In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")

sns.set_style("whitegrid")
```

```
C:\Users\pramo\anaconda3\Lib\site-packages\pandas\core\arrays\masked.py:60: Us
erWarning: Pandas requires version '1.3.6' or newer of 'bottleneck' (version
'1.3.5' currently installed).
  from pandas.core import (
```

Dataset

Loading Dataset

In [2]:
```
df=pd.read_csv('cricket_data_2025.csv')

df.head(10)
```

Out[2]:

	Year	Player_Name	Matches_Batted	Not_Outs	Runs_Scored	Highest_Score	Batting_Average
0	NaN	Aaron Hardie	No stats	No stats	No stats	No stats	No stats
1	2024.0	Abdul Samad	16	2	182	37*	18.2
2	2023.0	Abdul Samad	9	4	169	37*	42.25
3	2022.0	Abdul Samad	2	0	4	4	2.0
4	2021.0	Abdul Samad	11	1	111	28	12.33
5	2020.0	Abdul Samad	12	3	111	33	22.2
6	NaN	Abhinandan Singh	No stats	No stats	No stats	No stats	No stats
7	2024.0	Abhinav Manohar	2	0	9	8	4.5
8	2023.0	Abhinav Manohar	9	0	114	42	19.0
9	2022.0	Abhinav Manohar	8	1	108	43	18.0

10 rows × 25 columns

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1008 entries, 0 to 1007
Data columns (total 25 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Year                 957 non-null    float64
 1   Player_Name          1008 non-null   object 
 2   Matches_Batted       1008 non-null   object 
 3   Not_Outs             1008 non-null   object 
 4   Runs_Scored          1008 non-null   object 
 5   Highest_Score        1008 non-null   object 
 6   Batting_Average      1008 non-null   object 
 7   Balls_Faced          1008 non-null   object 
 8   Batting_Strike_Rate  1008 non-null   object 
 9   Centuries            1008 non-null   object 
 10  Half_Centuries       1008 non-null   object 
 11  Fours                1008 non-null   object 
 12  Sixes                1008 non-null   object 
 13  Catches_Taken        1008 non-null   object 
 14  Stumpings            1008 non-null   object 
 15  Matches_Bowled       1008 non-null   object 
 16  Balls_Bowled         1008 non-null   object 
 17  Runs_Conceded        1008 non-null   object 
 18  Wickets_Taken        1008 non-null   object 
 19  Best_Bowling_Match   1008 non-null   object 
 20  Bowling_Average      1008 non-null   object 
 21  Economy_Rate         1008 non-null   object 
 22  Bowling_Strike_Rate  1008 non-null   object 
 23  Four_Wicket_Hauls    1008 non-null   object 
 24  Five_Wicket_Hauls    1008 non-null   object 
dtypes: float64(1), object(24)
memory usage: 197.0+ KB
```

Cleaning Dataset

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
Year                  51
Player_Name            0
Matches_Batted         0
Not_Outs               0
Runs_Scored            0
Highest_Score          0
Batting_Average        0
Balls_Faced            0
Batting_Strike_Rate    0
Centuries              0
Half_Centuries         0
Fours                  0
Sixes                  0
Catches_Taken          0
Stumpings              0
Matches_Bowled         0
Balls_Bowled           0
Runs_Conceded          0
Wickets_Taken          0
Best_Bowling_Match     0
Bowling_Average        0
Economy_Rate           0
Bowling_Strike_Rate    0
Four_Wicket_Hauls      0
Five_Wicket_Hauls      0
dtype: int64
```

In [5]:
```
df['Player_Name'].value_counts()
```

Out[5]:
```
Player_Name
MS Dhoni           17
Manish Pandey      17
Rohit Sharma       17
Virat Kohli        17
Ravindra Jadeja    16
                   ..
Kwena Maphaka       1
Kunal Rathore       1
Kumar Kushagra      1
Karim Janat         1
Zeeshan Ansari      1
Name: count, Length: 227, dtype: int64
```

In [6]:
```
df['Year'].value_counts()
```

Out[6]:
```
Year
2024.0    154
2023.0    128
2022.0    115
2021.0     91
2020.0     87
2019.0     77
2018.0     60
2017.0     53
2016.0     46
2015.0     36
2014.0     33
2013.0     26
2012.0     16
2011.0     11
2009.0      9
2010.0      8
2008.0      7
Name: count, dtype: int64
```

In [7]:
```
df['Year'].fillna(df['Year'].mode()[0], inplace=True)
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
Year                 0
Player_Name          0
Matches_Batted       0
Not_Outs             0
Runs_Scored          0
Highest_Score        0
Batting_Average      0
Balls_Faced          0
Batting_Strike_Rate  0
Centuries            0
Half_Centuries       0
Fours                0
Sixes                0
Catches_Taken        0
Stumpings            0
Matches_Bowled       0
Balls_Bowled         0
Runs_Conceded        0
Wickets_Taken        0
Best_Bowling_Match   0
Bowling_Average      0
Economy_Rate         0
Bowling_Strike_Rate  0
Four_Wicket_Hauls    0
Five_Wicket_Hauls    0
dtype: int64
```

In [9]:
```
df = df.drop(index=0).reset_index(drop=True)
df.head()
```

Out[9]:

	Year	Player_Name	Matches_Batted	Not_Outs	Runs_Scored	Highest_Score	Batting_Average
0	2024.0	Abdul Samad	16	2	182	37*	18.2
1	2023.0	Abdul Samad	9	4	169	37*	42.25
2	2022.0	Abdul Samad	2	0	4	4	2.0
3	2021.0	Abdul Samad	11	1	111	28	12.33
4	2020.0	Abdul Samad	12	3	111	33	22.2

5 rows × 25 columns

Data Transformation

In [10]:
```
columns_to_convert = df.columns.difference(['Player_Name','Best_Bowling_Match']
df[columns_to_convert] = df[columns_to_convert].apply(pd.to_numeric, errors='co
df.info()
```

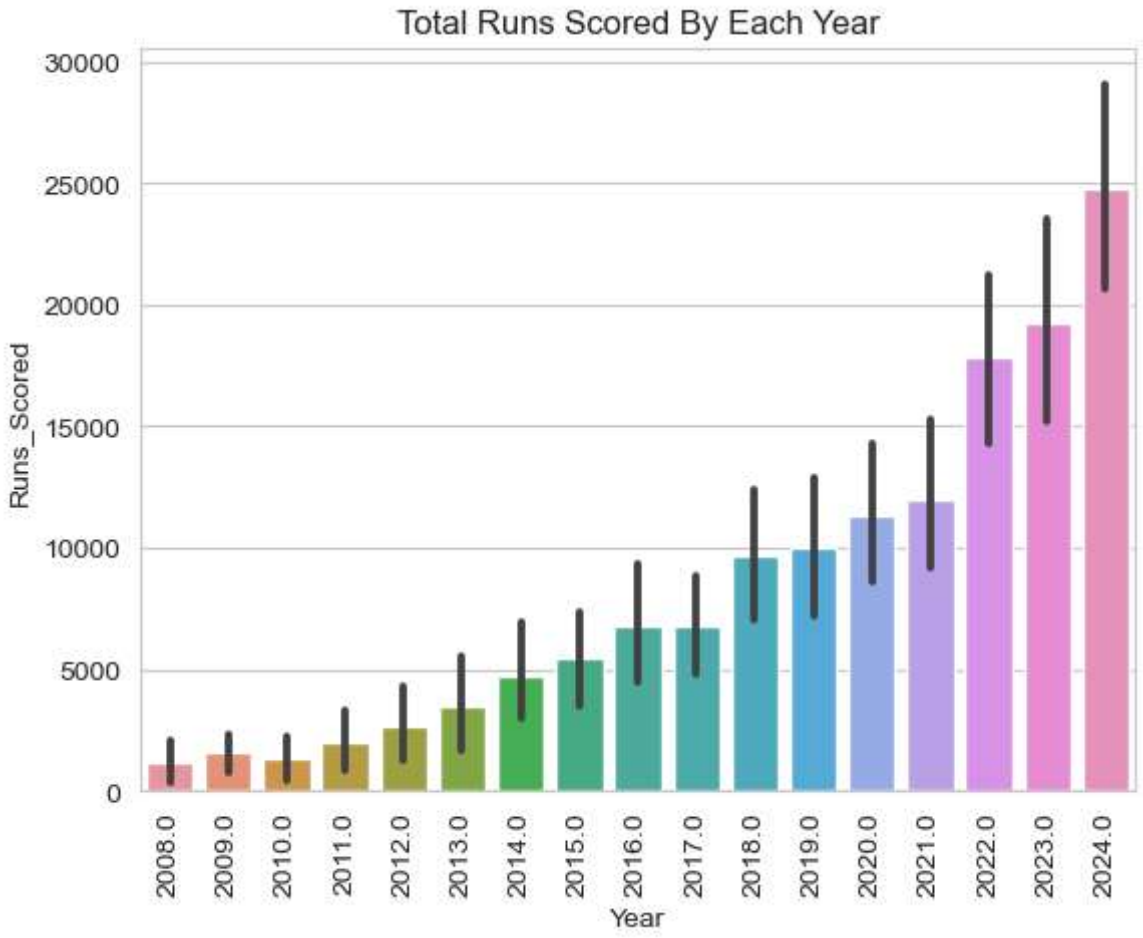
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1007 entries, 0 to 1006
Data columns (total 25 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Year                 1007 non-null   float64
 1   Player_Name          1007 non-null   object 
 2   Matches_Batted       957 non-null    float64
 3   Not_Outs             957 non-null    float64
 4   Runs_Scored          957 non-null    float64
 5   Highest_Score        574 non-null    float64
 6   Batting_Average      957 non-null    float64
 7   Balls_Faced          957 non-null    float64
 8   Batting_Strike_Rate  957 non-null    float64
 9   Centuries            957 non-null    float64
 10  Half_Centuries       957 non-null    float64
 11  Fours                957 non-null    float64
 12  Sixes                957 non-null    float64
 13  Catches_Taken        957 non-null    float64
 14  Stumpings            957 non-null    float64
 15  Matches_Bowled       958 non-null    float64
 16  Balls_Bowled         958 non-null    float64
 17  Runs_Conceded        958 non-null    float64
 18  Wickets_Taken        958 non-null    float64
 19  Best_Bowling_Match   1007 non-null   object 
 20  Bowling_Average      958 non-null    float64
 21  Economy_Rate         958 non-null    float64
 22  Bowling_Strike_Rate  958 non-null    float64
 23  Four_Wicket_Hauls    958 non-null    float64
 24  Five_Wicket_Hauls    958 non-null    float64
dtypes: float64(23), object(2)
memory usage: 196.8+ KB
```

In [11]:
```
df.fillna(df.median(numeric_only=True), inplace=True)
```

EDA

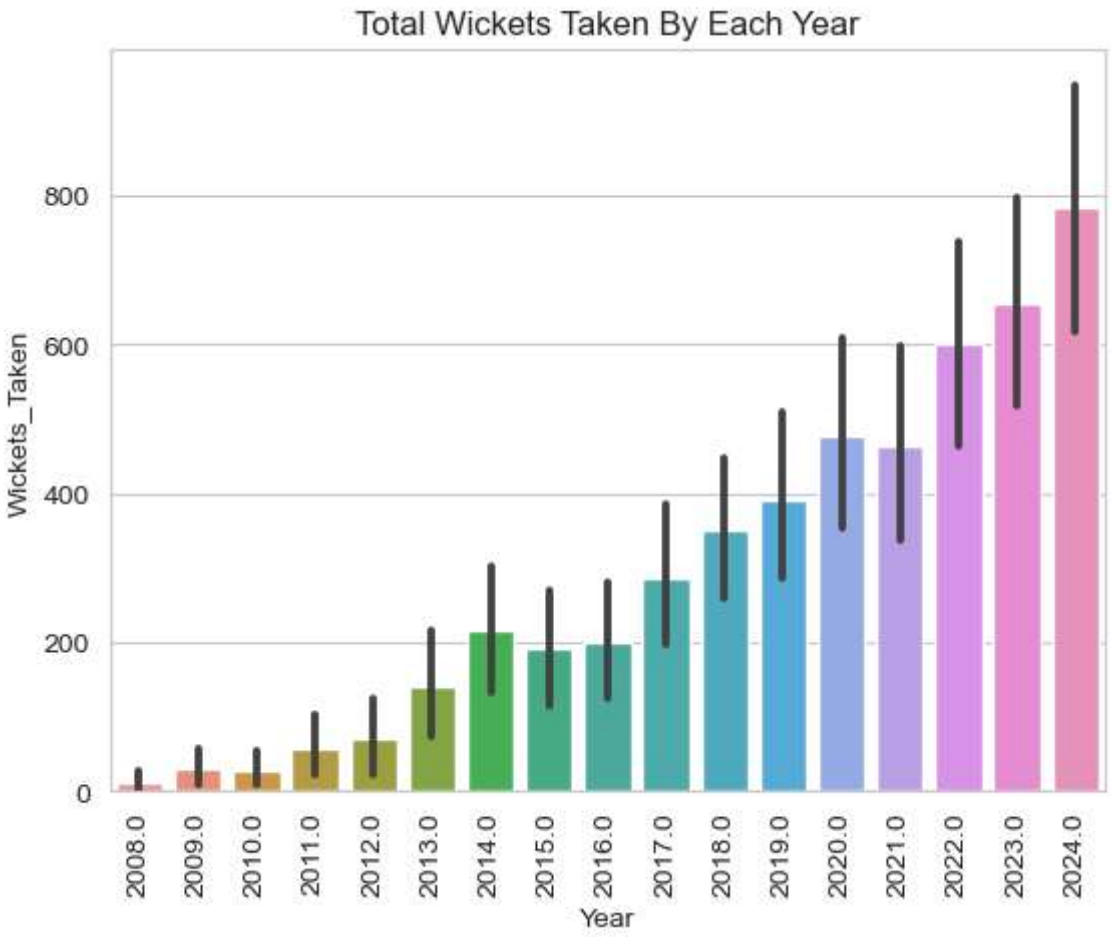
Total Runs Scored By Each Year

In [12]:
```
sns.barplot(data=df,x='Year',y='Runs_Scored',estimator="sum")
plt.xticks(rotation=90)
plt.title("Total Runs Scored By Each Year")
plt.show()
```



Total Wickets Taken By Each Year

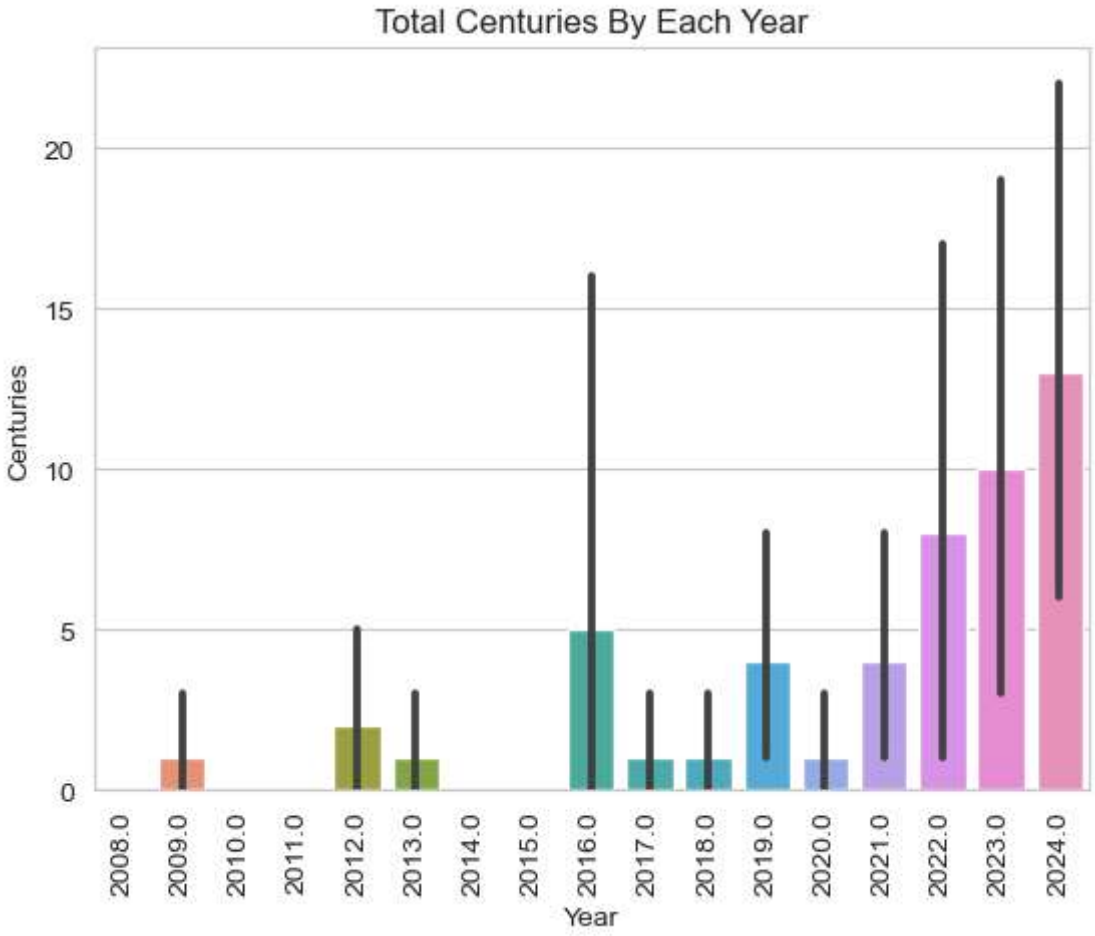
In [13]:
```
sns.barplot(data=df,x='Year',y='Wickets_Taken',estimator="sum")
plt.xticks(rotation=90)
plt.title("Total Wickets Taken By Each Year")
plt.show()
```



Total Centures By each Year

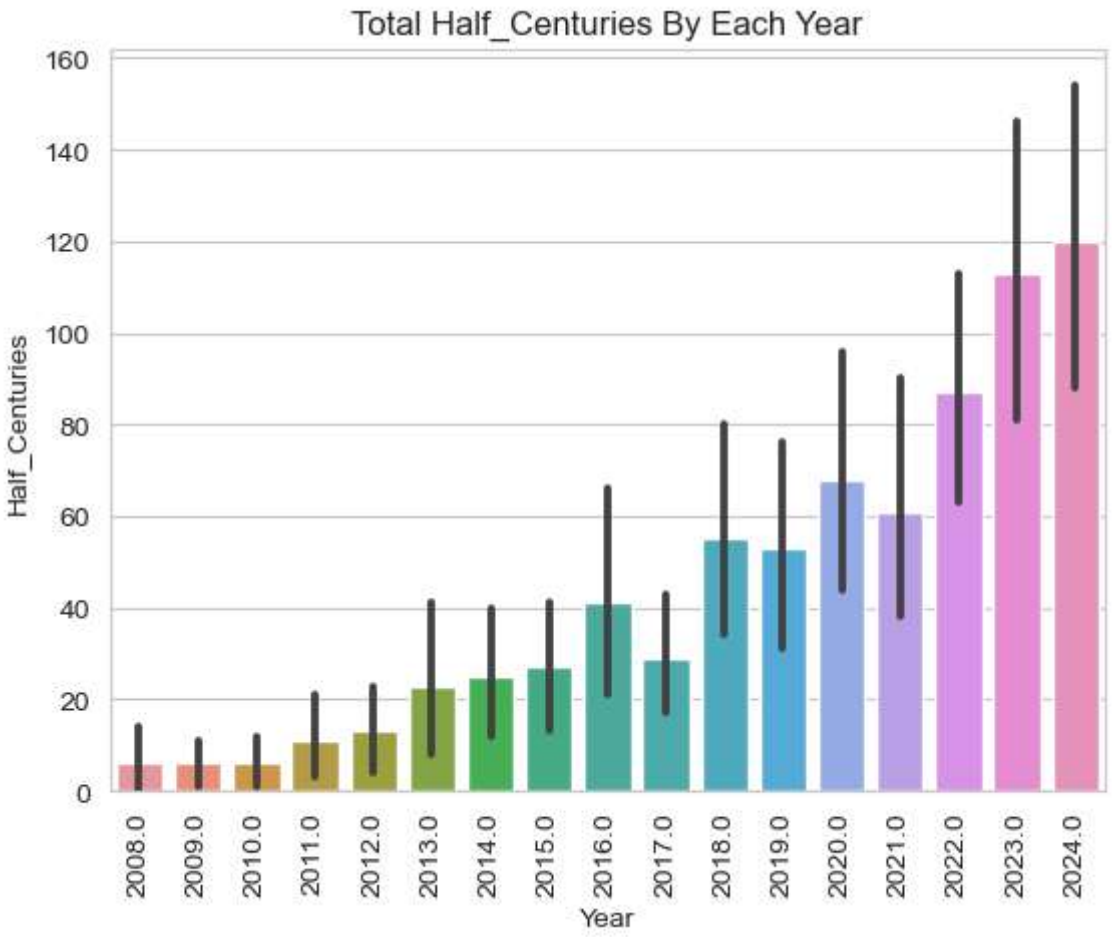
In [14]:

```
sns.barplot(data=df,x='Year',y='Centuries',estimator="sum")
plt.xticks(rotation=90)
plt.title("Total Centuries By Each Year")
plt.show()
```



Total Half Centures By each Year

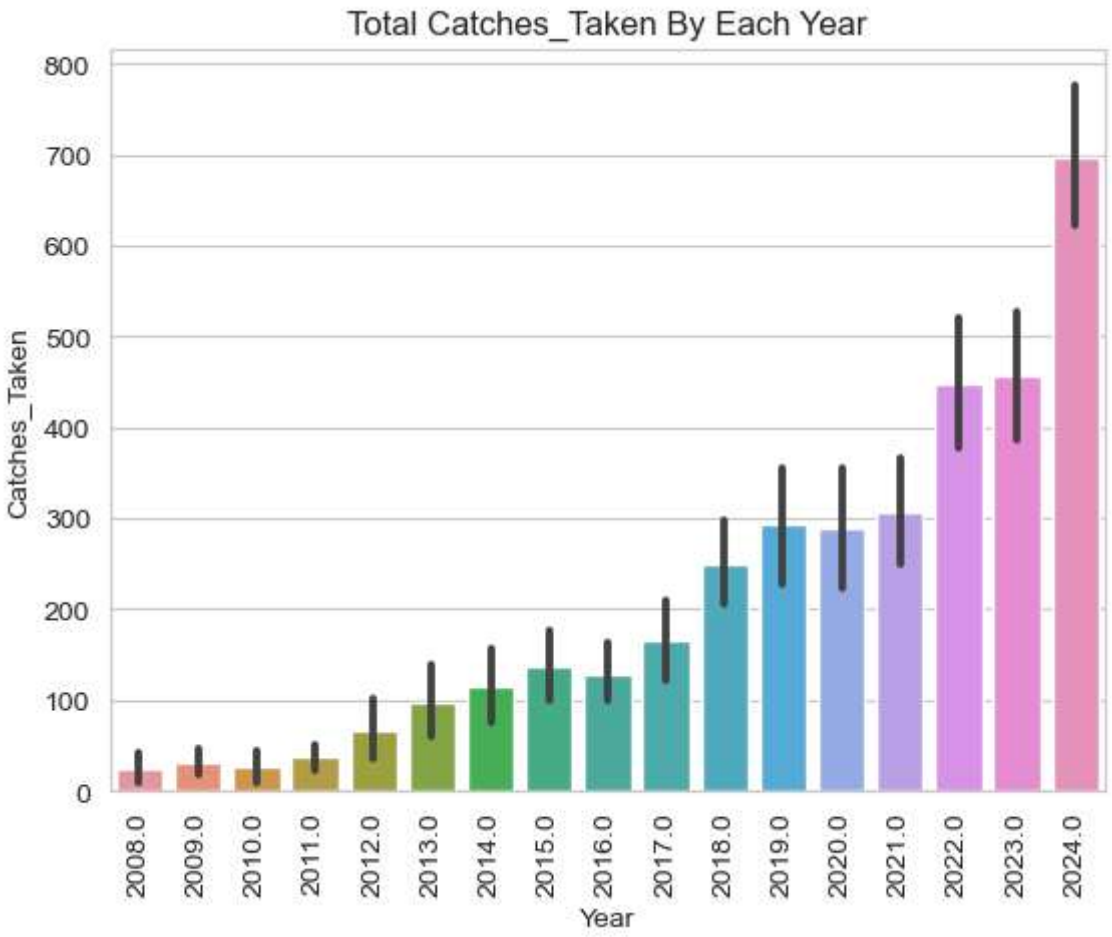
In [15]:
```
sns.barplot(data=df,x='Year',y='Half_Centuries',estimator="sum")
plt.xticks(rotation=90)
plt.title("Total Half_Centuries By Each Year")
plt.show()
```



Total Catches_Taken By each Year

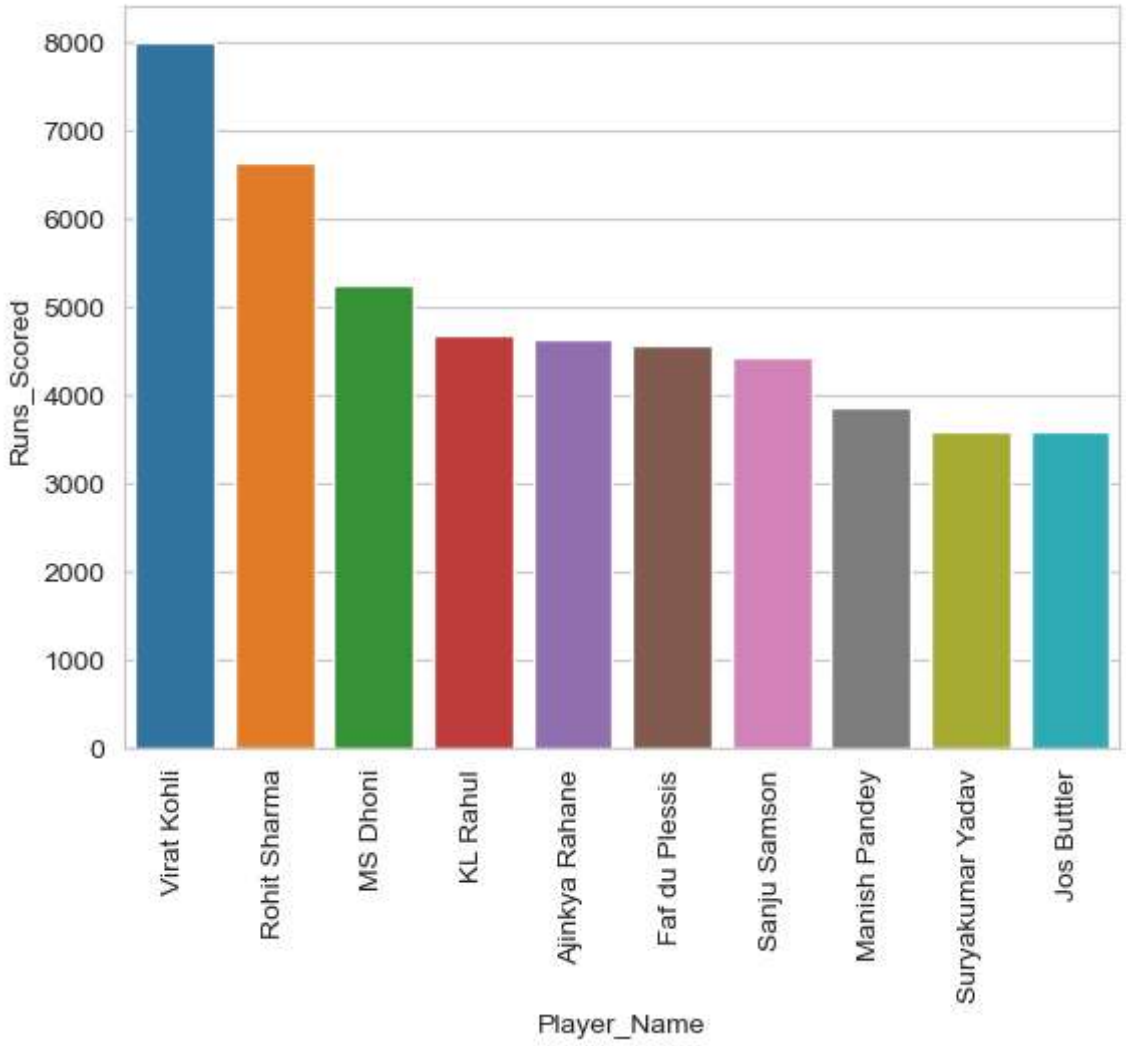
In [16]:

```
sns.barplot(data=df,x='Year',y='Catches_Taken',estimator="sum")
plt.xticks(rotation=90)
plt.title("Total Catches_Taken By Each Year")
plt.show()
```



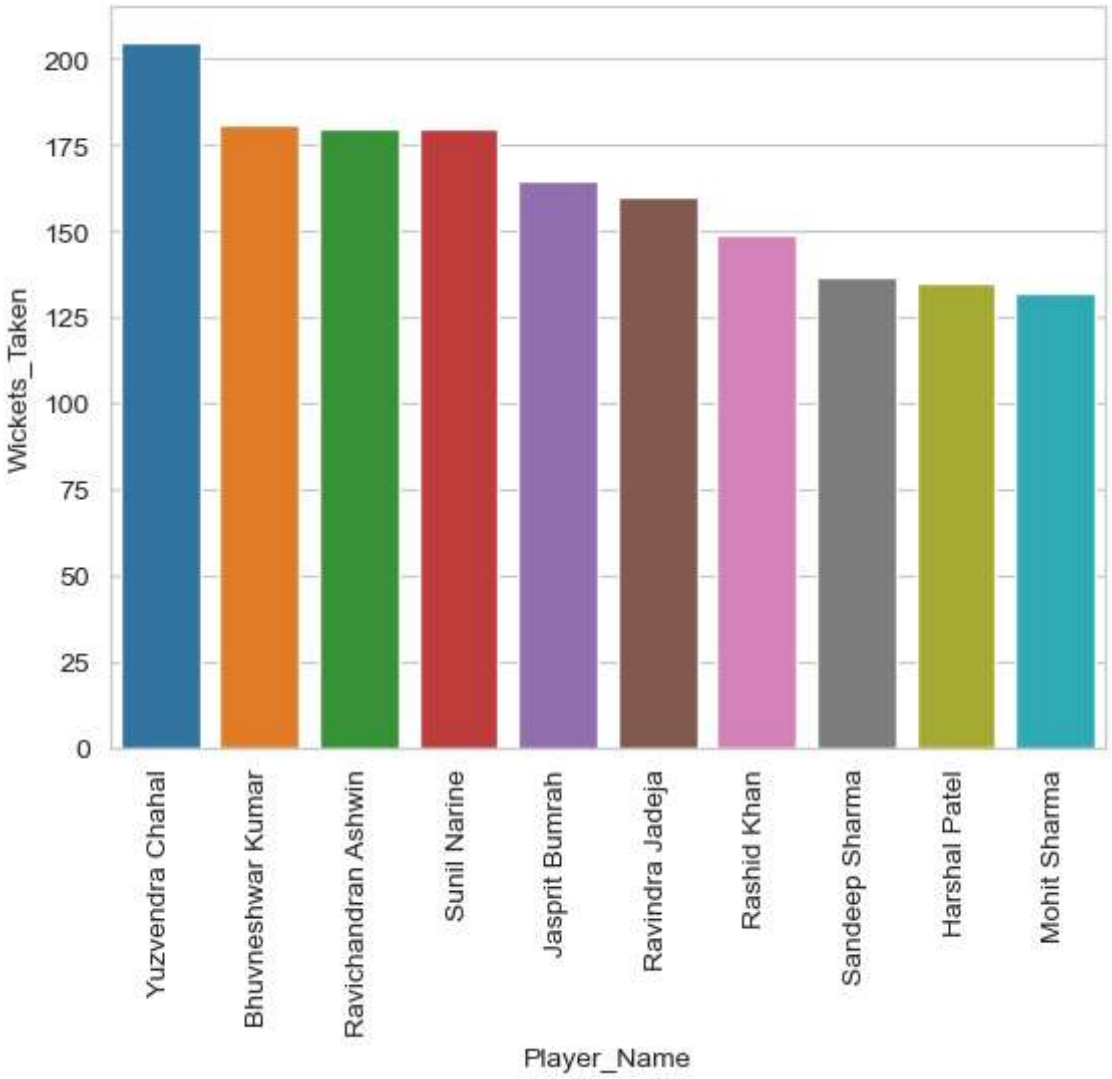
Top 10 Player With Most Runs Taken

In [17]:
```
df_grouped = df.groupby("Player_Name")['Runs_Scored'].sum().reset_index()

top_10_players = df_grouped.nlargest(10, 'Runs_Scored')
sns.barplot(data=top_10_players, x='Player_Name', y='Runs_Scored')
plt.xticks(rotation=90)
plt.show()
```



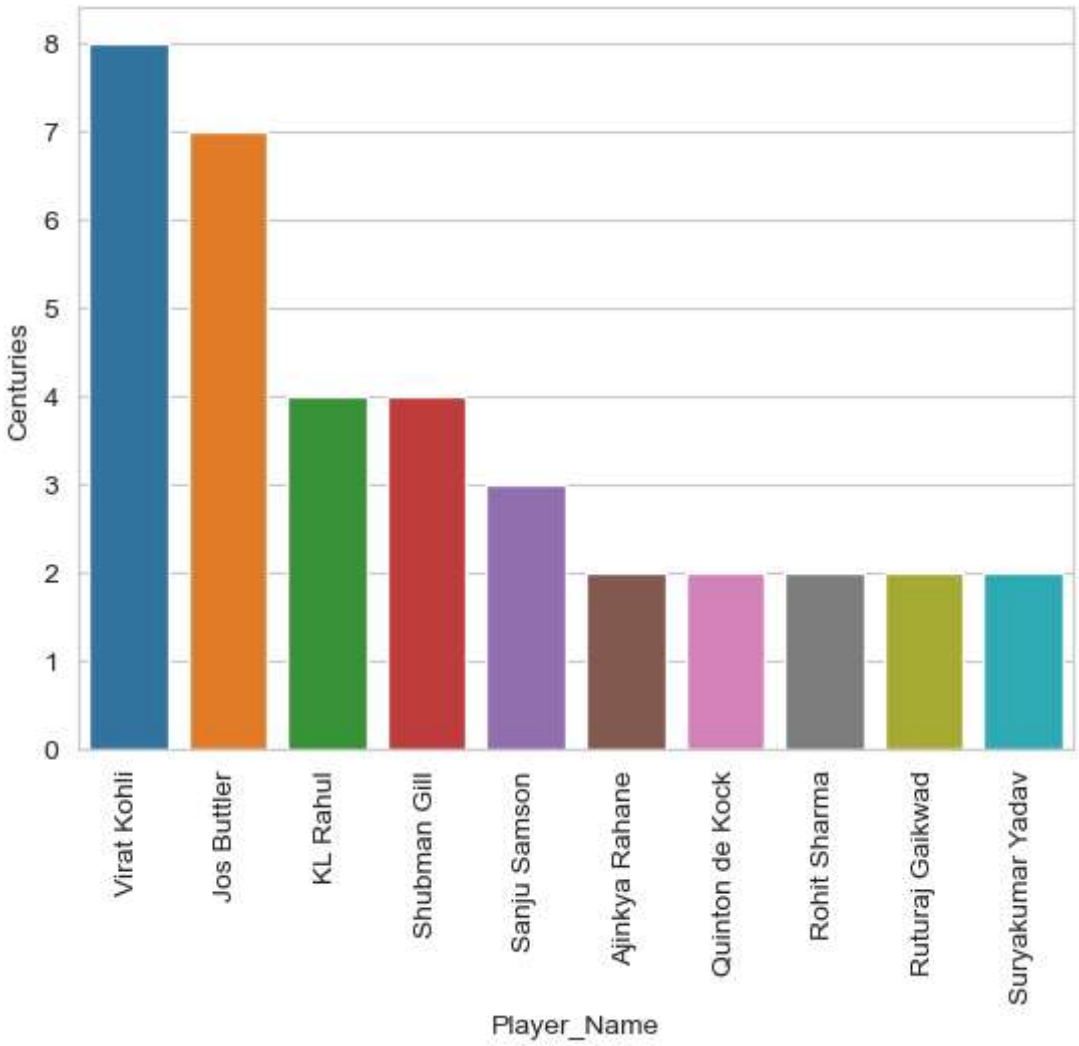
Top 10 Player With Highest Wicket Taken

In [18]:
```
df_grouped = df.groupby("Player_Name")['Wickets_Taken'].sum().reset_index()

top_10_players = df_grouped.nlargest(10, 'Wickets_Taken')
sns.barplot(data=top_10_players, x='Player_Name', y='Wickets_Taken')
plt.xticks(rotation=90)
plt.show()
```



Top 10 Players With Most Centuries

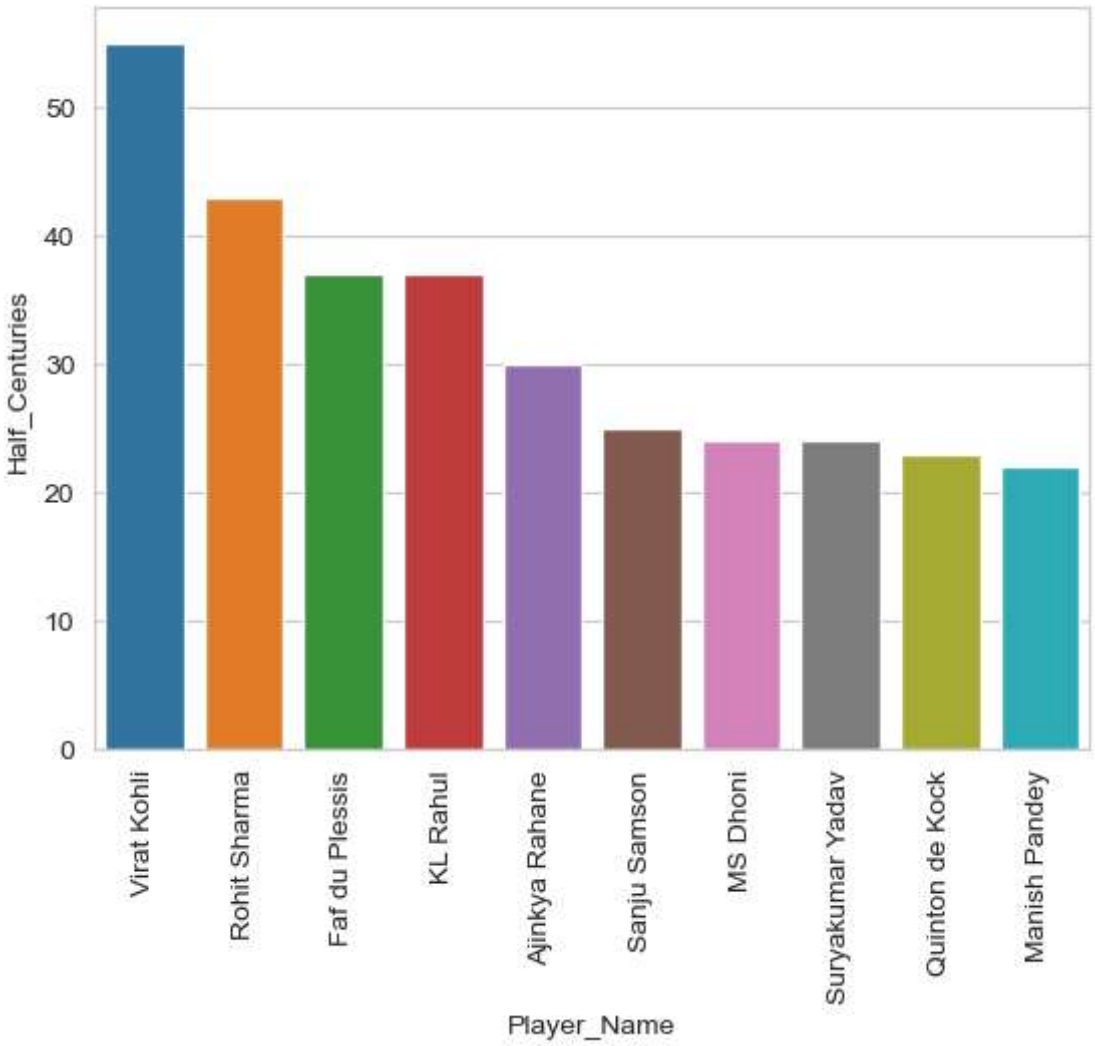
In [19]:
```
df_grouped = df.groupby("Player_Name")['Centuries'].sum().reset_index()

top_10_players = df_grouped.nlargest(10, 'Centuries')
sns.barplot(data=top_10_players, x='Player_Name', y='Centuries')
plt.xticks(rotation=90)
plt.show()
```



Top 10 Players with Most Half Centuries

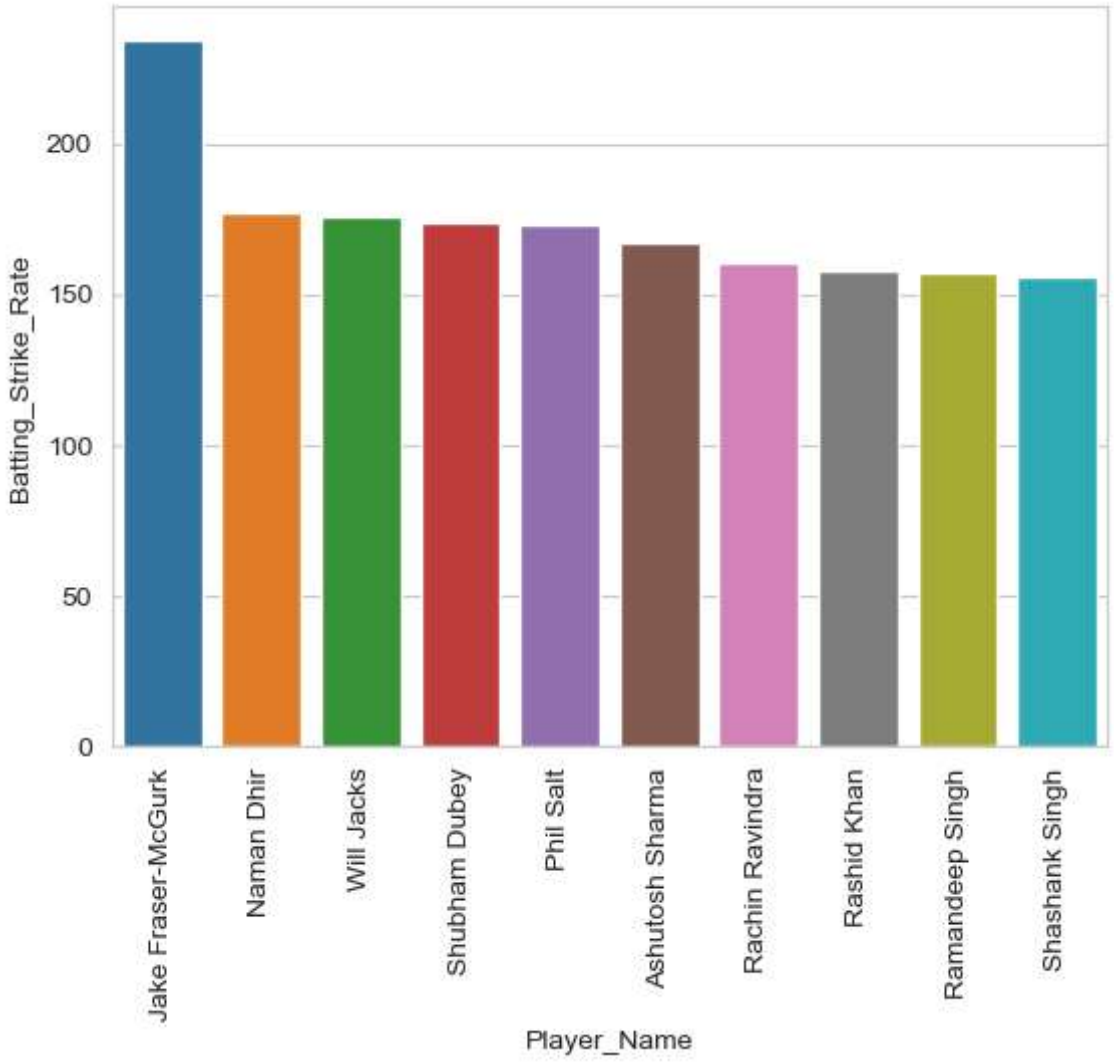
In [20]:

```
df_grouped = df.groupby("Player_Name")['Half_Centuries'].sum().reset_index()

top_10_players = df_grouped.nlargest(10, 'Half_Centuries')
sns.barplot(data=top_10_players, x='Player_Name', y='Half_Centuries')
plt.xticks(rotation=90)
plt.show()
```



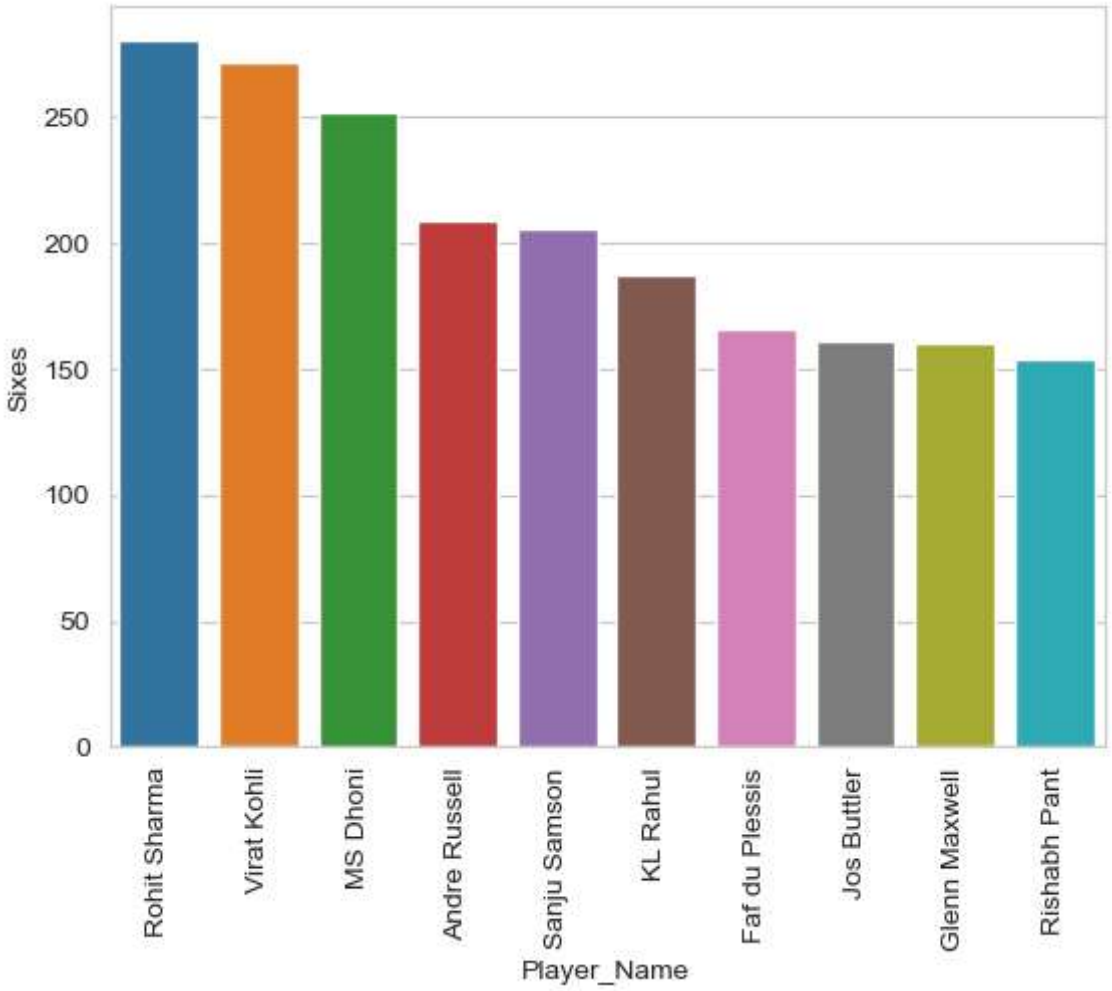
Top 10 Players With Highest Batting Strike Rate

In [21]:
```
df_grouped = df.groupby("Player_Name")['Batting_Strike_Rate'].mean().reset_inde

top_10_players = df_grouped.nlargest(10, 'Batting_Strike_Rate')
sns.barplot(data=top_10_players, x='Player_Name', y='Batting_Strike_Rate')
plt.xticks(rotation=90)
plt.show()
```



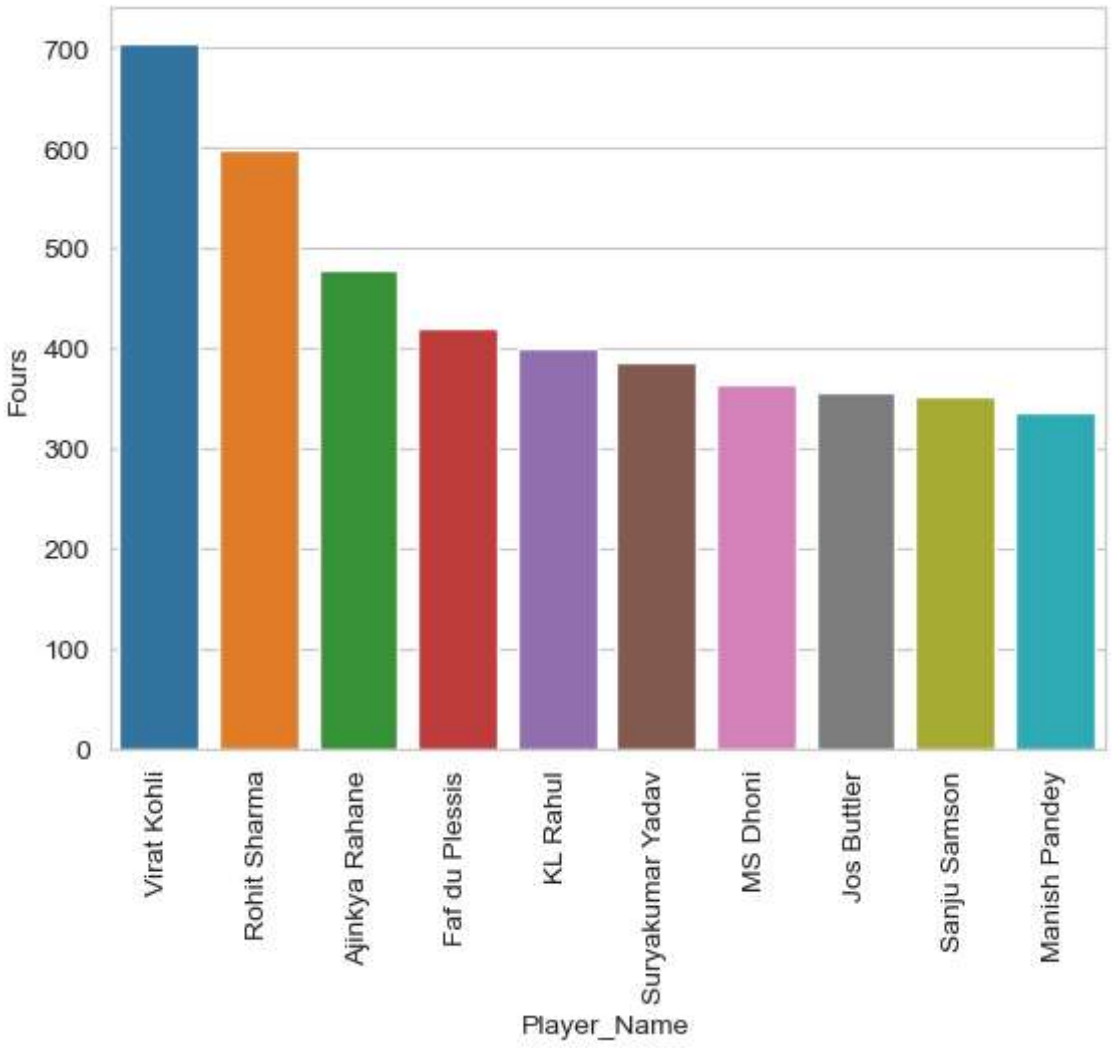
Top 10 Players With Most Sixes

In [22]:
```
df_grouped = df.groupby("Player_Name")['Sixes'].sum().reset_index()

top_10_players = df_grouped.nlargest(10, 'Sixes')
sns.barplot(data=top_10_players, x='Player_Name', y='Sixes')
plt.xticks(rotation=90)
plt.show()
```



Top 10 Players With Highest Fours

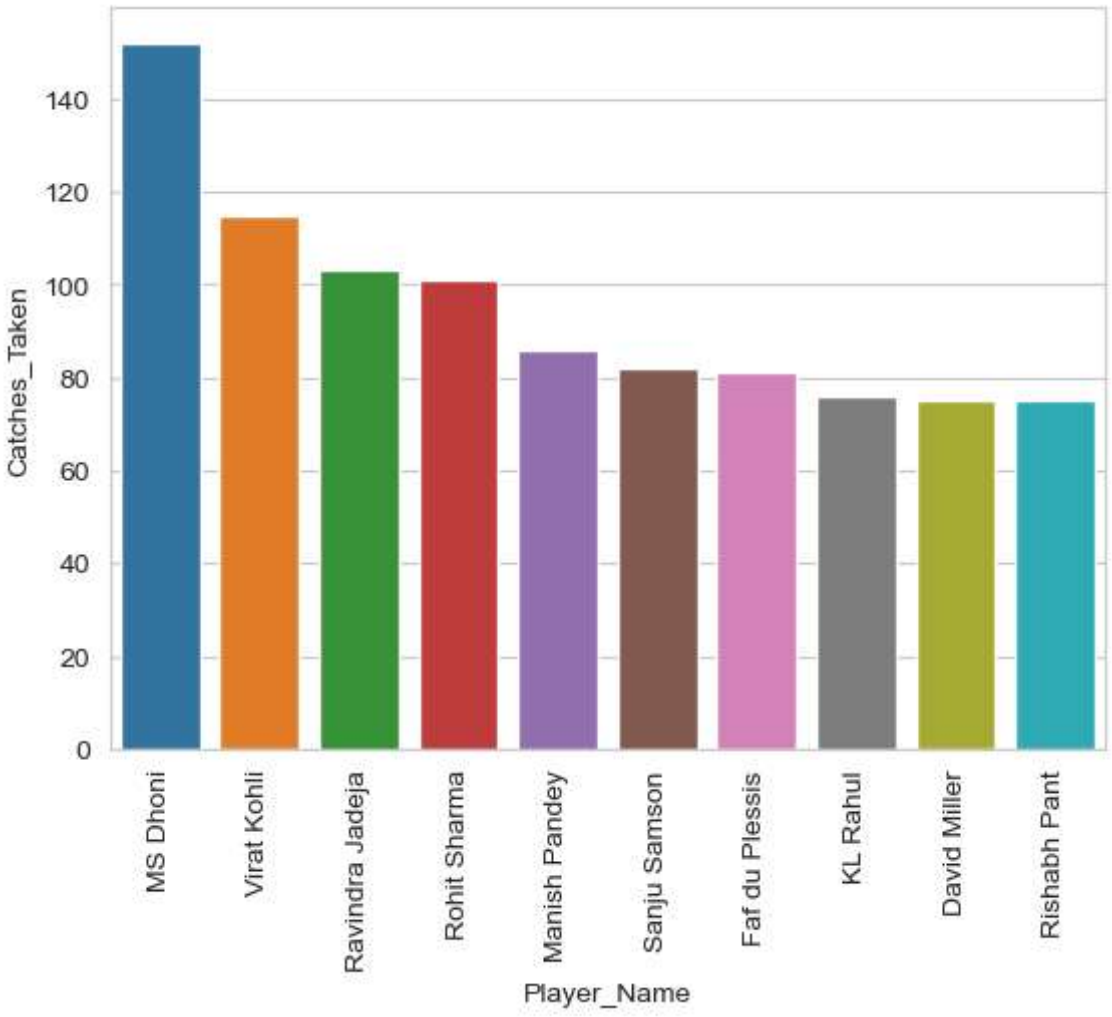
In [23]:
```
df_grouped = df.groupby("Player_Name")['Fours'].sum().reset_index()

top_10_players = df_grouped.nlargest(10, 'Fours')
sns.barplot(data=top_10_players, x='Player_Name', y='Fours')
plt.xticks(rotation=90)
plt.show()
```



Top 10 Most Catches Taken

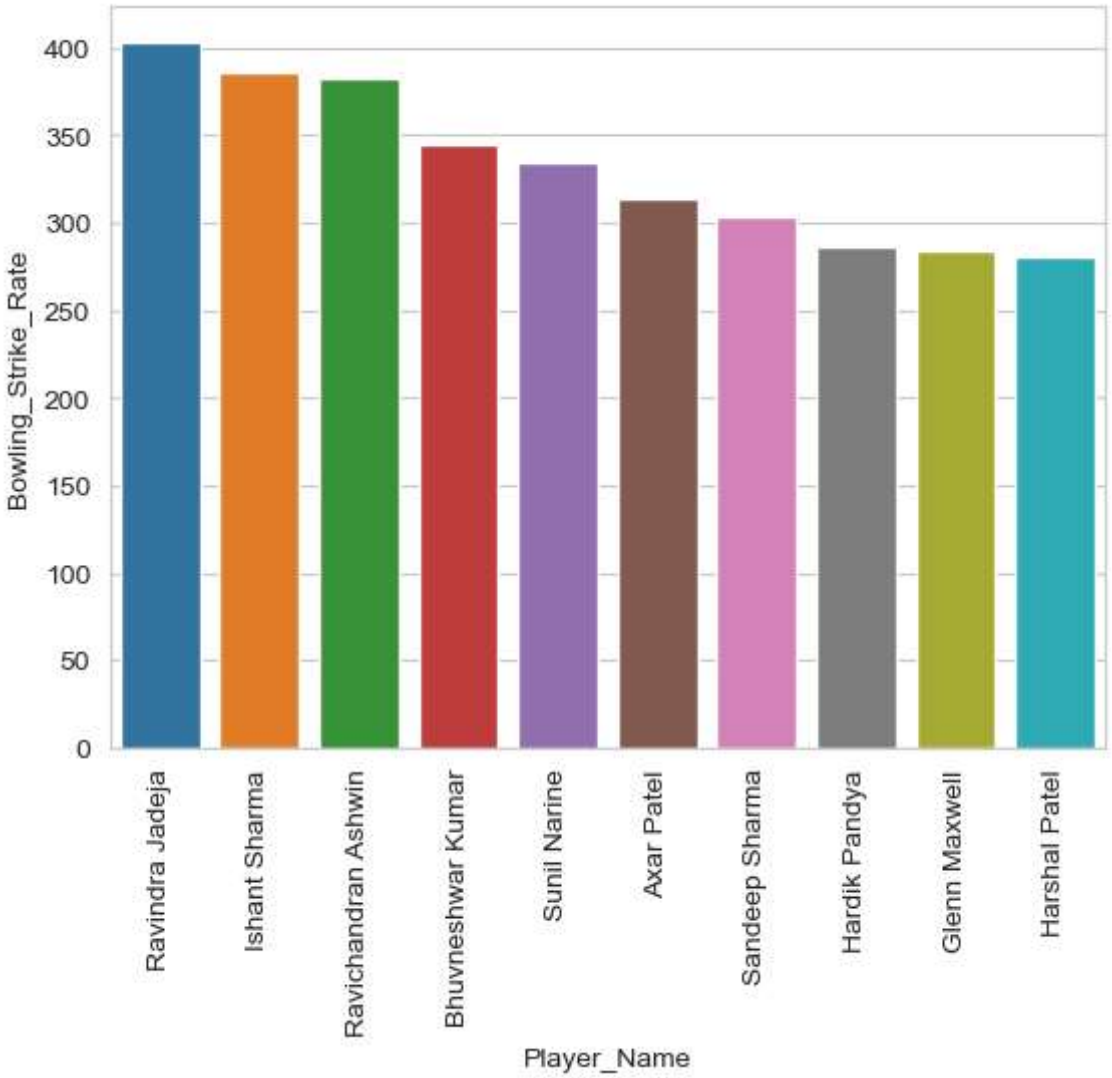
In [24]:

```
df_grouped = df.groupby("Player_Name")['Catches_Taken'].sum().reset_index()

top_10_players = df_grouped.nlargest(10, 'Catches_Taken')
sns.barplot(data=top_10_players, x='Player_Name', y='Catches_Taken')
plt.xticks(rotation=90)
plt.show()
```



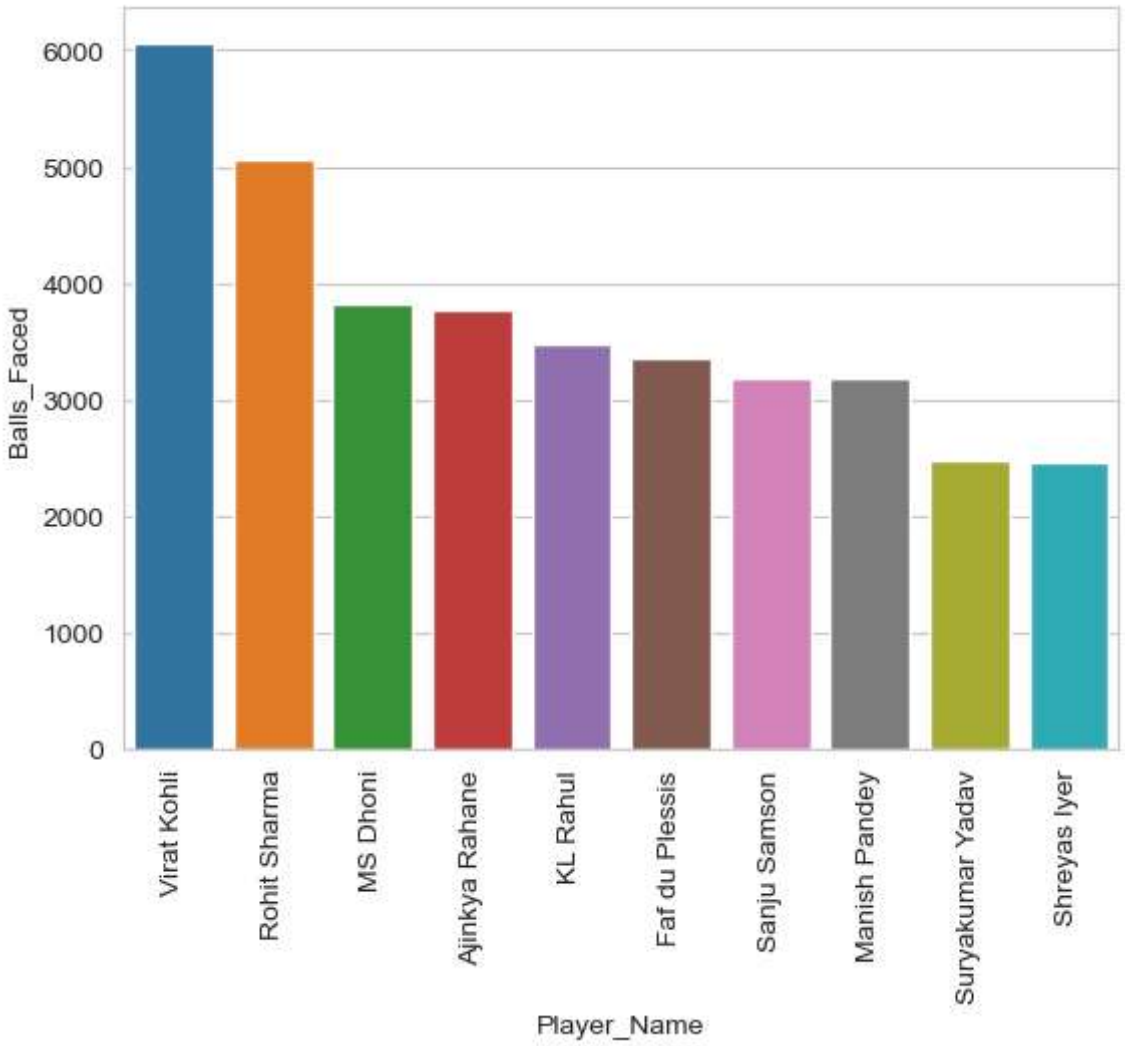
Top 10 Players who have Highest Bowling Strike Rate

In [25]:
```
df_grouped = df.groupby("Player_Name")['Bowling_Strike_Rate'].sum().reset_index

top_10_players = df_grouped.nlargest(10, 'Bowling_Strike_Rate')
sns.barplot(data=top_10_players, x='Player_Name', y='Bowling_Strike_Rate')
plt.xticks(rotation=90)
plt.show()
```



Top 10 players Who Faced Most Balls

In [26]:
```
df_grouped = df.groupby("Player_Name")['Balls_Faced'].sum().reset_index()

top_10_players = df_grouped.nlargest(10, 'Balls_Faced')
sns.barplot(data=top_10_players, x='Player_Name', y='Balls_Faced')
plt.xticks(rotation=90)
plt.show()
```

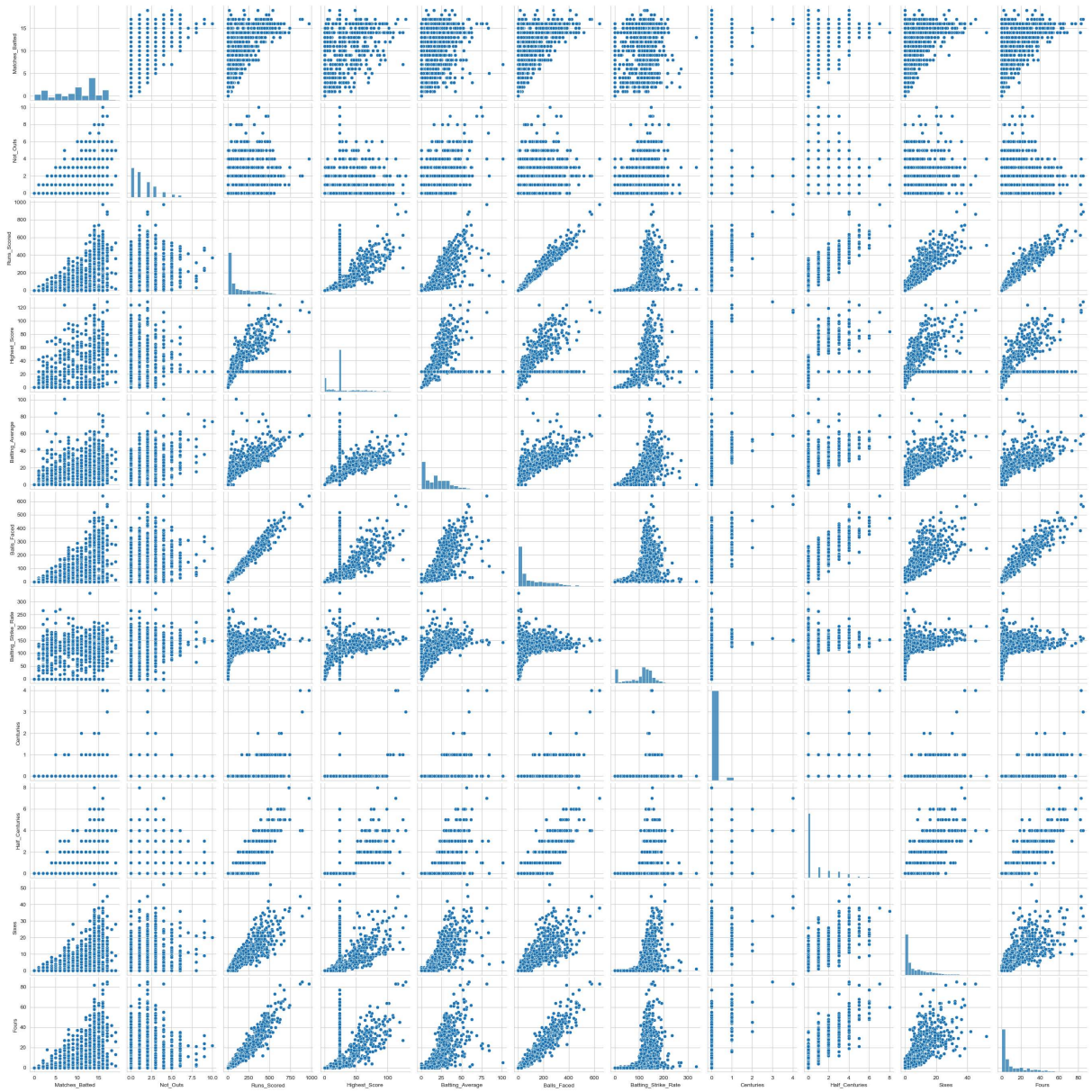


RelationShip

Batting

In [27]:
```
df_Batting = df[["Matches_Batted", "Not_Outs", "Runs_Scored", "Highest_Score",
                 "Batting_Strike_Rate", "Centuries", "Half_Centuries", "Sixes",
```

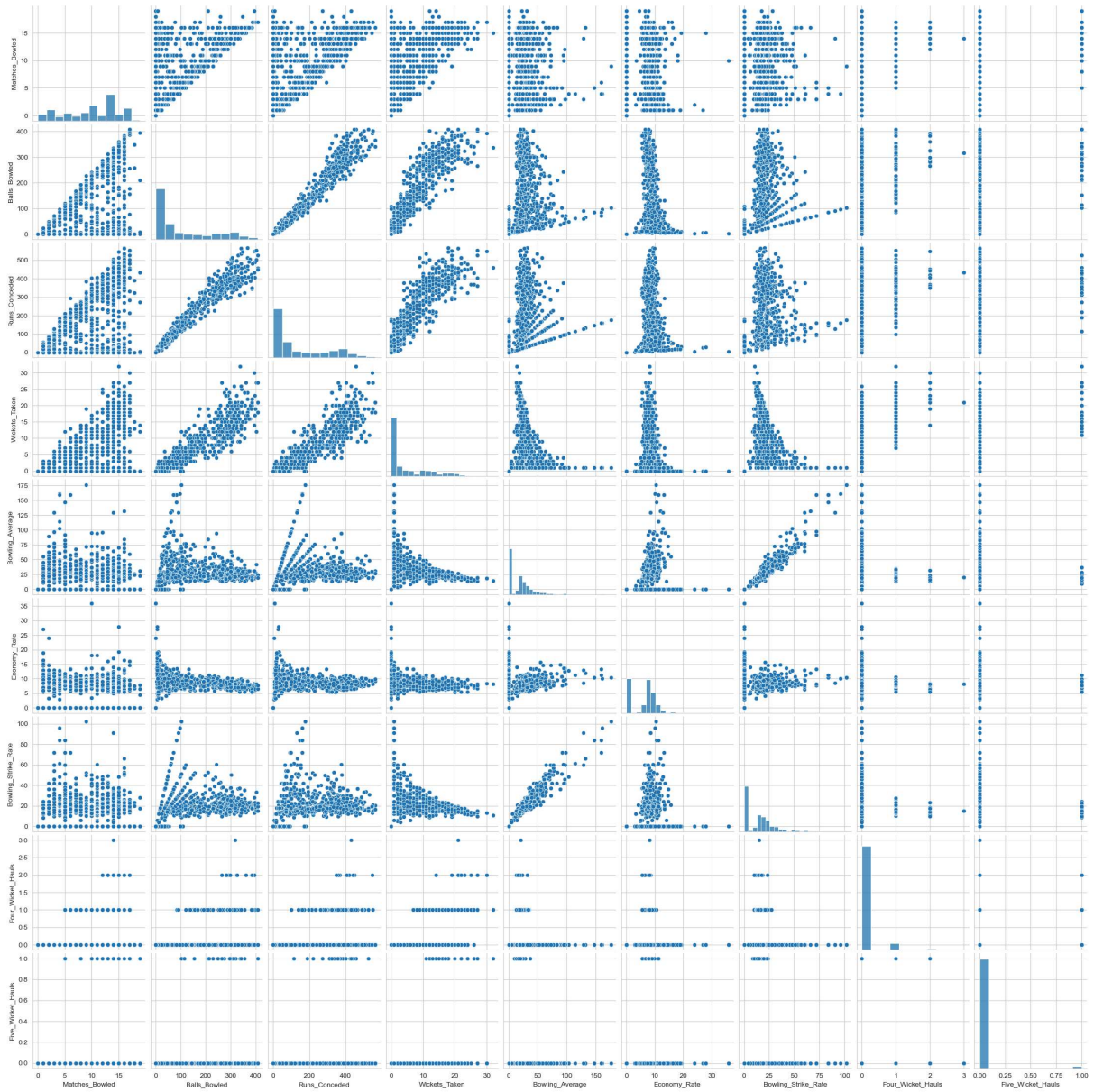
In [28]:
```
sns.pairplot(data=df_Batting)
plt.show()
```



Bowling

In [29]:
```
df_Bowling=df[["Matches_Bowled","Balls_Bowled","Runs_Conceded","Wickets_Taken",
               "Economy_Rate","Bowling_Strike_Rate","Four_Wicket_Hauls","Five_W
```

In [30]:
```
sns.pairplot(data=df_Bowling)
plt.show()
```



Predictive Modeling For Batting and Bowling

Runs Scored Predction

In [31]:
```
X = df_Batting.drop("Runs_Scored", axis=1)
y = df_Batting["Runs_Scored"]
```

In [32]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random
```

In [33]:
```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Linear Regression
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_predictions = lr_model.predict(X_test)
lr_mse = mean_squared_error(y_test, lr_predictions)
lr_r2 = r2_score(y_test, lr_predictions)

print("Linear Regression:")
print(f"Mean Squared Error: {lr_mse}")
print(f"R2 Score: {lr_r2}")

# Decision Tree Regressor
dt_model = DecisionTreeRegressor(random_state=42)
dt_model.fit(X_train, y_train)
dt_predictions = dt_model.predict(X_test)
dt_mse = mean_squared_error(y_test, dt_predictions)
dt_r2 = r2_score(y_test, dt_predictions)

print("\nDecision Tree Regressor:")
print(f"Mean Squared Error: {dt_mse}")
print(f"R2 Score: {dt_r2}")

# Random Forest Regressor
rf_model = RandomForestRegressor(random_state=42, n_estimators=100)
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)
rf_mse = mean_squared_error(y_test, rf_predictions)
rf_r2 = r2_score(y_test, rf_predictions)

print("\nRandom Forest Regressor:")
print(f"Mean Squared Error: {rf_mse}")
print(f"R2 Score: {rf_r2}")
```

```
Linear Regression:
Mean Squared Error: 76.74298764067167
R2 Score: 0.9974431975862951

Decision Tree Regressor:
Mean Squared Error: 703.5693069306931
R2 Score: 0.9765595820872659

Random Forest Regressor:
Mean Squared Error: 123.85889059405937
R2 Score: 0.9958734638803418
```

Wickets Taken Prediction

In [34]:
```
X = df_Bowling.drop('Wickets_Taken', axis=1)
y = df_Bowling['Wickets_Taken']
```

In [35]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random
```

In [36]:
```
# Linear Regression
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_predictions = lr_model.predict(X_test)
lr_mse = mean_squared_error(y_test, lr_predictions)
lr_r2 = r2_score(y_test, lr_predictions)

print("Linear Regression:")
print(f"Mean Squared Error: {lr_mse}")
print(f"R2 Score: {lr_r2}")

# Decision Tree Regressor
dt_model = DecisionTreeRegressor(random_state=42)
dt_model.fit(X_train, y_train)
dt_predictions = dt_model.predict(X_test)
dt_mse = mean_squared_error(y_test, dt_predictions)
dt_r2 = r2_score(y_test, dt_predictions)

print("\nDecision Tree Regressor:")
print(f"Mean Squared Error: {dt_mse}")
print(f"R2 Score: {dt_r2}")

# Random Forest Regressor
rf_model = RandomForestRegressor(random_state=42, n_estimators=100)
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)
rf_mse = mean_squared_error(y_test, rf_predictions)
rf_r2 = r2_score(y_test, rf_predictions)

print("\nRandom Forest Regressor:")
print(f"Mean Squared Error: {rf_mse}")
print(f"R2 Score: {rf_r2}")
```

```
Linear Regression:
Mean Squared Error: 3.2654475357472976
R2 Score: 0.924490236084747

Decision Tree Regressor:
Mean Squared Error: 0.7673267326732673
R2 Score: 0.982256441178204

Random Forest Regressor:
Mean Squared Error: 0.3940965346534653
R2 Score: 0.9908869654264121
```