```
# Upload the paysim csv file
from google.colab import files
uploaded = files.upload()
```

Choose Files Paysim.csv

- **Paysim.csv**(text/csv) - 493534783 bytes, last modified: 7/16/2025 - 100% done

```
Saving Paysim.csv to Paysim.csv
```

```
Start coding or generate with AI.
```

```
#import libaries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Load the uploaded file
df = pd.read_csv('Paysim.csv', nrows=500000)
```

```
df.head()
```

	step	type	amt	customer	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedI
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	

```
#Rename column name
df = df.rename(columns={'nameOrig':'customer', 'amount':'amt'})
```

```
# Aggregate by customer
agg = df.groupby('customer').agg(
    total_amt=('amt','sum'),
    freq=('customer','count'),
```

```
        avg_amt=('amt', 'mean')
).reset_index()


print(agg)
```

```
            customer  total_amt  freq    avg_amt
0       C1000008582  315626.96     1  315626.96
1       C1000009135    3849.38     1    3849.38
2       C1000012640  367527.28     1  367527.28
3       C1000018663   12454.86     1   12454.86
4       C1000022742    2368.68     1    2368.68
...             ...        ...   ...        ...
499948   C999983233  244680.83     1  244680.83
499949   C999983733   10675.90     1   10675.90
499950   C999983894  154203.57     1  154203.57
499951   C999996950  299714.39     1  299714.39
499952   C999998175   37516.21     1   37516.21

[499953 rows x 4 columns]
```

```
#check column number and row
df.shape
```

```
(500000, 11)
```

```
#Count the unque customers
df['customer'].nunique()
```

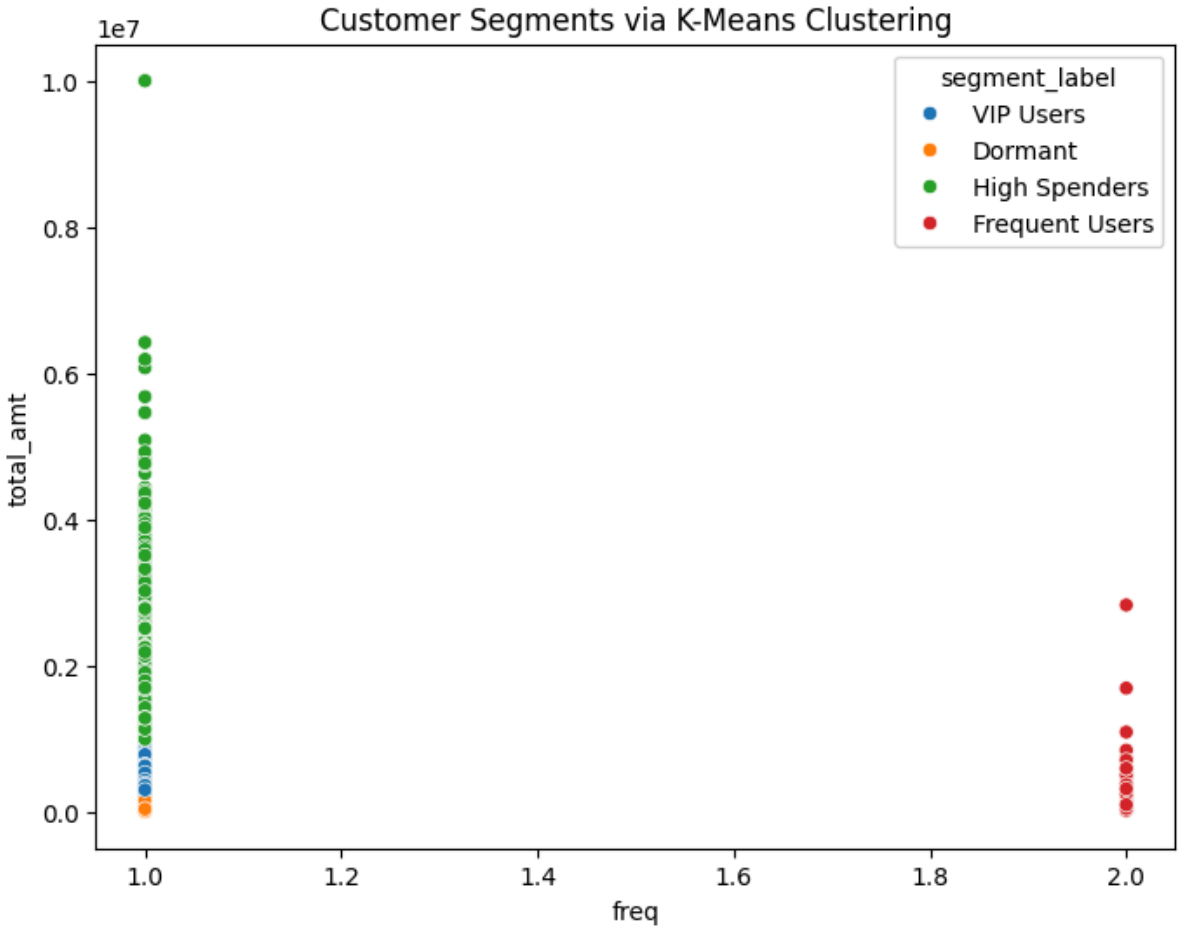
```
499953
```

```
#import the model libary
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

# Select the numeric features for clustering
X = agg[['freq', 'total_amt']].copy()

# Standardize the data to normalize scale
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

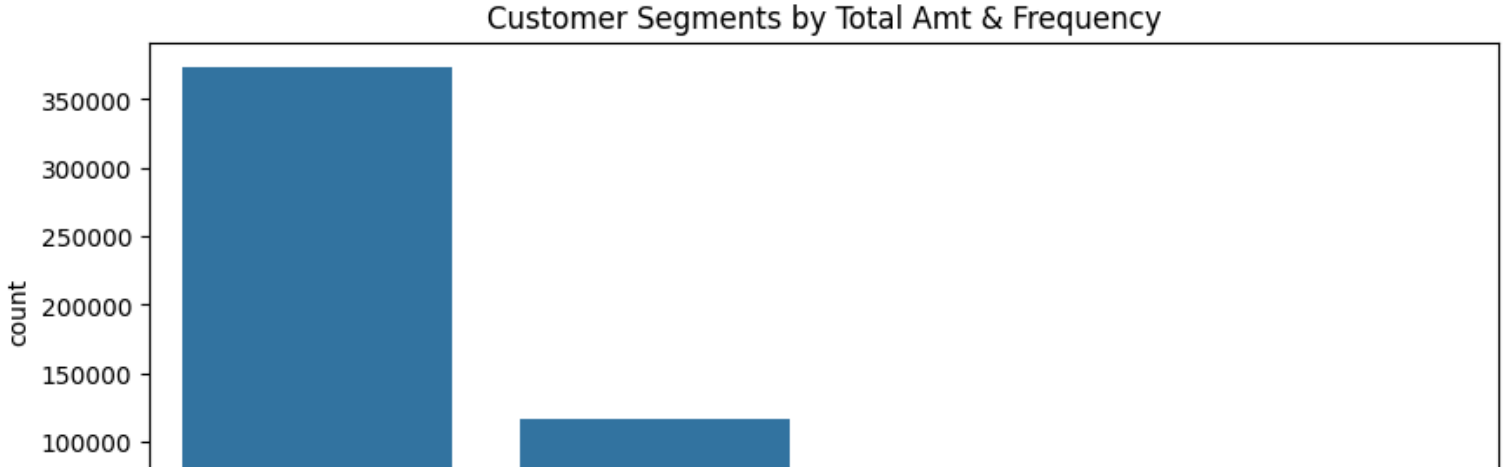
```
# Apply KMeans clustering
kmeans = KMeans(n_clusters=4, random_state=42)
agg['segment'] = kmeans.fit_predict(X_scaled)



# Preview cluster centers to understand patterns
import numpy as np

cluster_centers = scaler.inverse_transform(kmeans.cluster_centers_)
for i, center in enumerate(cluster_centers):
    print(f"Cluster {i}: Freq={center[0]:.2f}, Total_Amt={center[1]:.2f}")
```

```
Cluster 0: Freq=1.00, Total_Amt=62319.95
Cluster 1: Freq=2.00, Total_Amt=400707.18
Cluster 2: Freq=1.00, Total_Amt=1586738.60
Cluster 3: Freq=1.00, Total_Amt=377628.30
```

```
# Example mapping after analyzing centers (adjust as needed)
segment_map = {
    0: 'Dormant',
    1: 'Frequent Users',
    2: 'High Spenders',
    3: 'VIP Users'
}

agg['segment_label'] = agg['segment'].map(segment_map)



import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8,6))
sns.scatterplot(data=agg, x='freq', y='total_amt', hue='segment_label', palette='tab10')
plt.title('Customer Segments via K-Means Clustering')
plt.show()
```




```
# Plot distribution of segments
plt.figure(figsize=(10,4))
sns.countplot(data=agg, x='segment', order=agg['segment'].value_counts().index)
plt.xticks(rotation=45)
plt.title('Customer Segments by Total Amt & Frequency')
plt.show()
```




```
# Save DataFrame
agg.to_csv('segmented_customers.csv', index=False)

# Download it
from google.colab import files
files.download('segmented_customers.csv')
```