Assignment 02

a. Data Preparation

b. Exploratory data analysis

c. Hyper parameter analysis and selection

a. Data preparation

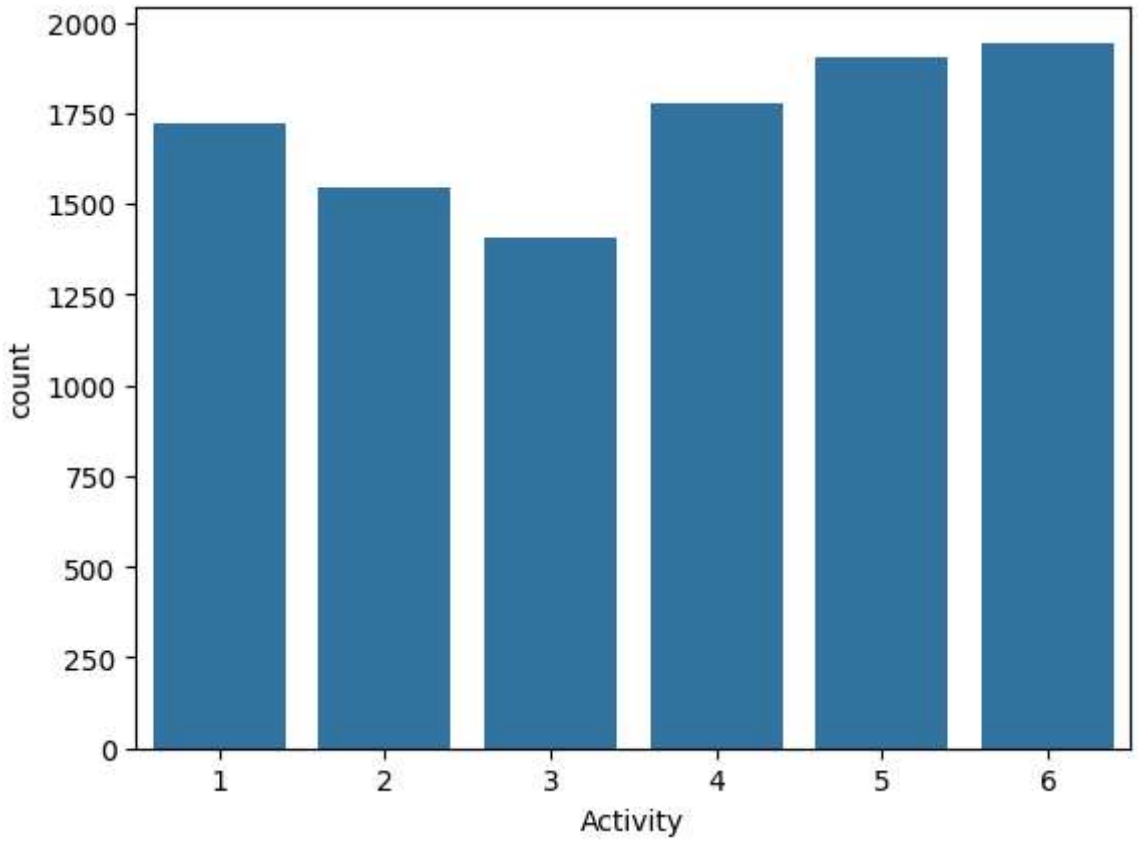
In [1]:
```
import pandas as pd
import zipfile
import requests
from io import BytesIO

# URL of the dataset
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/00240/UCI%20HAR%20Dat

# Download and extract the zip file
response = requests.get(url)
with zipfile.ZipFile(BytesIO(response.content)) as z:
    z.extractall()

# Load the train and test datasets
train_data = pd.read_csv('UCI HAR Dataset/train/X_train.txt', delim_whitespace=True, h
train_labels = pd.read_csv('UCI HAR Dataset/train/y_train.txt', delim_whitespace=True,
test_data = pd.read_csv('UCI HAR Dataset/test/X_test.txt', delim_whitespace=True, head
test_labels = pd.read_csv('UCI HAR Dataset/test/y_test.txt', delim_whitespace=True, he
features = pd.read_csv('UCI HAR Dataset/features.txt', delim_whitespace=True, header=N

# Combine train and test datasets
data = pd.concat([train_data, test_data], ignore_index=True)
labels = pd.concat([train_labels, test_labels], ignore_index=True)

# Name the columns according to the features
data.columns = features[1].values

# Add the labels to the data
data['Activity'] = labels

# Display the first few rows of the dataset
print(data.head())

# Check for missing values
print(data.isnull().sum())

# Normalize the dataset
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)
```

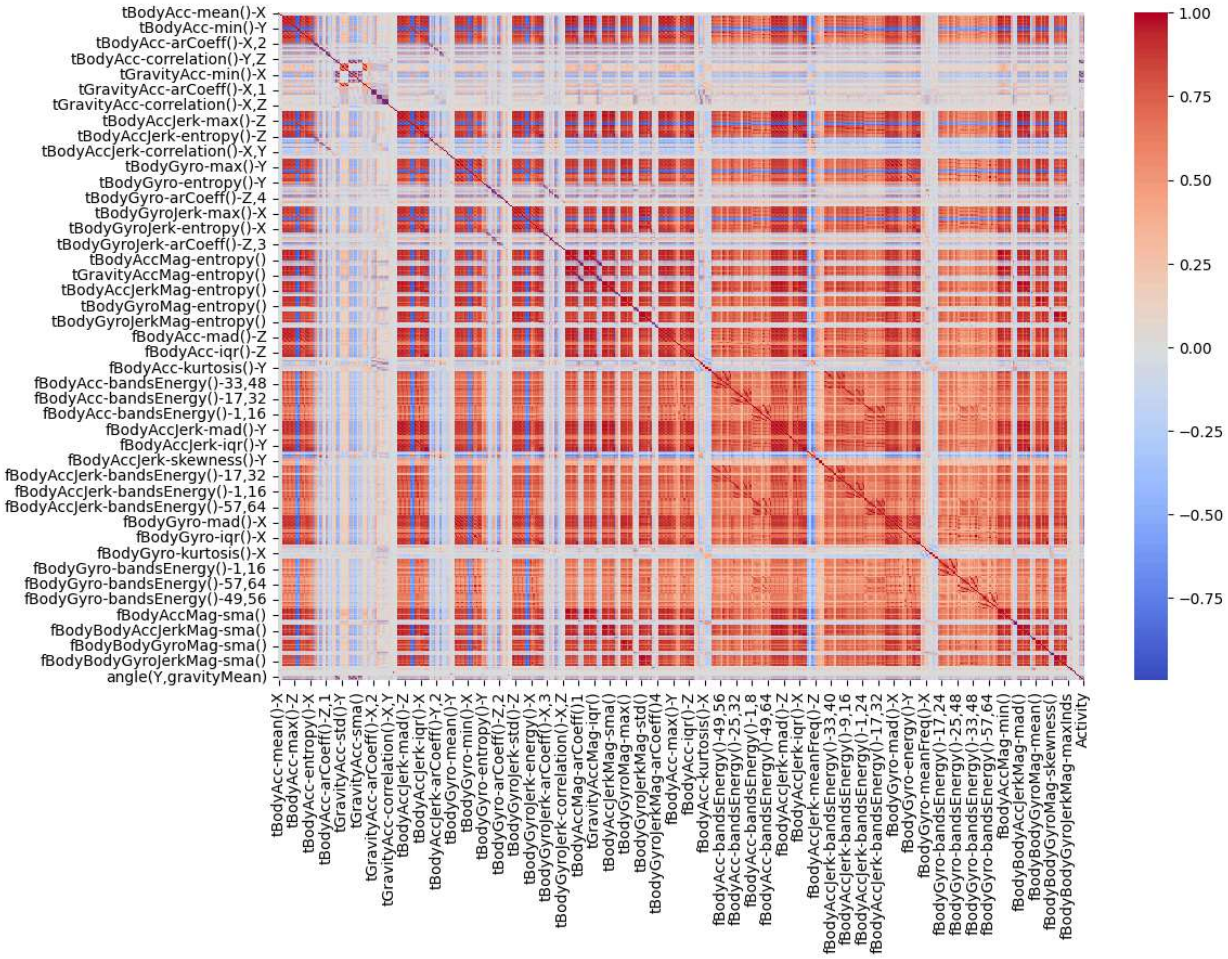
```
   tBodyAcc-mean()-X  tBodyAcc-mean()-Y  tBodyAcc-mean()-Z  tBodyAcc-std()-X  \
0           0.288585          -0.020294          -0.132905         -0.995279
1           0.278419          -0.016411          -0.123520         -0.998245
2           0.279653          -0.019467          -0.113462         -0.995380
3           0.279174          -0.026201          -0.123283         -0.996091
4           0.276629          -0.016570          -0.115362         -0.998139

   tBodyAcc-std()-Y  tBodyAcc-std()-Z  tBodyAcc-mad()-X  tBodyAcc-mad()-Y  \
0         -0.983111         -0.913526         -0.995112         -0.983185
1         -0.975300         -0.960322         -0.998807         -0.974914
2         -0.967187         -0.978944         -0.996520         -0.963668
3         -0.983403         -0.990675         -0.997099         -0.982750
4         -0.980817         -0.990482         -0.998321         -0.979672

   tBodyAcc-mad()-Z  tBodyAcc-max()-X  ...  fBodyBodyGyroJerkMag-skewness()  \
0         -0.923527         -0.934724  ...                         -0.298676
1         -0.957686         -0.943068  ...                         -0.595051
2         -0.977469         -0.938692  ...                         -0.390748
3         -0.989302         -0.938692  ...                         -0.117290
4         -0.990441         -0.942469  ...                         -0.351471

   fBodyBodyGyroJerkMag-kurtosis()  angle(tBodyAccMean,gravity)  \
0                        -0.710304                   -0.112754
1                        -0.861499                    0.053477
2                        -0.760104                   -0.118559
3                        -0.482845                   -0.036788
4                        -0.699205                    0.123320

   angle(tBodyAccJerkMean),gravityMean)  angle(tBodyGyroMean,gravityMean)  \
0                              0.030400                         -0.464761
1                             -0.007435                         -0.732626
2                              0.177899                          0.100699
3                             -0.012892                          0.640011
4                              0.122542                          0.693578

   angle(tBodyGyroJerkMean,gravityMean)  angle(X,gravityMean)  \
0                             -0.018446             -0.841247
1                              0.703511             -0.844788
2                              0.808529             -0.848933
3                             -0.485366             -0.848649
4                             -0.615971             -0.847865

   angle(Y,gravityMean)  angle(Z,gravityMean)  Activity
0              0.179941             -0.058627         5
1              0.180289             -0.054317         5
2              0.180637             -0.049118         5
3              0.181935             -0.047663         5
4              0.185151             -0.043892         5

[5 rows x 562 columns]
tBodyAcc-mean()-X                       0
tBodyAcc-mean()-Y                       0
tBodyAcc-mean()-Z                       0
tBodyAcc-std()-X                        0
tBodyAcc-std()-Y                        0
                                       ..
angle(tBodyGyroJerkMean,gravityMean)    0
angle(X,gravityMean)                    0
angle(Y,gravityMean)                    0
angle(Z,gravityMean)                    0
```

```
Activity                                  0
Length: 562, dtype: int64
```

b. Exploratary data analysis

In [2]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Visualize the distribution of activities
sns.countplot(x='Activity', data=data)
plt.show()

