Sales Data Analysis and Reporting for a Retail Chain

```
# Installing Libraries
import pandas as pd

trxn = pd.read_csv('Retail_Data_Transactions.csv')

trxn
```

```
       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 x 3 columns]
```

```
response = pd.read_csv('Retail_Data_Response.csv')
response
```

```
     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]
```

```
df = trxn.merge(response, on='customer_id', how='left')
df
```

```
       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]
```

```
df.dtypes
```

```
customer_id     object
trans_date      object
tran_amount      int64
response       float64
dtype: object
```

```
df.shape
```

```
(125000, 4)
```

```
df.tail()
```

```
       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
```

```
df.describe()
```

```
         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
```

```
# Missing Values
df.isnull().sum()
```

```
customer_id     0
trans_date      0
tran_amount     0
```

```
response       31
dtype: int64
```

```
(31/125000) * 100
```

```
0.024800000000000003
```

```
df = df.dropna()
df
```

```
       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

[124969 rows x 4 columns]
```

```
# Change dtypes
df['trans_date'] = pd.to_datetime(df['trans_date'])
df['response'] = df['response'].astype('int64')
df
```

```
C:\Users\pavan\AppData\Local\Temp\ipykernel_16128\1735897301.py:2:
UserWarning: Could not infer format, so each element will be parsed
individually, falling back to `dateutil`. To ensure parsing is
consistent and as-expected, please specify a format.
  df['trans_date'] = pd.to_datetime(df['trans_date'])
C:\Users\pavan\AppData\Local\Temp\ipykernel_16128\1735897301.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['trans_date'] = pd.to_datetime(df['trans_date'])
C:\Users\pavan\AppData\Local\Temp\ipykernel_16128\1735897301.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['response'] = df['response'].astype('int64')
```

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

[124969 rows x 4 columns]
```

```
set(df['response'])
```

```
{0, 1}
```

```
df.dtypes
```

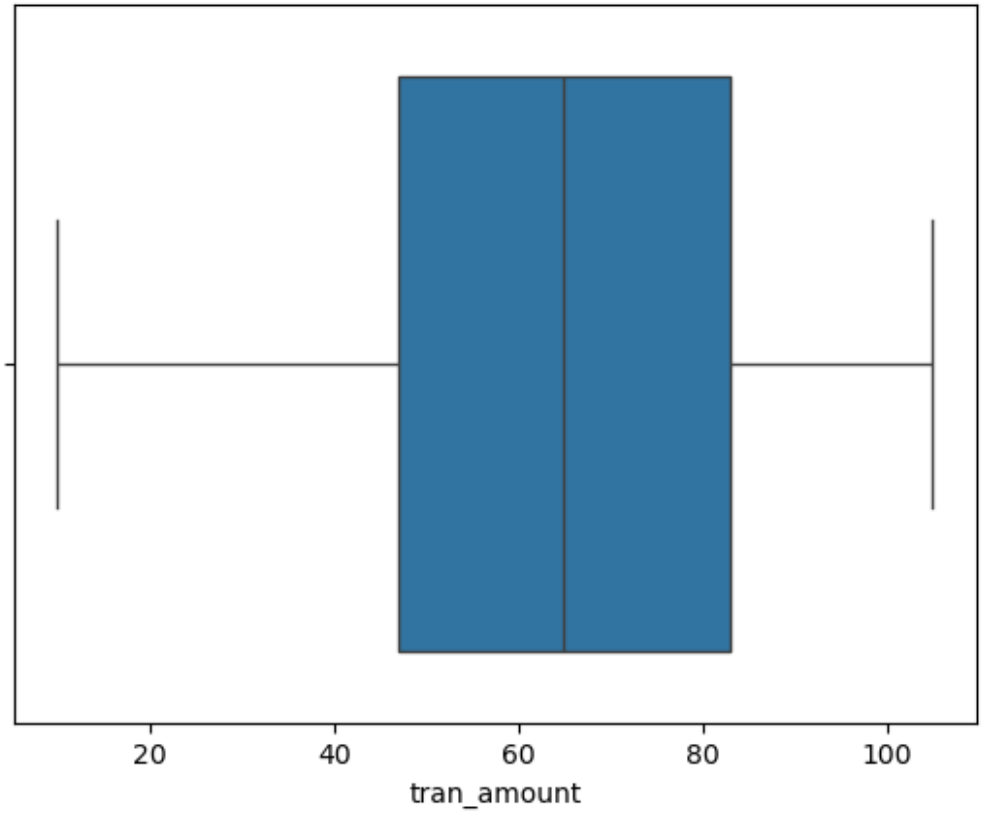
```
customer_id            object
trans_date     datetime64[ns]
tran_amount             int64
response                int64
dtype: object
```

```
# Check for Outliers

from scipy import stats
import numpy as np

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

# IQR
# Set a threshold
threshold = 3
outliers = z_scores > threshold
print(df[outliers])
```

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

```
import seaborn as sns
import matplotlib.pyplot as plt
```

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




```
df['month'] = df['trans_date'].dt.month
df
```

```
C:\Users\pavan\AppData\Local\Temp\ipykernel_16128\530967800.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['month'] = df['trans_date'].dt.month
```

	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 x 5 columns]
```

```
# Grouping by 'month' and summing the 'tran_amount'
monthly_Sales = df.groupby('month')['tran_amount'].sum()

# Sorting the months by transaction amount in descending order
monthly_Sales =
monthly_Sales.sort_values(ascending=False).reset_index()

monthly_Sales
```

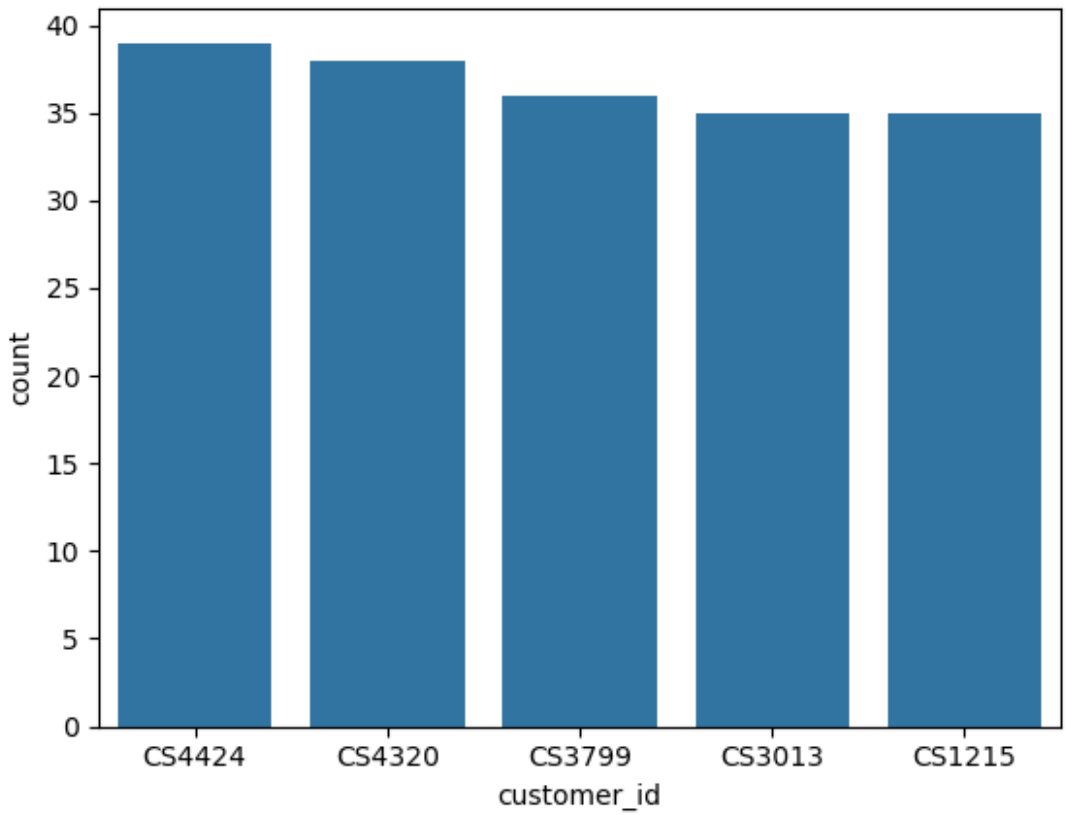
```
    month  tran_amount
0       8       726775
1      10       725058
2       1       724089
3       7       717011
4      12       709795
5      11       698024
6       6       697014
7       9       694201
8       2       645028
9       3       636475
10      5       633162
11      4       515746
```

```
# Customers having the highest number of orders
customer_counts = df['customer_id'].value_counts().reset_index()
customer_counts.columns=['customer_id','count']

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

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

```
sns.barplot(x='customer_id', y='count', data=top_5_cus)
```

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

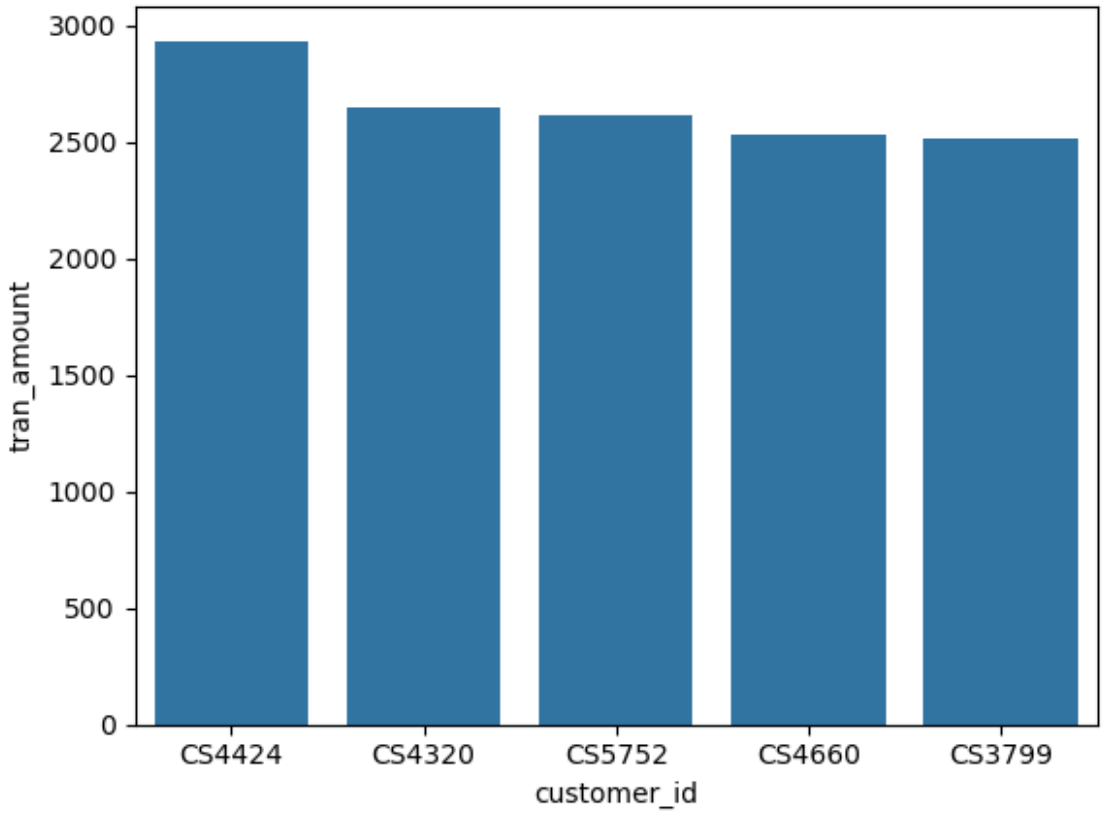



```
customer_sales = df.groupby('customer_id')
['tran_amount'].sum().reset_index()
customer_sales

# Sort
top_5_sal = customer_sales.sort_values(by='tran_amount',
ascending=False).head(5)
top_5_cus
```

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

```
sns.barplot(x='customer_id', y='tran_amount', data=top_5_sal)
```

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

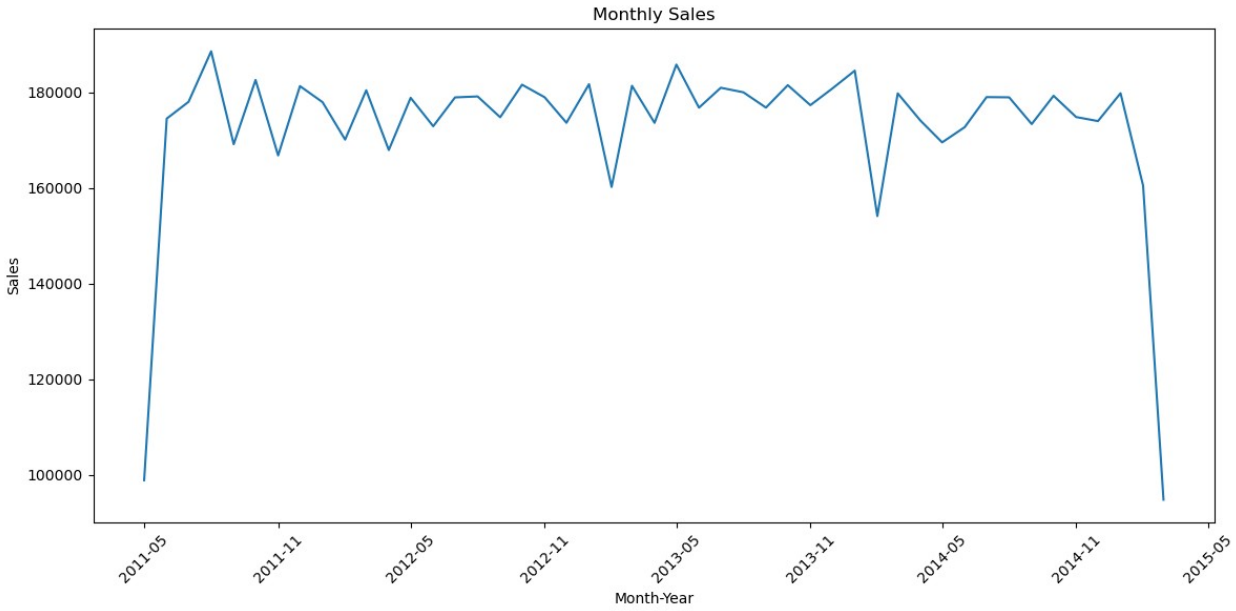


Advanced Analytics

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\pavan\AppData\Local\Temp\ipykernel_16128\3052091493.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')
```




```
df
```

	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]
```

Cohort Segmentation

```
#Recency

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

#Frequency

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

# Monetory

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

#Combine

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

rfm
```

customer_id	recency	frequency	monetary
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]

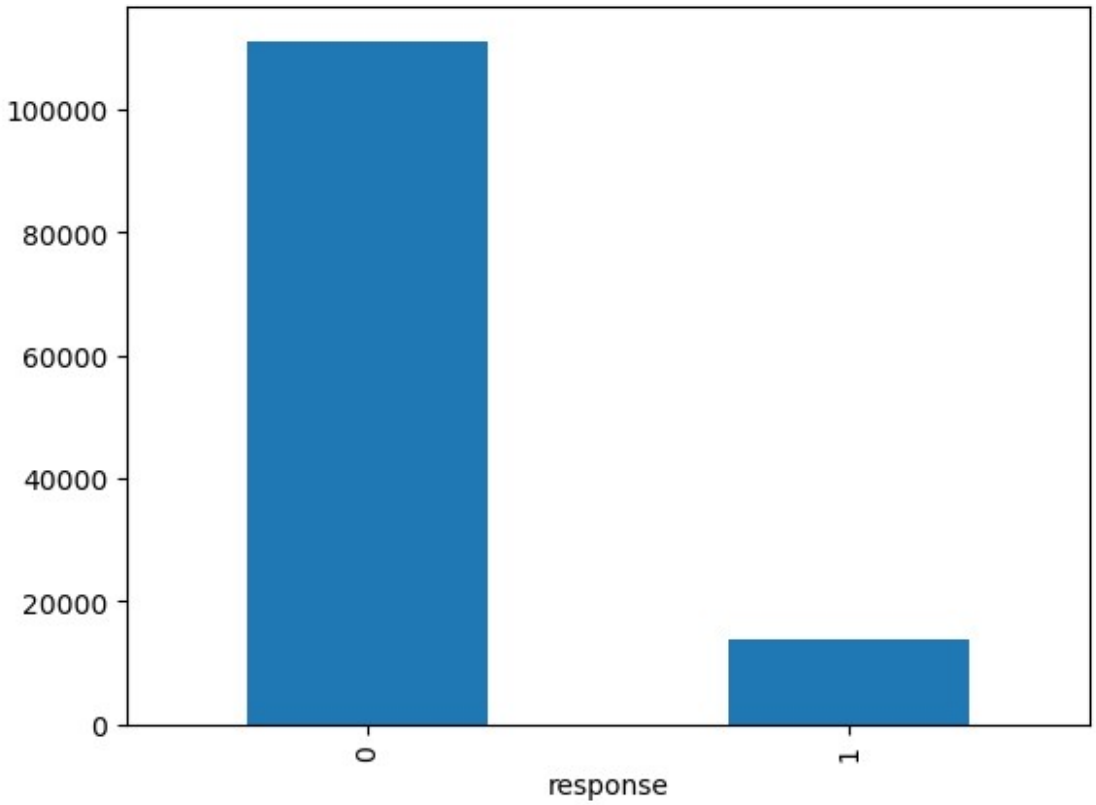
```
# customer segmentation

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)

rfm
```

```
                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]
```

Churn Analysis

```
# Count the numbers of churned and active customers

churn_counts= df['response']. value_counts()

#Plot

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

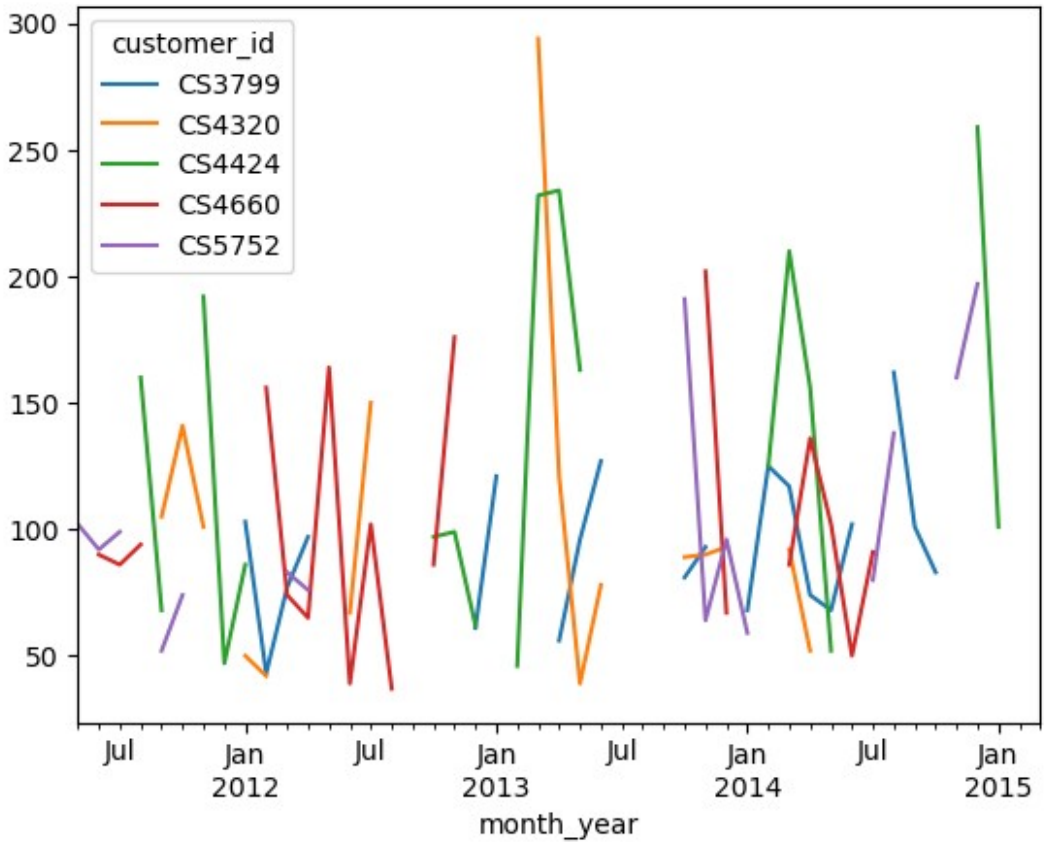
```
<Axes: xlabel='response'>
```



Analyzing top Customers

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

top_customers_df =  df[df['customer_id'].isin(top_5_cus)]


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

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




```
df
```

	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]

df.to_csv('MainData.csv')

rfm.to_csv('AddAnlys.csv')
```