DOCUMENTATION OF CLUSTERING CREDIT CARD

```python
# Data Visualization
sns.histplot(data=df, x='BALANCE')
plt.title('Distribution of Balance')
plt.show()

sns.histplot(data=df, x='PURCHASES')
plt.title('Distribution of Purchases')
plt.show()

sns.histplot(data=df, x='CASH_ADVANCE')
plt.title('Distribution of Cash Advances')
plt.show()

# Based on Income Levels
b = [0, 50000, 100000, 150000, 200000, np.inf]
lab = ['<50K', '50K-100K', '100K-150K', '150K-200K', '>200K']
df['INCOME_LEVEL'] = pd.cut(df['PURCHASES']/df['TENURE']*12, bins=b, labels=lab)

plt.figure(figsize=(12,6))
sns.histplot(x='BALANCE', hue='INCOME_LEVEL', data=df, kde=True)
plt.title('Balance Distribution for different Income Levels')
plt.show()

plt.figure(figsize=(12,6))
sns.histplot(x='PURCHASES', hue='INCOME_LEVEL', data=df, kde=True)
plt.title('Purchases Distribution for different Income Levels')
plt.show()

plt.figure(figsize=(12,6))
sns.boxplot(x='INCOME_LEVEL', y='BALANCE', data=df)
plt.title('Balance Distribution for different Income Levels')
plt.show()

plt.figure(figsize=(12,6))
sns.boxplot(x='INCOME_LEVEL', y='PURCHASES', data=df)
plt.title('Purchases Distribution for different Income Levels')
plt.show()

plt.hist(df['CREDIT_LIMIT'], bins=20)
plt.title('Credit Limit Distribution')
plt.xlabel('Credit Limit')
plt.ylabel('Frequency')
plt.show()
```

Basic analysis obtained from the given graphs:

- **Distribution of Balance:** The histogram shows the distribution of balance values in the dataset, indicating that most customers have a balance between 0 and 5000, and there are some customers with very high balances (over 15000).

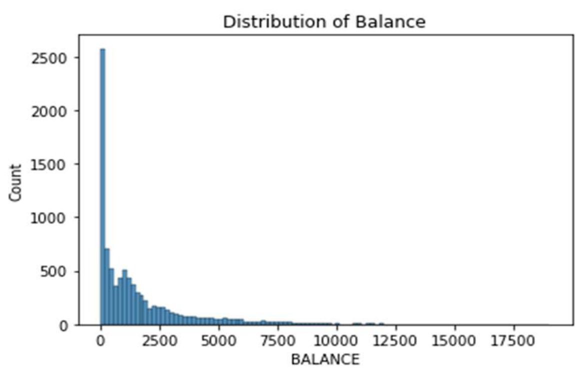
- **Distribution of Purchases:** The histogram shows the distribution of purchase values in the dataset, indicating that most customers have purchases between 0 and 2500, and there are some customers with very high purchase values (over 10000).
- **Distribution of Cash Advances:** The histogram shows the distribution of cash advance values in the dataset, indicating that most customers have cash

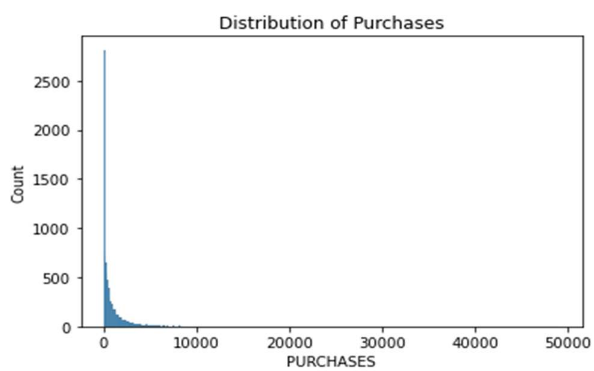
advances between 0 and 1000, and there are some customers with very high cash advance values (over 5000).

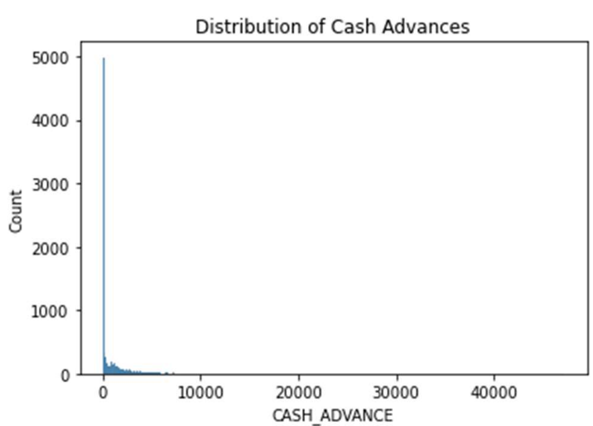
- **Balance and Purchases Distribution by Income Level:** The histograms show the distribution of balance and purchases values in the dataset, grouped by income level. The plots indicate that customers with higher income levels tend to have higher balances and make more purchases than customers with lower income levels.
- **Balance and Purchases Boxplot by Income Level:** The boxplots show the distribution of balance and purchases values in the dataset, grouped by income level. The plots indicate that customers with higher income levels tend to have higher median balances and make more purchases than customers with lower income levels.
- **Credit Limit Distribution:** The histogram shows the distribution of credit limit values in the dataset, indicating that most customers have a credit limit between 0 and 10000, and there are some customers with very high credit limits (over 20000).

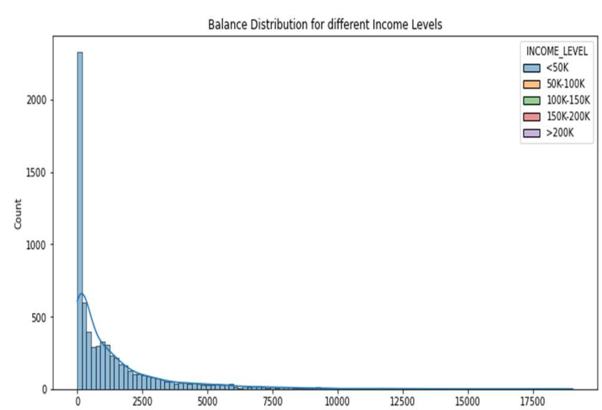
This can be useful in clustering analysis as it can help in identifying the relevant features for clustering and in defining the clusters.For example, the balance and purchases distribution for different income levels suggest that these features could be important in clustering customers. The credit limit distribution could also be useful in identifying different groups of customers based on their creditworthiness.

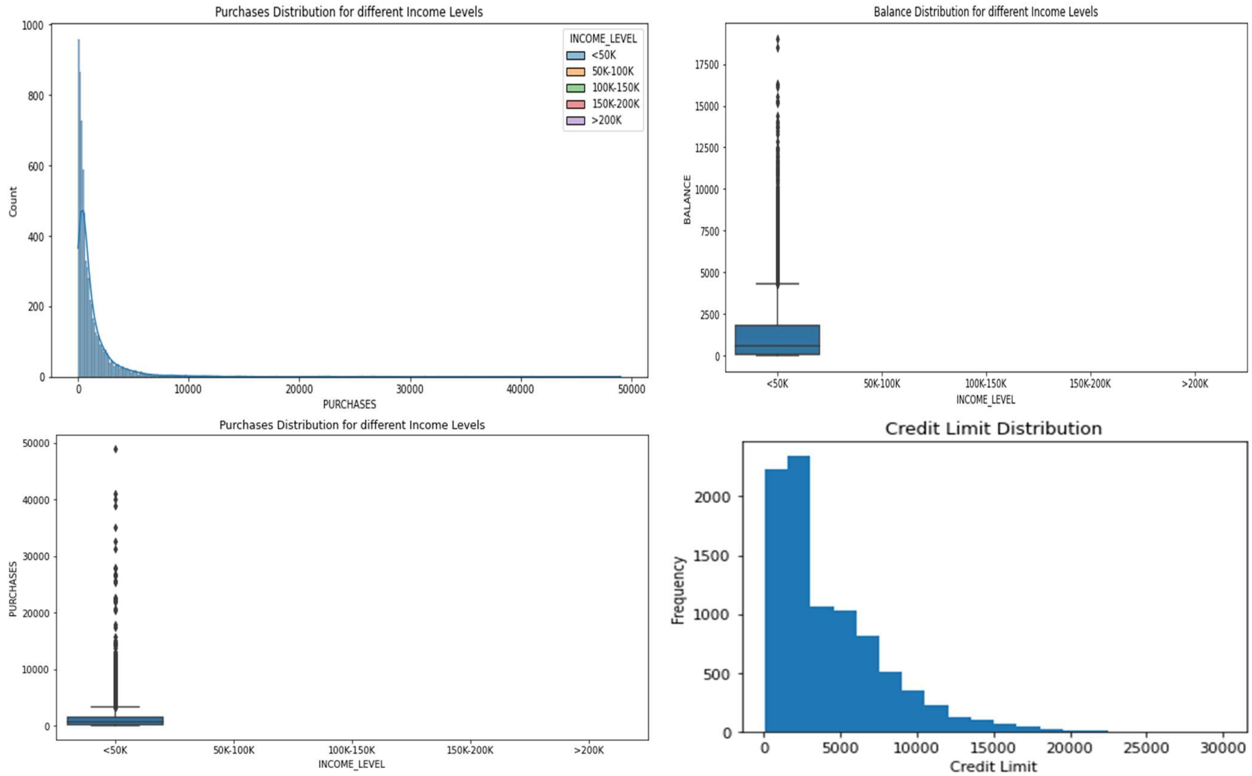
Graphs:












```
# Clustering

wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(df_sc)
    wcss.append(kmeans.inertia_)

plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()

km = KMeans(n_clusters=3, init='k-means++', max_iter=300, n_init=10, random_state=0)
py = kmeans.fit_predict(df_sc)

df['Cluster'] = py
# Analyzing
df_cluster_0 = df[df['Cluster']==0]
df_cluster_1 = df[df['Cluster']==1]
df_cluster_2 = df[df['Cluster']==2]
```

This code is performing K-means clustering on the credit card dataset. The purpose of K-means clustering is to group similar customers together based on their purchasing behavior and other characteristics.

This code is using the elbow method to determine the optimal number of clusters to use for the K-means algorithm. This involves running the algorithm for a range of cluster numbers and plotting the within-cluster sum of squares (WCSS)* against the number of clusters. The "elbow" point on the graph is where the rate of decrease in WCSS slows down, indicating that additional clusters do not significantly improve the clustering performance. This elbow point is used to determine the optimal number of clusters for the data.

*(WCSS stands for "within-cluster sum of squares". In K-means clustering, WCSS is used as a metric to evaluate how well the data points within each cluster are grouped together.)

```
# Visulization
# 1. Hierarchical cluster
Z = linkage(df_sc, method='ward')
plt.figure(figsize=(12, 6))
dendrogram(Z)
plt.show()

# 2. DBSCAN cluster
db = DBSCAN(eps=0.5, min_samples=5)
clust = db.fit_predict(df_sc)
plt.scatter(df_sc[:,0], df_sc[:,1], c=clust, cmap='rainbow')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.show()

# 3. Spectral cluster
sp_c = SpectralClustering(n_clusters=3, affinity='nearest_neighbors', assign_labels='kmeans')
sp_c = sp_c.fit_predict(df_sc)
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df_sc[:, 0], df_sc[:, 1], df_sc[:, 2], c=sp_c, cmap='rainbow')
plt.show()

from sklearn.metrics import silhouette_score, silhouette_samples
import matplotlib.cm as cm

# silhouette
km = KMeans(n_clusters=3)
km.fit(df_sc)
labels = km.labels_
sil = silhouette_samples(df_sc, km.labels_)
avg = silhouette_score(df_sc, km.labels_)
fig, ax = plt.subplots(figsize=(8,6))
y_lower, y_upper = 0, 0
for i, cl in enumerate(np.unique(km.labels_)):
    cl_vals = sil[km.labels_ == cl]
    cl_vals.sort()
    y_upper += len(cl_vals)
    col = cm.nipy_spectral(float(i) / len(np.unique(km.labels_)))
    ax.barh(range(y_lower, y_upper), cl_vals, height=1.0,
            edgecolor='none', color=col)
    y_lower += len(cl_vals)
ax.axvline(avg, color="red", linestyle="--")
ax.set_yticks([])
ax.set_xlim([-0.1, 1.0])
ax.set_xlabel("Silhouette coefficient values")
ax.set_ylabel("Cluster label")

plt.show()
```

From the graphs, following conclusions are drawn :

- **Elbow Method:** The elbow method is used to determine the optimal number of clusters for k-means clustering. In this case, the graph shows a sharp decrease in the Within-Cluster-Sum-of-Squares (WCSS) until 3 clusters, after which the decrease is less steep. Therefore, the optimal number of clusters for this dataset could be 3.
- **Hierarchical Clustering:** The dendrogram shows the relationship between the data points and how they are grouped based on their similarity. The height of the dendrogram represents the distance between the clusters. From the dendrogram, we can see that there are three main clusters.
- **DBSCAN:** Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a clustering algorithm that groups together points that are close to each other. In this case,

we can see that there are three main clusters and some noise points that are not part of any cluster.

- **Spectral Clustering:** Spectral clustering is a clustering algorithm that uses the eigenvectors of a similarity matrix to perform dimensionality reduction before clustering. The 3D scatter plot shows the clusters formed by the algorithm. From the plot, we can see that there are three distinct clusters.
- **Silhouette Analysis:** The silhouette analysis measures how well each data point fits into its assigned cluster. The graph shows the silhouette coefficient for each cluster, and a red line represents the average silhouette coefficient for all clusters. In this case, all clusters have a silhouette coefficient above the average, indicating that the clustering is effective.

GRAPHS: