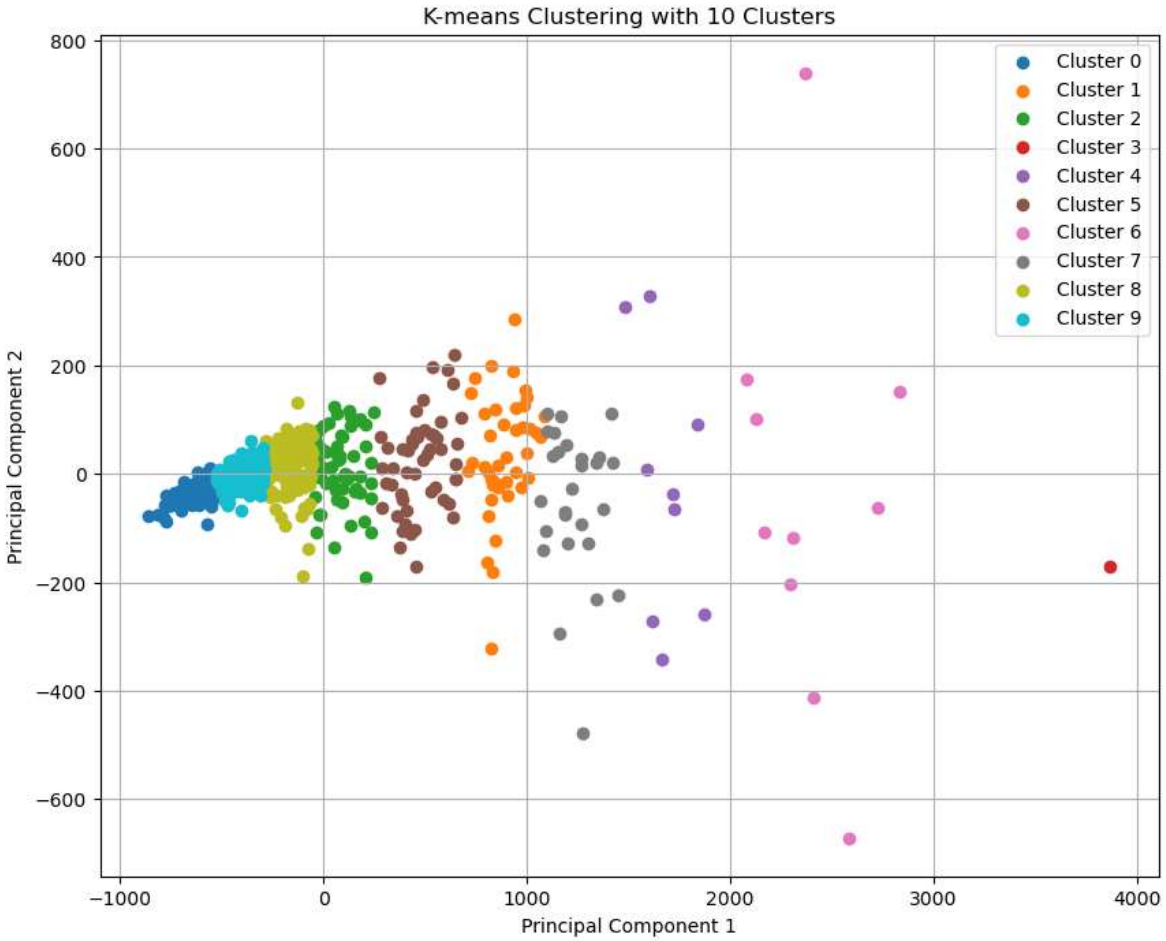
K-Mean CIsutering with 10 Clusters

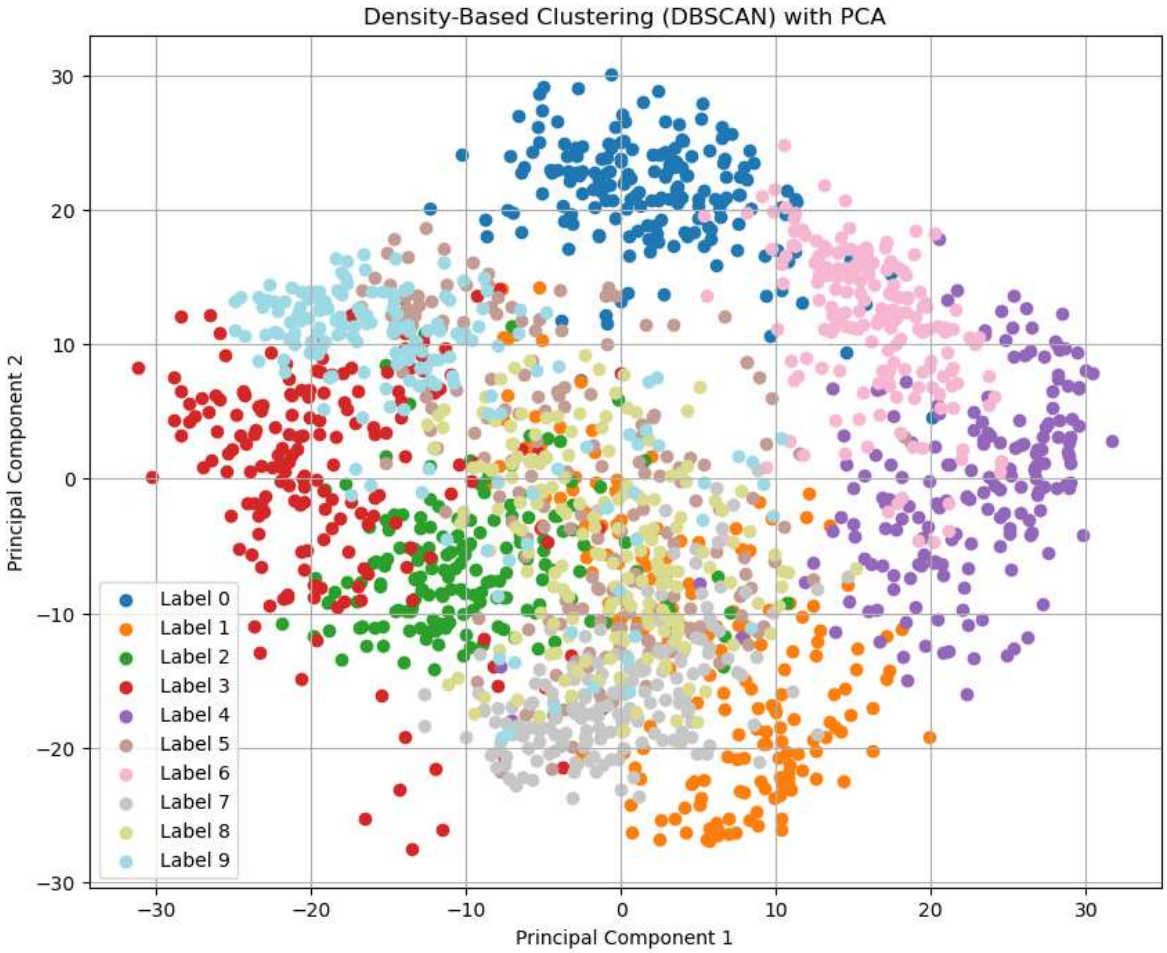
In [ ]:

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.datasets import load_breast_cancer
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

def load_breast_cancer_dataset():
    # Load the Breast Cancer dataset
    data = load_breast_cancer()
    features = data.data
    labels = data.target
    feature_names = data.feature_names
    target_names = data.target_names
    return features, labels, feature_names, target_names

if __name__ == "__main__":
    features, labels, feature_names, target_names = load_breast_cancer_dataset

    # Apply PCA to reduce the number of features to 2 principal components
    pca = PCA(n_components=2)
    principal_components = pca.fit_transform(features)

    # Implement K-means clustering with 10 clusters
    kmeans = KMeans(n_clusters=10, random_state=42)
    clusters = kmeans.fit_predict(principal_components)

    # Plot all observations on a 2D coordinate system with colored clusters
    plt.figure(figsize=(10, 8))

    unique_clusters = np.unique(clusters)
    colors = plt.cm.tab10(np.linspace(0, 1, len(unique_clusters)))

    for i, cluster_label in enumerate(unique_clusters):
        cluster_mask = clusters == cluster_label
        plt.scatter(
            principal_components[cluster_mask, 0],
            principal_components[cluster_mask, 1],
            color=colors[i],
            label=f'Cluster {cluster_label}'
        )

    plt.title("K-means Clustering with 10 Clusters")
    plt.xlabel("Principal Component 1")
    plt.ylabel("Principal Component 2")
    plt.legend()
    plt.grid(True)
    plt.show()
```



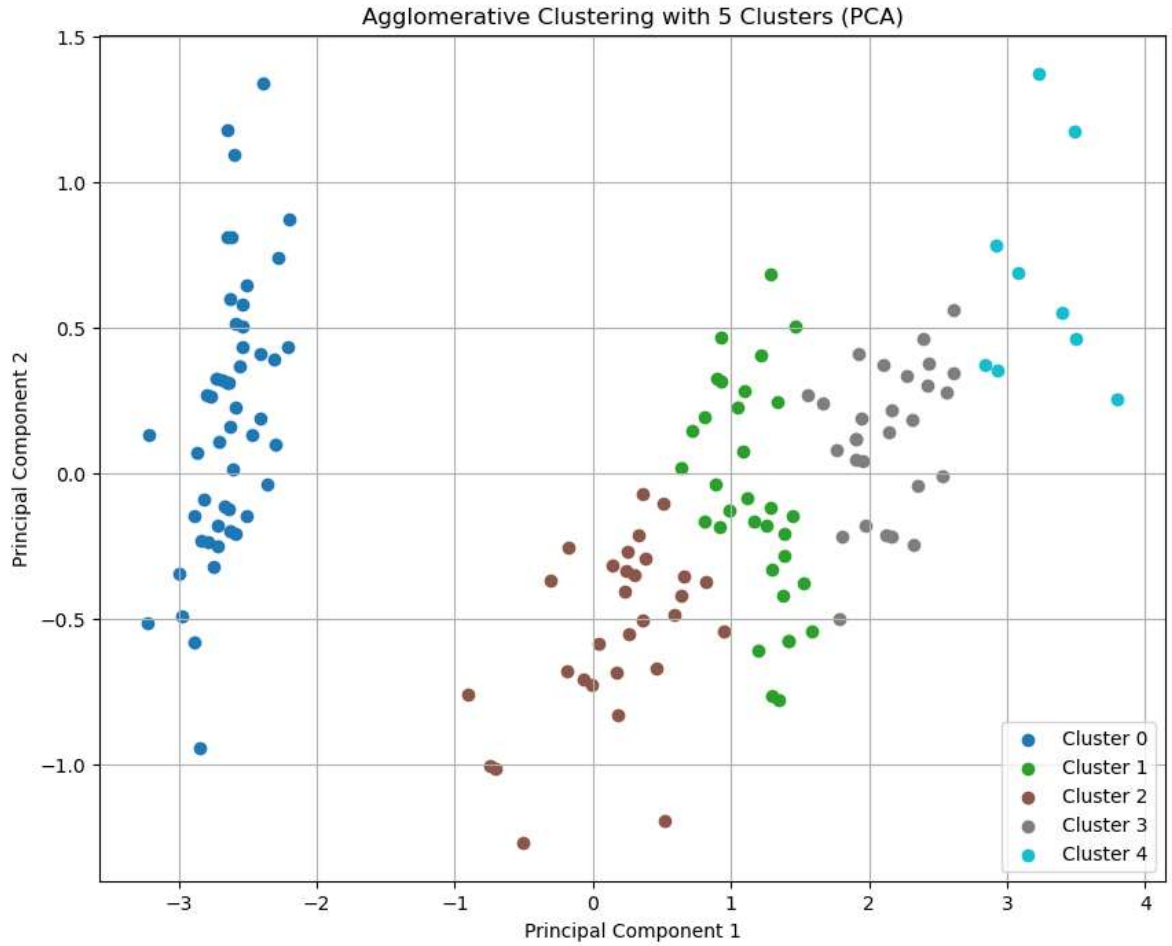
Use density-based clustering (use DBSCAN(min_samples = 10, eps = 1.5) instead of KMeans(n_clusters = 10)). Draw the plot for labels 0, 1, 2, ..., 22

In [2]:
```
import pandas as pd
import numpy as np
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt

def load_digits_dataset():
    # Load the digits dataset
    data = load_digits()
    features = data.data
    labels = data.target
    return features, labels

if __name__ == "__main__":
    features, labels = load_digits_dataset()

    # Apply PCA to reduce the number of features to 2 principal components
    pca = PCA(n_components=2)
    principal_components = pca.fit_transform(features)

    # Implement DBSCAN clustering with min_samples=10 and eps=1.5
    dbscan = DBSCAN(min_samples=10, eps=1.5)
    clusters = dbscan.fit_predict(principal_components)

    # Plot observations for each label separately
    plt.figure(figsize=(10, 8))

    unique_labels = np.unique(labels)
    colors = plt.cm.tab20(np.linspace(0, 1, len(unique_labels)))

    for i, label in enumerate(unique_labels):
        label_mask = labels == label
        plt.scatter(
            principal_components[label_mask, 0],
            principal_components[label_mask, 1],
            color=colors[i],
            label=f'Label {label}'
        )

    plt.title("Density-Based Clustering (DBSCAN) with PCA")
    plt.xlabel("Principal Component 1")
    plt.ylabel("Principal Component 2")
    plt.legend()
    plt.grid(True)
    plt.show()
```



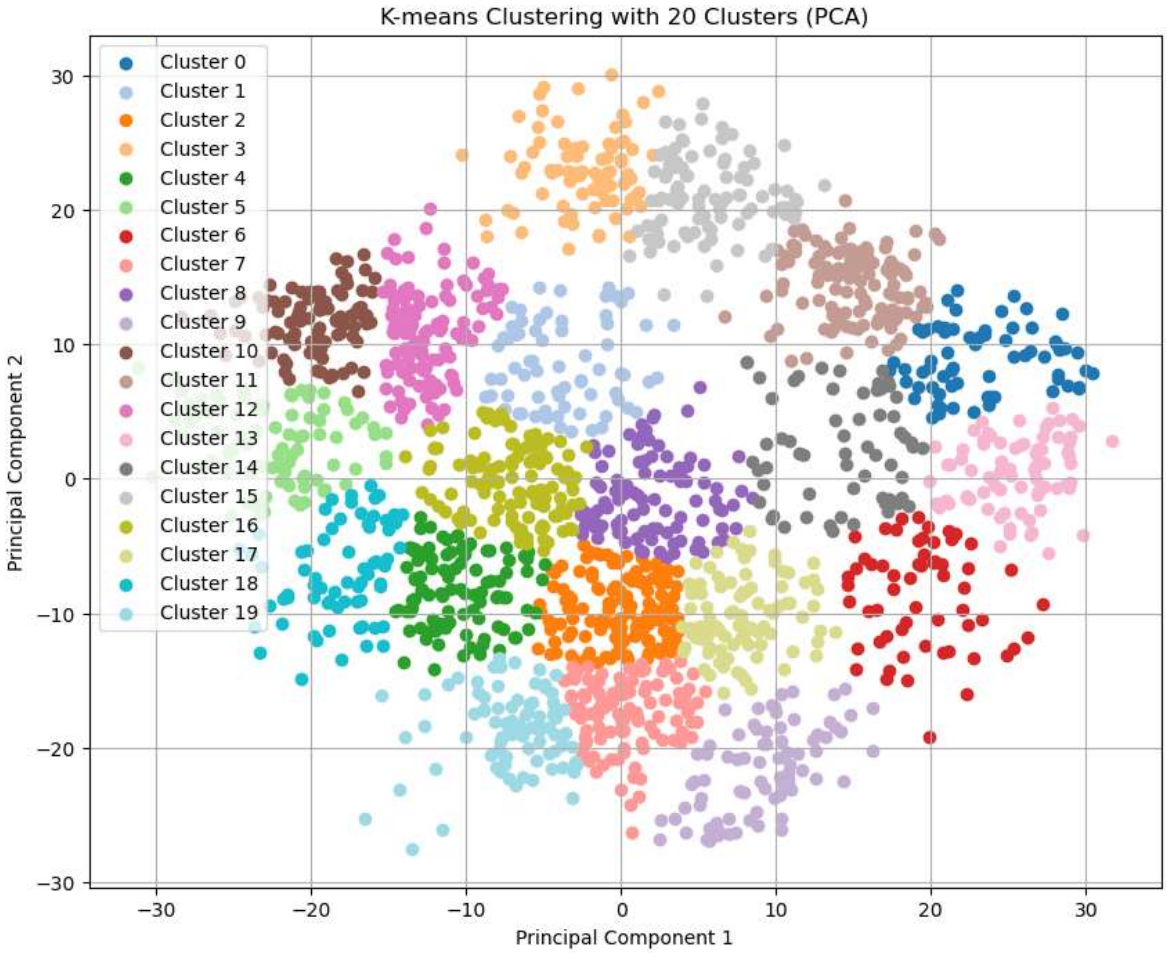
Use hierarchical clustering with 5 clusters (use AgglomerativeClustering(n_clusters = 5) instead of KMeans(n_clusters = 10))

In [3]:
```
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
from sklearn.cluster import AgglomerativeClustering
import matplotlib.pyplot as plt

def load_iris_dataset():
    # Load the Iris dataset
    data = load_iris()
    features = data.data
    labels = data.target
    target_names = data.target_names
    return features, labels, target_names

if __name__ == "__main__":
    features, labels, target_names = load_iris_dataset()

    # Apply PCA to reduce the number of features to 2 principal components
    pca = PCA(n_components=2)
    principal_components = pca.fit_transform(features)

    # Implement Agglomerative Clustering with 5 clusters
    agglomerative = AgglomerativeClustering(n_clusters=5)
    clusters = agglomerative.fit_predict(principal_components)

    # Create a DataFrame with the principal components and cluster labels
    df = pd.DataFrame(data=principal_components, columns=['Principal Componen
    df['cluster'] = clusters

    # Plot all observations on a 2D coordinate system with colored clusters
    plt.figure(figsize=(10, 8))

    unique_clusters = np.unique(clusters)
    colors = plt.cm.tab10(np.linspace(0, 1, len(unique_clusters)))

    for i, cluster_label in enumerate(unique_clusters):
        cluster_mask = clusters == cluster_label
        plt.scatter(
            principal_components[cluster_mask, 0],
            principal_components[cluster_mask, 1],
            color=colors[i],
            label=f'Cluster {cluster_label}'
        )

    plt.title("Agglomerative Clustering with 5 Clusters (PCA)")
    plt.xlabel("Principal Component 1")
    plt.ylabel("Principal Component 2")
    plt.legend()
    plt.grid(True)
    plt.show()
```



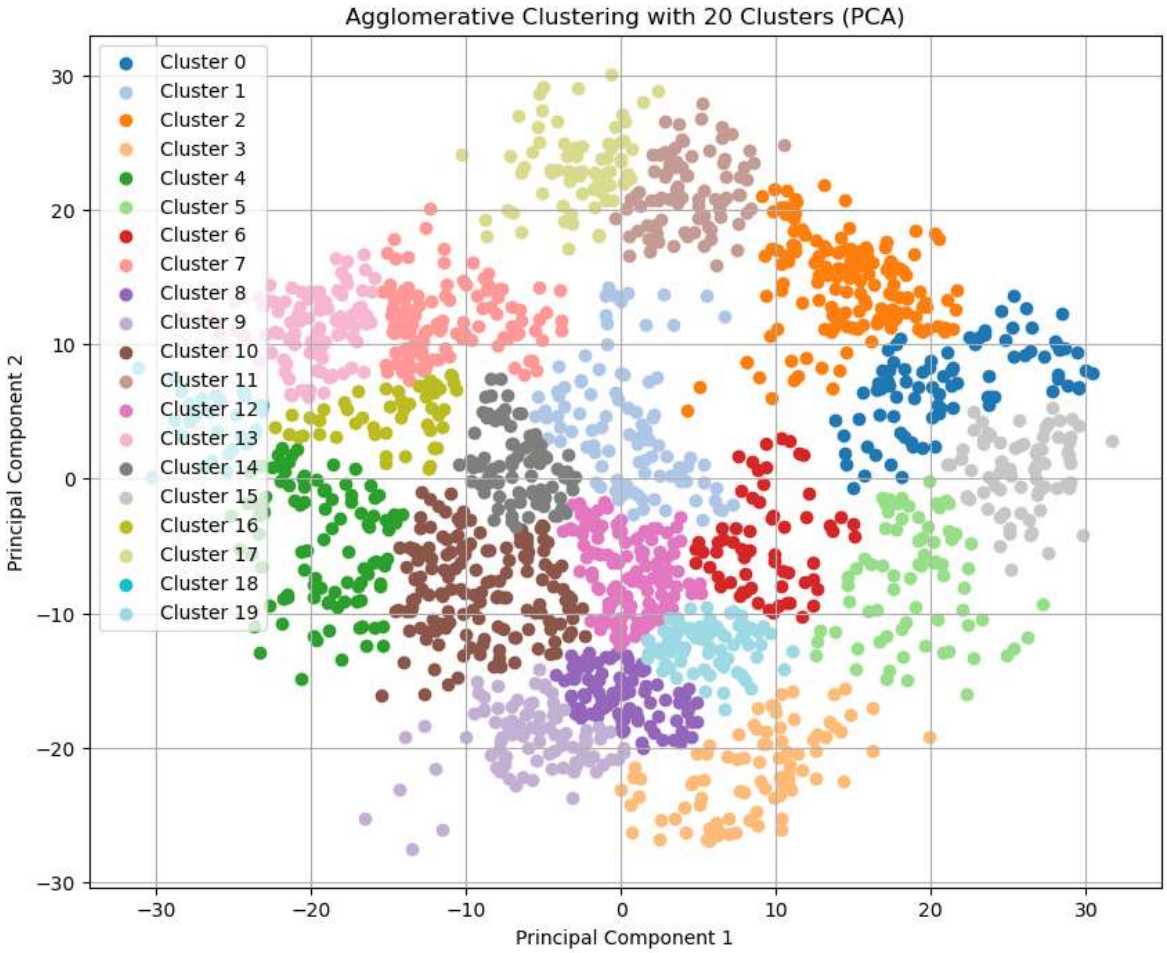
Clustering using K-means with 20 clusters

In [4]:
```
import pandas as pd
import numpy as np
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

def load_digits_dataset():
    # Load the digits dataset
    data = load_digits()
    features = data.data
    labels = data.target
    return features, labels

if __name__ == "__main__":
    features, labels = load_digits_dataset()

    # Apply PCA to reduce the number of features to 2 principal components
    pca = PCA(n_components=2)
    principal_components = pca.fit_transform(features)

    # Implement K-means clustering with 20 clusters
    kmeans = KMeans(n_clusters=20, random_state=42)
    clusters = kmeans.fit_predict(principal_components)

    # Plot observations for each cluster separately
    plt.figure(figsize=(10, 8))

    unique_clusters = np.unique(clusters)
    colors = plt.cm.tab20(np.linspace(0, 1, len(unique_clusters)))

    for i, cluster_label in enumerate(unique_clusters):
        cluster_mask = clusters == cluster_label
        plt.scatter(
            principal_components[cluster_mask, 0],
            principal_components[cluster_mask, 1],
            color=colors[i],
            label=f'Cluster {cluster_label}'
        )

    plt.title("K-means Clustering with 20 Clusters (PCA)")
    plt.xlabel("Principal Component 1")
    plt.ylabel("Principal Component 2")
    plt.legend()
    plt.grid(True)
    plt.show()
```



Hierarchical clustering with 20 clusters (Agglomerative Clustering) for the digits dataset

In [5]:
```
import pandas as pd
import numpy as np
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.cluster import AgglomerativeClustering
import matplotlib.pyplot as plt

def load_digits_dataset():
    # Load the digits dataset
    data = load_digits()
    features = data.data
    labels = data.target
    return features, labels

if __name__ == "__main__":
    features, labels = load_digits_dataset()

    # Apply PCA to reduce the number of features to 2 principal components
    pca = PCA(n_components=2)
    principal_components = pca.fit_transform(features)

    # Implement Agglomerative Clustering with 20 clusters
    agglomerative = AgglomerativeClustering(n_clusters=20)
    clusters = agglomerative.fit_predict(principal_components)

    # Create a DataFrame with the principal components and cluster labels
    df = pd.DataFrame(data=principal_components, columns=['Principal Componen
    df['cluster'] = clusters

    # Plot observations for each cluster separately
    plt.figure(figsize=(10, 8))

    unique_clusters = np.unique(clusters)
    colors = plt.cm.tab20(np.linspace(0, 1, len(unique_clusters)))

    for i, cluster_label in enumerate(unique_clusters):
        cluster_mask = clusters == cluster_label
        plt.scatter(
            principal_components[cluster_mask, 0],
            principal_components[cluster_mask, 1],
            color=colors[i],
            label=f'Cluster {cluster_label}'
        )

    plt.title("Agglomerative Clustering with 20 Clusters (PCA)")
    plt.xlabel("Principal Component 1")
    plt.ylabel("Principal Component 2")
    plt.legend()
    plt.grid(True)
    plt.show()
```


Agglomerative Clustering with 20 Clusters (PCA)
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Cluster 9
Cluster 10
Cluster 11
Cluster 12
Cluster 13
Cluster 14
Cluster 15
Cluster 16
Cluster 17
Cluster 18
Cluster 19
Principal Component 2
Principal Component 1
30
20
10
0
−10
−20
−30
−30
−20
−10
0
10
20
30

In [ ]: