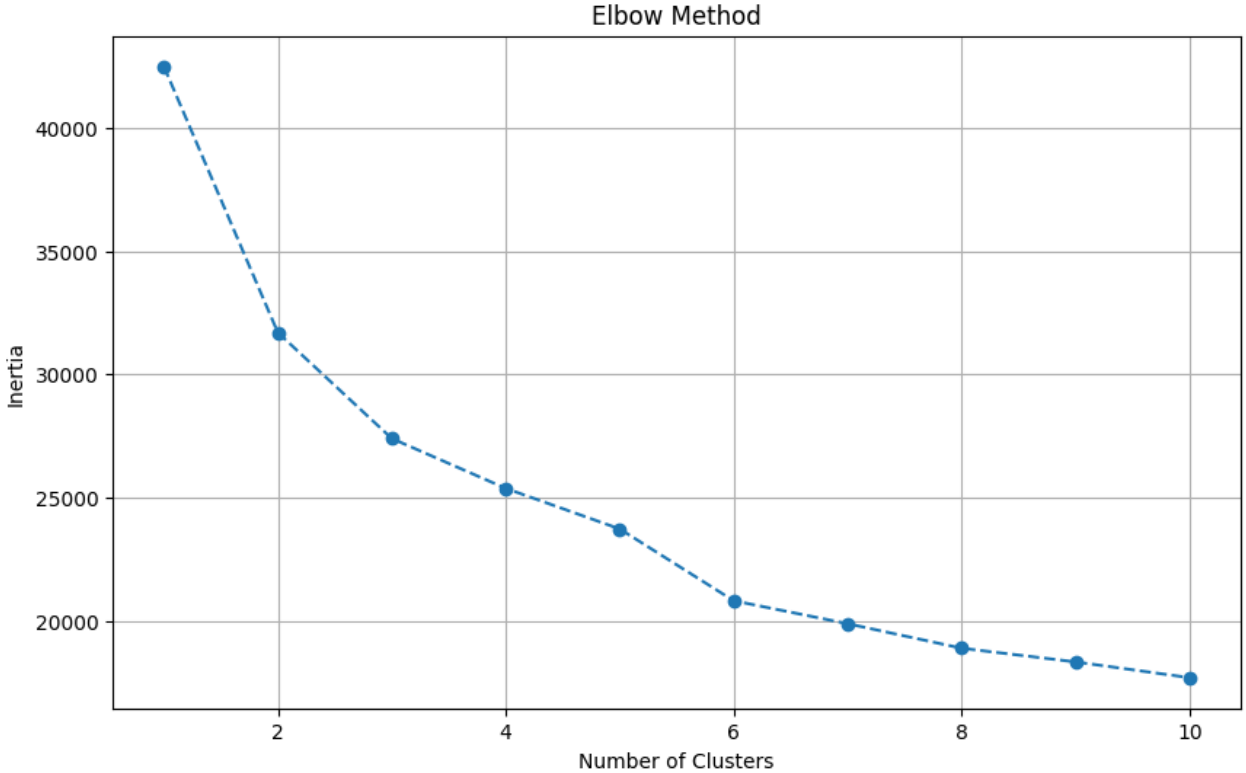
```
import pandas as pd

# Load the CSV file
file_path = "baseball_ratios.csv"
df = pd.read_csv(file_path)

#take out the categorical data
df = df.drop(columns=['year', 'team_name'])
df = df.dropna()
print(df.head())

from sklearn.preprocessing import StandardScaler

# Initialize the standard scaler
scaler = StandardScaler()

# Normalize the data
normalized_data = scaler.fit_transform(df)

normalized_df = pd.DataFrame(normalized_data, columns=df.columns)

# Display the first few rows and summary statistics of the normalized d
head_normalized = normalized_df.head()
summary_normalized = normalized_df.describe()

head_normalized, summary_normalized

from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Define a range of clusters
cluster_range = range(1, 11)

# Compute KMeans for each number of clusters
inertias = []
for k in cluster_range:
    kmeans = KMeans(n_clusters=k, random_state=42).fit(normalized_data)
    inertias.append(kmeans.inertia_)

# Plot the Elbow method
plt.figure(figsize=(10,6))
plt.plot(cluster_range, inertias, marker='o', linestyle='--')
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.grid(True)
```

```
plt.show()

#print inertias
for k, inertia in zip(cluster_range, inertias):
    print(f"Number of clusters: {k}, Inertia: {inertia:.2f}")
```

```
   rank  batting_avg  games_played  home_game_ratio  runs_scored_pg  \
0     5     0.293746           134         0.492537        5.671642   
1     2     0.278052           138         0.500000        5.500000   
2     3     0.286739           137         0.496350        5.430657   
3     5     0.248630           140         0.500000        3.792857   
4     1     0.275767           137         0.518248        5.978102   

   at_bats_pg    hits_pg  doubles_pg  triples_pg  homeruns_pg  ...  \
0   34.246269  10.059701    1.335821    0.828358     0.179104  ...   
1   35.260870   9.804348    1.326087    0.753623     0.268116  ...   
2   35.613139  10.211679    1.503650    0.678832     0.233577  ...   
3   33.900000   8.428571    0.964286    0.257143     0.200000  ...   
4   34.489051   9.510949    1.262774    0.649635     0.233577  ...   

   shutouts_pg  saves_pg  outs_pitches_pg  hits_allowed_pg  \
0     0.029851  0.022388        25.925373         9.798507   
1     0.050725  0.007246        26.456522         8.536232   
2     0.051095  0.036496        26.576642         9.080292   
3     0.078571  0.000000        27.064286         8.542857   
4     0.080292  0.014599        26.678832         9.124088   

   homeruns_allowed_pg  walks_allowed_pg  strikeouts_by_pitchers_pg  \
0             0.156716          2.567164                   2.022388   
1             0.239130          2.130435                   2.869565   
2             0.131387          3.175182                   4.255474   
3             0.207143          2.492857                   3.985714   
4             0.197080          2.277372                   2.875912   

   errors_pg  double_plays_pg  fielding_percentage_pg  
0   2.992537         0.567164                0.006910  
1   2.442029         0.753623                0.006833  
2   2.051095         0.722628                0.006934  
3   2.014286         0.635714                0.006800  
4   2.518248         0.729927                0.006869  
```

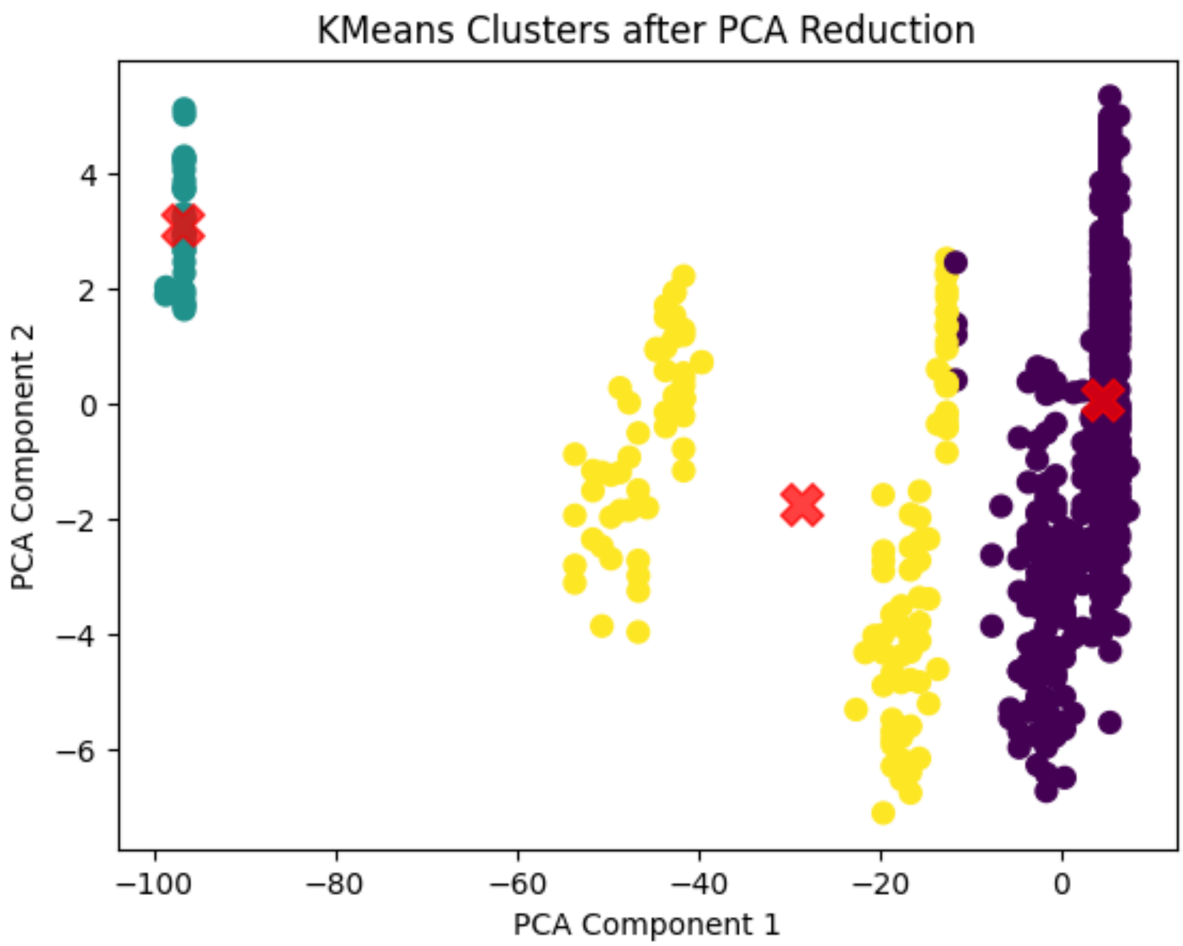
```
[5 rows x 27 columns]
```




```
Number of clusters: 1, Inertia: 42498.00
Number of clusters: 2, Inertia: 31697.26
Number of clusters: 3, Inertia: 27396.55
Number of clusters: 4, Inertia: 25372.85
Number of clusters: 5, Inertia: 23725.78
Number of clusters: 6, Inertia: 20811.97
Number of clusters: 7, Inertia: 19879.10
Number of clusters: 8, Inertia: 18883.56
Number of clusters: 9, Inertia: 18327.78
Number of clusters: 10, Inertia: 17688.58
```

```
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

X = df.values

# 1. Apply PCA to reduce to 2 components
pca = PCA(n_components=2)
```

```
X_pca = pca.fit_transform(X)

# 2. Perform KMeans k=3 from before
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X_pca)
y_kmeans = kmeans.predict(X_pca)

# 3. Plot the clustered points
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y_kmeans, cmap='viridis', s=5(
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.75,
plt.title('KMeans Clusters after PCA Reduction')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.show()

print(pca.explained_variance_ratio_)
```




```
[0.96331682 0.0167524 ]
```

```
import numpy as np
import seaborn as sns
from scipy import stats
```

```
# 1. Add cluster labels to your original dataframe
df['Cluster'] = kmeans.labels_

# 2. summary statistics for each cluster
unique_clusters = df['Cluster'].unique()

# Create a summary df with means for each feature by cluster
cluster_means = df.groupby('Cluster').mean()
cluster_medians = df.groupby('Cluster').median()
cluster_stds = df.groupby('Cluster').std()
cluster_counts = df.groupby('Cluster').size()

print("Cluster sizes:")
print(cluster_counts)
print("\nCluster means:")
print(cluster_means)

# 3. Find top 10 features with biggest differences between clusters
# Calc the var of means across clusters for each feature
feature_variance = cluster_means.var().sort_values(ascending=False)
most_distinctive_features = feature_variance.index[:10]

print("\nTop features (highest variance between clusters):")
print(feature_variance.head(10))

# Run ANOVA to find statistically significant differences
from scipy.stats import f_oneway

anova_results = {}
feature_columns = [col for col in df.columns if col != 'Cluster']

for feature in feature_columns:
    feature_by_cluster = [df[df['Cluster'] == c][feature].dropna() fo
    # Only run ANOVA if we have sufficient data in each group
    if all(len(group) > 1 for group in feature_by_cluster):
        anova_result = f_oneway(*feature_by_cluster)
        anova_results[feature] = {
            'F-statistic': anova_result.statistic,
            'p-value': anova_result.pvalue
        }

# Convert to df and sort by F-statistic
anova_df = pd.DataFrame(anova_results).T
anova_df = anova_df.sort_values('F-statistic', ascending=False)
print("\nANOVA Results (most significant differences):")
print(anova_df.head(10))
```

```
dtype: int64

Cluster means:
             rank  batting_avg  games_played  home_game_ratio  runs_s
Cluster
0        3.389986     0.258440    161.187588         0.499947
1        2.933333     0.244196     59.866667         0.499923
2        3.682540     0.264770    127.920635         0.500033

         at_bats_pg   hits_pg  doubles_pg  triples_pg  homeruns_pg  .
Cluster                                                             .
0         34.018116  8.797347    1.596714    0.220030     0.878615  .
1         32.864272  8.038391    1.571226    0.134253     1.282165  .
2         34.245280  9.073526    1.521404    0.329802     0.616704  .

         shutouts_pg  saves_pg  outs_pitches_pg  hits_allowed_pg  \
Cluster
0           0.063912  0.221949        26.815127         8.795287
1           0.006667  0.234904        25.836590         8.038659
2           0.061725  0.152523        26.722419         9.085837

         homeruns_allowed_pg  walks_allowed_pg  strikeouts_by_pitcher
Cluster
0                   0.878630          3.218613                   6.10
1                   1.283065          3.392031                   8.67
2                   0.616873          3.024662                   4.77

         errors_pg  double_plays_pg  fielding_percentage_pg
Cluster
0         0.806322         0.898966                0.006075
1         0.577261         0.797318                0.016430
2         1.316525         0.847623                0.007685

[3 rows x 27 columns]

Top features (highest variance between clusters):
games_played                 2667.326984
strikeouts_by_pitchers_pg       3.931369
strikeouts_by_batters_pg        3.911271
at_bats_pg                      0.548357
hits_allowed_pg                 0.292248
outs_pitches_pg                 0.291804
hits_pg                         0.287299
errors_pg                       0.143215
rank                            0.142572
homeruns_allowed_pg             0.112649
dtype: float64

ANOVA Results (most significant differences):
                          F-statistic       p-value
fielding_percentage_pg   18736.816379  0.000000e+00
games_played              8871.794288  0.000000e+00
outs_pitches_pg            181.691770  1.040376e-71
```

```
errors_pg                       114.747947  3.061054e-47
strikeouts_by_batters_pg        111.933802  3.580371e-46
strikeouts_by_pitchers_pg       111.346058  5.989883e-46
triples_pg                       83.404598  3.756219e-35
complete_games_pg                83.041942  5.213891e-35
homeruns_allowed_pg              67.327303  9.681534e-29
```

```
df['PCA1'] = X_pca[:, 0]
df['PCA2'] = X_pca[:, 1]

# Export to CSV
output_file_path = "baseball_clusters.csv"
df.to_csv(output_file_path, index=False)

print(f"Data exported to {output_file_path}")
```

```
Data exported to baseball_clusters.csv
```