Practical 3

Problem Statement - Implement Agglomerative hierarchical clustering algorithm using appropriate dataset.

Name : Mugdha Itake
Roll No : 8029

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.decomposition import PCA
```

Step 2: Load the Datase

```
data_url = "/content/CC GENERAL.csv"
data = pd.read_csv(data_url)
data.head()
```

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	C
0	C10001	40.900749	0.818182	95.40	0.00	95.4	0.000000	0.166667	
1	C10002	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	0.000000	
2	C10003	2495.148862	1.000000	773.17	773.17	0.0	0.000000	1.000000	
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	0.083333	
4	C10005	817.714335	1.000000	16.00	16.00	0.0	0.000000	0.083333	

Next steps: Generate code with data | View recommended plots | New interactive sheet

Step 3: Preprocess the Data

1. Handle missing values: Fill or remove any missing values.

2. Feature selection: Select the relevant features for clustering.

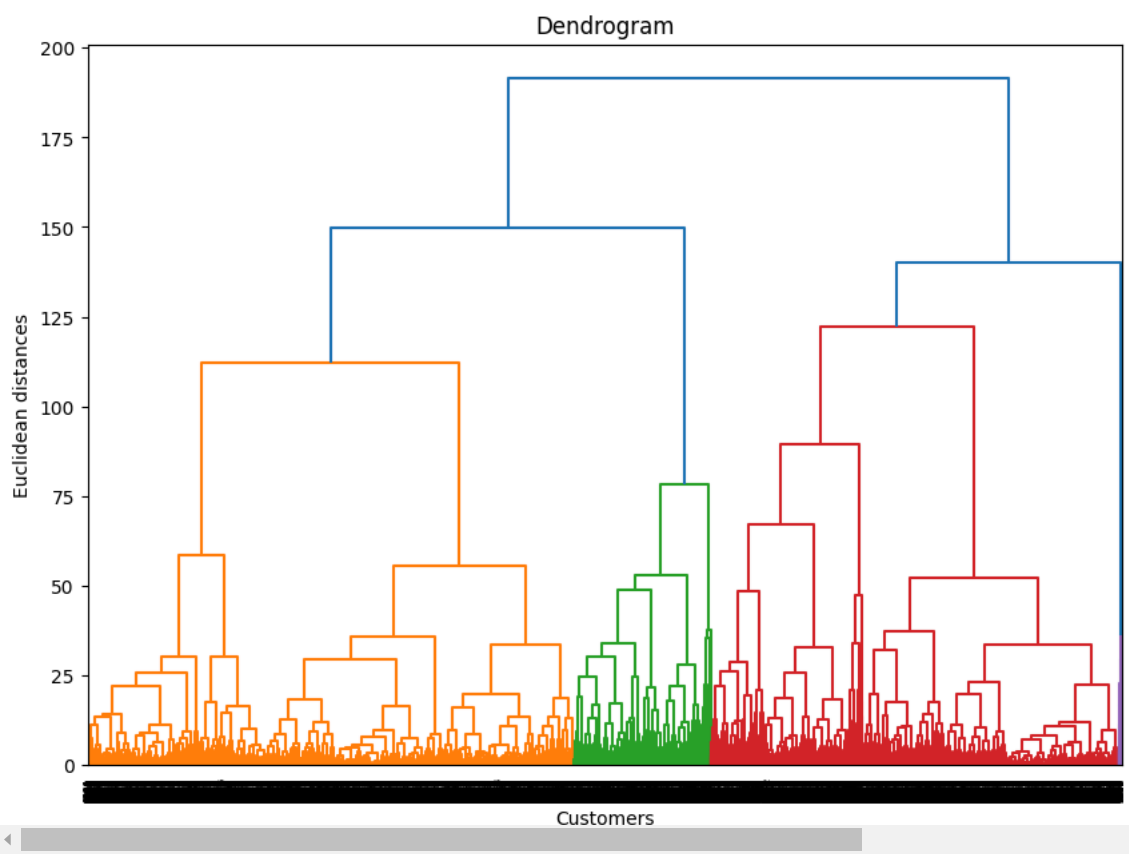
3. Scaling: Standardize the features to have zero mean and unit variance.

```
# Handling missing values (if any)
data.fillna(data.mode(), inplace=True)
data.fillna(data["CREDIT_LIMIT"].mode()[0], inplace=True)


# Selecting relevant features for clustering
features = data[['BALANCE', 'BALANCE_FREQUENCY', 'PURCHASES',
       'ONEOFF_PURCHASES', 'INSTALLMENTS_PURCHASES', 'CASH_ADVANCE',
       'PURCHASES_FREQUENCY', 'ONEOFF_PURCHASES_FREQUENCY',
       'PURCHASES_INSTALLMENTS_FREQUENCY', 'CASH_ADVANCE_FREQUENCY', 'CREDIT_LIMIT', 'PAYMENTS',
       ]]
features.dropna()

# Scaling the features
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)
print(np.isnan(scaled_features).sum())
print(np.isinf(scaled_features).sum())
```

```
0
0
```

Step 4: Hierarchical Clustering - Dendrogram

```
# Compute the linkage matrix
linked = linkage(scaled_features, method='ward')

# Plot the dendrogram
plt.figure(figsize=(10, 7))
dendrogram(linked, orientation='top', distance_sort='descending', show_leaf_counts=True)
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.show()
```



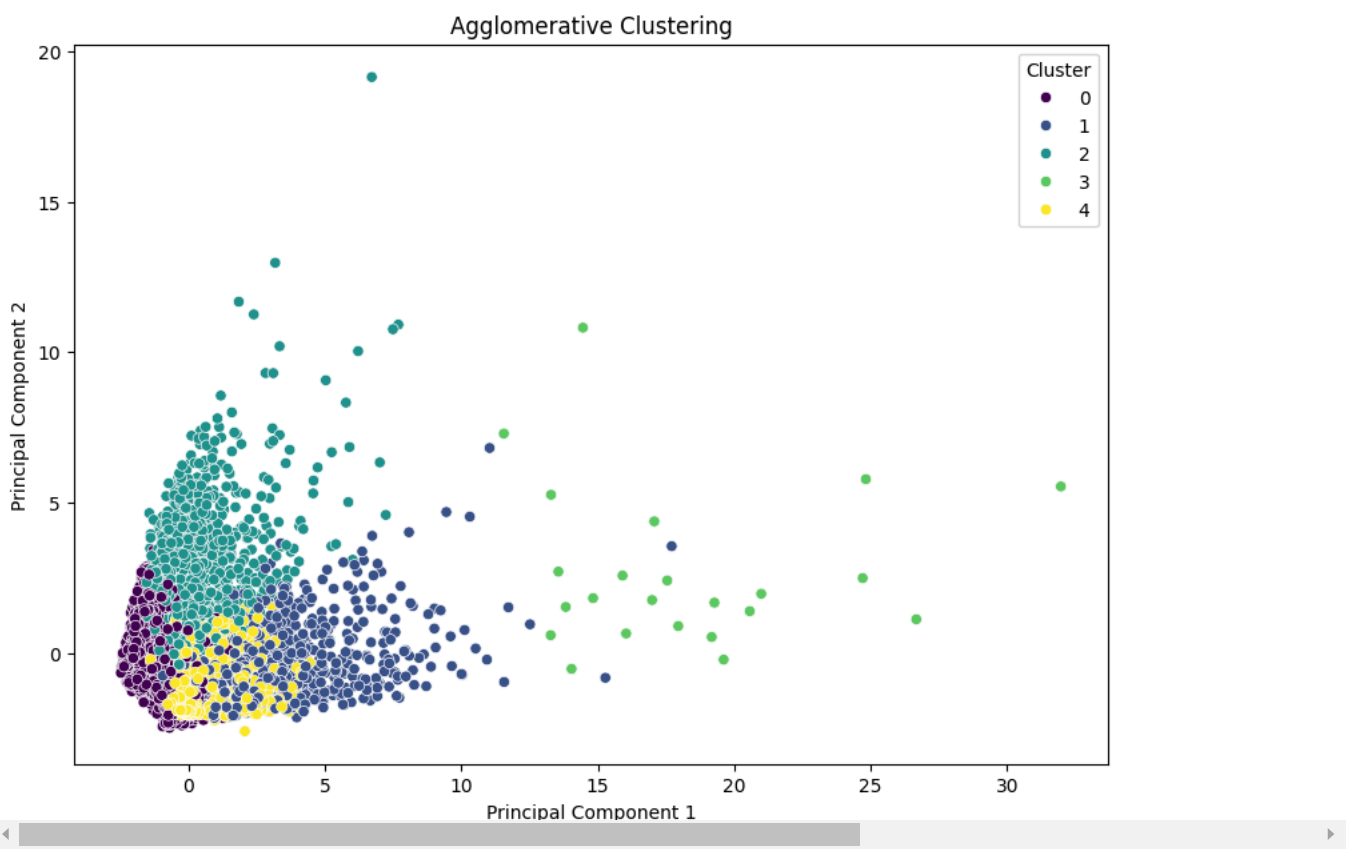
Step 5: Fit Agglomerative Clustering Model

```
# Fit the Agglomerative Clustering model
agg_clustering = AgglomerativeClustering(n_clusters=5, linkage='ward')
agg_clustering.fit(scaled_features)

# Adding the cluster labels to the original dataset
data['Cluster'] = agg_clustering.labels_
```

Step 6: Visualize the Clusters

Using PCA to reduce dimensions for visualization.

```
# Reduce dimensions with PCA for visualization
pca = PCA(n_components=2)
pca_features = pca.fit_transform(scaled_features)

# Plot the clusters
plt.figure(figsize=(10, 7))
sns.scatterplot(x=pca_features[:, 0], y=pca_features[:, 1], hue=data['Cluster'], palette='viridis')
plt.title('Agglomerative Clustering')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()
```



Step 7: Analyze and Interpret Clusters

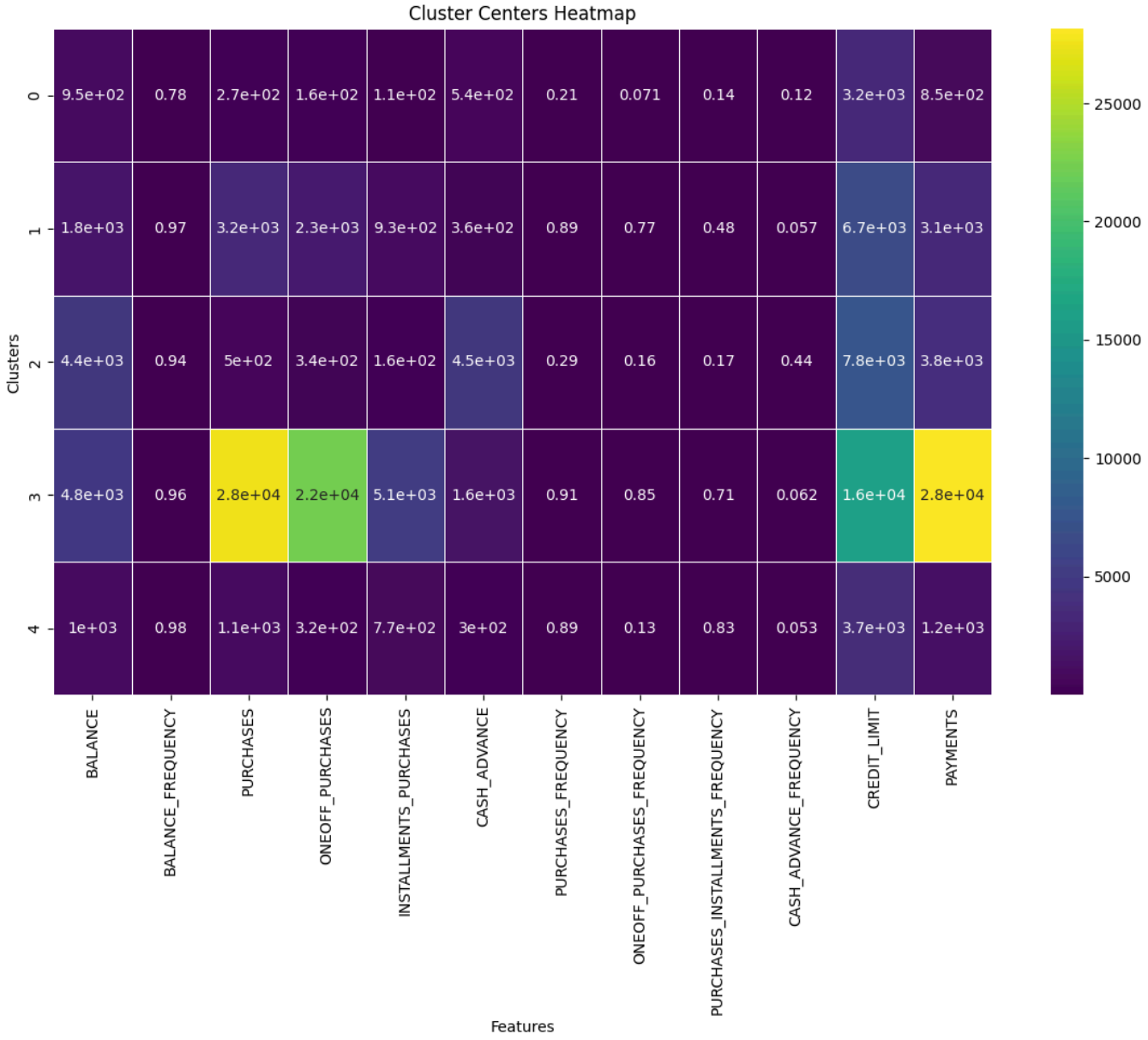
```
# Compute the cluster centers (only for numeric columns)
numeric_cols = features.columns
cluster_centers = data.groupby('Cluster')[numeric_cols].mean()

# Display the cluster centers
cluster_centers
```

Cluster	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	C
0	951.148163	0.776035	268.678913	163.815366	105.202602	542.948097	0.211207	
1	1837.774768	0.973742	3216.227288	2282.329511	934.393575	356.782203	0.885588	
2	4420.642815	0.938751	503.625794	342.074552	161.626368	4484.913975	0.288849	
3	4812.382778	0.956126	27505.339565	22417.452174	5087.887391	1617.786145	0.905072	
4	1010.549314	0.978763	1083.731058	316.893407	767.071935	296.860460	0.889732	

Next steps: Generate code with `cluster_centers` | View recommended plots | New interactive sheet

```
# Set the plot size
plt.figure(figsize=(14, 8))

# Create a heatmap for the cluster centers
sns.heatmap(cluster_centers, annot=True, cmap="viridis", linewidths=.5)
plt.title('Cluster Centers Heatmap')
plt.xlabel('Features')
plt.ylabel('Clusters')
plt.show()
```

[+]



Start coding or generate with AI.