In [1]:
```
#Caden Zadell
#BSAN 360
#Lab 4
```

In [2]:
```
import pandas as pd
import numpy as np
```

In [3]:
```
#1
df = pd.read_csv("baseball_hitting.csv")
df.head()
```

Out[3]:

	Player name	position	Games	At-bat	Runs	Hits	Double (2B)	third baseman	hom ru
0	B Bonds	LF	2986.0	9847.0	2227.0	2935.0	601.0	77.0	762
1	H Aaron	RF	3298.0	12364.0	2174.0	3771.0	624.0	98.0	755
2	B Ruth	RF	2504.0	8399.0	2174.0	2873.0	506.0	136.0	714
3	A Pujols	1B	3080.0	11421.0	1914.0	3384.0	686.0	16.0	703
4	A Rodriguez	SS	2784.0	10566.0	2021.0	3115.0	548.0	31.0	696

In [4]:
```
#2
print("Rows:", df.shape[0])
print("Columns:", df.shape[1])
```

```
Rows: 2508
Columns: 18
```

In [5]:
```
#2
# This helps confirm that the dataset loaded correctly.
# The expected dimensions depend on the data source
```

In [5]:
```
#3
print("First five rows:")
display(df.head())

print("\nLast five rows:")
display(df.tail())
```

```
First five rows:
```

	Player name	position	Games	At-bat	Runs	Hits	Double (2B)	third baseman	home run
0	B Bonds	LF	2986.0	9847.0	2227.0	2935.0	601.0	77.0	762.0
1	H Aaron	RF	3298.0	12364.0	2174.0	3771.0	624.0	98.0	755.0
2	B Ruth	RF	2504.0	8399.0	2174.0	2873.0	506.0	136.0	714.0
3	A Pujols	1B	3080.0	11421.0	1914.0	3384.0	686.0	16.0	703.0
4	A Rodriguez	SS	2784.0	10566.0	2021.0	3115.0	548.0	31.0	696.0

```
Last five rows:
```

	Player name	position	Games	At-bat	Runs	Hits	Double (2B)	third baseman	h
2503	R Derry	LF	187.0	553.0	68.0	124.0	17.0	7.0	
2504	M Kittridge	C	470.0	4027.0	375.0	882.0	108.0	31.0	
2505	G DeMontreville	2B	280.0	3615.0	537.0	1096.0	130.0	35.0	
2506	L Niekro	1B	195.0	499.0	61.0	123.0	26.0	5.0	
2507	B Whitehead	2B	924.0	3316.0	415.0	883.0	100.0	31.0	

In [7]:
```
#3
# Confirm that all rows are aligned, headers are correct,
# and that numeric columns are in the right format (no misaligned or m
```

In [6]:
```
#4
print(df.columns)
```

```
Index(['Player name', 'position', 'Games', 'At-bat', 'Runs', 'Hits',
       'Double (2B)', 'third baseman', 'home run', 'run batted in', 'a
walk',
       'Strikeouts', 'stolen base ', 'Caught stealing', 'AVG',
       'On-base Percentage', 'Slugging Percentage', 'On-base Plus Slugg
ing'],
      dtype='object')
```

In [7]:
```
#4
df = df.rename(columns={
    "a walk": "Walked",
    "third baseman": "Triples",
    "home run": "Home Runs",
    "stolen base": "Stolen Bases",
    "run batted in": "RBIs",
```

```
    "Double (2B)": "Doubles",
    "stolen base ": "Steals",
    "position": "Position"
})
```

In [8]:
```
print(df.columns)
```

```
Index(['Player name', 'Position', 'Games', 'At-bat', 'Runs', 'Hits', 'D
oubles',
       'Triples', 'Home Runs', 'RBIs', 'Walked', 'Strikeouts', 'Steal
s',
       'Caught stealing', 'AVG', 'On-base Percentage', 'Slugging Percen
tage',
       'On-base Plus Slugging'],
      dtype='object')
```

In [24]:
```
#4
# Renamed columns for uniformity and to be more clear on what stat is
```

In [26]:
```
# 5a. What information is provided in this dataset?
# Each row represents a player's batting statistics for a given season
# The columns include both basic (Runs, Hits, Home Runs) and advanced
# metrics (On-base Percentage, Slugging Percentage, OPS).
# These stats reflect offensive performance across a variety of metric
```

In [9]:
```
print(df.dtypes)
```

```
Player name                 object
Position                    object
Games                      float64
At-bat                     float64
Runs                       float64
Hits                       float64
Doubles                    float64
Triples                    float64
Home Runs                  float64
RBIs                       float64
Walked                     float64
Strikeouts                  object
Steals                     float64
Caught stealing             object
AVG                        float64
On-base Percentage         float64
Slugging Percentage        float64
On-base Plus Slugging      float64
dtype: object
```

In [28]:
```
#5b
# Most columns are correctly stored as numeric (float64), which is ide
# 'Strikeouts' and 'Caught stealing' are objects — they should likely
# so these columns may contain commas, text, or missing characters tha
# pandas from recognizing them as numbers.
```

In [10]:
```
#5c
df.describe
```

Out[10]:
```
<bound method NDFrame.describe of            Player name Position   Ga
mes   At-bat    Runs    Hits  Doubles  \
0              B Bonds       LF  2986.0   9847.0  2227.0  2935.0    60
1.0
1              H Aaron       RF  3298.0  12364.0  2174.0  3771.0    62
4.0
2               B Ruth       RF  2504.0   8399.0  2174.0  2873.0    50
6.0
3             A Pujols       1B  3080.0  11421.0  1914.0  3384.0    68
6.0
4          A Rodriguez       SS  2784.0  10566.0  2021.0  3115.0    54
8.0
...                ...      ...     ...      ...     ...     ...
...
2503           R Derry       LF   187.0    553.0    68.0   124.0     1
7.0
2504      M Kittridge         C   470.0   4027.0   375.0   882.0    10
8.0
2505  G DeMontreville       2B   280.0   3615.0   537.0  1096.0    13
0.0
2506          L Niekro       1B   195.0    499.0    61.0   123.0     2
6.0
2507      B Whitehead       2B   924.0   3316.0   415.0   883.0    10
0.0

      Triples  Home Runs    RBIs  Walked Strikeouts  Steals Caught st
ealing  \
0        77.0      762.0  1996.0  2558.0       1539   514.0
141
1        98.0      755.0  2297.0  1402.0       1383   240.0
73
2       136.0      714.0  2213.0  2062.0       1330   123.0
117
3        16.0      703.0  2218.0  1373.0       1404   117.0
43
4        31.0      696.0  2086.0  1338.0       2287   329.0
76
...       ...        ...     ...     ...        ...     ...
...
2503      7.0       17.0    73.0    78.0        124     2.0
0
2504     31.0       17.0   390.0   314.0        166    64.0
--
2505     35.0       17.0   497.0   174.0         35   228.0
--
2506      5.0       17.0    79.0    29.0         91     0.0
2
2507     31.0       17.0   245.0   150.0        138    51.0
--
```

```
        AVG  On-base Percentage  Slugging Percentage  On-base Plus Sl
ugging
0     0.298               0.444                0.607
1.051
1     0.305               0.374                0.555
0.929
2     0.342               0.474                0.690
1.164
3     0.296               0.374                0.544
0.918
4     0.295               0.380                0.550
0.930
...     ...                 ...                  ...
...
2503  0.224               0.322                0.373
0.695
2504  0.219               0.277                0.274
0.551
2505  0.303               0.340                0.373
0.713
2506  0.246               0.288                0.421
0.709
2507  0.266               0.304                0.331
0.635

[2508 rows x 18 columns]>
```

In [11]:

```python
#5c
numeric_df = df.select_dtypes(include=["float64", "int64"])
min_values = numeric_df.min()
max_values = numeric_df.max()
mean_values = numeric_df.mean()


print("Minimum values:\n", min_values)
print("\nMaximum values:\n", max_values)
print("\nMean values:\n", mean_values)
```

```
Minimum values:
 Games                         2.000
At-bat                       262.000
Runs                          32.000
Hits                          57.000
Doubles                        7.000
Triples                        0.000
Home Runs                     17.000
RBIs                          37.000
Walked                        19.000
Steals                         0.000
AVG                            0.123
On-base Percentage             0.157
Slugging Percentage            0.197
On-base Plus Slugging          0.354
dtype: float64

Maximum values:
 Games                      3562.000
At-bat                     14053.000
Runs                        2295.000
Hits                        4256.000
Doubles                      792.000
Triples                      309.000
Home Runs                    762.000
RBIs                        2297.000
Walked                      2558.000
Steals                      1406.000
AVG                            0.367
On-base Percentage             0.482
Slugging Percentage            0.690
On-base Plus Slugging          1.164
dtype: float64

Mean values:
 Games                      1084.558000
At-bat                     3714.962000
Runs                        521.644800
Hits                       1010.865600
Doubles                     181.858000
Triples                      32.330800
Home Runs                   100.611600
RBIs                        494.206400
Walked                      373.038000
Steals                       76.095200
AVG                           0.263320
On-base Percentage            0.331582
Slugging Percentage           0.409925
On-base Plus Slugging         0.741695
dtype: float64
```

In [35]:

```
#5c
# Games: min = 2, max = 3,562
```

```
# Represents very short and very long careers (2 = brief career; 3,562
#record length, like Pete Rose).
# Reasonable.

# At-bat: min = 262, max = 14,053
# Fits realistic career totals (around 10–14k = lifetime leaders such
#Pete Rose).
#Reasonable.

# Runs: min = 32, max = 2,295
# Matches all-time leaders (Rickey Henderson = 2,295).
# Reasonable.

# Hits: min = 57, max = 4,256
# 4,256 = Pete Rose’s exact hit total, confirming data validity.
# Reasonable.

# Doubles: min = 7, max = 792
# 792 ≈ Tris Speaker’s career record. Reasonable.

# Triples: min = 0, max = 309
# 309 ≈ Sam Crawford’s record. Reasonable.

# Home Runs: min = 17, max = 762
# 762 = Barry Bonds’ record. Fits perfectly. Reasonable.

# All the data ranges appear reasonable based on the min and max value
```

In [12]:

```
#5d
print(df.isna().sum())
```

```
Player name              8
Position                 8
Games                    8
At-bat                   8
Runs                     8
Hits                     8
Doubles                  8
Triples                  8
Home Runs                8
RBIs                     8
Walked                   8
Strikeouts               8
Steals                   8
Caught stealing          8
AVG                      8
On-base Percentage       8
Slugging Percentage      8
On-base Plus Slugging   20
dtype: int64
```

In [44]:

```
#5d
# Since the value's are missing, we can double check to see if the pla
```

```
# that stat and input a 0.
# If its data entry issue, since the values are low we can delete thos
# players wouldn't be apart of the question we are trying to answer an
```

In [ ]:
```
#6

# Research Question 1:
# Who had the best offensive prime (highest AVG, OBP, SLG, OPS)?
# Columns: ['Player name', 'AVG', 'On-base Percentage', 'Slugging Perc


# Research Question 2:
# 2. Which players produced the most runs during their best seasons?
# Columns: ['Player name', 'Home Runs', 'RBIs', 'Runs']
```

In [13]:
```
#7
prime_stats = df[['Player name', 'Position', 'AVG', 'On-base Percentag
                  'Slugging Percentage', 'On-base Plus Slugging',
                  'Home Runs', 'RBIs', 'Hits', 'Steals', 'On-base Plus
prime_stats.head()
```

Out[13]:

	Player name	Position	AVG	On-base Percentage	Slugging Percentage	On-base Plus Slugging	Home Runs	RBIs
0	B Bonds	LF	0.298	0.444	0.607	1.051	762.0	1996.0
1	H Aaron	RF	0.305	0.374	0.555	0.929	755.0	2297.0
2	B Ruth	RF	0.342	0.474	0.690	1.164	714.0	2213.0
3	A Pujols	1B	0.296	0.374	0.544	0.918	703.0	2218.0
4	A Rodriguez	SS	0.295	0.380	0.550	0.930	696.0	2086.0

In [ ]:
```
#7
# This DataFrame isolates the core stats that define a player's offens
# It will be used to evaluate players using summary statistics, scorin
```

In [ ]:
```
#8
# Who had the best offensive prime (highest AVG, OBP, SLG, OPS)?
# Method: Ranking analysis
# Compute top 10 players by On-base Plus Slugging (OPS)
# Calculate z-scores or percentiles to normalize across eras.

# Research Question 2:
# Which players generated the most runs during their peak?
# Method: Correlation and ranking
# Use correlation matrix to test relationship between HRs, RBIs, Runs,
# Rank players based on combined weighted score.
```

