In [ ]:
```
from preprocessing.dataset import Dataset

from sklearn.mixture import GaussianMixture
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

import json
```

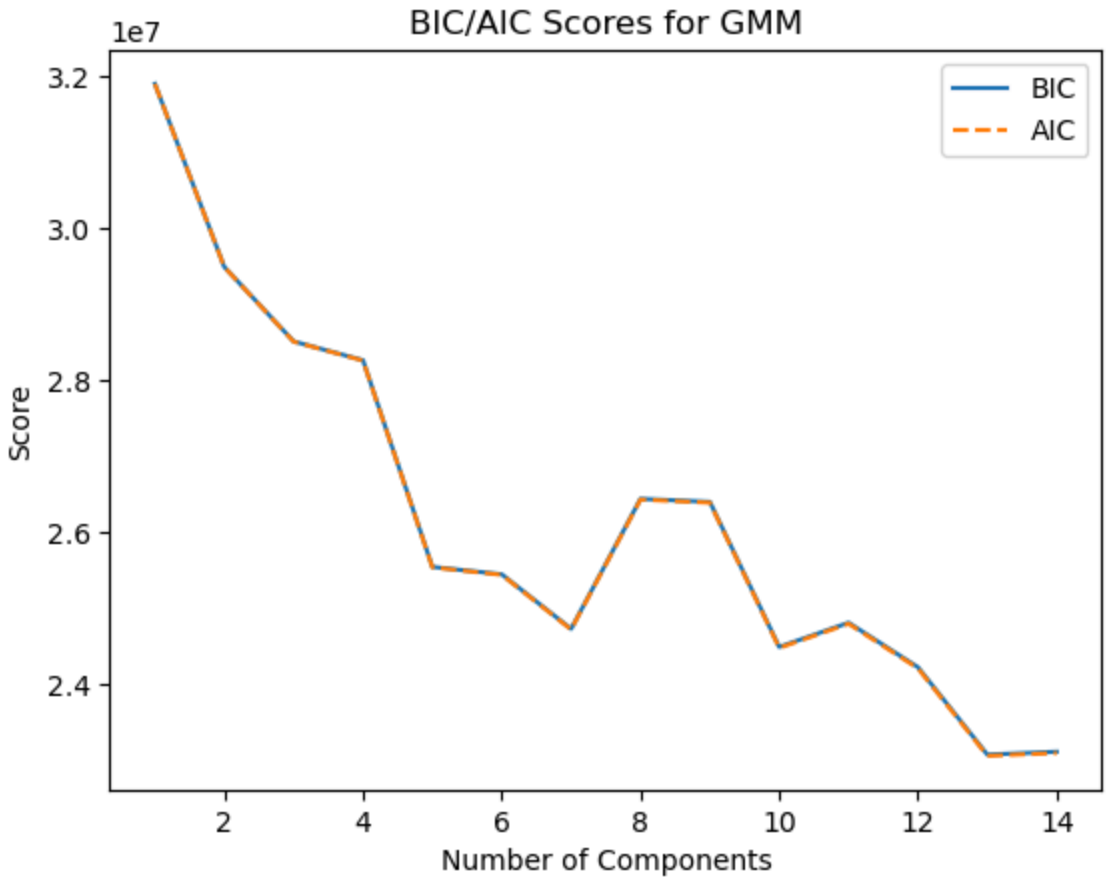
```
Running cells with 'Python 3.12.1' requires the ipykernel package.

Run the following command to install 'ipykernel' into the Python environment.

Command: '/opt/homebrew/bin/python3.12 -m pip install ipykernel -U --user --force-rei
nstall'
```

In [ ]:
```
dataset = Dataset("Datasets/olympics_athletes_1_200000_new.csv")
```

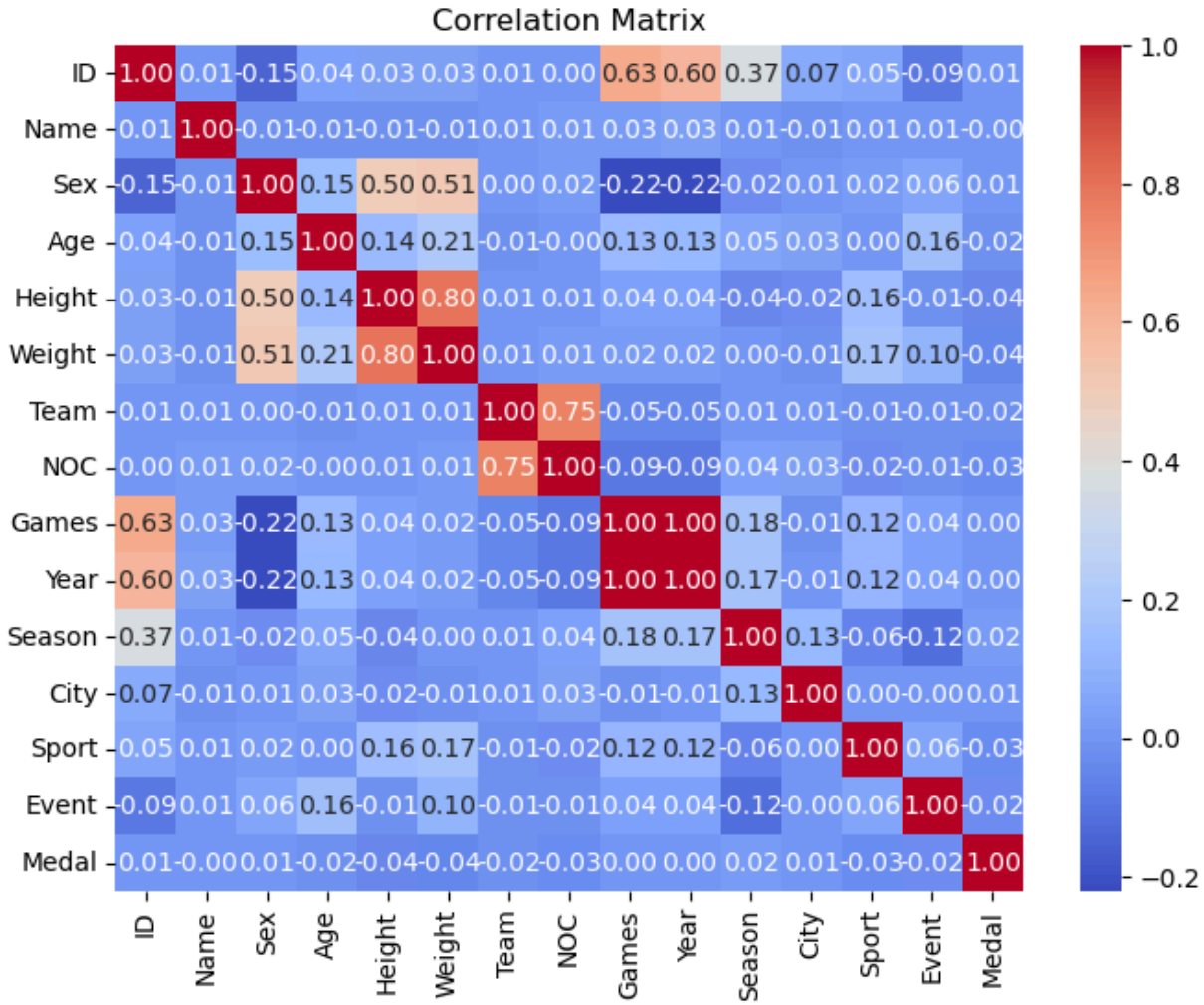
In [17]:
```
bics = []
aics = []
n_components = len(dataset.datapoints.columns)
data = dataset.datapoints.to_numpy()

for k in range(1, n_components+1):
    print(f"Fitting GMM with {k} components")
    gmm = GaussianMixture(n_components=k)
    gmm.fit(data)

    bics.append(gmm.bic(data))
    aics.append(gmm.aic(data))
```

```
Fitting GMM with 1 components
Fitting GMM with 2 components
Fitting GMM with 3 components
Fitting GMM with 4 components
Fitting GMM with 5 components
Fitting GMM with 6 components
Fitting GMM with 7 components
Fitting GMM with 8 components
Fitting GMM with 9 components
Fitting GMM with 10 components
Fitting GMM with 11 components
Fitting GMM with 12 components
Fitting GMM with 13 components
Fitting GMM with 14 components
```

In [19]:
```
comps = np.arange(1, n_components+1)

plt.plot(comps, bics, label='BIC')
plt.plot(comps, aics, '--', label='AIC')
plt.xlabel('Number of Components')
plt.ylabel('Score')
plt.title('BIC/AIC Scores for GMM')
plt.legend()
plt.show()
```



In [5]:
```
corr_mat = dataset.get_correlation()
```



Remove the highly correlated columns

In [6]:
```
selected_columns = set(dataset.datapoints.columns)
correlation_threshold = 0.7

for i in range(len(corr_mat.columns)):
    for j in range(i):
        if abs(corr_mat.iloc[i, j]) > correlation_threshold:
            colname_i = corr_mat.columns[i]
            colname_j = corr_mat.columns[j]

            # Remove one of the correlated columns
            if colname_i in selected_columns:
                selected_columns.remove(colname_j)

print(selected_columns)
```

```
{'Age', 'ID', 'Weight', 'Event', 'City', 'Sport', 'Medal', 'Name', 'Year', 'NOC', 'Se
ason', 'Sex'}
```

Remove the columns that won't contain any useful information for unsupervised learning

In [7]:
```
for col in ["Name", "ID", "City", "Year", "Season"]:
    selected_columns.remove(col)

print(selected_columns)
```

```
{'Age', 'Weight', 'Event', 'Sport', 'Medal', 'NOC', 'Sex'}
```

In [8]:
```
sub_df = dataset.datapoints[list(selected_columns)]
sub_data = sub_df.to_numpy()
```

