Project Overview

This project is intended to determine cricket's best test XI of all time (statistically) based on a problem statement and given criteria. The final results are to be displayed on an interactive Power BI Dashboard and validated using SQL queries.

Database Used: Cricinfo (https://www.espncricinfo.com/)

Problem Statement

Build cricket's best test 11 of all time (statistically) - The team should consist of 5 batters, 2 all-rounders, 1 wicket-keeper and 3 bowlers.

The individual criteria for each role should be based on the below -

1. Batters -
    - Matches played more than 100
    - Total career runs should be more than 10,000
    - Centuries more than 30
    - Average more than 50
    - Career span of more than 12 years

---

2. Bowlers -
    - Matches played more than 100
    - Total career wickets should be more than 500
    - Strike Rate of less than 60
    - Average less than 30
    - Career span of more than 12 years

---

3. All-Rounders -
    - Matches played more than 100
    - Total career runs more than 5000 and wickets should be more than 200
    - Career span of more than 12 years

---

4. Wicket Keepers -
    - Matches played more than 100
    - Total career Wk_Dismissals more than 300
    - Total career Dismissals_per_innings more than 1.5
    - Career span of more than 12 years

Import python libraries & data research

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pylab import rcParams
from IPython.display import Image

%matplotlib inline
rcParams['figure.figsize']=8,4
```

In [2]:
```
# webscraping

batting_url = 'https://www.espncricinfo.com/records/most-runs-in-career-223646'
bowling_url = 'https://www.espncricinfo.com/records/most-wickets-in-career-93276'
allrounder_url = 'https://www.espncricinfo.com/records/1000-runs-and-100-wickets-28
wkeep_url = 'https://www.espncricinfo.com/records/most-dismissals-in-career-283791'
```

In [3]:
```
# read html

bat = pd.read_html(batting_url)
bowl = pd.read_html(bowling_url)
all_round = pd.read_html(allrounder_url)
wicket_keeping = pd.read_html(wkeep_url)
```

In [4]:
```
# assign to dataframes

batters = bat[0]
bowlers = bowl[0]
all_rounders = all_round[0]
wicket_keepers = wicket_keeping[0]
```

In [5]:
```
# Show entire rows and columns when called (useful for quickly glancing through all

pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
```

In [6]:
```
batters.head(2)
```

Out[6]:

	Player	Span	Mat	Inns	NO	Runs	HS	Ave	BF	SR	100	50	0	4s	6s
0	SR Tendulkar (IND)	1989-2013	200	329	33	15921	248*	53.78	29437+	54.04	51	68	14	2058+	69
1	RT Ponting (AUS)	1995-2012	168	287	29	13378	257	51.85	22782	58.72	41	62	17	1509	73

In [7]:
```
bowlers.head(2)
```

Out[7]:

	Player	Span	Mat	Inns	Balls	Overs	Mdns	Runs	Wkts	BBI	Ave	Econ	SR	
0	M Muralidaran (ICC/SL)	1992-2010	133	230	44039	7339.5	1794	18180	800	9/51	22.72	2.47	55.04	4
1	SK Warne (AUS)	1992-2007	145	273	40705	6784.1	1761	17995	708	8/71	25.41	2.65	57.49	4

In [8]:
```
all_rounders.head(2)
```

Out[8]:

	Player	Span	Mat	Runs	HS	Ave	100	Wkts	BBI	Ave.1	5	Ct	St
0	G Giffen (AUS)	1881-1896	31	1238	161	23.35	1	103	7/117	27.09	7	24	-
1	MA Noble (AUS)	1898-1909	42	1997	133	30.25	1	121	7/17	25.00	9	26	-

In [9]:
```
wicket_keepers.head(2)
```

Out[9]:

	Player	Span	Mat	Inns	Dis	Ct	St	Max Dis Inns	Dis/Inn
0	MV Boucher (ICC/SA)	1997-2012	147	281	555	532	23	6 (6ct 0st)	1.975
1	AC Gilchrist (AUS)	1999-2008	96	191	416	379	37	5 (5ct 0st)	2.178

In [10]:
```
print("Shape of batters table:", batters.shape)
print("Shape of bowlers table:", bowlers.shape)
print("Shape of all_rounders table:", all_rounders.shape)
print("Shape of wicket_keepers table:", wicket_keepers.shape)
```

```
Shape of batters table: (107, 15)
Shape of bowlers table: (82, 15)
Shape of all_rounders table: (75, 13)
Shape of wicket_keepers table: (90, 9)
```

In [11]:
```
# Merge dataframes

best = pd.merge(batters,bowlers, left_on = ["Player","Span","Mat"], right_on = ["Pl
                ).merge(all_rounders, left_on = ["Player","Span","Mat"], right_or
                       ).merge(wicket_keepers, on = ["Player","Span","Mat"], how
```

In [12]:
```
best.head(3)
```

Out[12]:

	Player	Span	Mat	Inns_bat	NO	Runs_bat	HS_x	Ave_bat	BF	SR_bat	100_x	50
0	SR Tendulkar (IND)	1989-2013	200	329.0	33.0	15921.0	248*	53.78	29437+	54.04	51.0	68.0
1	RT Ponting (AUS)	1995-2012	168	287.0	29.0	13378.0	257	51.85	22782	58.72	41.0	62.0
2	JH Kallis (ICC/SA)	1995-2013	166	280.0	40.0	13289.0	224	55.37	28903	45.97	45.0	58.0

In [13]:
```
print("Shape of best table:", best.shape)
```

```
Shape of best table: (294, 43)
```

Data Cleaning

- Create a 'Role' column (batter, bowler, allrounder, wk) right next to 'player' column
- Separate country name from player name and create the new 'Country' column right next to 'Speciality' column.
- Split 'span' column and have 2 separate columns for start and end dates, create a new column 'Career Span'
- Remove unwanted columns, merge common columns and add new columns if required
- Remove unwanted characters (e.g. *, +) from the columns
- Check for missing values (and fill if found)
- Check and drop duplicates (if found)
- Check and change data types
- Rename column headers to something more generally comprehendable & rearrange column indexes

In [14]:
```
# check dataframe general info

best.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 294 entries, 0 to 293
Data columns (total 43 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Player        294 non-null    object 
 1   Span          294 non-null    object 
 2   Mat           294 non-null    int64  
 3   Inns_bat      107 non-null    float64
 4   NO            107 non-null    float64
 5   Runs_bat      107 non-null    float64
 6   HS_x          107 non-null    object 
 7   Ave_bat       107 non-null    float64
 8   BF            107 non-null    object 
 9   SR_bat        107 non-null    float64
 10  100_x         107 non-null    float64
 11  50            107 non-null    float64
 12  0             107 non-null    float64
 13  4s            107 non-null    object 
 14  6s            107 non-null    object 
 15  Inns_bowl     82 non-null     float64
 16  Balls         82 non-null     float64
 17  Overs         82 non-null     object 
 18  Mdns          82 non-null     float64
 19  Runs_bowl     82 non-null     float64
 20  Wkts_x        82 non-null     float64
 21  BBI_x         82 non-null     object 
 22  Ave_bowl      82 non-null     float64
 23  Econ          82 non-null     float64
 24  SR_bowl       82 non-null     float64
 25  4             82 non-null     float64
 26  5_x           82 non-null     float64
 27  Runs          75 non-null     float64
 28  HS_y          75 non-null     object 
 29  Ave           75 non-null     float64
 30  100_y         75 non-null     object 
 31  Wkts_y        75 non-null     float64
 32  BBI_y         75 non-null     object 
 33  Ave.1         75 non-null     float64
 34  5_y           75 non-null     float64
 35  Ct_x          75 non-null     float64
 36  St_x          75 non-null     object 
 37  Inns          90 non-null     float64
 38  Dis           90 non-null     float64
 39  Ct_y          90 non-null     float64
 40  St_y          90 non-null     float64
 41  Max Dis Inns  90 non-null     object 
 42  Dis/Inn       90 non-null     float64
dtypes: float64(29), int64(1), object(13)
memory usage: 101.1+ KB
```

In [15]:
```
# Create a 'Role' column (batter, bowler, allrounder, wk)

best.loc[best['Player'].isin(batters['Player']), 'Role'] = 'Batter'
best.loc[best['Player'].isin(all_rounders['Player']), 'Role'] = 'All_Rounder'
best.loc[best['Player'].isin(batters['Player']) & best['Player'].isin(bowlers['Play
best.loc[best['Player'].isin(bowlers['Player']) & ~best['Player'].isin(batters['Pla
best.loc[best['Player'].isin(bowlers['Player']) & ~best['Player'].isin(batters['Pla
best.loc[best['Player'].isin(wicket_keepers['Player']) & ~best['Player'].isin(batte
best.loc[best['Player'].isin(batters['Player']) & best['Player'].isin(wicket_keeper
```

In [16]:
```
# place it right next to 'player' column

```

```
role = best.pop('Role')
best.insert(1, role.name, role)
```

In [17]:
```
# Separate country name from player name in have it in a new column

best["Country"] = best["Player"].str.split(pat = '(').str[1]
```

In [18]:
```
best["Country"] = best["Country"].str.split(pat = ')').str[0]
```

In [19]:
```
best["Country"].head(3)
```

Out[19]:
```
0       IND
1       AUS
2    ICC/SA
Name: Country, dtype: object
```

In [20]:
```
best["Player"] = best["Player"].str.split(pat = '(').str[0]

best["Player"].head(3)
```

Out[20]:
```
0    SR Tendulkar
1      RT Ponting
2       JH Kallis
Name: Player, dtype: object
```

In [21]:
```
# Move 'Country' column next to 'Role' column

col = best.pop('Country')
best.insert(2, col.name, col)

best.head(2)
```

Out[21]:

	Player	Role	Country	Span	Mat	Inns_bat	NO	Runs_bat	HS_x	Ave_bat	BF	SR_b
0	SR Tendulkar	Batter	IND	1989-2013	200	329.0	33.0	15921.0	248*	53.78	29437+	54.
1	RT Ponting	Batter	AUS	1995-2012	168	287.0	29.0	13378.0	257	51.85	22782	58.

In [22]:
```
# Split 'span' column and have 2 separate columns for start and end dates

best["Start_Year"] = best["Span"].str.split(pat = '-').str[0].astype('int64')
best["End_Year"] = best["Span"].str.split(pat = '-').str[1].astype('int64')

best.head(3)
```

Out[22]:

	Player	Role	Country	Span	Mat	Inns_bat	NO	Runs_bat	HS_x	Ave_bat	BF
0	SR Tendulkar	Batter	IND	1989-2013	200	329.0	33.0	15921.0	248*	53.78	29437+
1	RT Ponting	Batter	AUS	1995-2012	168	287.0	29.0	13378.0	257	51.85	22782
2	JH Kallis	All_Rounder	ICC/SA	1995-2013	166	280.0	40.0	13289.0	224	55.37	28903

In [23]:
```
# Add a new column for calcuating 'career span'

best["Career_Span(yrs)"] = best["End_Year"] - best["Start_Year"]

best.head(2)
```

Out[23]:

	Player	Role	Country	Span	Mat	Inns_bat	NO	Runs_bat	HS_x	Ave_bat	BF	SR_b
0	SR Tendulkar	Batter	IND	1989-2013	200	329.0	33.0	15921.0	248*	53.78	29437+	54.
1	RT Ponting	Batter	AUS	1995-2012	168	287.0	29.0	13378.0	257	51.85	22782	58.

In [24]:
```
# remove unwanted columns

best.drop(best.columns[[3,5,6,10,13,14,15,16,17,18,19,20,21,23,27,28,34,36,37,38,39
```

In [25]:
```
best.head(3)
```

Out[25]:

	Player	Role	Country	Mat	Runs_bat	HS_x	Ave_bat	SR_bat	100_x	Wkts_x	Ave_b
0	SR Tendulkar	Batter	IND	200	15921.0	248*	53.78	54.04	51.0	NaN	N
1	RT Ponting	Batter	AUS	168	13378.0	257	51.85	58.72	41.0	NaN	N
2	JH Kallis	All_Rounder	ICC/SA	166	13289.0	224	55.37	45.97	45.0	292.0	32

In [26]:
```
# combine columns with common values

best["Batting_Average"] = best[["Ave_bat", "Ave"]].max(axis=1)
best["Bowling_Average"] = best[["Ave_bowl", "Ave.1"]].max(axis=1)
best["100s"] = best[["100_x", "100_y"]].max(axis=1)
best["Wickets"] = best[["Wkts_x", "Wkts_y"]].max(axis=1)
best["Runs_Scored"] = best[["Runs_bat", "Runs"]].max(axis=1)
```

```
C:\Users\Hryshikesh\AppData\Local\Temp\ipykernel_18476\2424287191.py:5: FutureWarn
ing: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=Non
e') is deprecated; in a future version this will raise TypeError.  Select only val
id columns before calling the reduction.
  best["100s"] = best[["100_x", "100_y"]].max(axis=1)
```

In [27]:
```
best.drop(columns=["Ave_bat","Ave","Ave_bowl", "Ave.1","100_x","100_y","Wkts_x","Wk
```

In [28]:
```
best.head(3)
```

Out[28]:

	Player	Role	Country	Mat	HS_x	Econ	SR_bowl	HS_y	Dis	Ct_y	St_y	Max Dis Inns	Dis/
0	SR Tendulkar	Batter	IND	200	248*	NaN	NaN	NaN	NaN	NaN	NaN	NaN	I
1	RT Ponting	Batter	AUS	168	257	NaN	NaN	NaN	NaN	NaN	NaN	NaN	I
2	JH Kallis	All_Rounder	ICC/SA	166	224	2.82	69.28	224	NaN	NaN	NaN	NaN	I

In [29]:
```
best.head(3)
```

Out[29]:

	Player	Role	Country	Mat	HS_x	Econ	SR_bowl	HS_y	Dis	Ct_y	St_y	Max Dis Inns	Dis/
0	SR Tendulkar	Batter	IND	200	248*	NaN	NaN	NaN	NaN	NaN	NaN	NaN	I
1	RT Ponting	Batter	AUS	168	257	NaN	NaN	NaN	NaN	NaN	NaN	NaN	I
2	JH Kallis	All_Rounder	ICC/SA	166	224	2.82	69.28	224	NaN	NaN	NaN	NaN	I

In [30]:
```
# Check and drop duplicates

best.duplicated().sum()
```

Out[30]:
```
0
```

In [31]:
```
# Check and change data types

best.dtypes
```

Out[31]:
```
Player               object
Role                 object
Country              object
Mat                   int64
HS_x                 object
Econ                float64
SR_bowl             float64
HS_y                 object
Dis                 float64
Ct_y                float64
St_y                float64
Max Dis Inns         object
Dis/Inn             float64
Start_Year            int64
End_Year              int64
Career_Span(yrs)      int64
Batting_Average     float64
Bowling_Average     float64
100s                float64
Wickets             float64
Runs_Scored         float64
dtype: object
```

In [32]:
```
# Remove unwanted characters (e.g. *, +) from the columns & change data types as re

#best["Mat"] = best["Mat"].str.replace(r'\W','').astype('Int64')
best["HS_x"] = best["HS_x"].str.replace(r'\W','').astype('Int64')
best["HS_y"] = best["HS_y"].str.replace(r'\W','').astype('Int64')
best["Dis"] = best["Dis"].astype('Int64')
best["Ct_y"] = best["Ct_y"].astype('Int64')
best["St_y"] = best["St_y"].astype('Int64')
best["100s"] = best["100s"].astype('Int64')
best["Wickets"] = best["Wickets"].astype('Int64')
best["Runs_Scored"] = best["Runs_Scored"].astype('Int64')
```

```
C:\Users\Hryshikesh\AppData\Local\Temp\ipykernel_18476\3964250371.py:4: FutureWarn
ing: The default value of regex will change from True to False in a future versio
n.
  best["HS_x"] = best["HS_x"].str.replace(r'\W','').astype('Int64')
C:\Users\Hryshikesh\AppData\Local\Temp\ipykernel_18476\3964250371.py:5: FutureWarn
ing: The default value of regex will change from True to False in a future versio
n.
  best["HS_y"] = best["HS_y"].str.replace(r'\W','').astype('Int64')
```

In [33]:
```
best["Highest_Score"] = best[["HS_x", "HS_y"]].max(axis=1)
```

In [34]:
```
best.drop(columns=["HS_x", "HS_y"], inplace=True)
```

In [35]:
```
best.head(2)
```

Out[35]:

	Player	Role	Country	Mat	Econ	SR_bowl	Dis	Ct_y	St_y	Max Dis Inns	Dis/Inn	Start_Yea
0	SR Tendulkar	Batter	IND	200	NaN	NaN	<NA>	<NA>	<NA>	NaN	NaN	198
1	RT Ponting	Batter	AUS	168	NaN	NaN	<NA>	<NA>	<NA>	NaN	NaN	199

In [36]:
```
# Rename column headers to something more generally comprehendable

best = best.rename(columns={"Player":"Player_Name","Mat":"Matches","Econ":"Economy'
```

In [37]:
```
best.head(3)
```

Out[37]:

	Player_Name	Role	Country	Matches	Economy	Bowling_Strike_Rate	Wk_Dismissals	Ca
0	SR Tendulkar	Batter	IND	200	NaN	NaN	<NA>	
1	RT Ponting	Batter	AUS	168	NaN	NaN	<NA>	
2	JH Kallis	All_Rounder	ICC/SA	166	2.82	69.28	<NA>	

In [38]:
```
# swap/reshuffle columns to put in a better order

best = best.reindex(columns=["Player_Name","Role","Country","Matches","Runs_Scored'
```

In [39]:
```
# check for null values

best.isnull().sum()
```

Out[39]:
```
Player_Name                0
Role                       0
Country                    0
Matches                    0
Runs_Scored              118
Highest_Score            118
Batting_Average          118
100s                     187
Wickets                  182
Bowling_Average          182
Bowling_Strike_Rate      212
Economy                  212
Wk_Dismissals            204
Catches                  204
Wk_Stumpings             204
Dismissals_per_innings   204
Max Dis Inns             204
Start_Year                 0
End_Year                   0
Career_Span(yrs)           0
dtype: int64
```

In [40]:
```
# fill null values with 0

best.fillna(value = 0, inplace=True)
```

In [41]:
```
best.head(3)
```

Out[41]:

	Player_Name	Role	Country	Matches	Runs_Scored	Highest_Score	Batting_Average	10
0	SR Tendulkar	Batter	IND	200	15921	248.0	53.78	
1	RT Ponting	Batter	AUS	168	13378	257.0	51.85	
2	JH Kallis	All_Rounder	ICC/SA	166	13289	224.0	55.37	

Problem Analysis

In [42]:
```
# to extract player names who fall within the problem statement criteria

best_batters = best[(best["Role"]=="Batter") & (best["Matches"]>100) & (best["Runs_
best_bowlers = best[(best["Role"]=="Bowler") & (best["Matches"]>100) & (best["Wicke
best_all_rounders = best[(best["Role"]=="All_Rounder") & (best["Runs_Scored"]>5000)
best_wicket_keepers = best[(best["Role"]=="Wicket_Keeper") & (best["Matches"]>100)
```

In [43]:
```
#export to excel (for dashboard)

best_batters.to_excel('best_batters.xlsx')
```

In [44]:
```
best_bowlers.to_excel('best_bowlers.xlsx')
```

In [45]:
```
best_all_rounders.to_excel('best_all_rounders.xlsx')
```

In [46]:
```
best_wicket_keepers.to_excel('best_wicket_keepers.xlsx')
```

In [47]:
```
best_batters
```

Out[47]:

	Player_Name	Role	Country	Matches	Runs_Scored	Highest_Score	Batting_Average	100s
0	SR Tendulkar	Batter	IND	200	15921	248.0	53.78	51
1	RT Ponting	Batter	AUS	168	13378	257.0	51.85	41
3	R Dravid	Batter	ICC/IND	164	13288	270.0	52.31	36
6	BC Lara	Batter	ICC/WI	131	11953	400.0	52.88	34
11	SR Waugh	Batter	AUS	168	10927	200.0	51.06	32
12	SM Gavaskar	Batter	IND	125	10122	236.0	51.12	34
13	Younis Khan	Batter	PAK	118	10099	313.0	52.05	34

In [48]:
```
best_bowlers
```

Out[48]:

	Player_Name	Role	Country	Matches	Runs_Scored	Highest_Score	Batting_Average	100s
107	M Muralidaran	Bowler	ICC/SL	133	1261	67.0	11.67	0
108	SK Warne	Bowler	AUS	145	3154	99.0	17.32	0
109	JM Anderson	Bowler	ENG	187	1353	81.0	8.96	0
111	SCJ Broad	Bowler	ENG	167	3662	169.0	18.03	0
112	GD McGrath	Bowler	AUS	124	0	0.0	0.00	0
114	CA Walsh	Bowler	WI	132	0	0.0	0.00	0

In [49]:
```
best_all_rounders
```

Out[49]:

	Player_Name	Role	Country	Matches	Runs_Scored	Highest_Score	Batting_Average
2	JH Kallis	All_Rounder	ICC/SA	166	13289	224.0	55.37
33	GS Sobers	All_Rounder	WI	93	8032	365.0	57.78
98	N Kapil Dev	All_Rounder	IND	131	5248	163.0	31.05
101	IT Botham	All_Rounder	ENG	102	5200	208.0	33.54

In [50]:
```
best_wicket_keepers
```

Out[50]:

	Player_Name	Role	Country	Matches	Runs_Scored	Highest_Score	Batting_Average
83	MV Boucher	Wicket_Keeper	ICC/SA	147	5515	125.0	30.3

Final result based on problem analysis

In [51]:
```
# So as per the problem analysis we have -
# 7 batters, 6 bowlers, 4 all rounders and 1 wicket keeper who meet the best 11 cri

# But we need the best 11 only

# So to select the best 11 (5 batter, 2 ar, 1 wk, 3 bowlers) out of these 18 player

# I am selecting - top 5 batters by runs, 2 all rounders (1 with highest runs and 1

ar = (best_all_rounders.nlargest(1, "Runs_Scored") + best_all_rounders.nlargest(1,

best_11 = pd.merge(best_batters.nlargest(5, "Runs_Scored"),ar, how="outer"
                   ).merge(best_wicket_keepers, how='outer'
```

```
                        ).merge(best_bowlers.nlargest(3, "Wickets"),how='outer'
                               ).reset_index(drop=True)
best_11
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\reshape\merge.py:916: Futur
eWarning: In a future version, the Index constructor will not infer numeric dtypes
when passed object-dtype sequences (matching Series behavior)
  key_col = Index(lvals).where(~mask_left, rvals)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\reshape\merge.py:916: Futur
eWarning: In a future version, the Index constructor will not infer numeric dtypes
when passed object-dtype sequences (matching Series behavior)
  key_col = Index(lvals).where(~mask_left, rvals)
```

Out[51]:

	Player_Name	Role	Country	Matches	Runs_Scored	Highest_Score	Batting_Average
0	SR Tendulkar	Batter	IND	200.0	15921	248.0	53.78
1	RT Ponting	Batter	AUS	168.0	13378	257.0	51.85
2	R Dravid	Batter	ICC/IND	164.0	13288	270.0	52.31
3	BC Lara	Batter	ICC/WI	131.0	11953	400.0	52.88
4	SR Waugh	Batter	AUS	168.0	10927	200.0	51.06
5	JH Kallis	All_Rounder	ICC/SA	166.0	13289	224.0	55.37
6	N Kapil Dev	All_Rounder	IND	131.0	5248	163.0	31.05
7	MV Boucher	Wicket_Keeper	ICC/SA	147.0	5515	125.0	30.30
8	M Muralidaran	Bowler	ICC/SL	133.0	1261	67.0	11.67
9	SK Warne	Bowler	AUS	145.0	3154	99.0	17.32
10	JM Anderson	Bowler	ENG	187.0	1353	81.0	8.96

Power BI Dashboard Snapshots

In [52]:

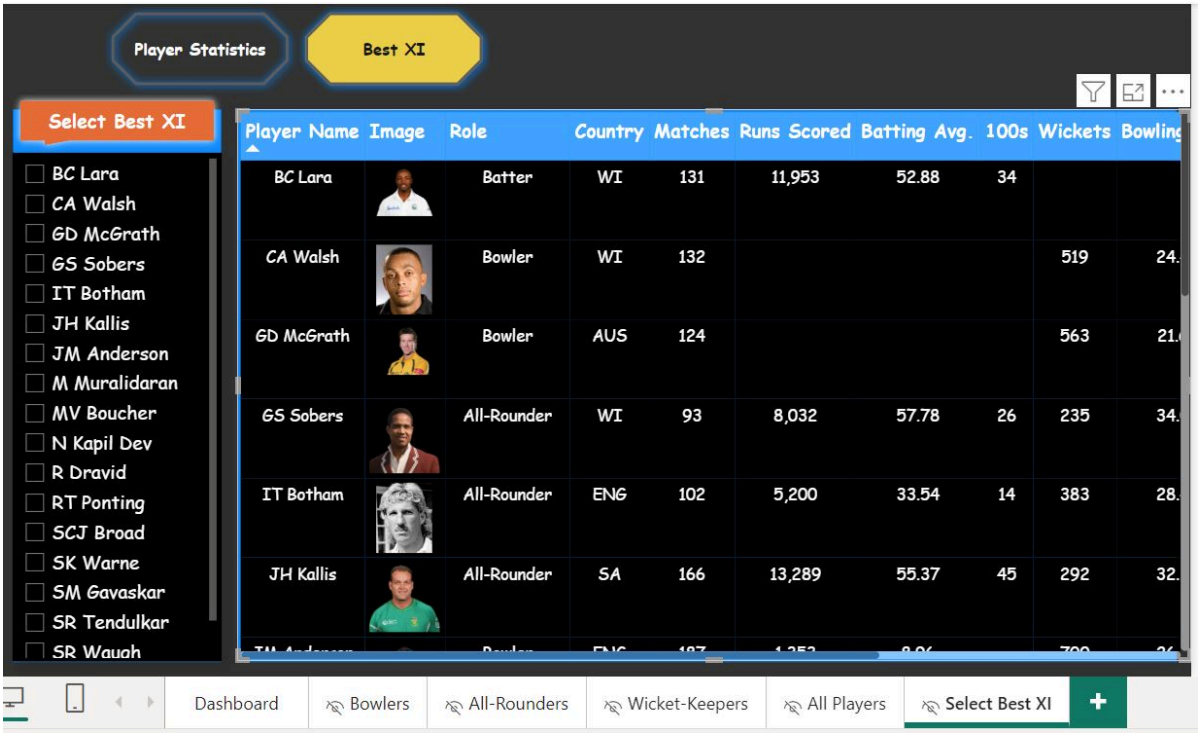
```
from IPython.display import Image
Image(r'C:\Users\Hryshikesh\Desktop\DA Projects\Projects\Cricket\Dashboard_Image1.j
```

Out[52]:

In [53]:

```
from IPython.display import Image
Image(r'C:\Users\Hryshikesh\Desktop\DA Projects\Projects\Cricket\Dashboard_BestXI.j
```

Out[53]:



Result validation using SQL Queries and Output - Best XI

In [54]:

```
# SQL Query

from IPython.display import Image
Image(r'C:\Users\Hryshikesh\Desktop\DA Projects\Projects\Cricket\SQL query.jpg')
```

Out[54]:

Database name – All_Players (combined players table)

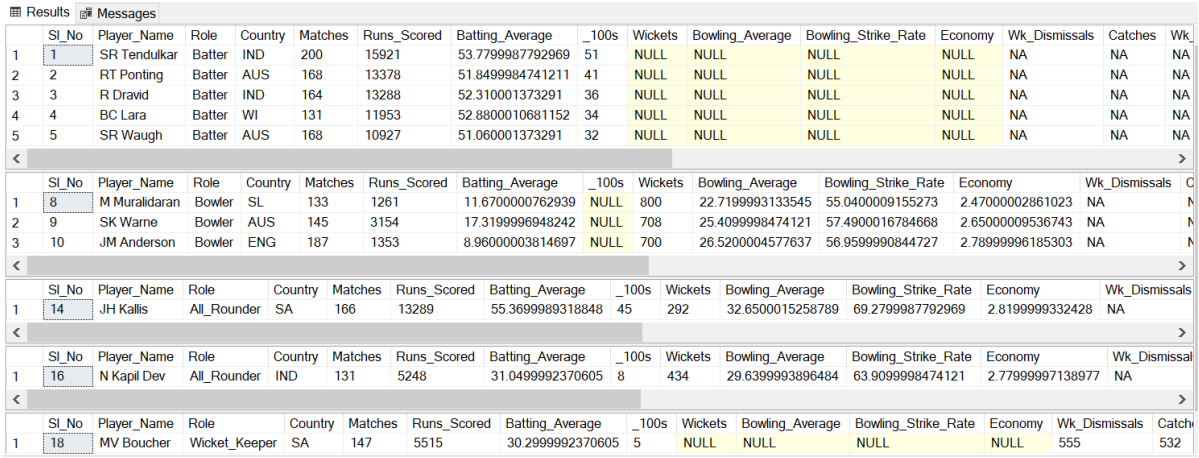
Query -

```
select top 5 * from All_Players
where Role='Batter'
order by Runs_Scored desc                --top 5 batters with most runs

select top 3 * from All_Players
where Role='Bowler'
order by Wickets desc              --top 3 bowlers with most wickets

select * from All_Players
where role = 'All_Rounder'
and Runs_Scored=(select MAX(Runs_Scored) from All_Players where role = 'All_Rounder') -- all_rounder
with highest runs

select * from All_Players
where role = 'All_Rounder'
and Wickets = (select MAX(Wickets) from All_Players where role = 'All_Rounder')    -- all_rounder with
highest wickets

select * from All_Players
where Role='Wicket_Keeper'         --wicket-keeper
```

In [55]:

```
# SQL Output

from IPython.display import Image
Image(r'C:\Users\Hryshikesh\Desktop\DA Projects\Projects\Cricket\SQL Output.png')
```

Out[55]:



Results Messages

	Sl_No	Player_Name	Role	Country	Matches	Runs_Scored	Batting_Average	_100s	Wickets	Bowling_Average	Bowling_Strike_Rate	Economy	Wk_Dismissals	Catches	Wk
1	1	SR Tendulkar	Batter	IND	200	15921	53.7799987792969	51	NULL	NULL	NULL	NULL	NA	NA	NA
2	2	RT Ponting	Batter	AUS	168	13378	51.8499984741211	41	NULL	NULL	NULL	NULL	NA	NA	NA
3	3	R Dravid	Batter	IND	164	13288	52.310001373291	36	NULL	NULL	NULL	NULL	NA	NA	NA
4	4	BC Lara	Batter	WI	131	11953	52.8800010681152	34	NULL	NULL	NULL	NULL	NA	NA	NA
5	5	SR Waugh	Batter	AUS	168	10927	51.060001373291	32	NULL	NULL	NULL	NULL	NA	NA	NA

	Sl_No	Player_Name	Role	Country	Matches	Runs_Scored	Batting_Average	_100s	Wickets	Bowling_Average	Bowling_Strike_Rate	Economy	Wk_Dismissals	C
1	8	M Muralidaran	Bowler	SL	133	1261	11.6700000762939	NULL	800	22.7199993133545	55.0400009155273	2.47000002861023	NA	
2	9	SK Warne	Bowler	AUS	145	3154	17.3199996948242	NULL	708	25.4099998474121	57.4900016784668	2.65000009536743	NA	
3	10	JM Anderson	Bowler	ENG	187	1353	8.96000003814697	NULL	700	26.5200004577637	56.9599990844727	2.78999996185303	NA	

	Sl_No	Player_Name	Role	Country	Matches	Runs_Scored	Batting_Average	_100s	Wickets	Bowling_Average	Bowling_Strike_Rate	Economy	Wk_Dismissals
1	14	JH Kallis	All_Rounder	SA	166	13289	55.3699989318848	45	292	32.6500015258789	69.2799987792969	2.8199999332428	NA

	Sl_No	Player_Name	Role	Country	Matches	Runs_Scored	Batting_Average	_100s	Wickets	Bowling_Average	Bowling_Strike_Rate	Economy	Wk_Dismissal
1	16	N Kapil Dev	All_Rounder	IND	131	5248	31.0499992370605	8	434	29.6399993896484	63.9099998474121	2.77999997138977	NA

	Sl_No	Player_Name	Role	Country	Matches	Runs_Scored	Batting_Average	_100s	Wickets	Bowling_Average	Bowling_Strike_Rate	Economy	Wk_Dismissals	Catch
1	18	MV Boucher	Wicket_Keeper	SA	147	5515	30.2999992370605	5	NULL	NULL	NULL	NULL	555	532

By - Hryshikesh Dihingia

Thank You :)