In [ ]:
```
#Caden Zadell
#Project 2 Assignment Work Starts Here
```

In [7]:
```
#Data Cleaning and Prep
```

In [ ]:
```
```

In [14]:
```
print(df.head())
```

```
   Player name Position   Games   At-bat    Runs    Hits  Doubles  Trip
les  \
0      B Bonds       LF  2986.0   9847.0  2227.0  2935.0    601.0     7
7.0
1      H Aaron       RF  3298.0  12364.0  2174.0  3771.0    624.0     9
8.0
2       B Ruth       RF  2504.0   8399.0  2174.0  2873.0    506.0    13
6.0
3     A Pujols       1B  3080.0  11421.0  1914.0  3384.0    686.0     1
6.0
4  A Rodriguez       SS  2784.0  10566.0  2021.0  3115.0    548.0     3
1.0

   Home Runs    RBIs  Walked Strikeouts  Steals Caught stealing    AVG
\
0      762.0  1996.0  2558.0       1539   514.0             141  0.298
1      755.0  2297.0  1402.0       1383   240.0              73  0.305
2      714.0  2213.0  2062.0       1330   123.0             117  0.342
3      703.0  2218.0  1373.0       1404   117.0              43  0.296
4      696.0  2086.0  1338.0       2287   329.0              76  0.295

   On-base Percentage  Slugging Percentage  On-base Plus Slugging
0               0.444                0.607                  1.051
1               0.374                0.555                  0.929
2               0.474                0.690                  1.164
3               0.374                0.544                  0.918
4               0.380                0.550                  0.930
```

In [15]:
```
df_clean = df[['Player name', 'AVG', 'On-base Percentage', 'Slugging P
               'On-base Plus Slugging', 'Home Runs', 'RBIs', 'Runs', '
```

In [16]:
```
print(df_clean.isnull().sum())
```

```
Player name               8
AVG                       8
On-base Percentage        8
Slugging Percentage       8
On-base Plus Slugging    20
Home Runs                 8
RBIs                      8
Runs                      8
Hits                      8
Steals                    8
dtype: int64
```

In [17]:
```
print(df_clean.dtypes)
```

```
Player name               object
AVG                      float64
On-base Percentage       float64
Slugging Percentage      float64
On-base Plus Slugging    float64
Home Runs                float64
RBIs                     float64
Runs                     float64
Hits                     float64
Steals                   float64
dtype: object
```

In [38]:
```
#Missing Data
```

In [18]:
```
df_clean = df_clean.dropna(subset=['AVG', 'On-base Percentage', 'Slugg

df_clean['Steals'] = df_clean['Steals'].fillna(0)

print("Missing values:")
print(df_clean.isnull().sum())


print("\nRows after cleaning:", len(df_clean))
```

```
Missing values:
Player name              0
AVG                      0
On-base Percentage       0
Slugging Percentage      0
On-base Plus Slugging    0
Home Runs                0
RBIs                     0
Runs                     0
Hits                     0
Steals                   0
dtype: int64

Rows after cleaning: 2488
```

In [19]:
```
print(df_clean.shape)
```

```
(2488, 10)
```

In [42]:
```
#Data Transformation
```

In [20]:
```
from sklearn.preprocessing import StandardScaler
```

In [21]:
```
numeric_cols = ['AVG', 'On-base Percentage', 'Slugging Percentage',
                'On-base Plus Slugging', 'Home Runs', 'RBIs', 'Runs',
```

In [22]:
```
scaler = StandardScaler()
```

In [23]:
```
scaled_data = scaler.fit_transform(df_clean[numeric_cols])
scaled_df = pd.DataFrame(scaled_data, columns=numeric_cols)
```

In [24]:
```
scaled_df['Player name'] = df_clean['Player name'].values
```

In [25]:
```
print(scaled_df.head())
```

```
        AVG  On-base Percentage  Slugging Percentage  On-base Plus Slug
ging  \
0  1.402200            3.660724             3.933776               4.29
9191
1  1.684847            1.381075             2.894876               2.60
3450
2  3.178838            4.637717             5.592020               5.86
9837
3  1.321443            1.381075             2.675109               2.45
0555
4  1.281065            1.576474             2.794982               2.61
7350

   Home Runs      RBIs      Runs      Hits    Steals  Player name
0   6.604641  4.133466  4.481532  2.819983  3.852164      B Bonds
1   6.534704  4.961928  4.342289  4.044516  1.444671      H Aaron
2   6.125074  4.730729  4.342289  2.729168  0.416654       B Ruth
3   6.015173  4.744491  3.659207  3.477657  0.363936     A Pujols
4   5.945236  4.381179  3.940322  3.083638  2.226667  A Rodriguez
```

In [26]:
```
print(scaled_df[numeric_cols].mean())
print(scaled_df[numeric_cols].std())
```

```
AVG                       -3.655525e-16
On-base Percentage         1.005269e-15
Slugging Percentage        3.655525e-16
On-base Plus Slugging      6.854110e-16
Home Runs                 -9.138813e-17
RBIs                       0.000000e+00
Runs                      -1.827763e-16
Hits                      -4.569407e-17
Steals                    -1.142352e-17
dtype: float64
AVG                        1.000201
On-base Percentage         1.000201
Slugging Percentage        1.000201
On-base Plus Slugging      1.000201
Home Runs                  1.000201
RBIs                       1.000201
Runs                       1.000201
Hits                       1.000201
Steals                     1.000201
dtype: float64
```

In [27]:

```
#Caden Zadell
#Project 3 Assignment Starts Here
import pandas as pd

df = pd.read_csv("baseball_hitting.csv")

df.columns = df.columns.str.strip()

df = df.rename(columns={
    "position": "Position",
    "Double (2B)": "Doubles",
    "third baseman": "Triples",
    "home run": "Home Runs",
    "run batted in": "RBIs",
    "a walk": "Walked",
    "stolen base ": "Steals",   #
    "stolen base": "Steals"
})


df = df.dropna(subset=["Player name", "AVG", "On-base Percentage", "Sl


df["On-base Plus Slugging"] = df["On-base Percentage"] + df["Slugging


df = df[df["Games"] >= 30]

df.to_csv("cleaned_baseball_data.csv", index=False)

print("Cleaned data shape:", df.shape)
print(df.columns)
```

```
Cleaned data shape: (2496, 18)
Index(['Player name', 'Position', 'Games', 'At-bat', 'Runs', 'Hits', 'D
oubles',
       'Triples', 'Home Runs', 'RBIs', 'Walked', 'Strikeouts', 'Steal
s',
       'Caught stealing', 'AVG', 'On-base Percentage', 'Slugging Percen
tage',
       'On-base Plus Slugging'],
      dtype='object')
```

In [28]:
```
#Professor Feedback
df = df.dropna(subset=["Player name"])

df["On-base Plus Slugging"] = df["On-base Percentage"] + df["Slugging

print(df["On-base Plus Slugging"].isna().sum())

df = df[df["Games"] >= 30]

print(len(df))
```

```
0
2496
```

In [29]:
```
#Processing Strings
df.columns = df.columns.str.strip()

df["Player name"] = df["Player name"].str.strip()

print(df["Player name"].head())
```

```
0         B Bonds
1         H Aaron
2          B Ruth
3        A Pujols
4    A Rodriguez
Name: Player name, dtype: object
```

In [ ]:
```
#Combining and Merging Datasets
#Only have one dataset for this project.
#No merging or combining is needed
```

In [31]:
```
#Reshaping and Pivoting

pivot_position = df.pivot_table(
    values=["AVG", "On-base Percentage", "Slugging Percentage", "On-ba
    index= "Position",
    aggfunc= "mean"
).round(3)

print("Average Offensive Performance by Position:")
print(pivot_position)
```

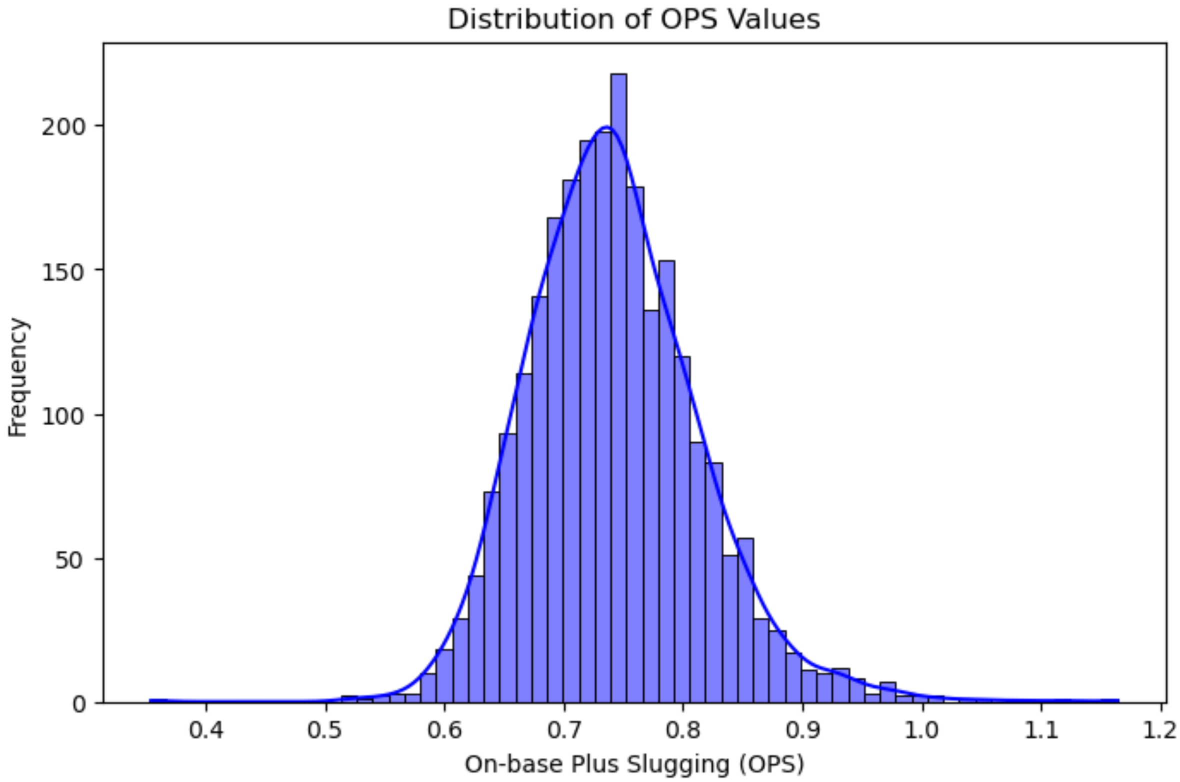
```
Average Offensive Performance by Position:
            AVG  On-base Percentage  On-base Plus Slugging  \
Position
1B        0.268               0.342                  0.777
2B        0.263               0.327                  0.711
3B        0.262               0.328                  0.736
C         0.249               0.319                  0.706
CF        0.267               0.335                  0.744
DH        0.270               0.350                  0.810
LF        0.269               0.338                  0.767
OF        0.268               0.336                  0.751
P         0.221               0.279                  0.619
RF        0.269               0.339                  0.771
SS        0.260               0.319                  0.697

          Slugging Percentage
Position
1B                      0.435
2B                      0.383
3B                      0.408
C                       0.387
CF                      0.409
DH                      0.460
LF                      0.428
OF                      0.415
P                       0.340
RF                      0.432
SS                      0.378
```

In [32]:
```
df_clean = df.copy()
df_clean.to_csv("cleaned_baseball_data.csv", index=False)
```

In [1]:
```
#Caden Zadell
#BSAN
#Project 4 Assignment Starts Here
```

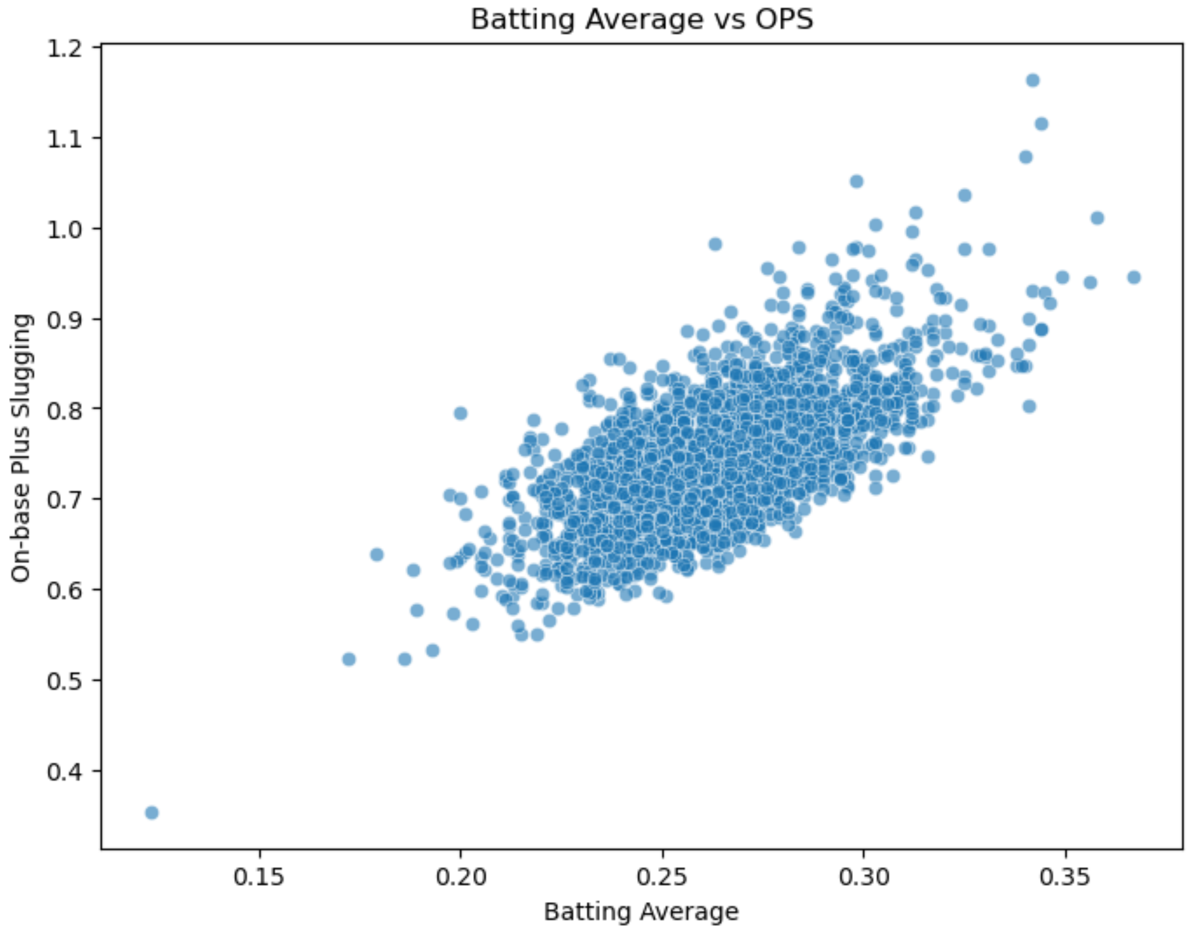
In [33]:
```
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv("cleaned_baseball_data.csv")
```

In [34]:
```
#1 Distribution of OPS

plt.figure(figsize=(8, 5))
sns.histplot(df["On-base Plus Slugging"], kde=True, color="blue")
plt.title("Distribution of OPS Values")
plt.xlabel("On-base Plus Slugging (OPS)")
plt.ylabel("Frequency")
plt.show()
```

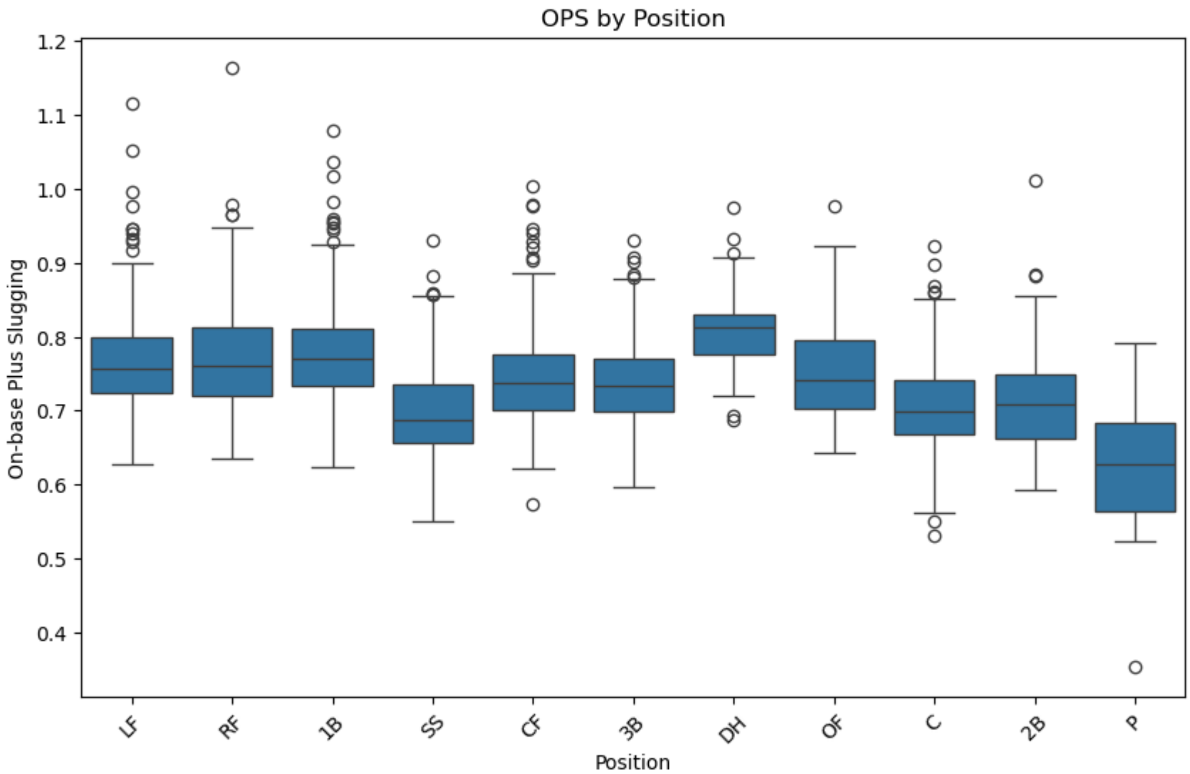


In [35]:

```
#2 AVG vs OPS Scatter Plot

plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x="AVG", y="On-base Plus Slugging", alpha=0.6
plt.title("Batting Average vs OPS")
plt.xlabel("Batting Average")
plt.ylabel("On-base Plus Slugging")
plt.show()
```



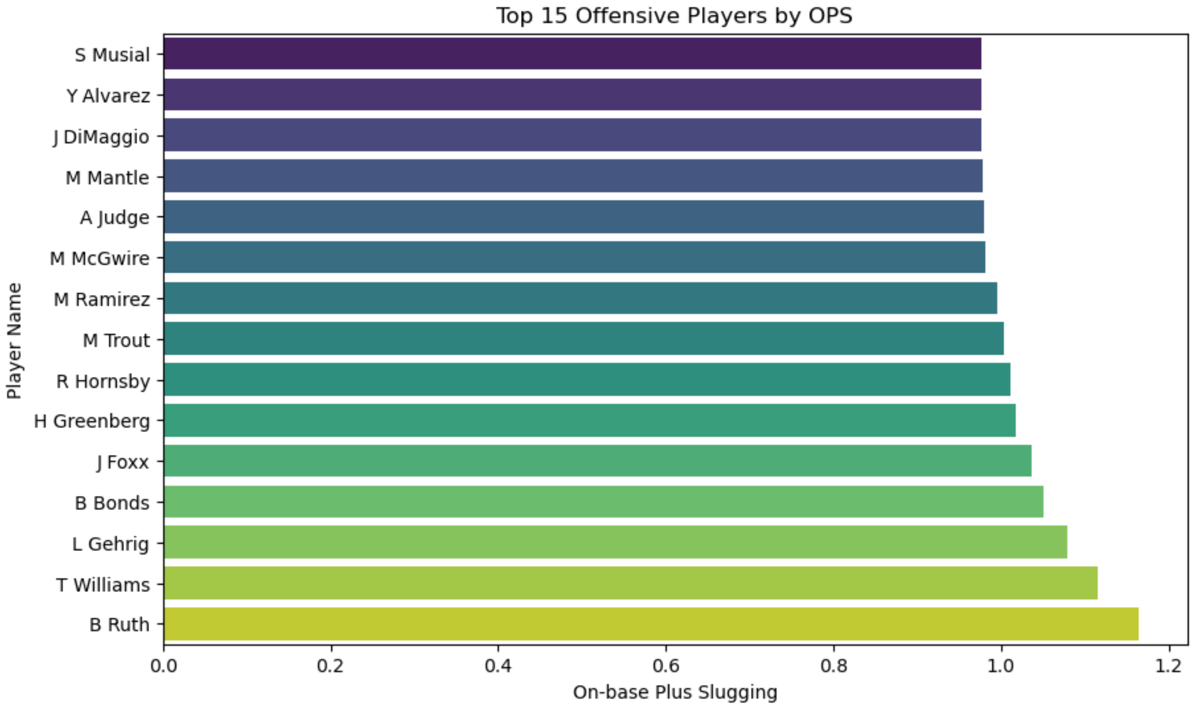
In [36]:
```
# 3. Boxplot of OPS by Position

plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x="Position", y="On-base Plus Slugging")
plt.xticks(rotation=45)
plt.title("OPS by Position")
plt.xlabel("Position")
plt.ylabel("On-base Plus Slugging")
plt.show()
```



In [37]:

```
#4 Top 15 Players by OPS

top_ops = df.nlargest(15, "On-base Plus Slugging")[["Player name", "On
plt.figure(figsize=(10, 6))
sns.barplot(data=top_ops, x="On-base Plus Slugging", y="Player name",
plt.title("Top 15 Offensive Players by OPS")
plt.xlabel("On-base Plus Slugging")
plt.ylabel("Player Name")
plt.show()
```

```
/var/folders/x9/5gqcqk055lb1bfdx10rnvrcw0000gn/T/ipykernel_3177/2071071
647.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be rem
oved in v0.14.0. Assign the `y` variable to `hue` and set `legend=False
` for the same effect.

  sns.barplot(data=top_ops, x="On-base Plus Slugging", y="Player name",
palette="viridis")
```



In [38]:
```
#Caden Zadell
#BSAN 360
#Project 5 Assignment
import pandas as pd

df = pd.read_csv("cleaned_baseball_data.csv")
print(df.head())
```

```
   Player name Position   Games    At-bat    Runs    Hits  Doubles  Trip
les  \
0      B Bonds       LF  2986.0   9847.0  2227.0  2935.0    601.0     7
7.0
1      H Aaron       RF  3298.0  12364.0  2174.0  3771.0    624.0     9
8.0
2       B Ruth       RF  2504.0   8399.0  2174.0  2873.0    506.0    13
6.0
3     A Pujols       1B  3080.0  11421.0  1914.0  3384.0    686.0     1
6.0
4  A Rodriguez       SS  2784.0  10566.0  2021.0  3115.0    548.0     3
1.0

   Home Runs    RBIs  Walked Strikeouts  Steals Caught stealing    AVG
\
0      762.0  1996.0  2558.0       1539   514.0             141  0.298
1      755.0  2297.0  1402.0       1383   240.0              73  0.305
2      714.0  2213.0  2062.0       1330   123.0             117  0.342
3      703.0  2218.0  1373.0       1404   117.0              43  0.296
4      696.0  2086.0  1338.0       2287   329.0              76  0.295

   On-base Percentage  Slugging Percentage  On-base Plus Slugging
0               0.444                0.607                  1.051
1               0.374                0.555                  0.929
2               0.474                0.690                  1.164
3               0.374                0.544                  0.918
4               0.380                0.550                  0.930
```

In [2]:

```
#New Research Question
#Do naturally occurring clusters of MLB players reveal different offen
#(power hitters, contact hitters, balanced hitters),
#and which statistics most define each group?
```

In [3]:

```
#Data Aggregation
```

In [39]:

```
numeric_cols = [
    "AVG", "On-base Percentage", "Slugging Percentage",
    "On-base Plus Slugging", "Home Runs", "RBIs",
    "Runs", "Hits", "Steals"
]

overall_summary = df[numeric_cols].agg(["mean", "median", "min", "max"

print("Overall summary of key offensive stats:")
print(overall_summary)
```

```
Overall summary of key offensive stats:
          AVG  On-base Percentage  Slugging Percentage  On-base Plus Sl
ugging  \
mean    0.263               0.332                0.410
0.741
median  0.262               0.330                0.407
0.737
min     0.123               0.157                0.197
0.354
max     0.367               0.482                0.690
1.164
std     0.025               0.031                0.050
0.072

        Home Runs      RBIs      Runs      Hits    Steals
mean      100.698   493.929   520.849  1010.128    75.701
median     69.000   404.000   423.000   853.000    32.000
min        17.000    37.000    32.000    57.000     0.000
max       762.000  2297.000  2295.000  4256.000  1406.000
std       100.032   362.475   379.204   681.278   113.160
```

In [40]:
```
#Data grouping by position
position_summary = (
    df.groupby("Position")
      .agg({
          "On-base Plus Slugging": ["mean", "median", "max"],
          "Home Runs": ["mean", "max"],
          "RBIs": "mean",
          "Player name": "count"
      })
      .round(3)
)
```

In [41]:
```
position_summary = position_summary.rename(columns={"Player name": "Pl

print("Offensive summary by position:")
print(position_summary)
```

```
Offensive summary by position:
         On-base Plus Slugging               Home Runs          RBIs  \
                          mean median    max       mean    max     mean
Position
1B                       0.777  0.771  1.079    131.099  703.0  589.958
2B                       0.711  0.709  1.011     70.487  377.0  448.857
3B                       0.736  0.734  0.930    102.859  548.0  501.757
C                        0.706  0.698  0.922     76.521  427.0  366.743
CF                       0.744  0.738  1.003     97.263  660.0  488.737
DH                       0.810  0.813  0.974    193.486  541.0  710.286
LF                       0.767  0.756  1.116    109.405  762.0  497.060
OF                       0.751  0.741  0.976     90.947  475.0  488.120
P                        0.619  0.627  0.791     26.786   61.0  169.143
RF                       0.771  0.761  1.164    122.700  755.0  538.672
SS                       0.697  0.688  0.930     78.978  696.0  510.780

         Player count
                count
Position
1B                354
2B                273
3B                284
C                 338
CF                274
DH                 35
LF                333
OF                 75
P                  14
RF                293
SS                223
```

In [9]:

```python
#Grouping by OPS
```

In [42]:

```python
ops_bins = [0.0, 0.700, 0.800, 0.900, 1.200]
ops_labels = [
    "Below Avg (<.700)",
    "Above Avg (.700-.800)",
    "Great (.800-.900)",
    "Elite (>.900)"
]

df["OPS_tier"] = pd.cut(
    df["On-base Plus Slugging"],
    bins=ops_bins,
    labels=ops_labels,
    include_lowest=True
)

tier_counts = df.groupby("OPS_tier")["Player name"].count()
```

```
/var/folders/x9/5gqcqk055lb1bfdx10rnvrcw0000gn/T/ipykernel_3177/2228173
739.py:16: FutureWarning: The default of observed=False is deprecated a
nd will be changed to True in a future version of pandas. Pass observed
=False to retain current behavior or observed=True to adopt the future
default and silence this warning.
  tier_counts = df.groupby("OPS_tier")["Player name"].count()
```

In [43]:
```
print("\nNumber of players in each OPS tier:")
print(tier_counts)
```

```
Number of players in each OPS tier:
OPS_tier
Below Avg (<.700)          714
Above Avg (.700-.800)     1320
Great (.800-.900)          402
Elite (>.900)               60
Name: Player name, dtype: int64
```

In [13]:
```
#Pivot Table
```

In [44]:
```
pivot_ops_mean = pd.pivot_table(
    df,
    values="On-base Plus Slugging",
    index="Position",
    columns="OPS_tier",
    aggfunc="mean"
).round(3)
```

```
/var/folders/x9/5gqcqk055lb1bfdx10rnvrcw0000gn/T/ipykernel_3177/1064842
689.py:1: FutureWarning: The default value of observed=False is depreca
ted and will change to observed=True in a future version of pandas. Spe
cify observed=False to silence this warning and retain the current beha
vior
  pivot_ops_mean = pd.pivot_table(
```

In [45]:
```
print("Average OPS by position and OPS tier:")
print(pivot_ops_mean)
```

```
Average OPS by position and OPS tier:
OPS_tier  Below Avg (<.700)  Above Avg (.700-.800)  Great (.800-.900)
\
Position
1B                    0.674                  0.754              0.840
2B                    0.657                  0.744              0.837
3B                    0.667                  0.744              0.831
C                     0.660                  0.741              0.830
CF                    0.668                  0.746              0.835
DH                    0.690                  0.768              0.828
LF                    0.678                  0.749              0.835
OF                    0.676                  0.745              0.825
P                     0.592                  0.781                NaN
RF                    0.672                  0.746              0.832
SS                    0.654                  0.740              0.841

OPS_tier  Elite (>.900)
Position
1B                0.959
2B                1.011
3B                0.913
C                 0.922
CF                0.945
DH                0.932
LF                0.975
OF                0.950
P                   NaN
RF                0.949
SS                0.930
```

In [46]:
```
ops_position_crosstab = pd.crosstab(df["Position"], df["OPS_tier"])

print("\nCross-tab of number of players by position and OPS tier:")
print(ops_position_crosstab)
```

```
Cross-tab of number of players by position and OPS tier:
OPS_tier  Below Avg (<.700)  Above Avg (.700-.800)  Great (.800-.900)
\
Position
1B                       38                    208                 93
2B                      127                    127                 18
3B                       73                    176                 32
C                       177                    134                 26
CF                       69                    162                 34
DH                        2                     13                 16
LF                       37                    213                 73
OF                       18                     39                 16
P                        12                      2                  0
RF                       35                    162                 82
SS                      126                     84                 12

OPS_tier  Elite (>.900)
Position
1B                   15
2B                    1
3B                    3
C                     1
CF                    9
DH                    4
LF                   10
OF                    2
P                     0
RF                   14
SS                    1
```

In [47]:
```
#Caden Zadell
#Project Assignment 6
import pandas as pd

df = pd.read_csv("cleaned_baseball_data.csv")
print(df.head())
print(df.columns)
```

```
   Player name Position   Games    At-bat    Runs    Hits  Doubles  Trip
les  \
0      B Bonds       LF  2986.0   9847.0  2227.0  2935.0    601.0     7
7.0
1      H Aaron       RF  3298.0  12364.0  2174.0  3771.0    624.0     9
8.0
2       B Ruth       RF  2504.0   8399.0  2174.0  2873.0    506.0    13
6.0
3     A Pujols       1B  3080.0  11421.0  1914.0  3384.0    686.0     1
6.0
4  A Rodriguez       SS  2784.0  10566.0  2021.0  3115.0    548.0     3
1.0

   Home Runs    RBIs  Walked Strikeouts  Steals Caught stealing    AVG
\
0      762.0  1996.0  2558.0       1539   514.0             141  0.298
1      755.0  2297.0  1402.0       1383   240.0              73  0.305
2      714.0  2213.0  2062.0       1330   123.0             117  0.342
3      703.0  2218.0  1373.0       1404   117.0              43  0.296
4      696.0  2086.0  1338.0       2287   329.0              76  0.295

   On-base Percentage  Slugging Percentage  On-base Plus Slugging
0               0.444                0.607                  1.051
1               0.374                0.555                  0.929
2               0.474                0.690                  1.164
3               0.374                0.544                  0.918
4               0.380                0.550                  0.930
Index(['Player name', 'Position', 'Games', 'At-bat', 'Runs', 'Hits', 'D
oubles',
       'Triples', 'Home Runs', 'RBIs', 'Walked', 'Strikeouts', 'Steal
s',
       'Caught stealing', 'AVG', 'On-base Percentage', 'Slugging Percen
tage',
       'On-base Plus Slugging'],
      dtype='object')
```

In [2]:

```
#K Means Clustering of Player Archetypes
```

In [48]:

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

cluster_features = [
    "AVG", "On-base Percentage", "Slugging Percentage",
    "Home Runs", "RBIs", "Runs", "Hits", "Steals"
]

X = df[cluster_features].dropna()

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

kmeans = KMeans(n_clusters=3, random_state=42)
```

```
clusters = kmeans.fit_predict(X_scaled)

X["Cluster"] = clusters
df["Cluster"] = clusters
```

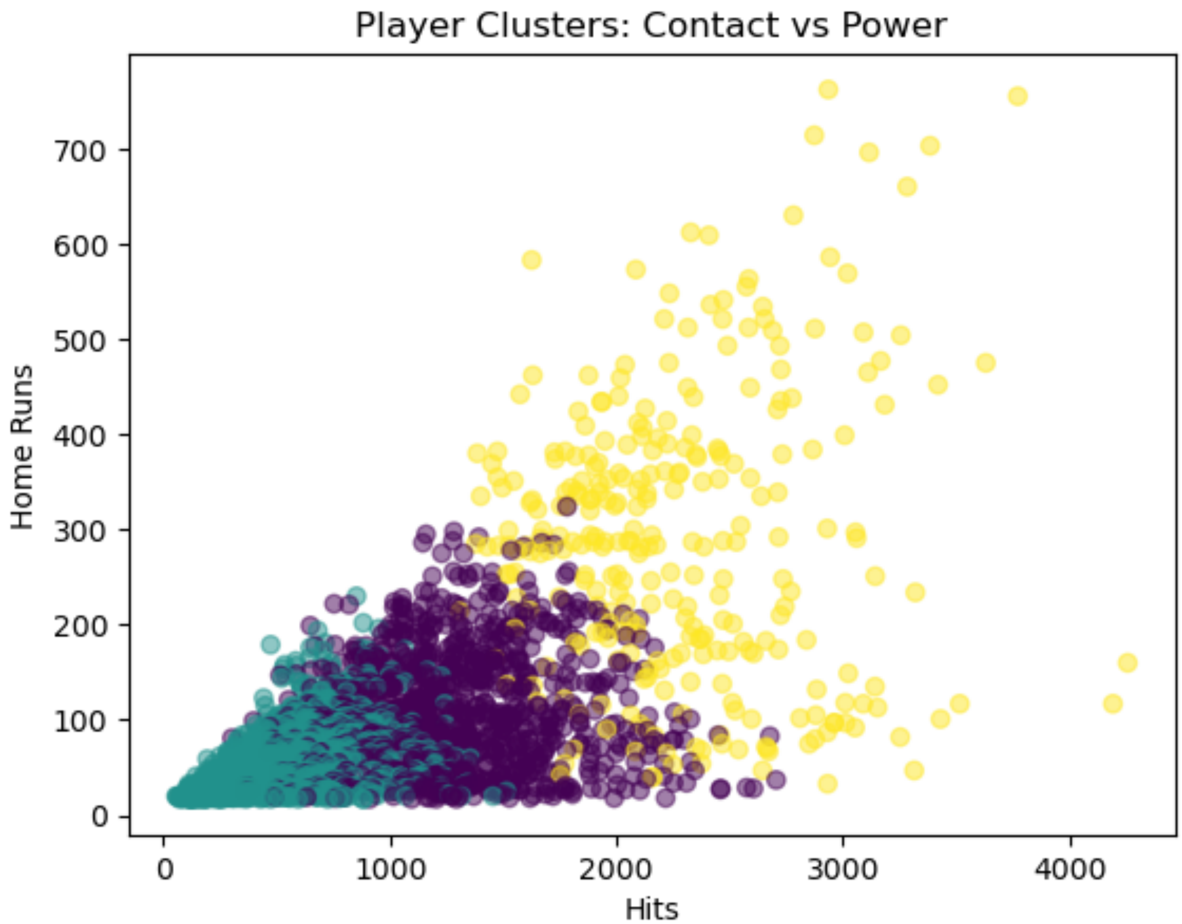
In [50]:
```
cluster_profiles = df.groupby("Cluster")[cluster_features].mean().roun
cluster_profiles
```

Out[50]:

Cluster	AVG	On-base Percentage	Slugging Percentage	Home Runs	RBIs	Runs	Hits	Stea
0	0.28	0.34	0.42	106.75	592.80	655.40	1276.82	105.9
1	0.25	0.31	0.39	52.21	244.40	247.39	513.53	26.4
2	0.29	0.37	0.48	289.02	1242.66	1256.08	2272.49	188.7

In [ ]:

In [51]:
```
import matplotlib.pyplot as plt

plt.scatter(df["Hits"], df["Home Runs"], c=df["Cluster"], alpha=0.5)
plt.xlabel("Hits")
plt.ylabel("Home Runs")
plt.title("Player Clusters: Contact vs Power")
plt.show()
```



In [ ]:

In [52]:
```
cluster_profiles.std(axis=0).sort_values(ascending=False)
```

Out[52]:
```
Hits                     882.034642
Runs                     507.402559
RBIs                     506.659584
Home Runs                124.013400
Steals                    81.195589
Slugging Percentage        0.045826
On-base Percentage         0.030000
AVG                        0.020817
dtype: float64
```

In [ ]: