Introduction:

Try to find out how MLB players' weight affect the time between their debut and Tommy John surgery date

Methods:

Gather data from Lahman's baseball database's 'People' table and list of players who underwent Tommy John surgery from Wikipedia

Use SQL Server to prepare the data

Use Python(Spyder) to perform cluster analysis(heirarchical and k-means)

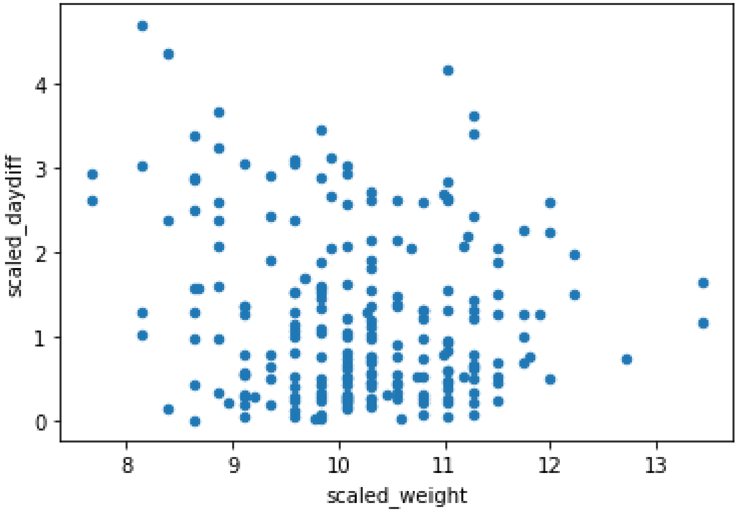
In [1]:

```
import pandas as pd
import pyodbc

#Gain data from SQL server, tables was imported into SQL Server from tw
o excel file
#excel file can be found in the same repository
sql_conn = pyodbc.connect('''DRIVER={ODBC Driver 13 for SQL Server};
                            SERVER=ALLENHO\MSSQLSERVER002;
                            DATABASE=TommyJohn;
                            Trusted_Connection=yes''')
query = '''
select distinct t.Player, t.Position, t.Throws, t.date_of_surgery, p.we
ight, datediff(day, p.debut, t.date_of_surgery) as daydiff
from TJ$ t
join People$ p
on t.Player=concat(nameFirst,' ',nameLast)
where p.weight is not null and datediff(day, p.debut, t.date_of_surger
y)>0 and datediff(day, p.debut, t.date_of_surgery)<7000
order by t.Player;
;
'''
```

```
df = pd.read_sql(query, sql_conn)

import matplotlib.pyplot as plt
```

In [2]:

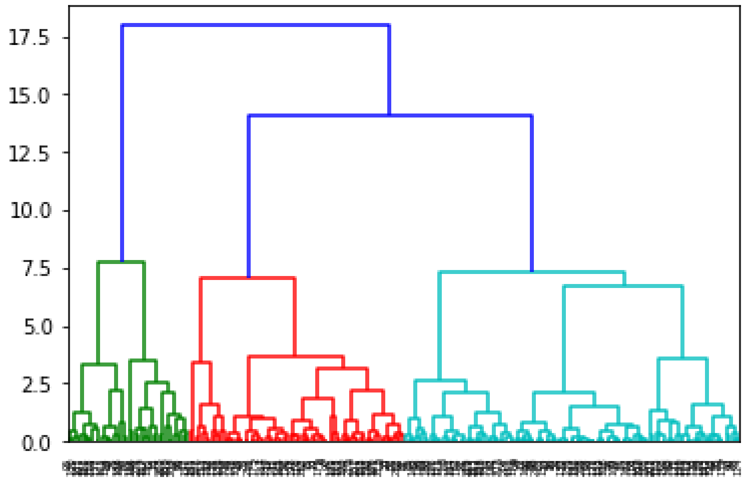
```
#-----standardize the data-----------------------------------------------------------
# Import the whiten function
from scipy.cluster.vq import whiten

# Use the whiten() function to standardize the data

# Scale weight and daydiff
df['scaled_weight'] = whiten(df['weight'])
df['scaled_daydiff'] = whiten(df['daydiff'])

# Plot the two columns in a scatter plot
df.plot(x='scaled_weight', y='scaled_daydiff', kind = 'scatter')
plt.show()

# Check mean and standard deviation of scaled values
print(df[['scaled_weight', 'scaled_daydiff']].describe())
```



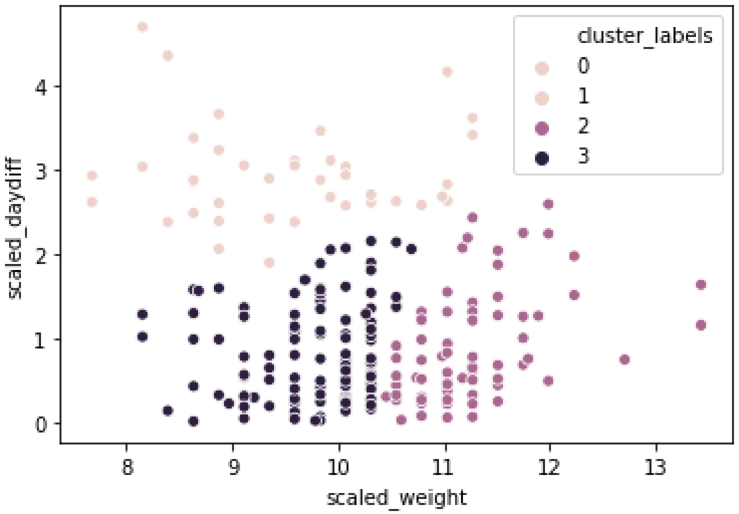
```
       scaled_weight  scaled_daydiff
count     221.000000      221.000000
count      221.000000      221.000000
mean        10.197669        1.234161
std          1.002270        1.002270
min          7.672282        0.009970
25%          9.590352        0.440808
50%         10.069870        0.987723
75%         11.028905        1.872899
max         13.426493        4.696490
```

In [3]:

```
#-----Hierarchical clustering ward method--------------------------------
------------------

# Import the fcluster and linkage functions
from scipy.cluster.hierarchy import fcluster, linkage
import seaborn as sns
# Use the linkage() function
distance_matrix = linkage(df[['scaled_weight', 'scaled_daydiff']], meth
od = 'ward', metric = 'euclidean')

from scipy.cluster.hierarchy import dendrogram
# Create a dendrogram
dn = dendrogram(distance_matrix)
plt.show()

# Assign cluster labels
df['cluster_labels'] = fcluster(distance_matrix, 3, criterion='maxclus
t')

# Plot clusters
sns.scatterplot(x='scaled_weight', y='scaled_daydiff',
                hue='cluster_labels', data = df )

# Display cluster centers of each cluster
print(df[['scaled_weight', 'scaled_daydiff', 'cluster_labels']].groupby
('cluster_labels').mean())

plt.show()
```



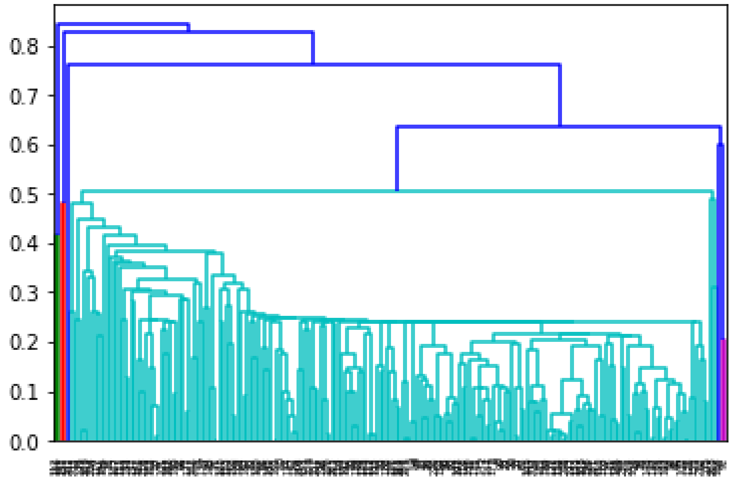
```
                scaled_weight  scaled_daydiff
cluster_labels
1                    9.563979        2.938938
2                   11.259209        0.900693
3                    9.742926        0.829481
```



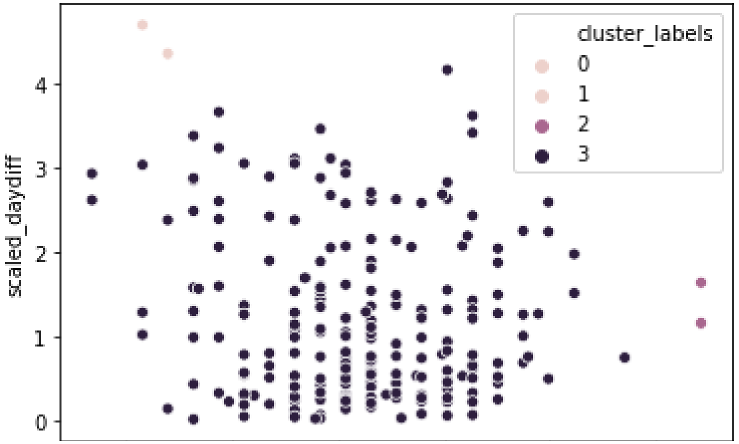
In [4]:

```python
#-----Hierarchical clustering single method------------------------------------

# Import the fcluster and linkage functions
from scipy.cluster.hierarchy import fcluster, linkage
import seaborn as sns

# Use the linkage() function
distance_matrix = linkage(df[['scaled_weight', 'scaled_daydiff']], method = 'single', metric = 'euclidean')

from scipy.cluster.hierarchy import dendrogram

# Create a dendrogram
dn = dendrogram(distance_matrix)
plt.show()

# Assign cluster labels
df['cluster_labels'] = fcluster(distance_matrix, 3, criterion='maxclust')

# Plot clusters
sns.scatterplot(x='scaled_weight', y='scaled_daydiff',
                hue='cluster_labels', data = df)
```

```
# Display cluster centers of each cluster
print(df[['scaled_weight', 'scaled_daydiff', 'cluster_labels']].groupby
('cluster_labels').mean())

plt.show()
```

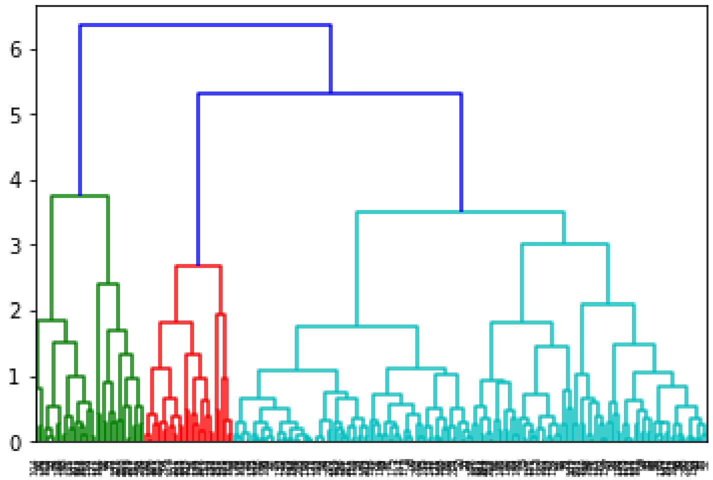


```
                scaled_weight  scaled_daydiff
cluster_labels
1                    8.271679        4.527003
2                   13.426493        1.395061
3                   10.185661        1.202330
```

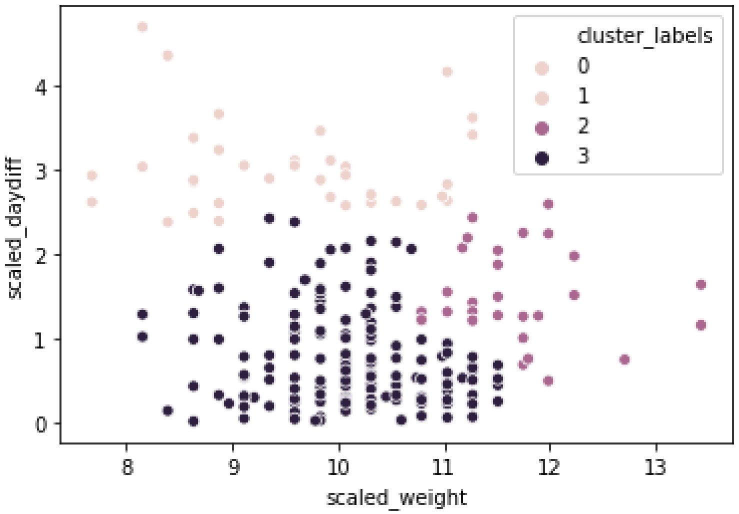


In [5]:

```
#-----Hierarchical clustering complete method------------------------------------
# Import the fcluster and linkage functions
from scipy.cluster.hierarchy import fcluster, linkage
import seaborn as sns

# Use the linkage() function
distance_matrix = linkage(df[['scaled_weight', 'scaled_daydiff']], method = 'complete', metric = 'euclidean')

from scipy.cluster.hierarchy import dendrogram

# Create a dendrogram
dn = dendrogram(distance_matrix)
plt.show()

# Assign cluster labels
df['cluster_labels'] = fcluster(distance_matrix, 3, criterion='maxclust')

# Plot clusters
sns.scatterplot(x='scaled_weight', y='scaled_daydiff', 
                hue='cluster_labels', data = df)

# Display cluster centers of each cluster
print(df[['scaled_weight', 'scaled_daydiff', 'cluster_labels']].groupby('cluster_labels').mean())

plt.show()
```



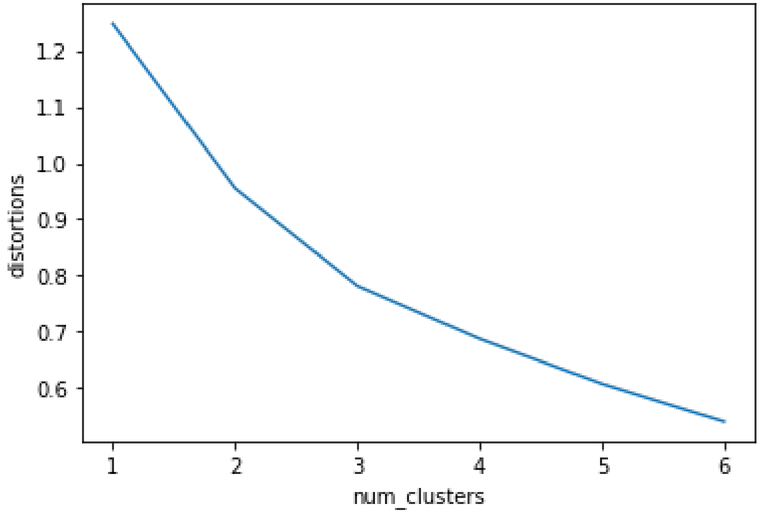
```
                scaled_weight  scaled_daydiff
cluster_labels
1                    9.594348        3.022018
2                   11.675427        1.502052
3                   10.062185        0.771779
```

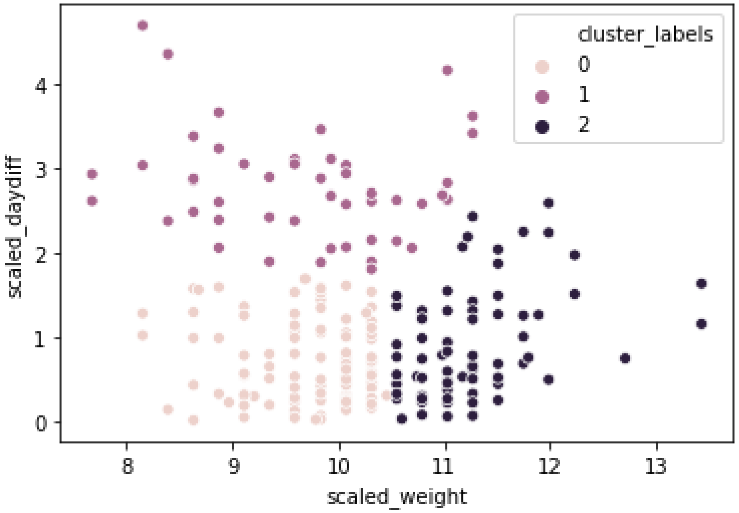


In [8]:

```python
#-----k-means clustering and elbow plot

# Import the kmeans and vq functions
from scipy.cluster.vq import kmeans, vq

distortions = []
num_clusters = range(1, 7)

# Create a list of distortions from the kmeans function
for i in num_clusters:
    cluster_centers, distortion = kmeans(df[['scaled_weight', 'scaled_daydiff']], i)
    distortions.append(distortion)

# Create a data frame with two lists - num_clusters, distortions
elbow_plot = pd.DataFrame({'num_clusters': num_clusters, 'distortions': distortions})

# Creat a line plot of num_clusters and distortions
sns.lineplot(x='num_clusters', y='distortions', data = elbow_plot)
plt.xticks(num_clusters)
plt.show()

# Generate cluster centers
cluster_centers, distortion = kmeans(df[['scaled_weight', 'scaled_daydiff']], 3)

# Assign cluster labels
```

```
df['cluster_labels'], distortion_list = vq(df[['scaled_weight', 'scaled_daydiff']], cluster_centers)

# Plot clusters
sns.scatterplot(x='scaled_weight', y='scaled_daydiff',
                hue='cluster_labels', data = df)
plt.show()
```

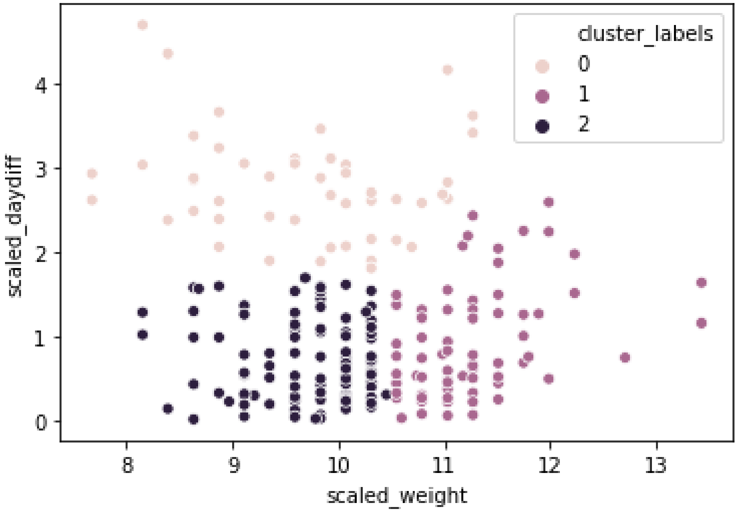




In [9]:

```
#-----experiment with random seed

from numpy import random

# Set up a random seed in numpy
random.seed([1000,2000])

# Fit the data into a k-means algorithm
cluster_centers,_ = kmeans(df[['scaled_weight', 'scaled_daydiff']], 3)

# Assign cluster labels
df['cluster_labels'], _ = vq(df[['scaled_weight','scaled_daydiff']], cluster_centers)

# Display cluster centers
print(df[['scaled_weight', 'scaled_daydiff', 'cluster_labels']].groupby('cluster_labels').mean())

# Create a scatter plot through seaborn
sns.scatterplot(x='scaled_weight', y='scaled_daydiff', hue='cluster_labels', data=df)
plt.show()
```

```
                scaled_weight  scaled_daydiff
cluster_labels
0                    9.678264        2.783786
1                   11.241075        0.929718
2                    9.685297        0.712585
```


cluster_labels
0
1
2
scaled_daydiff
0
1
2
3
4
8
9
10
11
12
13
scaled_weight