Task 2(a)

In [1]:
```
import pandas as pd

# Replace 'file_path.csv' with the actual file path of your CSV file
file_path = "C:/Users/Jaskaran/Downloads/kbopitchingdata.csv"

# Use the pd.read_csv() function to read the CSV file into a DataFrame
df_JS = pd.read_csv(file_path)
```

Displaying the first few rows of the dataset

In [2]:
```
print("Preview of the dataset:")
print(df_JS.head())
```

```
Preview of the dataset:
   id  year           team  average_age  runs_per_game  wins  losses  \
0   1  2021       LG Twins         26.3           3.90    72      57
1   2  2021         KT Wiz         28.4           4.06    75      59
2   3  2021   Doosan Bears         27.5           4.57    70      65
3   4  2021  Samsung Lions         28.8           4.57    75      59
4   5  2021       NC Dinos         27.7           4.80    67      67

   win_loss_percentage   ERA  run_average_9  ...  hit_batter  balks  \
0                0.558  3.57           3.96  ...          97    5.0
1                0.560  3.67           4.17  ...          42    1.0
2                0.519  4.28           4.66  ...          73    7.0
3                0.560  4.29           4.70  ...          51    3.0
4                0.500  4.50           4.95  ...          77    8.0

   wild_pitches  batters_faced   WHIP  hits_9  homeruns_9  walks_9  \
0          43.0           5416  1.312     8.0         0.6      3.9
1          56.0           5359  1.316     8.4         0.6      3.5
2          51.0           5596  1.487     9.2         0.7      4.2
3          56.0           5496  1.450     9.3         0.9      3.8
4          74.0           5575  1.476     9.1         0.9      4.2

   strikeouts_9  strikeout_walk
0           7.6            1.96
1           7.5            2.16
2           7.4            1.77
3           7.4            1.96
4           7.5            1.79

[5 rows x 34 columns]
```

Understanding basic information about the dataset

In [3]:
```
print("\nDataset information:")
print(df_JS.info())
```

```
Dataset information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 323 entries, 0 to 322
Data columns (total 34 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   id                   323 non-null    int64  
 1   year                 323 non-null    int64  
 2   team                 323 non-null    object 
 3   average_age          323 non-null    float64
 4   runs_per_game        323 non-null    float64
 5   wins                 323 non-null    int64  
 6   losses               323 non-null    int64  
 7   win_loss_percentage  323 non-null    float64
 8   ERA                  323 non-null    float64
 9   run_average_9        323 non-null    float64
 10  games                323 non-null    int64  
 11  games_started        184 non-null    float64
 12  games_finished       184 non-null    float64
 13  complete_game        323 non-null    int64  
 14  shutouts             323 non-null    int64  
 15  saves                323 non-null    int64  
 16  innings_pitched      323 non-null    float64
 17  hits                 323 non-null    int64  
 18  runs                 323 non-null    int64  
 19  earned_runs          323 non-null    int64  
 20  home_runs            323 non-null    int64  
 21  walks                323 non-null    int64  
 22  intentional_walks    184 non-null    float64
 23  strikeouts           323 non-null    int64  
 24  hit_batter           323 non-null    int64  
 25  balks                184 non-null    float64
 26  wild_pitches         184 non-null    float64
 27  batters_faced        323 non-null    int64  
 28  WHIP                 323 non-null    float64
 29  hits_9               323 non-null    float64
 30  homeruns_9           323 non-null    float64
 31  walks_9              323 non-null    float64
 32  strikeouts_9         323 non-null    float64
 33  strikeout_walk       323 non-null    float64
dtypes: float64(17), int64(16), object(1)
memory usage: 85.9+ KB
None
```

Printing summary statistics of the numerical columns

In [4]:

```
print("\nSummary Statistics:")
print(df_JS.describe())
```

```
Summary Statistics:
               id         year  average_age  runs_per_game        wins  \
count  323.000000   323.000000   323.000000     323.000000  323.000000   
mean   162.000000  2002.944272    26.886687       4.621858   62.507740   
std     93.386294    11.501957     1.608472       0.734223   12.508225   
min      1.000000  1982.000000    23.300000       2.980000   15.000000   
25%     81.500000  1993.000000    25.700000       4.040000   54.000000   
50%    162.000000  2003.000000    26.900000       4.620000   63.000000   
75%    242.500000  2013.000000    28.000000       5.060000   71.000000   
max    323.000000  2021.000000    32.400000       7.180000   93.000000   

           losses  win_loss_percentage         ERA  run_average_9       games  \
count  323.000000           323.000000  323.000000     323.000000  323.000000   
mean    62.482972             0.500043    4.207833       4.689783  128.142415   
std     12.446988             0.087081    0.750075       0.768520   12.996350   
min     24.000000             0.188000    2.540000       3.030000   80.000000   
25%     53.000000             0.444500    3.630000       4.090000  126.000000   
50%     62.000000             0.504000    4.220000       4.670000  128.000000   
75%     71.500000             0.561500    4.700000       5.180000  133.000000   
max     97.000000             0.706000    6.350000       7.470000  144.000000   

       ...  hit_batter       balks  wild_pitches  batters_faced        WHIP  \
count  ...  323.000000  184.000000    184.000000     323.000000  323.000000   
mean   ...   66.294118    3.902174     56.983696    4935.439628    1.400588   
std    ...   20.035144    2.244896     15.775223     574.410547    0.115192   
min    ...   29.000000    0.000000     21.000000    2830.000000    1.106000   
25%    ...   51.500000    2.000000     46.000000    4697.500000    1.314000   
50%    ...   66.000000    4.000000     55.000000    4969.000000    1.402000   
75%    ...   79.000000    5.000000     66.250000    5264.000000    1.478000   
max    ...  120.000000   11.000000    103.000000    5937.000000    1.761000   

           hits_9  homeruns_9     walks_9  strikeouts_9  strikeout_walk  
count  323.000000  323.000000  323.000000    323.000000      323.000000  
mean     9.063777    0.836223    3.543963      5.943653        1.703282  
std      0.784845    0.246000    0.495432      1.194754        0.392679  
min      7.300000    0.300000    2.400000      2.300000        0.560000  
25%      8.500000    0.700000    3.200000      5.100000        1.445000  
50%      9.000000    0.800000    3.500000      6.200000        1.700000  
75%      9.500000    1.000000    3.900000      6.800000        1.960000  
max     11.600000    1.500000    5.100000      8.400000        2.820000  

[8 rows x 33 columns]
```

Check for missing values in the entire data frame

True indicating missing values and False indicates non missing values

In [5]:
```
missing_values = df_JS.isnull()
print(missing_values)
```

```
        id   year   team  average_age  runs_per_game   wins  losses  \
0    False  False  False        False          False  False   False   
1    False  False  False        False          False  False   False   
2    False  False  False        False          False  False   False   
3    False  False  False        False          False  False   False   
4    False  False  False        False          False  False   False   
..     ...    ...    ...          ...            ...    ...     ...   
318  False  False  False        False          False  False   False   
319  False  False  False        False          False  False   False   
320  False  False  False        False          False  False   False   
321  False  False  False        False          False  False   False   
322  False  False  False        False          False  False   False   

     win_loss_percentage    ERA  run_average_9  ...  hit_batter  balks  \
0                  False  False          False  ...       False  False   
1                  False  False          False  ...       False  False   
2                  False  False          False  ...       False  False   
3                  False  False          False  ...       False  False   
4                  False  False          False  ...       False  False   
..                   ...    ...            ...  ...         ...    ...   
318                False  False          False  ...       False   True   
319                False  False          False  ...       False   True   
320                False  False          False  ...       False   True   
321                False  False          False  ...       False   True   
322                False  False          False  ...       False   True   

     wild_pitches  batters_faced   WHIP  hits_9  homeruns_9  walks_9  \
0           False          False  False   False       False    False   
1           False          False  False   False       False    False   
2           False          False  False   False       False    False   
3           False          False  False   False       False    False   
4           False          False  False   False       False    False   
..            ...            ...    ...     ...         ...      ...   
318          True          False  False   False       False    False   
319          True          False  False   False       False    False   
320          True          False  False   False       False    False   
321          True          False  False   False       False    False   
322          True          False  False   False       False    False   

     strikeouts_9  strikeout_walk  
0           False           False  
1           False           False  
2           False           False  
3           False           False  
4           False           False  
..            ...             ...  
318         False           False  
319         False           False  
320         False           False  
321         False           False  
322         False           False  

[323 rows x 34 columns]
```

Get the count of non missing values in each column

In [6]:

```
missing_count = df_JS.isnull().sum()
print(missing_count)
```

```
id                       0
year                     0
team                     0
average_age              0
runs_per_game            0
wins                     0
losses                   0
win_loss_percentage      0
ERA                      0
run_average_9            0
games                    0
games_started          139
games_finished         139
complete_game            0
shutouts                 0
saves                    0
innings_pitched          0
hits                     0
runs                     0
earned_runs              0
home_runs                0
walks                    0
intentional_walks      139
strikeouts               0
hit_batter               0
balks                  139
wild_pitches           139
batters_faced            0
WHIP                     0
hits_9                   0
homeruns_9               0
walks_9                  0
strikeouts_9             0
strikeout_walk           0
dtype: int64
```

Printing the data types for all the columns

In [7]:

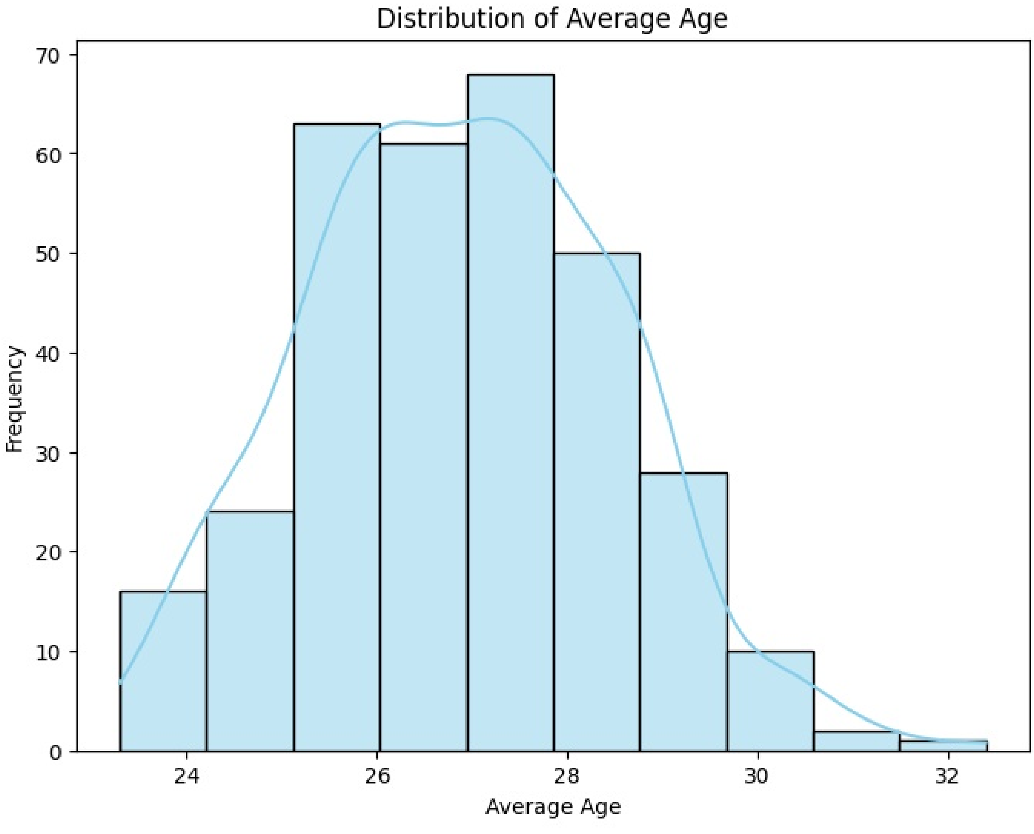
```
data_types = df_JS.dtypes
print(data_types)
```

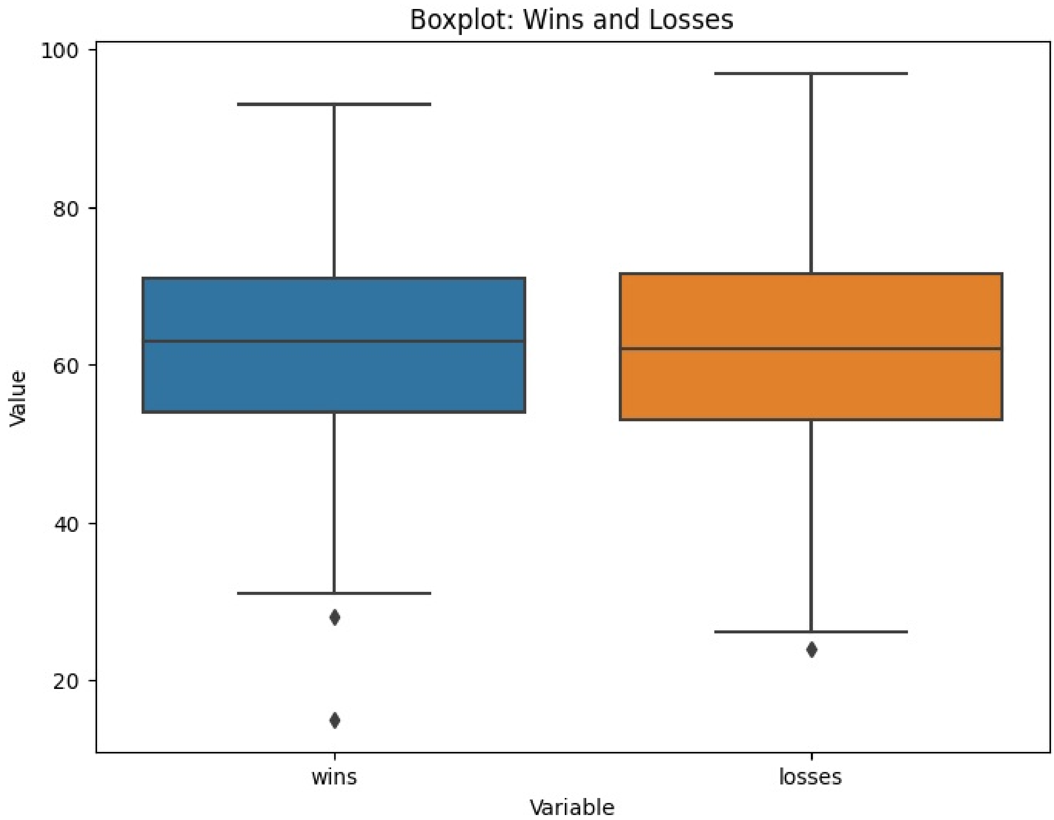
```
id                       int64
year                     int64
team                    object
average_age            float64
runs_per_game          float64
wins                     int64
losses                   int64
win_loss_percentage    float64
ERA                    float64
run_average_9          float64
games                    int64
games_started          float64
games_finished         float64
complete_game            int64
shutouts                 int64
saves                    int64
innings_pitched        float64
hits                     int64
runs                     int64
earned_runs              int64
home_runs                int64
walks                    int64
intentional_walks      float64
strikeouts               int64
hit_batter               int64
balks                  float64
wild_pitches           float64
batters_faced            int64
WHIP                   float64
hits_9                 float64
homeruns_9             float64
walks_9                float64
strikeouts_9           float64
strikeout_walk         float64
dtype: object
```

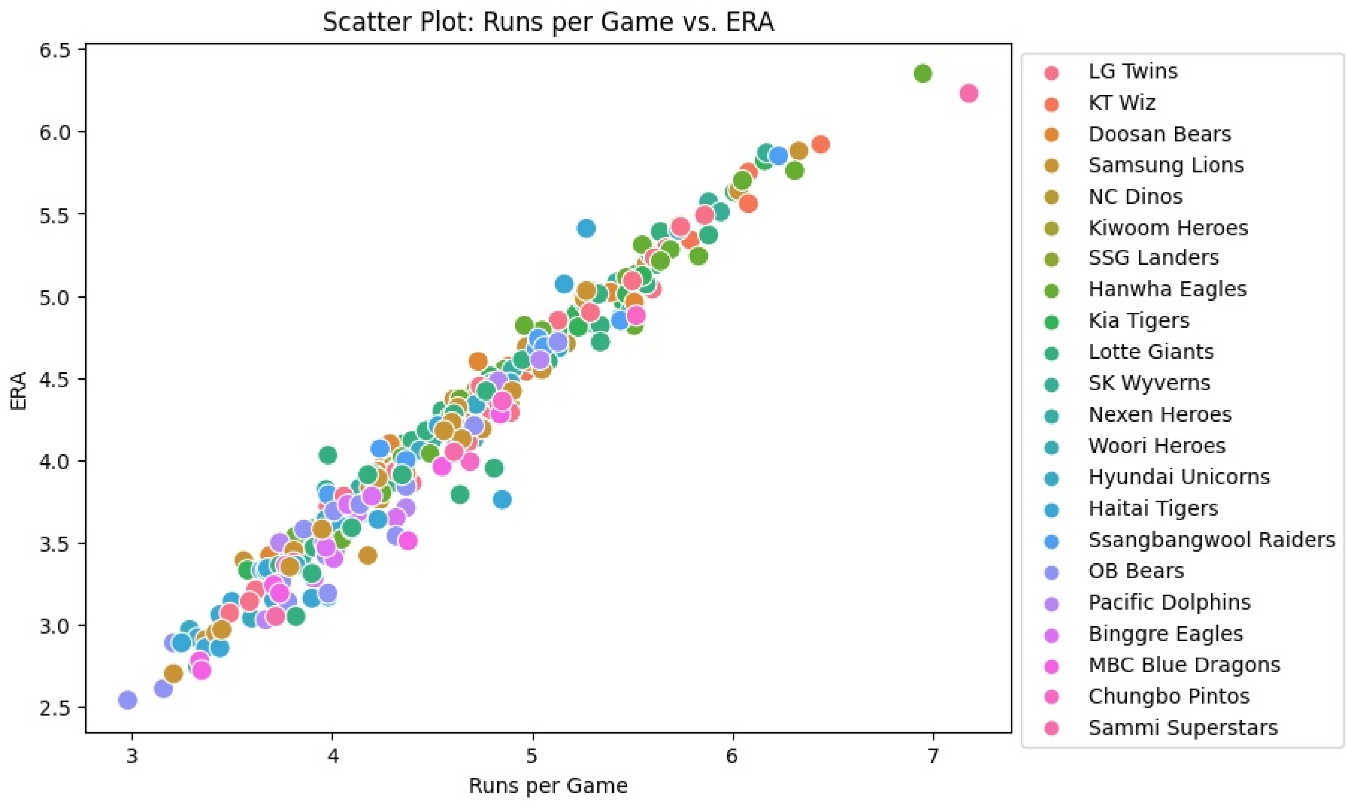
Visualize data distribution using Box plot, Histogram and scatter plot

In [8]:

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv("C:/Users/Jaskaran/Downloads/kbopitchingdata.csv")

# Histogram: Distribution of 'average_age'
plt.figure(figsize=(8, 6))
sns.histplot(data['average_age'], bins=10, kde=True, color='skyblue')
plt.title('Distribution of Average Age')
plt.xlabel('Average Age')
plt.ylabel('Frequency')
plt.show()

# Boxplot: Comparing 'wins' and 'losses'
plt.figure(figsize=(8, 6))
sns.boxplot(data=data[['wins', 'losses']])
plt.title('Boxplot: Wins and Losses')
plt.xlabel('Variable')
plt.ylabel('Value')
plt.show()

# Scatter plot: 'runs_per_game' vs 'ERA'
plt.figure(figsize=(8, 6))
sns.scatterplot(data=data, x='runs_per_game', y='ERA', hue='team', s=100)
plt.title('Scatter Plot: Runs per Game vs. ERA')
plt.xlabel('Runs per Game')
plt.ylabel('ERA')
plt.legend(loc='upper left', bbox_to_anchor=(1, 1))
plt.show()
()
```




Boxplot: Wins and Losses
Value
Variable
wins
losses
100
80
60
40
20


Scatter Plot: Runs per Game vs. ERA
ERA
Runs per Game
LG Twins
KT Wiz
Doosan Bears
Samsung Lions
NC Dinos
Kiwoom Heroes
SSG Landers
Hanwha Eagles
Kia Tigers
Lotte Giants
SK Wyverns
Nexen Heroes
Woori Heroes
Hyundai Unicorns
Haitai Tigers
Ssangbangwool Raiders
OB Bears
Pacific Dolphins
Binggre Eagles
MBC Blue Dragons
Chungbo Pintos
Sammi Superstars

Out[8]: ()

Task 3(a)

In [10]:
```python
#3.1
import pandas as pd

# Load the data from the CSV file
data = pd.read_csv("C:/Users/Jaskaran/Downloads/kbopitchingdata.csv")

# Remove unnecessary columns
columns_to_drop = ['hits_9', 'homeruns_9', 'walks_9', 'strikeouts_9']
data = data.drop(columns=columns_to_drop)

# Clean team names by removing special characters
data['team'] = data['team'].str.replace('[^a-zA-Z\s]', '')

# Handle missing values by filling with appropriate values
data['games_started'].fillna(0, inplace=True)
data['games_finished'].fillna(0, inplace=True)
data['intentional_walks'].fillna(0, inplace=True)
data['balks'].fillna(0, inplace=True)
data['wild_pitches'].fillna(0, inplace=True)


# Define the path to save the preprocessed data
preprocessed_file_path = "C:/Users/Jaskaran/Desktop/Preprocessed_kbopitchingdata.csv"

# Save the preprocessed data to a CSV file
data.to_csv(preprocessed_file_path, index=False)
data=pd.read_csv(r"C:/Users/Jaskaran/Desktop/Preprocessed_kbopitchingdata.csv")

print("Data preprocessing completed and saved to:", preprocessed_file_path)
print(data.head())
```

```
Data preprocessing completed and saved to: C:/Users/Jaskaran/Desktop/Preprocessed_kbopitchingdata.csv
   id  year           team  average_age  runs_per_game  wins  losses  \
0   1  2021       LG Twins         26.3           3.90    72      57   
1   2  2021         KT Wiz         28.4           4.06    75      59   
2   3  2021   Doosan Bears         27.5           4.57    70      65   
3   4  2021  Samsung Lions         28.8           4.57    75      59   
4   5  2021       NC Dinos         27.7           4.80    67      67   

   win_loss_percentage   ERA  run_average_9  ...  home_runs  walks  \
0                0.558  3.57           3.96  ...         79    542   
1                0.560  3.67           4.17  ...         85    486   
2                0.519  4.28           4.66  ...        104    586   
3                0.560  4.29           4.70  ...        129    526   
4                0.500  4.50           4.95  ...        122    585   

   intentional_walks  strikeouts  hit_batter  balks  wild_pitches  \
0               17.0        1062          97    5.0          43.0   
1               18.0        1051          42    1.0          56.0   
2               16.0        1037          73    7.0          51.0   
3               13.0        1031          51    3.0          56.0   
4               14.0        1046          77    8.0          74.0   

   batters_faced   WHIP  strikeout_walk  
0           5416  1.312            1.96  
1           5359  1.316            2.16  
2           5596  1.487            1.77  
3           5496  1.450            1.96  
4           5575  1.476            1.79  

[5 rows x 30 columns]
```

Task3(b)

In [14]:
```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler

# Load the preprocessed data from the CSV file
preprocessed_file_path = "C:/Users/Jaskaran/Desktop/Preprocessed_kbopitchingdata.csv"
data = pd.read_csv(preprocessed_file_path)

# Missing values were handled already in the preprocessing step.

# 1. Drop unnecessary columns
```

```python
data.drop(['id', 'year', 'team'], axis=1, inplace=True)


# Identifying Outliers
# Let's identify outliers using the Interquartile Range (IQR) method.


def identify_outliers(column):
    Q1 = np.percentile(column, 25)
    Q3 = np.percentile(column, 75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return (column < lower_bound) | (column > upper_bound)

outliers = identify_outliers(data['ERA'])
data = data[~outliers]

# Feature Engineering

# Let's create a new feature 'win percentage' which is wins / games.
data['Wins_Percentage'] = data['wins'] / data['games']
print(data.head())


# Standardize Numeric Features
numeric_columns = ['average_age', 'runs_per_game', 'wins', 'losses', 'win_loss_percentage', 'ERA', 'run_average_
                   'games_finished','complete_game', 'shutouts', 'saves', 'innings_pitched', 'hits', 'runs', 'ea
                   'intentional_walks', 'strikeouts', 'hit_batter', 'balks', 'wild_pitches', 'batters_faced', 'W

scaler = StandardScaler()
data[numeric_columns] = scaler.fit_transform(data[numeric_columns])


# Save the preprocessed and engineered data to a new CSV file
final_data_file_path = "C:/Users/Jaskaran/Desktop/Final_kbopitchingdata.csv"
data.to_csv(final_data_file_path, index=False)

print("Data preprocessing, outlier removal, feature engineering, and scaling completed.")
print("Final data saved to:", final_data_file_path)
```

```
   average_age  runs_per_game  wins  losses  win_loss_percentage   ERA  \
0         26.3           3.90    72      57                0.558  3.57
1         28.4           4.06    75      59                0.560  3.67
2         27.5           4.57    70      65                0.519  4.28
3         28.8           4.57    75      59                0.560  4.29
4         27.7           4.80    67      67                0.500  4.50

   run_average_9  games  games_started  games_finished  ...  walks  \
0           3.96    143          143.0           143.0  ...    542
1           4.17    143          143.0           141.0  ...    486
2           4.66    143          143.0           141.0  ...    586
3           4.70    143          143.0           141.0  ...    526
4           4.95    143          143.0           140.0  ...    585

   intentional_walks  strikeouts  hit_batter  balks  wild_pitches  \
0               17.0        1062          97    5.0          43.0
1               18.0        1051          42    1.0          56.0
2               16.0        1037          73    7.0          51.0
3               13.0        1031          51    3.0          56.0
4               14.0        1046          77    8.0          74.0

   batters_faced   WHIP  strikeout_walk  Wins_Percentage
0           5416  1.312            1.96         0.503497
1           5359  1.316            2.16         0.524476
2           5596  1.487            1.77         0.489510
3           5496  1.450            1.96         0.524476
4           5575  1.476            1.79         0.468531

[5 rows x 28 columns]
Data preprocessing, outlier removal, feature engineering, and scaling completed.
Final data saved to: C:/Users/Jaskaran/Desktop/Final_kbopitchingdata.csv
```

Task 4(a)

Implementing Random Forest Classifier Model Below.

In [18]:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
```

```python
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load the dataset
data = pd.read_csv('C:/Users/Jaskaran/Desktop/Final_kbopitchingdata.csv')

# Convert Wins_Percentage into classes
bins = [-float('inf'), 0.4, 0.6, float('inf')]
labels = ['low', 'medium', 'high']
data['Wins_Class'] = pd.cut(data['Wins_Percentage'], bins=bins, labels=labels)

# Split the dataset into features (X) and target (y)
X = data.drop(columns=['Wins_Percentage', 'Wins_Class'])
y = data['Wins_Class']

# Split the dataset into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Random Forest Classifier model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Evaluate Random Forest Classifier model
rf_pred = rf_model.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_pred)
rf_report = classification_report(y_test, rf_pred)
rf_conf_matrix = confusion_matrix(y_test, rf_pred)

# Print evaluation results
print("Random Forest Classifier Accuracy:", rf_accuracy)
print("Random Forest Classifier Classification Report:")
print(rf_report)
print("Random Forest Classifier Confusion Matrix:")
print(rf_conf_matrix)
```

```
Random Forest Classifier Accuracy: 0.9692307692307692
Random Forest Classifier Classification Report:
              precision    recall  f1-score   support

        high       1.00      1.00      1.00         4
         low       1.00      0.80      0.89        10
      medium       0.96      1.00      0.98        51

    accuracy                           0.97        65
   macro avg       0.99      0.93      0.96        65
weighted avg       0.97      0.97      0.97        65

Random Forest Classifier Confusion Matrix:
[[ 4  0  0]
 [ 0  8  2]
 [ 0  0 51]]
```

In [19]:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load the dataset
data = pd.read_csv('C:/Users/Jaskaran/Desktop/Final_kbopitchingdata.csv')

# Convert Wins_Percentage into classes
bins = [-float('inf'), 0.4, 0.6, float('inf')]
labels = ['low', 'medium', 'high']
data['Wins_Class'] = pd.cut(data['Wins_Percentage'], bins=bins, labels=labels)

# Split the dataset into features (X) and target (y)
X = data.drop(columns=['Wins_Percentage', 'Wins_Class'])
y = data['Wins_Class']

# Split the dataset into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train SVM model
svm_model = SVC(kernel='linear', random_state=42)
svm_model.fit(X_train, y_train)

# Evaluate SVM model
svm_pred = svm_model.predict(X_test)
svm_accuracy = accuracy_score(y_test, svm_pred)
svm_report = classification_report(y_test, svm_pred)
svm_conf_matrix = confusion_matrix(y_test, svm_pred)

# Print evaluation results
print("SVM Classifier Accuracy:", svm_accuracy)
print("SVM Classifier Classification Report:")
```

```
print(svm_report)
print("SVM Classifier Confusion Matrix:")
print(svm_conf_matrix)
```

```
SVM Classifier Accuracy: 0.9538461538461539
SVM Classifier Classification Report:
              precision    recall  f1-score   support

        high       1.00      0.75      0.86         4
         low       1.00      0.80      0.89        10
      medium       0.94      1.00      0.97        51

    accuracy                           0.95        65
   macro avg       0.98      0.85      0.91        65
weighted avg       0.96      0.95      0.95        65

SVM Classifier Confusion Matrix:
[[ 3  0  1]
 [ 0  8  2]
 [ 0  0 51]]
```

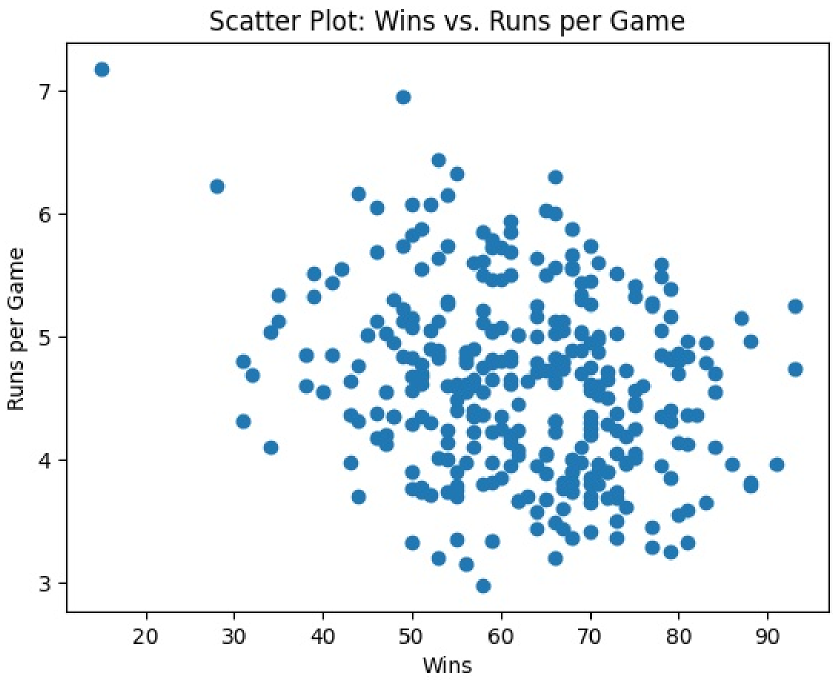
In [21]:
```
# Print evaluation results for Random Forest Classifier model
print("Random Forest Classifier Accuracy:", rf_accuracy)
print("Random Forest Classifier Classification Report:")
print(rf_report)
print("Random Forest Classifier Confusion Matrix:")
print(rf_conf_matrix)

# Train SVM model
svm_model = SVC(kernel='linear', random_state=42)
svm_model.fit(X_train, y_train)

# Evaluate SVM model
svm_pred = svm_model.predict(X_test)
svm_accuracy = accuracy_score(y_test, svm_pred)
svm_report = classification_report(y_test, svm_pred)
svm_conf_matrix = confusion_matrix(y_test, svm_pred)

# Print evaluation results for SVM Classifier model
print("\nSVM Classifier Accuracy:", svm_accuracy)
print("SVM Classifier Classification Report:")
print(svm_report)
print("SVM Classifier Confusion Matrix:")
print(svm_conf_matrix)
```

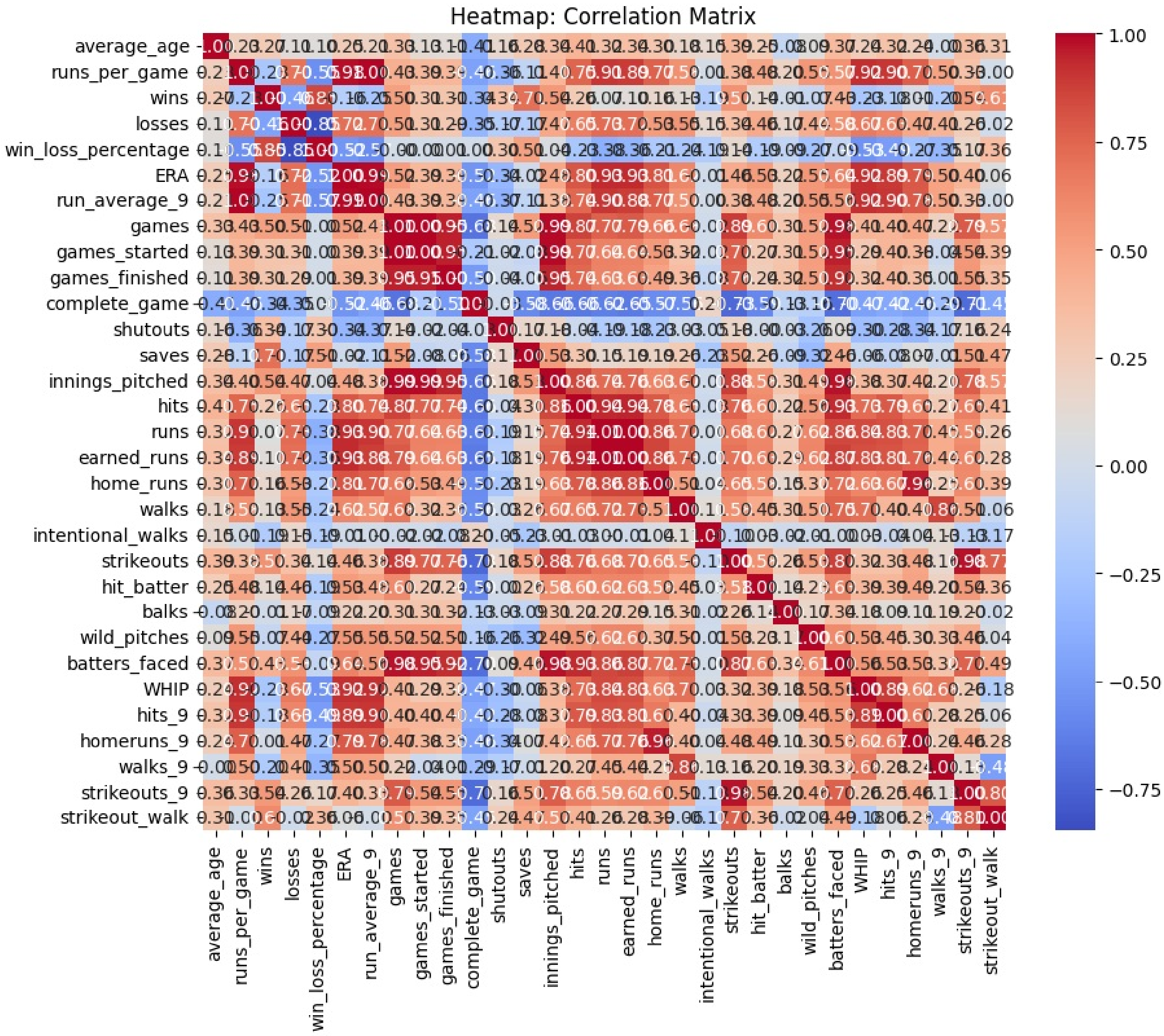
```
Random Forest Classifier Accuracy: 0.9692307692307692
Random Forest Classifier Classification Report:
              precision    recall  f1-score   support

        high       1.00      1.00      1.00         4
         low       1.00      0.80      0.89        10
      medium       0.96      1.00      0.98        51

    accuracy                           0.97        65
   macro avg       0.99      0.93      0.96        65
weighted avg       0.97      0.97      0.97        65

Random Forest Classifier Confusion Matrix:
[[ 4  0  0]
 [ 0  8  2]
 [ 0  0 51]]

SVM Classifier Accuracy: 0.9538461538461539
SVM Classifier Classification Report:
              precision    recall  f1-score   support

        high       1.00      0.75      0.86         4
         low       1.00      0.80      0.89        10
      medium       0.94      1.00      0.97        51

    accuracy                           0.95        65
   macro avg       0.98      0.85      0.91        65
weighted avg       0.96      0.95      0.95        65

SVM Classifier Confusion Matrix:
[[ 3  0  1]
 [ 0  8  2]
 [ 0  0 51]]
```

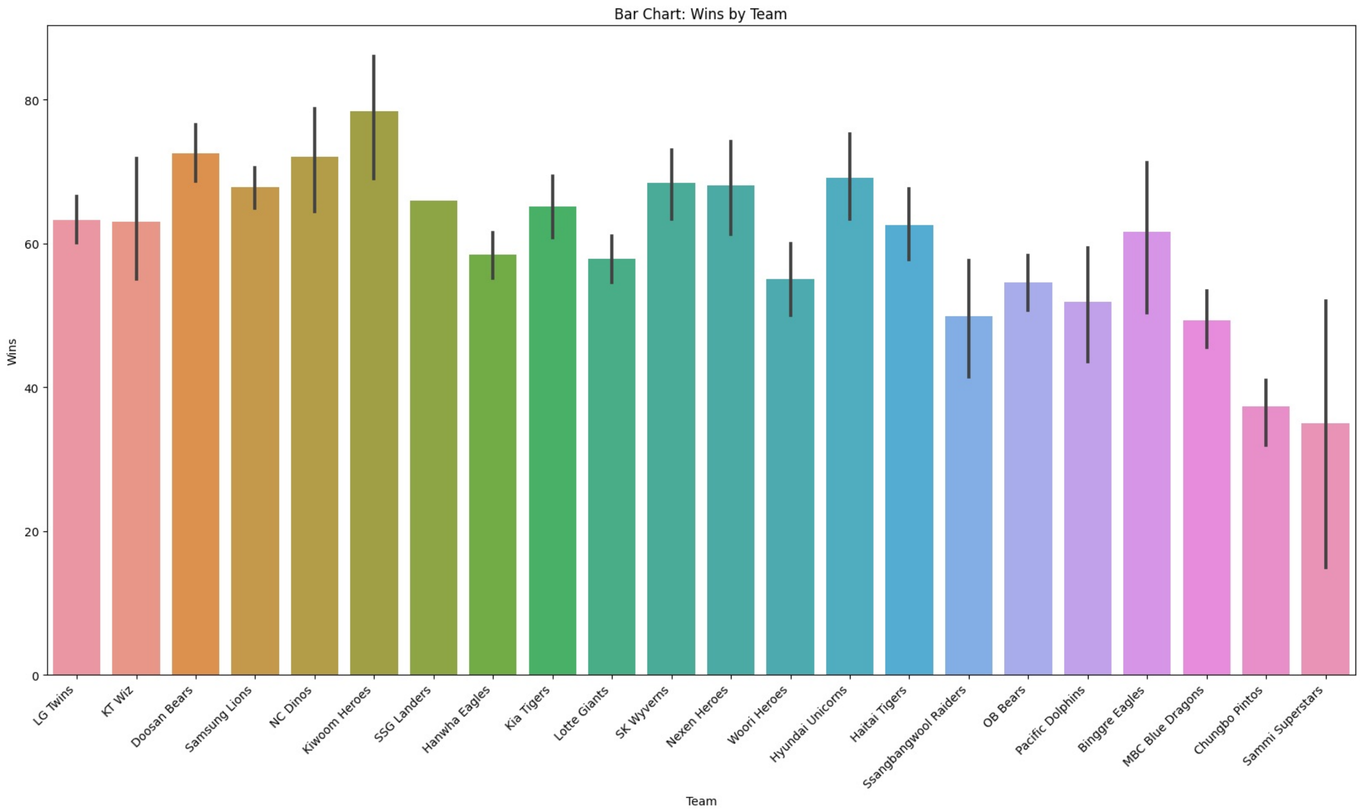
Task5 Visualization with Python (Sctterplot, Heatmap & Barchart)

In [24]:
```
import pandas as pd
```

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
data = pd.read_csv('C:/Users/Jaskaran/Downloads/kbopitchingdata.csv')

# Scatter Plot: Wins vs. Runs_Per_Game
plt.scatter(data['wins'], data['runs_per_game'])
plt.xlabel('Wins')
plt.ylabel('Runs per Game')
plt.title('Scatter Plot: Wins vs. Runs per Game')
plt.show()

# Heatmap: Correlation Matrix (excluding non-numeric columns)
numeric_data = data.drop(columns=['id', 'year', 'team'])
corr_matrix = numeric_data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Heatmap: Correlation Matrix')
plt.show()

# Bar Chart: Team vs. Wins
plt.figure(figsize=(20, 10))
sns.barplot(x='team', y='wins', data=data)
plt.xticks(rotation=45, ha='right')
plt.xlabel('Team')
plt.ylabel('Wins')
plt.title('Bar Chart: Wins by Team')
plt.show()
```




Heatmap: Correlation Matrix
average_age
runs_per_game
wins
losses
win_loss_percentage
ERA
run_average_9
games
games_started
games_finished
complete_game
shutouts
saves
innings_pitched
hits
runs
earned_runs
home_runs
walks
intentional_walks
strikeouts
hit_batter
balks
wild_pitches
batters_faced
WHIP
hits_9
homeruns_9
walks_9
strikeouts_9
strikeout_walk
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75



In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js