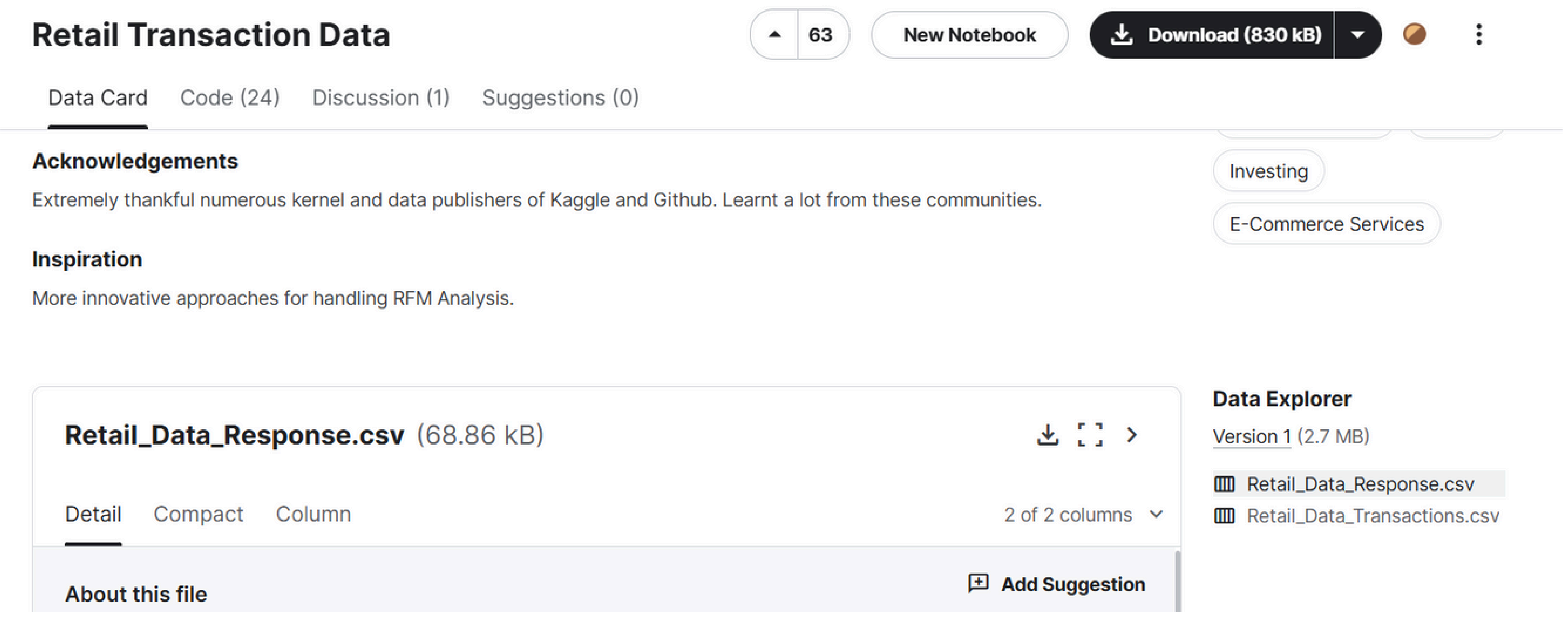
Find the data here: Retail Data. This dataset includes the following fields:

Customer_id: A unique identifier for each Customer.

Transaction_date: The Date the transaction took place.

Transaction_amount: The Amount paid by a Customer.

Response: Kind of Feedback about Purchases.



DATA CLEANING AND PREPARATION

Data Reading:

Sales_Data_Transaction

In [1]:
```python
#Installing Libraries
import pandas as pd
```

In [2]:
```python
Trans = pd.read_csv('Retail_Data_Transactions.csv')
Trans
```

Out[2]:

	customer_id	trans_date	tran_amount
0	CS5295	11-Feb-13	35
1	CS4768	15-Mar-15	39
2	CS2122	26-Feb-13	52
3	CS1217	16-Nov-11	99
4	CS1850	20-Nov-13	78
...	...	...	...
124995	CS8433	26-Jun-11	64
124996	CS7232	19-Aug-14	38
124997	CS8731	28-Nov-14	42
124998	CS8133	14-Dec-13	13
124999	CS7996	13-Dec-14	36

125000 rows × 3 columns

Sales_Data_Response

In [3]:
```python
response = pd.read_csv('Retail_Data_Response.csv')
response
```

Out[3]:

	customer_id	response
0	CS1112	0
1	CS1113	0
2	CS1114	1
3	CS1115	1
4	CS1116	1
...	...	...
6879	CS8996	0
6880	CS8997	0
6881	CS8998	0
6882	CS8999	0
6883	CS9000	0

6884 rows × 2 columns

Data Merging:

In [4]:
```python
df = pd.merge(Trans, response, on = 'customer_id', how='left')
df
```

Out[4]:

	customer_id	trans_date	tran_amount	response
0	CS5295	11-Feb-13	35	1.0
1	CS4768	15-Mar-15	39	1.0
2	CS2122	26-Feb-13	52	0.0
3	CS1217	16-Nov-11	99	0.0
4	CS1850	20-Nov-13	78	0.0
...	...	...	...	...
124995	CS8433	26-Jun-11	64	0.0
124996	CS7232	19-Aug-14	38	0.0
124997	CS8731	28-Nov-14	42	0.0
124998	CS8133	14-Dec-13	13	0.0
124999	CS7996	13-Dec-14	36	0.0

125000 rows × 4 columns

DATA CLEANING AND PREPARATION

Data Features:

Performed various features to know more about the dataset which will help to analyze the data more effectively.

i)df.shape gives the shape of the dataset means the number of Rows and Columns the dataset contains.

ii)df.head() gives the first 5 rows of the dataset.

iii)df.tail() gives the last 5 rows of the dataset.

iv)df.dtypes defines the datatypes of the attributes.

v)df.describe() gives the statistics of the numerical data.

In [48]:
```
#Feature
df.shape
```
Out[48]:
```
(124969, 5)
```

In [49]:
```
df.head()
```
Out[49]:

	customer_id	trans_date	tran_amount	response	month
0	CS5295	2013-02-11	35	1	2
1	CS4768	2015-03-15	39	1	3
2	CS2122	2013-02-26	52	0	2
3	CS1217	2011-11-16	99	0	11
4	CS1850	2013-11-20	78	0	11

In [50]:
```
df.tail()
```
Out[50]:

	customer_id	trans_date	tran_amount	response	month
124995	CS8433	2011-06-26	64	0	6
124996	CS7232	2014-08-19	38	0	8
124997	CS8731	2014-11-28	42	0	11
124998	CS8133	2013-12-14	13	0	12
124999	CS7996	2014-12-13	36	0	12

In [51]:
```
df.dtypes
```
Out[51]:
```
customer_id             object
trans_date      datetime64[ns]
tran_amount              int64
response                 int64
month                    int64
dtype: object
```

In [52]:
```
df.info()
```
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 124969 entries, 0 to 124999
Data columns (total 5 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   customer_id  124969 non-null  object
 1   trans_date   124969 non-null  datetime64[ns]
 2   tran_amount  124969 non-null  int64
 3   response     124969 non-null  int64
 4   month        124969 non-null  int64
dtypes: datetime64[ns](1), int64(3), object(1)
memory usage: 5.7+ MB
```

In [53]:
```
df.describe()
```
Out[53]:

	tran_amount	response	month
count	124969.000000	124969.000000	124969.000000
mean	64.995143	0.110763	6.631725
std	22.860059	0.313840	3.475188
min	10.000000	0.000000	1.000000
25%	47.000000	0.000000	4.000000
50%	65.000000	0.000000	7.000000
75%	83.000000	0.000000	10.000000
max	105.000000	1.000000	12.000000

DATA CLEANING AND PREPARATION

Drop the missing values and change datatypes:

i)df.dropna() helps to drop the missing values in the dataset.

ii)pd.to_datetime helps to change the datatype to datetime value

iii)astype() helps to convert column to int, float, str values.

In [14]:
```
df = df.dropna()
df
```

Out[14]:

	customer_id	trans_date	tran_amount	response
0	CS5295	2013-02-11	35	1.0
1	CS4768	2015-03-15	39	1.0
2	CS2122	2013-02-26	52	0.0
3	CS1217	2011-11-16	99	0.0
4	CS1850	2013-11-20	78	0.0
...	...	...	...	...
124995	CS8433	2011-06-26	64	0.0
124996	CS7232	2014-08-19	38	0.0
124997	CS8731	2014-11-28	42	0.0
124998	CS8133	2013-12-14	13	0.0
124999	CS7996	2014-12-13	36	0.0

124969 rows × 4 columns

In [16]:
```
#Change DataTypes
df['trans_date'] = pd.to_datetime(df['trans_date'])
df['response'] = df['response'].astype('int64')
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_22572\3585468186.py:2: S
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/panda
rsus-a-copy
  df['trans_date'] = pd.to_datetime(df['trans_date'])
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_22572\3585468186.py:3: S
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/panda
rsus-a-copy
  df['response'] = df['response'].astype('int64')
```

In [17]:
```
df
```

DATA CLEANING AND PREPARATION

Check Outliers using Z_Score:

1. Both "SciPy" and "NumPy" are essential libraries in Python, particularly for scientific and numerical computing. They offer a range of functionalities that make complex mathematical and statistical operations more accessible and efficient
2. In statistics, a Z-score (or standard score) is a measure of how many standard deviations a data point is from the mean of the dataset. It is calculated using the formula:

$$Z=(X-\mu)/\sigma$$

where:

- X is the value of the data point,
- μ is the mean of the dataset,
- σ(sigma) is the standard deviation of the dataset.

3. A Z-score helps identify outliers by showing how far a point is from the mean. Typically, a Z-score above 3 or below -3 is considered an outlier.

Here, we have use Z_score function to find the outliers in the column tran_amoount with the use of above formula. Hence, There is No Outlier in the Column Tran_amount.

```
In [41]: #Check Outliers
#Z_Score
from scipy import stats
import numpy as np

#Calculate Z_Score
z_scores = np.abs(stats.zscore(df['tran_amount']))

#Use of Abs makes the threshold value always greater than 3

#set a threshold
threshold = 3

#Outlier
Outliers = z_scores>threshold

print(Outliers)

0         False
1         False
2         False
3         False
4         False
          ...
124995    False
124996    False
124997    False
124998    False
124999    False
Name: tran_amount, Length: 124969, dtype: bool
```

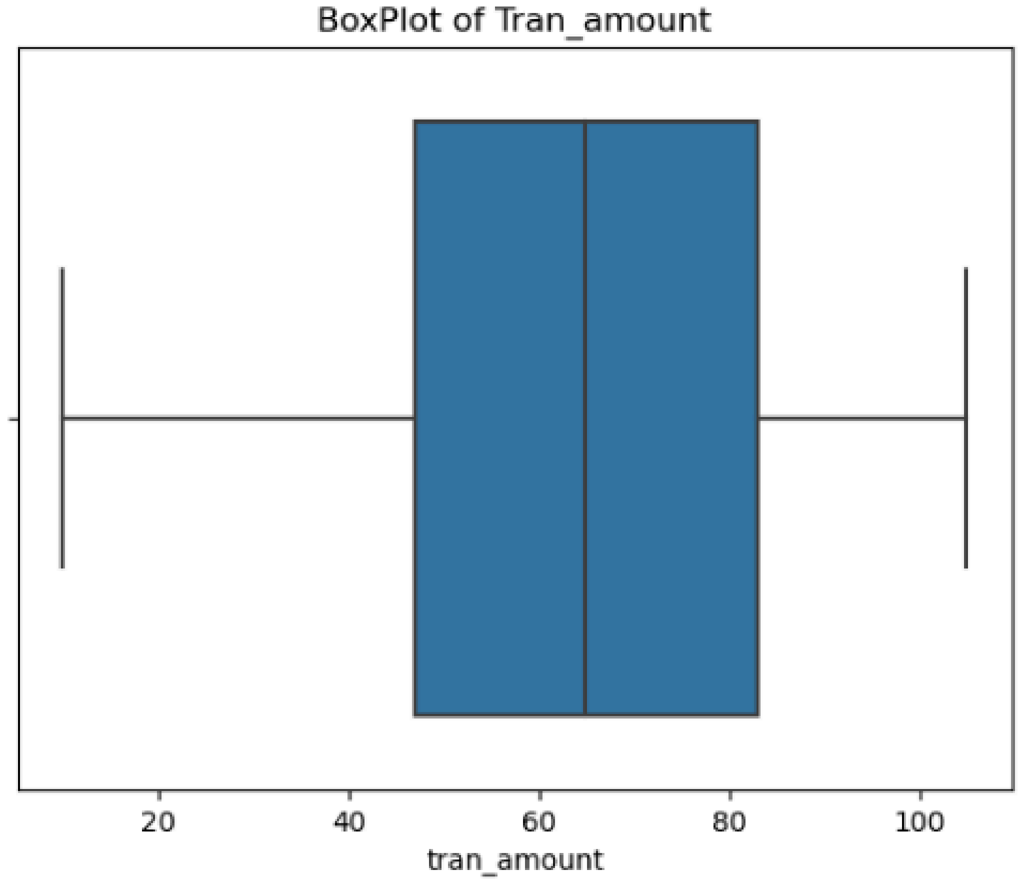
DATA ANALYSIS

Check Outliers using BoxPlot:

1. Both Seaborn and Matplotlib are the necessary libraries for visualizing BoxPlot.
2. A boxplot, also known as a box-and-whisker plot, is a standardized way of displaying the distribution of data based on a five-number summary: minimum, first quartile (Q1), median, third quartile (Q3), and maximum. It can also show outliers explicitly.

- Here, we have use BoxPlot Graph to find the outliers in the column tran_amount. We can make the insights of the tran_amount data from the BoxPlot such as Min value lies in the range of 10, Max value lies above 100, whereas Q1 is between 40 to 50, Q3 lies between 80 to 90 and Q2 lies mid of the Inter-quartile range having 50% diff from Q1 and Q3. Hence, There is No Outlier in the Column Tran_amount plotted in the BoxPlot.

```
In [19]: #BoxPlot
import seaborn as sns
import matplotlib.pyplot as plt


sns.boxplot(x = df['tran_amount'])
plt.title('BoxPlot of Tran_amount')
plt.show()
```



DATA ANALYSIS

Add New Column to Dataset: Added New Column called "month" for effective data analysis of the Sales.

```
n [20]: #Creating New Columns
        #It gives the number of month
        df['month']=df['trans_date'].dt.month
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_22572\1824379682.py:3: S
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/panda
rsus-a-copy
  df['month']=df['trans_date'].dt.month
```

```
n [21]: df
```

ut[21]:

	customer_id	trans_date	tran_amount	response	month
0	CS5295	2013-02-11	35	1	2
1	CS4768	2015-03-15	39	1	3
2	CS2122	2013-02-26	52	0	2
3	CS1217	2011-11-16	99	0	11
4	CS1850	2013-11-20	78	0	11
...	...	...	...	...	...
124995	CS8433	2011-06-26	64	0	6
124996	CS7232	2014-08-19	38	0	8
124997	CS8731	2014-11-28	42	0	11
124998	CS8133	2013-12-14	13	0	12
124999	CS7996	2014-12-13	36	0	12

124969 rows × 5 columns

DATA ANALYSIS

Which Top 3months had the highest Sum of Sales?

After applying the Groupby and sort_values functions on the tran_amount column to calculate the total Sales per month, the analysis revealed that the sales figures for the 8th, 10th, and 1st months were 726,775, 725,058, and 724,089, respectively.

In [22]:
```python
#Which 3 months have had the highest tran_amount
monthly_Sales = df.groupby('month')['tran_amount'].sum()
monthly_Sales = monthly_Sales.sort_values(ascending=False).reset_index().head(3)
monthly_Sales
```

Out[22]:

	month	tran_amount
0	8	726775
1	10	725058
2	1	724089

Which Top 5 Customers have made highest purchase of Orders?

After applying the value_counts function on the customer_id column to calculate the highest purchase made by the customer, the analysis revealed that the Customer_id CS4424, CS4320, CS3799, CS3013, CS1215 made purchase of count 39, 38, 36, 35, 35 respectively.

In [55]:
```python
#Group by Customer id and
# find out which 5 customers have had the highest number of orders

customer_counts = df['customer_id'].value_counts().reset_index()     #value_counts function give you count of things
customer_counts.columns=['customer_id','count']            #Rename the Columns
customer_counts
```

Out[55]:

	customer_id	count
0	CS4424	39
1	CS4320	38
2	CS3799	36
3	CS3013	35
4	CS1215	35
...	...	...
6879	CS8559	4
6880	CS7224	4
6881	CS7716	4
6882	CS8504	4
6883	CS7333	4

6884 rows × 2 columns

DATA ANALYSIS

Data Visualization:

1. Utilize a Bar Plot for Data Visualization showcasing the top 5 customers with the highest order purchases.
2. Both Seaborn and Matplotlib libraries were employed to generate a Bar Chart illustrating the sales count by each customer.

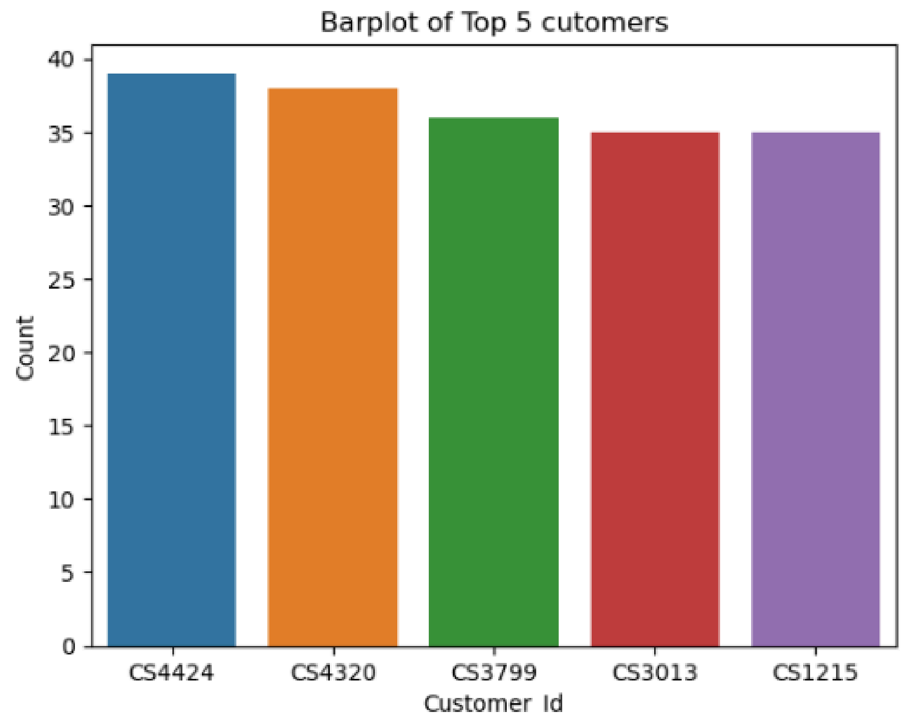
The analysis of the bar chart reveals that customer_id CS4424 had the highest count of 39, while customer_ids CS3013 and CS1215 had the lowest count of 35.

In [56]:
```
#Customers having the highest number of order

top_5_cus = customer_counts.sort_values(by ='count', ascending=False).head(5)
top_5_cus
```

Out[56]:

	customer_id	count
0	CS4424	39
1	CS4320	38
2	CS3799	36
3	CS3013	35
4	CS1215	35

In [59]:
```
#BarPlot of Top 5 Customers
sns.barplot(x='customer_id', y='count', data=top_5_cus)
plt.xlabel('Customer_Id')
plt.ylabel('Count')
plt.title('Barplot of Top 5 cutomers')
plt.show()
```



DATA ANALYSIS

Data Visualization:

1. Utilize Bar Plot for Data Visualization showcasing the Top 5 Customers with the highest total transaction amounts.
2. Seaborn and Matplotlib libraries are employed to generate Bar Charts illustrating the total sales made by each customer.

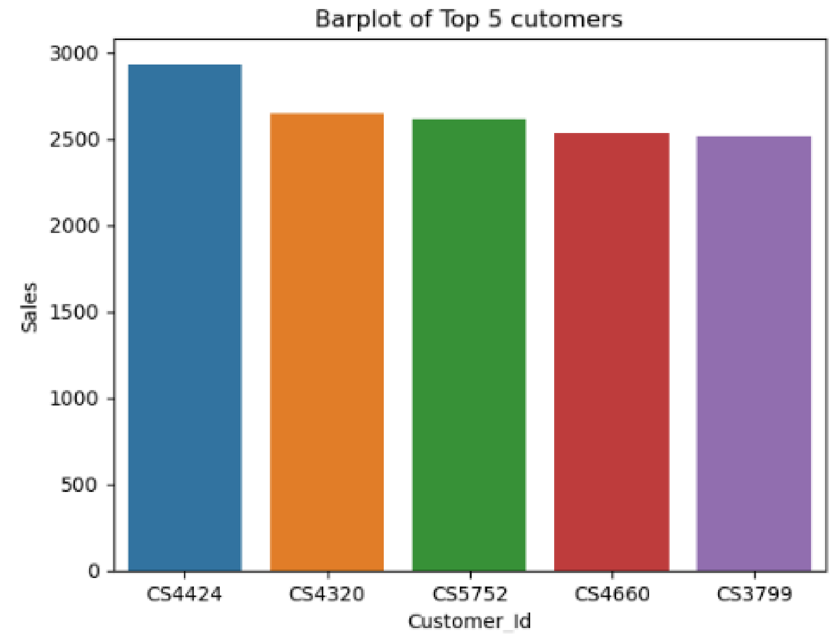
The analysis of the bars reveals a range of transaction amounts, with customer_id CS4424 having the highest sum of 2933, and CS4660 and CS3799 having the lowest sums of 2527 and 2513 respectively.

In [60]:
```
#Group by Customer id and
# find out which 5 customers have had the highest value of orders
customer_sales = df.groupby('customer_id')['tran_amount'].sum().reset_index()

customer_sales

#Top 5 Customers having the highest value of order
top_5_sales = customer_sales.sort_values(by ='tran_amount', ascending=False).head(5)
top_5_sales
```

Out[60]:

	customer_id	tran_amount
3312	CS4424	2933
3208	CS4320	2647
4640	CS5752	2612
3548	CS4660	2527
2687	CS3799	2513

In [63]:
```
#BarPlot of Top 5 Customers
sns.barplot(x='customer_id', y='tran_amount', data=top_5_sales)
plt.xlabel('Customer_Id')
plt.ylabel('Sales')
plt.title('Barplot of Top 5 cutomers')
plt.show()
```



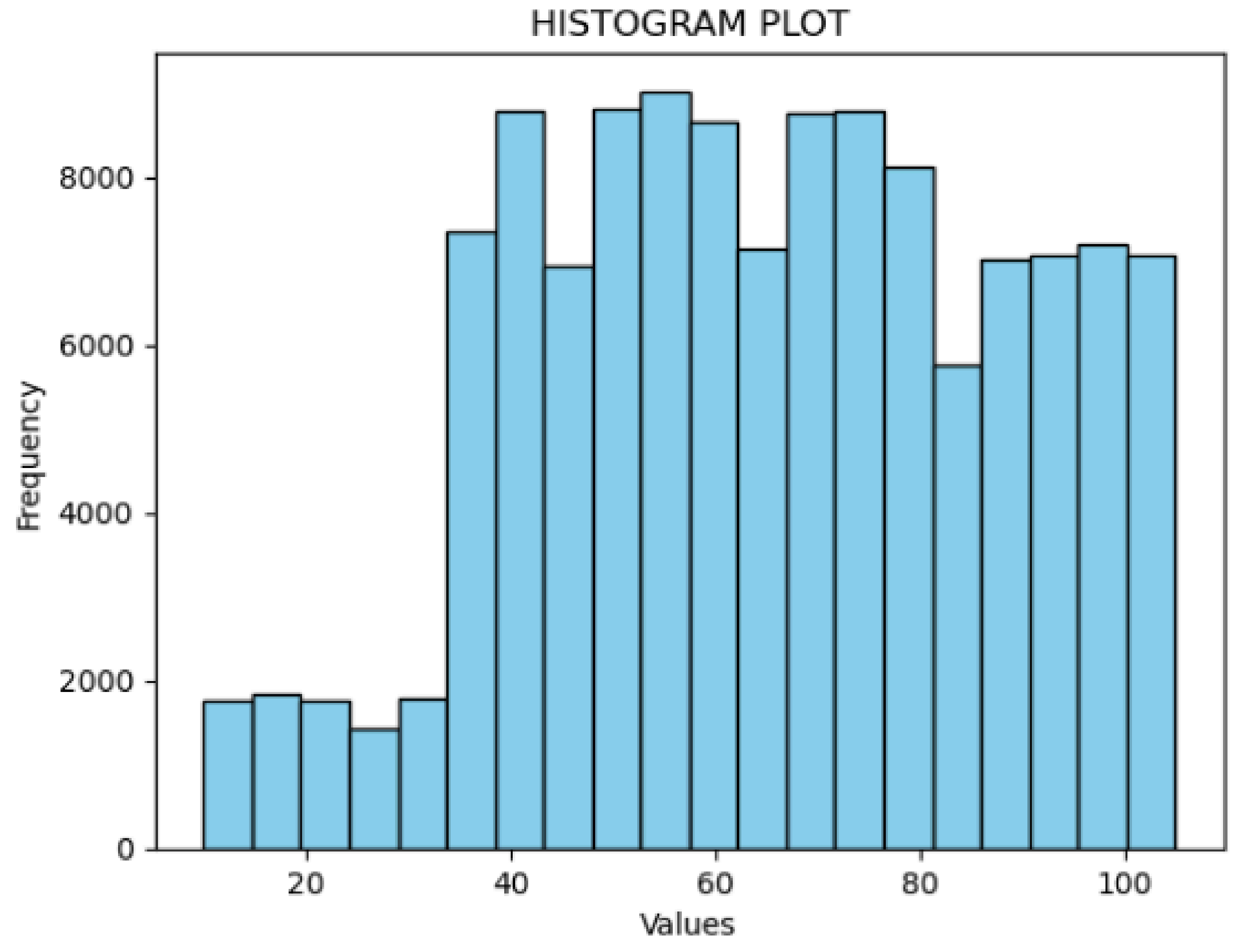
DATA ANALYSIS

Data Visualization:

1. Use Histogram for Data Visualization to display the Frequency alongside the total transaction amounts.
2. Matplotlib library is used to create a Histogram that shows the frequency of total sales.

By analyzing the bins, it is evident that the range begins at the lowest frequency of the sum of trans_amount 1900 and spans various amount ranges up to the highest frequency of the sum of trans_amount 9000.

In [13]:
```
#Visualization
plt.hist(df['tran_amount'], bins=20, color='skyblue', edgecolor='black')
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.title('HISTOGRAM PLOT')
plt.show()
```



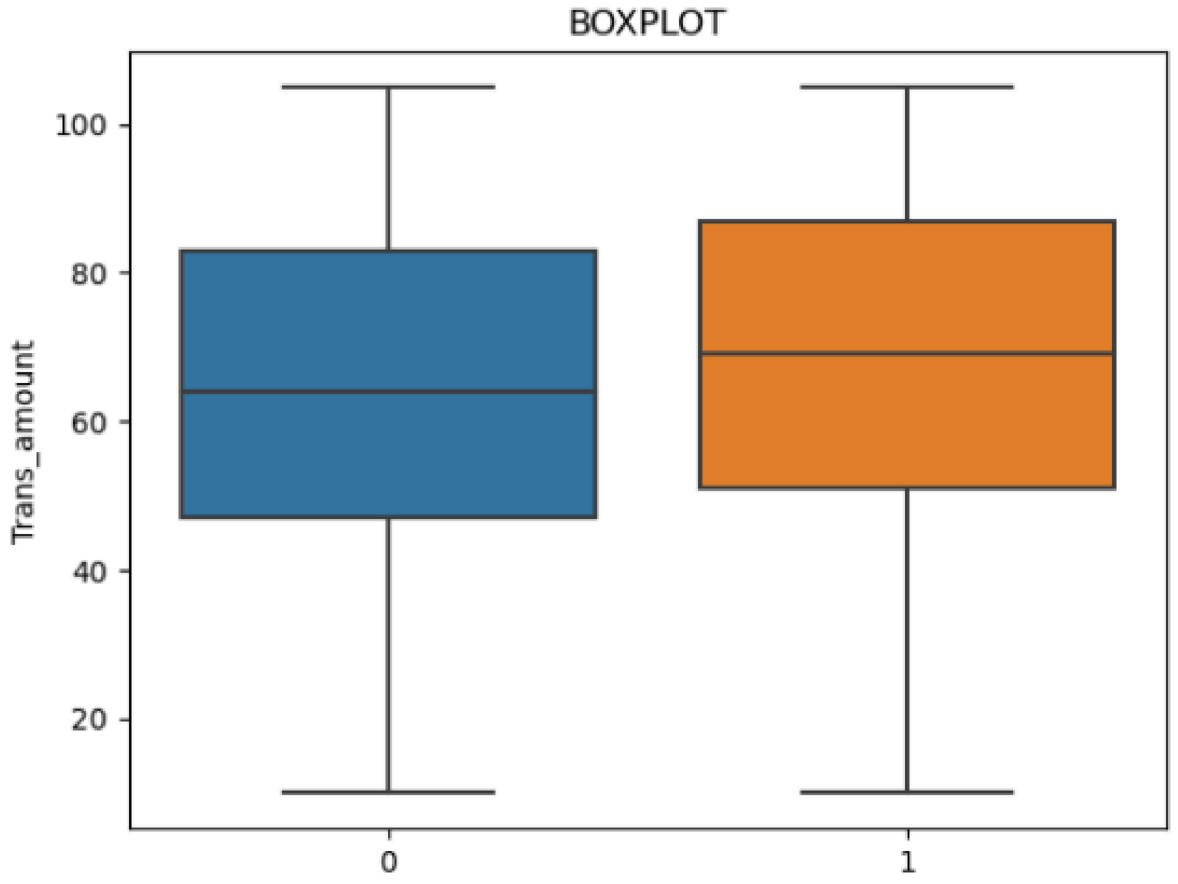
DATA ANALYSIS

Data Visualization:

1. Utilizing BoxPlot for Data Visualization to present Total transaction amounts with the Response.
2. The BoxPlot is crafted using Seaborn and Matplotlib libraries to illustrate the total sales response.

Upon examining the Boxplot of Response 0 and 1, it is clear that both plots have a minimum value within the 10 range. The first quartile (Q1) of response 0 falls between 40 to 50, while the first quartile(Q1) of response 1 ranges from 50 to 60. The third quartile (Q3) of both plots for response 0 and 1 falls between 80 to 90, with the Interquartile Range being the same size, albeit with a slight variation between them.

In [43]:
```
import seaborn as sns

sns.boxplot(x='response', y='tran_amount', data=df)
plt.xlabel('Response')
plt.ylabel('Trans_amount')
plt.title('BOXPLOT')
plt.show()
```



ADVANCED ANALYSIS

Time Series Analysis:

1. We have added a new column, 'month-year,' to the dataset to facilitate more effective analysis of the transaction amounts using time-series analysis in Python.
2. The matplotlib.dates library has been utilized to plot the graph, allowing for clear visualization of sales trends over time.

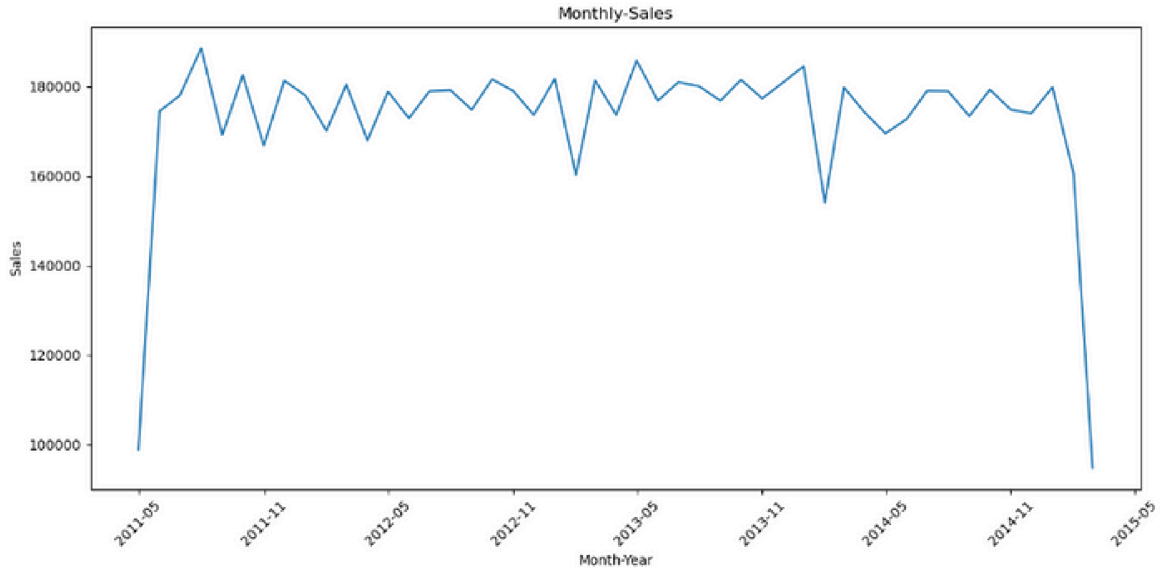
By examining the plot, we observed a decline in sales during the months of February 2013 and February 2014, whereas there was a significant increase in sales in July 2011.

Advanced Analytics

Time Series Analysis

In [67]:
```
import matplotlib.dates as mdates

df['month_year']=df['trans_date'].dt.to_period('M')
monthly_Sales = df.groupby('month_year')['tran_amount'].sum()


monthly_Sales.index = monthly_Sales.index.to_timestamp()


plt.figure(figsize=(12, 6))
plt.plot(monthly_Sales.index, monthly_Sales.values)

plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=6))
plt.xlabel('Month-Year')
plt.ylabel('Sales')
plt.title('Monthly-Sales')
plt.xticks(rotation=45)                    #Helps to rotate the data label of axis
plt.tight_layout()
plt.show()
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_23948\2237945691.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['month_year']=df['trans_date'].dt.to_period('M')
```



ADVANCED ANALYSIS

Cohort Segmentation:

To better analyze customer behavior using cohort segmentation, we have created a new dataframe called RFM. In this dataframe:

- R stands for Recency, representing the most recent date a sale was made.
- F stands for Frequency, indicating the count of orders made by each customer (grouped by customer_id).
- M stands for Monetary, which is the total transaction amount for each customer (grouped by customer_id).

Cohort Segmentation ---Depending on Customer Behaviour

In [23]:

```
#Recency
#recency gives you what is the most recent order of a particular person
recency = df.groupby('customer_id')['trans_date'].max()

#Frequency
#Frequency gives you how many orders did the customer placed
frequency = df.groupby('customer_id')['trans_date'].count()

#Monetary
monetary = df.groupby('customer_id')['tran_amount'].sum()

#Combine
rfm = pd.DataFrame({'recency': recency,
                    'frequency': frequency,
                    'monetary': monetary})
```

In [69]:

```
rfm
```

Out[69]:

	recency	frequency	monetary
customer_id			
CS1112	2015-01-14	15	1012
CS1113	2015-02-09	20	1490
CS1114	2015-02-12	19	1432
CS1115	2015-03-05	22	1659
CS1116	2014-08-25	13	857
...	...	...	...
CS8996	2014-12-09	13	582
CS8997	2014-06-28	14	543
CS8998	2014-12-22	13	624
CS8999	2014-07-02	12	383
CS9000	2015-02-28	13	533

6884 rows × 3 columns

ADVANCED ANALYSIS

Cohort Segmentation:

We have added a new column named Segment to the rfm dataframe. The Segment column has values P0, P1, and P2. We applied the function segment_customer to the rfm dataframe. This function encodes the Segment column using the different parameters mentioned.

In [24]:
```
#Customer Segmentation

def segment_customer(row):
    if row['recency'].year>=2012 and row['frequency']>=15 and row['monetary']>1000:
        return 'P0'
    elif (2011<=row['recency'].year<2012) and (10<row['frequency']<15) and (500<=['monetary']<=1000):
        return 'P1'
    else:
        return 'P2'

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


#P0 is the highest cohort of customer segment
```

In [25]:
```
rfm
```

Out[25]:

customer_id	recency	frequency	monetary	segment
CS1112	2015-01-14	15	1012	P0
CS1113	2015-02-09	20	1490	P0
CS1114	2015-02-12	19	1432	P0
CS1115	2015-03-05	22	1659	P0
CS1116	2014-08-25	13	857	P2
...	...	...	...	...
CS8996	2014-12-09	13	582	P2
CS8997	2014-06-28	14	543	P2
CS8998	2014-12-22	13	624	P2
CS8999	2014-07-02	12	383	P2
CS9000	2015-02-28	13	533	P2

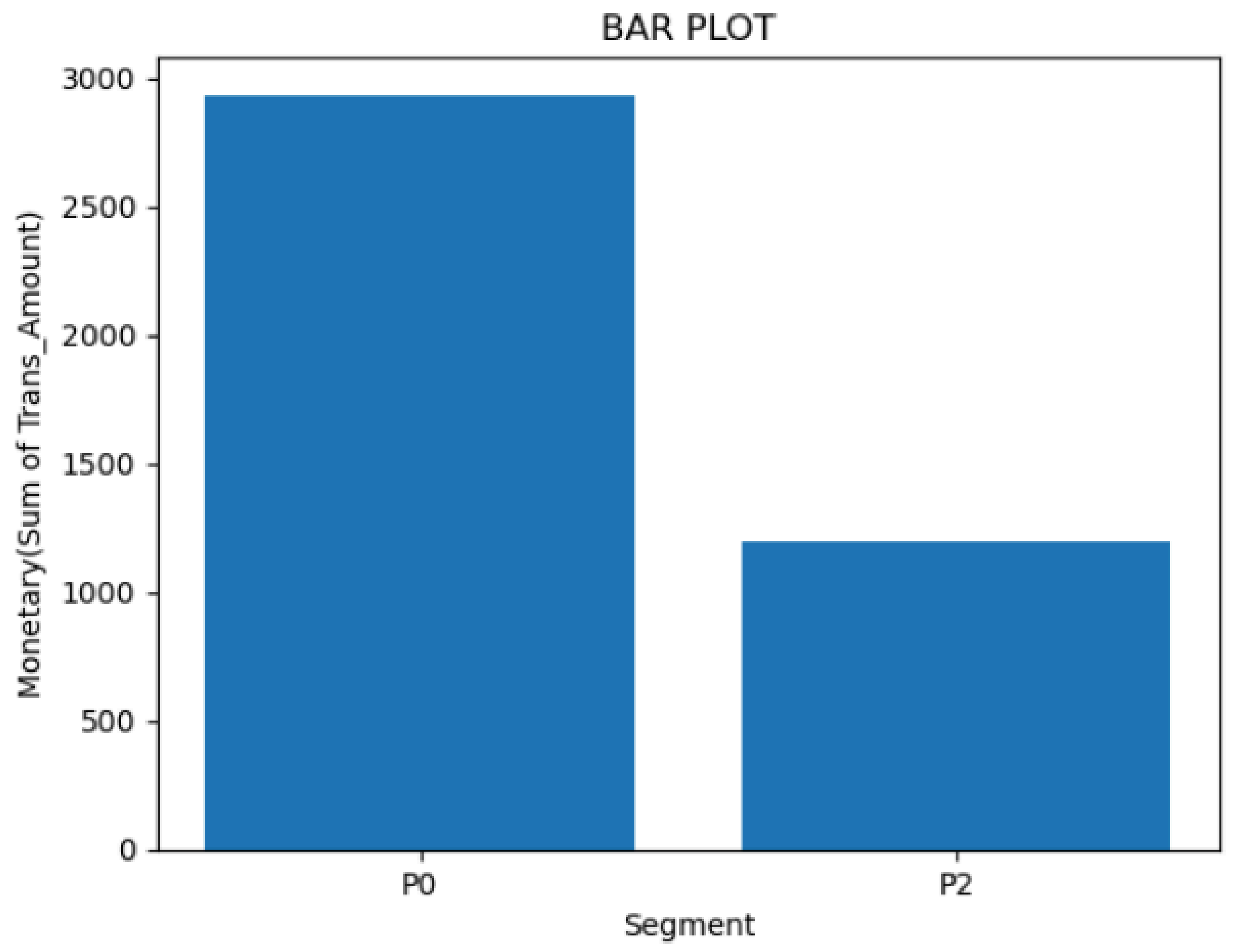
6884 rows × 4 columns

ADVANCED ANALYSIS

Cohort Segmentation:

1. Using BarPlot for Data Visualization to display Total Transaction Amounts by Segment.
2. Upon analyzing the data with Bar Plot, it is evident that segment P0 has the largest cohort in comparison to P2.
3. The bar representing P0 ranges from 2700 to 3000 in transaction amounts, while the P2 bar falls within the range of 1000 to 1500.

In [56]:
```
#Create a ScatterPlot
plt.bar(rfm['segment'], rfm['monetary'])
plt.xlabel('Segment')
plt.ylabel('Monetary(Sum of Trans_Amount)')
plt.title('BAR PLOT')
plt.show()
```



ADVANCED ANALYSIS

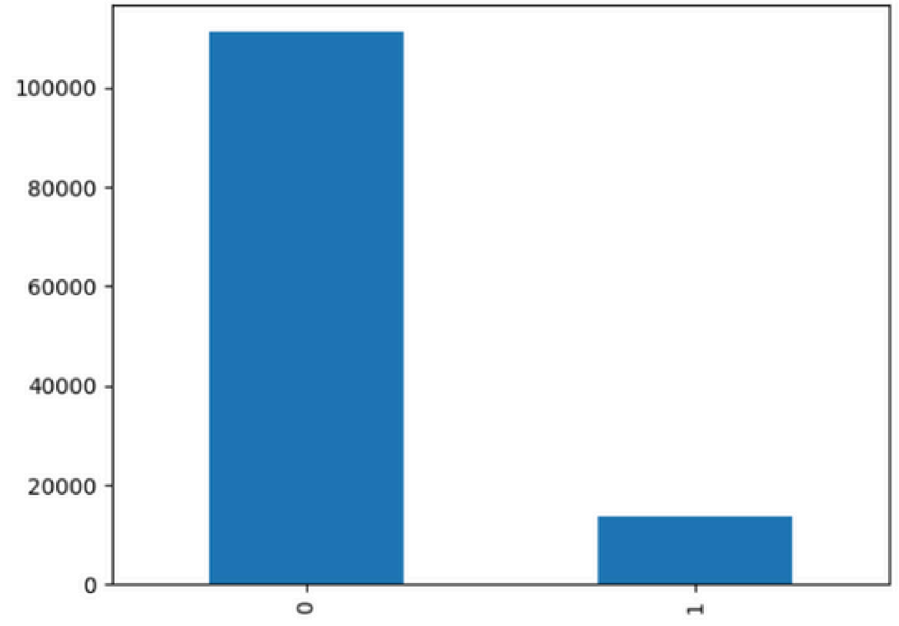
Churn Analysis:

1. Customer Churn Analysis, also known as customer attrition analysis, involves examining and comprehending customer turnover in a business setting. Churn refers to the frequency at which customers cease using a product, service, or cancel their membership.
2. In our churn analysis, we utilized the count of customer responses regarding their sales purchases.
3. Upon analyzing the BarPlot, it is apparent that a large number of individuals are categorized under response 0. This suggests that customers are dissatisfied with their sales purchases, while only a small number of customers fall into the response 1 category.

Churn Analysis

In [74]:
```
#Churn Analysis
# to know how well is your business performs and
# how are the reviews and ratings of your customer

#--------0=Not Satisfies and 1=Satisfied
#-------peole who are not not satisfied are the potential churner
#If they are not satisfied by flipkart and
# they move on to amazon they are known as Churned Customer


#Count the number of churned and active customers
churn_counts= df['response'].value_counts()

#PLots
churn_counts.plot(kind='bar')
```

Out[74]: <AxesSubplot:>



EXPORTING THE FILE

Exporting the jupyter notebook as Csv file

pd.to_csv is been used to upload the jupyter notebook as Csv file

```
In [89]: #Exporting the Main Data as CSV file
         df.to_csv('MainData.csv')
```

```
In [54]: #Exporting the Rfm data as CSV file
         rfm.to_csv('AddAnalysis.csv')
```

DATA ANALYSIS USING EXCEL

We possess two datasets from a Retail establishment: one holds MainData specifics, while the other records AdvAnalysis. By employing data analysis and visualization on MS-Excel, we intend to derive meaningful insights from these datasets. The initial steps are formatting the data, creating Pivot tables, Pivot Charts and manymore.

DATA ANALYSIS USING EXCEL

DataSets:

MainData.csv

J6 | MainData

	A	B	C	D	E	F
1	customer_id	trans_date	response	tran_amount	month	month_yea
2	CS5295	2/11/2013	1	35	2	2013-02
3	CS4768	3/15/2015	1	39	3	2015-03
4	CS2122	2/26/2013	0	52	2	2013-02
5	CS1217	11/16/2011	0	99	11	2011-11
6	CS1850	11/20/2013	0	78	11	2013-11
7	CS5539	3/26/2014	0	81	3	2014-03
8	CS2724	2/6/2012	0	93	2	2012-02
9	CS5902	1/30/2015	0	89	1	2015-01
10	CS6040	1/8/2013	0	76	1	2013-01
11	CS3802	8/20/2013	1	75	8	2013-08
12	CS3494	7/2/2013	0	94	7	2013-07
13	CS3780	3/25/2013	0	80	3	2013-03
14	CS1171	11/3/2012	0	59	11	2012-11
15	CS2892	5/12/2013	0	43	5	2013-05
16	CS5552	12/29/2014	0	78	12	2014-12
17	CS6043	1/15/2014	0	98	1	2014-01
18	CS4147	7/8/2013	0	81	7	2013-07
19	CS4655	12/30/2013	0	93	12	2013-12
20	CS3904	7/20/2014	0	103	7	2014-07
21	CS4102	7/9/2011	0	96	7	2011-07
22	CS2086	3/5/2013	0	75	3	2013-03
23	CS6085	1/9/2013	0	49	1	2013-01
24	CS1328	2/6/2013	0	54	2	2013-02
25	CS4564	3/27/2012	0	48	3	2012-03
26	CS5910	1/1/2012	0	98	1	2012-01
27	CS2748	3/23/2013	1	37	3	2013-03

AdvAnalysis.csv

H18 | AdvanceAnalysis

	A	B	C	D	E	F
1	customer_id	recency	segment	frequency	monetary	
2	CS1112	1/14/2015	P0	15	1012	
3	CS1113	2/9/2015	P0	20	1490	
4	CS1114	2/12/2015	P0	19	1432	
5	CS1115	3/5/2015	P0	22	1659	
6	CS1116	8/25/2014	P2	13	857	
7	CS1117	7/2/2014	P0	17	1185	
8	CS1118	3/14/2015	P0	15	1011	
9	CS1119	3/5/2015	P0	15	1158	
10	CS1120	3/6/2015	P0	24	1677	
11	CS1121	2/3/2015	P0	26	1524	
12	CS1122	2/2/2015	P0	16	1156	
13	CS1123	11/27/2014	P0	19	1331	
14	CS1124	1/2/2015	P0	18	1127	
15	CS1125	2/19/2015	P2	12	885	
16	CS1126	9/18/2014	P0	19	1165	
17	CS1127	2/5/2015	P0	24	1676	
18	CS1128	12/31/2014	P0	26	1921	
19	CS1129	11/30/2014	P2	12	853	
20	CS1130	3/13/2015	P0	19	1185	
21	CS1131	2/24/2015	P2	14	998	
22	CS1132	2/17/2015	P0	16	1074	
23	CS1133	2/21/2015	P0	19	1413	
24	CS1134	11/27/2014	P0	16	1166	
25	CS1135	4/15/2014	P0	20	1267	
26	CS1136	2/17/2015	P0	20	1339	
27	CS1137	1/20/2015	P0	20	1380	

PIVOT TABLES USING EXCEL

Pivot Tables from MainData.csv:

Created a pivot table showing the total transaction amount per month.

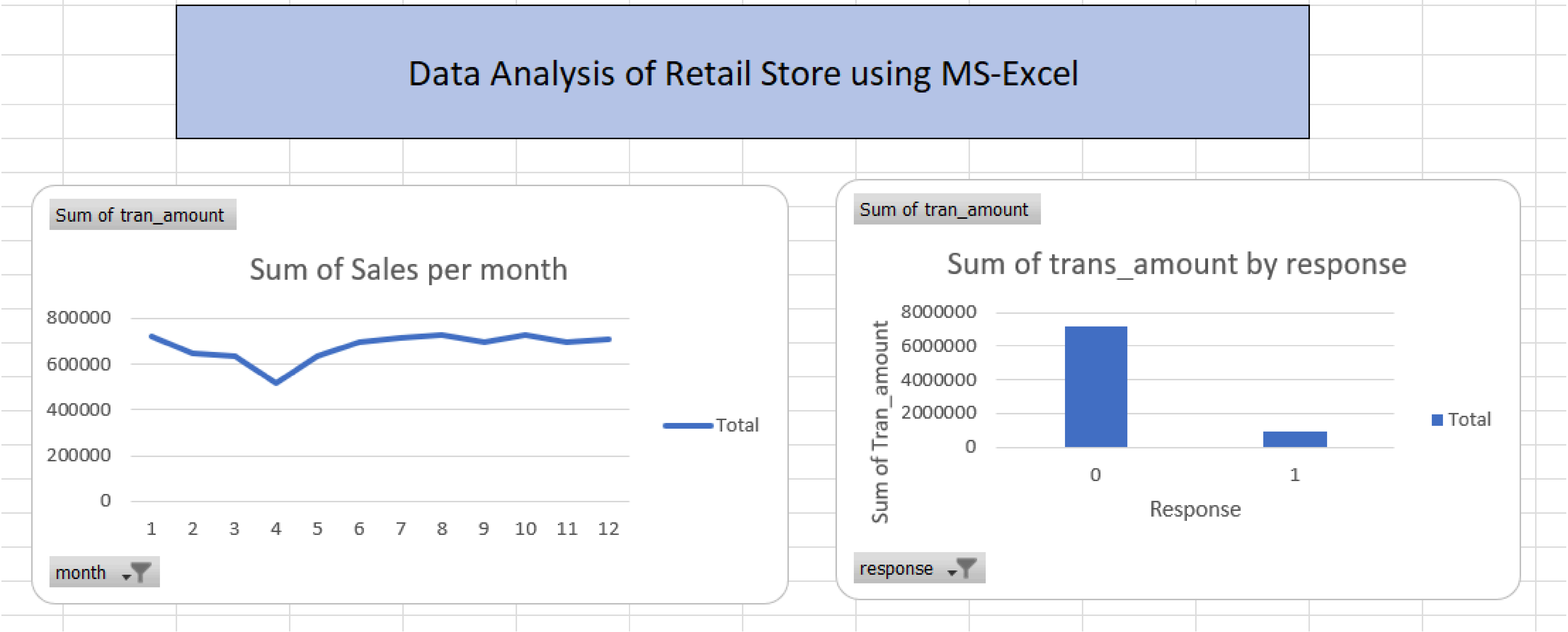
Row Labels	Sum of tran_amount
1	724089
2	645028
3	636475
4	515746
5	633162
6	697014
7	717011
8	726775
9	694201
10	725058
11	698024
12	709795
Grand Total	**8122378**

Created a pivot table that displays the total transaction amount based on responses.

Row Labels	Sum of tran_amount
0	7166830
1	955548
Grand Total	**8122378**

DATA VISUALIZATION USING EXCEL

Data Visualization:



PIVOT TABLES USING EXCEL

Pivot Tables from AdvAnalysis.csv:

Created a pivot table showing the Sum of Frequency based on segment.

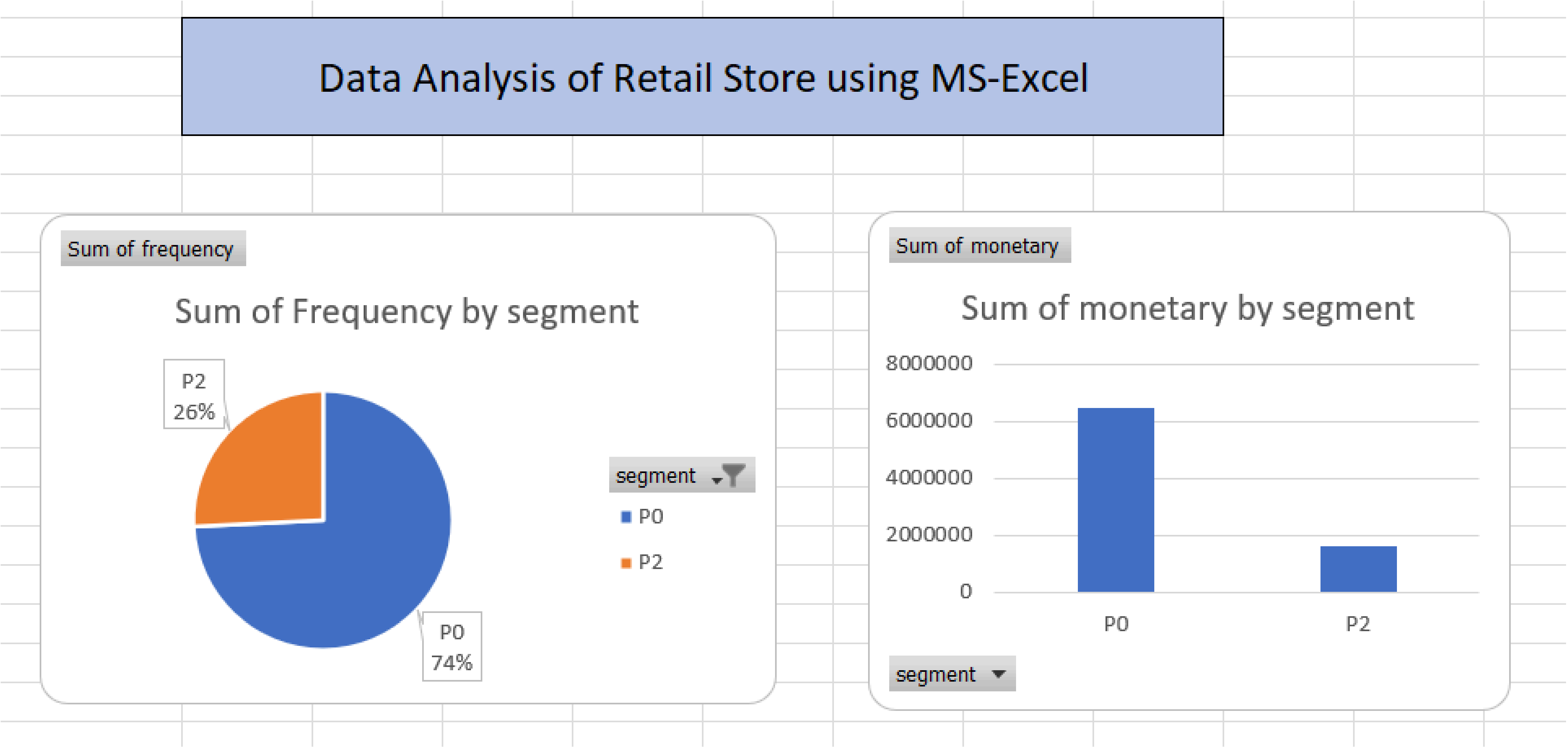
Row Labels	Sum of frequency
P0	92750
P2	32219
Grand Total	**124969**

Created a pivot table showing the Sum of Monetary based on segment.

segment	Sum of monetary
P0	6498293
P2	1624085

DATA VISUALIZATION USING EXCEL

Data Visualization:



DATA ANALYSIS USING EXCEL

1. We face limitations with MS Excel in terms of analysis and visualization due to the large data size, which often causes the file to hang and prevents proper performance. On the other hand, Python with Jupyter Notebook handles large datasets more efficiently, allowing for more comprehensive analysis and visualization.
2. One advantage of MS Excel is its ability to recommend charts, making visualization easier. As an alternative, Power BI is a powerful tool that enables us to extract, transform, and load data, creating effective dashboards that facilitate decision-making and business insights.

DATA ANALYSIS USING MYSQL

1. We have two datasets from a Retail store: one contains details about <u>Sales Data Transactions</u>, and the other captures <u>Sales Data Responses.</u>
2. Our goal is to extract valuable insights from these datasets through data analysis using MySQL queries.
3. The first actions involve establishing a database named "RetailSalesdata" and setting up tables named sales_data_transactions and sales_data_response.

DATA ANALYSIS USING MYSQL

Imported dataset csv files in Tables:

sales_data_transactions

```sql
SELECT * FROM retailsalesdata.sales_data_transactions;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	trans_date	trans_amount
CS5295	11-Feb-13	35
CS4768	15-Mar-15	39
CS2122	26-Feb-13	52
CS1217	16-Nov-11	99
CS1850	20-Nov-13	78
CS5539	26-Mar-14	81
CS2724	6-Feb-12	93
CS5902	30-Jan-15	89
CS6040	8-Jan-13	76

sales_data_response

```sql
SELECT * FROM retailsalesdata.sales_data_response;
```

Result Grid | Filter Rows: | Edit: | Export/Im

customer_id	response
CS1112	0
CS1113	0
CS1114	1
CS1115	1
CS1116	1
CS1117	0
CS1118	1
CS1119	0
CS1120	0
CS1121	0

DATA ANALYSIS USING MYSQL

Count the number of Customers according to Response

```
SELECT
    COUNT(customer_id) AS Count_of_Customers, response
FROM
    RetailSalesData.sales_data_response
GROUP BY response;
```

Result Grid | Filter Rows:

Count_of_Customers	response
6237	0
647	1

DATA ANALYSIS USING MYSQL

Join both the tables

```
SELECT
    *
FROM
    sales_data_transactions
        JOIN
    sales_data_response ON sales_data_transactions.customer_id = sales_data_response.customer_id;
```

Result Grid | Filter Rows: | Export: | Wrap

customer_id	trans_date	trans_amount	customer_id	response
CS5295	11-Feb-13	35	CS5295	1
CS4768	15-Mar-15	39	CS4768	1
CS2122	26-Feb-13	52	CS2122	0
CS1217	16-Nov-11	99	CS1217	0
CS1850	20-Nov-13	78	CS1850	0

DATA ANALYSIS USING MYSQL

Highest Count and Sum of amount of Sales per customer id

```
SELECT
    customer_id,
    SUM(trans_amount) AS Sum_Amount,
    COUNT(trans_Amount) AS CountOfTransactions
FROM
    retailsalesdata.sales_data_transactions
GROUP BY customer_id
ORDER BY Sum_Amount DESC;
```

Result Grid | Filter Rows: | Expo

customer_id	Sum_Amount	CountOfTransactions
CS4424	2933	39
CS4320	2647	38
CS5752	2612	33
CS4660	2527	33
CS3799	2513	36

DATA ANALYSIS USING MYSQL

Top 5 customers based on the sum of trans_amount

```sql
SELECT
    customer_id, SUM(trans_amount) AS Total_Amount
FROM
    retailsalesdata.sales_data_transactions
GROUP BY customer_id
ORDER BY Total_Amount DESC limit 5;
```

Result Grid | Filter Rows:

customer_id	Total_Amount
CS4424	2933
CS4320	2647
CS5752	2612
CS4660	2527
CS3799	2513

DATA ANALYTICS

We analyzed a Retail store Data utilizing various tools such as Python, MS-Excel, and MySQL. This analysis provided us with diverse insights and visualizations that will aid in decision-making and enhance business perspectives from multiple angles.

"Data analytics is the discovery, interpretation, and communication of meaningful patterns in data to drive informed decision-making and strategic planning."

Thank you