# Visualize the correlation matrix
corr_matrix = data.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, cmap='coolwarm')
plt.show()
```



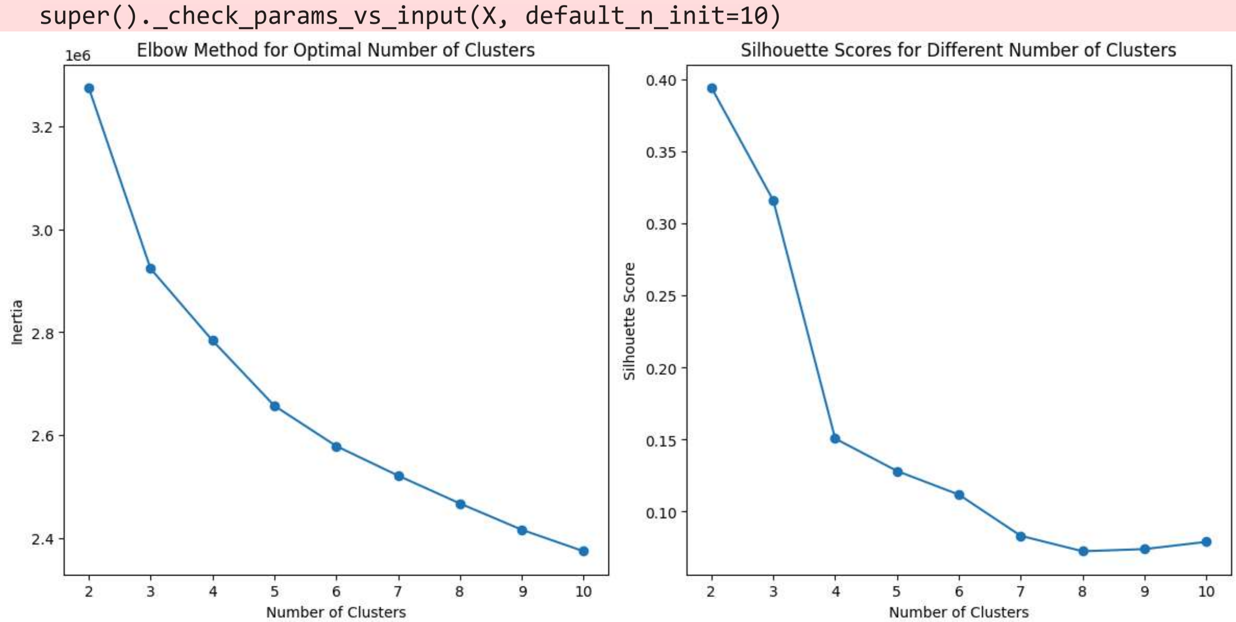


c. Hyper parameter analysis and selecction

In [3]:
```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# Apply K-Means clustering with a range of cluster numbers to find the optimal one
inertia = []
silhouette_scores = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans_labels = kmeans.fit_predict(scaled_data)
    inertia.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(scaled_data, kmeans_labels))

# Plot the inertia to visualize the elbow point
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(range(2, 11), inertia, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal Number of Clusters')

# Plot the Silhouette Scores
plt.subplot(1, 2, 2)
plt.plot(range(2, 11), silhouette_scores, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Scores for Different Number of Clusters')
```

```
plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



Assignmnet 03

a. Clustering

b. Visualization

c. Outer or Outlier Detection and discussion

d. Improvisation of the overall experiment

a. Clustering

Here I have chosen three main clustering methods.

1. K-Means clustering method - chosen for its simplicity and efficiency when the number of clusters is known or can be estimated
2. Hierarchial Clustering - selected for its ability to explore different levels of cluster granularity and its interpretability through dendrograms.
3. DBSCAN - used for its strength in identifying clusters of arbitrary shapes and handling noise, especially when the number of clusters is not known.

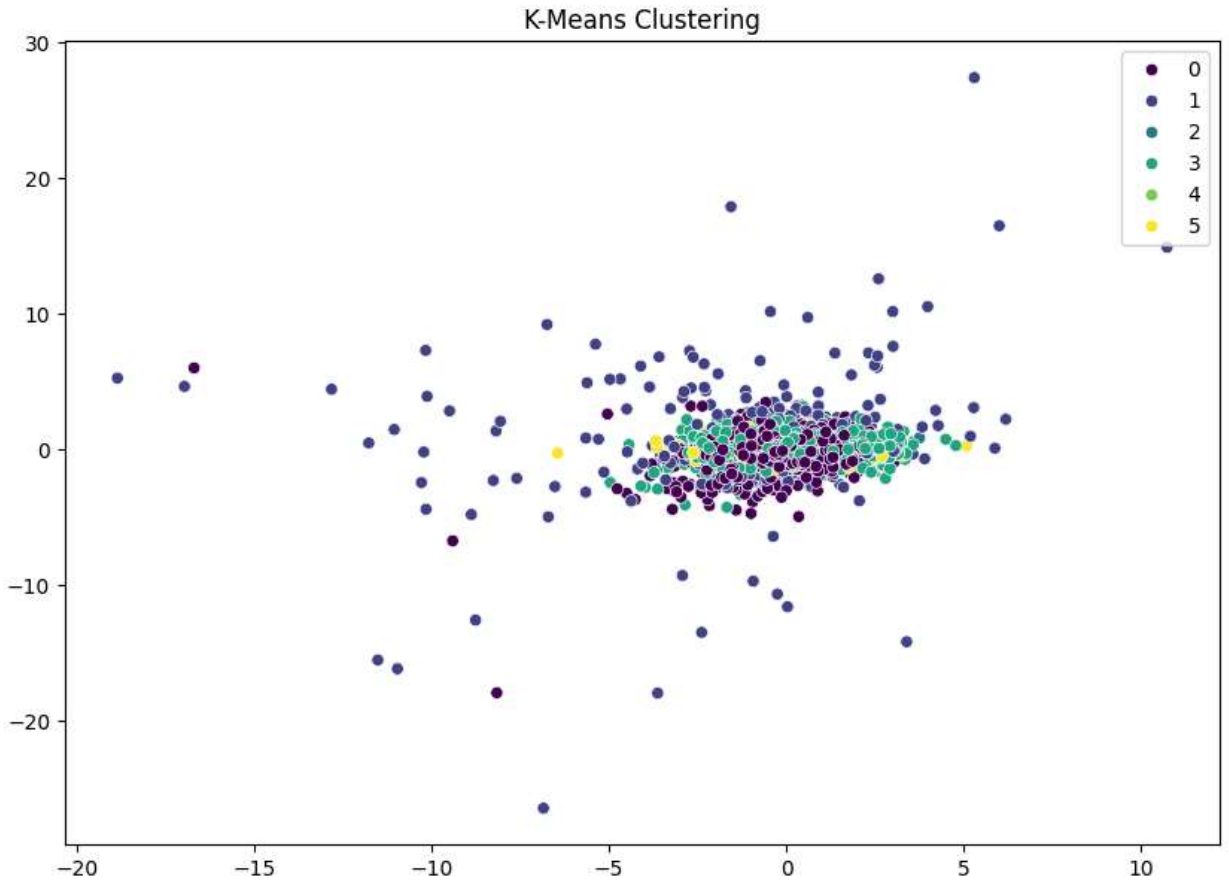
In [4]:
```
# K-Means Clustering
kmeans = KMeans(n_clusters=6, random_state=42)
kmeans_labels = kmeans.fit_predict(scaled_data)

# Hierarchical Clustering
from scipy.cluster.hierarchy import linkage, fcluster

linked = linkage(scaled_data, 'ward')
max_d = 50  # This distance can be adjusted based on your requirements
hierarchical_labels = fcluster(linked, max_d, criterion='distance')

# DBSCAN Clustering
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=10, min_samples=5)
dbscan_labels = dbscan.fit_predict(scaled_data)
```

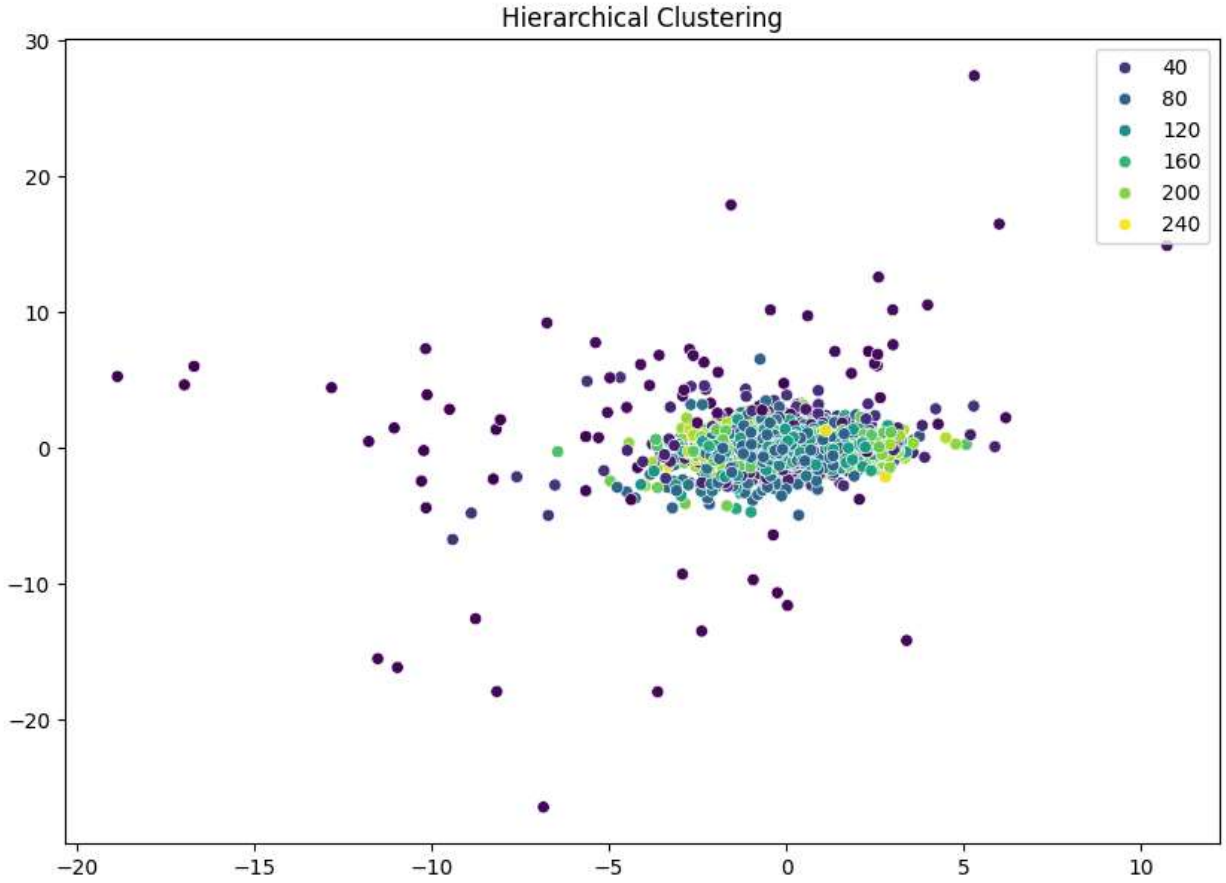
```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

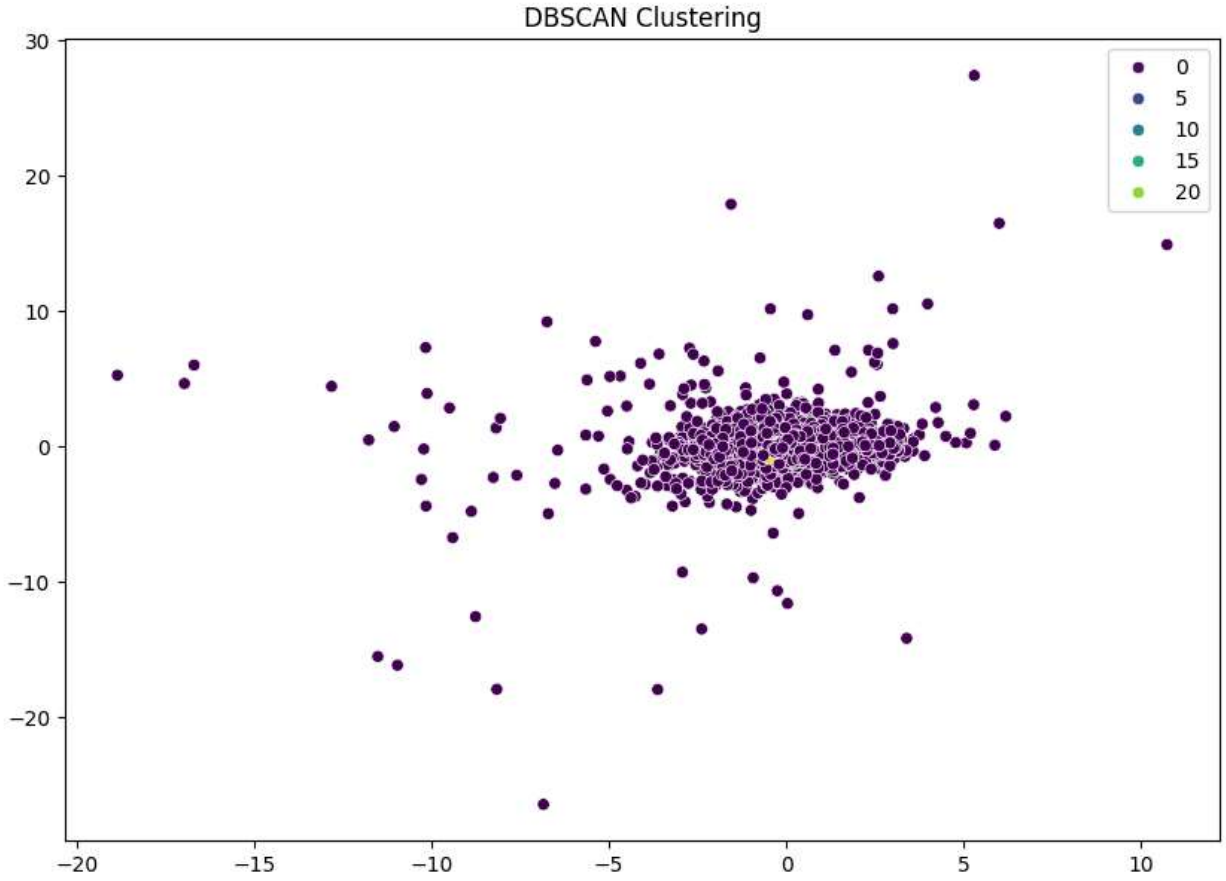
b. visualization

In [5]:
```
# Visualize K-Means Clustering
plt.figure(figsize=(10, 7))
sns.scatterplot(x=scaled_data[:, 0], y=scaled_data[:, 1], hue=kmeans_labels, palette='
plt.title('K-Means Clustering')
plt.show()

# Visualize Hierarchical Clustering
plt.figure(figsize=(10, 7))
sns.scatterplot(x=scaled_data[:, 0], y=scaled_data[:, 1], hue=hierarchical_labels, pal
plt.title('Hierarchical Clustering')
```

```
plt.show()

# Visualize DBSCAN Clustering
plt.figure(figsize=(10, 7))
sns.scatterplot(x=scaled_data[:, 0], y=scaled_data[:, 1], hue=dbscan_labels, palette='
plt.title('DBSCAN Clustering')
plt.show()
```







c. Outlier Detection and Discussion

In [6]:
```
import numpy as np
from sklearn.metrics import pairwise_distances_argmin_min
from sklearn.cluster import DBSCAN

# DBSCAN Clustering
dbscan = DBSCAN(eps=10, min_samples=5)
dbscan_labels = dbscan.fit_predict(scaled_data)
data['DBSCAN_Labels'] = dbscan_labels

# K-Means Outliers
cluster_centers = kmeans.cluster_centers_
closest, distances = pairwise_distances_argmin_min(scaled_data, cluster_centers)
threshold = np.percentile(distances, 95)
kmeans_outliers = data[distances > threshold]
print("K-Means Outliers:")
print(kmeans_outliers)

# DBSCAN Outliers
dbscan_outliers = data[data['DBSCAN_Labels'] == -1]
print("DBSCAN Outliers:")
print(dbscan_outliers)

# Hierarchical Outliers
from scipy.spatial.distance import cdist

hierarchical_centers = []
for label in np.unique(hierarchical_labels):
    cluster_points = scaled_data[hierarchical_labels == label]
    cluster_center = cluster_points.mean(axis=0)
    hierarchical_centers.append(cluster_center)
hierarchical_centers = np.array(hierarchical_centers)
hierarchical_distances = cdist(scaled_data, hierarchical_centers, 'euclidean').min(axi
threshold = np.percentile(hierarchical_distances, 95)
hierarchical_outliers = data[hierarchical_distances > threshold]
print("Hierarchical Clustering Outliers:")
print(hierarchical_outliers)
```

K-Means Outliers:

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	\
27	0.144504	0.189263	0.062769	
66	-0.166426	-0.119353	0.133035	
67	-0.239103	-0.096904	0.148035	
69	0.303833	0.103648	-0.245331	
70	-0.361205	-0.268121	0.176896	
...	...	...	...	
9888	0.248988	-0.054485	-0.094429	
9893	0.029089	-0.126367	-0.136597	
9894	0.242932	-0.017346	-0.160880	
9913	0.436585	0.016542	-0.107352	
10055	0.293911	0.027124	-0.107460	

	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	\
27	-0.904300	-0.181937	-0.443151	-0.901100	
66	-0.633135	-0.189775	-0.319336	-0.623804	
67	-0.637325	-0.127692	-0.258924	-0.643642	
69	-0.755994	-0.642571	-0.695501	-0.767537	
70	-0.599392	0.532506	0.004324	-0.664108	
...	...	...	...	...	
9888	-0.069988	0.594506	-0.298640	-0.179355	
9893	-0.055291	0.361742	-0.131526	-0.153722	
9894	-0.086651	0.348458	-0.224892	-0.166818	
9913	0.031037	0.414080	-0.165218	-0.006197	
10055	0.026827	-0.225615	-0.126099	-0.027171	

	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	\
27	-0.110813	-0.400599	-0.931896	...	
66	-0.199675	-0.281088	-0.925208	...	
67	-0.138551	-0.239634	-0.878149	...	
69	-0.709978	-0.730464	-0.717081	...	
70	0.502260	0.048472	-0.717081	...	
...	...	...	...	...	
9888	0.426224	-0.364143	0.231756	...	
9893	0.456402	-0.202129	0.308609	...	
9894	0.421767	-0.235293	0.216959	...	
9913	0.498857	-0.188523	0.185132	...	
10055	-0.267850	-0.147456	0.434766	...	

	fBodyBodyGyroJerkMag-kurtosis()	angle(tBodyAccMean,gravity)	\
27	-0.241781	0.013526	
66	-0.128895	0.139475	
67	-0.561918	0.002418	
69	-0.609644	-0.116769	
70	-0.330520	0.152830	
...	...	...	
9888	-0.274321	-0.000801	
9893	-0.877139	0.217975	
9894	-0.943052	0.229744	
9913	-0.757693	-0.347441	
10055	-0.682952	-0.030807	

	angle(tBodyAccJerkMean),gravityMean)	angle(tBodyGyroMean,gravityMean)	\
27	0.043354	0.021485	
66	0.039216	-0.495303	
67	-0.002366	-0.101878	
69	-0.391404	0.413596	
70	0.149602	0.127245	
...	...	...	

```
9888                              -0.761262                          0.923759
9893                               0.800783                         -0.953714
9894                              -0.825444                         -0.961149
9913                              -0.489860                         -0.898488
10055                              0.938794                         -0.939260

       angle(tBodyGyroJerkMean,gravityMean)  angle(X,gravityMean)  \
27                                 0.046689            -0.667085
66                                 0.082696             0.272083
67                                 0.543207             0.378792
69                                 0.051020             0.494535
70                                 0.192852             0.508733
...                                     ...                  ...
9888                              -0.488060            -0.701915
9893                              -0.109055            -0.599037
9894                               0.006384            -0.545412
9913                               0.599798            -0.467906
10055                              0.705437            -0.835261

       angle(Y,gravityMean)  angle(Z,gravityMean)  Activity  DBSCAN_Labels
27                 0.054216             -0.218875         4             -1
66                -0.534122             -0.397428         6             -1
67                -0.488443             -0.487057         6             -1
69                -0.465940             -0.535802         6             -1
70                -0.496132             -0.506197         6             -1
...                     ...                   ...       ...            ...
9888               0.295335             -0.042546         3             -1
9893               0.375074              0.013516         2             -1
9894               0.412123              0.016589         2             -1
9913               0.463660             -0.018171         2             -1
10055              0.154491              0.121594         3             -1

[515 rows x 563 columns]
DBSCAN Outliers:
       tBodyAcc-mean()-X  tBodyAcc-mean()-Y  tBodyAcc-mean()-Z  \
0               0.288585          -0.020294          -0.132905
14              0.297946           0.027094          -0.061668
15              0.279203          -0.023020          -0.122080
27              0.144504           0.189263           0.062769
28              0.287252          -0.037455          -0.145974
...                  ...                ...                ...
10294           0.310155          -0.053391          -0.099109
10295           0.363385          -0.039214          -0.105915
10296           0.349966           0.030077          -0.115788
10297           0.237594           0.018467          -0.096499
10298           0.153627          -0.018437          -0.137018

       tBodyAcc-std()-X  tBodyAcc-std()-Y  tBodyAcc-std()-Z  tBodyAcc-mad()-X  \
0             -0.995279         -0.983111         -0.913526         -0.995112
14            -0.988641         -0.816699         -0.901907         -0.988958
15            -0.996839         -0.974848         -0.983386         -0.997094
27            -0.904300         -0.181937         -0.443151         -0.901100
28            -0.982915         -0.891605         -0.941438         -0.984418
...                 ...               ...               ...               ...
10294         -0.287866         -0.140589         -0.215088         -0.356083
10295         -0.305388          0.028148         -0.196373         -0.373540
10296         -0.329638         -0.042143         -0.250181         -0.388017
10297         -0.323114         -0.229775         -0.207574         -0.392380
10298         -0.330046         -0.195253         -0.164339         -0.430974
```

	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	\
0	-0.983185	-0.923527	-0.934724	...	
14	-0.794280	-0.888015	-0.925977	...	
15	-0.973332	-0.984065	-0.941716	...	
27	-0.110813	-0.400599	-0.931896	...	
28	-0.891373	-0.933361	-0.931896	...	
...	...	...	...	...	
10294	-0.148775	-0.232057	0.185361	...	
10295	-0.030036	-0.270237	0.185361	...	
10296	-0.133257	-0.347029	0.007471	...	
10297	-0.279610	-0.289477	0.007471	...	
10298	-0.218295	-0.229933	-0.111527	...	

	fBodyBodyGyroJerkMag-kurtosis()	angle(tBodyAccMean,gravity)	\
0	-0.710304	-0.112754	
14	-0.068054	0.062297	
15	-0.953520	-0.121852	
27	-0.241781	0.013526	
28	-0.899523	0.194735	
...	...	...	
10294	-0.750809	-0.337422	
10295	-0.700274	-0.736701	
10296	-0.467179	-0.181560	
10297	-0.617737	0.444558	
10298	-0.436940	0.598808	

	angle(tBodyAccJerkMean),gravityMean)	angle(tBodyGyroMean,gravityMean)	\
0	0.030400	-0.464761	
14	-0.058719	0.031208	
15	-0.029077	-0.013034	
27	0.043354	0.021485	
28	-0.148056	0.033529	
...	...	...	
10294	0.346295	0.884904	
10295	-0.372889	-0.657421	
10296	0.088574	0.696663	
10297	-0.819188	0.929294	
10298	-0.287951	0.876030	

	angle(tBodyGyroJerkMean,gravityMean)	angle(X,gravityMean)	\
0	-0.018446	-0.841247	
14	-0.268791	-0.730937	
15	-0.056927	-0.761101	
27	0.046689	-0.667085	
28	-0.127028	-0.564807	
...	...	...	
10294	-0.698885	-0.651732	
10295	0.322549	-0.655181	
10296	0.363139	-0.655357	
10297	-0.008398	-0.659719	
10298	-0.024965	-0.660080	

	angle(Y,gravityMean)	angle(Z,gravityMean)	Activity	DBSCAN_Labels
0	0.179941	-0.058627	5	-1
14	0.283159	0.036444	5	-1
15	0.263119	0.024172	5	-1
27	0.054216	-0.218875	4	-1
28	-0.027045	-0.266055	4	-1
...	...	...	...	...
10294	0.274627	0.184784	2	-1

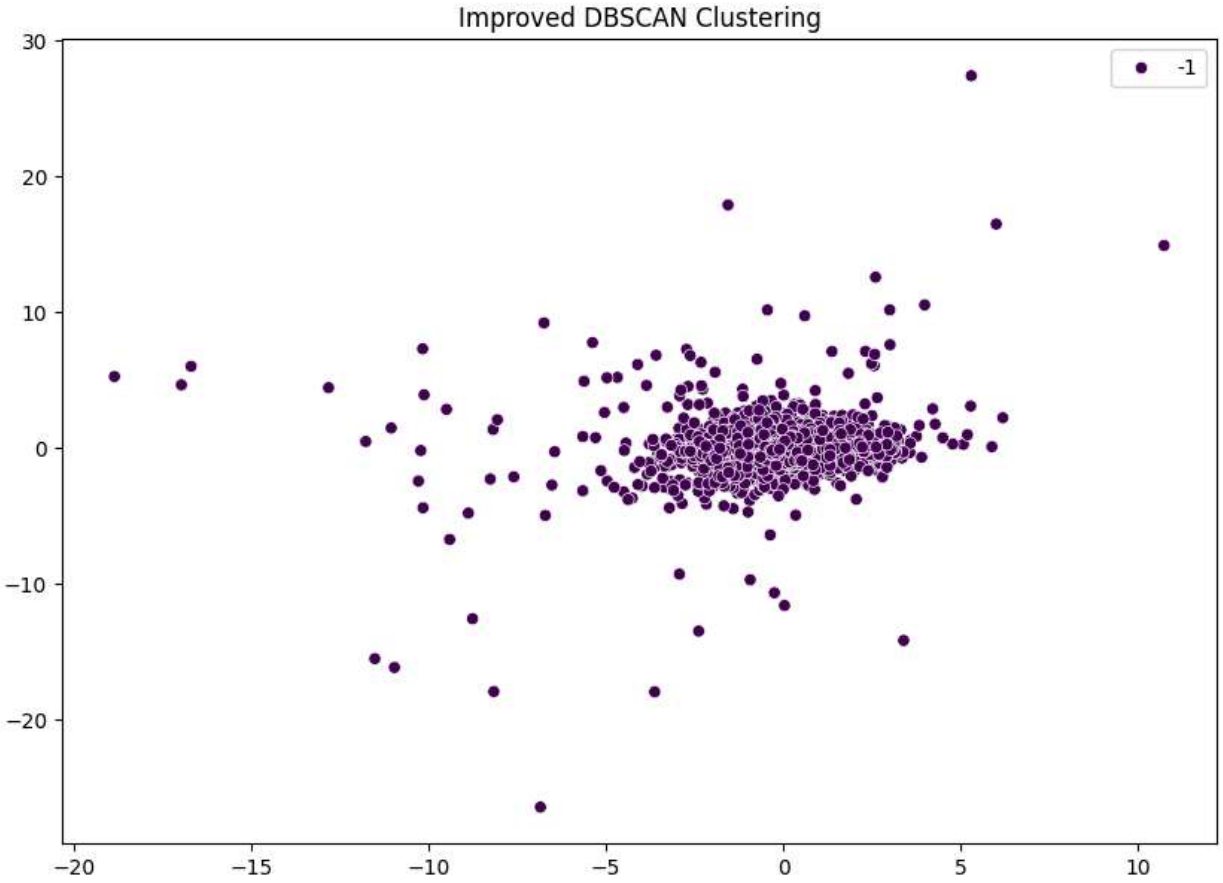
```
10295              0.273578              0.182412         2             -1
10296              0.274479              0.181184         2             -1
10297              0.264782              0.187563         2             -1
10298              0.263936              0.188103         2             -1

[7550 rows x 563 columns]
Hierarchical Clustering Outliers:
       tBodyAcc-mean()-X  tBodyAcc-mean()-Y  tBodyAcc-mean()-Z  \
69              0.303833           0.103648          -0.245331
70             -0.361205          -0.268121           0.176896
71             -0.277066          -0.684097           0.346658
101             0.261081          -0.015636          -0.102588
134             0.285213          -0.020202          -0.011024
...                  ...                ...                ...
9913            0.436585           0.016542          -0.107352
10055           0.293911           0.027124          -0.107460
10061           0.364164           0.012940          -0.154306
10062           0.183601          -0.003479          -0.108573
10069           0.334550          -0.018273          -0.113456

       tBodyAcc-std()-X  tBodyAcc-std()-Y  tBodyAcc-std()-Z  tBodyAcc-mad()-X  \
69            -0.755994         -0.642571         -0.695501         -0.767537
70            -0.599392          0.532506          0.004324         -0.664108
71            -0.596410          0.024683         -0.160404         -0.631819
101           -0.171239          0.201236         -0.261589         -0.217560
134            0.122989          0.081946          0.049997          0.050769
...                 ...               ...               ...               ...
9913           0.031037          0.414080         -0.165218         -0.006197
10055          0.026827         -0.225615         -0.126099         -0.027171
10061         -0.016621          0.067345          0.156549         -0.081920
10062         -0.159185          0.015795          0.163604         -0.289626
10069         -0.291864          0.017717         -0.277876         -0.355955

       tBodyAcc-mad()-Y  tBodyAcc-mad()-Z  tBodyAcc-max()-X  ...  \
69            -0.709978         -0.730464         -0.717081  ...
70             0.502260          0.048472         -0.717081  ...
71            -0.068857         -0.235160         -0.866296  ...
101            0.131643         -0.287320          0.022737  ...
134            0.012863         -0.014688          0.544512  ...
...                 ...               ...               ...  ...
9913           0.498857         -0.188523          0.185132  ...
10055         -0.267850         -0.147456          0.434766  ...
10061         -0.069539          0.055654          0.593914  ...
10062         -0.095105          0.081055          0.593914  ...
10069         -0.043549         -0.354995          0.077831  ...

       fBodyBodyGyroJerkMag-kurtosis()  angle(tBodyAccMean,gravity)  \
69                           -0.609644                    -0.116769
70                           -0.330520                     0.152830
71                           -0.437059                     0.123108
101                           0.034323                     0.717645
134                          -0.732613                     0.003646
...                                ...                          ...
9913                         -0.757693                    -0.347441
10055                        -0.682952                    -0.030807
10061                         0.024203                    -0.458646
10062                        -0.008584                     0.799963
10069                        -0.711214                    -1.000000

       angle(tBodyAccJerkMean),gravityMean)  angle(tBodyGyroMean,gravityMean)  \
```

```
69                                -0.391404                          0.413596
70                                 0.149602                          0.127245
71                                 0.432540                         -0.083446
101                                0.401581                         -0.038345
134                               -0.705994                          0.977199
...                                     ...                               ...
9913                              -0.489860                         -0.898488
10055                              0.938794                         -0.939260
10061                              0.946931                         -0.784799
10062                             -0.867253                          0.924071
10069                              0.577611                          0.943144

       angle(tBodyGyroJerkMean,gravityMean)  angle(X,gravityMean)  \
69                                 0.051020              0.494535
70                                 0.192852              0.508733
71                                 0.025648              0.668284
101                               -0.593632             -0.749963
134                               -0.174691             -0.746162
...                                     ...                   ...
9913                               0.599798             -0.467906
10055                              0.705437             -0.835261
10061                              0.665273             -0.797666
10062                             -0.679476             -0.797192
10069                              0.146741             -0.848083

       angle(Y,gravityMean)  angle(Z,gravityMean)  Activity  DBSCAN_Labels
69                -0.465940             -0.535802         6             -1
70                -0.496132             -0.506197         6             -1
71                -0.336641             -0.672685         6             -1
101                0.269151              0.043100         1             -1
134                0.269038              0.058288         3             -1
...                     ...                   ...       ...            ...
9913               0.463660             -0.018171         2             -1
10055              0.154491              0.121594         3             -1
10061              0.161195              0.148388         3             -1
10062              0.150833              0.153868         3             -1
10069              0.130046              0.123182         3             -1

[515 rows x 563 columns]
```

d. Improvisation of the Overall Experiment

In [7]:

```
# Example: Adjust DBSCAN parameters
dbscan = DBSCAN(eps=5, min_samples=10)
dbscan_labels = dbscan.fit_predict(scaled_data)

# Visualize the new DBSCAN clustering results
plt.figure(figsize=(10, 7))
sns.scatterplot(x=scaled_data[:, 0], y=scaled_data[:, 1], hue=dbscan_labels, palette='
plt.title('Improved DBSCAN Clustering')
plt.show()
```



Assignment 04

a. Evaluation of the algorithm's performance

b. Improvisation of the algorithm's performance

a. Evaluation of the algorithm's performance

In [8]:
```
from sklearn.metrics import silhouette_score, calinski_harabasz_score, davies_bouldin_

# Silhouette Scores
kmeans_silhouette = silhouette_score(scaled_data, kmeans_labels)
hierarchical_silhouette = silhouette_score(scaled_data, hierarchical_labels)
print(f"K-Means Silhouette Score: {kmeans_silhouette}")
print(f"Hierarchical Clustering Silhouette Score: {hierarchical_silhouette}")

# Calinski-Harabasz Index
kmeans_ch = calinski_harabasz_score(scaled_data, kmeans_labels)
hierarchical_ch = calinski_harabasz_score(scaled_data, hierarchical_labels)
print(f"K-Means Calinski-Harabasz Index: {kmeans_ch}")
print(f"Hierarchical Clustering Calinski-Harabasz Index: {hierarchical_ch}")


# Davies-Bouldin Index (Lower is better)
kmeans_db = davies_bouldin_score(scaled_data, kmeans_labels)
hierarchical_db = davies_bouldin_score(scaled_data, hierarchical_labels)


print(f"K-Means Davies-Bouldin Index: {kmeans_db}")
print(f"Hierarchical Clustering Davies-Bouldin Index: {hierarchical_db}")
```

```
# Perform DBSCAN clustering
dbscan = DBSCAN(eps=10, min_samples=5)
dbscan_labels = dbscan.fit_predict(scaled_data)

# Check the number of unique clusters
n_clusters_dbscan = len(set(dbscan_labels)) - (1 if -1 in dbscan_labels else 0)  # Exc

if n_clusters_dbscan > 1:
    # Calculate evaluation metrics if valid clusters are found
    dbscan_silhouette = silhouette_score(scaled_data, dbscan_labels)
    dbscan_ch = calinski_harabasz_score(scaled_data, dbscan_labels)
    dbscan_db = davies_bouldin_score(scaled_data, dbscan_labels)

    print(f"DBSCAN Silhouette Score: {dbscan_silhouette}")
    print(f"DBSCAN Calinski-Harabasz Index: {dbscan_ch}")
    print(f"DBSCAN Davies-Bouldin Index: {dbscan_db}")
else:
    print("DBSCAN did not produce valid clusters for evaluation metrics calculation.")
```

```
K-Means Silhouette Score: 0.11170815665278065
Hierarchical Clustering Silhouette Score: 0.04056762666661061
K-Means Calinski-Harabasz Index: 2562.0763318007994
Hierarchical Clustering Calinski-Harabasz Index: 123.68508594826704
K-Means Davies-Bouldin Index: 2.361993399028419
Hierarchical Clustering Davies-Bouldin Index: 2.500257236694774
DBSCAN Silhouette Score: -0.28952693325433454
DBSCAN Calinski-Harabasz Index: 79.73574446917516
DBSCAN Davies-Bouldin Index: 2.0498396135520225
```

b. Improvisation of the algorithm's performance

In [9]:

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

best_k = 0
best_score = -1
for k in range(2, 5):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans_labels = kmeans.fit_predict(scaled_data)
    score = silhouette_score(scaled_data, kmeans_labels)
    print(f"Silhouette Score for {k} clusters: {score}")
    if score > best_score:
        best_k = k
        best_score = score

print(f"Best number of clusters for K-Means: {best_k} with Silhouette Score: {best_sco
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
Silhouette Score for 2 clusters: 0.3942991642725773
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
Silhouette Score for 3 clusters: 0.3159136575554689
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value
of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
Silhouette Score for 4 clusters: 0.15064582218461098
Best number of clusters for K-Means: 2 with Silhouette Score: 0.3942991642725773
```

In [10]:

```python
#Adjusting DBSCAN Parameters
from sklearn.cluster import DBSCAN

best_eps = 0
best_dbscan_score = -1
for eps in np.arange(0.5, 2.0, 0.1):
    dbscan = DBSCAN(eps=eps, min_samples=5)
    dbscan_labels = dbscan.fit_predict(scaled_data)
    if len(set(dbscan_labels)) > 1:
        score = silhouette_score(scaled_data, dbscan_labels)
        print(f"Silhouette Score for eps={eps}: {score}")
        if score > best_dbscan_score:
            best_eps = eps
            best_dbscan_score = score

print(f"Best eps for DBSCAN: {best_eps} with Silhouette Score: {best_dbscan_score}")
```

```
Best eps for DBSCAN: 0 with Silhouette Score: -1
```