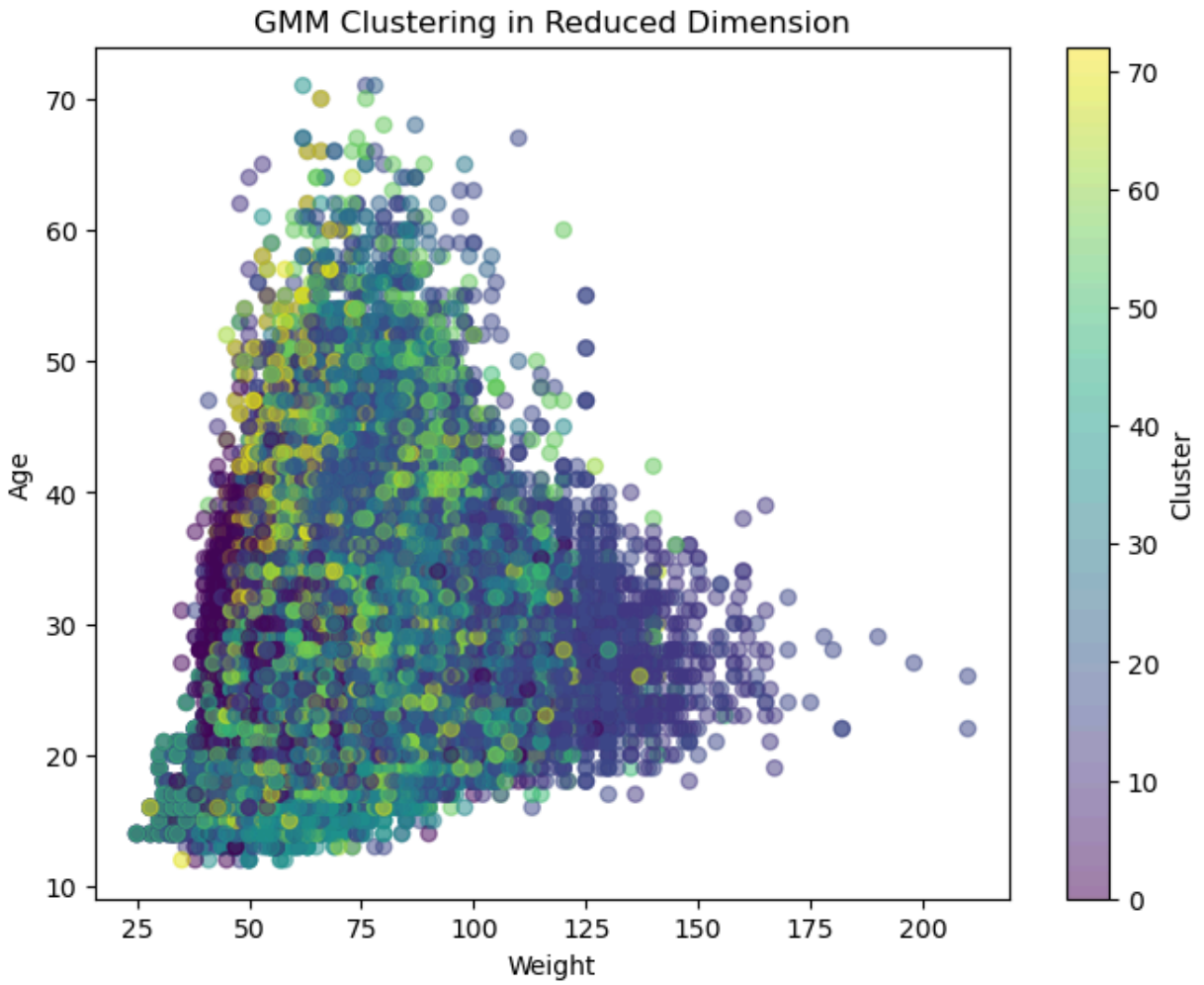
In [46]:
```
gmm = GaussianMixture(n_components=73)
```

In [47]:
```
gmm.fit(sub_data)
BIC = gmm.bic(sub_data)

print(f"BIC on the reduced data: {BIC:.2E}")
```

```
BIC on the reduced data: 4.17E+06
```

In [48]:
```
cluster_labels = gmm.predict(sub_data)
```

In [49]:
```
plt.figure(figsize=(8, 6))
plt.scatter(sub_data[:, 0], sub_data[:, 3], c=cluster_labels, cmap='viridis', alpha=0.
plt.colorbar(label='Cluster')
plt.title('GMM Clustering in Reduced Dimension')
plt.xlabel('Weight')
plt.ylabel('Age')
plt.show()
```



In [9]:
```
encoded = pd.get_dummies(sub_df)
scaler = StandardScaler()
```

```
scaled = scaler.fit_transform(encoded)
dbscan = DBSCAN(eps=1.5)
clusters = dbscan.fit_predict(scaled)
```

In [10]:
```
with open("Datasets/olympics_athletes_1_200000.json", "r") as file:
    dic = json.load(file)
```

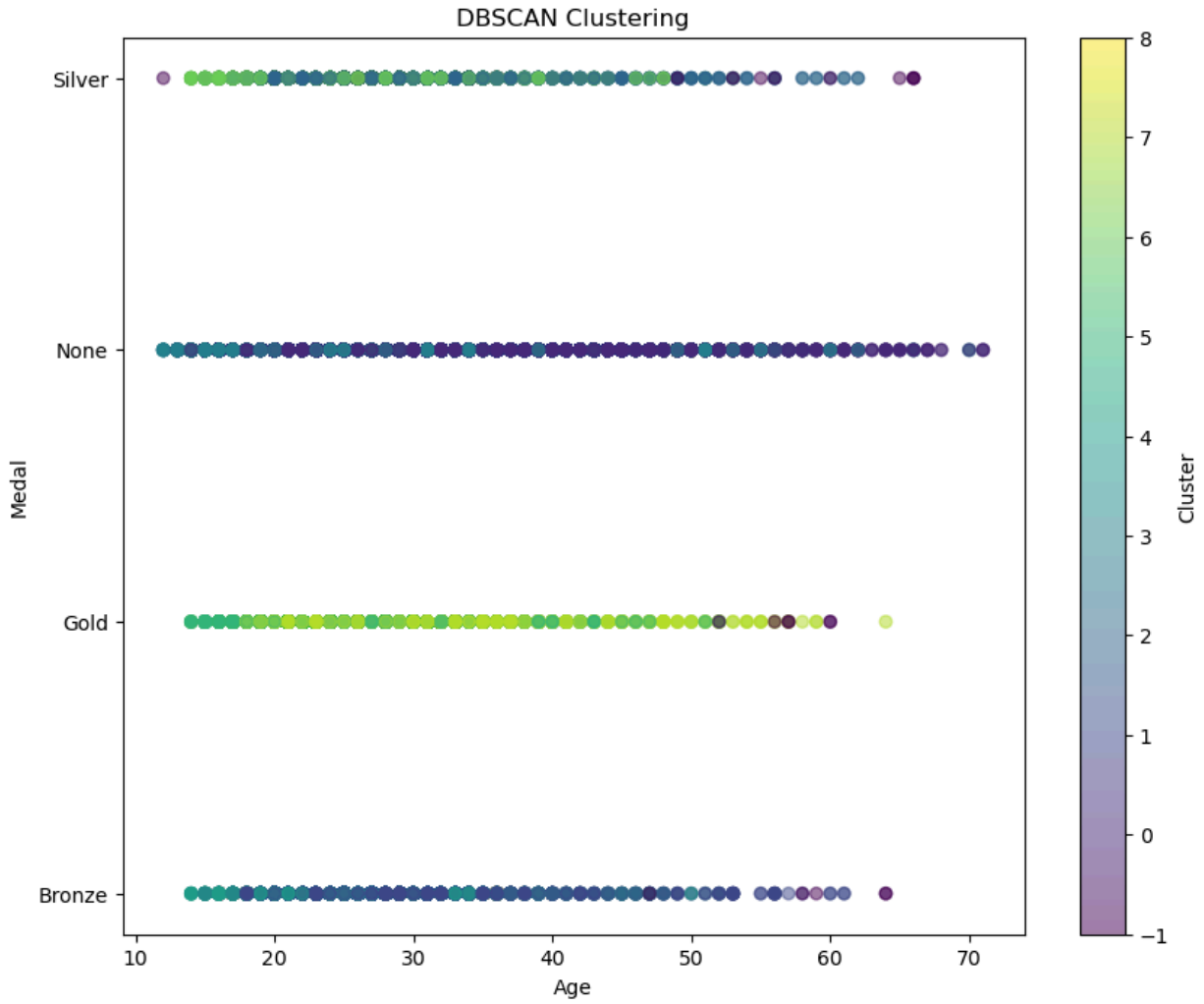
In [11]:
```
medal_type, medal_code = dic["Medal"]

medal_type = np.array(medal_type)
medal_code = np.array(medal_code)

print(medal_type)
print(medal_code)

no_medal = np.nonzero(medal_type == "None")[0]
no_med_code = medal_code[no_medal][0]

print(no_med_code)
```

```
['None' 'Bronze' 'Silver' 'Gold']
[2 0 3 1]
2
```

In [12]:
```
medals = encoded
```

In [43]:
```
msk = encoded["Medal"] != no_med_code

medals = encoded.loc[msk, :]
clusters = clusters[msk]
```

In [13]:
```
plt.figure(figsize=(10, 8))
plt.scatter(medals['Age'], medals['Medal'], c=clusters, cmap='viridis', alpha=0.5)
plt.colorbar(label='Cluster')
plt.yticks(medal_code, medal_type)
plt.title('DBSCAN Clustering')
plt.xlabel('Age')
plt.ylabel('Medal')
plt.show()
```



In [14]:
```
clust_inds = np.unique(clusters)
n_clust = clust_inds.shape[0]
print(n_clust)
```

```
10
```

In [58]:
```
clust_inds
```

Out[58]:
```
array([-1,  0,  1,  2,  3,  4,  5,  6,  7,  8])
```

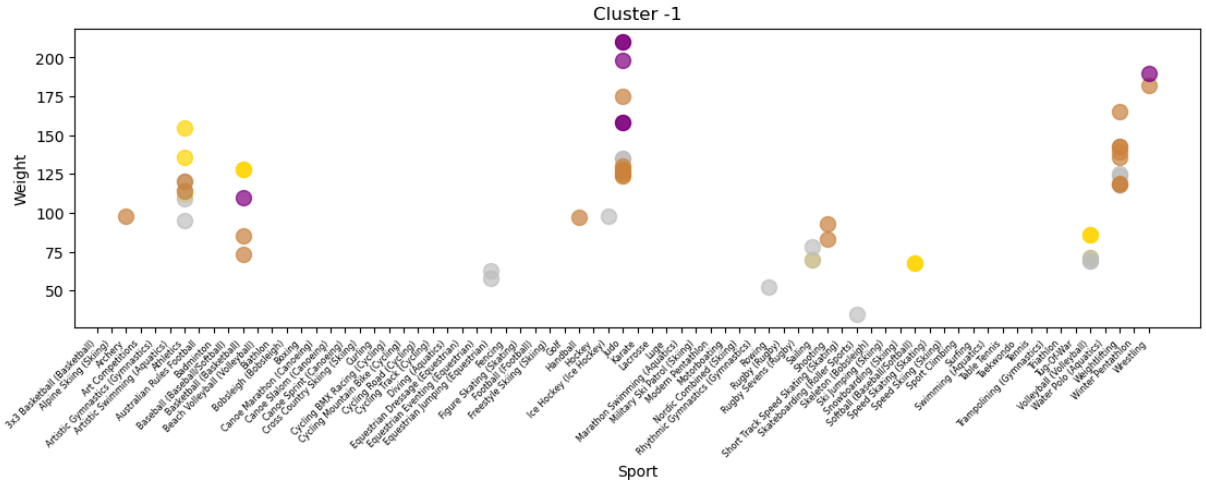
In [32]:
```
sport_type, sport_code = dic["Sport"]

sport_type = np.array(sport_type)
sport_code = np.array(sport_code)
```

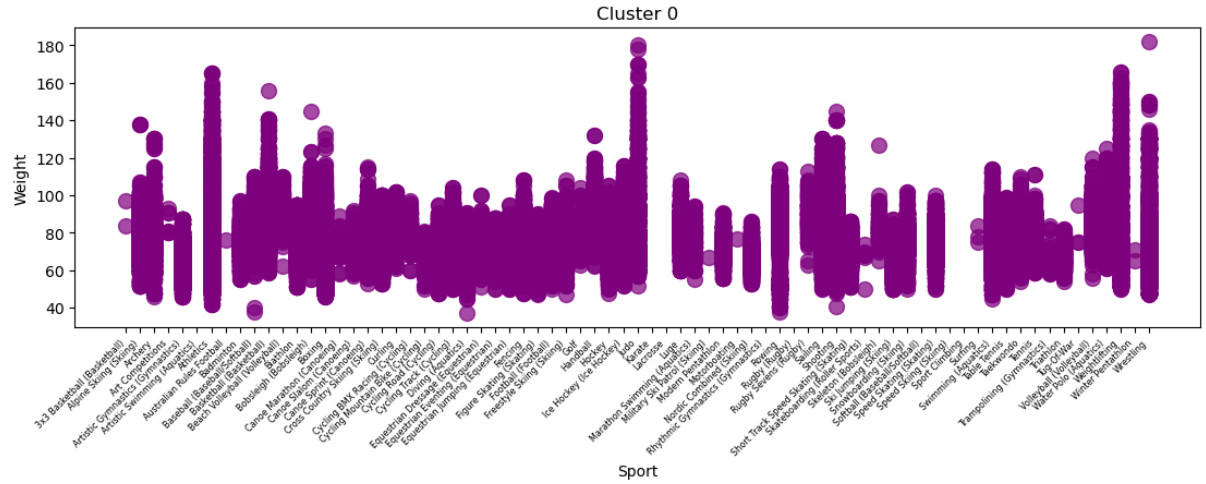
In [42]:
```
print(len(sport_code))
sport_code
```

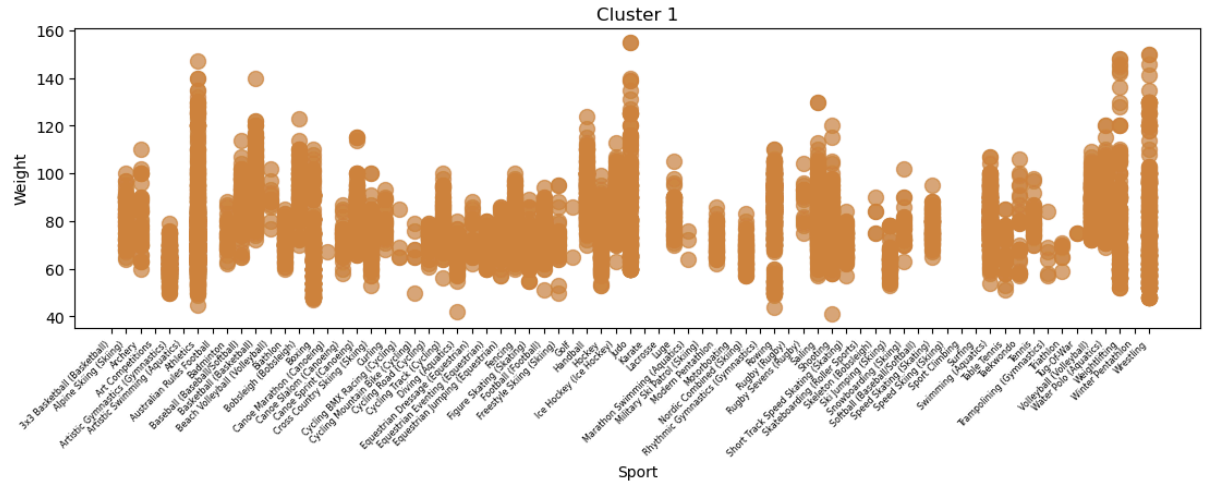
Out[42]:
```
73
array([33, 39, 58, 47, 15, 50,  8, 22, 12, 53, 51, 65, 30, 25, 17, 34,  2,
       44, 59, 55, 57, 64, 67, 24,  9, 48,  6, 54, 62, 32,  7, 36, 63, 46,
       13, 45, 11, 60, 14, 71, 38, 16, 66, 69, 42, 27, 40, 72, 68, 19,  5,
       20, 70,  1, 56, 35, 21, 61, 43, 28, 18, 52,  0, 10, 49, 23, 26,  4,
        3, 31, 41, 29, 37])
```

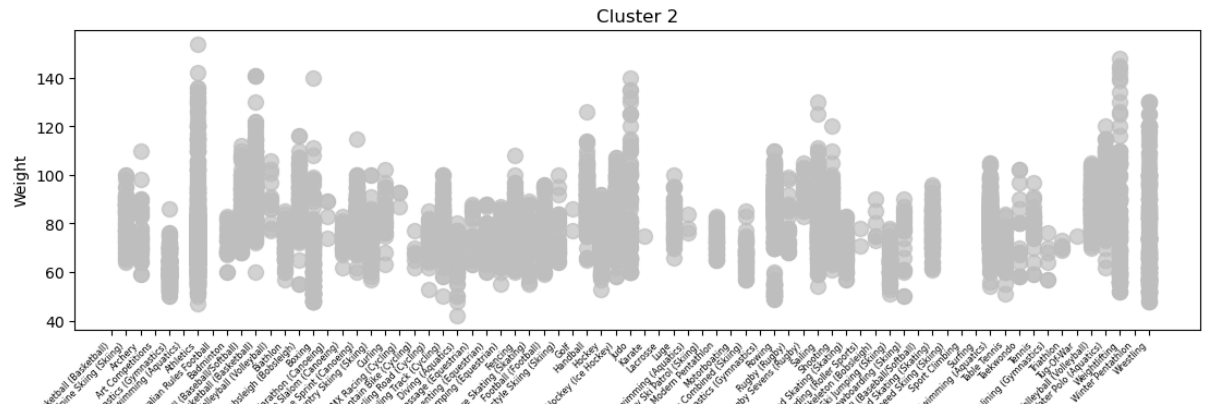
In [48]:
```
fig, axs = plt.subplots(10, 1, figsize=(12, 48))
medal_colors = {1: 'gold', 3: 'silver', 0: 'peru', 2: 'purple'}
```

```
for idx, (ax, clust_ind) in enumerate(zip(axs.flatten(), clust_inds)):

    tmp = encoded.loc[clusters == clust_ind, :]

    for medal_type, color in medal_colors.items():
        df = tmp[tmp["Medal"] == medal_type]
        ax.scatter(df["Sport"], df["Weight"], color=color, label=medal_type, alpha=0.7

    ax.set_xlabel("Sport")
    ax.set_ylabel("Weight")
    # ax.set_xticks(sport_code)
    ticks = np.arange(0, 73)
    ax.set_xticks(ticks)
    sports = [sport_type[np.nonzero(sport_code == tick)[0][0]] for tick in ticks]
    ax.set_xticklabels(sports, rotation=45, ha="right", fontsize="xx-small")
    ax.set_title(f"Cluster {clust_ind}")

fig.tight_layout(pad=3)
```

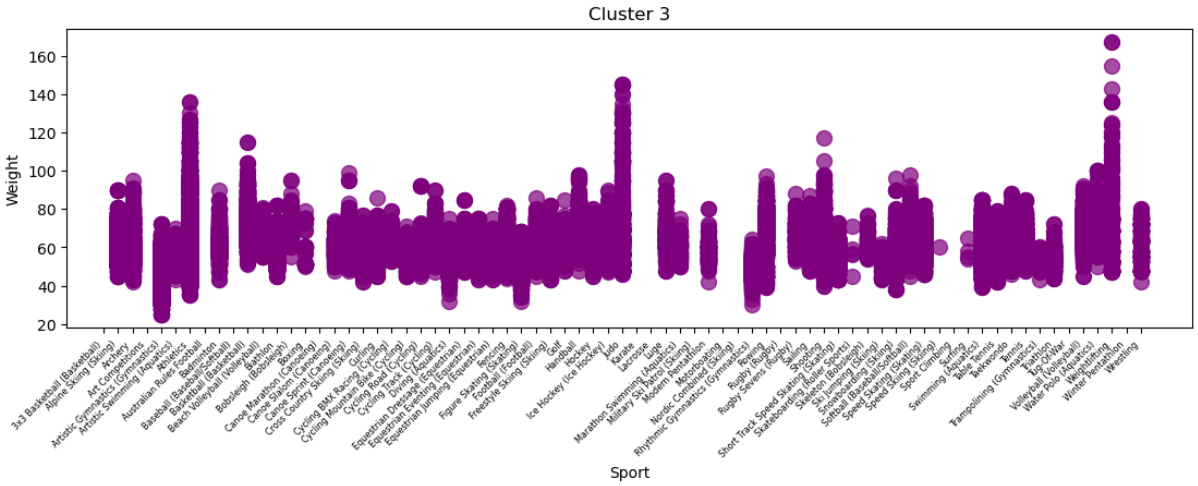

Cluster -1
Weight
Sport

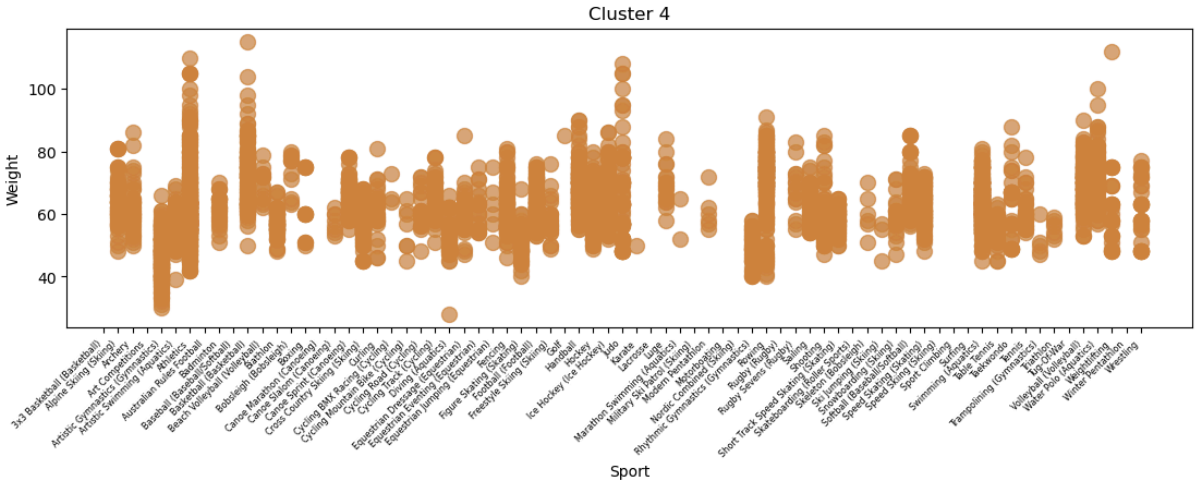

Cluster 0
Weight
Sport

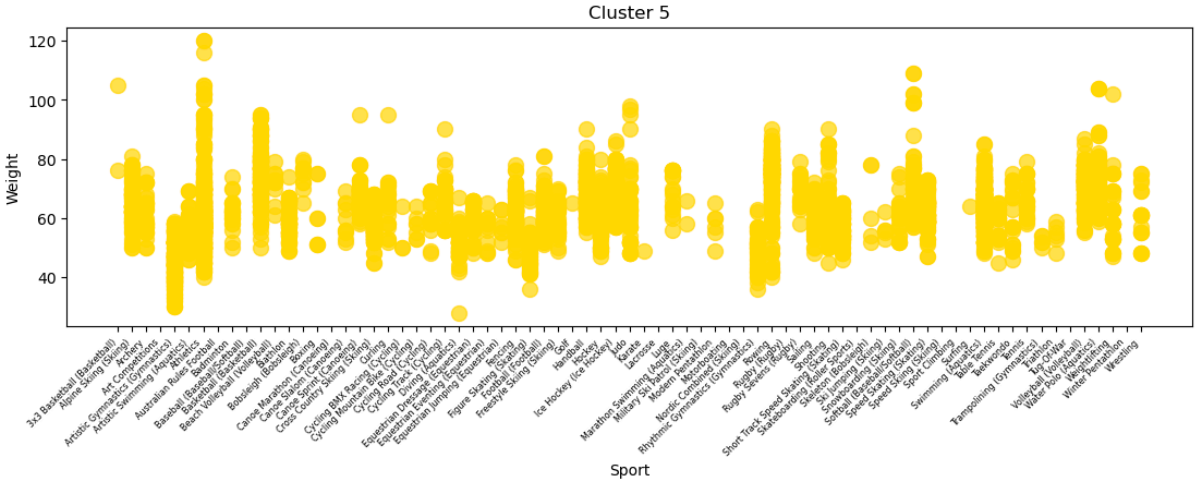

Cluster 1
Weight
Sport


Cluster 2
Weight

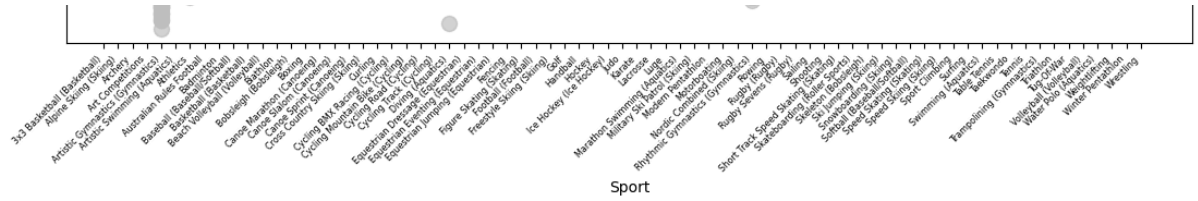

Sport

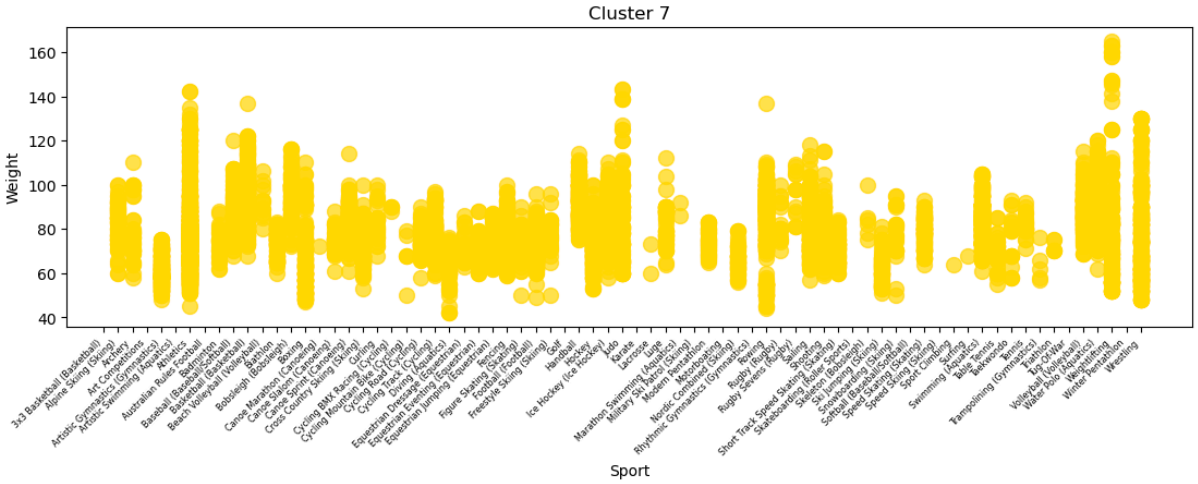

Cluster 3
Weight
Sport

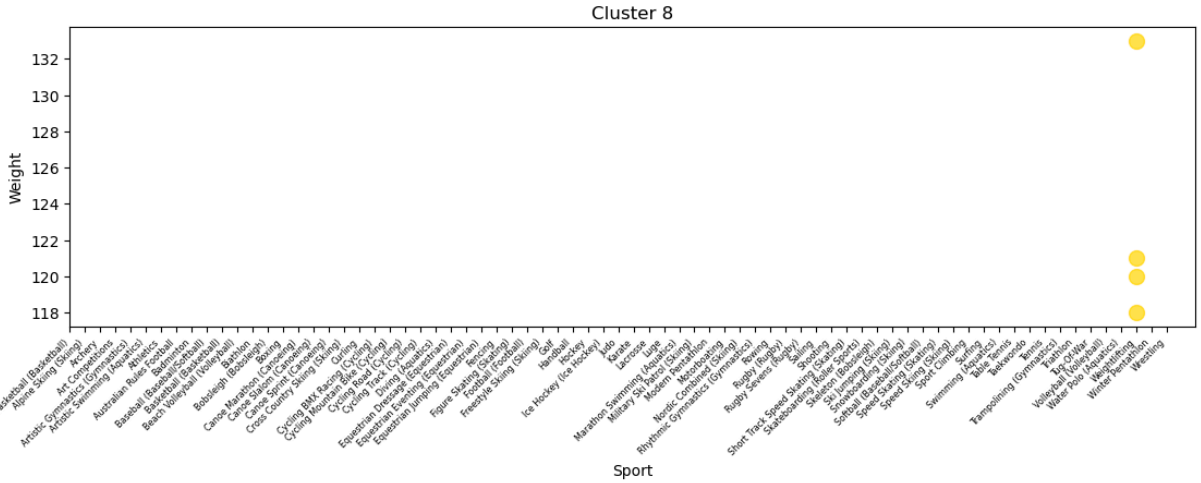

Cluster 4
Weight
Sport


Cluster 5
Weight
Sport


Cluster 6
Weight


Sport


Cluster 7
Weight
160
140
120
100
80
60
40
Sport


Cluster 8
Weight
132
130
128
126
124
122
120
118
Sport