Data Analytics Internship

Final Project

Project Plan

The aim of this project is to use Python, SQL, and Excel to analyze sales data and generate meaningful reports for a retail chain.

Phase 1: Data Collection and Database Setup

1. Data Collection
2. Set up a SQL database to hold the data. Design the database schema, and create the necessary tables using SQL DDL commands.

Phase 2: Data Cleaning and Preparation

1. Use SQL queries and Python (pandas) to clean the data. Look for and handle missing or inconsistent data, outliers, etc.
2. Prepare the data for analysis. This may involve creating additional calculated fields, such as total sales value, month/year fields for time-based analysis, etc.

Phase 3: Data Analysis

1. Use SQL queries and Python (pandas, matplotlib, seaborn, etc) to explore the data and identify trends and patterns.
2. Perform more complex analysis as needed. For example, time series analysis for sales trends, cohort analysis for customer behavior, etc.

Phase 4: Reporting

1. Prepare reports summarizing the findings. These can include:
   - *Tabular Reports*
   - *Visual Reports*
   - *Automated Reports*
2. Report Presentation: Present the reports to in either your Excel dashboard or prepare a PPT to present the findings

1. Data Collection
2. Set up a SQL database to hold the data. Design the database schema, and create the necessary tables using SQL DDL commands.

Phase 1

Data Collection and Database Setup

Data Collection and Database Setup

Data Collection

The two datasets are from Kaggle –

1. Retail_Data_Transactions
2. Retail_Data_Response

This dataset includes the following fields:

- Customer ID
- TransactionDate
- Transaction Amount
- Response

Database Setup

SQL Commands –

```
create database RetailProject;
use RetailProject;
```

Create Tables and Load Data

```
create table Transactions(
customer_id varchar(20),
trans_date varchar(20),
tran_amount int);

create table Response(
customer_id varchar(20) primary key,
response int);
```

```
load data infile 'C:/ProgramData/MySQL/MySQL Server8.0/Uploads/Retail_Data_Transactions.csv'
into table Transactionsfields
terminated by ','
lines terminated by '\n'
ignore 1 rows;

load data infile 'C:/ProgramData/MySQL/MySQL Server8.0/Uploads/Retail_Data_Response.csv'
into table Responsefields terminated by ','
lines terminated by '\n'
ignore 1 rows;
select * from transactions;
select * from response;
```

1. Use SQL queries and Python (pandas) to clean the data. Look for and handle missing or inconsistent data, outliers, etc.
2. Prepare the data for analysis. This may involve creating additional calculated fields, such as total sales value, month/year fields for time-based analysis, etc.

Phase 2

Data Cleaning and Preparation

Data Cleaning

```
In [16]: data.isnull().sum()
```

```
Out[16]: customer_id     0
         trans_date      0
         tran_amount     0
         response       31
         dtype: int64
```

Response field has 31 NULL values.

All the NULL values are dropped as it does not affect the whole data.

```
In [17]: data=data.dropna()
         data
```

```
In [21]: data['trans_date']=pd.to_datetime(data['trans_date'])
         data['response']=data['response'].astype('int64')
         data
```

Out[21]:

	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 × 4 columns

Data Type of response field changed to integer for further analysis.

Data type of trans_date field changed to date time format.

```
In [22]: data.dtypes
```

```
Out[22]: customer_id             object
         trans_date      datetime64[ns]
         tran_amount              int64
         response                 int64
         dtype: object
```

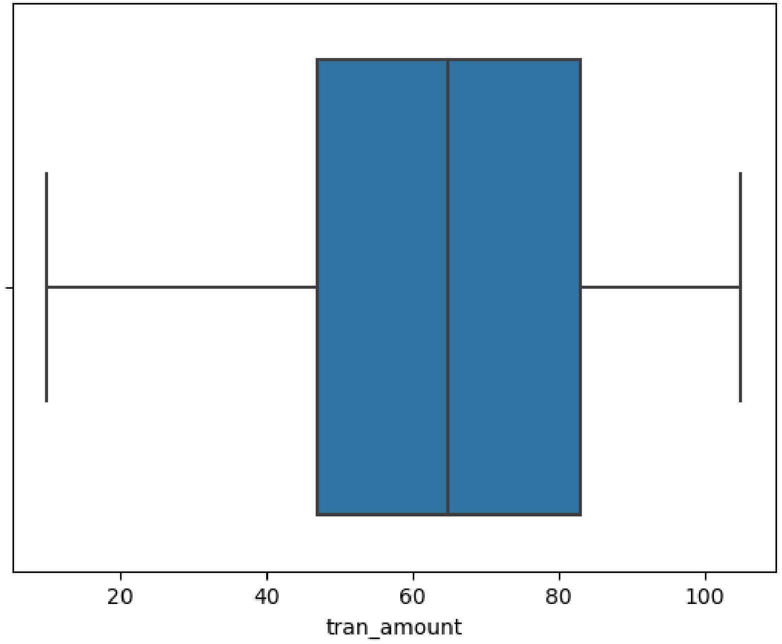
Data Cleaning – Outliers Detection

In [32]:
```
z_score=np.abs(stats.zscore(data['tran_amount']))
threshold=3
outliers=z_score>threshold
print(data[outliers])
```

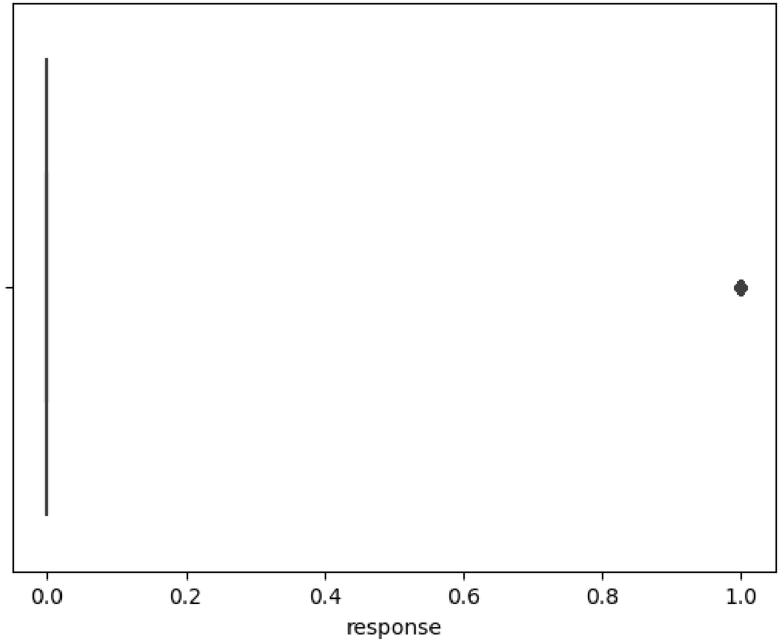
```
Empty DataFrame
Columns: [customer_id, trans_date, tran_amount, response]
Index: []
```

In [33]:
```
z_score=np.abs(stats.zscore(data['response']))
threshold=3
outliers=z_score>threshold
print(data[outliers])
```

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



No outliers found in both tran_amount and response field.



Data Preparation

Adding month field to the dataset

In [38]:
```
data['month']=data['trans_date'].dt.month
data
```

Out[38]:

	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

In [41]:
```
#which 3 months have had the highest transaction amount
ms=data.groupby('month')['tran_amount'].sum()
ms=ms.sort_values(ascending=False).reset_index().head(3)
ms
```

Out[41]:

	month	tran_amount
0	8	726775
1	10	725058
2	1	724089

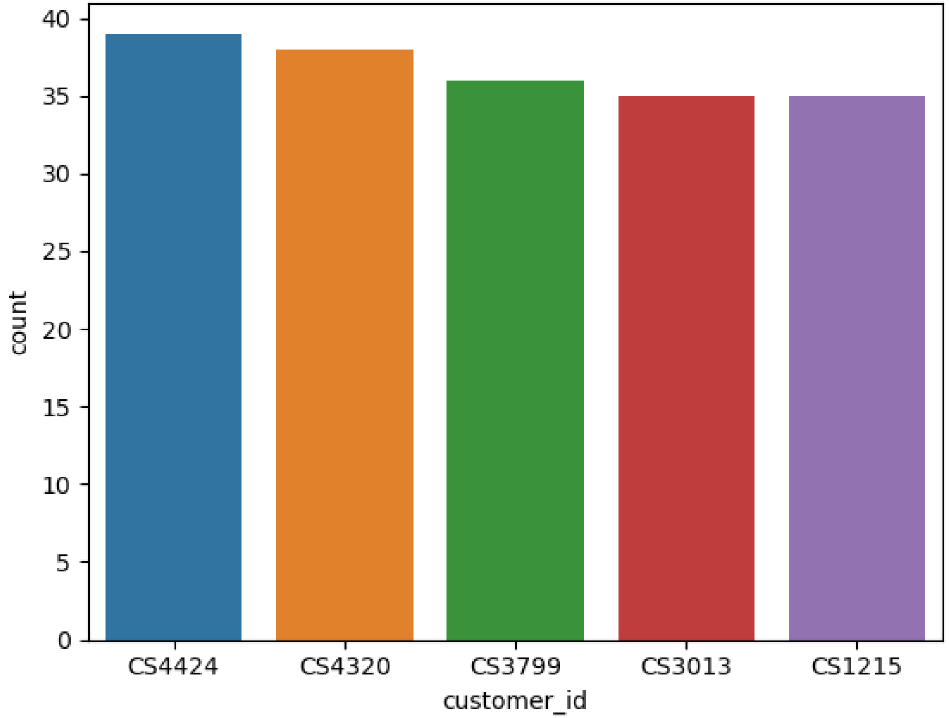
In [46]:
```
#which 5 customers have had highest no.of orders
cus=data['customer_id'].value_counts().reset_index()
cus.columns=['customer_id','count']
cus=cus.sort_values(by='count',ascending=False).head(5)
cus
```

Out[46]:

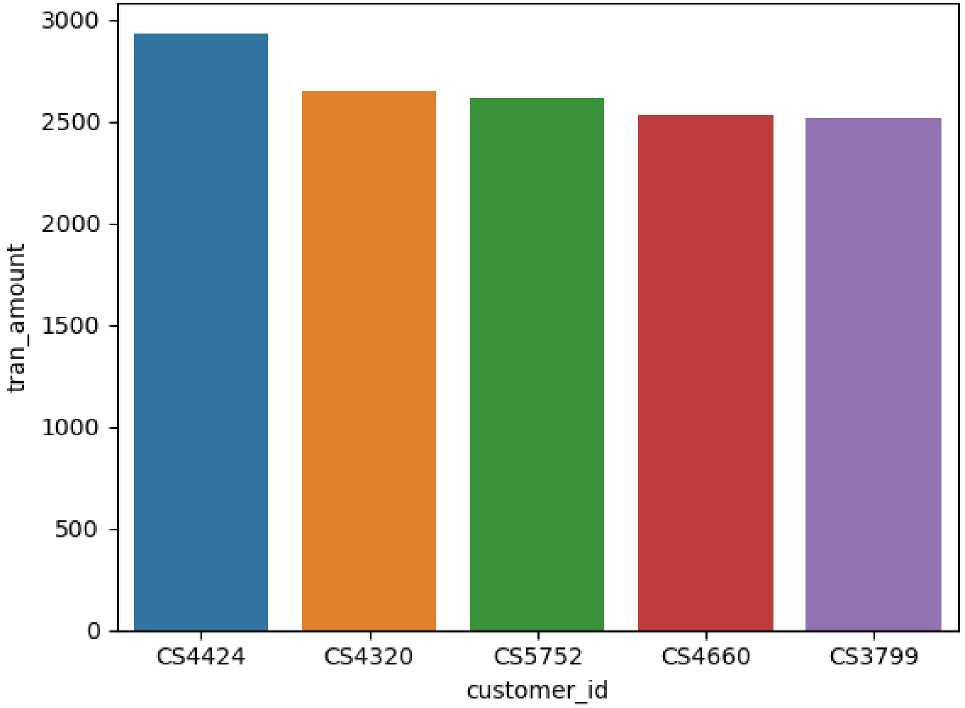
	customer_id	count
0	CS4424	39
1	CS4320	38
2	CS3799	36
3	CS3013	35
4	CS1215	35

Data Preparation

5 customers that have highest no.of orders



5 customers that have highest no.of transaction amount



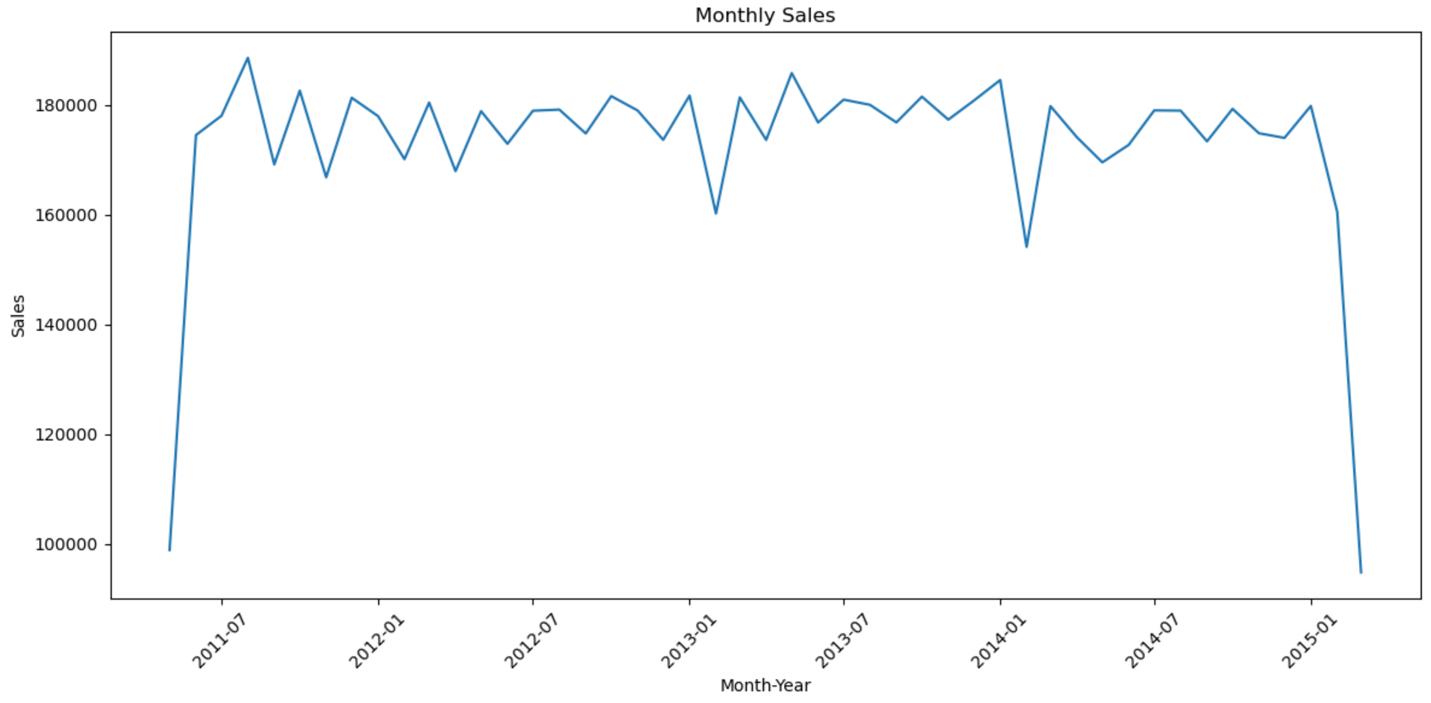
Phase 3

Data Analysis

1. Use SQL queries and Python (pandas, matplotlib, seaborn, etc) to explore the data and identify trends and patterns.
2. Perform more complex analysis as needed. For example, time series analysis for sales trends, cohort analysis for customer behavior, etc.

Time Series Analysis

In [59]:
```
msales=data.groupby('month_year')['tran_amount'].sum()
msales.index=msales.index.to_timestamp()
plt.figure(figsize=(12,6))
plt.plot(msales.index,msales.values)
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))
plt.xlabel('Month-Year')
plt.ylabel('Sales')
plt.title('Monthly Sales')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



The highest sales was in 2011-07 and lowest was in 2015-01.

Transaction has decreased significantly in 2013-01, 2014-01 and 2015-01.

RFM Analysis

In [60]:
```
#RFM Analysis
recency=data.groupby('customer_id')['trans_date'].max()
frequency=data.groupby('customer_id')['trans_date'].count()
monetary=data.groupby('customer_id')['tran_amount'].sum()
rfm=pd.DataFrame({'recency':recency,'frequency':frequency,'monetary':monetary})
rfm
```

Out[60]:

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 × 3 columns

Recency – how recent a transaction has occurred.

Frequency – how frequently the transaction occurs.

Monetary – for how munch value the transaction occurs.

Cohort Segmentation

In [61]:
```
def seg(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(seg,axis=1)
rfm
```

Out[61]:

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

The customers are divided into different segments – P0, P1, and P2 based on the RFM Values.

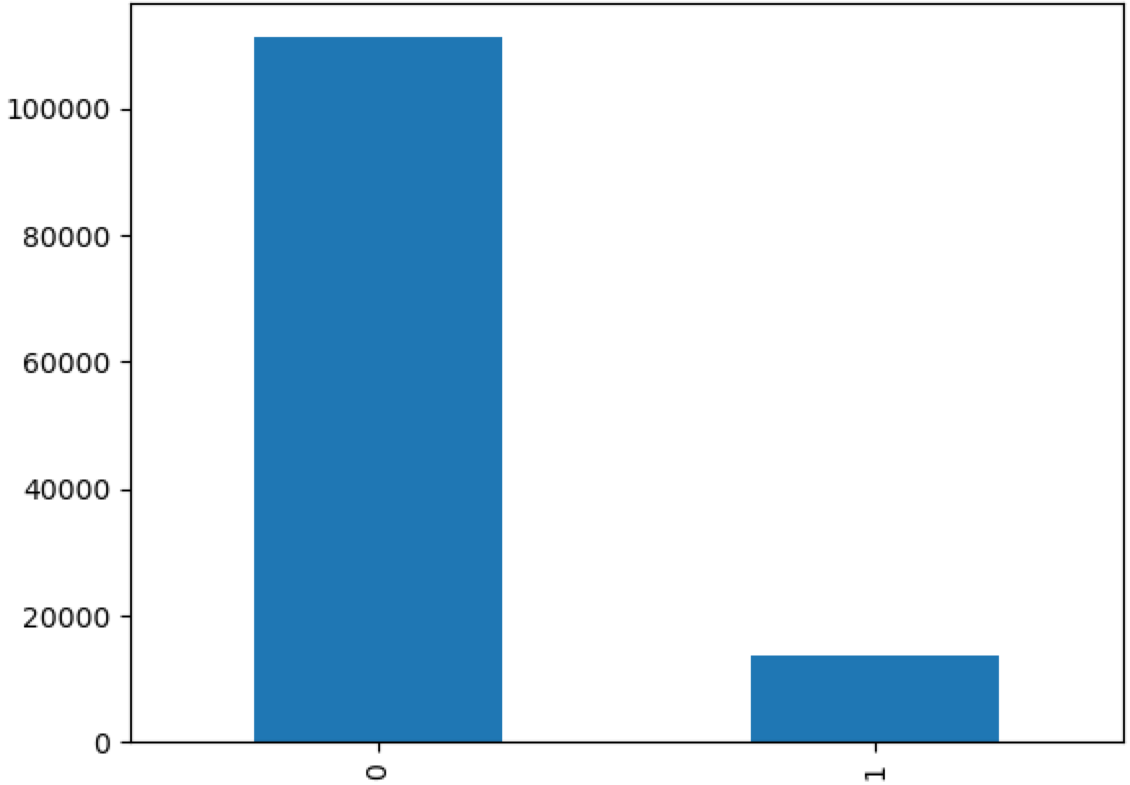
Churn Analysis

```
In [62]: #no.of churned and active customers
         churn=data['response'].value_counts()
         churn.plot(kind='bar')
```

Most of the customer response is 0.

But no.of customers churned is based on the Recency value. If a customer response is 0 and he/she has done a transaction recently then that customer is not churned.

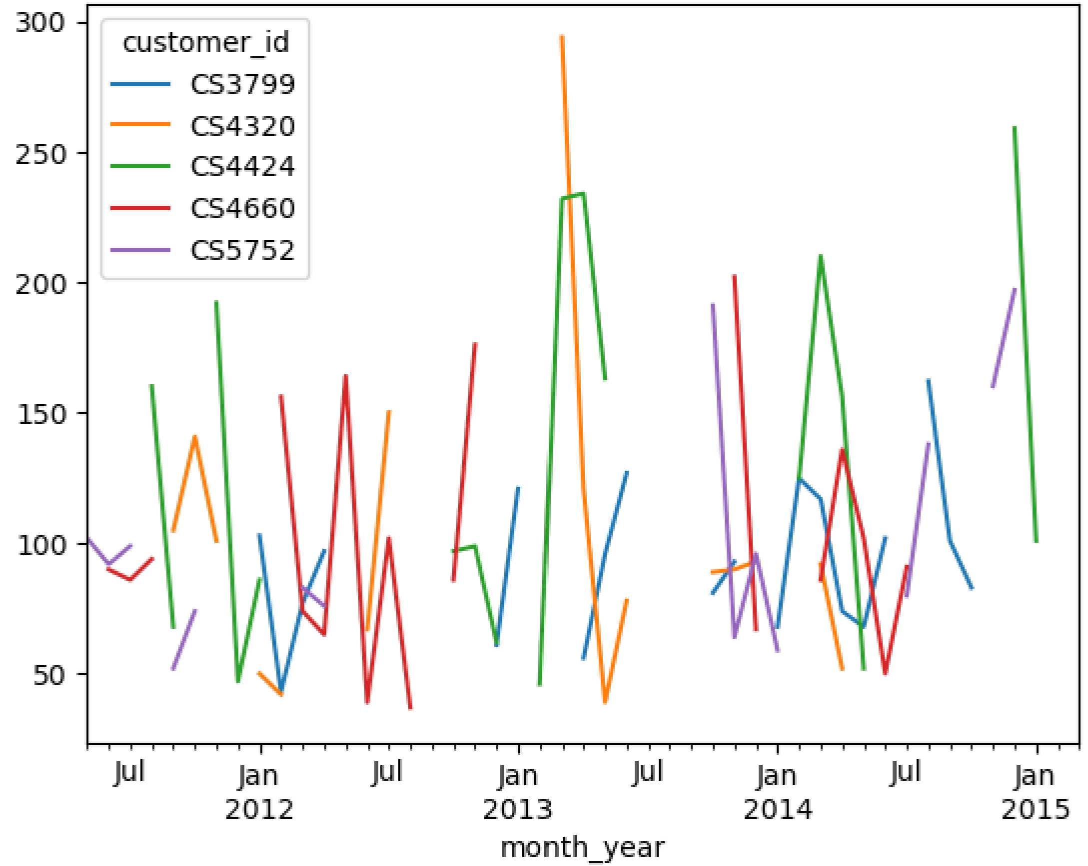
Hence, churning customers need to be identified based on the recency and frequency of transactions.



Top Customers

```
In [63]: top=monetary.sort_values(ascending=False).head(5).index
         top=data[data['customer_id'].isin(top)]
         tops=top.groupby(['customer_id','month_year'])['tran_amount'].sum().unstack(level=0)
         tops.plot(kind='line')
```

The top 5 customers based the transaction amount according to the month-year is shown in the line graph.



Prepare reports summarizing the findings.

These can include:

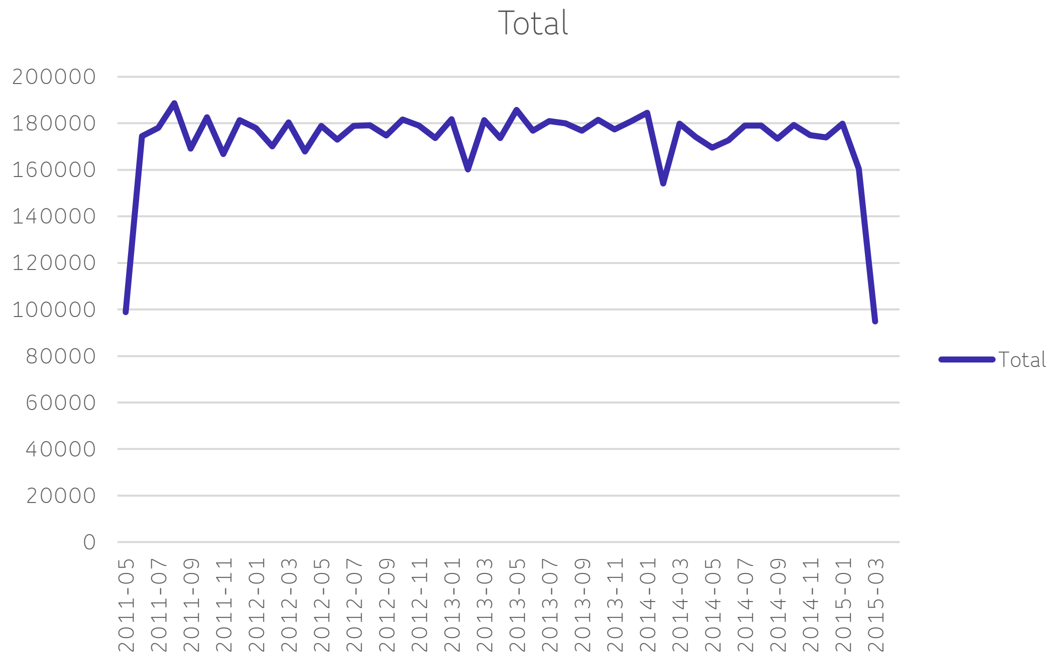
- Tabular Reports
- Visual Reports
- Automated Reports

Phase 4

Reporting

Pivot Table & Chart – Month-Year wise Transaction amount

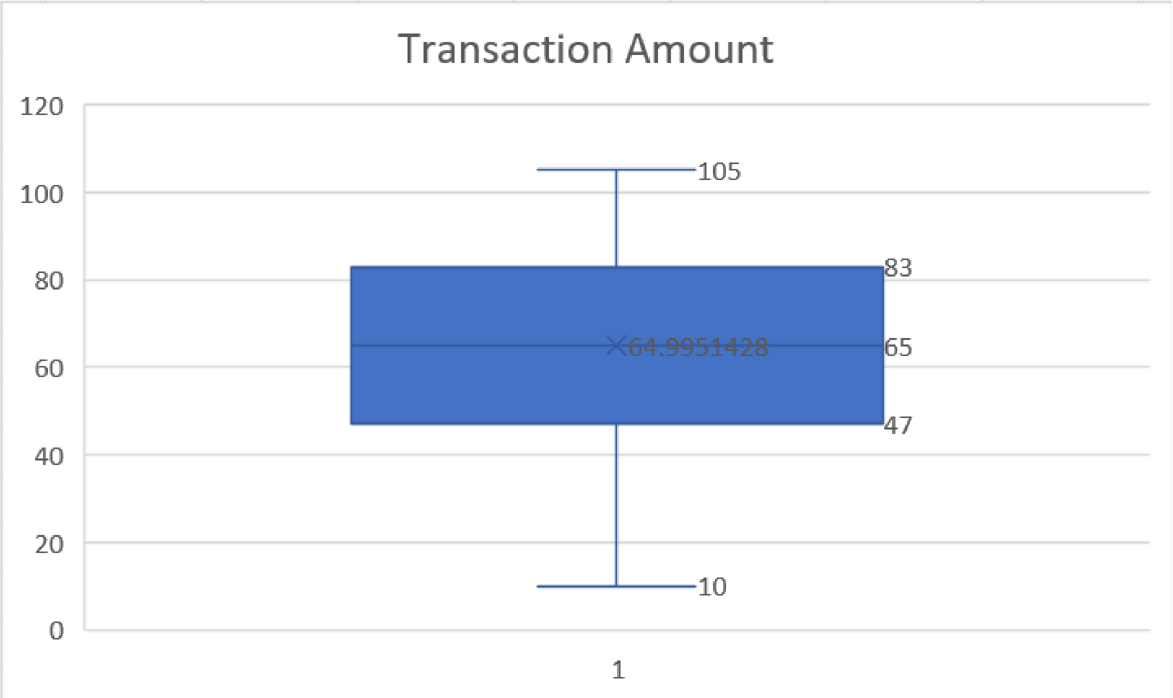
Row Labels	Sum of tran_amount
2011-05	98901
2011-06	174527
2011-07	178038
2011-08	188605
2011-09	169173
2011-10	182613
2011-11	166830
2011-12	181326
2012-01	177969
2012-02	170135
2012-03	180453
2012-04	167955
2012-05	178880
2012-06	172933
2012-07	178964
2012-08	179164
2012-09	174813
2012-10	181621
2012-11	178998
2012-12	173657
2013-01	181729
2013-02	160233
2013-03	181389
2013-04	173642
2013-05	185826
2013-06	176813
2013-07	180983
2013-08	180031
2013-09	176830
2013-10	181521
2013-11	177341
2013-12	180802
2014-01	184554
2014-02	154151
2014-03	179804
2014-04	174149
2014-05	169555
2014-06	172741
2014-07	179026
2014-08	178975
2014-09	173385
2014-10	179303
2014-11	174855
2014-12	174010
2015-01	179837
2015-02	160509
2015-03	94829
(blank)	
Grand Total	**8122378**



Transaction was highest in 2011-07 and lowest in 2015-03.

Transaction increased significantly in 2011 and decreased significantly in 2015.

Box Plot



Box plot for transaction amount.

Minimum value=10

Maximum value=105

Mean=65

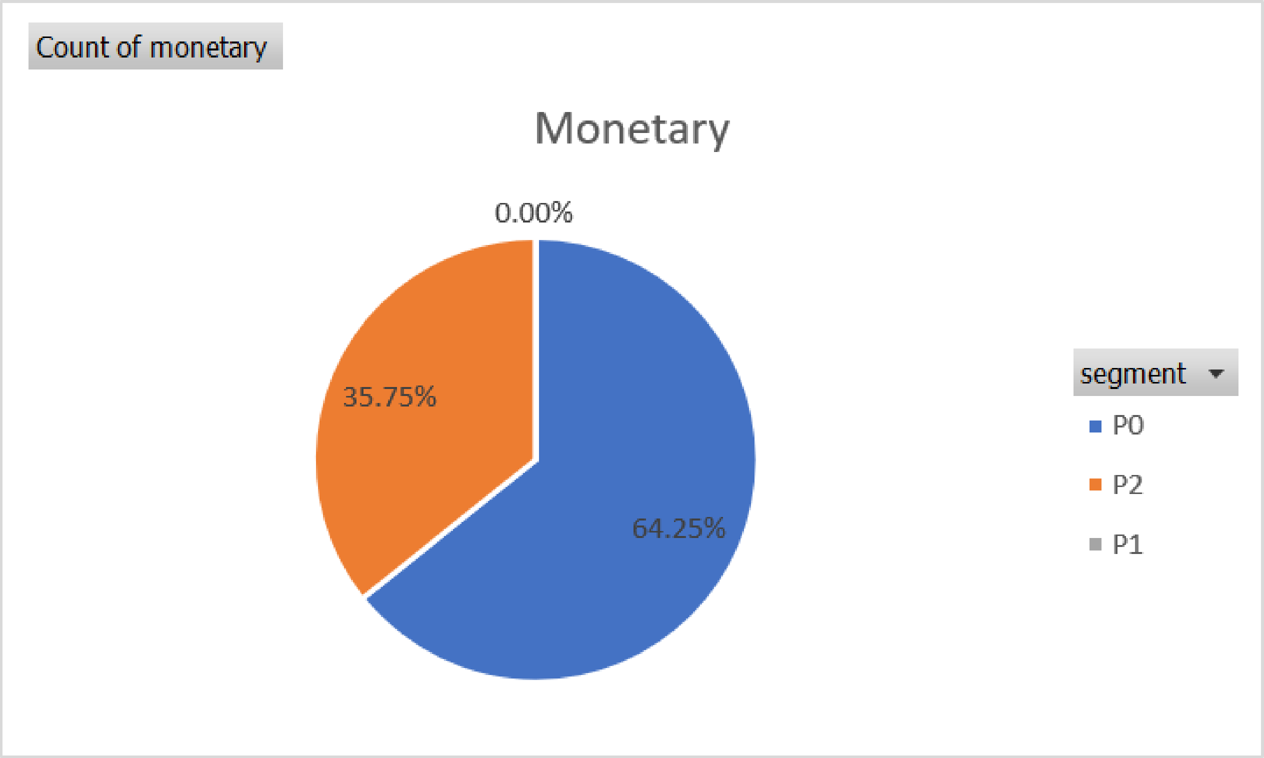
Quartile 1=47

Quartile 3=83

The average transaction amount is around 65.

Pie Chart

Row Labels	Count of monetary
P0	64.25%
P2	35.75%
P1	0.00%
Grand Total	**100.00%**



Pie Chart of Cohort Segmentation based on Monitary value.

P0 segment=64.25%

P1 segment=0%

P2 segment=35.75%

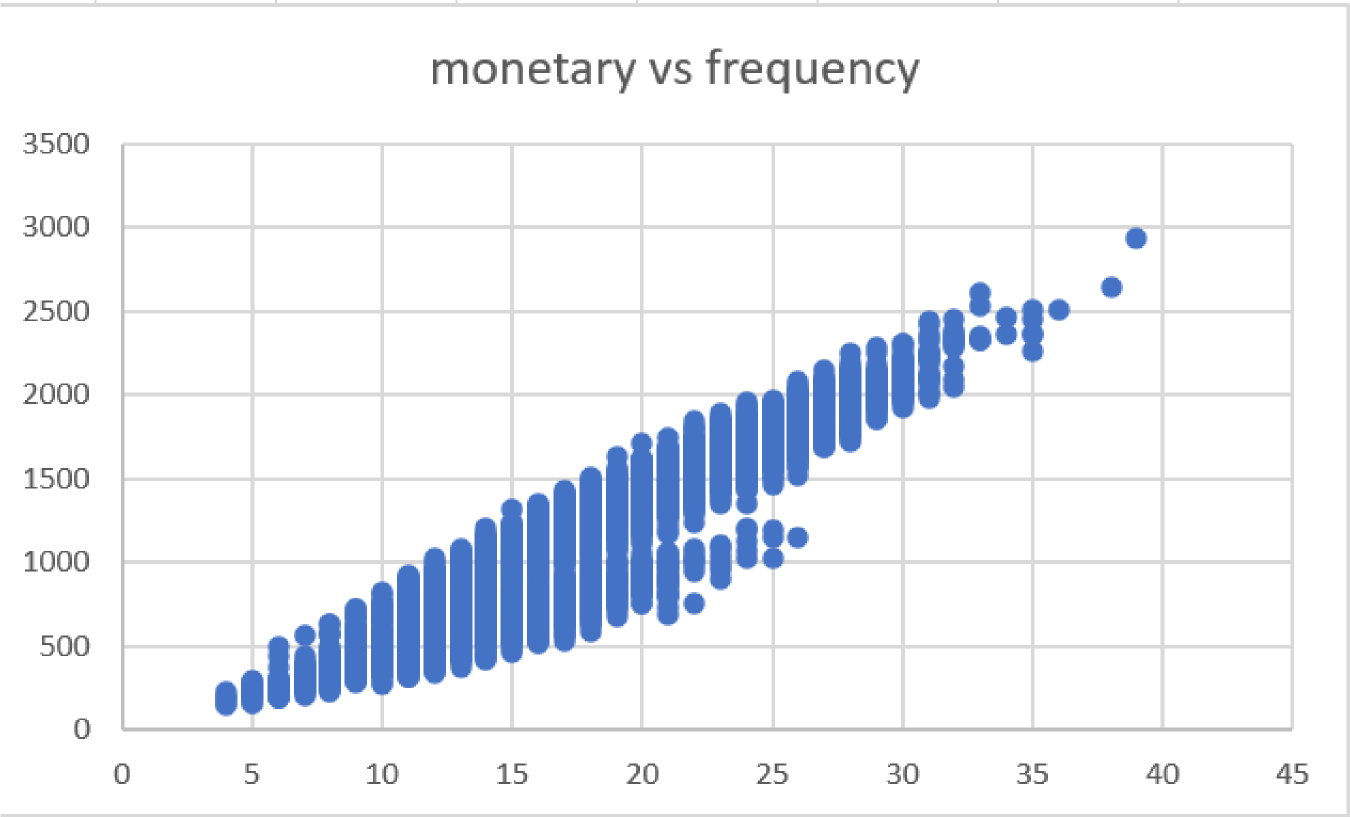
Maximum Customers belong to P0 and none belong to P1 cohort.

Scatter Plot

Scatter Plot between monetary value and frequency value.

The graph shows that monetary and frequency are positively correlated.

That is, if monetary increases the frequency also increases and vice-versa.



20XX

THANK YOU!

S Yashuvanthra Dhevi