In [1]:
```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy
from sklearn.metrics import silhouette_samples, silhouette_score
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import KMeans
sns.set()
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler, MinMaxScaler, RobustScaler, MaxAbsScaler, OneHotEncoder, OrdinalEncoder, Polync
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import matplotlib.animation as animation
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns
import plotly.express as px
import warnings
warnings.filterwarnings('ignore')
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.metrics.pairwise import cosine_similarity
import missingno as msno
from sklearn.metrics import silhouette_score
from sklearn.metrics import pairwise_distances
from sklearn.metrics import davies_bouldin_score
from sklearn.metrics import calinski_harabasz_score
```

In [2]:
```
df = pd.read_csv('https://raw.githubusercontent.com/dentawina/Dataset/main/cellphone_price.csv')
df.head()
```

Out[2]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time
0	842	0	2.2	0	1	0	7	0.6	188	2	2	20	756	2549	9	7	19
1	1021	1	0.5	1	0	1	53	0.7	136	3	6	905	1988	2631	17	3	7
2	563	1	0.5	1	2	1	41	0.9	145	5	6	1263	1716	2603	11	2	9
3	615	1	2.5	0	0	0	10	0.8	131	6	9	1216	1786	2769	16	8	11
4	1821	1	1.2	0	13	1	44	0.6	141	2	14	1208	1212	1411	8	2	15

In [3]:
```
df.shape
```

Out[3]:
```
(3000, 20)
```

Dataset Cellphone ini memiliki 3000 Baris dan 20 Kolom

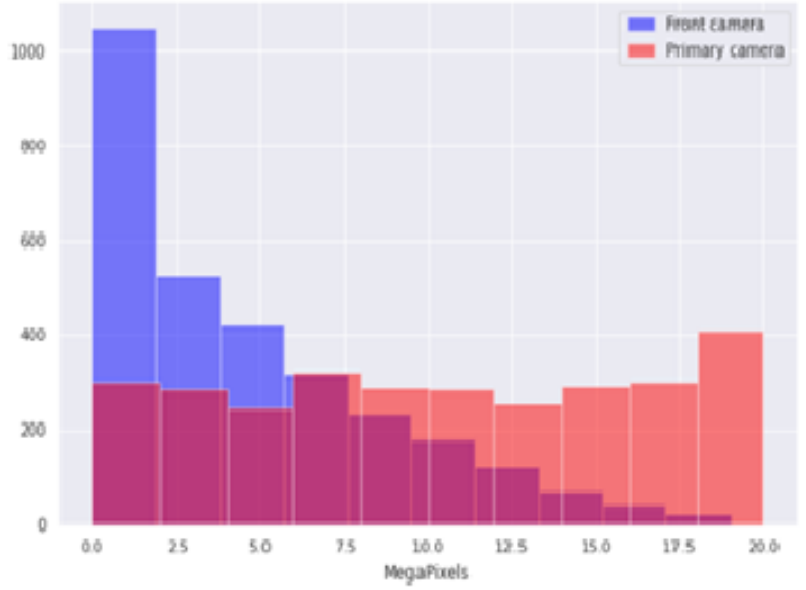
In [4]:
```
df.describe()
```

Out[4]:

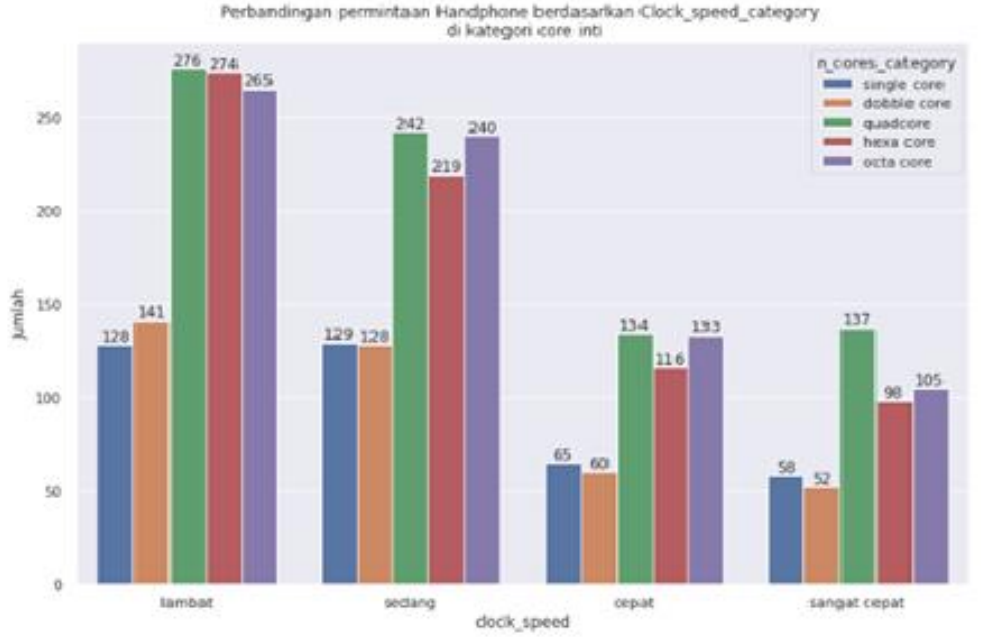
	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc
count	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000
mean	1241.849000	0.502000	1.528467	0.512000	4.404000	0.510000	32.581667	0.507000	140.003000	4.456333	9.962333
std	437.063804	0.500079	0.820358	0.499939	4.383742	0.499983	18.152810	0.285969	35.213809	2.289361	6.073923
min	500.000000	0.000000	0.500000	0.000000	0.000000	0.000000	2.000000	0.100000	80.000000	1.000000	0.000000
25%	863.750000	0.000000	0.700000	0.000000	1.000000	0.000000	16.000000	0.200000	109.000000	2.000000	5.000000
50%	1232.000000	1.000000	1.500000	1.000000	3.000000	1.000000	33.000000	0.500000	140.000000	4.000000	10.000000

In [12]:
```
plt.figure(figsize=(10,6))
df1['fc'].hist(alpha=0.5,color='blue',label='Front camera')
df1['pc'].hist(alpha=0.5,color='red',label='Primary camera')
plt.legend()
plt.xlabel('MegaPixels')
```

Out[12]: Text(0.5, 0, 'MegaPixels')



In [14]:
```
fig, ax = plt.subplots(figsize = (12,8))
_ = sns.countplot(data = df1, x = 'clock_speed_category', hue = 'n_cores_category')
_ = ax.set_ylabel('Jumlah')
_ = ax.set_xlabel('clock_speed')
_ = ax.set_title('Perbandingan permintaan Handphone berdasarkan Clock_speed_category \n di kategori core inti')
for i in ax.containers:
    ax.bar_label(i)
```



In [16]:
```
from sklearn import preprocessing
rs = RobustScaler()
X_scaled = pd.DataFrame(rs.fit_transform(df), columns = rs.feature_names_in_)
X_scaled
```

Out[16]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_
0	-0.516385	-1.0	0.4375	-1.0	-0.333333	-1.0	-0.81250	0.166667	0.786885	-0.50	-0.8	-0.830851	-0.639792	0.216734	-0.4285
1	-0.279378	0.0	-0.6250	0.0	-0.500000	0.0	0.62500	0.333333	-0.065574	-0.25	-0.4	0.520809	0.962289	0.269999	0.7142
2	-0.885799	0.0	-0.6250	0.0	-0.166667	0.0	0.25000	0.666667	0.081967	0.25	-0.4	1.067583	0.608583	0.245884	-0.1428
3	-0.818948	0.0	0.6250	-1.0	-0.500000	-1.0	-0.71875	0.500000	-0.147541	0.50	-0.1	0.986800	0.696610	0.335493	0.5714
4	0.779874	0.0	-0.1875	-1.0	1.666667	0.0	0.34375	0.166667	0.016393	-0.50	0.4	0.983582	-0.046814	-0.397571	-0.5714
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2995	0.619682	0.0	0.2500	-1.0	-0.500000	0.0	0.65625	0.000000	0.491803	0.75	0.7	0.122184	-0.435831	-0.014305	0.2857
2996	-0.824892	-1.0	0.1875	0.0	-0.500000	-1.0	-0.62500	0.666667	0.754098	0.00	-0.8	0.898053	0.499350	-0.115789	-0.5714
2997	-0.062231	-1.0	-0.0625	-1.0	-0.333333	0.0	-0.78125	0.000000	-0.983607	-0.75	0.2	-0.132875	-0.550065	-0.499055	-1.0000
2998	0.308544	0.0	-0.6250	0.0	-0.500000	-1.0	0.53125	-0.166667	0.508197	-0.50	0.2	-0.803380	-0.540962	0.106142	0.4285
2999	0.050314	0.0	-0.6250	-1.0	0.166667	0.0	0.06250	-0.666667	0.000000	0.50	0.9	-0.163421	-0.832250	0.367341	-0.4285

3000 rows × 20 columns

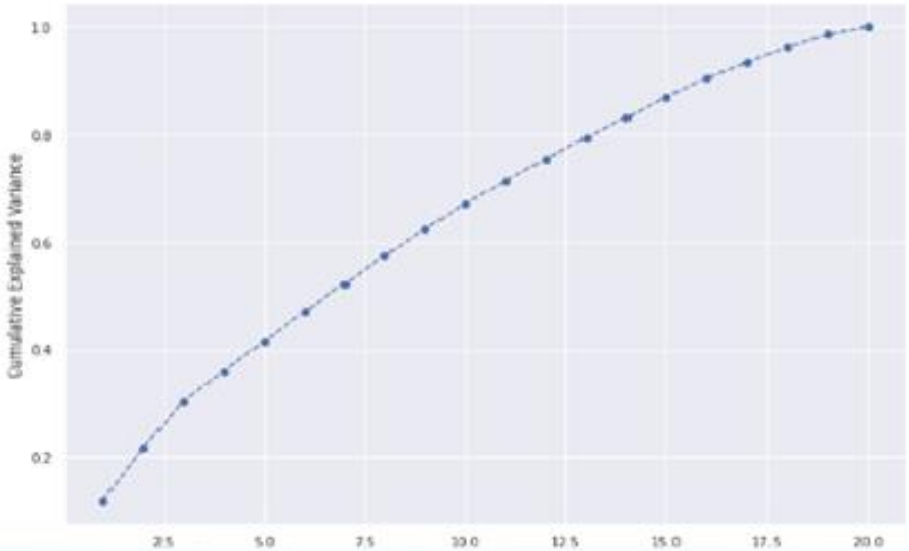
PCA

In [17]:
```
from sklearn.decomposition import PCA

pca = PCA()
pca.fit(x_scaled)

pca.explained_variance_ratio_
```

Out[17]:
```
array([0.11934082, 0.09644618, 0.08784264, 0.05601789, 0.05511328,
       0.05438372, 0.05337881, 0.05169493, 0.04972215, 0.04698247,
       0.0428622 , 0.04009082, 0.0397311 , 0.03810098, 0.03786052,
       0.03527368, 0.02946613, 0.02779307, 0.02385453, 0.01404408])
```

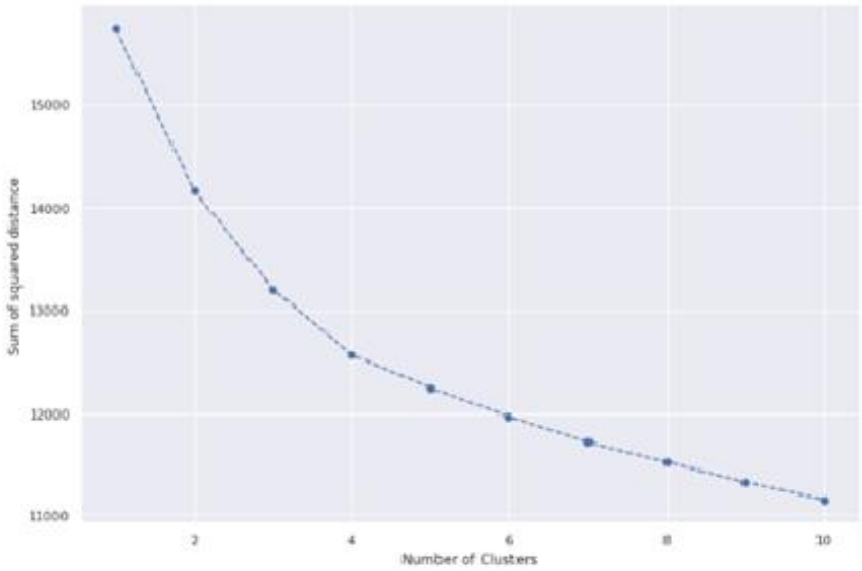
In [18]:
```
plt.figure(figsize=(12, 8))
plt.plot(range(1, 21), pca.explained_variance_ratio_.cumsum(), marker='o', linestyle='--')
plt.xlabel("Number of Components")
plt.ylabel("Cumulative Explained Variance")
```

Out[18]:
```
Text(0, 0.5, 'Cumulative Explained Variance')
```



Elbow Method

In [22]:
```
plt.figure(figsize=(12, 8))
plt.plot(results.keys(), results.values(), marker='o', linestyle='--')
plt.xlabel('Number of Clusters')
plt.ylabel('Sum of squared distance')
plt.show()
```



Kmeans

In [23]:
```
kmeans_pca = KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300, n_clusters=4, n_init=10, random_state=42, tol
kmeans_pca.fit(pca_scores)
#Labels = K_means.labels_
#df_pca_components['cluster_labels'] = Labels
#df_pca_components.head()
```

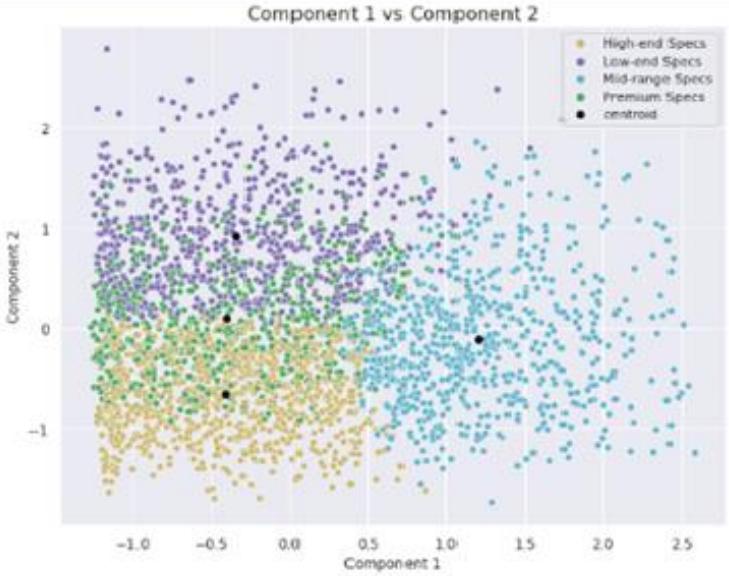
Out[23]:
```
KMeans(algorithm='auto', n_clusters=4, n_init=10, random_state=42)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [24]:
```
df_segm_pca = pd.concat([df.reset_index(drop=True), pd.DataFrame(pca_scores)], axis=1)
df_segm_pca.columns.values[-14:] = ['component 1', 'component 2', 'component 3','component 4','component 5','component 6','compo
df_segm_pca['K-means PCA'] = kmeans_pca.labels_

df_segm_pca
```

Out[24]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	component 6	component 7	component 8	component 9	
0	842	0	2.2	0	1	0	7	0.6	188	2	...	0.301062	-0.720638	0.443895	-0.536457	
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	-0.010220	0.226578	-0.702242	0.330210	
2	563	1	0.5	1	2	1	41	0.9	145	5	...	0.530198	0.670388	-0.218829	0.388684	
3	615	1	2.5	0	0	0	10	0.8	131	6	...	0.428693	0.688432	0.589876	-0.537256	
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	-0.467230	-0.709028	-0.502300	0.364508	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2995	1700	1	1.9	0	0	1	54	0.5	170	7	...	-0.380253	-0.342329	0.530085	1.130293	
2996	609	0	1.8	1	0	0	13	0.9	186	4	...	0.732138	-0.034219	0.481390	-0.227193	
2997	1185	0	1.4	0	1	1	8	0.5	80	1	...	0.081305	-0.184183	-0.776820	-1.116574	
2998	1533	1	0.5	1	0	0	50	0.4	171	2	...	-0.493674	-0.547556	-0.271315	0.220546	
2999	1270	1	0.5	0	4	1	35	0.1	140	6	...	0.083413	0.192434	0.088017	0.289263	

In [27]:
```
# Menggunakan warna yang lebih terang
plt.figure(figsize=(10, 8))
sns.scatterplot(
    x=df_segm_pca['component 1'],
    y=df_segm_pca['component 2'],
    hue=df_segm_pca['Segment'],
    palette=['y','m','c','g']
)

centers=np.array(kmeans_pca.cluster_centers_)
plt.scatter(centers[:,0],centers[:,1],color='black',alpha=0.9,label = 'centroid')

plt.title('Component 1 vs Component 2', fontsize=16)
plt.xlabel('Component 1', fontsize=12)
plt.ylabel('Component 2', fontsize=12)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.legend()
plt.show()
```



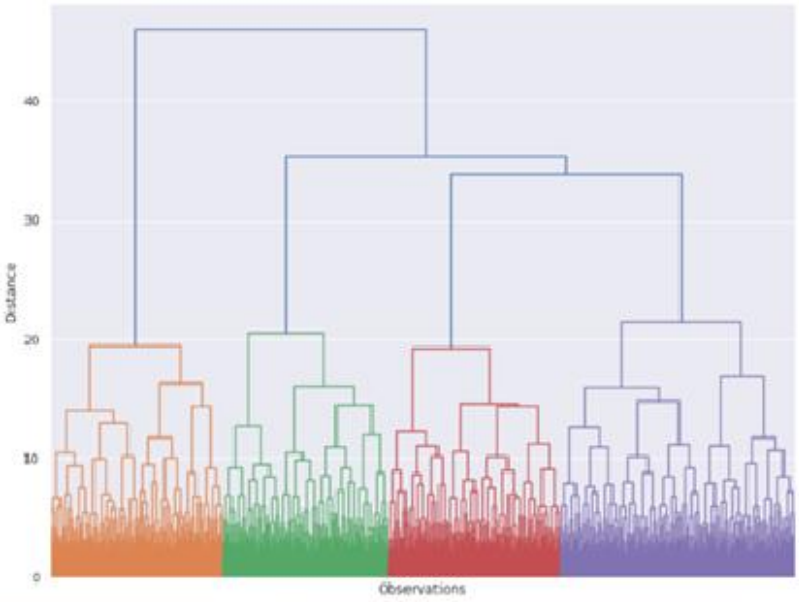
In [30]:
```
def highlight_max_min(s):

    is_max = s == s.max()
    is_min = s == s.min()
    return ['background-color: red' if v else 'background-color: lightgreen' if is_min[i] else '' for i,v in enumerate(is_max)]

# Calculate the mean value for each feature within each cluster
profil = df.groupby('cluster').mean().round(2).T

# Highlight the maximum and minimum values in each column
profil_styled = profil.style.apply(highlight_max_min, axis=1)

# Display the styled DataFrame
profil_styled
```

Out[30]:

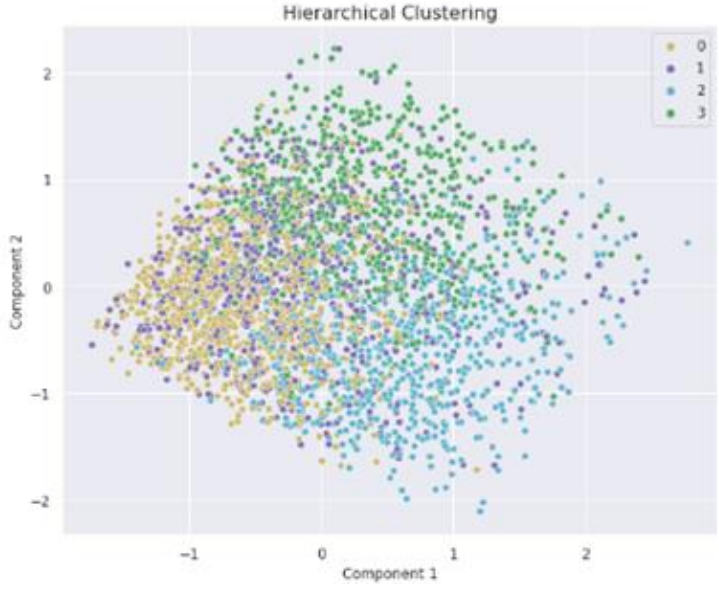
cluster	0	1	2	3
battery_power	1259.740000	1267.070000	1244.260000	1194.640000
blue	0.490000	0.490000	0.520000	0.510000
clock_speed	1.530000	1.570000	1.560000	1.440000
dual_sim	0.520000	0.520000	0.520000	0.500000
fc	2.680000	10.540000	2.250000	2.470000
four_g	0.500000	0.500000	0.520000	0.530000
int_memory	32.450000	31.580000	32.050000	34.460000
m_dep	0.520000	0.510000	0.510000	0.490000
mobile_wt	138.980000	141.540000	140.910000	138.190000
n_cores	4.310000	4.580000	4.490000	4.430000
pc	8.180000	16.350000	7.780000	7.850000
px_height	1220.880000	574.470000	413.360000	437.520000
px_width	1673.080000	1202.170000	1086.540000	1090.250000
ram	2134.170000	2145.730000	2073.220000	2180.520000
sc_h	12.600000	12.180000	9.110000	15.920000
sc_w	5.710000	5.120000	2.750000	9.830000
talk_time	11.140000	10.880000	11.020000	11.110000
three_g	0.750000	0.750000	0.780000	0.760000
touch_screen	0.500000	0.500000	0.510000	0.500000
wifi	0.530000	0.490000	0.480000	0.540000

Hieritical

In [31]:
```
h_cluster = linkage(pca_scores, method='ward')

plt.figure(figsize=(12, 9))
plt.xlabel('Observations')
plt.ylabel('Distance')
dendrogram(h_cluster,
           show_leaf_counts=False,
           no_labels=True)
plt.show()
```




```
plt.figure(figsize=(10, 8))
sns.scatterplot(
    x=1,
    y=2,
    data=df_hc,
    hue='hc_PCA',
    palette=['y','m','c','g']
)

plt.title('Hierarchical Clustering', fontsize=16)
plt.xlabel('Component 1', fontsize=12)
plt.ylabel('Component 2', fontsize=12)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

# Menampilkan legenda
plt.legend(fontsize=12)

plt.show()
```



```python
def create_comparison_table(data, kmeans_model, hierarchical_model):
    # Calculate scores for K-Means Clustering
    kmeans_silhouette_score = silhouette_score(data, kmeans_model.labels_, metric='manhattan')
    kmeans_calinski_harabasz_score = calinski_harabasz_score(data, kmeans_model.labels_)
    kmeans_davies_bouldin_score = davies_bouldin_score(data, kmeans_model.labels_)

    # Calculate scores for Hierarchical Clustering
    hierarchical_silhouette_score = silhouette_score(data, hierarchical_model.labels_, metric='manhattan')
    hierarchical_calinski_harabasz_score = calinski_harabasz_score(data, hierarchical_model.labels_)
    hierarchical_davies_bouldin_score = davies_bouldin_score(data, hierarchical_model.labels_)

    # Create table
    table_data = {'Metric': ['Silhouette Score', 'Calinski-Harabasz Index', 'Davies-Bouldin Index'],
                  'K-Means Clustering': [kmeans_silhouette_score, kmeans_calinski_harabasz_score, kmeans_davies_bouldin_score],
                  'Hierarchical Clustering': [hierarchical_silhouette_score, hierarchical_calinski_harabasz_score, hierarchical_

    comparison_table = pd.DataFrame(table_data)

    return comparison_table
```

```python
comparison_table = create_comparison_table(df, kmeans_pca, model)

print(comparison_table)
```

```
                  Metric  K-Means Clustering  Hierarchical Clustering
0        Silhouette Score            0.025262                 0.002105
1  Calinski-Harabasz Index           99.327709                62.666941
2    Davies-Bouldin Index           13.527218                18.277674
```