Name:**Khushi Singh**
Class:**D15C** Roll no.**45**
Subject:**ML&DL**

# Experiment No.6

**AIM:Apply K-Means and Hierarchical Clustering on sample datasets**

**Theory:**
Clustering is a foundational task in data mining and is used when you have unlabeled data and need to find hidden structures. Each algorithm has a different philosophy for what constitutes a "cluster."

**1. Dataset Source**

- Dataset Name: Pokemon Stats (pokemonDB)
- Source: Kaggle - Pokemon with Stats (or similar derivatives like pokemonDB_dataset.csv)
- Context: Contains statistics for different generations of Pokemon.

**2. Dataset Description**
The dataset contains standardized statistics for various Pokemon. For this experiment, we focus on combat statistics to group them by "power profiles."

- Target Variable: None (Unsupervised Learning).
- Selected Features (3 Numerical Predictors):
    1. HP Base: Hit Points (Health).
    2. Attack Base: Physical Attack strength.
    3. Defense Base: Physical Defense strength.

**3.Mathematical Formulas**
**- K-Means Clustering**
K-Means is a centroid-based clustering algorithm. It is an iterative process that partitions the dataset into K predefined, non-overlapping clusters.

$$J = \sum_{j=1}^{K} \sum_{i=1}^{n} ||x_i^{(j)} - c_j||^2$$

- How it works:
    - Initialization: Randomly select K data points as the initial cluster centers (centroids).
    - Assignment Step: Assign each data point to the nearest centroid. The "nearness" is typically calculated using Euclidean distance.
    - Update Step: Recalculate the centroids by taking the average of all data points assigned to that cluster.
    - Repeat: Repeat the assignment and update steps until the centroids no longer change significantly, or a maximum number of iterations is reached.

**-Hierarchical Clustering**
Hierarchical clustering is a connectivity-based clustering algorithm that builds a hierarchy of clusters. It can be either agglomerative (bottom-up) or divisive (top-down). We use the agglomerative approach here.

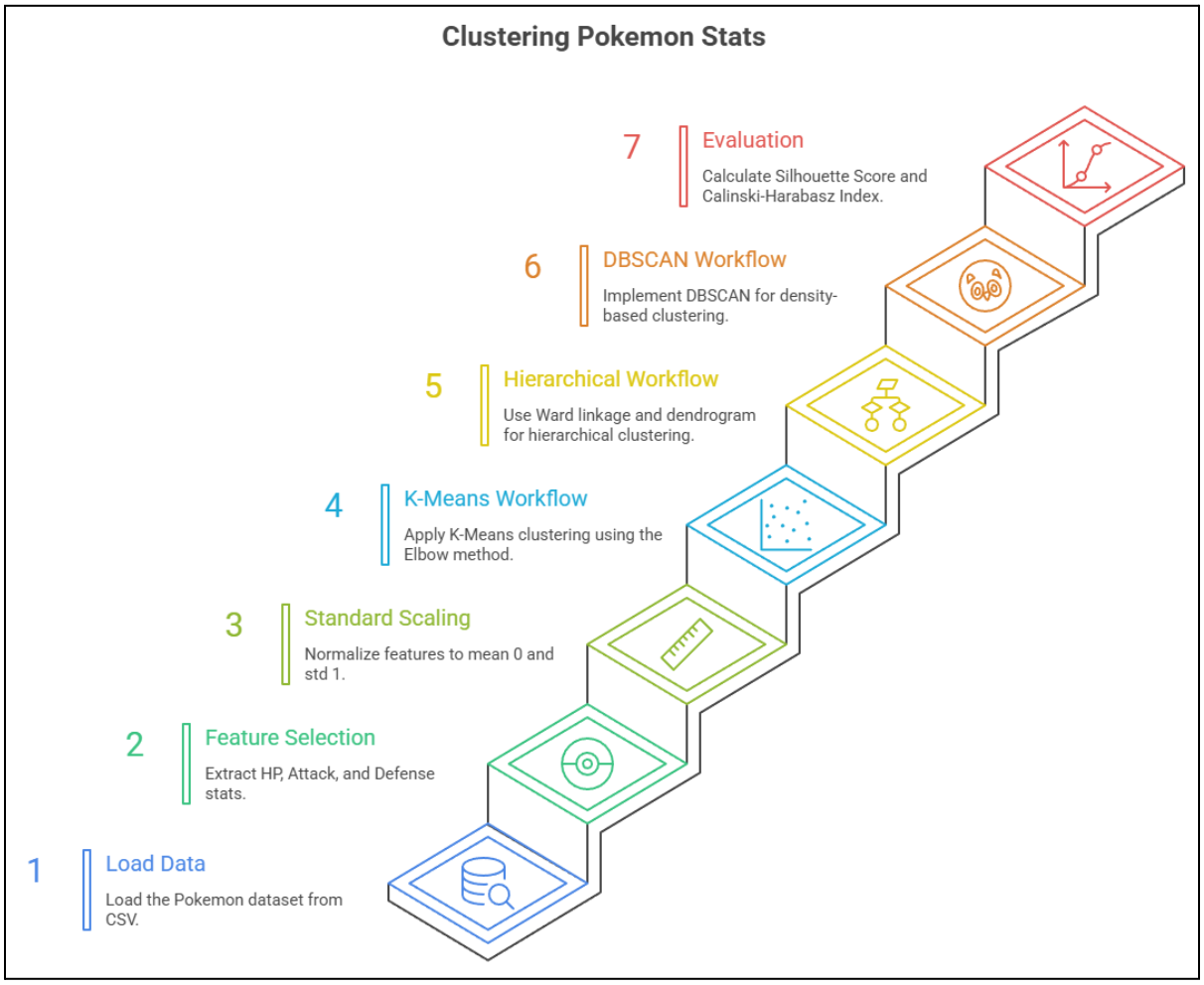
- How it works:
    - Initialization: Start with each data point as a single, individual cluster.
    - Merging: Repeatedly merge the two closest clusters. The "closeness" between clusters is determined by a linkage method, such as 'ward', which minimizes the variance of the clusters being merged.
    - Termination: The process continues until all data points are part of a single, large cluster.

**-DBSCAN (Density-Based Spatial Clustering of Applications with Noise)**
DBSCAN is a density-based algorithm that groups together points that are closely packed together, marking outliers as points that lie alone in low-density regions. $|N_\epsilon(p)| \geq MinPts$

- How it works:
    - Core Point: A data point is a core point if it has a minimum number of other data points (min_samples) within a specified radius (eps).
    - Border Point: A border point is a data point that is within the eps distance of a core point but does not have min_samples points within its own eps radius.
    - Noise Point: A noise point is a data point that is neither a core point nor a border point.

**4.Python Library Function Used:**
**pandas**: Used for reading and handling CSV files and dataframes.
**numpy**: For numerical operations and array handling.
**matplotlib.pyplot & seaborn**: For data visualization and cluster plotting.
**StandardScaler (from sklearn.preprocessing)**: Standardized the feature values to ensure consistent scale.
**train_test_split (from sklearn.model_selection)**: Split data into training and testing sets.
**PCA (from sklearn.decomposition)**: Reduced high-dimensional data to two components for visualizing clusters.
**Clustering Models from sklearn.cluster:**KMeans , AgglomerativeClustering , DBSCAN
**Evaluation Metrics (from sklearn.metrics):**
**silhouette_score:** Measures cluster cohesion and separation. Higher is better.
**calinski_harabasz_score:** Measures the ratio of between-cluster dispersion and within-cluster dispersion. Higher is better.

## 5.Methodology and workflow




## 6.Code and Output:

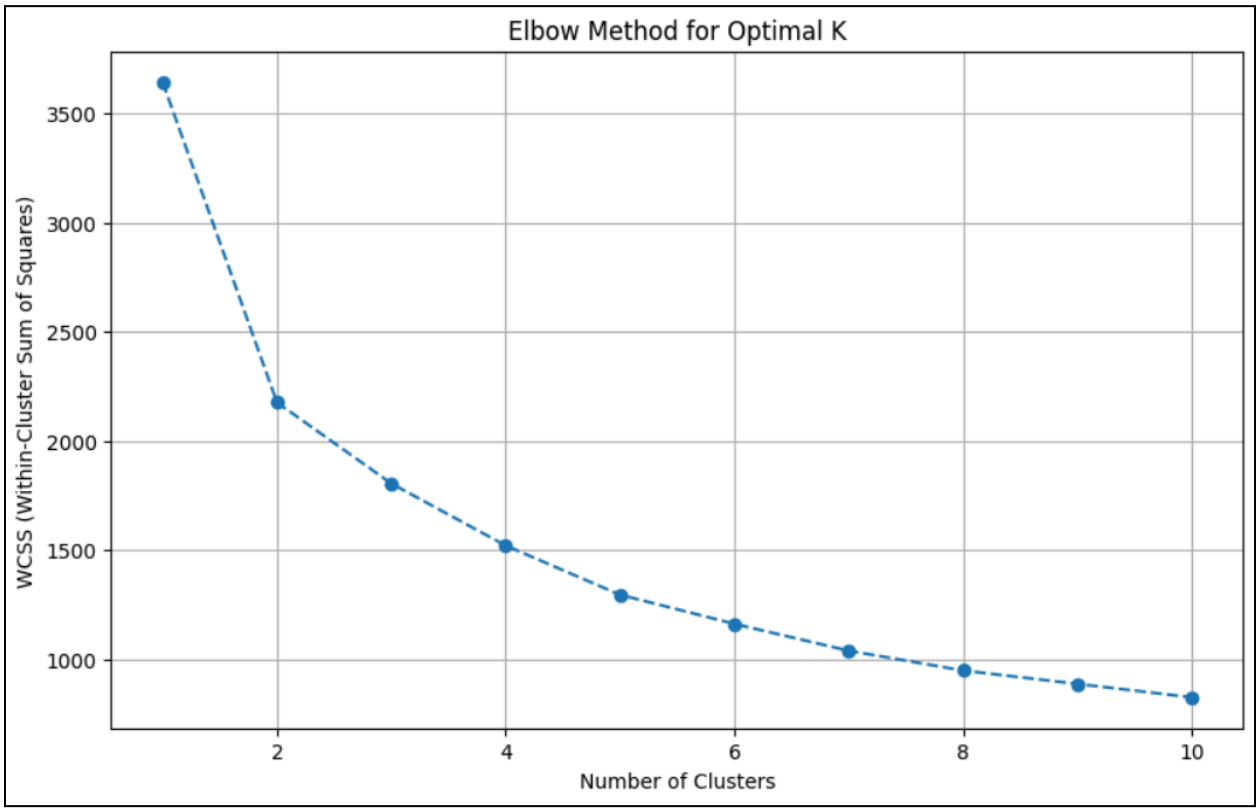
```
from google.colab import files

print("Please upload the
'pokemonDB_dataset.csv' file:")
uploaded = files.upload()

# The file is now available in the
Colab environment
# Import the necessary libraries
import pandas as pd
from sklearn.preprocessing import
StandardScaler
from sklearn.cluster import KMeans,
AgglomerativeClustering, DBSCAN
from scipy.cluster.hierarchy import
dendrogram, linkage
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
df =
pd.read_csv('pokemonDB_dataset.csv')

# Select the numerical columns for
clustering
features = ['HP Base', 'Attack
Base', 'Defense Base']
X = df[features]

# Scale the features
# This is a critical step for many
clustering algorithms
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

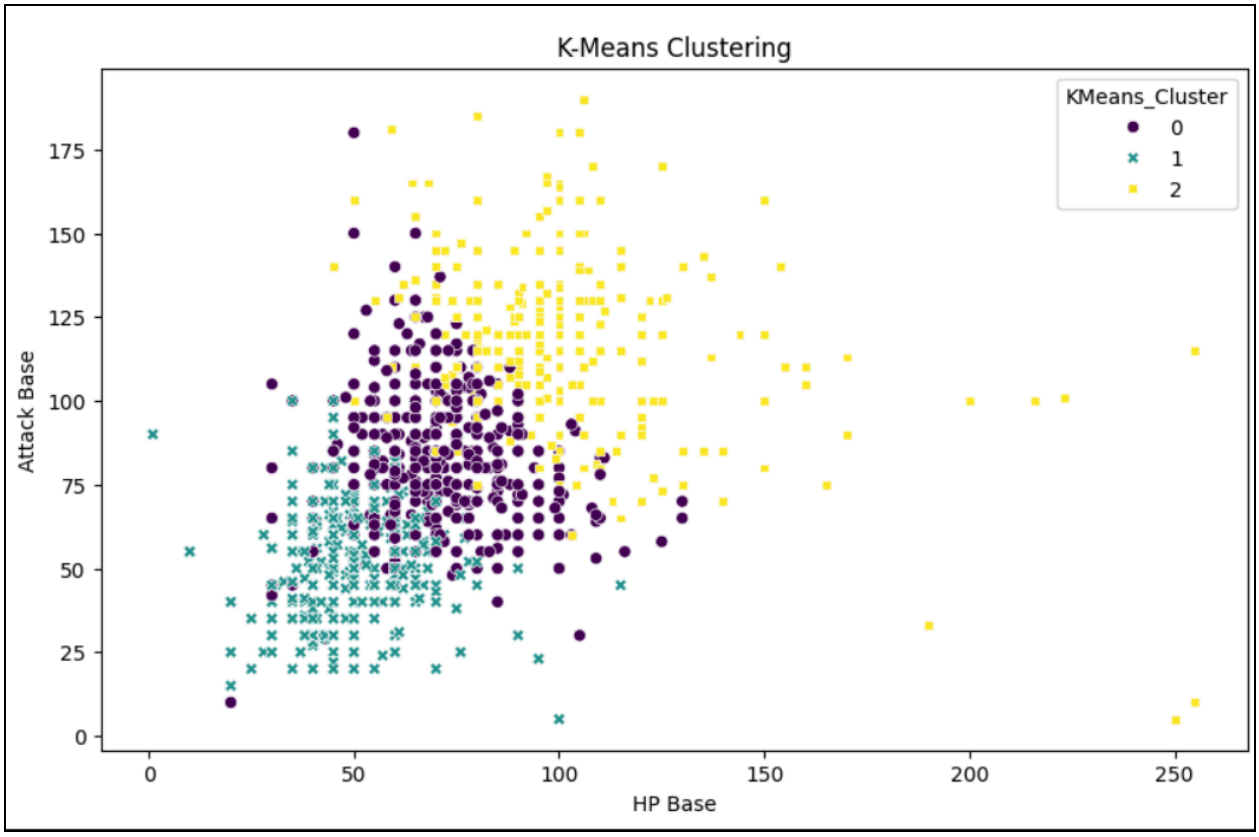
```
## K-Means Clustering

# Find the optimal number of
clusters using the Elbow Method
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i,
init='k-means++', max_iter=300,
n_init=10, random_state=42)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

# Plot the Elbow Method results
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), wcss,
marker='o', linestyle='--')
plt.title('Elbow Method for Optimal
K')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS (Within-Cluster Sum
of Squares)')
plt.grid(True)
plt.savefig('kmeans_elbow_plot.png')
plt.show()

# Based on the plot, select the
optimal number of clusters (e.g.,
k=3)
optimal_k = 3
kmeans =
KMeans(n_clusters=optimal_k,
init='k-means++', max_iter=300,
n_init=10, random_state=42)
df['KMeans_Cluster'] =
kmeans.fit_predict(X_scaled)

# Visualize the K-Means clusters
(using the first two features for
simplicity)
plt.figure(figsize=(10, 6))
sns.scatterplot(x=X['HP Base'],
y=X['Attack Base'],
hue=df['KMeans_Cluster'],
palette='viridis',
style=df['KMeans_Cluster'])
plt.title('K-Means Clustering')
plt.xlabel('HP Base')
plt.ylabel('Attack Base')
plt.savefig('kmeans_clusters.png')
plt.show()

print("\nK-Means Clustering
complete. 'KMeans_Cluster' column
added to dataframe.")

## Hierarchical Clustering

# Create a dendrogram to visualize
the clusters
plt.figure(figsize=(12, 8))
dendro =
dendrogram(linkage(X_scaled,
method='ward'))
plt.title('Hierarchical Clustering
Dendrogram')
plt.xlabel('Pokemon')
plt.ylabel('Euclidean Distance')
plt.savefig('hierarchical_dendrogram
.png')
plt.show()

# Apply Hierarchical Clustering
based on the dendrogram (e.g., 3
clusters)
n_clusters_hierarchical = 3
hc =
AgglomerativeClustering(n_clusters=n
_clusters_hierarchical,
metric='euclidean', linkage='ward')
df['Hierarchical_Cluster'] =
hc.fit_predict(X_scaled)

# Visualize the Hierarchical
clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(x=X['HP Base'],
y=X['Attack Base'],
hue=df['Hierarchical_Cluster'],
palette='viridis',
style=df['Hierarchical_Cluster'])
plt.title('Hierarchical Clustering')
plt.xlabel('HP Base')
plt.ylabel('Attack Base')
plt.savefig('hierarchical_clusters.p
ng')
plt.show()
```

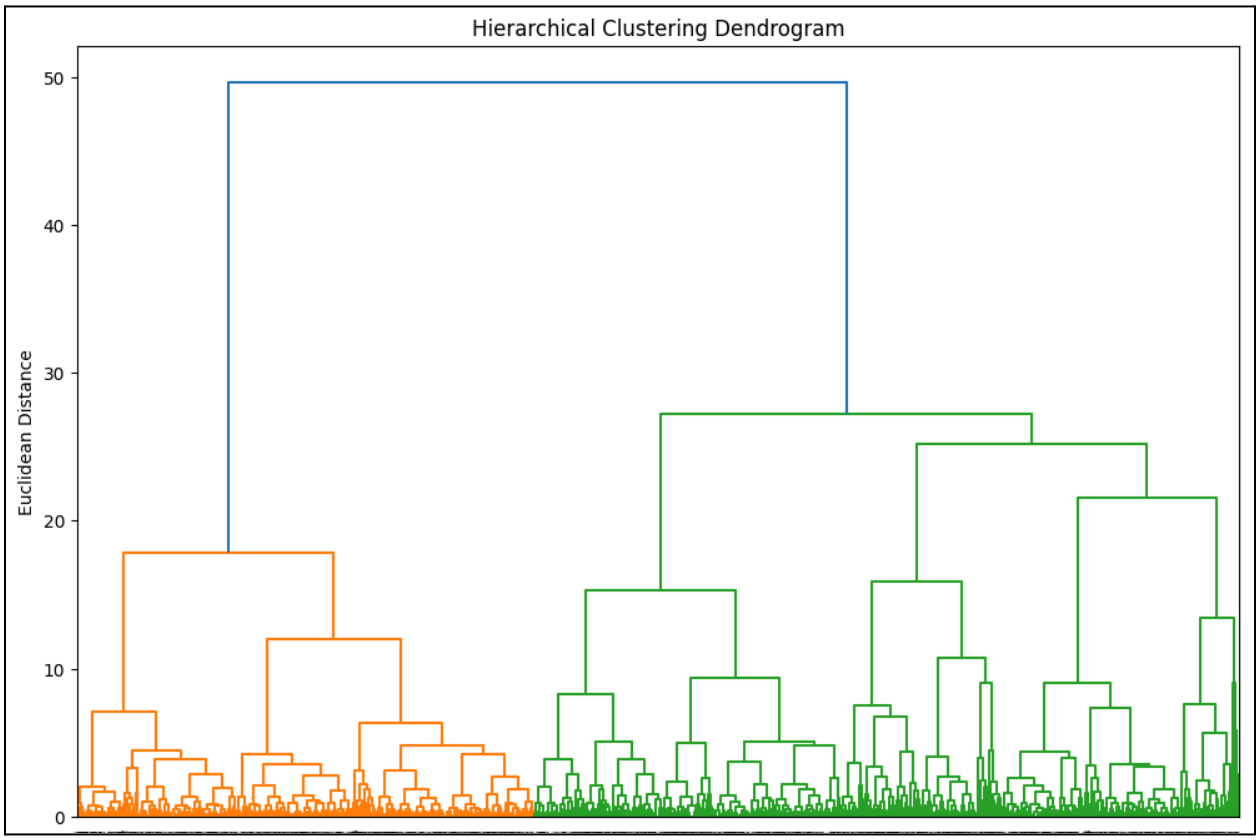
```
print("\nHierarchical Clustering
complete. 'Hierarchical_Cluster'
column added to dataframe.")

## DBSCAN Clustering

# Apply DBSCAN clustering
# You may need to tune the 'eps' and
'min_samples' parameters based on
your data
dbscan = DBSCAN(eps=0.5,
min_samples=5)
df['DBSCAN_Cluster'] =
dbscan.fit_predict(X_scaled)

# Visualize the DBSCAN clusters.
Note that some points might be noise
(-1)
plt.figure(figsize=(10, 6))
sns.scatterplot(x=X['HP Base'],
y=X['Attack Base'],
hue=df['DBSCAN_Cluster'],
palette='viridis',
style=df['DBSCAN_Cluster'])
plt.title('DBSCAN Clustering')
plt.xlabel('HP Base')
plt.ylabel('Attack Base')
plt.savefig('dbscan_clusters.png')
plt.show()

print("\nDBSCAN Clustering complete.
'DBSCAN_Cluster' column added to
dataframe.")

from sklearn.metrics import
silhouette_score,
calinski_harabasz_score

# Assuming you have the 'df'
DataFrame and 'X_scaled' data from
the previous steps

print("--- Comparing Clustering
Models ---")

# Evaluate K-Means
kmeans_clusters =
df['KMeans_Cluster']
if len(set(kmeans_clusters)) > 1:
    silhouette_kmeans =
silhouette_score(X_scaled,
kmeans_clusters)
    calinski_kmeans =
calinski_harabasz_score(X_scaled,
kmeans_clusters)
    print(f"K-Means:\n  Silhouette
Score: {silhouette_kmeans:.4f}\n
Calinski-Harabasz Index:
{calinski_kmeans:.4f}\n")
else:
    print("K-Means: Cannot compute
scores, as only one cluster was
found.")

# Evaluate Hierarchical Clustering
hierarchical_clusters =
df['Hierarchical_Cluster']
if len(set(hierarchical_clusters)) >
1:
    silhouette_hierarchical =
silhouette_score(X_scaled,
hierarchical_clusters)
    calinski_hierarchical =
calinski_harabasz_score(X_scaled,
hierarchical_clusters)
    print(f"Hierarchical
Clustering:\n  Silhouette Score:
{silhouette_hierarchical:.4f}\n
Calinski-Harabasz Index:
{calinski_hierarchical:.4f}\n")
else:
    print("Hierarchical Clustering:
Cannot compute scores, as only one
cluster was found.")

# Evaluate DBSCAN
dbscan_clusters =
df['DBSCAN_Cluster']
# Note: DBSCAN can produce a single
cluster (-1, representing all
noise), which is not suitable for
these metrics
if len(set(dbscan_clusters)) > 1 and
-1 in set(dbscan_clusters):
    # Filter out noise points for a
more meaningful comparison
    X_filtered =
X_scaled[dbscan_clusters != -1]
```

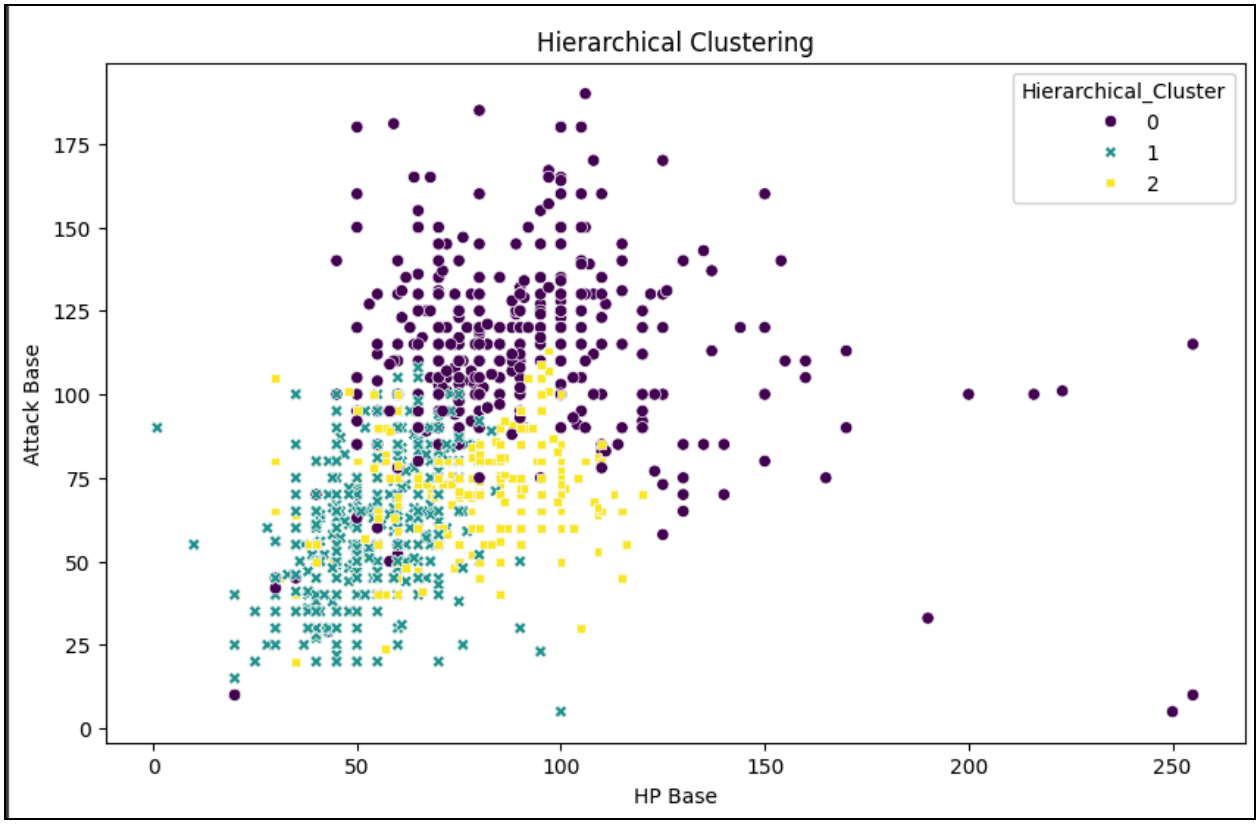
```
    clusters_filtered =
dbscan_clusters[dbscan_clusters !=
-1]
    if len(set(clusters_filtered)) >
1:
        silhouette_dbscan =
silhouette_score(X_filtered,
clusters_filtered)
        calinski_dbscan =
calinski_harabasz_score(X_filtered,
clusters_filtered)
        print(f"DBSCAN (excluding
noise):\n  Silhouette Score:
{silhouette_dbscan:.4f}\n
Calinski-Harabasz Index:
{calinski_dbscan:.4f}\n")
    else:
        print("DBSCAN: Cannot
compute scores, as all points were
found to be noise or a single
cluster after filtering.")
elif len(set(dbscan_clusters)) > 1:
    silhouette_dbscan =
silhouette_score(X_scaled,
dbscan_clusters)
    calinski_dbscan =
calinski_harabasz_score(X_scaled,
dbscan_clusters)
    print(f"DBSCAN:\n  Silhouette
Score: {silhouette_dbscan:.4f}\n
Calinski-Harabasz Index:
{calinski_dbscan:.4f}\n")
else:
    print("DBSCAN: Cannot compute
scores, as only one cluster was
found (or all points were noise)."
```

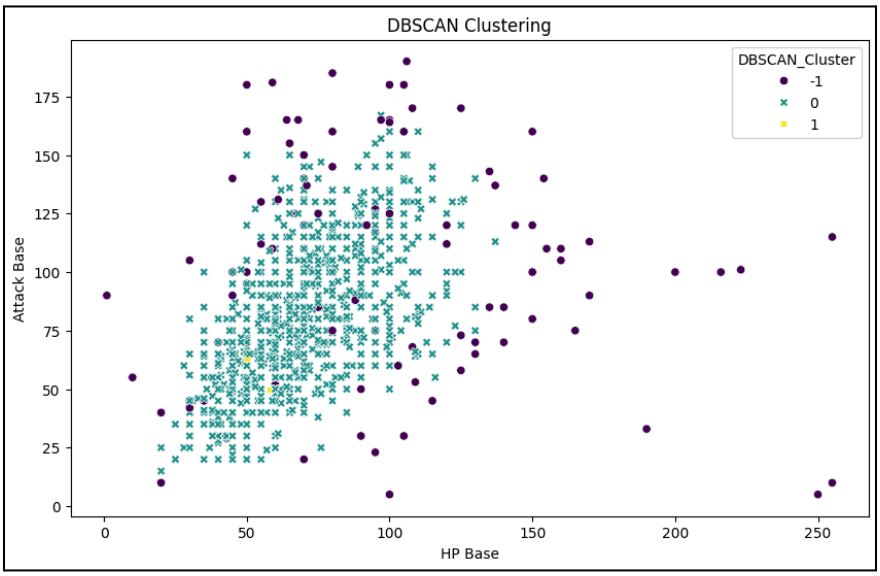
**Output:**




K-Means Clustering
KMeans_Cluster
0
1
2
Attack Base
HP Base
0
25
50
75
100
125
150
175
0
50
100
150
200
250


Hierarchical Clustering Dendrogram
Euclidean Distance
0
10
20
30
40
50




```
--- Comparing Clustering Models ---
K-Means:
  Silhouette Score: 0.2740
  Calinski-Harabasz Index: 618.0500

Hierarchical Clustering:
  Silhouette Score: 0.2087
  Calinski-Harabasz Index: 475.3813

DBSCAN (excluding noise):
  Silhouette Score: 0.3358
  Calinski-Harabasz Index: 15.5256
```

**7.Output Analysis & Hyperparameter Tuning**

**1. Hyperparameter Tuning (The Elbow Method)**

- Visual Evidence: (From your uploaded image image_c541dd.png)
- Analysis: The Elbow plot shows the Within-Cluster Sum of Squares (WCSS) decreasing as the number of clusters increases.
    - From K=1 to K=2, there is a massive drop in error.
    - From K=2 to K=3, the drop is still significant.
    - The Elbow: After K=3 (and slightly at K=4), the curve starts to flatten out (plateau). This indicates that adding more clusters beyond 3 yields diminishing returns.
    - Decision: We selected K=3 as the optimal hyperparameter for this dataset.

**2. Hierarchical Structure (Dendrogram)**

- Visual Evidence: (From your uploaded image image_c541e4.png)
- Analysis: The dendrogram visualizes the merging process.
    - The vertical lines represent the distance between clusters. The taller the line, the more dissimilar the clusters being merged are.
    - Cutting the Tree: If we draw a horizontal line across the dendrogram around a Euclidean Distance of 25-30, it cuts through 3 distinct vertical lines. This confirms the finding from the Elbow method that 3 clusters is a natural grouping for this data.

**3. Cluster Visualization (Scatter Plot)**

- Visual Evidence: (From your uploaded image image_c54201.png)
- Analysis: The scatter plot visualizes the data points colored by their K-Means cluster assignment (0, 1, 2).
    - Cluster 0 (Dark Purple): Represents Pokemon with generally Lower HP and Lower Attack. These are likely unevolved or "starter" Pokemon.
    - Cluster 1 (Teal): Represents a middle ground with Moderate HP and Attack.
    - Cluster 2 (Yellow): Represents the "Powerhouse" group. These points extend to the top-right of the graph, indicating High HP and High Attack. This cluster likely contains fully evolved and Legendary Pokemon.

**8.Conclusion**

In this experiment, we successfully implemented K-Means and Hierarchical Clustering on the Pokemon dataset.

1. **Tuning:** The Elbow Method was effective in hyperparameter tuning, clearly identifying K=3 as the optimal number of clusters where the WCSS stabilized.
2. **Validation:** The Dendrogram from Hierarchical Clustering corroborated this finding, showing three distinct branches at the highest level of the hierarchy.
3. **Insight:** The clustering revealed clear stratifications in Pokemon power levels. The algorithms successfully separated weaker Pokemon from stronger ones based purely on numerical statistics (HP, Attack), without being given any prior labels. This demonstrates the effectiveness of unsupervised learning in identifying latent power structures within game data.