RETAIL SALES DATA ANALYSIS

Internship Studio Project By Arya Bagde

PROJECT DESCRIPTION

This project involved analyzing retail sales data using SQL, Python, and Excel to uncover valuable business insights and understand customer behavior patterns. The primary goal was to evaluate customer transactions and their responses to marketing efforts in order to improve marketing effectiveness and identify key revenue-driven customer segments. By analyzing the data, we were able to extract actionable insights. These insights helped us understand customer engagement, spending habits, and campaign effectiveness, ultimately supporting data-driven business decisions.

PROJECT OVERVIEW

1. **Objectives:** Analyze customer transaction and response data to identify spending patterns, customer behavior, and campaign effectiveness.

2. **Dataset Contains:**
   - Retail_Data_Transactions: customer_id, trans_date, tran_amount
   - Retail_Data_Response: customer_id, response (0 or 1)

3. **Tool Used:**
   - SQL – Data cleaning & querying
   - Python – Advanced analysis (Time_Series, RFM, Cohort, Churn)
   - Excel – Final charts & Data Visualization

APPROACH

Data Cleaning, Preparation & Exploration

- Handled missing values, corrected data types, and standardized formats.
- Merged transactional and response data based on customer_id.

Data Analysis & Advanced Analysis

- Analyzed monthly revenue, order trends, top spender, Customer engagement levels, Response vs. Non-response customer behavior
- Performed advanced techniques such as:
    - **Time Series Trends**
    - **Churn Analysis**
    - **Cohort Analysis**
    - **RFM Analysis (Recency, Frequency, Monetary)**

SQL ANALYSIS

Database Setup & Data Loading

•Created database: RetailSalesData

•Created 2 tables:

- •Sales_Data_Transactions
- •Sales_Data_Response

•Loaded CSVs using LOAD DATA INFILE

```sql
CREATE DATABASE RetailSalesData;
Use RetailSalesData;

CREATE TABLE Sales_Data_Transactions(
customer_id VARCHAR(255),
trans_date VARCHAR(255),
tran_amount INT);

CREATE TABLE Sales_Data_Response(
customer_id VARCHAR(255),
response INT);

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Retail_Data_Transactions.csv'
INTO TABLE Sales_Data_Transactions
FIELDS terminated by ','
LINES terminated by '\n'
IGNORE 1 ROWS;

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Retail_Data_Response.csv'
INTO TABLE Sales_Data_Response
FIELDS terminated by ','
```

Output: Both datasets successfully loaded into tables.

SQL ANALYSIS

Data Cleaning:

Converted trans_date from string to proper DATE format to enable time-based analysis like monthly trends, yearly comparisons, and RFM calculations.

```sql
-- Changing Datatype of Trans_date column to date
-- Adding a new DATE column
ALTER TABLE Sales_Data_Transactions
ADD COLUMN trans_date_new DATE;

-- Converting text to date
UPDATE Sales_Data_Transactions
SET trans_date_new = STR_TO_DATE(trans_date, '%d-%b-%y');

SELECT trans_date AS old_txt,
trans_date_new  AS new_date
FROM   Sales_Data_Transactions
LIMIT  10;
```

Result Grid | Filter Rows

old_txt	new_date
11-Feb-13	2013-02-11
15-Mar-15	2015-03-15
26-Feb-13	2013-02-26
16-Nov-11	2011-11-16
20-Nov-13	2013-11-20
26-Mar-14	2014-03-26
06-Feb-12	2012-02-06
30-Jan-15	2015-01-30
08-Jan-13	2013-01-08
20-Aug-13	2013-08-20

```sql
-- Droping old column and Making the new column official one
ALTER TABLE Sales_Data_Transactions
DROP COLUMN trans_date,
CHANGE COLUMN trans_date_new trans_date DATE NOT NULL;
```

SQL ANALYSIS – DATA ANALYSIS

1. Row Count & Transaction Date Range

```sql
-- Row count & min / max dates
SELECT COUNT(*) AS total_row,
MIN(trans_date) AS first_txn,
MAX(trans_date) AS last_txn
FROM Sales_Data_Transactions;

-- Null / negative amounts
```

Result Grid | Filter Rows: | Export:

total_row	first_txn	last_txn
125000	2011-05-16	2015-03-16

Insight:

•Shows how many transactions are available in the dataset.

•MIN(trans_date) and MAX(trans_date) reveal the time range of the data.

•Useful for understanding coverage of historical trends (e.g., 2011–2015).

SQL ANALYSIS – DATA ANALYSIS

2. Check for Null or Invalid Amounts

```sql
-- Null / negative amounts
SELECT COUNT(*) AS null_values
FROM   Sales_Data_Transactions
WHERE  tran_amount IS NULL
OR  tran_amount < 0;
```

Result Grid | Filter Rows: | Export:

	null_values
▶	0

Insight:
Identifies data quality issues, such as missing or invalid transaction amounts. These may need to be cleaned or excluded from analysis.

SQL ANALYSIS – DATA ANALYSIS

3. Total Revenue

```sql
-- Aggregate revenue & order metrics
-- Total Revenue
SELECT SUM(tran_amount) AS total_revenue
FROM Sales_Data_Transactions;
```

Result Grid | Filter Rows: | Export: | Wrap Ce

total_revenue
8123989

Insight:

The total revenue generated across all transactions is **81,23,989**

SQL ANALYSIS – DATA ANALYSIS

4. Monthly Revenue and Order Volume

Insight:

•**Reveals seasonal trends –** months or years with higher/lower revenue.

•**Average Order Value (AOV**) helps evaluate spending behavior per purchase.

```sql
-- Total Revenue and Order By Month & Year
SELECT YEAR(trans_date)  AS yr,
       MONTH(trans_date) AS mon,
       COUNT(*)          AS orders,
       SUM(tran_amount)  AS revenue,
       AVG(tran_amount)  AS avg_order_value
FROM   Sales_Data_Transactions
GROUP  BY yr, mon
ORDER  BY yr, mon;
```

Result Grid | Filter Rows: | Export: | Wrap C

yr	mon	orders	revenue	avg_order_value
2011	5	1485	98951	66.6337
2011	6	2707	174527	64.4725
2011	7	2726	178097	65.3327
2011	8	2914	188631	64.7327
2011	9	2605	169173	64.9417
2011	10	2839	182634	64.3304
2011	11	2570	166921	64.9498
2011	12	2812	181405	64.5110
2012	1	2737	177987	65.0300
2012	2	2619	170135	64.9618
2012	3	2743	180453	65.7867
2012	4	2613	168000	64.2939
2012	5	2729	178880	65.5478
2012	6	2674	172933	64.6720

SQL ANALYSIS – DATA ANALYSIS

5. Customer-Level KPIs

Insights:

- **Customer Lifetime Value** (LTV): based on total revenue.
- **Engagement span**: difference between the first and last order.
- Helps in identifying loyal customers and creating segments.

```sql
-- Customer-level KPIs
-- Metrics per customer
SELECT customer_id,
COUNT(*) AS orders,
MIN(trans_date) AS first_order,
MAX(trans_date) AS last_order,
SUM(tran_amount) AS customer_revenue,
AVG(tran_amount) AS avg_order_value
FROM  Sales_Data_Transactions
GROUP BY customer_id;
```

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

customer_id	orders	first_order	last_order	customer_revenue	avg_order_value
CS1112	15	2011-06-15	2015-01-14	1012	67.4667
CS1113	20	2011-05-27	2015-02-09	1490	74.5000
CS1114	19	2011-07-14	2015-02-12	1432	75.3684
CS1115	22	2011-08-10	2015-03-05	1659	75.4091
CS1116	13	2011-06-27	2014-08-25	857	65.9231
CS1117	17	2011-05-20	2014-07-02	1185	69.7059
CS1118	15	2011-05-18	2015-03-14	1011	67.4000
CS1119	15	2012-02-28	2015-03-05	1158	77.2000
CS1120	24	2011-05-26	2015-03-06	1677	69.8750
CS1121	26	2011-05-30	2015-02-03	1524	58.6154
CS1122	16	2011-07-19	2015-02-02	1156	72.2500
CS1123	19	2011-05-26	2014-11-27	1331	70.0526

SQL ANALYSIS – DATA ANALYSIS

6. Top 10 Customers by Revenue

Insight:
The top 10 customers generate a large share of revenue. These are high-value customers and should be prioritized for loyalty programs and special offers.

```sql
-- Top 10 customers by revenue
SELECT customer_id,
SUM(tran_amount) AS revenue
FROM Sales_Data_Transactions
GROUP BY customer_id
ORDER BY revenue DESC
LIMIT 10;
```

Result Grid | Filter Rows:

customer_id	revenue
CS4424	2933
CS4320	2647
CS5752	2612
CS4660	2527
CS3799	2513
CS5109	2506
CS4074	2462
CS3805	2453
CS4608	2449
CS5555	2439

SQL ANALYSIS – DATA ANALYSIS

7. Create Merged View: v_sales_with_response

Insight:
Creates a unified table to analyze how customer transactions correlate with the response to marketing.

```sql
--  Merging Response Table And Creating View
CREATE OR REPLACE VIEW v_sales_with_response AS
SELECT t.customer_id,
t.trans_date,
t.tran_amount,
r.response
FROM Sales_Data_Transactions  AS t
LEFT JOIN Sales_Data_Response AS r
ON r.customer_id = t.customer_id;

SELECT * FROM v_sales_with_response;
```

Result Grid | Filter Rows: | Export: | Wrap C

customer_id	trans_date	tran_amount	response
CS5295	2013-02-11	35	1
CS4768	2015-03-15	39	1
CS2122	2013-02-26	52	0
CS1217	2011-11-16	99	0
CS1850	2013-11-20	78	0
CS5539	2014-03-26	81	0
CS2724	2012-02-06	93	0
CS5902	2015-01-30	89	0
CS6040	2013-01-08	76	0
CS3802	2013-08-20	75	1
CS3494	2013-07-02	94	0
CS3780	2013-03-25	80	0
CS1171	2012-11-03	59	0

SQL ANALYSIS – DATA ANALYSIS

8. Response Metrics Overview

```sql
-- Total Count Check
SELECT
COUNT(*) AS total_row,
SUM(response = 1) AS responses,
SUM(response = 0 OR response IS NULL) AS non_responses,
COUNT(DISTINCT customer_id) AS customers
```

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

total_row	responses	non_responses	customers
125000	13842	111158	6889

Insight:

•A total of **13,842 customers responded** to the marketing efforts (marked with response = 1).

•**111,158 customers did not respond** or had no recorded response (marked with response = 0 or NULL).

•This gives a **response rate** of approximately **11.07%** and a **non-response rate** of **88.93%**.

•The response rate is **relatively low,** indicating potential to improve marketing effectiveness.

SQL ANALYSIS – DATA ANALYSIS

9. Detailed Response Analysis

```sql
-- How many customers responded?
SELECT response,
COUNT(DISTINCT customer_id) AS customers,
COUNT(*) AS transactions,
SUM(tran_amount) AS revenue,
AVG(tran_amount) AS avg_order_value
FROM v_sales_with_response
GROUP BY response;
```

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

response	customers	transactions	revenue	avg_order_value
NULL	5	31	1611	51.9677
0	6237	111127	7166830	64.4922
1	647	13842	955548	69.0325

Insight:

- **Responders (1):**
  647 customers made **13,842 transactions**, generating **₹9.55L** with the **highest Avg Order Value ₹69.03.**
- **Non-Responders (0):**
  6,237 customers made **1.11L transactions**, generating **₹71.66L** with **AOV ₹64.49.**
- **NULL Responses:**
  Minimal impact (₹1,611 from 31 transactions) — can be ignored.
- Responders are fewer but more valuable. Focus marketing on high-frequency, high-value customers to boost ROI.

SQL ANALYSIS – DATA ANALYSIS

10. Year-wise Revenue Split by Response

Insights:

- **Responder revenue** increased consistently from **2011 to 2014**, peaking in **2014 (₹2.8L)**.
- **Non-responders** consistently contributed **more total revenue**, but **responders showed higher value per customer.**
- Both segments dropped in 2015, possibly due to **limited data** for that year.
- While non-responders bring more volume, **responders steadily drive growth,** indicating the effectiveness of targeted marketing over time.

```sql
-- Year-by-year revenue split
SELECT
YEAR(trans_date) AS yr,
SUM(CASE WHEN response = 1 THEN tran_amount END) AS rev_responder,
SUM(CASE WHEN response = 0 THEN tran_amount END) AS rev_nonresponder
FROM v_sales_with_response
GROUP BY yr
ORDER BY yr;
```

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

yr	rev_responder	rev_nonresponder
2011	139272	1200741
2012	226010	1889532
2013	270517	1866623
2014	280143	1814365
2015	39606	395569

SQL ANALYSIS – DATA ANALYSIS

11. RFM (Recency, Frequency, Monetary) Analysis

Insights:

- **Responders** purchase more frequently (**21.4 times**) and spend **more per customer** than non-responders.
- Their **average revenue** is **28% higher** than non-responders.
- **The null response group** has the lowest engagement - rare & low-value customers.
- **Recency** is high for all (due to older dataset), but responders show stronger overall engagement.
- Responders are **more loyal and profitable.** Targeting similar profiles can improve marketing success.

```sql
-- Recency/Frequency/Monetary (RFM) comparison
WITH rfm AS (
SELECT customer_id,
response,
MAX(trans_date) AS last_txn,
COUNT(*) AS freq,
SUM(tran_amount) AS monetary
FROM v_sales_with_response
GROUP BY customer_id, response
)
SELECT response,
AVG(DATEDIFF(CURDATE(), last_txn)) AS avg_recency_days,
AVG(freq) AS avg_frequency,
AVG(monetary) AS avg_monetary
FROM rfm
GROUP BY response;
```

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

response	avg_recency_days	avg_frequency	avg_monetary
0	3840.3975	17.8174	1149.0829
1	3847.0742	21.3941	1476.8903
NULL	4548.2000	6.2000	322.2000

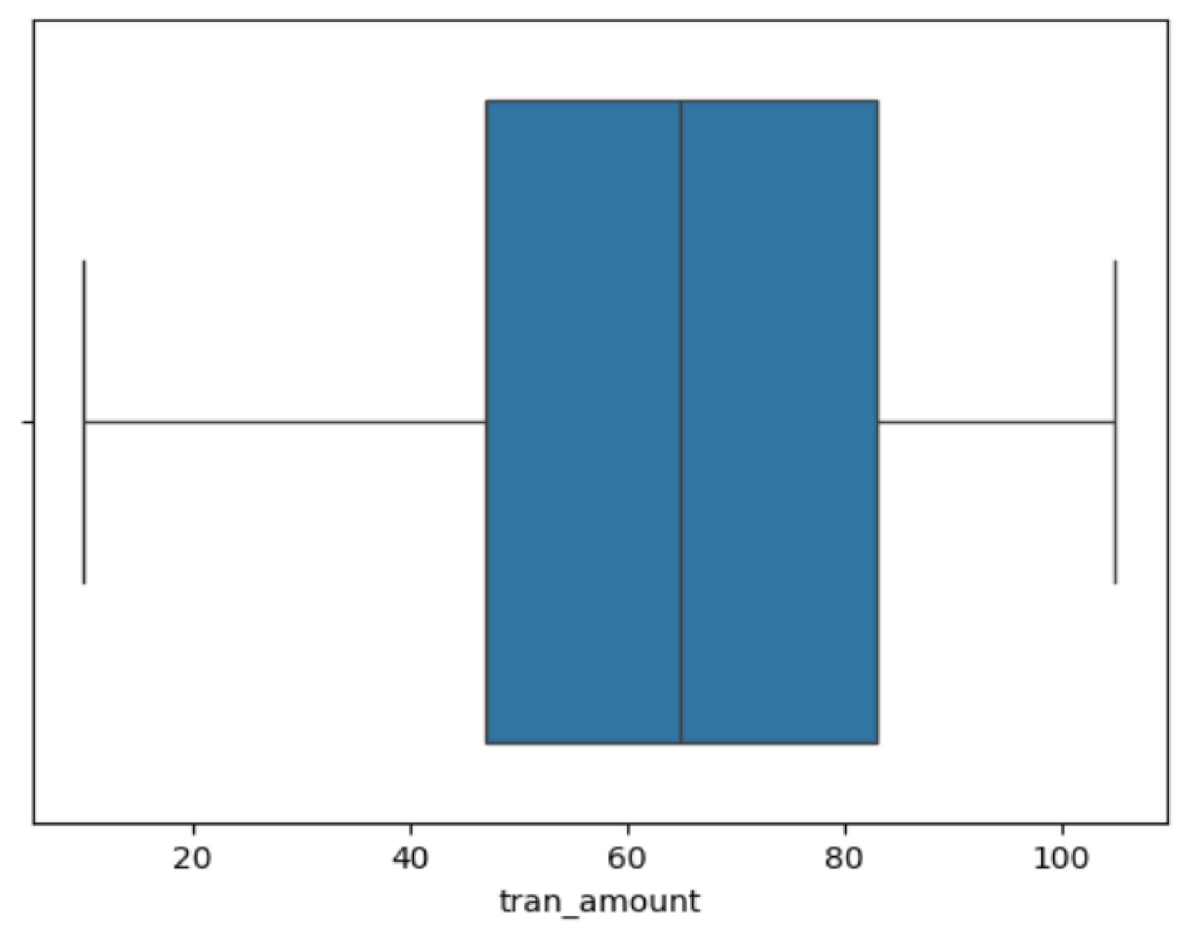
PYTHON ANALYSIS - DATA CLEANING &PREPARATION

- Imported necessary libraries: numpy, pandas, matplotlib, and seaborn.
- Loaded transaction and response data using pd.read_csv().
- Merged both datasets using a **left join** on customer_id.
- Checked for null values. In the response column (**31 nulls found**), I removed them using dropna().
- Converted:
    - trans_date to datetime format
    - response to int64 data type

PYTHON ANALYSIS - DATA EXPLORATION

1. Outlier Detection Using the Z-Score Method for Transaction Amount and Response.

```
16]: # Checking for outliers
     # Z-Score

     from scipy import stats

     # Calculating Z-Score
     z_score = np.abs(stats.zscore(df['tran_amount']))

     # set threshold
     threshold = 3

     outliers = z_score>threshold
     print(df[outliers])

     Empty DataFrame
     Columns: [customer_id, trans_date, tran_amount, response]
     Index: []

17]: sns.boxplot(x = df['tran_amount'])
     plt.show()
```



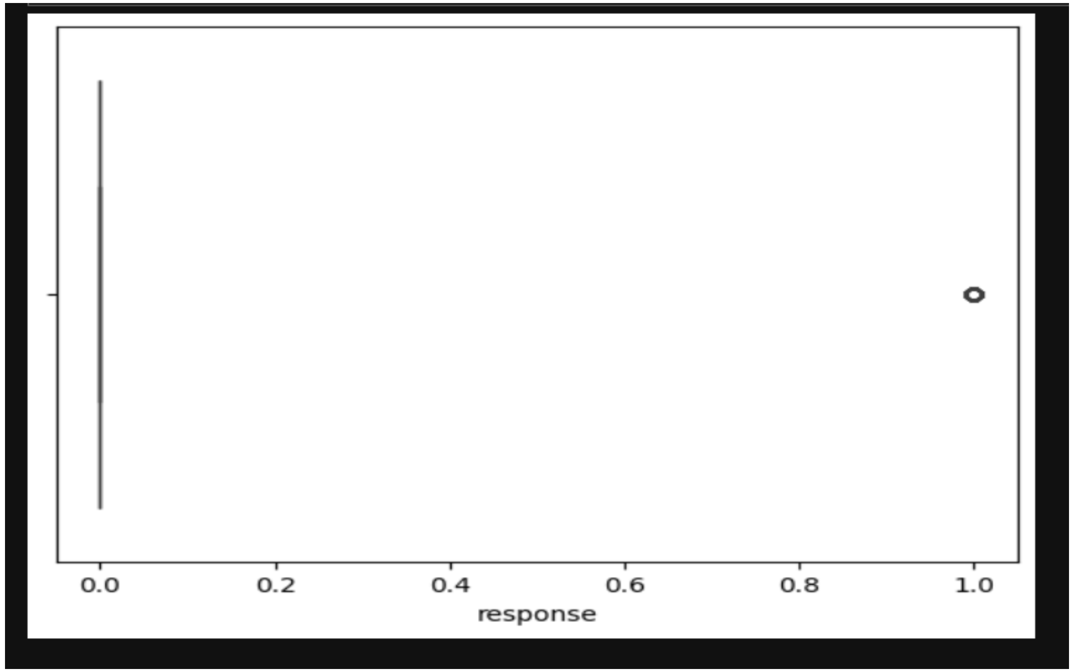
PYTHON ANALYSIS - DATA EXPLORATION

1. Outlier Detection Using the Z-Score Method for Transaction Amount and Response.

```
# Checking for outliers on Response
z_score = np.abs(stats.zscore(df['response']))

threshold = 3

outliers = z_score>threshold
print(df[outliers])
```

```
Empty DataFrame
Columns: [customer_id, trans_date, tran_amount, response]
Index: []
```

```
sns.boxplot(x = df['response'])
plt.show()
```



Insight:

•Both **tran_amount** and **response** showed **no significant outliers** (Z-score analysis and boxplots confirm clean distribution).

•The data is reliable for further analysis.

PYTHON ANALYSIS - DATA EXPLORATION

2. Monthly Transaction Trend

```
[22]: # Which 3 months have had the highest transaction amounts?

monthly_Sales = df.groupby('month')['tran_amount'].sum()
monthly_Sales = monthly_Sales.sort_values(ascending=False).reset_index().head(3)
monthly_Sales
```

[22]:

	month	tran_amount
0	8	726775
1	10	725058
2	1	724089

Insight: Months **August, October**, and **January** saw the **highest sales,** possibly due to seasonal or promotional campaigns.

PYTHON ANALYSIS - DATA EXPLORATION

3. Customers with Highest Order Counts

23]:

```
# Customers having highest number of orders

customer_counts = df['customer_id'].value_counts().reset_index()
customer_counts.columns =['customer_id','count']

# Sorting Top 5 Customers

top_5_cust = customer_counts.sort_values(by='count', ascending=False).head(5)
top_5_cust
```

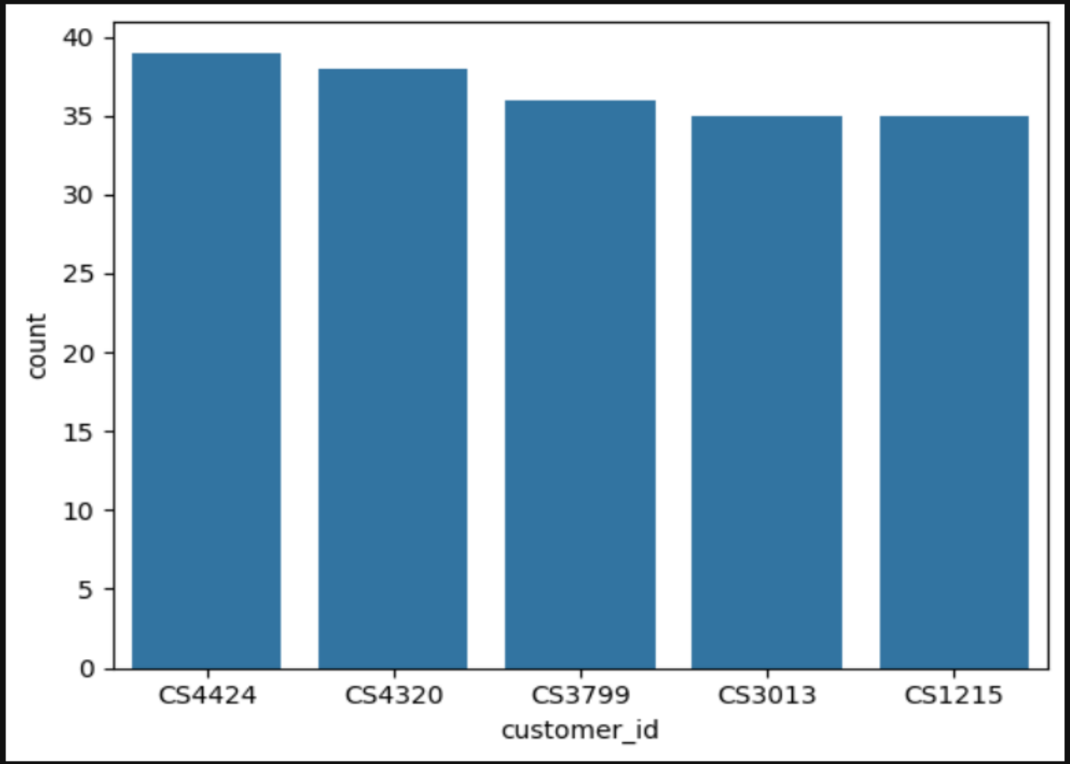
23]:

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

24]:

```
sns.barplot(x = 'customer_id', y = 'count', data = top_5_cust)
```

```
[24]:  <Axes: xlabel='customer_id', ylabel='count'>
```



Insight: These customers are **highly engaged.** Ideal for loyalty or premium campaigns.

PYTHON ANALYSIS - DATA EXPLORATION

4. Customers with Highest Total Spending

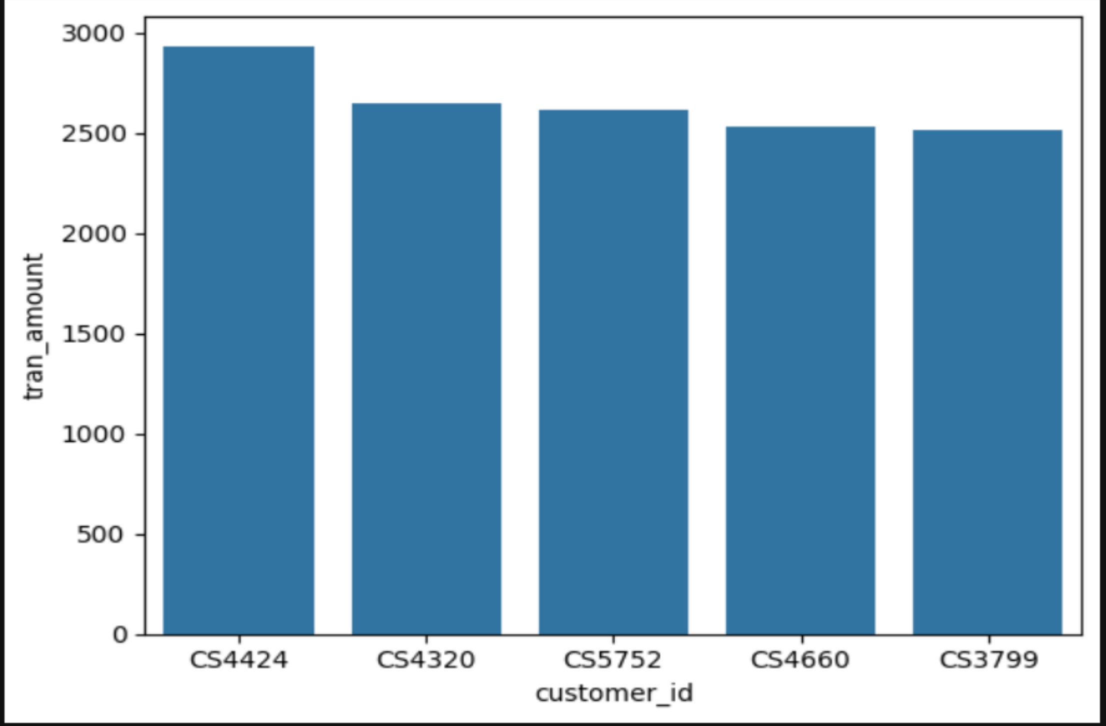
[25]:
```
# Customers having highest value of orders

customer_sales= df.groupby('customer_id')['tran_amount'].sum().reset_index()
customer_sales

# Sorting Top 5 Sales

top_5_sales= customer_sales.sort_values(by='tran_amount', ascending=False).head(5)
top_5_sales
```

[25]:

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

[26]:
```
sns.barplot(x = 'customer_id', y = 'tran_amount', data = top_5_sales)
```

6]: <Axes: xlabel='customer_id', ylabel='tran_amount'>



Insight: These customers are **top revenue contributors.**

PYTHON ANALYSIS – ADVANCED ANALYSIS

1. Time Series Analysis

Time Series Analysis

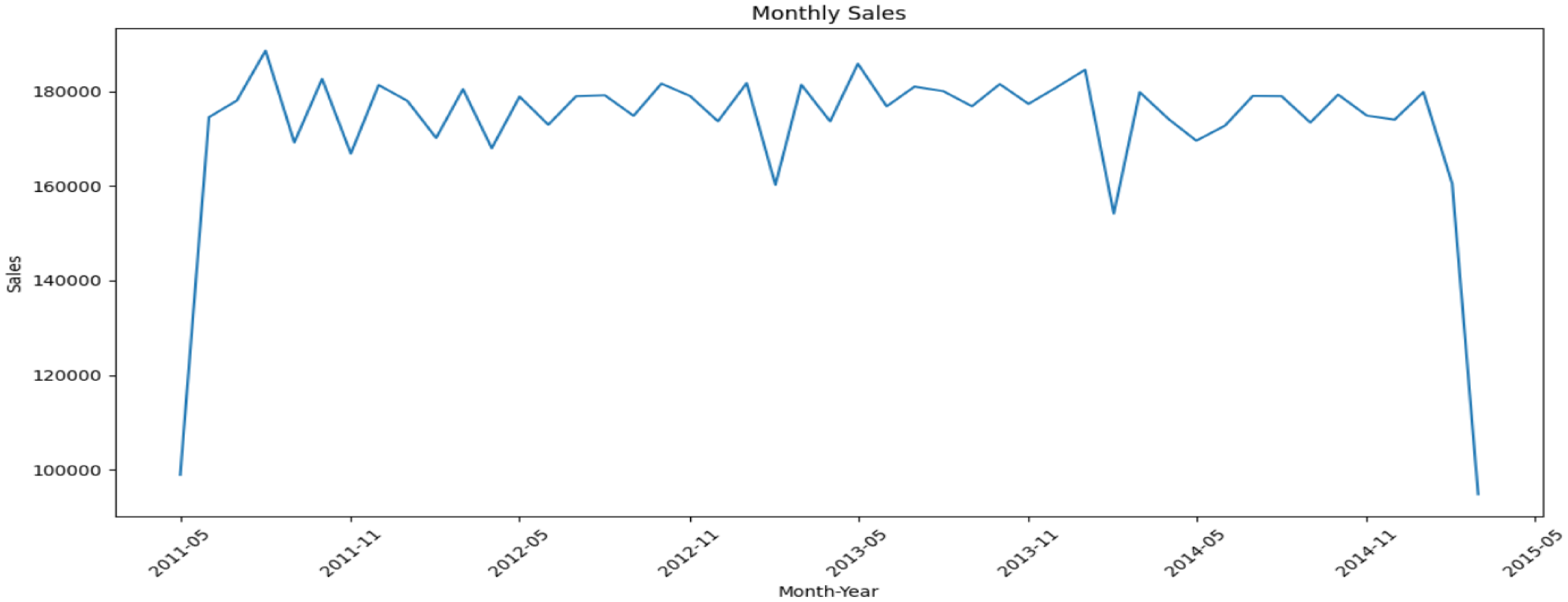
```
import matplotlib.dates as mdates

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

# Convert the PeriodIndex to DateTimeIndex
monthly_sales.index = monthly_sales.index.to_timestamp()

plt.figure(figsize=(12,6))  # Increase the size of the figure
plt.plot(monthly_sales.index, monthly_sales.values)  # Plot the data
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))  # Format the x-axis labels
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=6))  # Set the x-axis interval
plt.xlabel('Month-Year')
plt.ylabel('Sales')
plt.title('Monthly Sales')
plt.xticks(rotation=45)  # Rotate the x-axis labels
plt.tight_layout()  # Adjust the layout for better visibility
plt.show()
```

Insight:

- Sales remained **consistently high** across most months from 2011 to 2015.
- A few **noticeable dips** suggest possible data gaps or off-season periods.
- Overall, the trend shows **stable revenue flow**, with **minor fluctuations** throughout the timeline.

PYTHON ANALYSIS – ADVANCED ANALYSIS

1. Time Series Analysis



PYTHON ANALYSIS – ADVANCED ANALYSIS

2. Cohort Segmentation

Cohort Segmentation

```
# Recency will be the maximum of trans_date
recency = df.groupby('customer_id')['trans_date'].max()

# Frequency will be the count of transactions
frequency = df.groupby('customer_id')['trans_date'].count()

# Monetary will be the sum of tran_amount
monetary = df.groupby('customer_id')['tran_amount'].sum()

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

```
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 < row['monetary'] <= 1000):
        return 'P1'
    else:
        return 'P2'

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

PYTHON ANALYSIS – ADVANCED ANALYSIS

2. Cohort Segmentation

Insight:

- Customers were segmented into **P0, P1, and P2** based on their **recency, frequency, and monetary value.**
- **P0**: Most active and high-value customers (recent, frequent, high spenders).
- **P1**: Moderately active customers with average spend and engagement.
- **P2**: Older or low-engagement customers with lower activity and spending.
- This segmentation helps **target the right customer group** with personalized campaigns and retention strategies.

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

```
set(rfm['Segment'])
```

```
{'P0', 'P2'}
```

PYTHON ANALYSIS – ADVANCED ANALYSIS

3. Churn Analysis

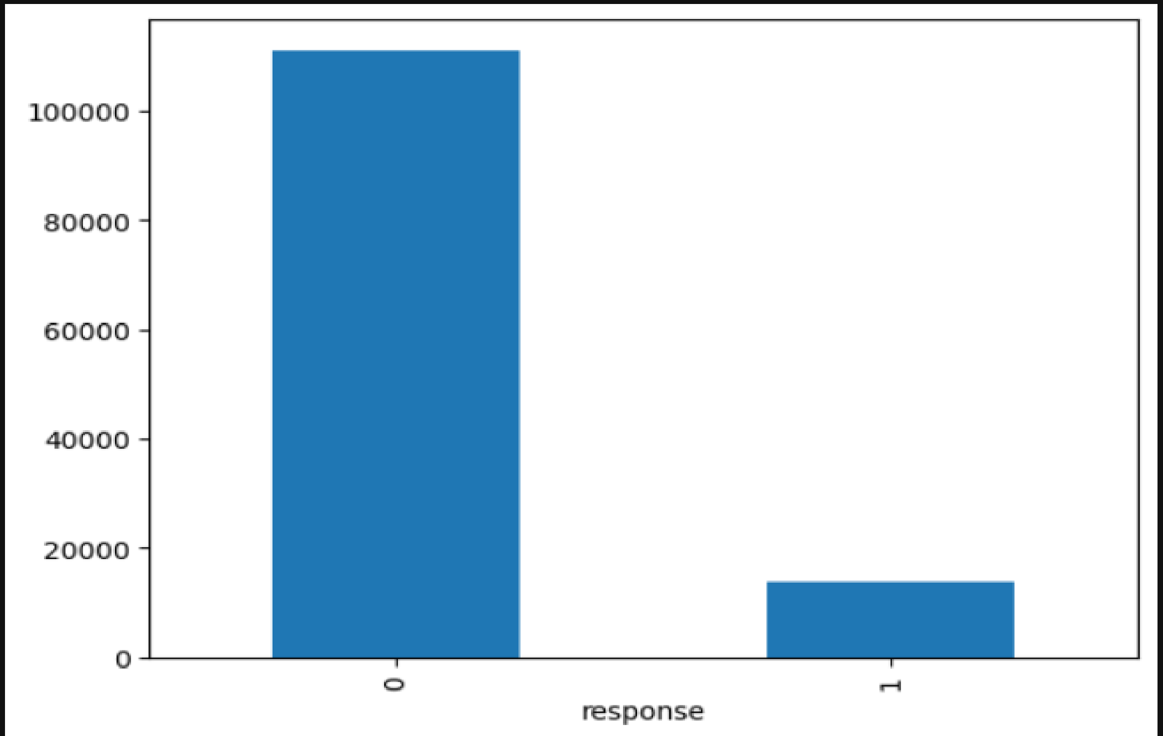
Insight:

- A majority of customers (**response = 0**) have **churned or did not engage** with the campaign.
- Only a small portion (**response = 1**) remained **active or responded.**
- This indicates a **high churn rate.**
- Highlights the need for **better re-engagement strategies.**

Churn Analysis

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

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

```
5]: <Axes: xlabel='response'>
```



PYTHON ANALYSIS – ADVANCED ANALYSIS

4. Analyzing top customers

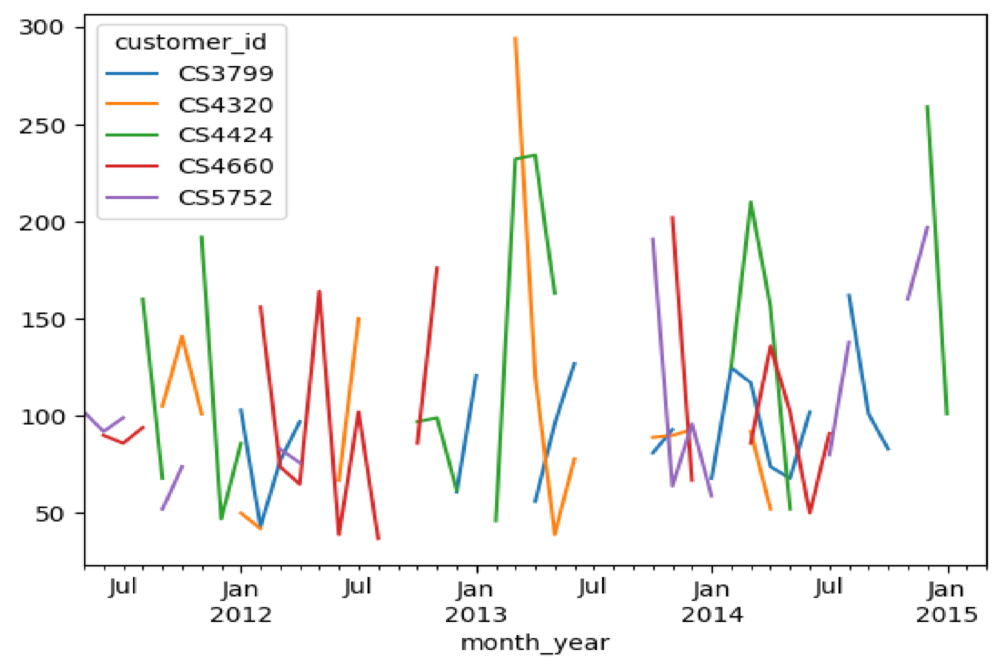
Insight:

- The top 5 customers (e.g., **CS4424, CS4320**) show **high and repeated monthly transactions**, indicating strong engagement.
- **CS4424** and **CS5752** show the **most consistent and high spending patterns** over time.
- Monthly sales fluctuate, but these customers contribute **significantly across multiple periods.**
- These are **loyal, high-value customers** ideal for **premium offers, loyalty rewards, and retention campaigns.**

Analyzing top customers

```python
# Top 5 customers
top_5_customers = monetary.sort_values(ascending=False).head(5).index

# Filter transactions of top 5 customers
top_customers_df = df[df['customer_id'].isin(top_5_customers)]

# Plot their monthly sales
top_customers_sales = top_customers_df.groupby(['customer_id', 'month_year'])['tran_amount'].sum().unstack(level=0)
top_customers_sales.plot(kind ='line')
```

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

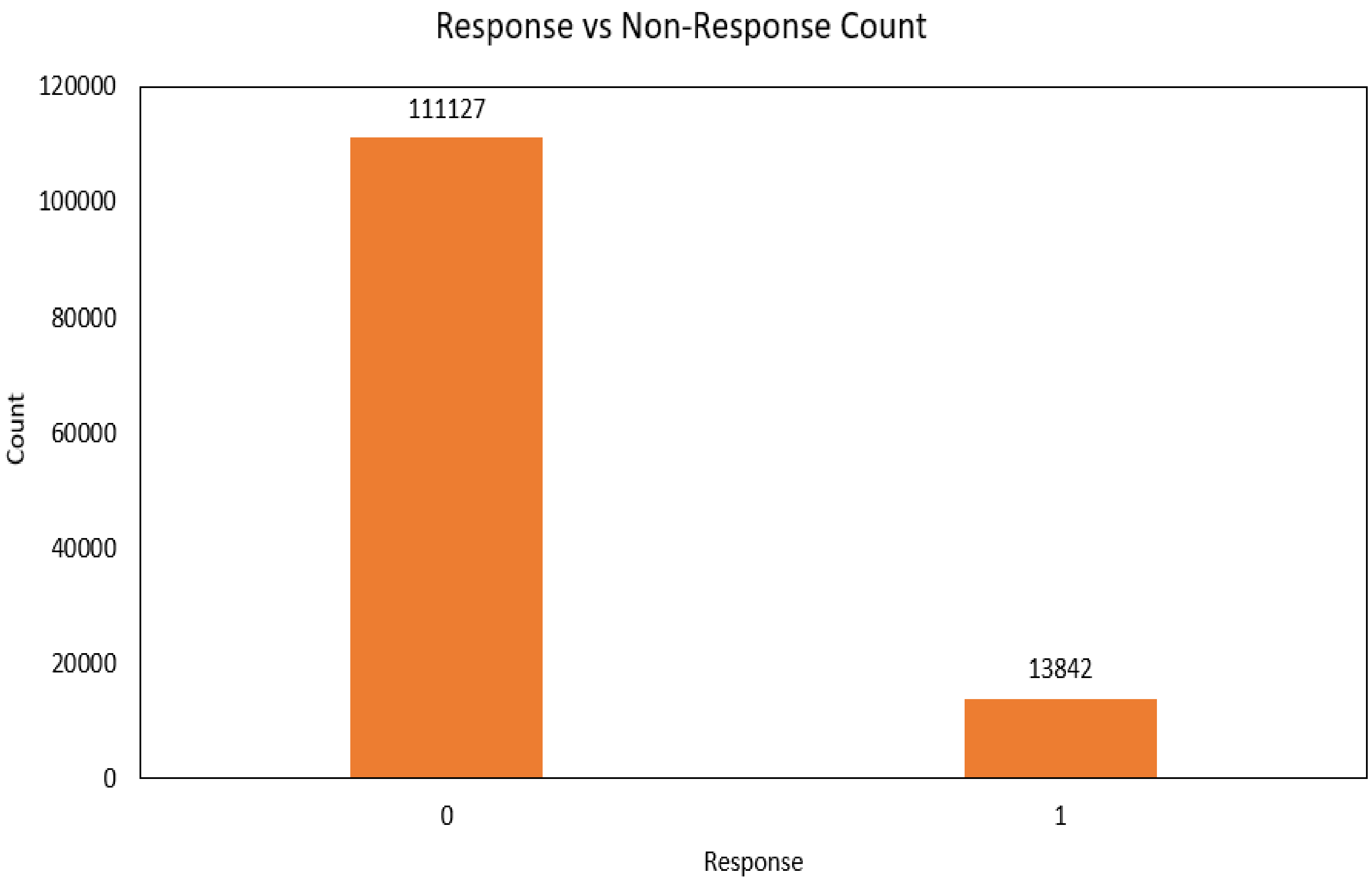


EXCEL ANALYSIS

1. Response VS Non-Response

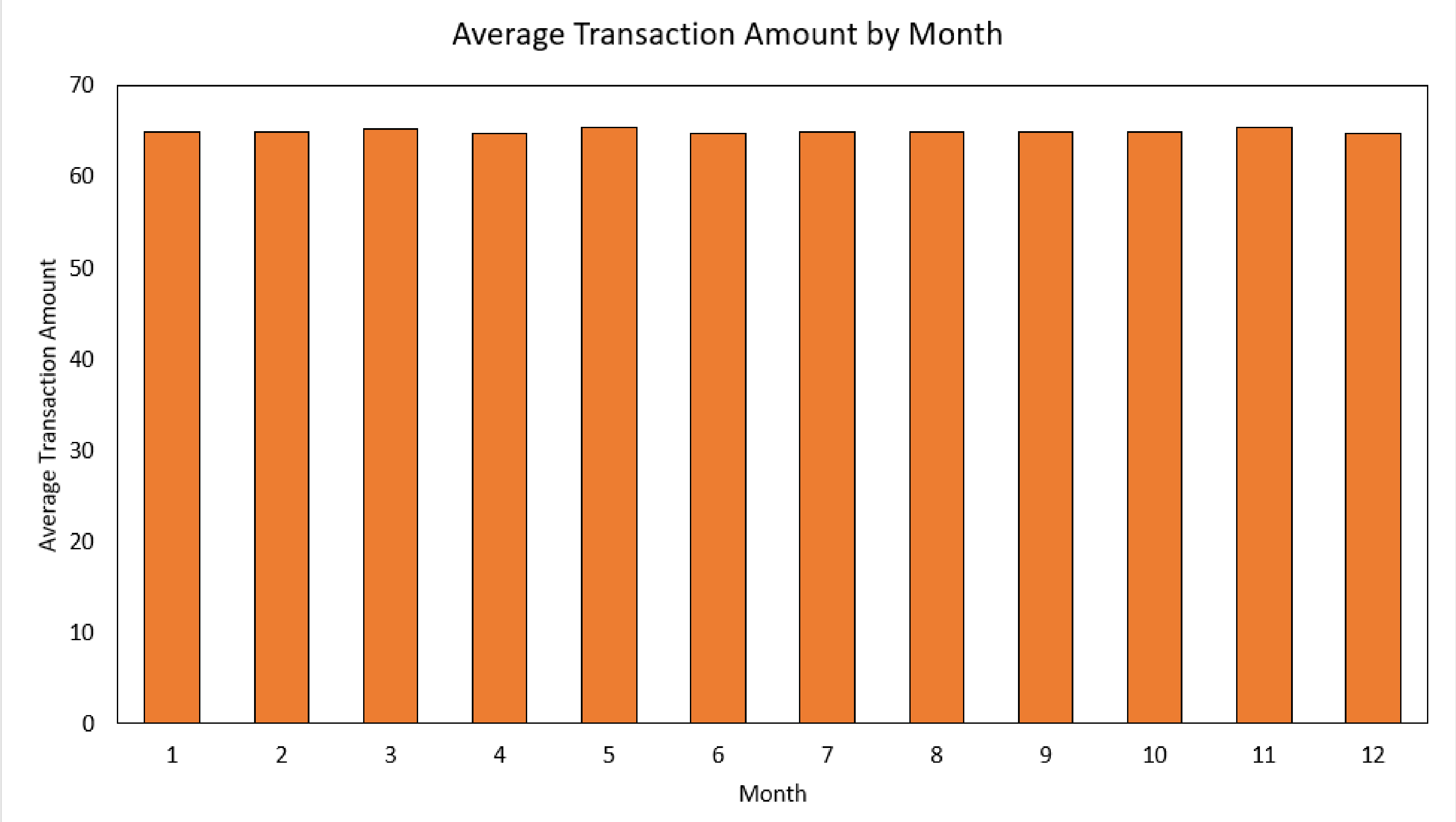
Insight:

- **Non-Responders (0):** 111,127 customers (≈ 89%)
- **Responders (1):** 13,842 customers (≈ 11%)
- The campaign had a **low response rate (~11%),** indicating a need for **improved targeting or personalization** to increase customer engagement.



EXCEL ANALYSIS

2. Monthly Average Transaction



Insight:

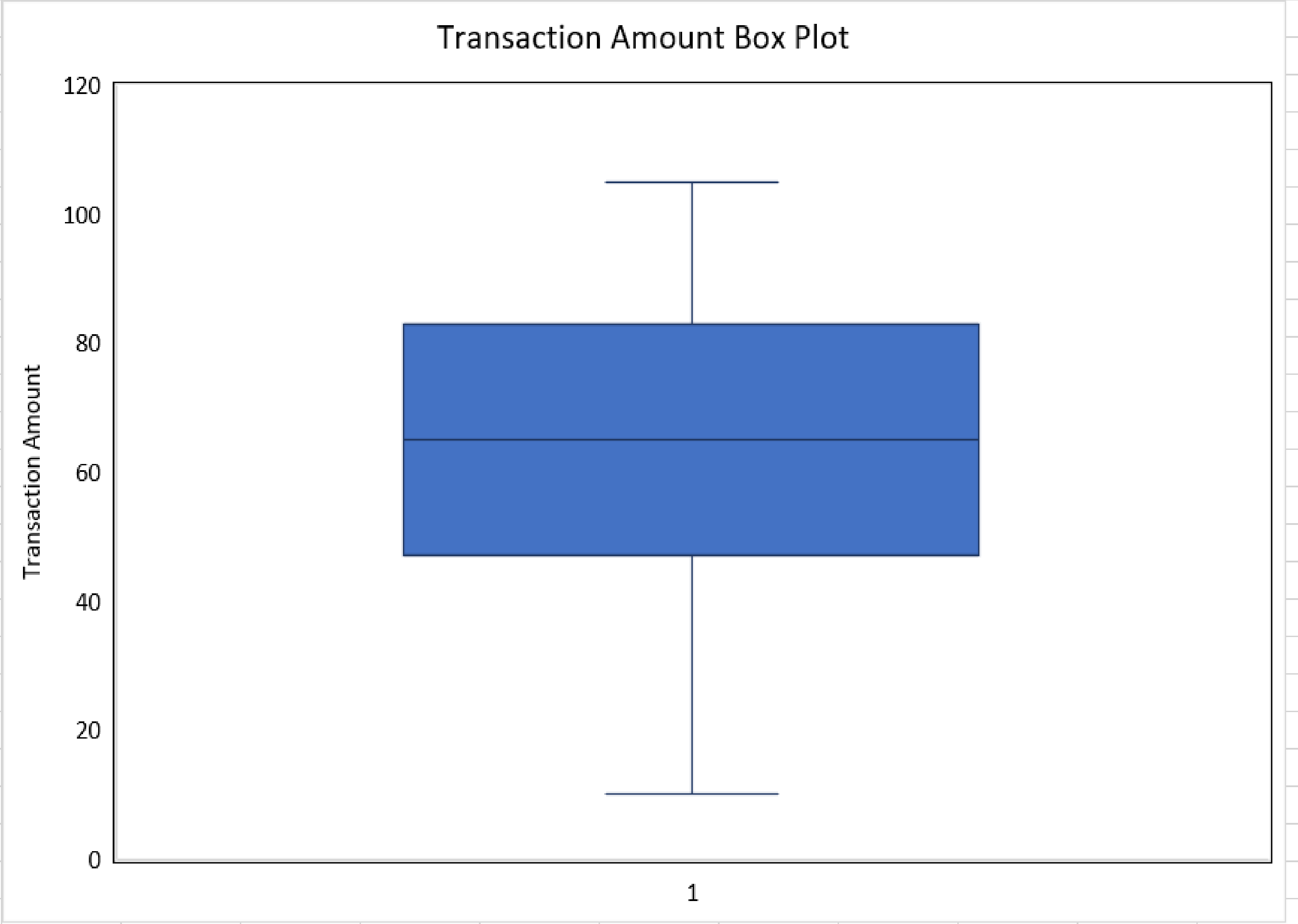
- The **average transaction amount** is fairly consistent across all months, ranging around **₹65.**
- **November (Month 11)** recorded the **highest average** at **₹65.43.**
- This suggests **stable customer spending behavior** throughout the year, with a slight peak in the festive season.

EXCEL ANALYSIS

3. Transaction Amount Box Plot

Insight:

- Created a box plot of the **transaction amount** column.
- Calculated values manually and confirmed that there are **no outliers.**
- The transaction data is **evenly distributed**, with **no extreme or unusual values.**

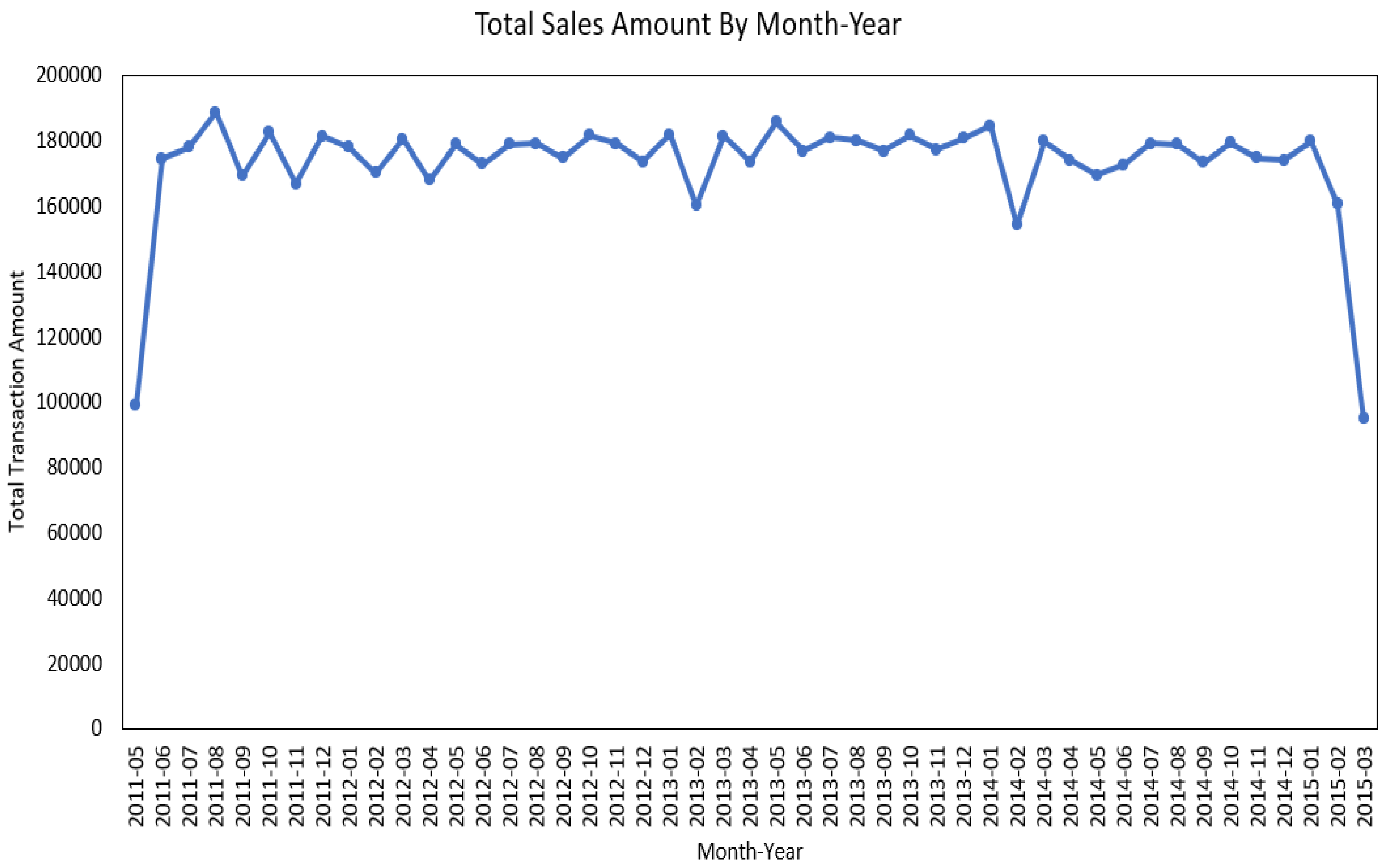


EXCEL ANALYSIS

4. Time Series Analysis

Insight:

- Analyzed monthly transaction trends over multiple years.
- Sales remained **stable with small fluctuations** across the timeline.
- Observed **seasonal spikes** in a few months like **August, October, and January**.
- No sharp drops or anomalies, indicating a **healthy and consistent sales pattern.**
- Useful for **sales forecasting** and **inventory planning** based on historical trends.

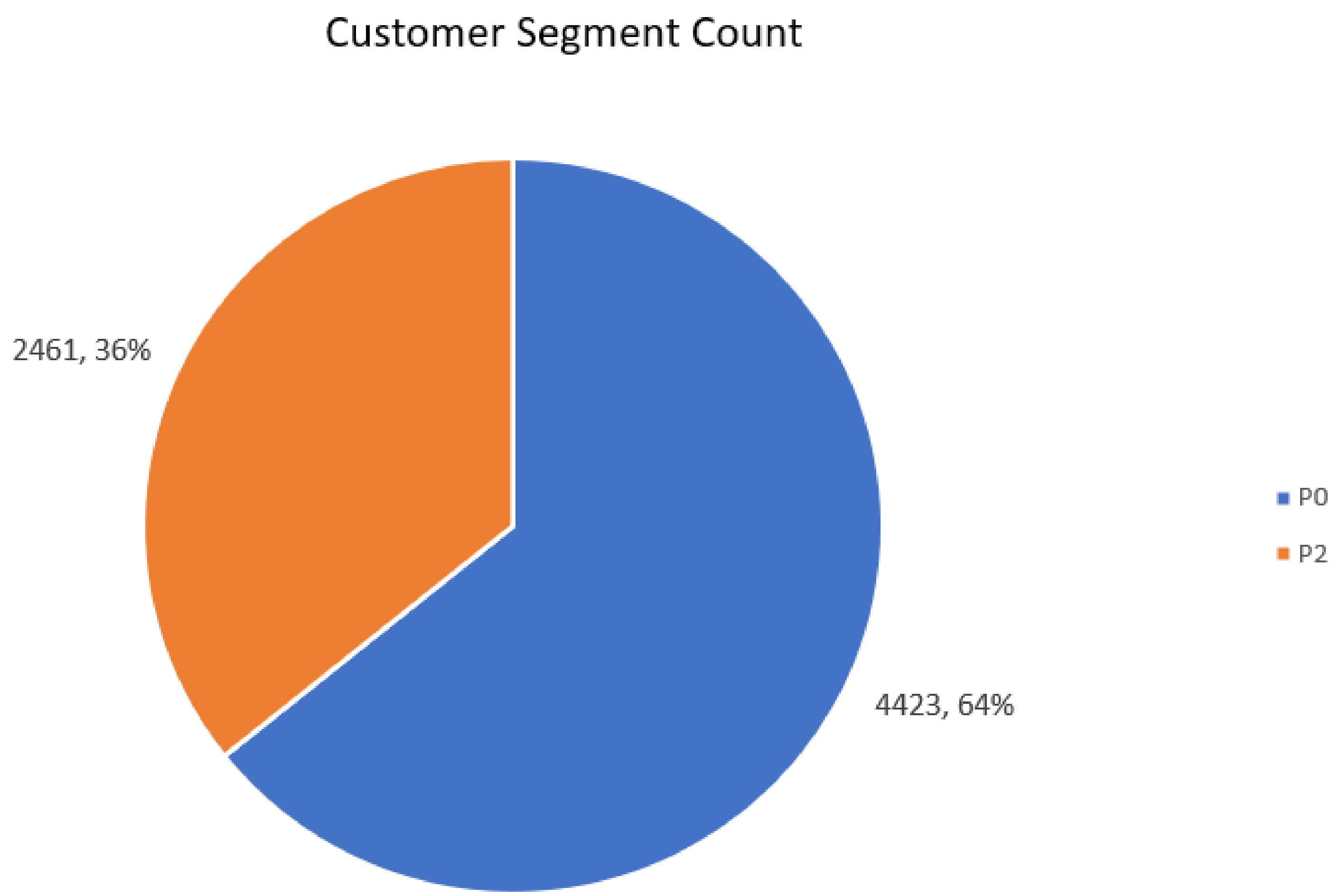


EXCEL ANALYSIS

5. Customer Segmentation

Insight:

- **P0 Segment (High-Value Customers):**
  4,423 customers (64%) — frequent, recent, and high spenders.
- **P2 Segment (Low Engagement Customers):**
  2,461 customers (36%) — older transactions or lower spending.

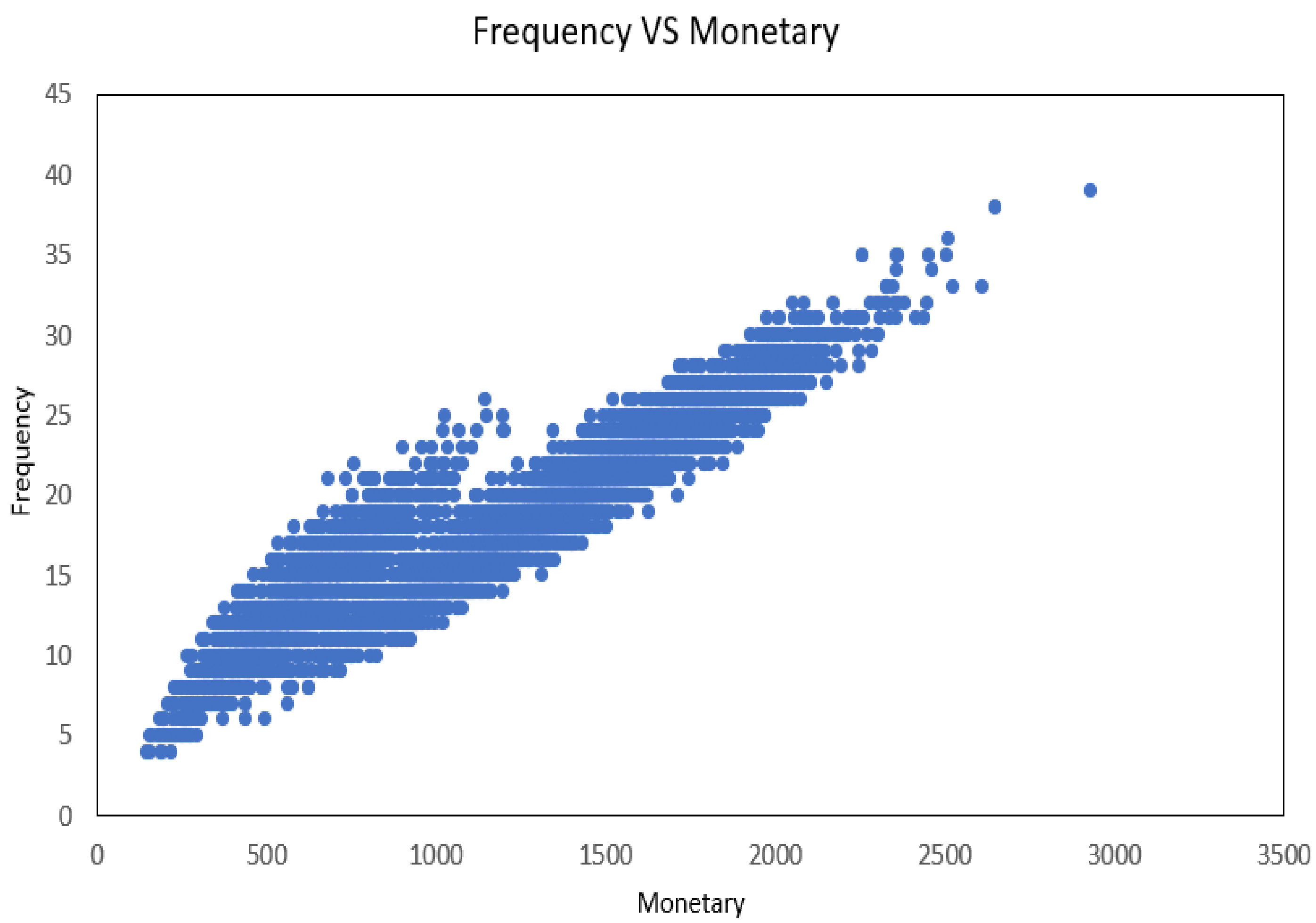


EXCEL ANALYSIS

6. Frequency vs Monetary

Insight:

- The plot shows a **strong positive correlation:** Customers who purchase **more frequently** also tend to **spend more** overall.
- Indicates a **valuable customer base** where increasing frequency can directly boost revenue.
- No major outliers — data is well-aligned and consistent.



DRIVE LINK

Dataset –

https://drive.google.com/file/d/1T4ah099BRXgbB5-amdsqCVZarGyAQjDm/view?usp=sharing

SQL File –

https://drive.google.com/file/d/1-zrh30_HOvDKbtQsr0EtvqDVpDo0qgq0/view?usp=sharing

Python - https://drive.google.com/file/d/1ipWH6KLtKVS5qSRR5RR3gXSaDbb7y02w/view?usp=sharing

Excel - https://docs.google.com/spreadsheets/d/13qrnjcDafN3UO_T8aQdfAZzUprZTiR4c/edit?usp=sharing&ouid=116931277368003559920&rtpof=true&sd=true

https://docs.google.com/spreadsheets/d/1a7j3tyO_fKxiq3q7-mZ8GB8kuHfywaPV/edit?usp=sharing&ouid=116931277368003559920&rtpof=true&sd=true