DCS 540

# Week 5-6, Week 7-8, Week 9-10, Week 11-12

## Project Milestone 2, Milestone 3, Milestone 4, Milestone 5

Perform at least 5 data transformation and/or cleansing steps to your flat file data.

### Name: Aniruddha Joshi

### Date: Nov 18, 2023

# Table of Contents



# Milestone 2

In [ ]:
```
#Defining Libraries required for import
import numpy as np
import random
import matplotlib.pyplot as plt
import pandas as pd
from bs4 import BeautifulSoup
import sqlite3
```

```
import seaborn as sns
import requests
import urllib.request, urllib.parse, urllib.error
import ssl
import re
import json
#Display the plots inline
#%matplotlib inline
```

In [ ]:
```
# Load the dataset 1
deliveries_df = pd.read_csv('deliveries.csv')
```

In [ ]:
```
# Load the dataset 2
matches_df = pd.read_csv('matches.csv')
```

In [ ]:
```
# Load the dataset 3
cricket_data_df = pd.read_csv('Cricket-Data.csv')
```

```
C:\Users\joshi\AppData\Local\Temp\ipykernel_24088\3835400717.py:2: DtypeWarning: Columns (101,107,108,140,141,145,146,1
47,153,154,158,159,160) have mixed types. Specify dtype option on import or set low_memory=False.
  cricket_data_df = pd.read_csv('Cricket-Data.csv')
```

In [ ]:
```
cricket_data_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90308 entries, 0 to 90307
Columns: 177 entries, Unnamed: 0 to BOWLING_T20s_10
dtypes: float64(132), int64(2), object(43)
memory usage: 122.0+ MB
```

## get basic information on datasets

In [ ]:
```
deliveries_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150460 entries, 0 to 150459
Data columns (total 21 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   match_id          150460 non-null  int64 
 1   inning            150460 non-null  int64 
 2   batting_team      150460 non-null  object
 3   bowling_team      150460 non-null  object
 4   over              150460 non-null  int64 
 5   ball              150460 non-null  int64 
 6   batsman           150460 non-null  object
 7   non_striker       150460 non-null  object
 8   bowler            150460 non-null  object
 9   is_super_over     150460 non-null  int64 
 10  wide_runs         150460 non-null  int64 
 11  bye_runs          150460 non-null  int64 
 12  legbye_runs       150460 non-null  int64 
 13  noball_runs       150460 non-null  int64 
 14  penalty_runs      150460 non-null  int64 
 15  batsman_runs      150460 non-null  int64 
 16  extra_runs        150460 non-null  int64 
 17  total_runs        150460 non-null  int64 
 18  player_dismissed  7438 non-null    object
 19  dismissal_kind    7438 non-null    object
 20  fielder           5369 non-null    object
dtypes: int64(13), object(8)
memory usage: 24.1+ MB
```

In [ ]:
```
matches_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 636 entries, 0 to 635
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   id               636 non-null    int64  
 1   season           636 non-null    int64  
 2   city             629 non-null    object 
 3   date             636 non-null    object 
 4   team1            636 non-null    object 
 5   team2            636 non-null    object 
 6   toss_winner      636 non-null    object 
 7   toss_decision    636 non-null    object 
 8   result           636 non-null    object 
 9   dl_applied       636 non-null    int64  
 10  winner           633 non-null    object 
 11  win_by_runs      636 non-null    int64  
 12  win_by_wickets   636 non-null    int64  
 13  player_of_match  633 non-null    object 
 14  venue            636 non-null    object 
 15  umpire1          635 non-null    object 
 16  umpire2          635 non-null    object 
 17  umpire3          0 non-null      float64
dtypes: float64(1), int64(5), object(12)
memory usage: 89.6+ KB
```

## Perform some initial stat on the database

In [ ]:
```
deliveries_df.describe().transpose()
```

Out[ ]:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **match_id** | 150460.0 | 318.281317 | 182.955531 | 1.0 | 161.0 | 319.0 | 476.0 | 636.0 |
| **inning** | 150460.0 | 1.482188 | 0.501768 | 1.0 | 1.0 | 1.0 | 2.0 | 4.0 |
| **over** | 150460.0 | 10.142649 | 5.674338 | 1.0 | 5.0 | 10.0 | 15.0 | 20.0 |
| **ball** | 150460.0 | 3.616483 | 1.807698 | 1.0 | 2.0 | 4.0 | 5.0 | 9.0 |
| **is_super_over** | 150460.0 | 0.000538 | 0.023196 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| **wide_runs** | 150460.0 | 0.037498 | 0.257398 | 0.0 | 0.0 | 0.0 | 0.0 | 5.0 |
| **bye_runs** | 150460.0 | 0.004885 | 0.114234 | 0.0 | 0.0 | 0.0 | 0.0 | 4.0 |
| **legbye_runs** | 150460.0 | 0.022232 | 0.200104 | 0.0 | 0.0 | 0.0 | 0.0 | 5.0 |
| **noball_runs** | 150460.0 | 0.004340 | 0.072652 | 0.0 | 0.0 | 0.0 | 0.0 | 5.0 |
| **penalty_runs** | 150460.0 | 0.000066 | 0.018229 | 0.0 | 0.0 | 0.0 | 0.0 | 5.0 |
| **batsman_runs** | 150460.0 | 1.222445 | 1.594509 | 0.0 | 0.0 | 1.0 | 1.0 | 6.0 |
| **extra_runs** | 150460.0 | 0.069022 | 0.349667 | 0.0 | 0.0 | 0.0 | 0.0 | 7.0 |
| **total_runs** | 150460.0 | 1.291466 | 1.583240 | 0.0 | 0.0 | 1.0 | 1.0 | 7.0 |

In [ ]:
```
matches_df.describe().transpose()
```

Out[ ]:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **id** | 636.0 | 318.500000 | 183.741666 | 1.0 | 159.75 | 318.5 | 477.25 | 636.0 |
| **season** | 636.0 | 2012.490566 | 2.773026 | 2008.0 | 2010.00 | 2012.0 | 2015.00 | 2017.0 |
| **dl_applied** | 636.0 | 0.025157 | 0.156726 | 0.0 | 0.00 | 0.0 | 0.00 | 1.0 |
| **win_by_runs** | 636.0 | 13.682390 | 23.908877 | 0.0 | 0.00 | 0.0 | 20.00 | 146.0 |
| **win_by_wickets** | 636.0 | 3.372642 | 3.420338 | 0.0 | 0.00 | 4.0 | 7.00 | 10.0 |
| **umpire3** | 0.0 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

## Changing the headers

In [ ]:
```
matches_df = matches_df.rename(columns={'win_by_runs': 'RunsWin', 'umpire3': 'Not Used'})
matches_df.transpose()
```

Out[ ]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
|---|---|---|---|---|---|---|---|---|---|---|
| **id** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
| **season** | 2017 | 2017 | 2017 | 2017 | 2017 | 2017 | 2017 | 2017 | 2017 | 2( |
| **city** | Hyderabad | Pune | Rajkot | Indore | Bangalore | Hyderabad | Mumbai | Indore | Pune | Mum |
| **date** | 2017-04-05 | 2017-04-06 | 2017-04-07 | 2017-04-08 | 2017-04-08 | 2017-04-09 | 2017-04-09 | 2017-04-10 | 2017-04-11 | 2017- |
| **team1** | Sunrisers Hyderabad | Mumbai Indians | Gujarat Lions | Rising Pune Supergiant | Royal Challengers Bangalore | Gujarat Lions | Kolkata Knight Riders | Royal Challengers Bangalore | Delhi Daredevils | Sunris Hyderab |
| **team2** | Royal Challengers Bangalore | Rising Pune Supergiant | Kolkata Knight Riders | Kings XI Punjab | Delhi Daredevils | Sunrisers Hyderabad | Mumbai Indians | Kings XI Punjab | Rising Pune Supergiant | Mum Indi |
| **toss_winner** | Royal Challengers Bangalore | Rising Pune Supergiant | Kolkata Knight Riders | Kings XI Punjab | Royal Challengers Bangalore | Sunrisers Hyderabad | Mumbai Indians | Royal Challengers Bangalore | Rising Pune Supergiant | Mum Indi |
| **toss_decision** | field | field | field | field | bat | field | field | bat | field | fi |
| **result** | normal | normal | normal | normal | normal | normal | normal | normal | normal | nor |
| **dl_applied** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| **winner** | Sunrisers Hyderabad | Rising Pune Supergiant | Kolkata Knight Riders | Kings XI Punjab | Royal Challengers Bangalore | Sunrisers Hyderabad | Mumbai Indians | Kings XI Punjab | Delhi Daredevils | Mum Indi |
| **RunsWin** | 35 | 0 | 0 | 0 | 15 | 0 | 0 | 0 | 97 | |
| **win_by_wickets** | 0 | 7 | 10 | 6 | 0 | 9 | 4 | 8 | 0 | |
| **player_of_match** | Yuvraj Singh | SPD Smith | CA Lynn | GJ Maxwell | KM Jadhav | Rashid Khan | N Rana | AR Patel | SV Samson | JJ Bum |
| **venue** | Rajiv Gandhi International Stadium, Uppal | Maharashtra Cricket Association Stadium | Saurashtra Cricket Association Stadium | Holkar Cricket Stadium | M Chinnaswamy Stadium | Rajiv Gandhi International Stadium, Uppal | Wankhede Stadium | Holkar Cricket Stadium | Maharashtra Cricket Association Stadium | Wankhe Stadi |
| **umpire1** | AY Dandekar | A Nand Kishore | Nitin Menon | AK Chaudhary | NaN | A Deshmukh | Nitin Menon | AK Chaudhary | AY Dandekar | N Mer |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
|---|---|---|---|---|---|---|---|---|---|---|
| **umpire2** | NJ Llong | S Ravi | CK Nandan | C Shamshuddin | NaN | NJ Llong | CK Nandan | C Shamshuddin | S Ravi | Nanc |
| **Not Used** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | N |

In [ ]:
```
#Reverting back the column names
matches_df = matches_df.rename(columns={'RunsWin':'win_by_runs', 'Not Used' :'umpire3'})
```

## Identify outliers and bad data

In [ ]:
```
# Identifying the bad data
print(deliveries_df.isna().sum())
```

```
match_id                0
inning                  0
batting_team            0
bowling_team            0
over                    0
ball                    0
batsman                 0
non_striker             0
bowler                  0
is_super_over           0
wide_runs               0
bye_runs                0
legbye_runs             0
noball_runs             0
penalty_runs            0
batsman_runs            0
extra_runs              0
total_runs              0
player_dismissed    143022
dismissal_kind      143022
fielder             145091
dtype: int64
```

In [ ]:
```
# Identifying the bad data
print(matches_df.isna().sum())
```

```
id                 0
season             0
city               7
date               0
team1              0
team2              0
toss_winner        0
toss_decision      0
result             0
dl_applied         0
winner             3
win_by_runs        0
win_by_wickets     0
player_of_match    3
venue              0
umpire1            1
umpire2            1
umpire3          636
dtype: int64
```

In [ ]:
```
#Printing the data having null records
matches_df[matches_df.city.isna()]
```

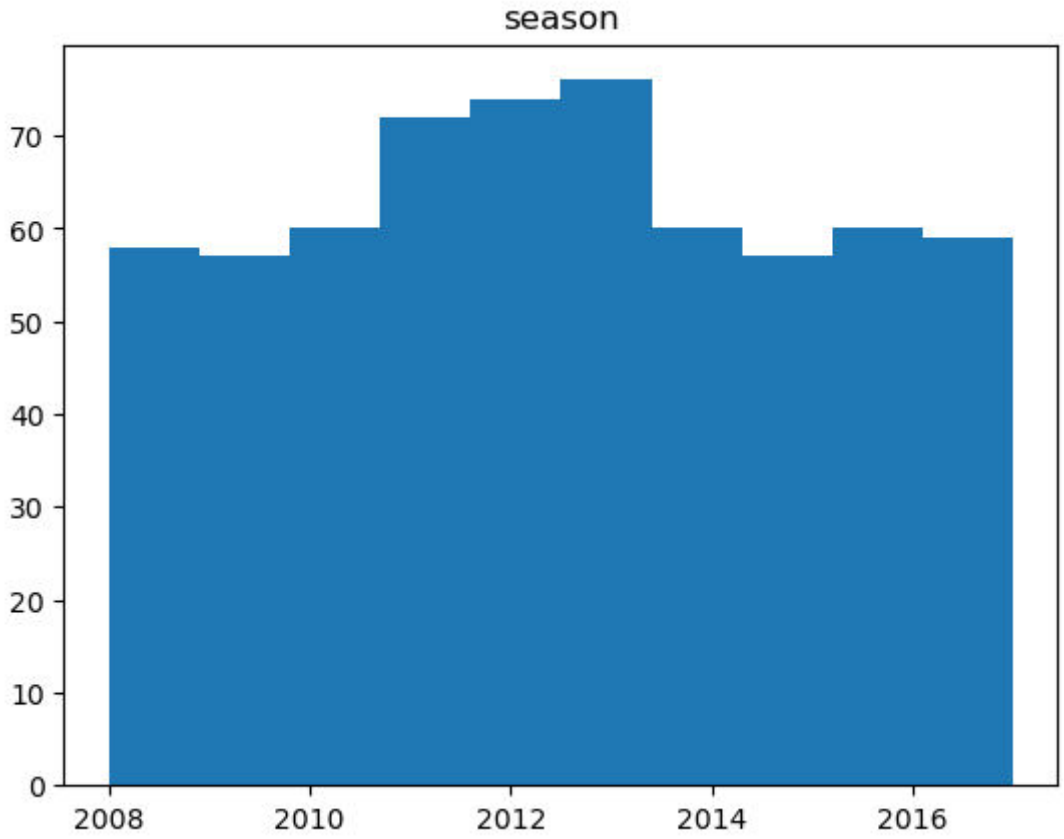
Out[ ]:

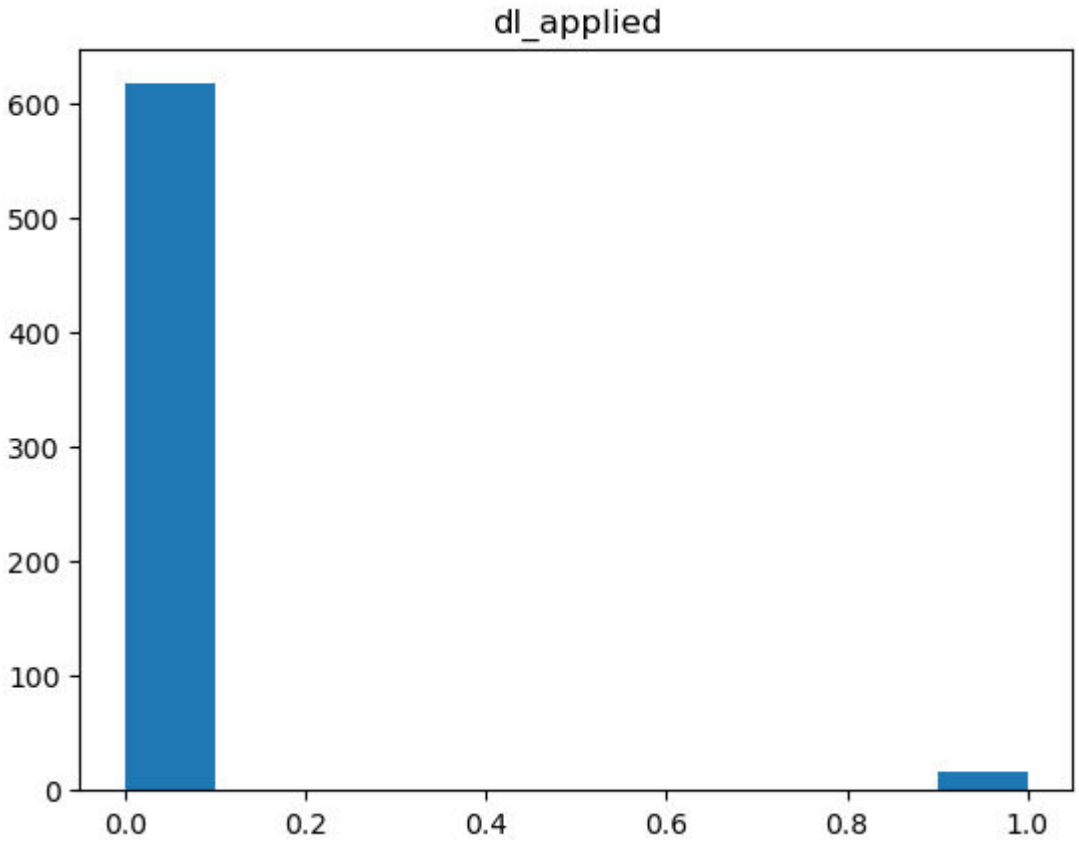
| | id | season | city | date | team1 | team2 | toss_winner | toss_decision | result | dl_applied | winner | win_by_runs | win_by_wicke |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **461** | 462 | 2014 | NaN | 2014-04-19 | Mumbai Indians | Royal Challengers Bangalore | Royal Challengers Bangalore | field | normal | 0 | Royal Challengers Bangalore | 0 | |
| **462** | 463 | 2014 | NaN | 2014-04-19 | Kolkata Knight Riders | Delhi Daredevils | Kolkata Knight Riders | bat | normal | 0 | Delhi Daredevils | 0 | |
| **466** | 467 | 2014 | NaN | 2014-04-23 | Chennai Super Kings | Rajasthan Royals | Rajasthan Royals | field | normal | 0 | Chennai Super Kings | 7 | |
| **468** | 469 | 2014 | NaN | 2014-04-25 | Sunrisers Hyderabad | Delhi Daredevils | Sunrisers Hyderabad | bat | normal | 0 | Sunrisers Hyderabad | 4 | |
| **469** | 470 | 2014 | NaN | 2014-04-25 | Mumbai Indians | Chennai Super Kings | Mumbai Indians | bat | normal | 0 | Chennai Super Kings | 0 | |
| **474** | 475 | 2014 | NaN | 2014-04-28 | Royal Challengers Bangalore | Kings XI Punjab | Kings XI Punjab | field | normal | 0 | Kings XI Punjab | 0 | |
| **476** | 477 | 2014 | NaN | 2014-04-30 | Sunrisers Hyderabad | Mumbai Indians | Mumbai Indians | field | normal | 0 | Sunrisers Hyderabad | 15 | |

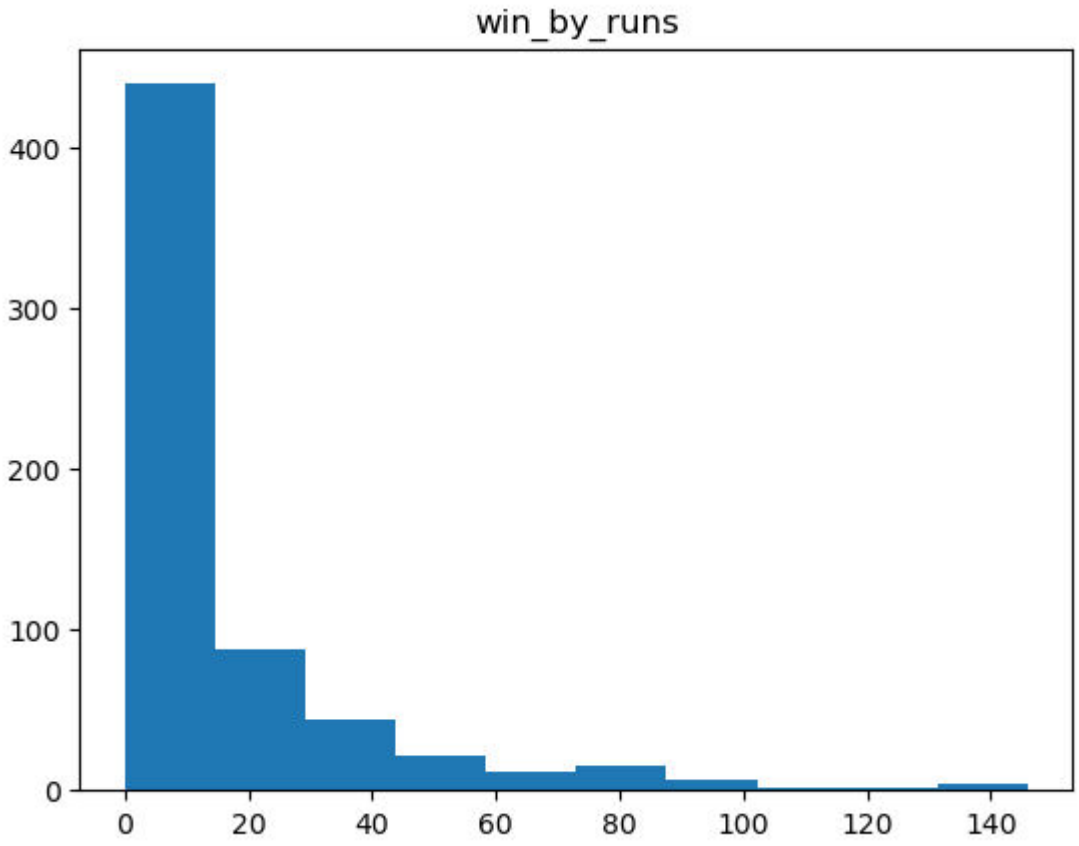
In [ ]:
```
matches_df = matches_df

matches_df = matches_df.dropna(subset=['player_of_match'])
```

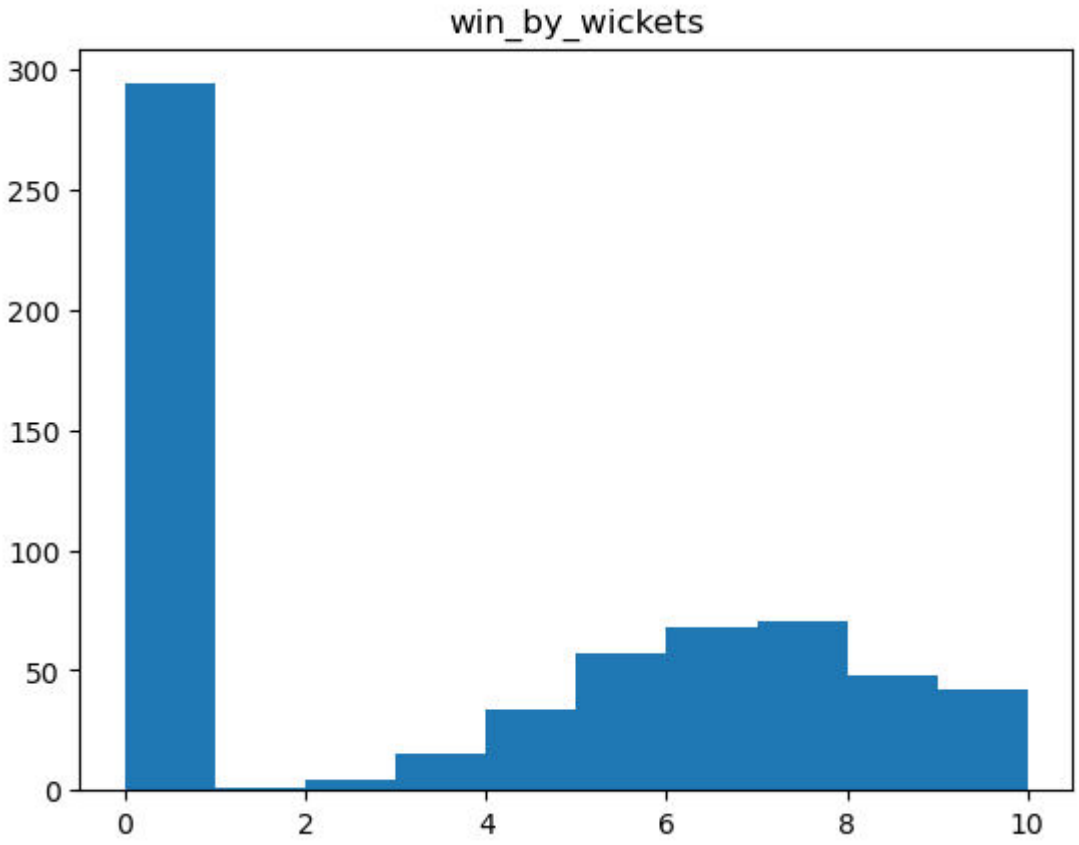
## Create a histogram for each numerical column

In [ ]:
```
# Create a histogram for each numerical column
# Create a histogram for each numerical column


matches_df_Mod_df = matches_df.drop('umpire3', axis=1)
matches_df_Mod_df = matches_df_Mod_df.drop('id', axis=1)

for col in matches_df_Mod_df.select_dtypes('number'):
    plt.hist(matches_df_Mod_df[col])
    plt.title(col)
    plt.show()
```




dl_applied
600
500
400
300
200
100
0
0.0
0.2
0.4
0.6
0.8
1.0


win_by_runs
400
300
200
100
0
0
20
40
60
80
100
120
140




## Create a correlation matrix to see the relationships between the variables

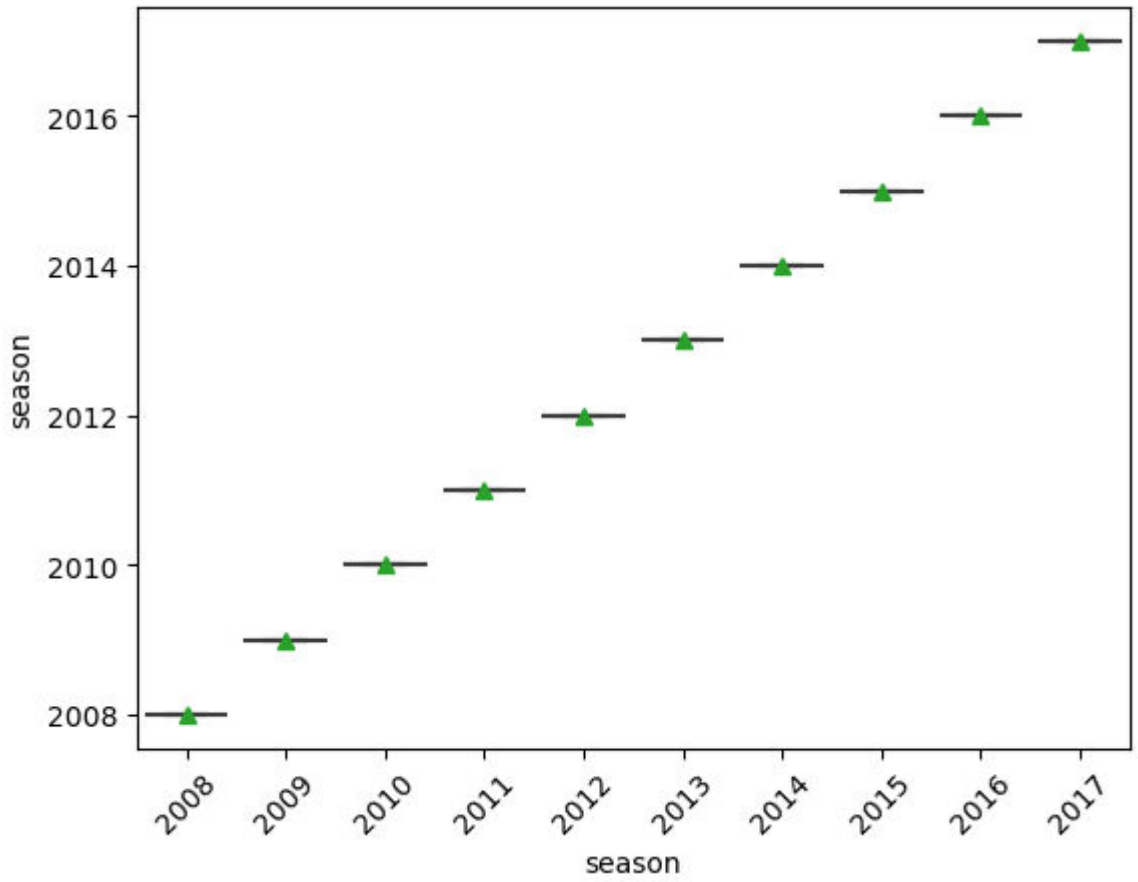
In [ ]:
```
# Create a correlation matrix to see the relationships between the variables
corr_matrix = matches_df.corr()
sns.heatmap(corr_matrix, annot=True)
plt.show()
```

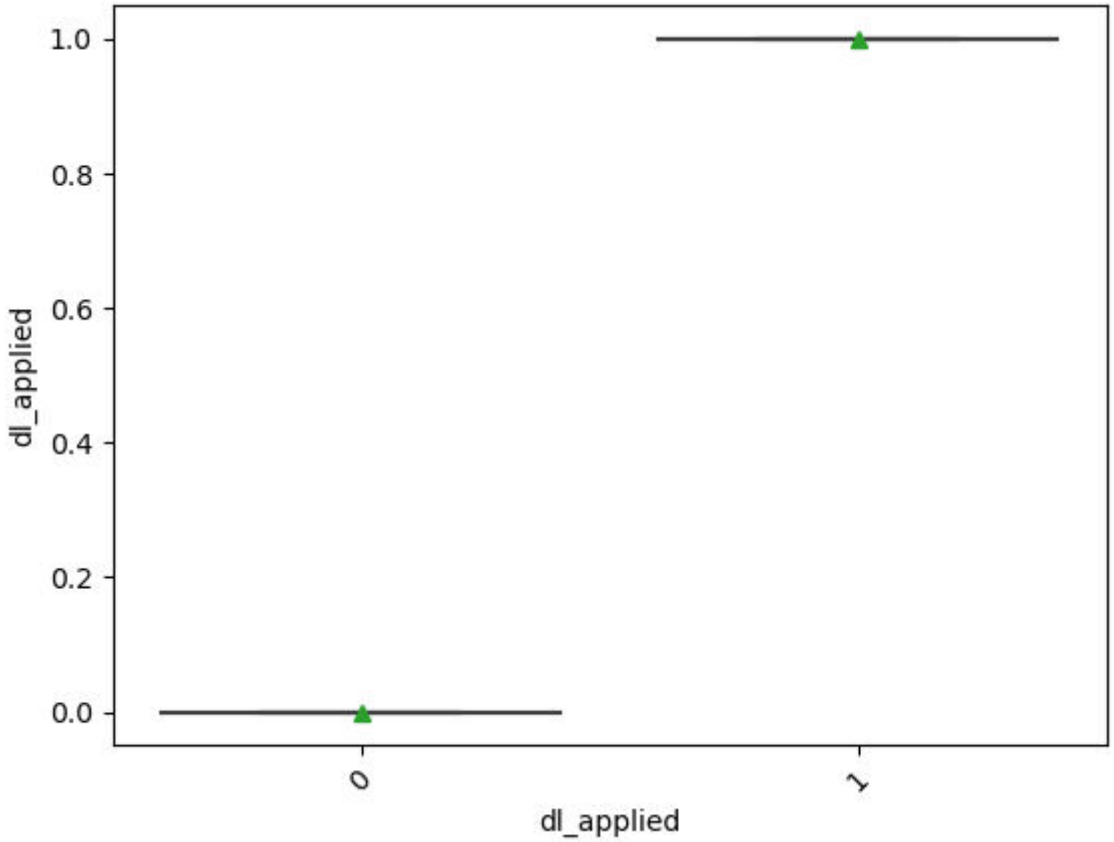
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
c:\Users\joshi\OneDrive\MS Data Sceinece\DCS 540\Project\JoshiAniruddhaDCS540ProjectMilestone5.ipynb Cell 26 line 2
      <a href='vscode-notebook-cell:/c%3A/Users/joshi/OneDrive/MS%20Data%20Sceinece/DCS%20540/Project/JoshiAniruddhaDCS540ProjectMilestone5.ipynb#X34sZmlsZQ%3D%3D?line=0'>1</a> # Create a correlation matrix to see the relationships between the variables
----> <a href='vscode-notebook-cell:/c%3A/Users/joshi/OneDrive/MS%20Data%20Sceinece/DCS%20540/Project/JoshiAniruddhaDCS540ProjectMilestone5.ipynb#X34sZmlsZQ%3D%3D?line=1'>2</a> corr_matrix = matches_df.corr()
      <a href='vscode-notebook-cell:/c%3A/Users/joshi/OneDrive/MS%20Data%20Sceinece/DCS%20540/Project/JoshiAniruddhaDCS540ProjectMilestone5.ipynb#X34sZmlsZQ%3D%3D?line=2'>3</a> sns.heatmap(corr_matrix, annot=True)
      <a href='vscode-notebook-cell:/c%3A/Users/joshi/OneDrive/MS%20Data%20Sceinece/DCS%20540/Project/JoshiAniruddhaDCS540ProjectMilestone5.ipynb#X34sZmlsZQ%3D%3D?line=3'>4</a> plt.show()

File c:\ProgramData\anaconda3\Lib\site-packages\pandas\core\frame.py:10054, in DataFrame.corr(self, method, min_periods, numeric_only)
  10052 cols = data.columns
  10053 idx = cols.copy()
> 10054 mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  10056 if method == "pearson":
  10057     correl = libalgos.nancorr(mat, minp=min_periods)

File c:\ProgramData\anaconda3\Lib\site-packages\pandas\core\frame.py:1838, in DataFrame.to_numpy(self, dtype, copy, na_value)
   1836 if dtype is not None:
   1837     dtype = np.dtype(dtype)
-> 1838 result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
   1839 if result.dtype is not dtype:
   1840     result = np.array(result, dtype=dtype, copy=False)

File c:\ProgramData\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:1732, in BlockManager.as_array(self, dtype, copy, na_value)
   1730         arr.flags.writeable = False
   1731 else:
-> 1732     arr = self._interleave(dtype=dtype, na_value=na_value)
   1733     # The underlying data was copied within _interleave, so no need
   1734     # to further copy if copy=True or setting na_value
   1736 if na_value is not lib.no_default:

File c:\ProgramData\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:1794, in BlockManager._interleave(self, dtype, na_value)
   1792     else:
   1793         arr = blk.get_values(dtype)
-> 1794     result[rl.indexer] = arr
   1795     itemmask[rl.indexer] = 1
   1797 if not itemmask.all():
```

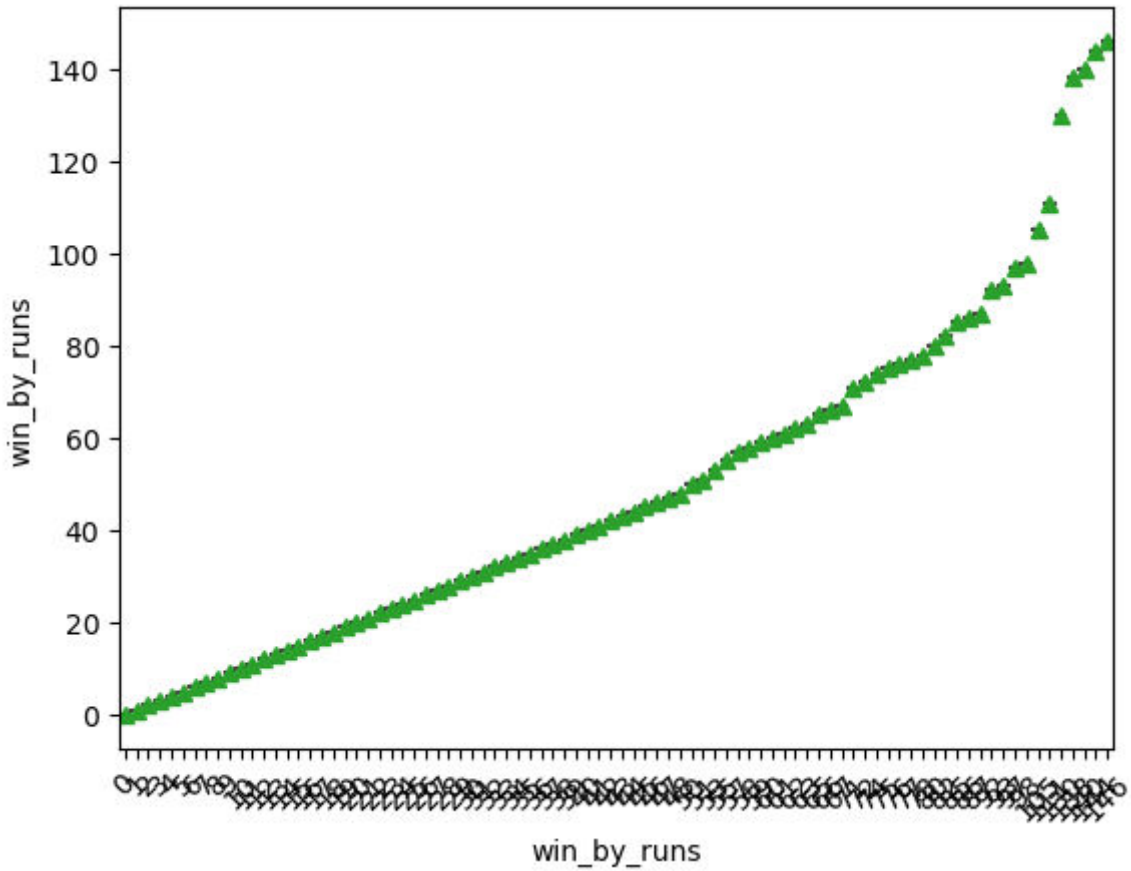
```
ValueError: could not convert string to float: 'Hyderabad'
```

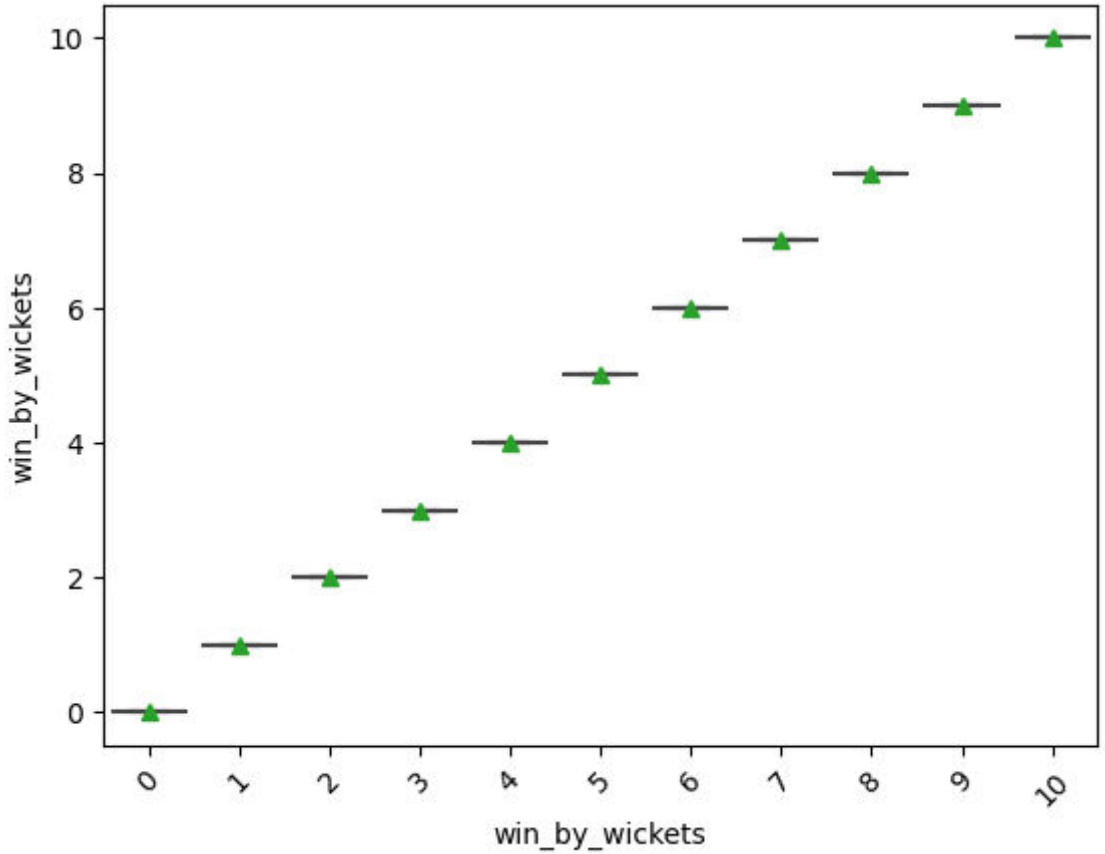
## Create a boxplot for each numerical column to see the distribution of the data

In [ ]:
```
# Create a boxplot for each numerical column to see the distribution of the data
for col in matches_df_Mod_df.select_dtypes('number'):
    sns.boxplot(x=col, y=col, showmeans=True, data=matches_df_Mod_df)
    # Rotate x labels to prevent overlapping
    plt.xticks(rotation=45)
    plt.show()
```




1.0
0.8
0.6
0.4
0.2
0.0
dl_applied
0
1
dl_applied


140
120
100
80
60
40
20
0
win_by_runs
win_by_runs




## Find duplicates

In [ ]:
```
# Find duplicates in the DataFrame
duplicates = deliveries_df[deliveries_df.duplicated()]

# Print the duplicate rows
(duplicates)
```

Out[ ]:

| | match_id | inning | batting_team | bowling_team | over | ball | batsman | non_striker | bowler | is_super_over | ... | bye_runs | legbye_runs |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **52178** | 221 | 1 | Mumbai Indians | Delhi Daredevils | 4 | 1 | SR Tendulkar | C Madan | PJ Sangwan | 0 | ... | 0 | 0 |

1 rows × 21 columns

In [ ]:
```
# refer to the row number 52178 is duplicated in data set deliveries_df
# Hence dropping the duplicate row
# Remove duplicates from the DataFrame
before_size = deliveries_df.size
deliveries_df = deliveries_df.drop_duplicates()
after_size = deliveries_df.size
print(f"The beore size is {before_size} and the after size is {after_size}")
```

```
The beore size is 3159660 and the after size is 3159639
```

In [ ]:
```
# Find duplicates in the DataFrame
duplicates = matches_df[matches_df.duplicated()]

# Print the duplicate rows
(duplicates)
```

Out[ ]:

| | id | season | city | date | team1 | team2 | toss_winner | toss_decision | result | dl_applied | winner | win_by_runs | win_by_wickets | player_of_match |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

In [ ]:
```
print("There are no duplicates rows in matches_df.")
```

```
There are no duplicates rows in matches_df.
```

## Fuzzy Matching of Strings

In [ ]:
```
#Finding any mismatches
from Levenshtein import distance
Player_Names=matches_df.player_of_match

name_of_player = "DA Warner"
for index, name in enumerate(Player_Names):
    # Check if the object has a length
    #if hasattr(name_of_player, 'len'):
```

```
        # The object has a length
        #Length = len(obj)
        #print(f"{name}")
        if distance(name_of_player, name) < 3:
            print(f"{name} {name_of_player} {distance(name_of_player, name)}")
```

```
DA Warner DA Warner 0
DA Warner DA Warner 0
DA Warner DA Warner 0
DA Warner DA Warner 0
DA Warner DA Warner 0
DA Warner DA Warner 0
DA Warner DA Warner 0
DA Warner DA Warner 0
DA Warner DA Warner 0
DA Warner DA Warner 0
DA Warner DA Warner 0
DA Warner DA Warner 0
DA Warner DA Warner 0
DA Warner DA Warner 0
DA Warner DA Warner 0
DA Warner DA Warner 0
```

In [ ]:
```
#Finding any mismatches
from Levenshtein import distance
Player_Names=matches_df.player_of_match
master_List_Player_name= deliveries_df.batsman
list_of_player=[]
name_of_player = "DA Warner"
for index, name in enumerate(Player_Names):
    name_of_player = name
    for index, name1 in enumerate(master_List_Player_name):
        if distance(name_of_player, name1) < 3 & distance(name_of_player, name1) > 0:
            print(f"{name1} {name_of_player} {distance(name_of_player, name1)}")
            list_of_player.append(distance(name_of_player, name1) )
```

# Milestone 3

## Funtion to read the website page

In [ ]:
```
def ReadWebsite(url ):
    # URL of the website with the HTML table
    #url = "https://www.espncricinfo.com/records/team/team-highest-match-aggregates/australia-2/twenty20-internationals
```

```
    # Send an HTTP GET request to the URL
    response = requests.get(url)
    return response
```

## Function to create a dataframe

In [ ]:
```
def CreateDataframe(response, tableIndex=0):
    # Check if the request was successful (status code 200)
    if response.status_code == 200:
        # Parse the HTML content of the webpage
        soup = BeautifulSoup(response.text, "html.parser")

        # Find the HTML table element you want to convert to a DataFrame
        table = soup.find_all("table")

        df = pd.read_html(str(table))[tableIndex]  # Assumes the first table on the page
        return df
    else:
        return None
```

In [ ]:
```
#Read website page result summary
response = ReadWebsite("https://www.espncricinfo.com/records/results-summary-283307")
results_sum_df = CreateDataframe(response)
results_sum_df.head()
```

Out[ ]:

| | Team | Span | Mat | Won | Lost | Draw | Tied | Tie+W | Tie+L | NR | W/L | %W | %L | %D | % |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Afghanistan | 2010-2023 | 118 | 74 | 42 | 0 | 0 | 0 | 1 | 1 | 1.761 | 62.71 | 35.59 | 0.0 | 63.67 |
| **1** | Argentina | 2019-2023 | 19 | 10 | 9 | 0 | 0 | 0 | 0 | 0 | 1.111 | 52.63 | 47.36 | 0.0 | 52.63 |
| **2** | Australia | 2005-2023 | 177 | 94 | 76 | 0 | 0 | 1 | 2 | 4 | 1.236 | 53.10 | 42.93 | 0.0 | 55.2 |
| **3** | Austria | 2019-2023 | 39 | 20 | 17 | 0 | 0 | 0 | 0 | 2 | 1.176 | 51.28 | 43.58 | 0.0 | 54.05 |
| **4** | Bahamas | 2021-2023 | 15 | 3 | 12 | 0 | 0 | 0 | 0 | 0 | 0.250 | 20.00 | 80.00 | 0.0 | 20 |

In [ ]:
```
#Read website page for match results
response = ReadWebsite("https://www.espncricinfo.com/records/year/team-match-results/2023-2023/twenty20-internationals-1
Match_results_df = CreateDataframe(response)
Match_results_df.head()
```

Out[ ]:

| | Team 1 | Team 2 | Winner | Margin | Ground | Match Date | Scorecard |
|---|---|---|---|---|---|---|---|
| **0** | India | Sri Lanka | India | 2 runs | Wankhede | Jan 3, 2023 | T20I # 1984 |
| **1** | India | Sri Lanka | Sri Lanka | 16 runs | Pune | Jan 5, 2023 | T20I # 1985 |
| **2** | India | Sri Lanka | India | 91 runs | Rajkot | Jan 7, 2023 | T20I # 1986 |
| **3** | Zimbabwe | Ireland | Zimbabwe | 5 wickets | Harare | Jan 12, 2023 | T20I # 1987 |
| **4** | Zimbabwe | Ireland | Ireland | 6 wickets | Harare | Jan 14, 2023 | T20I # 1988 |

In [ ]:
```
#Read website page for match results
response = ReadWebsite("https://www.espncricinfo.com/records/most-matches-as-captain-283747")
Captian_df = CreateDataframe(response)
Captian_df.head()
```

Out[ ]:

| | Player | Span | Mat | Won | Lost | Tied | Draw | NR | W/L | %W | %L | % Tied | % Draw | %NR | % |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | RT Ponting (AUS/ICC) | 2002-2012 | 230 | 165 | 51 | 2 | 0 | 12 | 3.23 | 71.73 | 22.17 | 0.86 | 0.0 | 5.21 | 76.14 |
| **1** | SP Fleming (NZ) | 1997-2007 | 218 | 98 | 106 | 1 | 0 | 13 | 0.92 | 44.95 | 48.62 | 0.45 | 0.0 | 5.96 | 48.04 |
| **2** | MS Dhoni (IND) | 2007-2018 | 200 | 110 | 74 | 5 | 0 | 11 | 1.48 | 55.00 | 37.00 | 2.50 | 0.0 | 5.50 | 59.52 |
| **3** | A Ranatunga (SL) | 1988-1999 | 193 | 89 | 95 | 1 | 0 | 8 | 0.93 | 46.11 | 49.22 | 0.51 | 0.0 | 4.14 | 48.37 |
| **4** | AR Border (AUS) | 1985-1994 | 178 | 107 | 67 | 1 | 0 | 3 | 1.59 | 60.11 | 37.64 | 0.56 | 0.0 | 1.68 | 61.42 |

## Replace Headers

In [ ]:
```
#Replace headers of results_sum_d
test_df= results_sum_df
print(f"Before modifying the columns names: {test_df.columns}")

test_df.columns ={'Team', 'Span', 'Mat', 'Won', 'Lost', 'Draw', 'Tied', 'Tie and Win', 'Tie and Lost',
       'NR', 'W/L', 'PCT W', 'PCT L', '%D', '%'}

print(f"After modifying the columns names: {test_df.columns}")
```

```
Before modifying the columns names: Index(['Team', 'Span', 'Mat', 'Won', 'Lost', 'Draw', 'Tied', 'Tie+W', 'Tie+L',
       'NR', 'W/L', '%W', '%L', '%D', '%'],
      dtype='object')
After modifying the columns names: Index(['Lost', 'Won', 'Mat', 'Team', '%', '%D', 'PCT L', 'Span', 'Tied',
       'Draw', 'NR', 'Tie and Win', 'Tie and Lost', 'PCT W', 'W/L'],
      dtype='object')
```

## Splitting the one Column data into multiple columns + Formating data into a more readable format

In [ ]:
```
# Check head
Match_results_df.head()
```

Out[ ]:

| | Team 1 | Team 2 | Winner | Margin | Ground | Match Date | Scorecard |
|---|---|---|---|---|---|---|---|
| **0** | India | Sri Lanka | India | 2 runs | Wankhede | Jan 3, 2023 | T20I # 1984 |
| **1** | India | Sri Lanka | Sri Lanka | 16 runs | Pune | Jan 5, 2023 | T20I # 1985 |
| **2** | India | Sri Lanka | India | 91 runs | Rajkot | Jan 7, 2023 | T20I # 1986 |
| **3** | Zimbabwe | Ireland | Zimbabwe | 5 wickets | Harare | Jan 12, 2023 | T20I # 1987 |
| **4** | Zimbabwe | Ireland | Ireland | 6 wickets | Harare | Jan 14, 2023 | T20I # 1988 |

In [ ]:
```
#Split the Margin data in to runs

Match_results_df['Win-Runs'] = Match_results_df['Margin'].apply(lambda x: int(x.split(' ')[0]) if x.find('runs') > -1 e
```

In [ ]:
```
#Split the Margin data in to wickets

Match_results_df['Win-Wickets'] = Match_results_df['Margin'].apply(lambda x: int(x.split(' ')[0]) if x.find('wickets')
```

In [ ]:
```
#Check the modified database - refer the two additonal columns based on Margin column
Match_results_df.head()
```

Out[ ]:

| | Team 1 | Team 2 | Winner | Margin | Ground | Match Date | Scorecard | Win-Runs | Win-Wickets |
|---|---|---|---|---|---|---|---|---|---|
| **0** | India | Sri Lanka | India | 2 runs | Wankhede | Jan 3, 2023 | T20I # 1984 | 2.0 | NaN |
| **1** | India | Sri Lanka | Sri Lanka | 16 runs | Pune | Jan 5, 2023 | T20I # 1985 | 16.0 | NaN |
| **2** | India | Sri Lanka | India | 91 runs | Rajkot | Jan 7, 2023 | T20I # 1986 | 91.0 | NaN |
| **3** | Zimbabwe | Ireland | Zimbabwe | 5 wickets | Harare | Jan 12, 2023 | T20I # 1987 | NaN | 5.0 |
| **4** | Zimbabwe | Ireland | Ireland | 6 wickets | Harare | Jan 14, 2023 | T20I # 1988 | NaN | 6.0 |

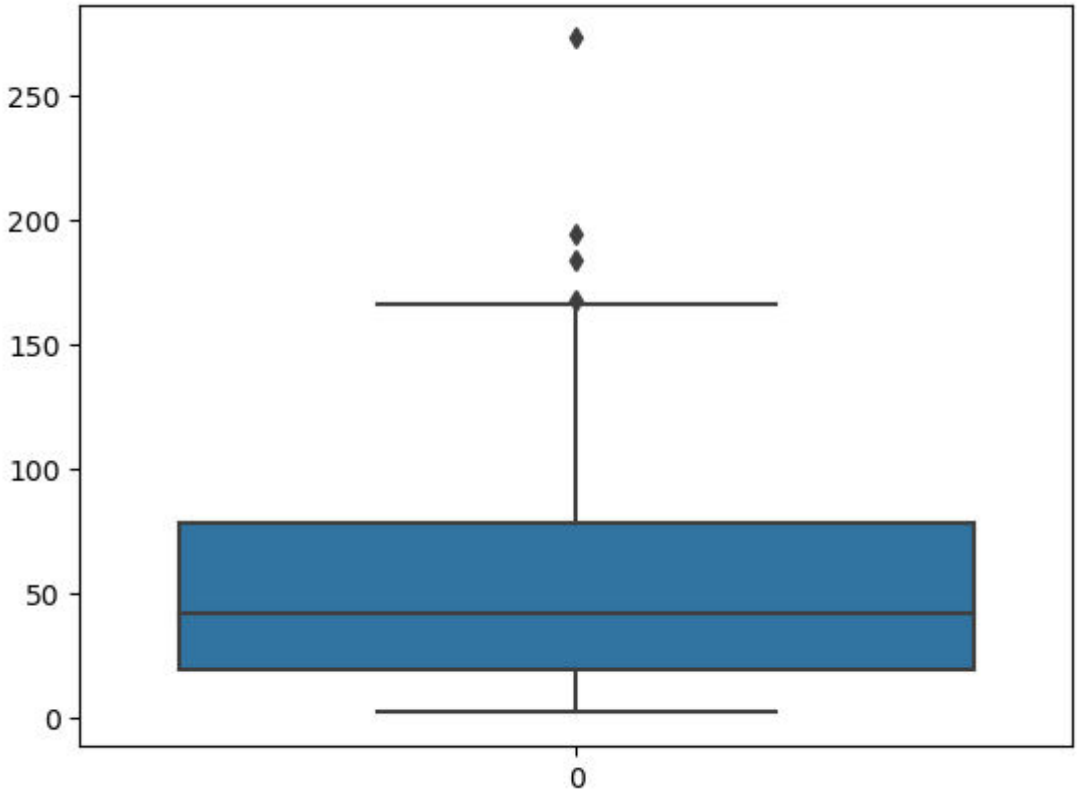
## Identify outliers and bad data

In [ ]:
```
#identifying the outliers on Match_results_df dataset
Match_results_df.head()
```

Out[ ]:

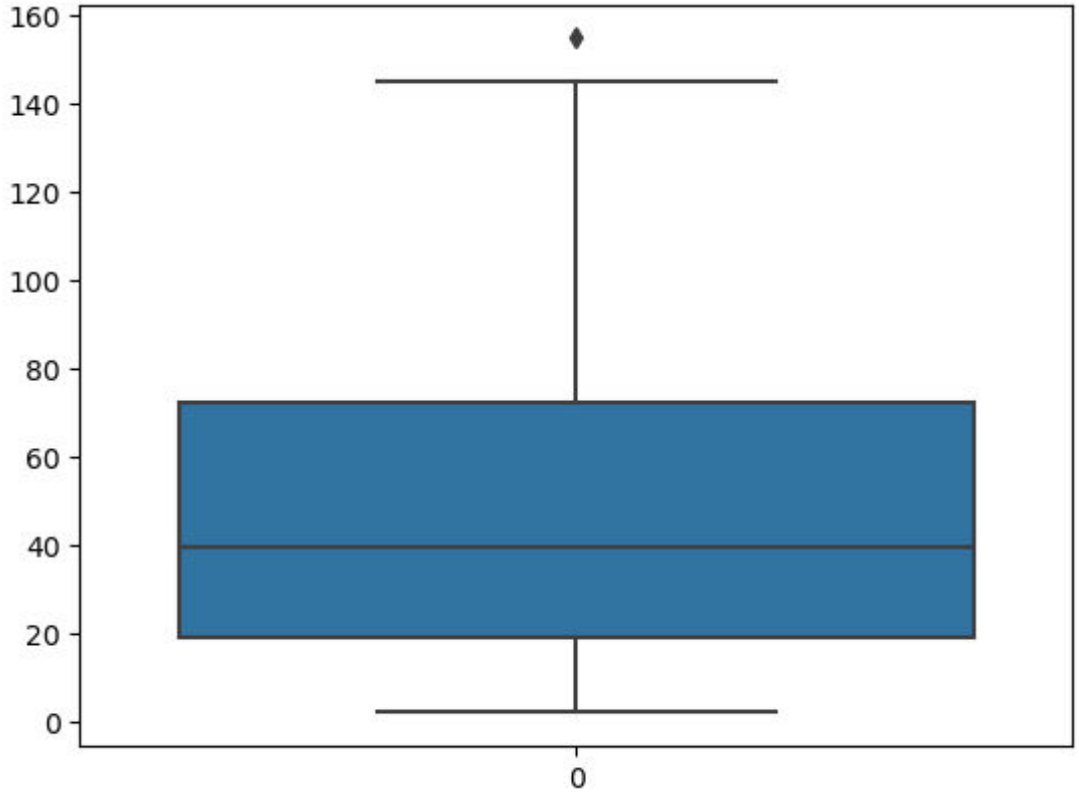
| | Team 1 | Team 2 | Winner | Margin | Ground | Match Date | Scorecard | Win-Runs | Win-Wickets |
|---|---|---|---|---|---|---|---|---|---|
| **0** | India | Sri Lanka | India | 2 runs | Wankhede | Jan 3, 2023 | T20I # 1984 | 2.0 | NaN |
| **1** | India | Sri Lanka | Sri Lanka | 16 runs | Pune | Jan 5, 2023 | T20I # 1985 | 16.0 | NaN |
| **2** | India | Sri Lanka | India | 91 runs | Rajkot | Jan 7, 2023 | T20I # 1986 | 91.0 | NaN |
| **3** | Zimbabwe | Ireland | Zimbabwe | 5 wickets | Harare | Jan 12, 2023 | T20I # 1987 | NaN | 5.0 |
| **4** | Zimbabwe | Ireland | Ireland | 6 wickets | Harare | Jan 14, 2023 | T20I # 1988 | NaN | 6.0 |

In [ ]:
```
sns.boxplot(data=Match_results_df['Win-Runs'])
 # Show the plot.
plt.show()
```




In [ ]:
```
#We can clearly see the outliers are above 160 runs
#Removing outliers above 160 runs
Match_results_df['Win-Runs']=Match_results_df['Win-Runs'].apply(lambda x: x if x < 160 else np.nan)
```

In [ ]:
```
#Data After correction by removing outliers
sns.boxplot(data=Match_results_df['Win-Runs'])
 # Show the plot.
plt.show()
```




## Find duplicates

In [ ]:
```
#In general find if there are any duplicate records

print(f"There are {Match_results_df.duplicated().sum()} duplicate records in the Match_results_df.")
print(f"There are {results_sum_df.duplicated().sum()} duplicate records in the results_sum_df.")
print(f"There are {Captian_df.duplicated().sum()} duplicate records in the Captian_df.")
```

```
There are 0 duplicate records in the Match_results_df.
There are 0 duplicate records in the results_sum_df.
There are 0 duplicate records in the Captian_df.
```

In [ ]:
```
#Finding duplicates in the data sets

print(f"There are {Match_results_df['Winner'].duplicated().sum()} duplicate values in Winner columns of Match_results_d
print(f"The unique values are :\n {Match_results_df['Winner'].unique()}")
```

```
There are 288 duplicate values in Winner columns of Match_results_df.


The unique values are :
 ['India' 'Sri Lanka' 'Zimbabwe' 'Ireland' 'New Zealand' 'Afghanistan'
 'U.A.E.' 'Bermuda' 'Spain' 'no result' 'Panama' 'Bahamas' 'Cayman Is'
 'Argentina' 'Malaysia' 'Hong Kong' 'Bangladesh' 'Bahrain' 'Kuwait'
 'Vanuatu' 'Fiji' 'West Indies' 'South Africa' 'Pakistan' 'tied'
 'Portugal' 'Indonesia' 'Singapore' 'Cambodia' 'Gibraltar' 'Malta'
 'Denmark' 'Norway' 'Sweden' 'Finland' 'Malawi' 'Botswana' 'Mozambique'
 'Kenya' 'Uganda' 'Germany' 'Czech Rep.' 'Rwanda' 'Serbia' 'Bulgaria'
 'Turkey' 'Luxembourg' 'Switzerland' 'Jersey' 'Isle of Man' 'France'
 'Romania' 'Scotland' 'Japan' 'P.N.G.' 'Italy' 'Philippines' 'Bhutan'
 'Thailand' 'China' 'Hungary' 'Guernsey' 'Tanzania' 'Australia' 'England'
 'Qatar' 'Oman' 'Nepal' 'Maldives' 'Saudi Arabia' 'Estonia' 'Canada'
 'Nigeria' 'Ghana' 'Sierra Leone' 'Mexico' 'Namibia']
```

## Combining two dataframes

In [ ]:
```
# Combining results_sum_df and Match_results_df
results_sum_df.head()
```

Out[ ]:

| | Lost | Won | Mat | Team | % | %D | PCT L | Span | Tied | Draw | NR | Tie and Win | Tie and Lost | PCT W | W/L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Afghanistan | 2010-2023 | 118 | 74 | 42 | 0 | 0 | 0 | 1 | 1 | 1.761 | 62.71 | 35.59 | 0.0 | 63.67 |
| **1** | Argentina | 2019-2023 | 19 | 10 | 9 | 0 | 0 | 0 | 0 | 0 | 1.111 | 52.63 | 47.36 | 0.0 | 52.63 |
| **2** | Australia | 2005-2023 | 177 | 94 | 76 | 0 | 0 | 1 | 2 | 4 | 1.236 | 53.10 | 42.93 | 0.0 | 55.2 |
| **3** | Austria | 2019-2023 | 39 | 20 | 17 | 0 | 0 | 0 | 0 | 2 | 1.176 | 51.28 | 43.58 | 0.0 | 54.05 |
| **4** | Bahamas | 2021-2023 | 15 | 3 | 12 | 0 | 0 | 0 | 0 | 0 | 0.250 | 20.00 | 80.00 | 0.0 | 20 |

In [ ]:
```
Match_results_df['Mat']=Match_results_df['Winner']
Match_results_df.head()
```

Out[ ]:

| | Team 1 | Team 2 | Winner | Margin | Ground | Match Date | Scorecard | Win-Runs | Win-Wickets | Mat |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | India | Sri Lanka | India | 2 runs | Wankhede | Jan 3, 2023 | T20I # 1984 | 2.0 | NaN | India |
| **1** | India | Sri Lanka | Sri Lanka | 16 runs | Pune | Jan 5, 2023 | T20I # 1985 | 16.0 | NaN | Sri Lanka |
| **2** | India | Sri Lanka | India | 91 runs | Rajkot | Jan 7, 2023 | T20I # 1986 | 91.0 | NaN | India |
| **3** | Zimbabwe | Ireland | Zimbabwe | 5 wickets | Harare | Jan 12, 2023 | T20I # 1987 | NaN | 5.0 | Zimbabwe |
| **4** | Zimbabwe | Ireland | Ireland | 6 wickets | Harare | Jan 14, 2023 | T20I # 1988 | NaN | 6.0 | Ireland |

In [ ]:
```
# combining two data sets based on Winner and Mat
#Combined_df=pd.merge(results_sum_df,Match_results_df,how='inner',on='Mat')
Combined_df = pd.concat([results_sum_df, Match_results_df], axis=0, ignore_index=True)
Combined_df.head()
```

Out[ ]:

| | Lost | Won | Mat | Team | % | %D | PCT L | Span | Tied | Draw | ... | W/L | Team 1 | Team 2 | Winner | Margin | Ground | Match Date | Scorecard |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Afghanistan | 2010-2023 | 118 | 74.0 | 42.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | ... | 63.67 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **1** | Argentina | 2019-2023 | 19 | 10.0 | 9.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 52.63 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **2** | Australia | 2005-2023 | 177 | 94.0 | 76.0 | 0.0 | 0.0 | 1.0 | 2.0 | 4.0 | ... | 55.2 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **3** | Austria | 2019-2023 | 39 | 20.0 | 17.0 | 0.0 | 0.0 | 0.0 | 0.0 | 2.0 | ... | 54.05 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **4** | Bahamas | 2021-2023 | 15 | 3.0 | 12.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 20 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

5 rows × 24 columns

In [ ]:
```
print(f"The size of results_sum_df is {results_sum_df.shape}.")
print(f"The size of Match_results_df is {Match_results_df.shape}.")
print(f"The size of Combined_df is {Combined_df.shape}.")
```

```
The size of results_sum_df is (103, 15).
The size of Match_results_df is (365, 10).
The size of Combined_df is (468, 24).
```

## Ethical Implementation

The data represents the number of matches won by the countries and the player performance. I see below ethical impications

1. The number of matches won by the countries may encourage the advertisers to concentrate on winning teams only.
2. The underground betting market may use this information to bet on the specific teams and/or players only.
3. The losing team may lose the ethusiasum of the game and resumed they are going to lose the match.
4. Only specfic players may get a chance to play multiple matches or the selection will be based on the playesr who did best againt specific teams

In [ ]:
```
##
```

../Week7-8/JoshiAniruddhaDCS540Week7-8-Assignment1.ipynb

# Milestone 4

## Reading the data from APIs

In [ ]:
```
## Define the APi URL

api_url = "https://cricket.sportmonks.com/api/v2.0/players?api_token=PClqGEdYVpY0Pc1fzjwxrFHRDzCeVS4AHbXk4Ef90D4roFStme
```

In [ ]:
```
# Send a request
response = requests.get(api_url)
```

In [ ]:
```
#If the response if good then print it
if response.status_code == 200:
    data = response.json()

    #use prety print function
    retty_json = json.dumps(data, indent=4)
    #print(retty_json)
else:
    print("Error:", response.status_code)
```

In [ ]:
```
players_df=pd.json_normalize(data, record_path=['data'])
players_df.head()
```

Out[ ]:

| | resource | id | country_id | firstname | lastname | fullname | image_path | dateofbirth | gender | battingstyle | b |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | players | 2 | 52126 | Ahmed | Shehzad | Ahmed Shehzad | https://cdn.sportmonks.com/images/cricket/play... | 1991-11-23 | m | right-hand-bat | |
| **1** | players | 3 | 52126 | Anwar | Ali | Anwar Ali | https://cdn.sportmonks.com/images/cricket/play... | 1987-11-25 | m | right-hand-bat | |
| **2** | players | 4 | 52126 | Sarfraz | Ahmed | Sarfraz Ahmed | https://cdn.sportmonks.com/images/cricket/play... | 1987-05-22 | m | right-hand-bat | |
| **3** | players | 5 | 52126 | Azhar | Ali | Azhar Ali | https://cdn.sportmonks.com/images/cricket/play... | 1985-02-19 | m | right-hand-bat | |
| **4** | players | 6 | 52126 | Fakhar | Zaman | Fakhar Zaman | https://cdn.sportmonks.com/images/cricket/play... | 1990-04-10 | m | left-hand-bat | |

In [ ]:
```
#Now get the league information
api_url = "https://cricket.sportmonks.com/api/v2.0/leagues?api_token=PClqGEdYVpY0Pc1fzjwxrFHRDzCeVS4AHbXk4Ef90D4roFStme1
response = requests.get(api_url)
#If the response if good then print it
if response.status_code == 200:
    data = response.json()

    #use prety print function
    retty_json = json.dumps(data, indent=4)
    #print(retty_json)
else:
    print("Error:", response.status_code)
```

In [ ]:
```
leagues_df=pd.json_normalize(data, record_path=['data'])
leagues_df.head()
```

Out[ ]:

| | resource | id | season_id | country_id | name | code | image_path | type | updated_at |
|---|---|---|---|---|---|---|---|---|---|
| **0** | leagues | 3 | 1427 | 99474 | Twenty20 International | T20I | https://cdn.sportmonks.com/images/cricket/leag... | phase | 2023-10-03T13:31:56.000000Z |
| **1** | leagues | 5 | 1349 | 98 | Big Bash League | BBL | https://cdn.sportmonks.com/images/cricket/leag... | league | 2023-07-07T21:33:29.000000Z |
| **2** | leagues | 10 | 1145 | 146 | CSA T20 Challenge | T20C | https://cdn.sportmonks.com/images/cricket/leag... | league | 2022-09-18T15:09:59.000000Z |

In [ ]:
```
#Now get the fixtures information
api_url = "https://cricket.sportmonks.com/api/v2.0/fixtures?includes=runs&api_token=PClqGEdYVpY0Pc1fzjwxrFHRDzCeVS4AHbXl
#headers ={'include':'runs'}
response = requests.get(api_url)#, headers=headers)
#If the response if good then print it
if response.status_code == 200:
    data = response.json()

    #use prety print function
    retty_json = json.dumps(data, indent=4)
    #print(retty_json)
else:
    print("Error:", response.status_code)
```

In [ ]:
```
fixtures_df=pd.json_normalize(data, record_path=['data'])
fixtures_df.head()
```

Out[ ]:

| | resource | id | league_id | season_id | stage_id | round | localteam_id | visitorteam_id | starting_at | type | ... | weather_report | localteam |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | fixtures | 3 | 3 | 6 | 1755 | 2nd T20I | 40 | 41 | 2018-10-12T16:00:00.000000Z | T20I | ... | [] | |
| **1** | fixtures | 4 | 3 | 6 | 1755 | 3rd T20I | 40 | 41 | 2018-10-14T12:30:00.000000Z | T20I | ... | [] | |
| **2** | fixtures | 216 | 3 | 6 | 24 | 1st T20I | 10 | 43 | 2018-11-04T13:30:00.000000Z | T20I | ... | [] | |
| **3** | fixtures | 217 | 3 | 6 | 24 | 2nd T20I | 10 | 43 | 2018-11-06T13:30:00.000000Z | T20I | ... | [] | |
| **4** | fixtures | 218 | 3 | 6 | 24 | 3rd T20I | 10 | 43 | 2018-11-11T13:30:00.000000Z | T20I | ... | [] | |

5 rows × 40 columns

In [ ]:
```
#Now get the teams information
api_url = "https://cricket.sportmonks.com/api/v2.0/teams?api_token=PClqGEdYVpY0Pc1fzjwxrFHRDzCeVS4AHbXk4Ef90D4roFStme1lI
#headers ={'include':'runs'}
response = requests.get(api_url)#, headers=headers)
#If the response if good then print it
if response.status_code == 200:
    data = response.json()

    #use prety print function
    retty_json = json.dumps(data, indent=4)
    #print(retty_json)
else:
    print("Error:", response.status_code)
```

In [ ]:
```
teams_df=pd.json_normalize(data, record_path=['data'])
#get the size of teams_df
teams_df.head()
```

Out[ ]:

| | resource | id | name | code | image_path | country_id | national_team | updated_at |
|---|---|---|---|---|---|---|---|---|
| **0** | teams | 1 | Pakistan | PAK | https://cdn.sportmonks.com/images/cricket/team... | 190324 | True | 2018-11-29T11:47:20.000000Z |
| **1** | teams | 10 | India | IND | https://cdn.sportmonks.com/images/cricket/team... | 190324 | True | 2018-11-29T11:47:20.000000Z |
| **2** | teams | 36 | Australia | AUS | https://cdn.sportmonks.com/images/cricket/team... | 190324 | True | 2018-11-29T11:47:20.000000Z |
| **3** | teams | 37 | Bangladesh | BGD | https://cdn.sportmonks.com/images/cricket/team... | 190324 | True | 2018-11-29T11:47:20.000000Z |
| **4** | teams | 38 | England | ENG | https://cdn.sportmonks.com/images/cricket/team... | 190324 | True | 2019-01-29T11:07:04.000000Z |

## Modifyig column names

In [ ]:
```
#Replace headers of results_sum_d
test_df= players_df.copy()
print(f"Before modifying the columns names: {test_df.columns}")

test_df.columns ={'ResoureName', 'id', 'country_id', 'firstname', 'lastname', 'fullname',
       'image_path', 'dateofbirth', 'gender', 'battingstyle', 'BlowingStyle',
       'updated_at', 'ResourcePosition', 'position.id', 'PositionName'}


print(f"After modifying the columns names: {test_df.columns}")
```

```
Before modifying the columns names: Index(['resource', 'id', 'country_id', 'firstname', 'lastname', 'fullname',
       'image_path', 'dateofbirth', 'gender', 'battingstyle', 'bowlingstyle',
       'updated_at', 'position.resource', 'position.id', 'position.name'],
      dtype='object')
After modifying the columns names: Index(['position.id', 'gender', 'ResourcePosition', 'PositionName',
       'dateofbirth', 'firstname', 'BlowingStyle', 'id', 'fullname',
       'ResoureName', 'image_path', 'lastname', 'battingstyle', 'updated_at',
       'country_id'],
      dtype='object')
```

## Splitting the rows/filtering data into multiple datasets

In [ ]:
```
# Check head
players_df.head()
```

Out[ ]:

| | resource | id | country_id | firstname | lastname | fullname | image_path | dateofbirth | gender | battingstyle | b |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | players | 2 | 52126 | Ahmed | Shehzad | Ahmed Shehzad | https://cdn.sportmonks.com/images/cricket/play... | 1991-11-23 | m | right-hand-bat | |
| **1** | players | 3 | 52126 | Anwar | Ali | Anwar Ali | https://cdn.sportmonks.com/images/cricket/play... | 1987-11-25 | m | right-hand-bat | |
| **2** | players | 4 | 52126 | Sarfraz | Ahmed | Sarfraz Ahmed | https://cdn.sportmonks.com/images/cricket/play... | 1987-05-22 | m | right-hand-bat | |
| **3** | players | 5 | 52126 | Azhar | Ali | Azhar Ali | https://cdn.sportmonks.com/images/cricket/play... | 1985-02-19 | m | right-hand-bat | |
| **4** | players | 6 | 52126 | Fakhar | Zaman | Fakhar Zaman | https://cdn.sportmonks.com/images/cricket/play... | 1990-04-10 | m | left-hand-bat | |

In [ ]:
```
#Splitting the data
players_batsman = players_df[players_df['position.name']=='Batsman'].copy()
players_batsman.head()
```

Out[ ]:

| | resource | id | country_id | firstname | lastname | fullname | image_path | dateofbirth | gender | battingstyle | b |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | players | 2 | 52126 | Ahmed | Shehzad | Ahmed Shehzad | https://cdn.sportmonks.com/images/cricket/play... | 1991-11-23 | m | right-hand-bat | |
| **3** | players | 5 | 52126 | Azhar | Ali | Azhar Ali | https://cdn.sportmonks.com/images/cricket/play... | 1985-02-19 | m | right-hand-bat | |
| **4** | players | 6 | 52126 | Fakhar | Zaman | Fakhar Zaman | https://cdn.sportmonks.com/images/cricket/play... | 1990-04-10 | m | left-hand-bat | |
| **5** | players | 7 | 52126 | Imam | ul Haq | Imam ul Haq | https://cdn.sportmonks.com/images/cricket/play... | 1995-12-12 | m | left-hand-bat | |
| **6** | players | 8 | 52126 | Babar | Azam | Babar Azam | https://cdn.sportmonks.com/images/cricket/play... | 1994-10-15 | m | right-hand-bat | |

In [ ]:
```
#Splitting the data
players_bolwers = players_df[players_df['position.name']=='Bowler']
players_bolwers.head()
```

Out[ ]:

| | resource | id | country_id | firstname | lastname | fullname | image_path | dateofbirth | gender | battingst |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | players | 3 | 52126 | Anwar | Ali | Anwar Ali | https://cdn.sportmonks.com/images/cricket/play... | 1987-11-25 | m | right-ha |
| **10** | players | 12 | 52126 | Yasir | Shah | Yasir Shah | https://cdn.sportmonks.com/images/cricket/play... | 1986-05-02 | m | right-ha |
| **13** | players | 15 | 52126 | Mohammad | Abbas | Mohammad Abbas | https://cdn.sportmonks.com/images/cricket/play... | 1990-03-10 | m | right-ha |
| **14** | players | 16 | 52126 | Hasan | Ali | Hasan Ali | https://cdn.sportmonks.com/images/cricket/play... | 1994-07-02 | m | right-ha |
| **15** | players | 17 | 52126 | Wahab | Riaz | Wahab Riaz | https://cdn.sportmonks.com/images/cricket/play... | 1985-06-28 | m | right-ha |

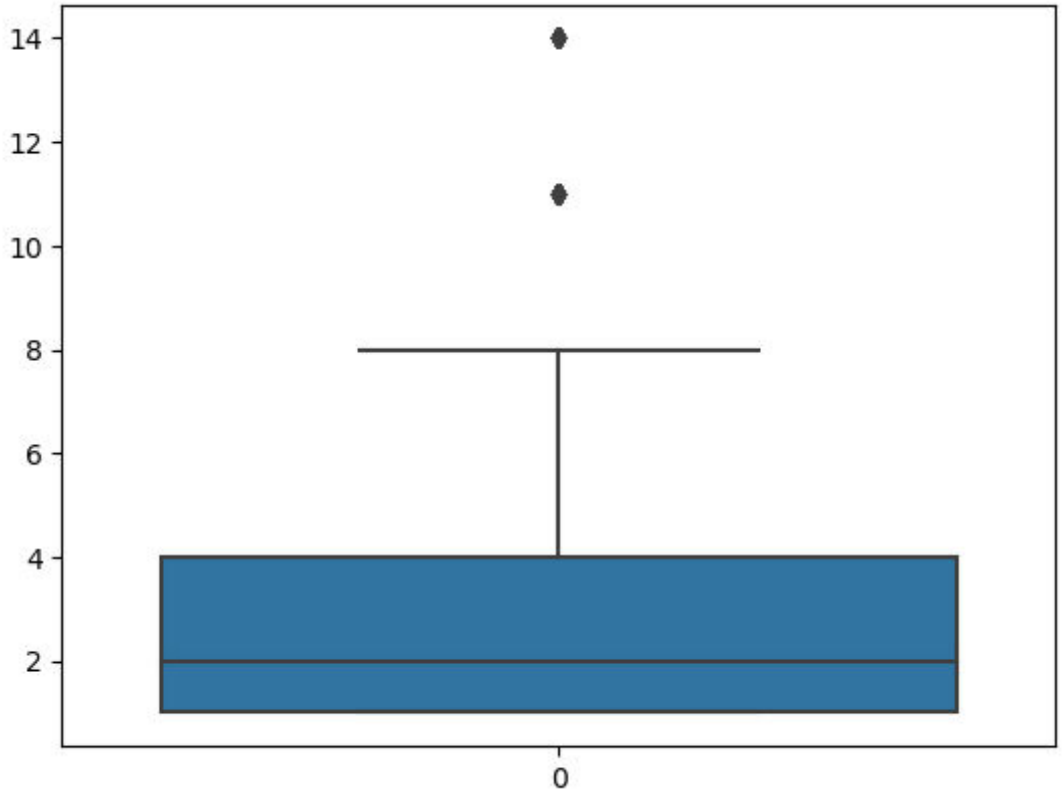
## Identify outliers and bad data

In [ ]:
```
#identifying the outliers on Match_results_df dataset
players_df.describe()
```

Out[ ]:

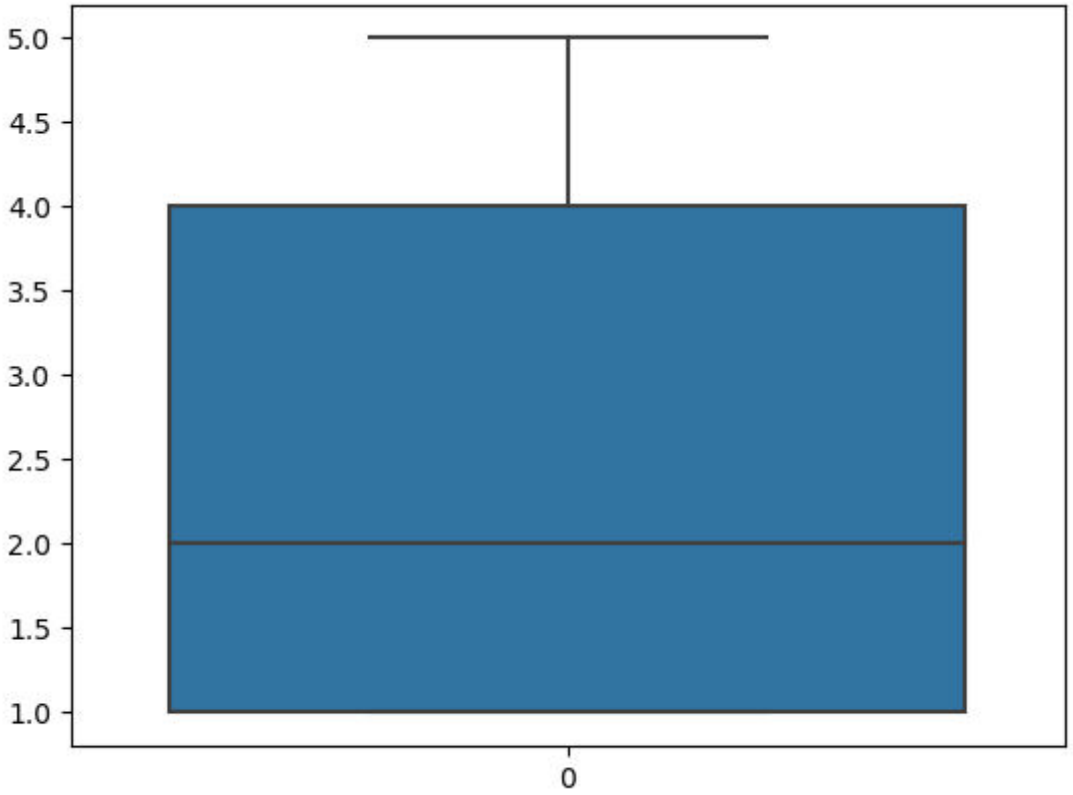
| | id | country_id | position.id |
|---|---|---|---|
| **count** | 17566.000000 | 1.756600e+04 | 17566.000000 |
| **mean** | 25489.075316 | 7.517936e+05 | 2.455084 |
| **std** | 15543.458965 | 3.876135e+06 | 1.404682 |
| **min** | 2.000000 | 2.000000e+00 | 1.000000 |
| **25%** | 11979.750000 | 1.430000e+02 | 1.000000 |
| **50%** | 25405.500000 | 1.233000e+03 | 2.000000 |
| **75%** | 38995.250000 | 1.537320e+05 | 4.000000 |
| **max** | 52433.000000 | 2.415087e+07 | 14.000000 |

In [ ]:
```
sns.boxplot(data=players_df['position.id'])
 # Show the plot.
plt.show()
```




In [ ]:
```
#We can clearly see the outliers are position 6
#Removing outliers above 160 runs
players_df['position.id']=players_df['position.id'].apply(lambda x: x if x < 6 else np.nan)
```

In [ ]:
```
sns.boxplot(data=players_df['position.id'])
 # Show the plot.
plt.show()
```




## Find duplicates

In [ ]:
```
#In general find if there are any duplicate records

print(f"There are {players_df['fullname'].duplicated().sum()} duplicate records in the Match_results_df.")
```

```
There are 737 duplicate records in the Match_results_df.
```

In [ ]:
```
#Removing the duplicates
test_df=players_df['fullname'].drop_duplicates()
test_df
# printing results afater removing duplicates
print(f"There are {test_df.duplicated().sum()} duplicate records in the Match_results_df.")
```

```
There are 0 duplicate records in the Match_results_df.
```

## Combining two dataframes

In [ ]:
```
#combining the dataframes based on player id
#players_df['id'] = fixtures_df['man_of_match_id']
#Create a new columns
players_df['man_of_match_id'] = players_df['id']
```

In [ ]:
```
# combining two data sets based on Winner and Mat
Combined_df=pd.merge(fixtures_df, players_df,how='left',on='man_of_match_id')
Combined_df.columns
#the Combined table now has name of the played based on player ID
```

Out[ ]:
```
Index(['resource_x', 'id_x', 'league_id', 'season_id', 'stage_id', 'round',
       'localteam_id', 'visitorteam_id', 'starting_at', 'type', 'live',
       'status', 'last_period', 'note', 'venue_id', 'toss_won_team_id',
       'winner_team_id', 'draw_noresult', 'first_umpire_id',
       'second_umpire_id', 'tv_umpire_id', 'referee_id', 'man_of_match_id',
       'man_of_series_id', 'total_overs_played', 'elected', 'super_over',
       'follow_on', 'rpc_overs', 'rpc_target', 'weather_report',
       'localteam_dl_data.score', 'localteam_dl_data.overs',
       'localteam_dl_data.wickets_out', 'visitorteam_dl_data.score',
       'visitorteam_dl_data.overs', 'visitorteam_dl_data.wickets_out',
       'visitorteam_dl_data.total_overs_played', 'localteam_dl_data.rpc_overs',
       'localteam_dl_data.rpc_targets', 'resource_y', 'id_y', 'country_id',
       'firstname', 'lastname', 'fullname', 'image_path', 'dateofbirth',
       'gender', 'battingstyle', 'bowlingstyle', 'updated_at',
       'position.resource', 'position.id', 'position.name'],
      dtype='object')
```

In [ ]:
```
print(f"The size of fixtures_df is {fixtures_df.shape}.")
print(f"The size of players_df is {players_df.shape}.")
print(f"The size of Combined_df is {Combined_df.shape}.")
```

```
The size of fixtures_df is (100, 40).
The size of players_df is (17566, 16).
The size of Combined_df is (100, 55).
```

## Ethical Implementation

The data represents the player details such as th ecountry associated with, matches played, lost the overall performance on each match etc. I see below ethical impications

1. This could impact the selection process of players as it will be purelye based on perfoamce. some played may do good with specific teams and may get chosen for the same opponent.

2. The underground betting market may use this information to bet on the specific teams and/or players only.
3. Players may develop skill sin specific areas only

# Milestone 5

## Listing three databases

In [ ]:
```
#The three data sets to be used are

#1. Excel reads
cricket_data_df.head()
```

Out[ ]:

| | Unnamed: 0 | ID | NAME | COUNTRY | Full name | Birthdate | Birthplace | Died | Date_of_death | Age | ... | BOWLING_T20s_Runs | BOWLI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 8772 | Henry Arkell | England | Henry John Denham Arkell | 1898-06-26 | Edmonton, Middlesex | Dead | 12/03/82 | 84.0 | ... | NaN | |
| **1** | 1 | 532565 | Richard Nyren | England | Richard Nyren | 1734-04-25 | Eartham, Sussex | Dead | 1797-04-25 | 63.0 | ... | NaN | |
| **2** | 2 | 16856 | Sydney Maartensz | England | Sydney Gratien Adair Maartensz | 1882-04-14 | Colombo, Ceylon | Dead | 10/09/67 | 85.0 | ... | NaN | |
| **3** | 3 | 16715 | Brian Lander | England | Brian Richard Lander | 09/01/42 | Bishop Auckland, Co Durham | Alive | NaN | 77.0 | ... | NaN | |
| **4** | 4 | 15989 | Derek Kenderdine | England | Derek Charles Kenderdine | 1897-10-28 | Chislehurst, Kent | Dead | 28/08/47 | 50.0 | ... | NaN | |

5 rows × 177 columns

In [ ]:
```
#2. data set from online website
Captian_df.head()
```

Out[ ]:

| | Player | Span | Mat | Won | Lost | Tied | Draw | NR | W/L | %W | %L | % Tied | % Draw | %NR | % |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | RT Ponting (AUS/ICC) | 2002-2012 | 230 | 165 | 51 | 2 | 0 | 12 | 3.23 | 71.73 | 22.17 | 0.86 | 0.0 | 5.21 | 76.14 |
| **1** | SP Fleming (NZ) | 1997-2007 | 218 | 98 | 106 | 1 | 0 | 13 | 0.92 | 44.95 | 48.62 | 0.45 | 0.0 | 5.96 | 48.04 |
| **2** | MS Dhoni (IND) | 2007-2018 | 200 | 110 | 74 | 5 | 0 | 11 | 1.48 | 55.00 | 37.00 | 2.50 | 0.0 | 5.50 | 59.52 |
| **3** | A Ranatunga (SL) | 1988-1999 | 193 | 89 | 95 | 1 | 0 | 8 | 0.93 | 46.11 | 49.22 | 0.51 | 0.0 | 4.14 | 48.37 |
| **4** | AR Border (AUS) | 1985-1994 | 178 | 107 | 67 | 1 | 0 | 3 | 1.59 | 60.11 | 37.64 | 0.56 | 0.0 | 1.68 | 61.42 |

In [ ]:
```
#3. data set from webapi
players_batsman.head()
```

Out[ ]:

| | resource | id | country_id | firstname | lastname | fullname | image_path | dateofbirth | gender | battingstyle | b |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | players | 2 | 52126 | Ahmed | Shehzad | Ahmed Shehzad | https://cdn.sportmonks.com/images/cricket/play... | 1991-11-23 | m | right-hand-bat | |
| **3** | players | 5 | 52126 | Azhar | Ali | Azhar Ali | https://cdn.sportmonks.com/images/cricket/play... | 1985-02-19 | m | right-hand-bat | |
| **4** | players | 6 | 52126 | Fakhar | Zaman | Fakhar Zaman | https://cdn.sportmonks.com/images/cricket/play... | 1990-04-10 | m | left-hand-bat | |
| **5** | players | 7 | 52126 | Imam | ul Haq | Imam ul Haq | https://cdn.sportmonks.com/images/cricket/play... | 1995-12-12 | m | left-hand-bat | |
| **6** | players | 8 | 52126 | Babar | Azam | Babar Azam | https://cdn.sportmonks.com/images/cricket/play... | 1994-10-15 | m | right-hand-bat | |

## Creat SQL Database and store all three dataframes

In [ ]:
```
# Create a connection to the SQLite database
conn = sqlite3.connect('ProjectDatabase.db')
```

In [ ]:
```
# Create a cursor object
cursor = conn.cursor()
```

In [ ]:
```
# Store the first DataFrame in the database
cricket_data_df.to_sql('Excel-Cricket-data', conn, if_exists='replace', index=False)
```

Out[ ]:
```
90308
```

In [ ]:
```
# Store the second DataFrame in the database
Captian_df.to_sql('http-Captian-data', conn, if_exists='replace', index=False)
```

Out[ ]:
```
100
```

In [ ]:
```
# Store the third DataFrame in the database
players_batsman.to_sql('Webapi-batsman-data', conn, if_exists='replace', index=False)
```

Out[ ]:
```
4966
```

In [ ]:
```
# Close the connection to the database
conn.close()
```

## Combining the three datasets together

In [ ]:
```
Captian_df['LastName'] =Captian_df['Player'].str.split(' ').str[-2]
```

In [ ]:
```
Captian_df['LastName']
```

Out[ ]:
```
0       Ponting
1       Fleming
2         Dhoni
3     Ranatunga
4        Border
         ...
95      Shahidi
96     Johnston
97      Romaine
98          Ali
99       Borren
Name: LastName, Length: 100, dtype: object
```

In [ ]:
```
cricket_data_df['LastName'] =cricket_data_df['NAME'].str.split(' ').str[-1]
```

In [ ]:
```
cricket_data_df['LastName']
```

Out[ ]:
```
0              Arkell
1               Nyren
2            Maartensz
3               Lander
4           Kenderdine
              ...
90303         Robinson
90304    Hine-Haycock
90305           Hughes
90306          Clayton
90307          Clewitt
Name: LastName, Length: 90308, dtype: object
```

In [ ]:
```
Combined_project_df = cricket_data_df.merge(Captian_df,how='left',on='LastName')
```

In [ ]:
```
print(f"The size of Captian_df is {Captian_df.shape}.")
print(f"The size of cricket_data_df is {cricket_data_df.shape}.")
print(f"The size of Combined_project_df is {Combined_project_df.shape}.")
```

```
The size of Captian_df is (100, 16).
The size of cricket_data_df is (90308, 178).
The size of Combined_project_df is (92382, 193).
```

In [ ]:
```
#Creating a new column based on last name
players_batsman['LastName']=players_batsman['lastname']
```

In [ ]:
```
Combined_project_df = Combined_project_df.merge(players_batsman,how='inner',on='LastName')
```

In [ ]:
```
print(f"The size of players_batsman is {players_batsman.shape}.")
print(f"The size of Combined_project_df is {Combined_project_df.shape}.")
```

```
The size of players_batsman is (4966, 16).
The size of Combined_project_df is (697397, 208).
```

## Store the combined table in SQLite

In [ ]:
```
# Create a connection to the SQLite database
conn = sqlite3.connect('ProjectDatabase.db')
```

In [ ]:
```
# Create a cursor object
cursor = conn.cursor()
```

In [ ]:
```
Combined_project_df.columns.duplicated()
```

Out[ ]:
```
array([False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False])
```

In [ ]:
```
Combined_project_df.rename(columns={'id': 'project_id'}, inplace=True)
Combined_project_df.rename(columns={'lastname': 'lastname_'}, inplace=True)
```

In [ ]:
```
Combined_project_df.to_sql('Combined_project_df-data', conn, if_exists='replace', index=False)
```

Out[ ]:
```
697397
```

In [ ]:
```
# Close the connection to the database
conn.close()
```

In [ ]:
```
Combined_project_df.describe()
```

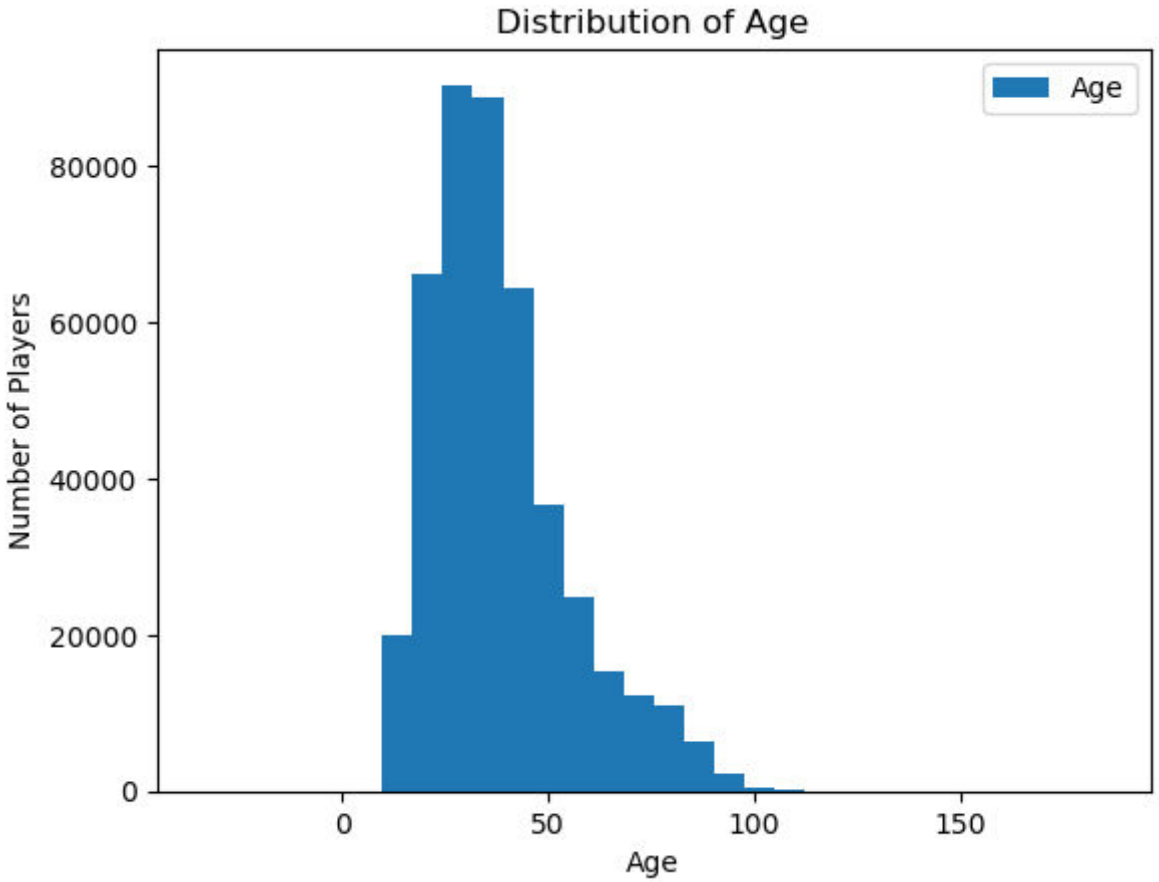
Out[ ]:

| | Unnamed: 0 | ID | Age | BATTING_Tests_Mat | BATTING_Tests_Inns | BATTING_Tests_NO | BATTING_Tests_Runs | BATTIN |
|---|---|---|---|---|---|---|---|---|
| **count** | 697397.000000 | 6.973970e+05 | 439062.000000 | 19426.000000 | 19136.000000 | 19136.000000 | 19136.000000 | 1 |
| **mean** | 32593.910890 | 4.248556e+05 | 38.452660 | 16.006435 | 25.444293 | 3.419367 | 664.055132 | |
| **std** | 18071.140143 | 4.307063e+05 | 16.647608 | 26.338697 | 42.481751 | 5.986866 | 1586.658648 | |
| **min** | 7.000000 | 3.933000e+03 | -34.000000 | 1.000000 | 1.000000 | 0.000000 | 0.000000 | |
| **25%** | 16318.000000 | 3.924000e+04 | 26.000000 | 2.000000 | 2.000000 | 0.000000 | 21.000000 | |
| **50%** | 34815.000000 | 3.006310e+05 | 35.000000 | 4.000000 | 7.000000 | 1.000000 | 69.000000 | |
| **75%** | 43219.000000 | 8.782270e+05 | 46.000000 | 14.000000 | 24.000000 | 3.000000 | 342.000000 | |
| **max** | 90306.000000 | 1.178645e+06 | 185.000000 | 200.000000 | 329.000000 | 86.000000 | 15921.000000 | |

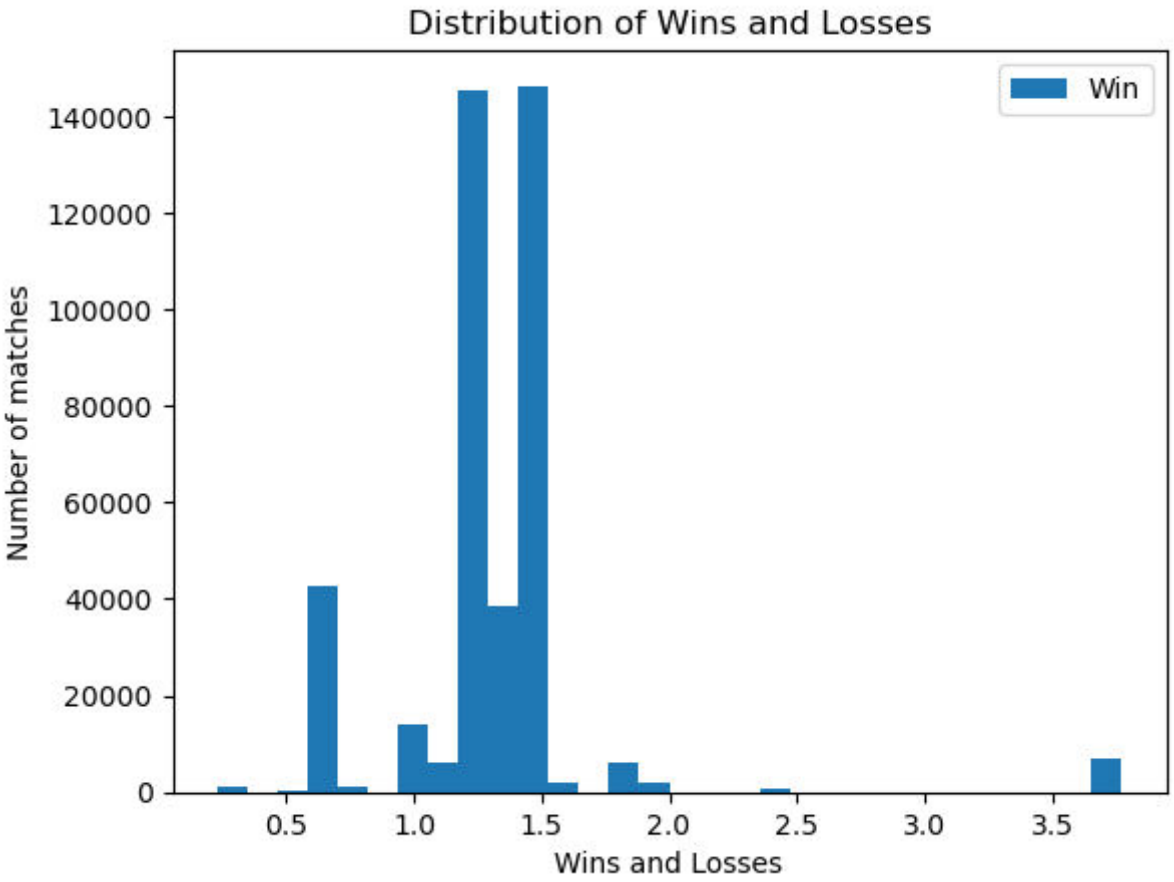
8 rows × 150 columns

In [ ]:

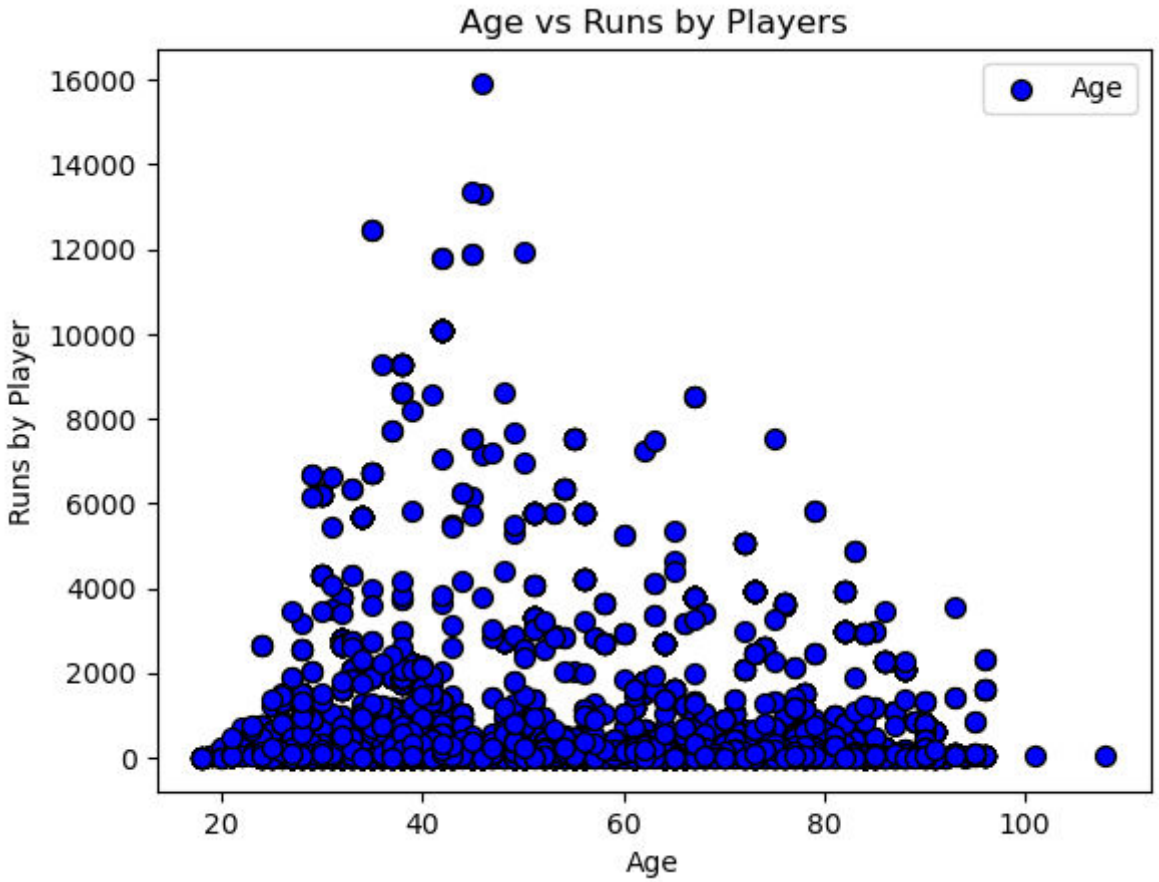
```
from matplotlib.pyplot import hist
import matplotlib.pyplot as plt
#Plot the hisory of the age disctribution
hist(Combined_project_df['Age'], bins = 30)
# Add labels and title
plt.xlabel('Age')
plt.ylabel('Number of Players')
plt.title('Distribution of Age')

# Add legend
labels = ['Age', 'Players']
plt.legend(labels)

# Show the plot
plt.show()
```



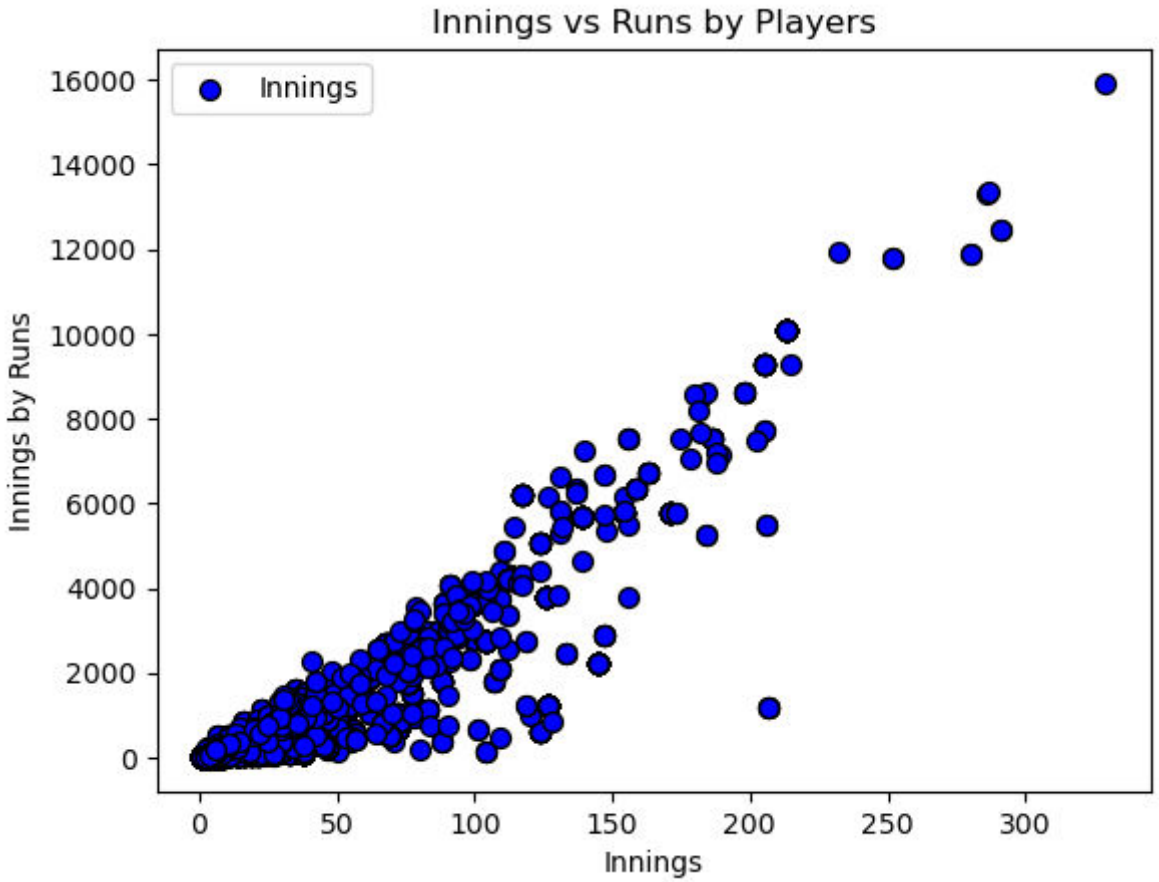

In [ ]:
```
hist(Combined_project_df['W/L'], bins = 30)
# Add labels and title
plt.xlabel('Wins and Losses')
plt.ylabel('Number of matches')
plt.title('Distribution of Wins and Losses')

# Add legend
labels = ['Win', 'Matches']
plt.legend(labels)

# Show the plot
plt.show()
```




In [ ]:
```
#scatter plots

plt.scatter(Combined_project_df['Age'], Combined_project_df['BATTING_Tests_Runs'], color='blue', edgecolors='black', s=

plt.xlabel('Age')
plt.ylabel('Runs by Player')
plt.title('Age vs Runs by Players')

# Add legend
labels = ['Age', 'Player Runs ']
plt.legend(labels)

# Show the plot
plt.show()
```




In [ ]:
```
plt.scatter(Combined_project_df['BATTING_Tests_Inns'], Combined_project_df['BATTING_Tests_Runs'], color='blue', edgecol

plt.xlabel('Innings')
plt.ylabel('Innings by Runs')
plt.title('Innings vs Runs by Players')

# Add Legend
labels = ['Innings', 'Player Runs ']
plt.legend(labels)

# Show the plot
plt.show()
```




## Learning on the project

1. The data can be available at multiple sources e.g. wexcel, odbc, access, html or a webapi
2. The data sceintiest shuold know how to pull the data from fifferent sources
3. The data sources should generally have some common key e.g. primary key and need to establish foreign key relationships
4. There are multiple databaes available in the market e.g. Oracle, MySQL, MongoDB, SQLite and other cloud based data sources .e.g. cosmos database
5. The cleaning, merging and processing of the data are the major activities and should be performed to ensure that the data sources are reliable.

## Ethical Implementation

The combined data represents the number of matches won by the countries, the payer details and statistics

1. The clubs may encourage only specific teams based on the past records.
2. The underground betting market may use this information to bet on the specific teams and/or players only.
3. Only specfic players may get a chance to play multiple matches or the selection will be based on the playesr who did best againt specific teams
4. The selection process will be purelye based on perfoamce. Some played may do good with specific teams and may get chosen for the same opponent.
5. Players may develop skills in specific areas only

In [ ]: