SQL Code

```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) PRIMARY KEY,
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;

SELECT * FROM Sales_Data_Transactions LIMIT 10;

EXPLAIN SELECT * FROM Sales_Data_Transactions WHERE customer_id = 'CS5295';

CREATE INDEX idx_id ON Sales_Data_Transactions(CUSTOMER_ID);

EXPLAIN SELECT * FROM Sales_Data_Transactions WHERE customer_id = 'CS5295';
```

sales-data-analysis

February 8, 2026

[1]:
```
import pandas as pd
```

[2]:
```
trxn= pd.read_csv('Retail_Data_Transactions.csv')
```

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

[4]:
```
response
```

[4]:
```
     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 x 2 columns]
```

[5]:
```
df= trxn.merge(response, on='customer_id', how='left')
```

[6]:
```
df
```

[6]:
```
        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 x 4 columns]
```

[7]:
```
df.dtypes
```

[7]:
```
customer_id     object
trans_date      object
tran_amount      int64
response       float64
dtype: object
```

[8]:
```
df.shape
```

[8]:
```
(125000, 4)
```

[9]:
```
df.head()
```

[9]:
```
  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
```

[10]:
```
df.tail()
```

[10]:
```
       customer_id trans_date  tran_amount  response
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
```

[11]:
```
df.describe()
```

[11]:
```
         tran_amount       response
count  125000.000000  124969.000000
mean       64.991912       0.110763
std        22.860006       0.313840
min        10.000000       0.000000
25%        47.000000       0.000000
50%        65.000000       0.000000
75%        83.000000       0.000000
max       105.000000       1.000000
```

[12]:
```
#MISSING VALUES:
df.isnull().sum()
```

[12]:
```
customer_id     0
trans_date      0
tran_amount     0
response       31
dtype: int64
```

[13]:
```
df= df.dropna()
```

[1]:
```
#CHANGE DTYPES:
df['trans_date']= pd.to_datetime(df['trans_date'])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[1], line 2
      1 #CHANGE DTYPES:
----> 2 df['trans_date']= pd.to_datetime(df['trans_date'])

NameError: name 'pd' is not defined
```

[15]:
```
df['response']= df['response'].astype('int64')
```

```
C:\Users\ashok\AppData\Local\Temp\ipykernel_22120\3326164836.py:1:
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['response']= df['response'].astype('int64')
```

[16]:
```
df.dtypes
```

[16]:
```
customer_id            object
trans_date     datetime64[ns]
tran_amount             int64
response                int64
dtype: object
```

[17]:
```
#check for outliers:
#z_score:

from scipy import stats
import numpy as np
```

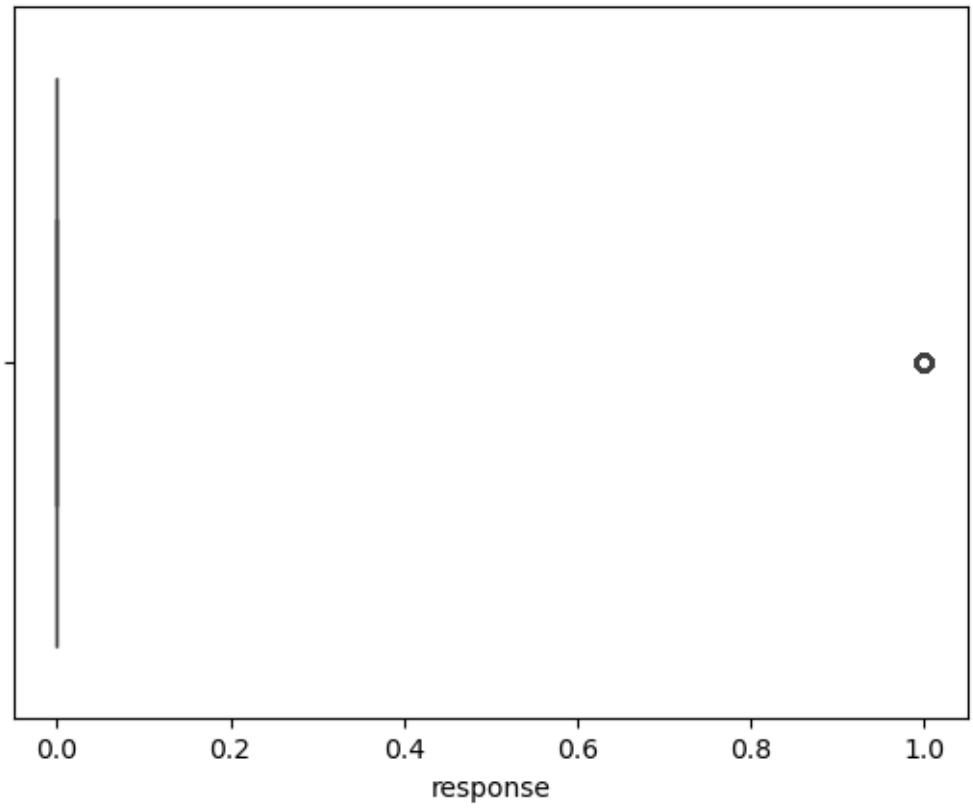
```
# calc z_score
z_score = np.abs(stats.zscore(df['tran_amount']))

#set a threshold:
threshold = 3
outliers = z_score > threshold
print(a[outliers])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[17], line 12
      10 threshold = 3
      11 outliers = z_score > threshold
---> 12 print(a[outliers])

NameError: name 'a' is not defined
```

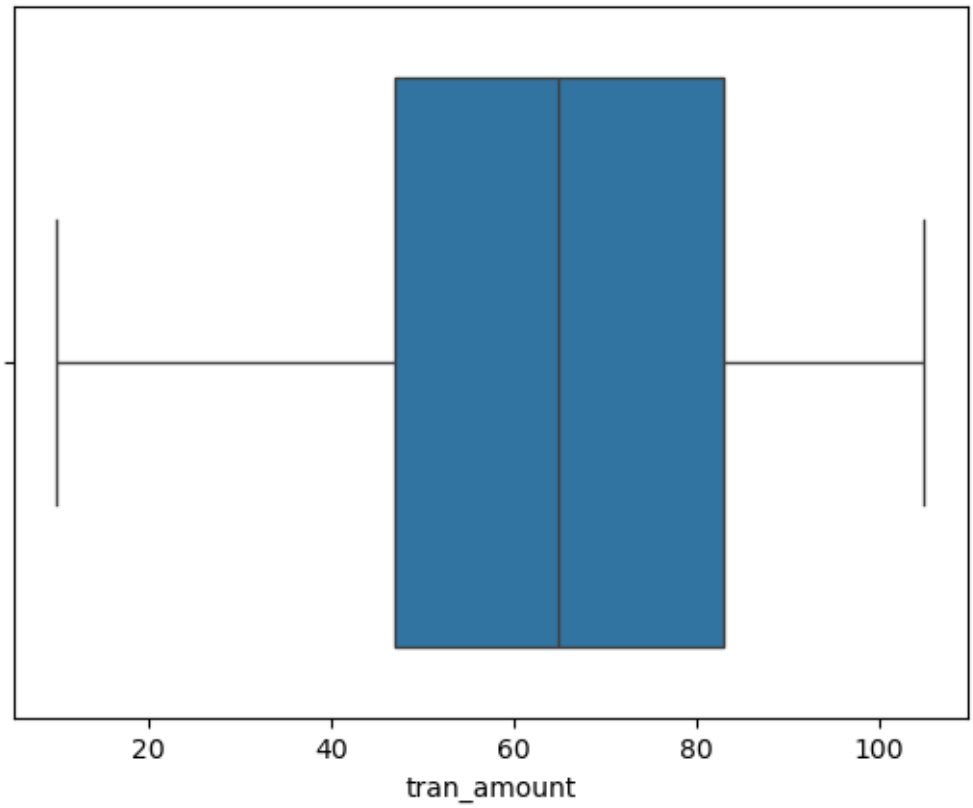
[18]:
```
#check for outliers:
#z_score:

from scipy import stats
import numpy as np
# calc z_score
z_score = np.abs(stats.zscore(df['response']))

#set a threshold:
threshold = 3
outliers = z_score > threshold
print(a[outliers])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[18], line 12
      10 threshold = 3
      11 outliers = z_score > threshold
---> 12 print(a[outliers])

NameError: name 'a' is not defined
```

[19]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(x=df['response'])
plt.show()
```



[20]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(x=df['tran_amount'])
plt.show()
```



[21]:
```
#creating a new columns:
df['month'] = df['trans_date'].dt.month
```

```
C:\Users\ashok\AppData\Local\Temp\ipykernel_22120\159651381.py:2:
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'] = df['trans_date'].dt.month
```

[22]:
```
#which 3 months have had the highest transaction amount?

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
```

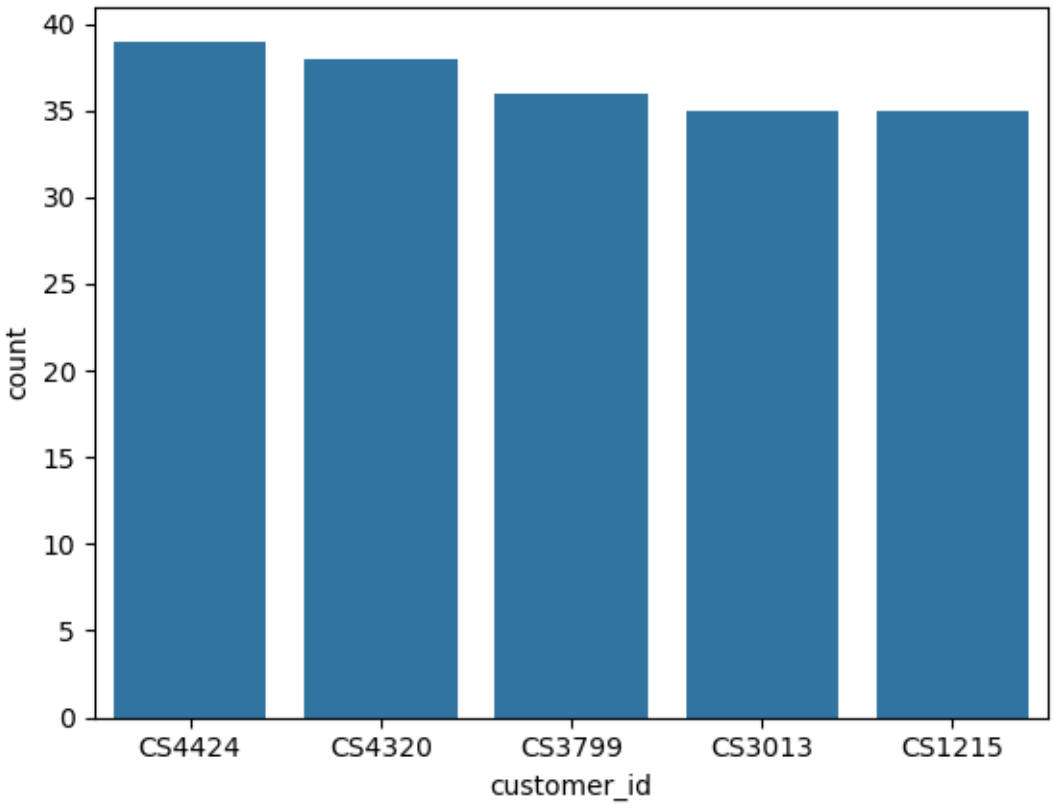
[23]:
```
#customers having highest num of orders.

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

#sort

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

[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_cus)
```

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



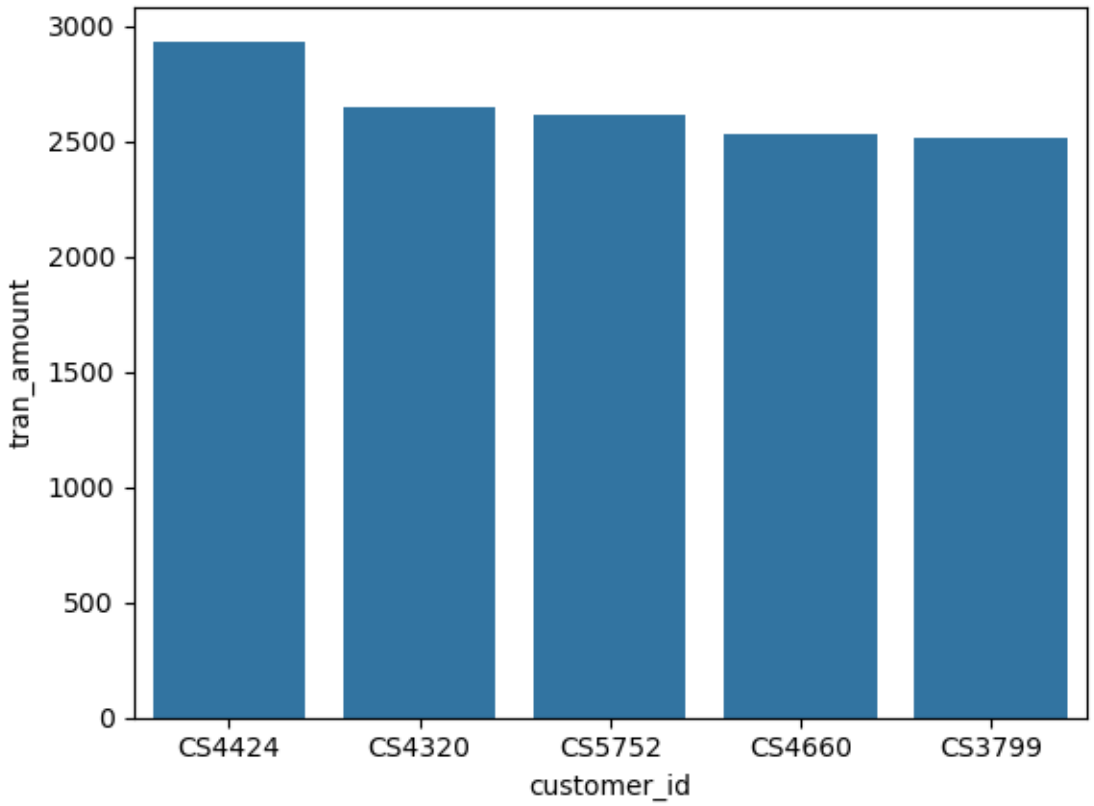
[25]:
```
#customers having highest value of orders.

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

#sort

top_5_sal = customer_sales.sort_values(by='tran_amount', ascending=False).
 ↪head(5)
top_5_sal
```

[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_sal)
```

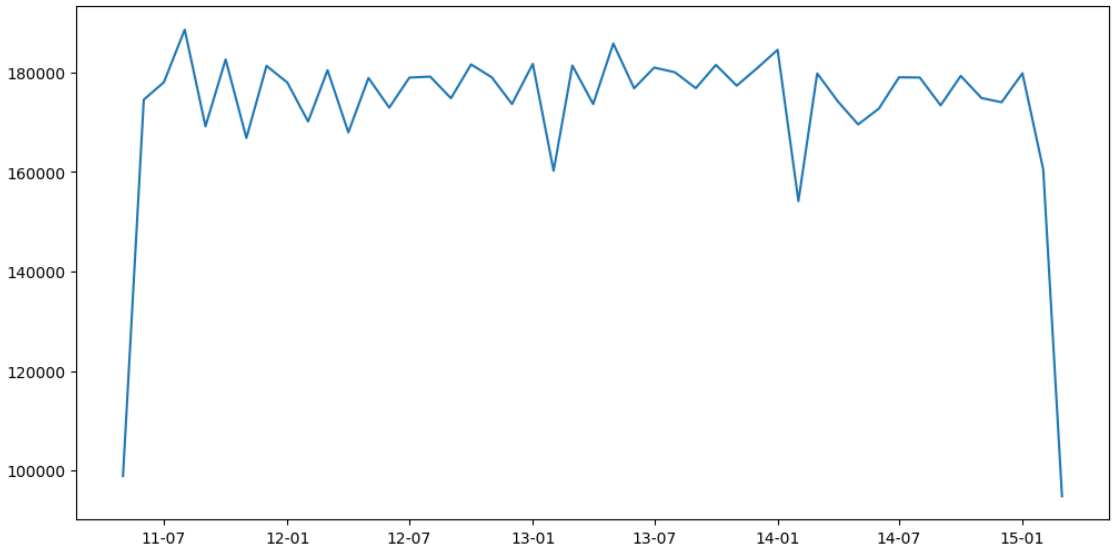
[26]:
```
<Axes: xlabel='customer_id', ylabel='tran_amount'>
```



[27]:
```
### 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()

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

```
C:\Users\ashok\AppData\Local\Temp\ipykernel_22120\191149756.py:4:
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')
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[27], line 13
     10 plt.plot(monthly_sales.index, monthly_sales.values)
     12 plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%y-%m'))
---> 13 plt.gca().xaxis.set_major_locator(mdates.monthlocator(interval = 6))
     14 plt.xlabel('Month-Year')
     15 plt.ylabel('Sales')

AttributeError: module 'matplotlib.dates' has no attribute 'monthlocator'
```



[28]:
```
df
```

[28]:

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

[124969 rows x 6 columns]

[30]:
```
# COHORT SEGMENTATION:

#recency:
recency = df.groupby('customer_id')['trans_date'].max()

# frequency:
frequency = df.groupby('customer_id')['trans_date'].count()

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

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

[31]:
```
rfm
```

[31]:
```
                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 x 3 columns]
```

[32]:
```
# 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<=row['monetary']<=1000):
        return 'P1'
    else:
        return 'P2'
rfm['Segment'] = rfm.apply(segment_customer, axis=1)
```

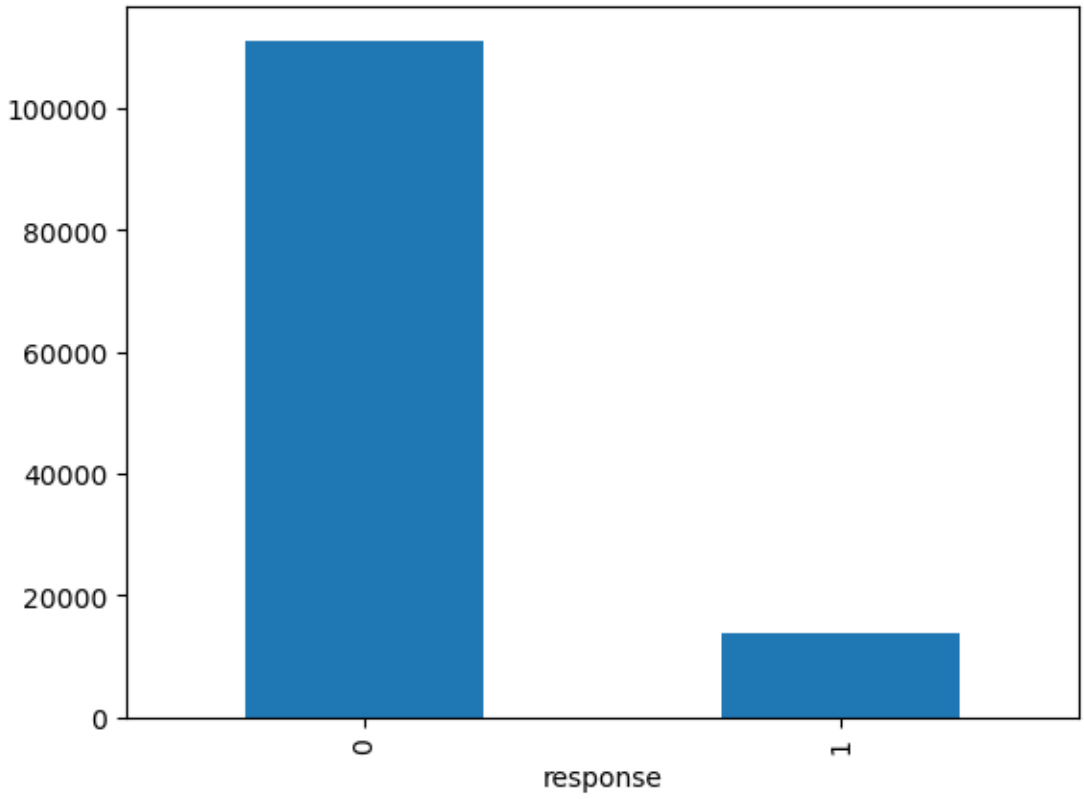
[33]:
```
rfm
```

[33]:
```
                recency  frequency  monetary Segment
customer_id
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 x 4 columns]
```

[34]:
```
#CHURN ANALYSIS:

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

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

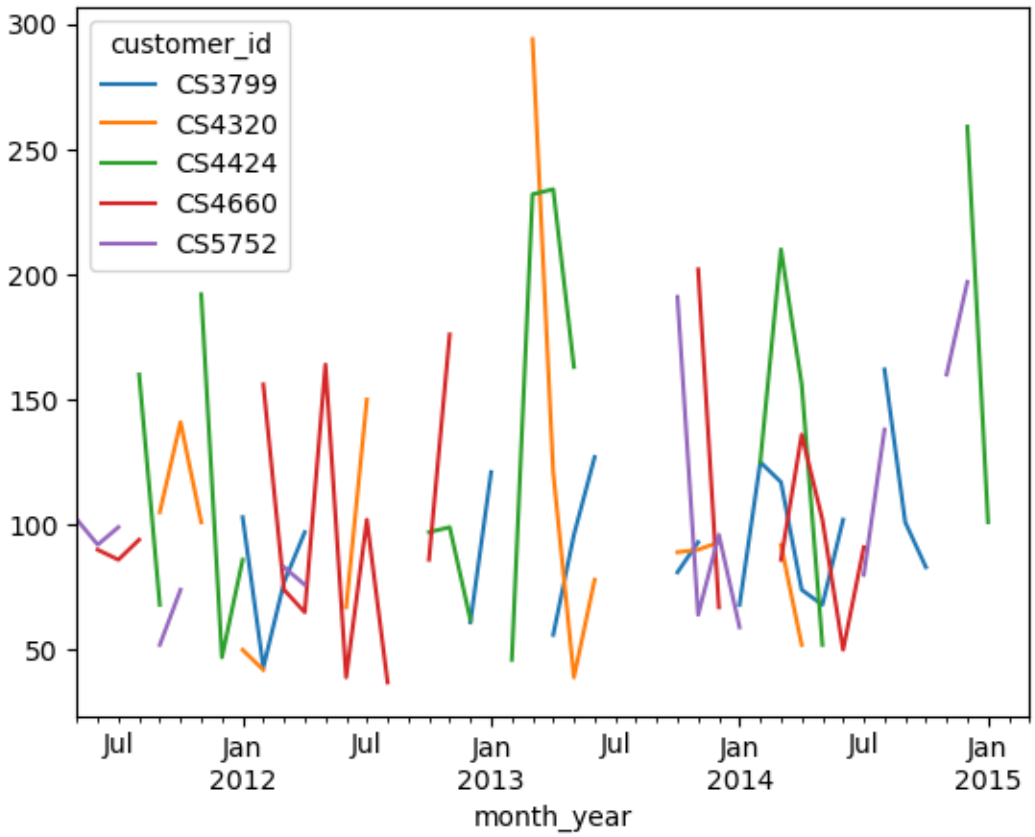
[34]:
```
<Axes: xlabel='response'>
```



[35]:
```
# ANALYSING TOP CUSTOMERS:

top_5_cus = monetary.sort_values(ascending=False).head(5).index

top_customer_df = df[df['customer_id'].isin(top_5_cus)]
```

```
top_customer_sales = top_customer_df.
  ↪groupby(['customer_id','month_year'])['tran_amount'].sum().unstack(level=0)
top_customer_sales.plot(kind='line')
```

[35]: <Axes: xlabel='month_year'>



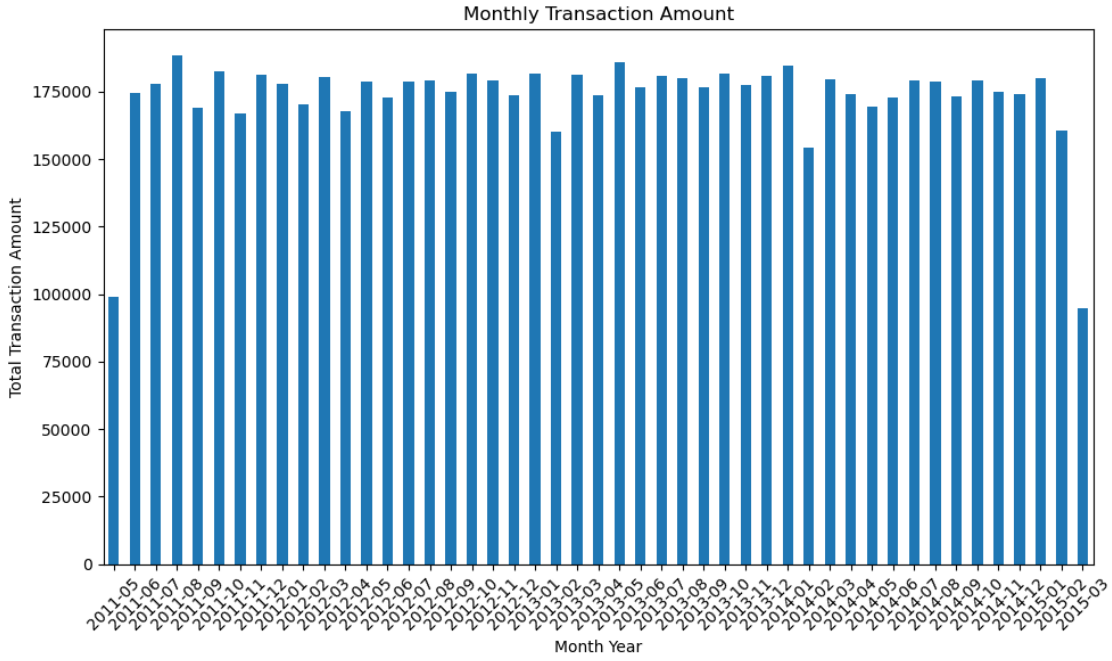
[36]:
```
df
```

[36]:

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

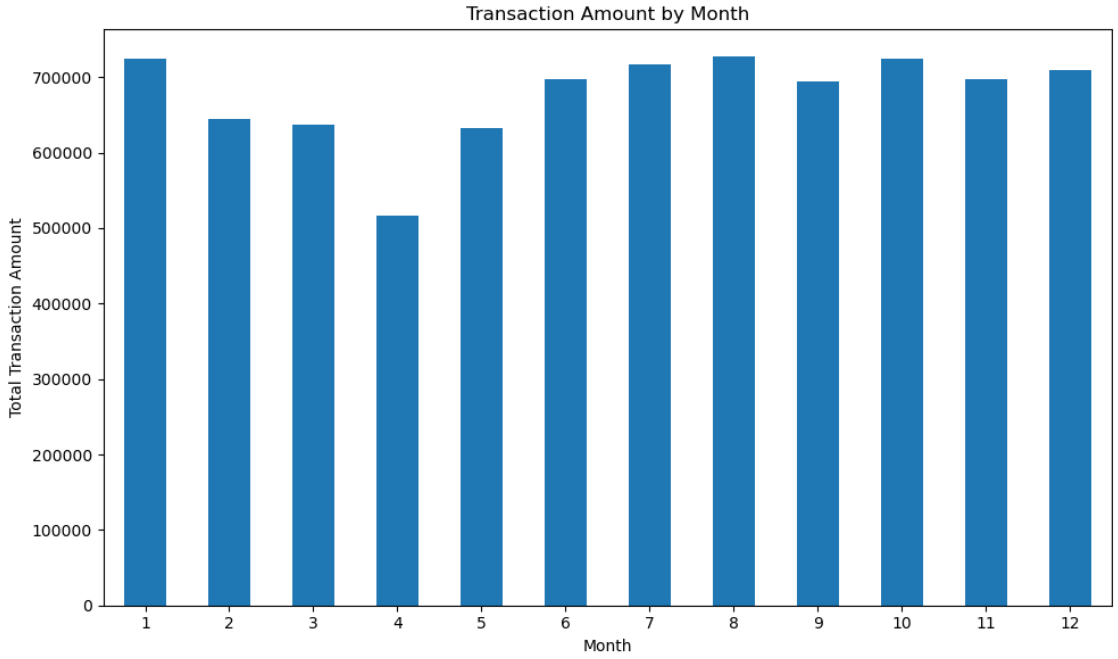
```
124999      CS7996 2014-12-13           36          0      12     2014-12

[124969 rows x 6 columns]
```

[37]:
```
df.groupby('month_year')['tran_amount'].sum().plot(
    kind='bar',
    figsize=(10,6),
    title='Monthly Transaction Amount'
)

plt.xlabel('Month Year')
plt.ylabel('Total Transaction Amount')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

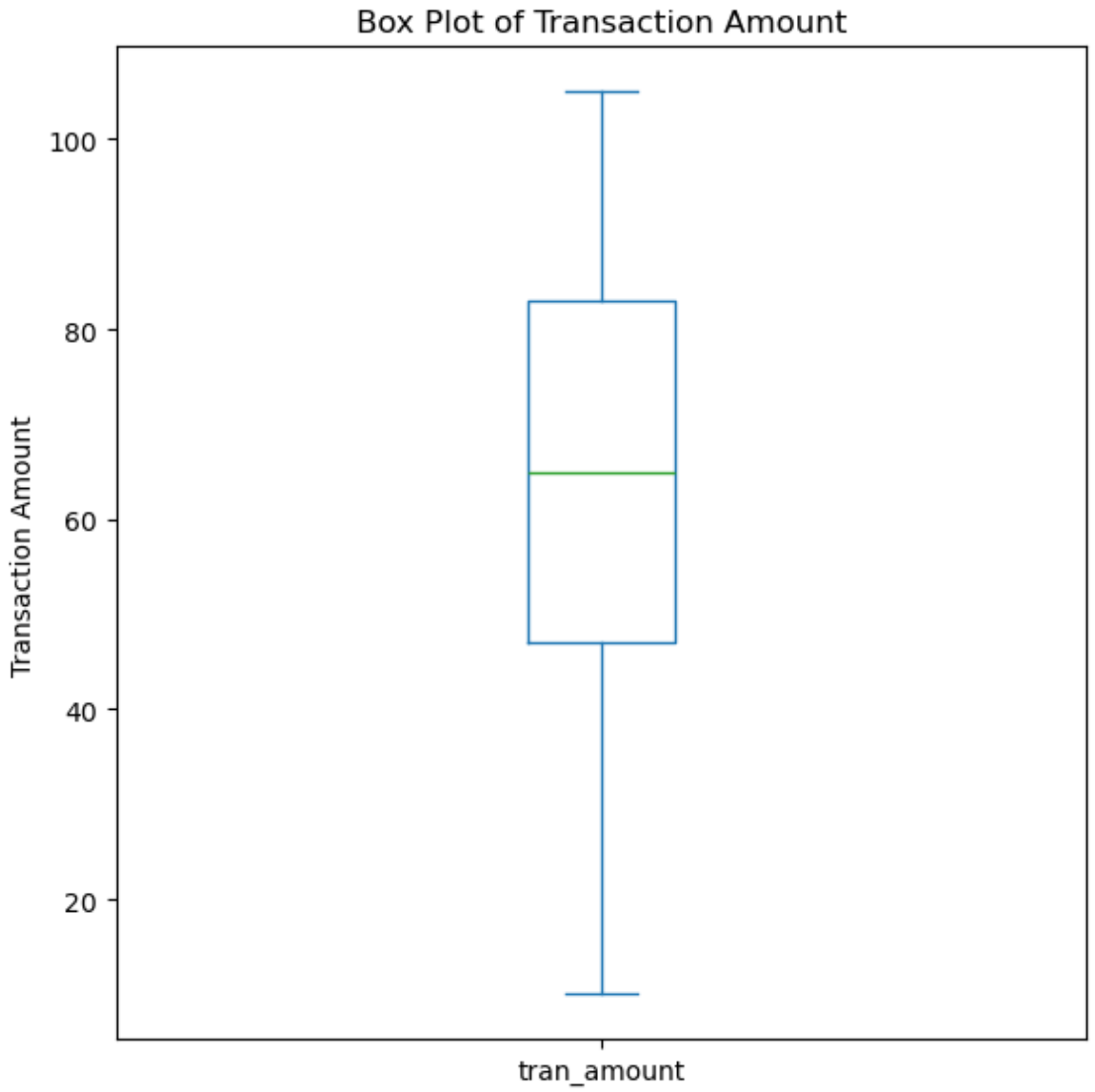


[38]:
```
import matplotlib.pyplot as plt

df.groupby('month')['tran_amount'].sum().plot(
    kind='bar',
    figsize=(10,6),
    title='Transaction Amount by Month'
)

plt.xlabel('Month')
```

```
plt.ylabel('Total Transaction Amount')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```

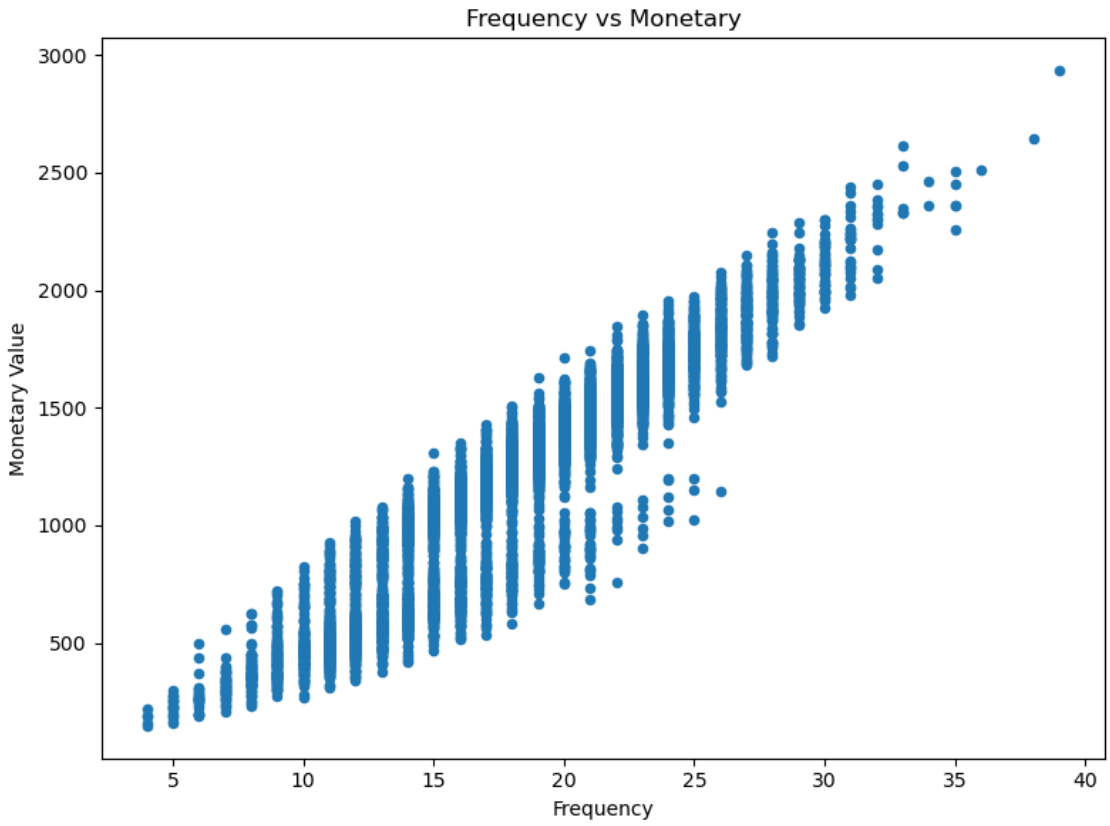


[39]:
```
import matplotlib.pyplot as plt

df['tran_amount'].plot(
    kind='box',
    figsize=(6,6),
    title='Box Plot of Transaction Amount'
)

plt.ylabel('Transaction Amount')
plt.tight_layout()
plt.show()
```

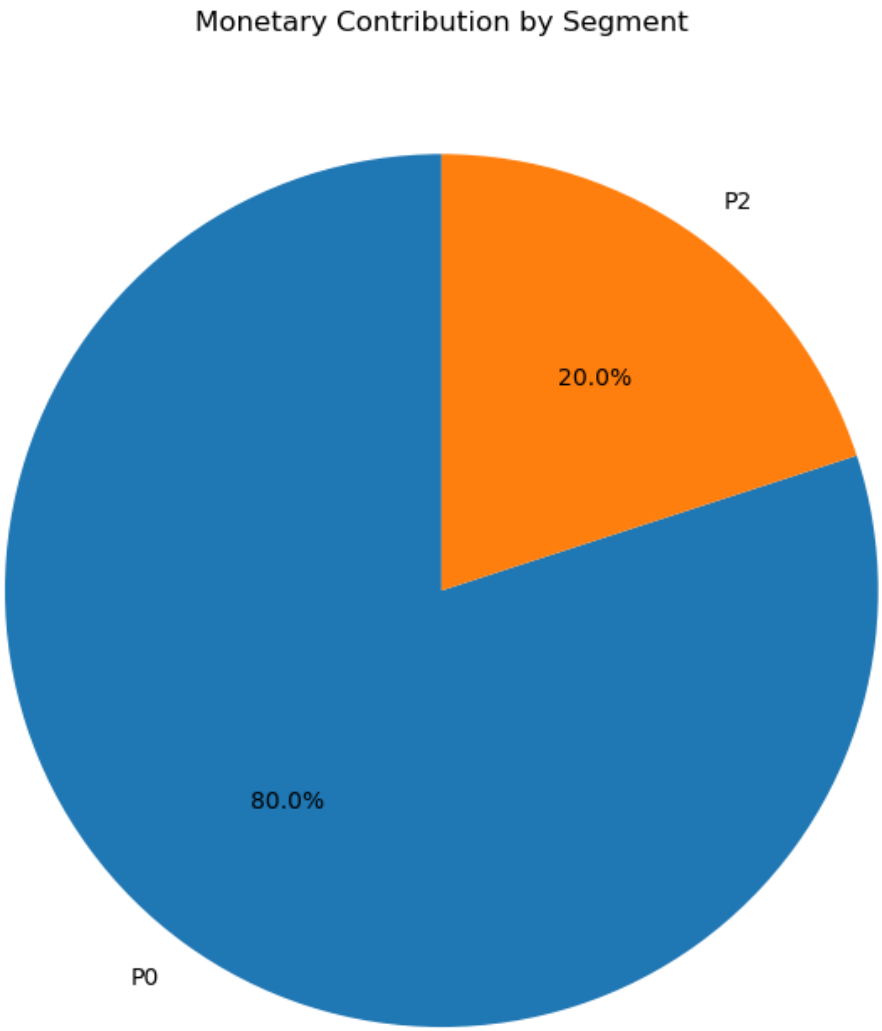


[40]:
```
import matplotlib.pyplot as plt

rfm.plot(
    kind='scatter',
    x='frequency',
    y='monetary',
    figsize=(8,6),
    title='Frequency vs Monetary'
)

plt.xlabel('Frequency')
plt.ylabel('Monetary Value')
plt.tight_layout()
```

```
plt.show()
```



[42]:
```
import matplotlib.pyplot as plt

segment_data = rfm.groupby('Segment')['monetary'].sum()

segment_data.plot(
    kind='pie',
    autopct='%1.1f%%',
    figsize=(7,7),
    startangle=90
)

plt.title('Monetary Contribution by Segment')
plt.ylabel('')   # removes default y-label
plt.tight_layout()
plt.show()
```


Monetary Contribution by Segment
P2
20.0%
80.0%
P0

[ ]: