```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

```

Dataset Preview

```

# Load the dataset from my google drive in colab

import sys
from google.colab import drive
drive.mount('/content/gdrive')
sys.path.insert(0, '/content/gdrive/My Drive/')
```

```
Mounted at /content/gdrive
```

```
# Data Preview
data = pd.read_excel('gdrive/My Drive/P2_user_segmentation.xlsx')

data
```

	Mobile Key	Person Is Alive	Person Birth Date	Person Gender Title	Person First Name	Person Last Name	Mobile Number Encrypt	Mobile Number Masked	FirstAppInstallDate	Mobile Operator Id	...	C
0	2	True	1997-10-09	مرد	مجید	ملکی	HN8x8N88m+sikOUIs4luxg==	0938***0926	2017-04-18 20:15:22.910	2	...	
1	4	NaN	NaT	NaN	NaN	NaN	/Y38H9ETC2tMantBTYBQzw==	0913***9963	2014-12-15 21:59:03.870	1	...	
2	6	NaN	NaT	NaN	NaN	NaN	ThuZBrT+/BueLw7LxSX15w==	0919***0132	2016-04-18 10:15:01.480	1	...	
3	7	True	1979-07-13	مرد	حمیدرضا	عزیزی مربویه	0sEbFPeIKs9cCJLA73cOoQ==	0917***4959	2020-01-13 23:07:15.570	1	...	
4	9	True	1979-03-11	مرد	امیدعلی	احمدپور	FZT2oVrc5UCFaN8gjELmCg==	0917***8979	2016-11-22 22:03:27.060	1	...	
...	...	...	...	...	...	...	...	...	...	...	...	
49995	125060	NaN	NaT	NaN	NaN	NaN	pDcHaQlw4rFir+7T1rBFEQ==	0915***7614	2017-11-16 21:45:59.757	1	...	
49996	125068	True	1979-03-21	مرد	حمید	پارسای جرفی	/lv74VOVi1nIVBK006oiTA==	0915***6893	2018-07-06 19:38:36.493	1	...	
49997	125071	NaN	NaT	NaN	NaN	NaN	SFqmPCgQQuhu9iyXJNxG+Q==	0914***7368	2021-09-04 20:47:17.843	1	...	
49998	125072	NaN	NaT	NaN	NaN	NaN	QsN0ea3HyEgMsqZpmq/ywA==	0917***4369	2016-08-02 20:37:16.367	1	...	
49999	125074	NaN	NaT	NaN	NaN	NaN	qpy0mX4gLOvrrzh7xhWuCg==	0936***5729	2017-12-23 22:42:00.613	2	...	

50000 rows × 106 columns

```
print(data.head())
```

```
   Mobile Key Person Is Alive Person Birth Date Person Gender Title  \
0           2            True        1997-10-09                  مرد
1           4             NaN               NaT                  NaN
2           6             NaN               NaT                  NaN
3           7            True        1979-07-13                  مرد
4           9            True        1979-03-11                  مرد

  Person First Name Person Last Name     Mobile Number Encrypt  \
0              ملکی              مجید  HN8x8N88m+sikOUIs4Iuxg==
1               NaN               NaN  /Y38H9ETC2tMantBTYBQzw==
2               NaN               NaN  ThuZBrT+/BueLw7LxSX15w==
3          0  عزیزی مربویه      حمیدرضاsEbFPeIKs9cCJLA73cOoQ==
4           احمدپور           امیدعلی  FZT2oVrc5UCFaN8gjELmCg==

  Mobile Number Masked     FirstAppInstallDate  Mobile Operator Id  ...  \
0          0938***0926 2017-04-18 20:15:22.910                   2  ...
1          0913***9963 2014-12-15 21:59:03.870                   1  ...
2          0919***0132 2016-04-18 10:15:01.480                   1  ...
```

```
3          0917***4959 2020-01-13 23:07:15.570                  1  ...
4          0917***8979 2016-11-22 22:03:27.060                  1  ...

   CMS_Balance    LastTrsDateWallet FirstProvince SecondProvince FirstRegion  \
0          NaN                  NaN           NaN            NaN         NaN
1          NaN                  NaN           NaN            NaN         NaN
2          NaN                  NaN           NaN            NaN         NaN
3          NaN                  NaN        شیراز        فارس          فارس
4     568580.0  2023-07-23 13:56:44        مرودشت       فارس          فارس

   SecondRegion FirstGuild SecondGuild ThirdGuild  \
0           NaN        NaN         NaN        NaN
1           NaN        NaN         NaN        NaN
2           NaN        NaN         NaN        NaN
3          5811       5499,5311   5411       شیراز
4          5411       5251        5411       شیراز

                                          AllGuild
0                                              NaN
1                                              NaN
2                                              NaN
3  5411,5441,7538,5533,5462,8062,5533,5411,5814,7...
4  1731,5261,5411,5812,5065,5065,5814,5441,5411,8...

[5 rows x 106 columns]
```

```
1 print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Columns: 106 entries, Mobile Key to AllGuild
dtypes: datetime64[ns](2), float64(77), int64(3), object(24)
memory usage: 40.4+ MB
None
```

Detecting Missing Values

```
1 print(data.isnull().sum())
```

```
Mobile Key                 0
Person Is Alive        33819
Person Birth Date      34421
Person Gender Title    33851
Person First Name      33819
                       ...
SecondRegion           39957
FirstGuild             38353
SecondGuild            39352
ThirdGuild             41023
AllGuild               38353
Length: 106, dtype: int64
```

```
1 data.isnull().sum() * 100 / len(data)
```

	0
Mobile Key	0.000
Person Is Alive	67.638
Person Birth Date	68.842
Person Gender Title	67.702
Person First Name	67.638
...	...
SecondRegion	79.914
FirstGuild	76.706
SecondGuild	78.704
ThirdGuild	82.046
AllGuild	76.706

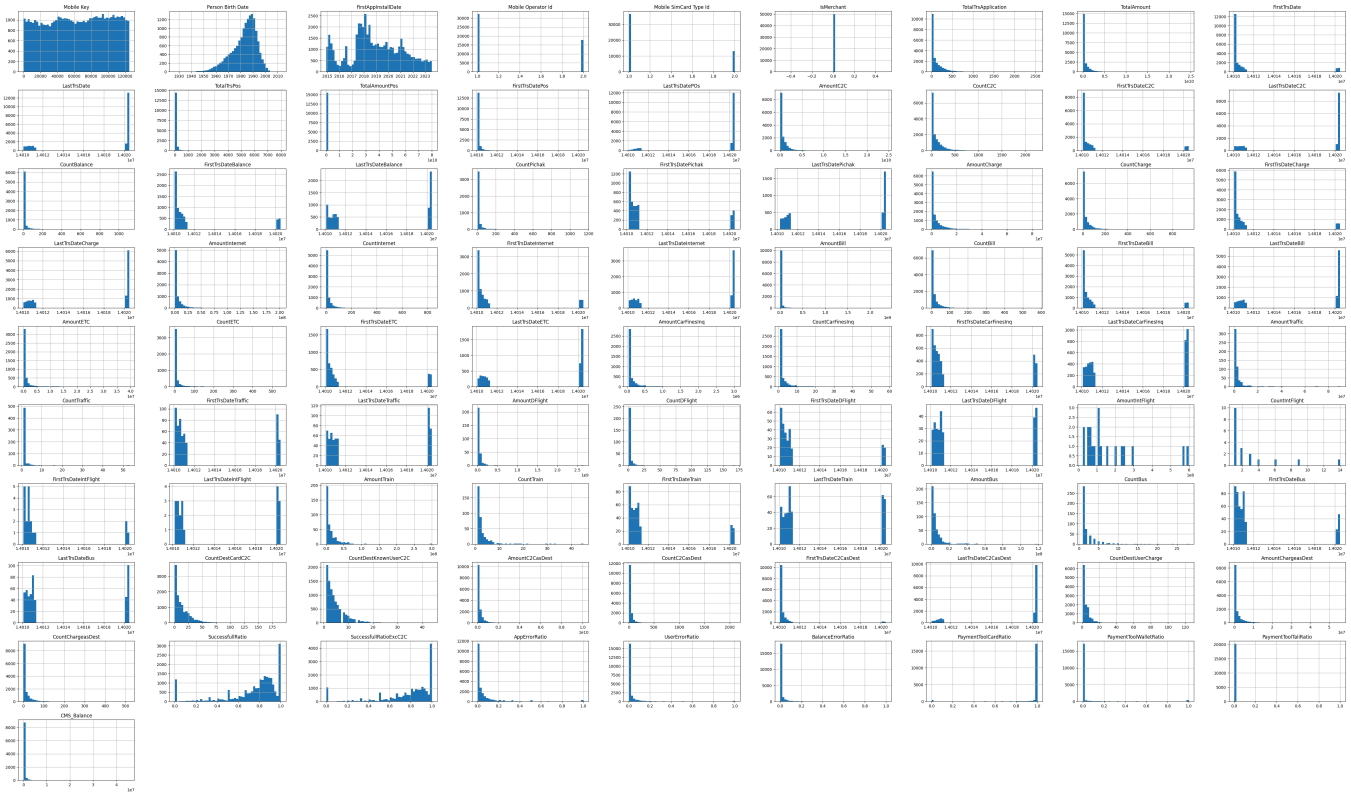
106 rows × 1 columns

Data Visualization

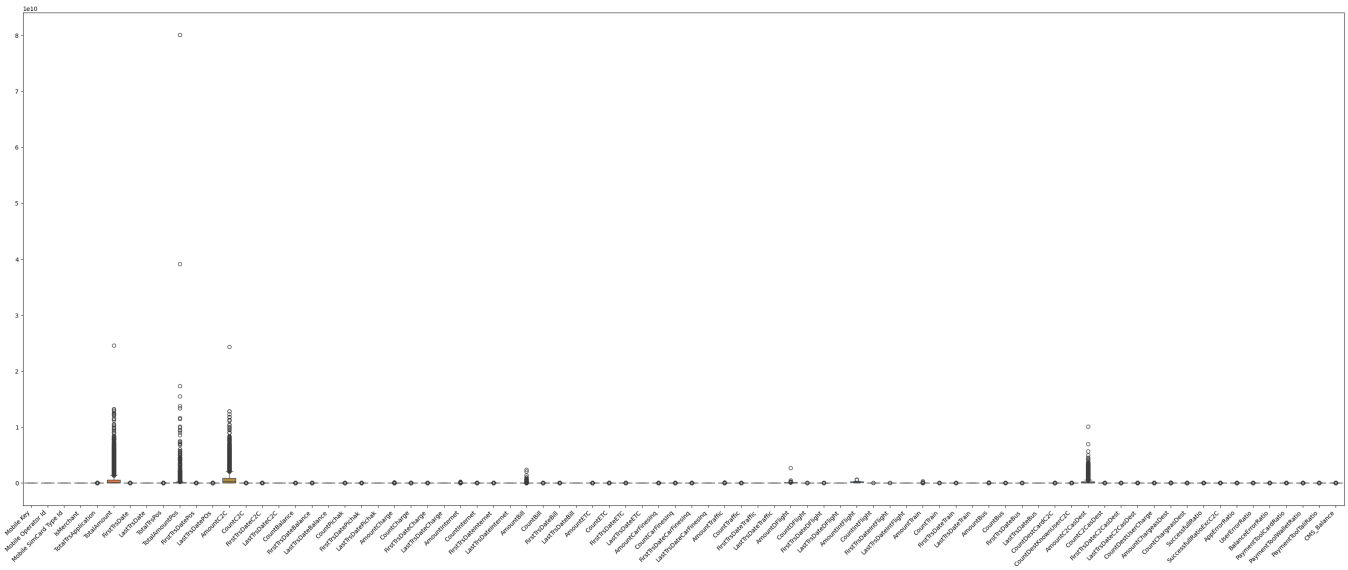
```
# Statistical Summary
print(data.describe())

```

```
min       2014-12-01 21:45:03.027000            1.000000
25%    2017-08-08 22:13:58.562500096            1.000000
50%       2018-09-11 15:00:31.040000            1.000000
75%    2020-08-16 08:26:59.817250048            2.000000
max       2023-07-24 17:37:55.923000            2.000000
std                               NaN            0.478391

       Mobile SimCard Type Id  IsMerchant  TotalTrsApplication   TotalAmount  \
count            49597.000000     50000.0         20285.000000  2.028500e+04
mean                 1.263705         0.0           112.099482  4.733102e+08
min                  1.000000         0.0             1.000000  0.000000e+00
25%                  1.000000         0.0             6.000000  4.000000e+06
50%                  1.000000         0.0            41.000000  6.428000e+07
75%                  2.000000         0.0           153.000000  5.517271e+08
max                  2.000000         0.0          2548.000000  2.452428e+10
std                  0.440646         0.0           170.979340  9.501309e+08

       FirstTrsDate   LastTrsDate  ...  CountChargeasDest  SuccessfullRatio  \
count  2.028500e+04  2.028500e+04  ...       13016.000000      21461.000000
mean   1.401112e+07  1.401775e+07  ...          13.752535          0.748779
min    1.401010e+07  1.401010e+07  ...           1.000000          0.000000
25%    1.401010e+07  1.401121e+07  ...           1.000000          0.680161
50%    1.401013e+07  1.402042e+07  ...           4.000000          0.830097
75%    1.401061e+07  1.402050e+07  ...          15.000000          0.910931
max    1.402050e+07  1.402050e+07  ...         518.000000          1.000000
std    2.722882e+03  4.333718e+03  ...          24.940581          0.256221

       SuccessfullRatioExcC2C  AppErrorRatio  UserErrorRatio  \
count            19663.000000   21461.000000    21461.000000
mean                 0.769655       0.072712        0.035646
min                  0.000000       0.000000        0.000000
25%                  0.685714       0.000000        0.000000
50%                  0.854838       0.015873        0.000000
75%                  0.958333       0.065476        0.018587
max                  1.000000       1.000000        1.000000
std                  0.261442       0.159385        0.114702

       BalanceErrorRatio  PaymentToolCardRatio  PaymentToolWalletRatio  \
count       21461.000000          20285.000000            20285.000000
mean            0.018635              0.937021                0.062040
min             0.000000              0.000000                0.000000
25%             0.000000              1.000000                0.000000
50%             0.000000              1.000000                0.000000
75%             0.008064              1.000000                0.000000
max             1.000000              1.000000                1.000000
std             0.072512              0.201744                0.200534

       PaymentToolTaliRatio   CMS_Balance
count          20285.000000  9.340000e+03
mean               0.000938  2.069254e+05
min                0.000000  0.000000e+00
25%                0.000000  0.000000e+00
50%                0.000000  0.000000e+00
75%                0.000000  9.500000e+04
max                1.000000  4.558500e+07
std                0.021943  9.211199e+05

[8 rows x 82 columns]
```

```
# Histograms for numerical features
data.hist(bins=50, figsize=(60, 35))
plt.show()

```



```
1 # Box plots for numerical features
2 plt.figure(figsize=(40, 15))
3 sns.boxplot(data=data.select_dtypes(include=['number']))
4 plt.xticks(rotation=45, ha='right')
5 plt.show()
```



Data Exploration

Correlation Matrix

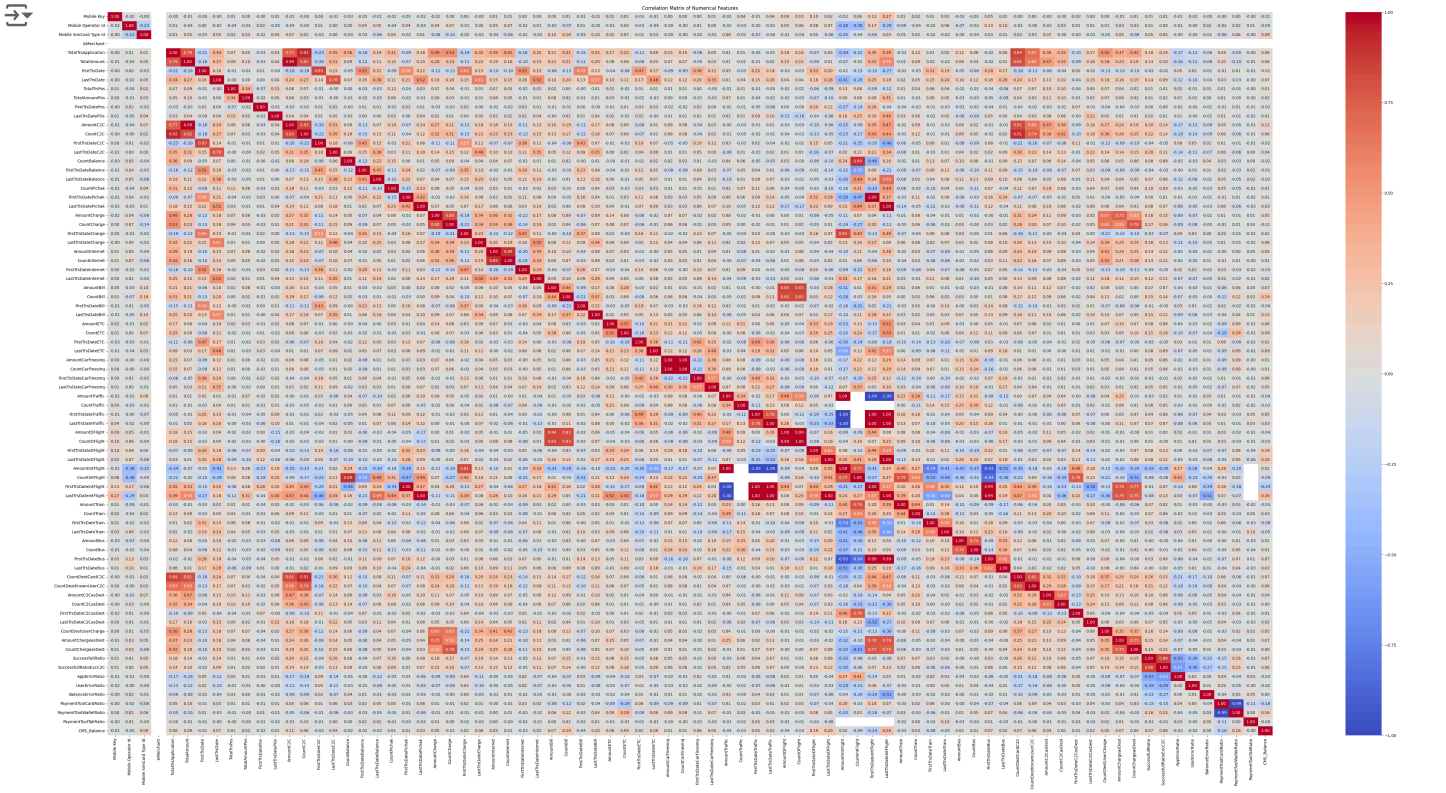
A correlation matrix is a valuable statistical tool used to understand relationships between multiple variables. It facilitates the identification of multicollinearity, helping to ensure the reliability of regression analyses by highlighting highly correlated independent variables. As part of exploratory data analysis (EDA), a correlation matrix provides insights into data structure and potential patterns, which can guide further modeling decisions. Moreover, it can assist in dimensionality reduction, enhancing model performance and efficiency by eliminating unnecessary complexity.

```
#First we need to find numerical features
numerical_df = data.select_dtypes(include=['number'])
numerical_df
```

	Mobile Key	Mobile Operator Id	Mobile SimCard Type Id	IsMerchant	TotalTrsApplication	TotalAmount	FirstTrsDate	LastTrsDate	TotalTrsPos	TotalAmount
0	2	2	2.0	0	18.0	8.131250e+06	14010107.0	14020326.0	NaN	N
1	4	1	1.0	0	NaN	NaN	NaN	NaN	NaN	N
2	6	1	1.0	0	NaN	NaN	NaN	NaN	NaN	N
3	7	1	1.0	0	290.0	1.692913e+09	14010101.0	14020501.0	46.0	9.920016e·
4	9	1	2.0	0	146.0	1.895458e+08	14010101.0	14020501.0	196.0	2.696955e·
...	...	...	...	...	...	...	...	...	...	
49995	125060	1	1.0	0	NaN	NaN	NaN	NaN	NaN	N
49996	125068	1	2.0	0	49.0	3.017794e+08	14010104.0	14020417.0	55.0	5.214060e·
49997	125071	1	1.0	0	NaN	NaN	NaN	NaN	NaN	N
49998	125072	1	1.0	0	NaN	NaN	NaN	NaN	NaN	N
49999	125074	2	1.0	0	NaN	NaN	NaN	NaN	NaN	N

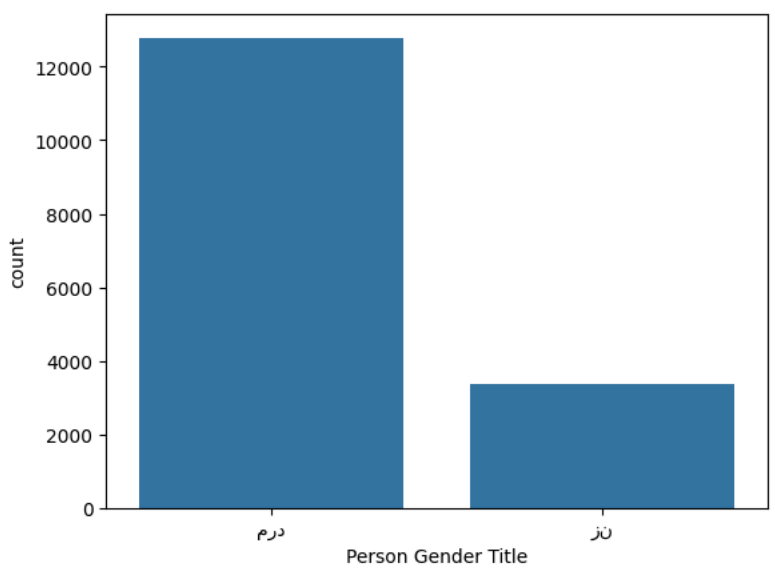
50000 rows × 80 columns

```
# prompt: Correlation matrix for numerical features

# Correlation matrix for numerical features
correlation_matrix = numerical_df.corr()
plt.figure(figsize=(66, 33))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix of Numerical Features')
plt.show()
```



We can also observe data disturbution by histogram charts

```

sns.countplot(x='Person Gender Title', data=data)
plt.show()
```



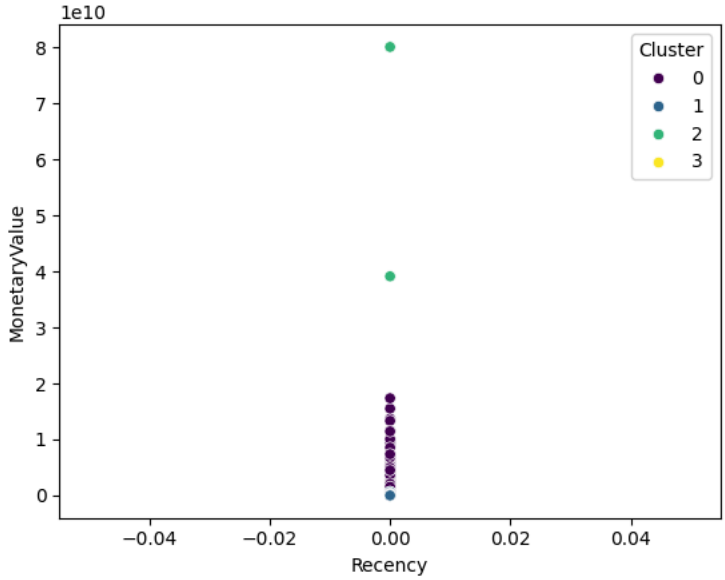

```
data.fillna(0, inplace=True)
```

```
<ipython-input-15-094b30ae9a00>:1: FutureWarning: Setting an item of incompatible dtype is deprecated and will raise an error in a f
  data.fillna(0, inplace=True)
```

Customer Segmentation:

```


# RFM Analysis
# Calculate Recency, Frequency, and Monetary Value
# Assuming 'Recency' is represented by 'LastTrsDate', 'Frequency' by 'TotalTrs', and 'MonetaryValue' by 'TotalAmountPos'
# Adapt column names as needed based on your actual dataset

# Convert 'LastTrsDate' to datetime if it's not already
data['LastTrsDate'] = pd.to_datetime(data['LastTrsDate'])

# Calculate Recency (days since last transaction)
data['Recency'] = (data['LastTrsDate'].max() - data['LastTrsDate']).dt.days


# Use existing columns for TotalTrsPos and Monetary Value
data['Frequency'] = data['TotalTrsPos']
data['MonetaryValue'] = data['TotalAmountPos']


# Scaling the RFM features
rfm_features = ['Recency', 'Frequency', 'MonetaryValue']
scaler = StandardScaler()
rfm_scaled = scaler.fit_transform(data[rfm_features])

# Perform K-means clustering
kmeans = KMeans(n_clusters=4, random_state=42, n_init=10)  # You can adjust the number of clusters
data['Cluster'] = kmeans.fit_predict(rfm_scaled)

# Analyze the clusters
print(data.groupby('Cluster')[rfm_features].mean())
# Visualize the clusters
sns.scatterplot(x='Recency', y='MonetaryValue', hue='Cluster', data=data, palette='viridis')
plt.show()

# Display the dataframe with clusters
data
```

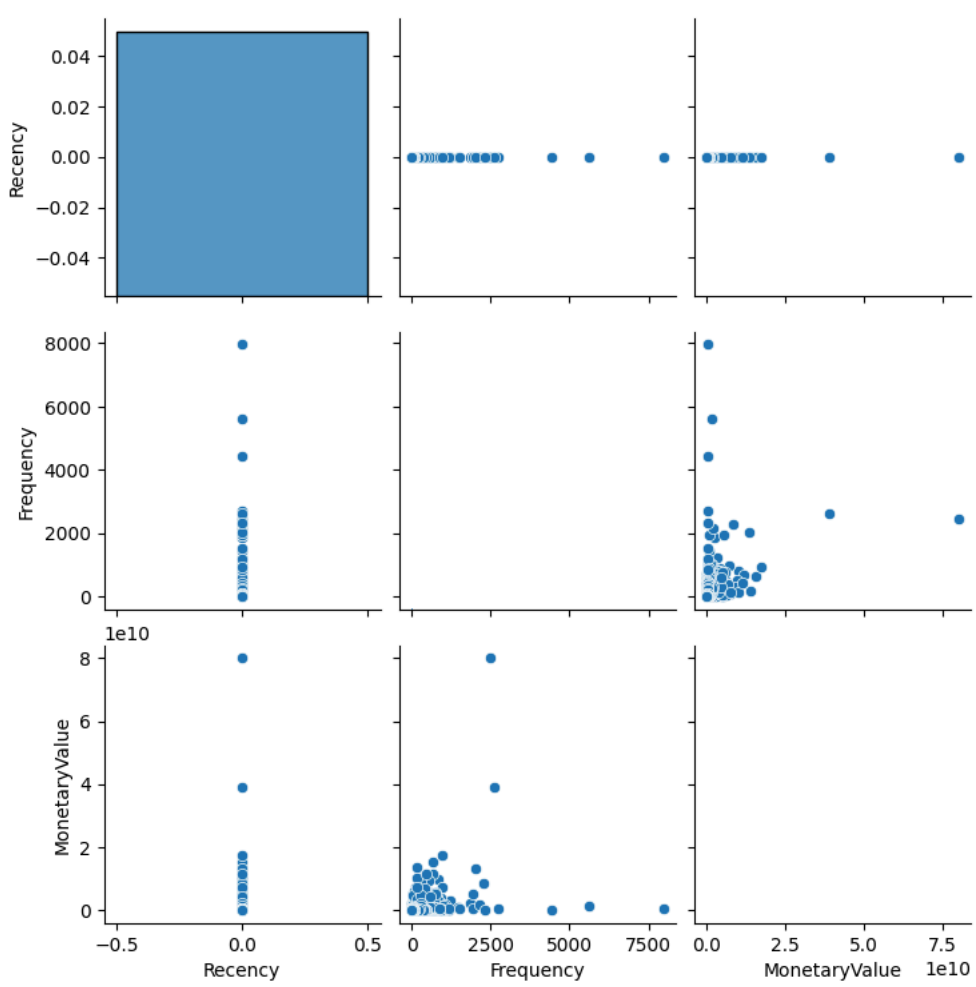
```
         Recency    Frequency  MonetaryValue
Cluster
0            0.0   263.405685   5.652000e+08
1            0.0    11.690899   1.913483e+07
2            0.0  2545.000000   5.958639e+10
3            0.0  6017.333333   7.196813e+08
```



	Mobile Key	Person Is Alive	Person Birth Date	Person Gender Title	Person First Name	Person Last Name	Mobile Number Encrypt	Mobile Number Masked	FirstAppInstallDate	Mobile Operator Id
0	2	True	1997-10-09 00:00:00	مرد	مجید	ملکی	HN8x8N88m+sikOUls4luxg==	0938***0926	2017-04-18 20:15:22.910	2
1	4	0	0	0	0	0	/Y38H9ETC2tMantBTYBQzw==	0913***9963	2014-12-15 21:59:03.870	1
2	6	0	0	0	0	0	ThuZBrT+/BueLw7LxSX15w==	0919***0132	2016-04-18 10:15:01.480	1
3	7	True	1979-07-13 00:00:00	مرد	حمیدرضا	عزیزی مربویه	0sEbFPeIKs9cCJLA73cOoQ==	0917***4959	2020-01-13 23:07:15.570	1
4	9	True	1979-03-11 00:00:00	مرد	امیدعلی	احمدپور	FZT2oVrc5UCFaN8gjELmCg==	0917***8979	2016-11-22 22:03:27.060	1
...	...	...	...	...	...	...	...	...	...	...
49995	125060	0	0	0	0	0	pDcHaQlw4rFir+7T1rBFEQ==	0915***7614	2017-11-16 21:45:59.757	1
49996	125068	True	1979-03-21 00:00:00	مرد	حمید	پارسای جرفی	/lv74VOVi1nIVBK006oiTA==	0915***6893	2018-07-06 19:38:36.493	1
49997	125071	0	0	0	0	0	SFqmPCgQQuhu9iyXJNxG+Q==	0914***7368	2021-09-04 20:47:17.843	1
49998	125072	0	0	0	0	0	QsN0ea3HyEgMsqZpmq/ywA==	0917***4369	2016-08-02 20:37:16.367	1
49999	125074	0	0	0	0	0	qpy0mX4gLOvrrzh7xhWuCg==	0936***5729	2017-12-23 22:42:00.613	2

50000 rows × 110 columns

```
1 sns.pairplot(data[['Recency', 'Frequency', 'MonetaryValue']])  # Example
2 plt.show()
```



Insights Generation

Here we present three meaningful insights that could help our company better understand our customer base and make informed business decisions

```
# insights that could help our company better understand our customer base and make informed
# business decisions.

# Insight 1: Identify high-value customers
high_value_cluster = data.groupby('Cluster')['MonetaryValue'].mean().idxmax()
print(f"High-value customer segment is Cluster {high_value_cluster}")
print(data[data['Cluster'] == high_value_cluster].describe())

# Insight 2: Analyze customer recency
low_recency_cluster = data.groupby('Cluster')['Recency'].mean().idxmin()
print(f"Customers who transacted recently are in Cluster {low_recency_cluster}")
print(data[data['Cluster'] == low_recency_cluster].describe())


# Insight 3:  Segmentation based on both MonetaryValue and Frequency
# Find the cluster with highest MonetaryValue and high Frequency
cluster_analysis = data.groupby('Cluster').agg({'MonetaryValue': 'mean', 'Frequency': 'mean'})
cluster_analysis['MonetaryValue_rank'] = cluster_analysis['MonetaryValue'].rank(ascending=False)
cluster_analysis['Frequency_rank'] = cluster_analysis['Frequency'].rank(ascending=False)

# Identify cluster with the highest combination of monetary and frequency
best_cluster = cluster_analysis[(cluster_analysis['MonetaryValue_rank'] <= 2) & (cluster_analysis['Frequency_rank'] <=2)].index.toli

print(f"Clusters with high monetary value and high frequency transactions are: {best_cluster}")

print(data[data['Cluster'].isin(best_cluster)].describe())
```

```
High-value customer segment is Cluster 2
          Mobile Key            FirstAppInstallDate  Mobile Operator Id  \
count       2.000000                              2            2.000000
mean    67353.500000  2018-07-02 06:06:56.436499968            1.500000
min     40944.000000     2017-12-15 22:38:42.913000            1.000000
25%     54148.750000  2018-03-25 02:22:49.674749952            1.250000
```

```
50%    67353.500000  2018-07-02 06:06:56.436499968           1.500000
75%    80558.250000  2018-10-09 09:51:03.198249984           1.750000
max    93763.000000     2019-01-16 13:35:09.960000           2.000000
std    37348.673075                             NaN           0.707107

       Mobile SimCard Type Id  IsMerchant  TotalTrsApplication   TotalAmount  \
count                2.000000         2.0                  2.0  2.000000e+00
mean                 1.500000         0.0                481.0  7.431266e+09
min                  1.000000         0.0                481.0  2.357801e+09
25%                  1.250000         0.0                481.0  4.894533e+09
50%                  1.500000         0.0                481.0  7.431266e+09
75%                  1.750000         0.0                481.0  9.967998e+09
max                  2.000000         0.0                481.0  1.250473e+10
std                  0.707107         0.0                  0.0  7.174963e+09

       FirstTrsDate                    LastTrsDate  TotalTrsPos  ...  \
count           2.0                              2      2.00000  ...
mean     14010101.0  1970-01-01 00:00:00.014015765   2545.00000  ...
min      14010101.0  1970-01-01 00:00:00.014011030   2472.00000  ...
25%      14010101.0  1970-01-01 00:00:00.014013397   2508.50000  ...
50%      14010101.0  1970-01-01 00:00:00.014015765   2545.00000  ...
75%      14010101.0  1970-01-01 00:00:00.014018133   2581.50000  ...
max      14010101.0  1970-01-01 00:00:00.014020501   2618.00000  ...
std             0.0                             NaN    103.23759  ...

       UserErrorRatio  BalanceErrorRatio  PaymentToolCardRatio  \
count        2.000000           2.000000                   2.0
mean         0.007311           0.025611                   1.0
min          0.001801           0.003603                   1.0
25%          0.004556           0.014607                   1.0
50%          0.007311           0.025611                   1.0
75%          0.010065           0.036615                   1.0
max          0.012820           0.047619                   1.0
std          0.007792           0.031124                   0.0

       PaymentToolWalletRatio  PaymentToolTaliRatio  CMS_Balance  Recency  \
count                     2.0                   2.0          2.0      2.0
mean                      0.0                   0.0          0.0      0.0
min                       0.0                   0.0          0.0      0.0
25%                       0.0                   0.0          0.0      0.0
50%                       0.0                   0.0          0.0      0.0
75%                       0.0                   0.0          0.0      0.0
max                       0.0                   0.0          0.0      0.0
std                       0.0                   0.0          0.0      0.0

        Frequency  MonetaryValue  Cluster
count     2.00000   2.000000e+00      2.0
mean   2545.00000   5.958639e+10      2.0
min    2472.00000   3.911289e+10      2.0
25%    2508.50000   4.934964e+10      2.0
50%    2545.00000   5.958639e+10      2.0
```

Presentation/Report

```
1
2 # Section 1: Executive Summary (Non-technical)
3 print("Executive Summary:")
4 print("Customer segmentation analysis reveals distinct groups based on their transaction behavior.")
5 print("Key insights include identifying high-value customers, understanding recent activity, and pinpointing segments with high moneta
6 print("These findings can inform targeted marketing strategies and personalized customer experiences.")
7
8 # Section 2: Data Overview (Technical/Non-technical)
9 print("\nData Overview:")
10 print(data.describe()) # Basic statistics for numerical features
11 print("\nMissing Values:")
12 print(data.isnull().sum()) # Display missing values before handling them
13
14 # Section 3: Customer Segmentation (Technical)
15 print("\nCustomer Segmentation (RFM Analysis):")
16 print(data.groupby('Cluster')[['Recency', 'Frequency', 'MonetaryValue']].mean())
17
18 # Visualization 1: Scatter plot for clusters
19 plt.figure(figsize=(8, 6))
20 sns.scatterplot(x='Recency', y='MonetaryValue', hue='Cluster', data=data, palette='viridis')
21 plt.title('Customer Segmentation based on Recency and Monetary Value')
22 plt.show()
23
24 # Visualization 2: Distribution of Monetary Value by Cluster
25 plt.figure(figsize=(8, 6))
26 sns.boxplot(x='Cluster', y='MonetaryValue', data=data)
27 plt.title('Distribution of Monetary Value across Clusters')
28 plt.show()
29
30 # Visualization 3: Distribution of Frequency by Cluster
31 plt.figure(figsize=(8, 6))
```

```
plt.figure(figsize=(8, 6))
sns.boxplot(x='Cluster', y='Frequency', data=data)
plt.title('Distribution of Frequency across Clusters')
plt.show()


# Section 4: Key Insights and Customer Segments (Non-technical/Technical)
print("\nKey Insights:")

print("\nHigh-Value Customers:")
high_value_cluster = data.groupby('Cluster')['MonetaryValue'].mean().idxmax()
print(f"High-value customer segment is Cluster {high_value_cluster}")
print(data[data['Cluster'] == high_value_cluster].describe())

print("\nRecent Customers:")
low_recency_cluster = data.groupby('Cluster')['Recency'].mean().idxmin()
print(f"Customers who transacted recently are in Cluster {low_recency_cluster}")
print(data[data['Cluster'] == low_recency_cluster].describe())


print("\nHigh Monetary & Frequency Customers:")
cluster_analysis = data.groupby('Cluster').agg({'MonetaryValue': 'mean', 'Frequency': 'mean'})
cluster_analysis['MonetaryValue_rank'] = cluster_analysis['MonetaryValue'].rank(ascending=False)
cluster_analysis['Frequency_rank'] = cluster_analysis['Frequency'].rank(ascending=False)
best_cluster = cluster_analysis[(cluster_analysis['MonetaryValue_rank'] <= 2) & (cluster_analysis['Frequency_rank'] <= 2)].index.tolis
print(f"Clusters with high monetary value and high frequency transactions are: {best_cluster}")
print(data[data['Cluster'].isin(best_cluster)].describe())


# Section 5: Recommendations (Non-technical)
print("\nRecommendations:")
print("Target high-value customer segments with premium offers and personalized promotions.")
print("Engage recently active customers with relevant product recommendations or loyalty programs.")
print("Focus on increasing transaction frequency from the identified high-potential customers.")

```