PROBLEM DEFINITION & ANALYSIS

The hardest part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirement. Defining and applying good, complete requirements are hard to work, and success in this endeavor has eluded many of us. yet. we continue to make progress.

The quality of a software product is only as good as the process that creates it. Problem definition is one of the most crucial steps in this creation process. Without defining a problem, developers do not know what to build, customers do not know what to expect, and there is no way to validate that the built system satisfies the requirement.

Problem definition and Analysis is the activity that encompasses learning about the problem to be solved, understanding the needs of customers and users, trying to find out who the user is, and understanding all the constraints on the solution. It includes all activities related to the following:-

- ✓ Identification and documentation of customers' or users' needs.
- ✓ Creation of a document that describes the external behavior and the association constraints that will satisfy those needs.
- ✓ Analysis and validation of the requirements documents to ensure consistency, completeness, and feasibility.

After the analysis of the functioning of the IPL Data Analysis system, the proposed system is expected to do the following: -

- ✓ To provide a user-friendly, Graphical User Interface (GUI) based integrated and centralized environment for computerized management.
- ✓ The proposed system should maintain all the records and generate the required reports and information when required.
- ✓ To provide efficient and secured Information storage, flow, and retrieval system, ensuring the integrity and validity of records.
- ✓ To provide a graphical and user-friendly interface to interact with a centralized database based on client-server architecture.
- ✓ To identify the critical operation procedure and possibilities of simplification using modern IT tools and practices.

SYSTEM IMPLEMENTATION

The Hardware used:

While developing the system, the used hardware is:

- ✓ PC with AMD Ryzen5 4600u processor
- ✓ 8.00 GB RAM
- ✓ HDMI
- ✓ USB3.0, USB2.0, And other required devices.

The software used:

- ✓ Microsoft Windows 10 as Operating System.
- ✓ Python 3.9.1
- ✓ Python libraries
  - Pandas
  - Matplotlib
  - Pandas table
  - Tkinter
  - MySQL connector
- ✓ PyCharm community edition 2020.3.2
- ✓ MySQL 8.0 as Back-end Server, CSV files made with Excel365 for Databases.
- ✓ Microsoft Word Professional365 for documentation.

Testing of software for compatibility

The Hardware used:

After developing the software, it's tested on hardware:

- ✓ PC with AMD Ryzen5 4600u processor
- ✓ PC with ITEL i5 10thGen processor
- ✓ PC with ITEL i7 8thGen processor
- ✓ All with 8.00GB RAM

The software used:

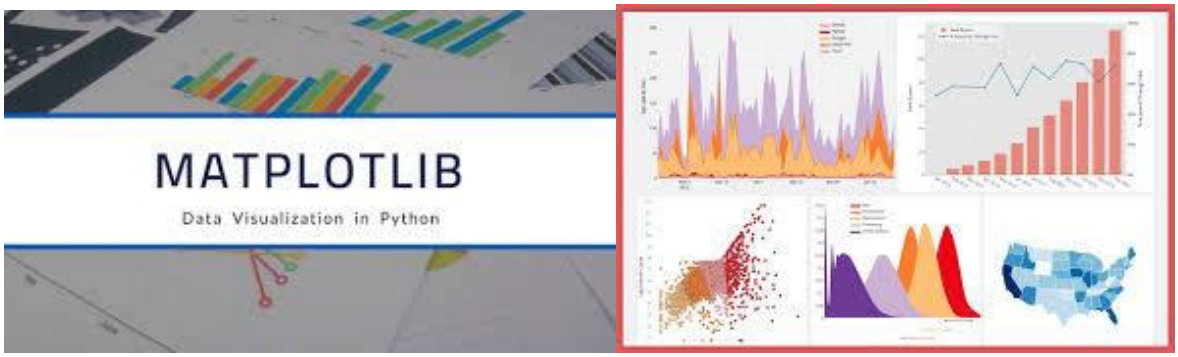
- ✓ Microsoft Windows 10 as Operating System.
- ✓ Microsoft Windows 11 as Operating System.
- ✓ Microsoft Windows 7 as Operating System.
- ✓ Python 3.9.1
- ✓ Python 3.8.3

Python libraries

Pandas:

Pandas is a high-level data manipulation tool developed by **Wes McKinney**. It is built on the NumPy package, and its key data structure is called the Dataframe. Dataframe allow you to store and manipulate tabular data in rows of observations and columns of variables.

Matplotlib:



The matplotlib Python library, developed by **John Hunter** and many other contributors, is used to create high-quality graphs, charts, and figures. The library is extensive and capable of changing very minute details of a figure.

Tkinter:

The Tkinter is the python library **Fredrik Lundh.** It supports a range of Tcl/Tk versions, built either with or without thread support. The official Python binary release bundles Tcl/Tk 8.6 threaded.

PYTHON: OVERVIEW

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **PYTHON IS INTERPRETED** – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.
- **PYTHON IS INTERACTIVE** – You can sit at a Python prompt and interact with the interpreter directly to write your programs.
- **PYTHON IS OBJECT-ORIENTED** – Python supports the Object-Oriented style or technique of programming that encapsulates code within objects.
- **PYTHON IS A BEGINNER'S LANGUAGE** – Python is a great language for beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

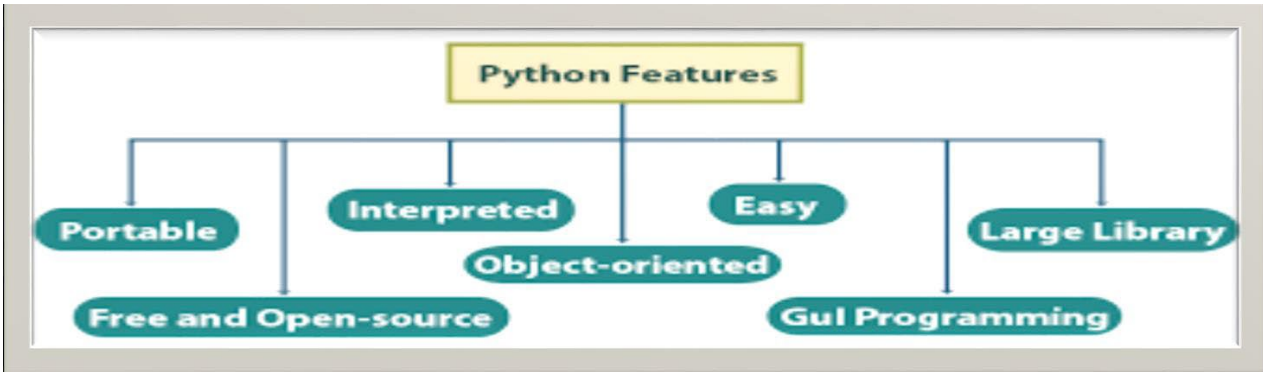
Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, Unix shell, and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features



- **EASY-TO-LEARN** – Python has few keywords, a simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **EASY-TO-READ** – Python code is more clearly defined and visible to the eyes.
- **EASY-TO-MAINTAIN** – Python's source code is fairly easy-to-maintain.

- **A BROAD STANDARD LIBRARY** – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **INTERACTIVE MODE** – Python has support for an interactive mode that allows interactive testing and debugging of snippets of code.
- **PORTABLE** – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **EXTENDABLE** – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **DATABASES** – Python provides interfaces to all major commercial databases.
- **GUI PROGRAMMING** – Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **SCALABLE** – Python provides a better structure and support for large programs than shell scripting.

MySQL Overview

What is MySQL?

SQL is the core of a relational database which is used for accessing and managing the database. By using SQL, you can add, update, or delete rows of data, retrieve subsets of information, modify databases, and perform many actions.

The different subsets of SQL are as follows:

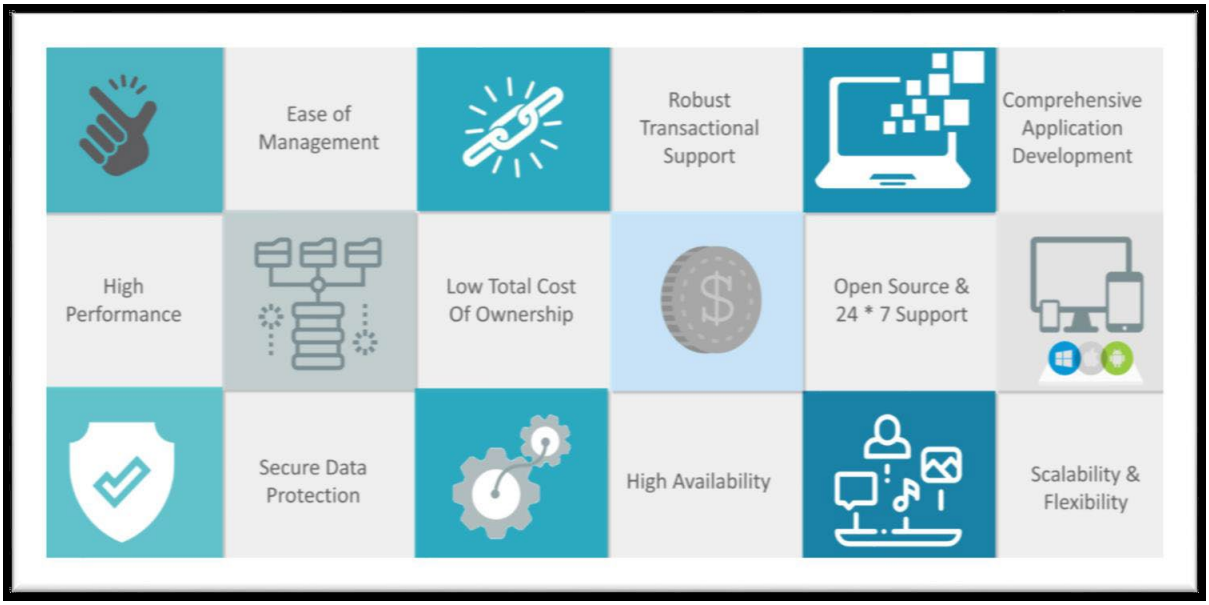
DDL (Data Definition Language) – It allows you to perform various operations on the database such as CREATE, ALTER, and DELETE objects.

DML (Data Manipulation Language) – It allows you to access and manipulate data. It helps you to insert, update, delete and retrieve data from the database.

DCL (Data Control Language) – It allows you to control access to the database. Example – Grant or Revoke access permissions.

TCL (Transaction Control Language) – It allows you to deal with the transaction of the database. Example – Commit, Rollback, Savepoint, Set Transaction.

Features of MySQL:



Some features and advantages of MySQL are:

- Robust Transactional Support
- Comprehensive Application Development
- Ease of Management
- High Performance
- Open Source & 24 * 7 Support
- Secure Data Protection
- High Availability
- Scalability & Flexibility

System design & development

Database design:

An important aspect of system design is the design of data storage structure. to begin with, a logical model of data structure is developed first.

A database is a container object which contains tables queries reports and data validation policies enforcement rules or constraints etc. the goodness of database design lies in a table structure and its relationship.

This software projects maintain a database named "crickters_data" which contains a table named "Player_data".

Table design:

The data basis of this system contains many tables of CSV also disable is normalized to minimize the residency of data in enforcing the validation rules of the organization the table is designed to store master record.

MYSQL TABLE

MySQL table description:~

```
mysql> desc player_data;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| PLAYER   | char(30)    | YES  |     | NULL    |       |
| TEAM     | char(5)     | YES  |     | NULL    |       |
| ROLL     | varchar(10) | YES  |     | NULL    |       |
| MATCHES  | int         | YES  |     | NULL    |       |
| RUNSWKTS | int         | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
5 rows in set (0.02 sec)
```

MySQL table:~

```
mysql> select * from player_data;
+----------------+------+---------+---------+----------+
| PLAYER         | TEAM | ROLL    | MATCHES | RUNSWKTS |
+----------------+------+---------+---------+----------+
| Virat Kohli    | RCB  | Batsman |    2076 |      283 |
| Shikhar Dhawan | DC   | Batsman |    1925 |      784 |
| Rohit Sharma   | MI   | Batsman |    2135 |      611 |
| Suresh Raina   | CSK  | Batsman |    2055 |      528 |
| David Warner   | SRH  | Batsman |     150 |     5499 |
| Lasith Malinga | Mi   | Bowler  |     120 |      177 |
| Dwayne Bravo   | CSK  | Bowler  |     151 |      167 |
| Dwayne Bravo   | CSK  | Bowler  |     151 |      167 |
| Amit Mishra    | DC   | Bowler  |     154 |      166 |
| Ms dhoni       | CSK  | Batsman |     200 |     4432 |
| Dwayne Bravo   | CSK  | Bowler  |     151 |      167 |
| Anshu          | MI   | Batsman |     254 |     2587 |
+----------------+------+---------+---------+----------+
12 rows in set (0.02 sec)
```

CSV TABLES

IPL Teams:-

	A	B	C	D	E	F	G	H
1	Teams	Abbreviation	Matches	Won	Lost	No result	Win %	Titles
2	Chennai Super Kings	CSK	195	117	76	1	60.56	4
3	Mumbai Indians	MI	217	125	88	0	58.6	5
4	Delhi Capitals	DC	210	93	111	2	45.67	0
5	Kolkata Knight Riders	KKR	209	107	98	0	52.15	2
6	Punjab Kings	PBKS	204	91	109	0	45.58	0
7	Rajasthan Royals	RR	175	84	86	2	49.42	1
8	Royale Challenge Bangalore	RCB	211	98	106	4	48.06	0
9	Sunrisers Hyderabad	SRH	138	68	66	0	50.72	1
10	Deccan Chargers	DC	75	29	46	0	38.66	1
11	Kochi Tuskers Kerala	KTK	14	6	8	0	42.85	0
12	Pune Warriors India	PWI	46	12	33	0	26.66	0
13	Gujarat Lions	GL	30	13	16	1	45	0
14	Rising Pune Suparjiants	RPS	30	15	15	0	50	0

Individual record:-

	A	B	C	D	E	F	G	H
1	Player	Matches	Runs	Average	Strike Rate	100s	50s	Highest
2	Virat Kohli	207	6,283	37.39	129.94	5	42	113
3	Shikhar Dhawan	192	5,784	34.84	126.64	2	44	106
4	Rohit Sharma	213	5,611	31.17	130.39	1	40	109
5	Suresh Raina	205	5,528	32.51	136.73	1	39	100
6	MS Dhoni	220	4,746	39.55	135.83	0	23	84
7	Robin Utthappa	193	4,722	27.94	130.15	0	25	87
8	Gautam Gambhr	154	4,217	31	123.88	0	36	93
9	Dinesh Karthik	213	4,096	25.77	129.72	0	19	97
10	Ajinkya Rahane	151	3,941	31.52	121.33	2	28	105
11	Ambati Rayudu	175	3,916	29.44	127.47	1	21	100
12	Manish Pandey	154	3,560	30.68	121.83	1	21	114
13	Lokesh Rahul	94	3,273	47.43	136.37	2	27	132
14	Yusuf Pathan	174	3,204	29.12	142.97	1	13	100
15	Sanju Samson	121	3,068	29.21	134.2	3	15	119
16	Parthiv Patel	139	2,848	22.6	120.78	0	13	81
17	Yuvraj Singh	132	2,750	24.77	129.71	0	13	83
18	Virender Sahwag	104	2,728	27.55	155.44	2	16	122
19	Murli Vijay	106	2,619	25.93	121.87	2	13	127
20	Rishabh Pant	84	2,498	35.18	147.46	1	15	128
21	Ravindra Jadedja	200	2,386	27.11	128.14	0	2	62
22	Shreys Iyer	87	2,375	31.66	123.95	0	16	96
23	Surya kumar Yadav	115	2,341	28.9	135.71	0	13	82
24	Sachin Tendulkar	78	2,334	34.83	119.81	1	13	100
25	Rahul Dravid	89	2,174	28.23	115.51	0	11	75
26	Mayank Agarwal	100	2,131	23.41	135.47	1	11	106

Single achievement record:-

	A	B
1	Individual Records	Player/Winner
2	Highest team total	263/5 (20) RCB v PWI
3	Lowest team total	49/10 RCB v KKR
4	most runs	Virat Kohli (6,283)
5	most wickets	Lasith Malinga (170)
6	most career fours	Shikhar Dhawan (654)
7	most career sixes	Chris Gayle (357)
8	most titles winner	Mumbai Indian (5 times)
9	Greatest win (by runs)	146 runs MI v DC
10	Highest succesful run chase	226/6 RR v PBKS
11	Lowest succesful defence	117 CSK v PBKS
12	Highest individual score	chris gayle 175* v PWI
13	Most 50s	David Warner (50)
14	Fastest 50	KL Rahul (14 ball) v DC
15	Most 100s	Chris Gayle (6)
16	Fastest 100	Chris Gayle (30 ball) v PWI
17	Most 6s in an inning	Chris Gayle (17) v PWI
18	Most 4s in an inning	Paul Valthy v CSK, AB devilliars v MI (19)
19	Highest Strike rates	Andre Russel (178.57) 1700 off 952 balls
20	Highest Strike rates in an inning	Chriss Morris (422.22) 38* off 9 balls
21	Most wkts in single edition	Dwayne Bravo and Harshal Patel (32)
22	Most runs in a single edition	Virat Kohli (973 runs in Ipl 2017)
23	Most runs in an over	Chris Gayle v KTK, R Jadeja v RCB (36)
24	Most Ducks	Piyush Chawla (13)
25	Best bowling figures	Alzarri Joseph v SRH (6/12)
26	Best career average	Adam Jampa (17.61)
27	Best career economy rate	Rashid Khan (6.33)
28	Best career strike rate (bowling)	Lungi Ngidi (12.9)
29	Most 4 wkts hauls	Sunil Naraine (8)
30	Most runs conceded in match	Basil Thampi (0/70)
31	Most career dismissals	MS Dhoni (161)
32	Most career catches	MS Dhoni (122)
33	Most career stumpings	Ms Dhni (39)
34	Most dismissals in an inning	Kumar Sangakkara (5) v RCB
35	Most dismissals in a single edition	Rishabh Pant (24) Ipl 2019
36	Most career catches (as fielder)	Suresh Raina (109)
37	Most catches in a match	Mohammad Nabi (5) v MI
38	Most successful Captain	MS Dhoni (win %age 59.61)
39	Most Consecutive wins	KKR (9) Ipl 2014
40	Most consecutive defeats	Delhi capitals and Pune Warrios (11)
41	Most Man of the Match	Ab De Villiars (23)
42	Most fairplay awards	CSK (6) 2008,2010,2011,2013,2014,2015
43	Highest partnership	V Kohli and Devilliars (229 for 2nd wkt) v GL

Most hattricks':-

	A	B
1	PLAYER	NO. of HATRICKS IN IPL
2	Amit mishra	3
3	Yuvraj Singh	2
4	Makhaya Ntini	1
5	Ajit Chandila	1
6	Samuel Badree	1
7	Sam Curran	1
8	Rohit Sharma	1
9	Andrew Tye	1
10	Pravin Tambe	1
11	Shreyas Gopal	1
12	Lakshmipathy Balaji	1
13	Jaydev Unadkat	1
14	Axar Patel	1
15	Shane Watson	1
16	Praveen Kumar	1
17	Sunil Narine	1

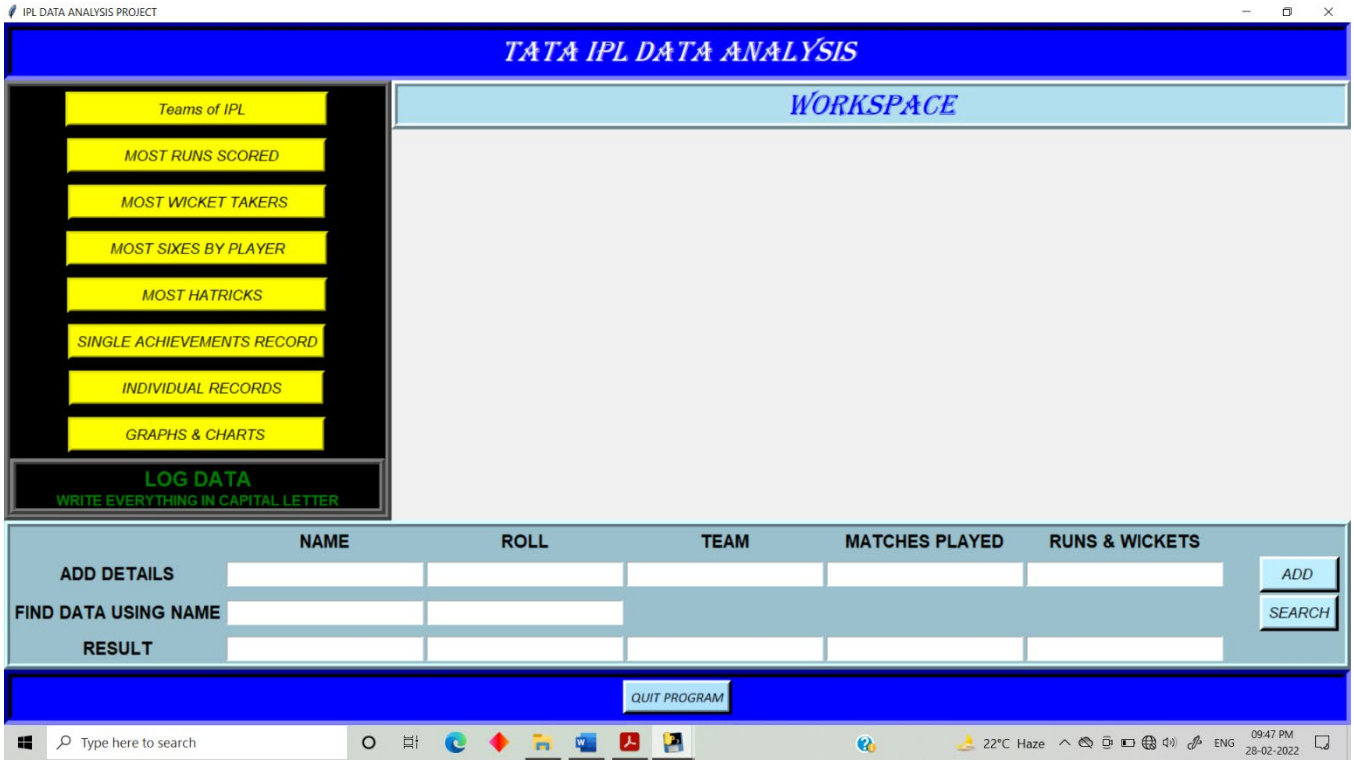
Most wicket takers:-

	A	B	C	D	E	F	G
1	Player	Matches	Inns	Overs	Mdns	Wkts	Econ.
2	Lasith Malinga	122	122	471.1	8	170	7.14
3	Dwayne Bravo	151	148	485.3	2	167	8.36
4	Amit Mishra	154	154	540.5	6	166	7.35
5	Piyush Chawla	165	164	545.4	2	157	7.88
6	Harbhajan Singh	163	160	569.2	6	150	7.07
7	Ravichandran Ashwin	167	164	583	4	145	6.91
8	Sunil Naraine	134	133	520.1	3	143	6.74
9	Bhuvneshwar Kumar	132	132	491.3	9	142	7.3
10	Yuzvendra Chahal	114	113	408	4	139	7.59
11	Jasprit Bumrah	106	106	403	6	130	7.42

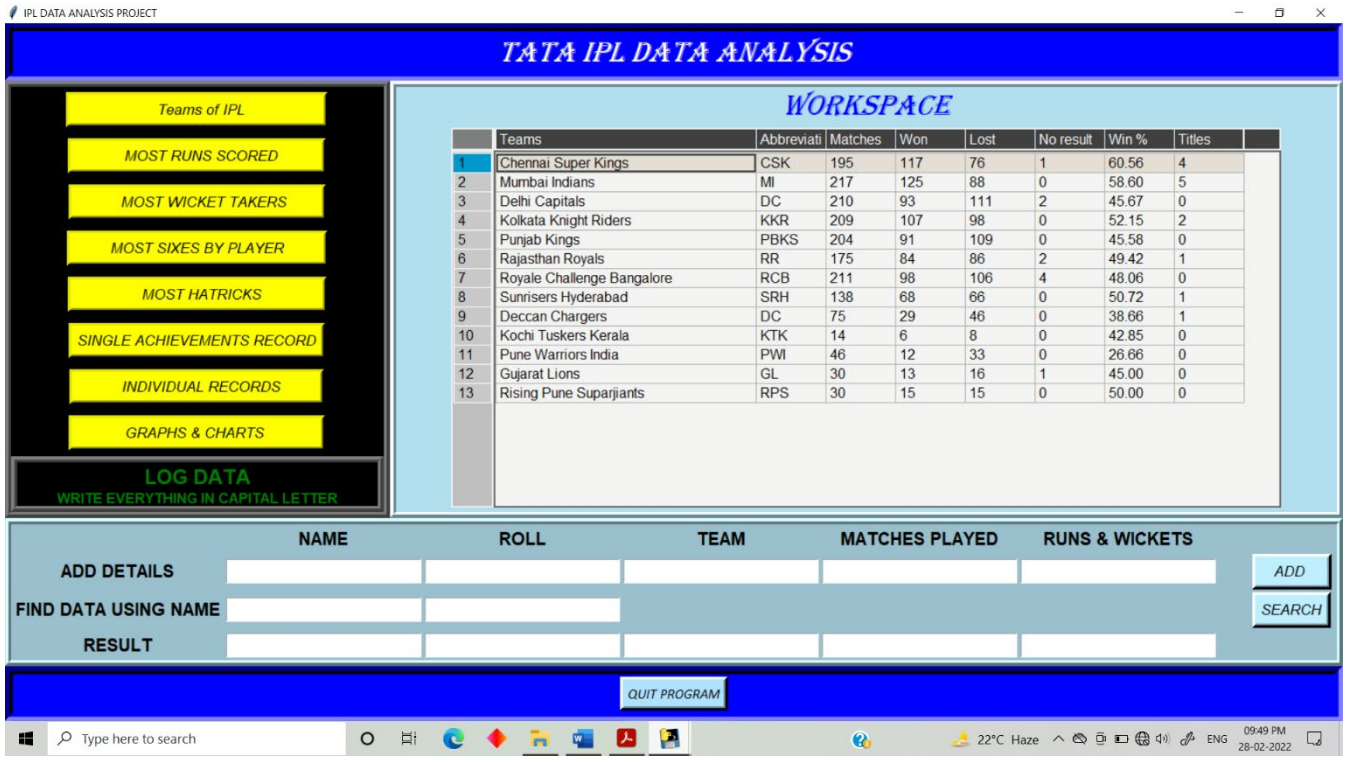
Most sixes:-

Position	PLAYER	Matches	Runs	Avg	6s
1	Chris Gayle	142	4965	39.72	357
2	AB de Villiers	182	5125	40.03	250
3	Rohit Sharma	212	5593	31.24	227
4	MS Dhoni	218	4728	39.4	218
5	Kieron Pollard	177	3255	30.13	214
6	Virat Kohli	205	6240	37.59	210
7	Suresh Raina	205	5528	32.51	203
8	David Warner	150	5449	41.59	201
9	Shane Watson	145	3874	30.99	190
10	Robin Uthappa	191	4628	27.71	163
11	Yusuf Pathan	174	3204	29.12	158
12	Yuvraj Singh	132	2750	24.77	149
13	Ambati Rayudu	173	3915	29.65	149
14	Andre Russell	84	1700	29.31	143
15	KL Rahul	94	3273	47.43	134
16	Sanju Samson	121	3068	29.21	132
17	Brendon McCullum	109	2880	27.69	130
18	Shikhar Dhawan	189	5698	34.95	120
19	Dwayne Smith	91	2385	28.39	117
20	Glenn Maxwell	95	1952	24.7	112
21	Rishabh Pant	81	2431	35.23	111
22	Dinesh Karthik	210	4027	26.14	111
23	Virender Sehwag	104	2728	27.55	106
24	Manish Pandey	153	3491	30.09	101
25	Hardik Pandya	91	1466	27.66	97
26	Faf du Plessis	98	2848	34.73	93
27	Adam Gilchrist	80	2069	27.22	92
28	Murali Vijay	106	2619	25.93	91
29	Jos Buttler	65	1968	35.14	90
30	David Miller	89	1974	32.9	90
31	Nitish Rana	74	1784	29.24	87
32	Shreyas Iyer	84	2326	31.86	87
33	Mayank Agarwal	100	2131	23.41	85
34	Ravindra Jadeja	198	2386	27.11	85
35	Quinton de Kock	77	2256	31.33	83
36	JP Duminy	83	2029	39.78	79
37	Naman Ojha	113	1554	20.72	79
38	Shaun Marsh	71	2477	39.95	78
39	Ajinkya Rahane	151	3941	31.52	76
40	Aaron Finch	87	2005	25.7	75
41	Ishan Kishan	60	1368	27.36	70
42	Wriddhiman Saha	132	2108	24.8	69
43	Chris Lynn	42	1329	34.07	66
44	Suryakumar Yadav	114	2259	28.23	65
45	Dwayne Bravo	149	1537	22.94	65
46	Eoin Morgan	80	1396	23.26	64
47	David Hussey	64	1322	26.97	60
48	Steve Smith	103	2485	34.51	60

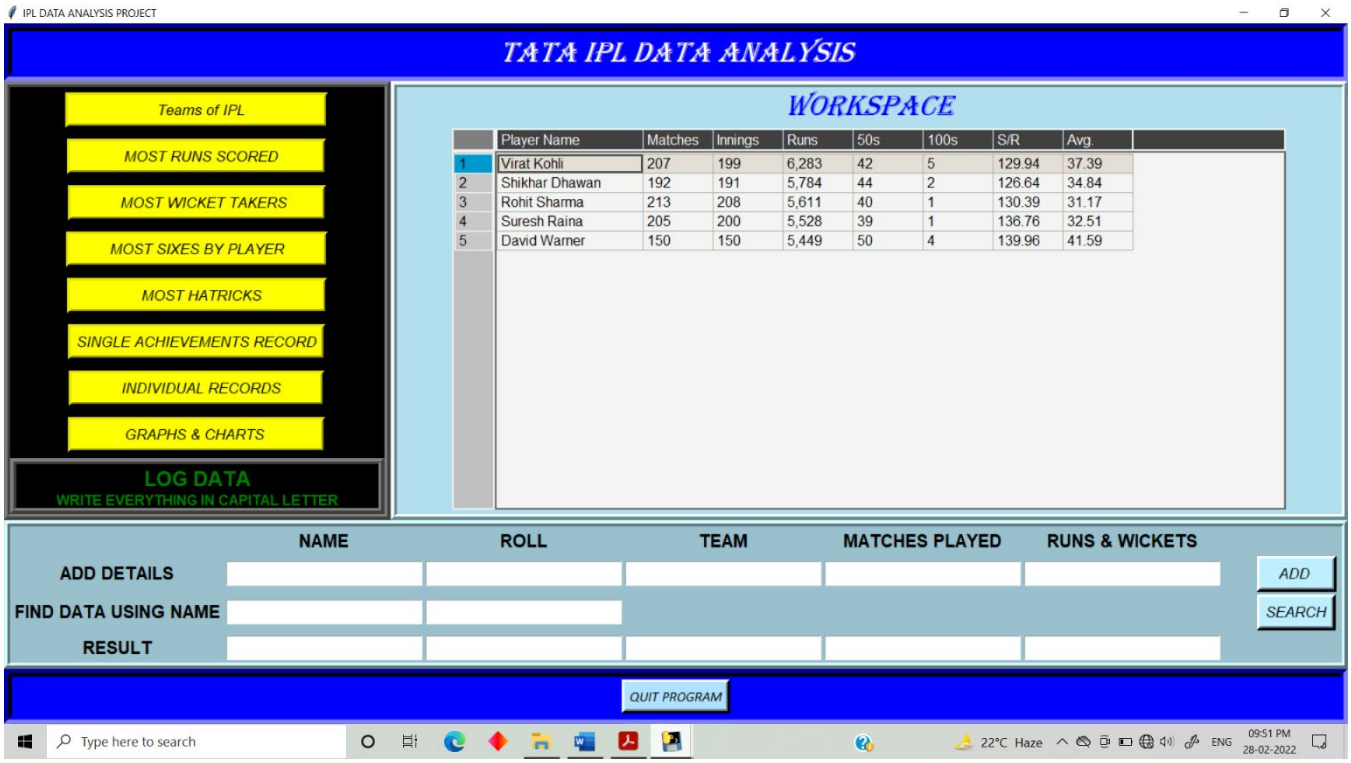
I/O Design & Event Coding



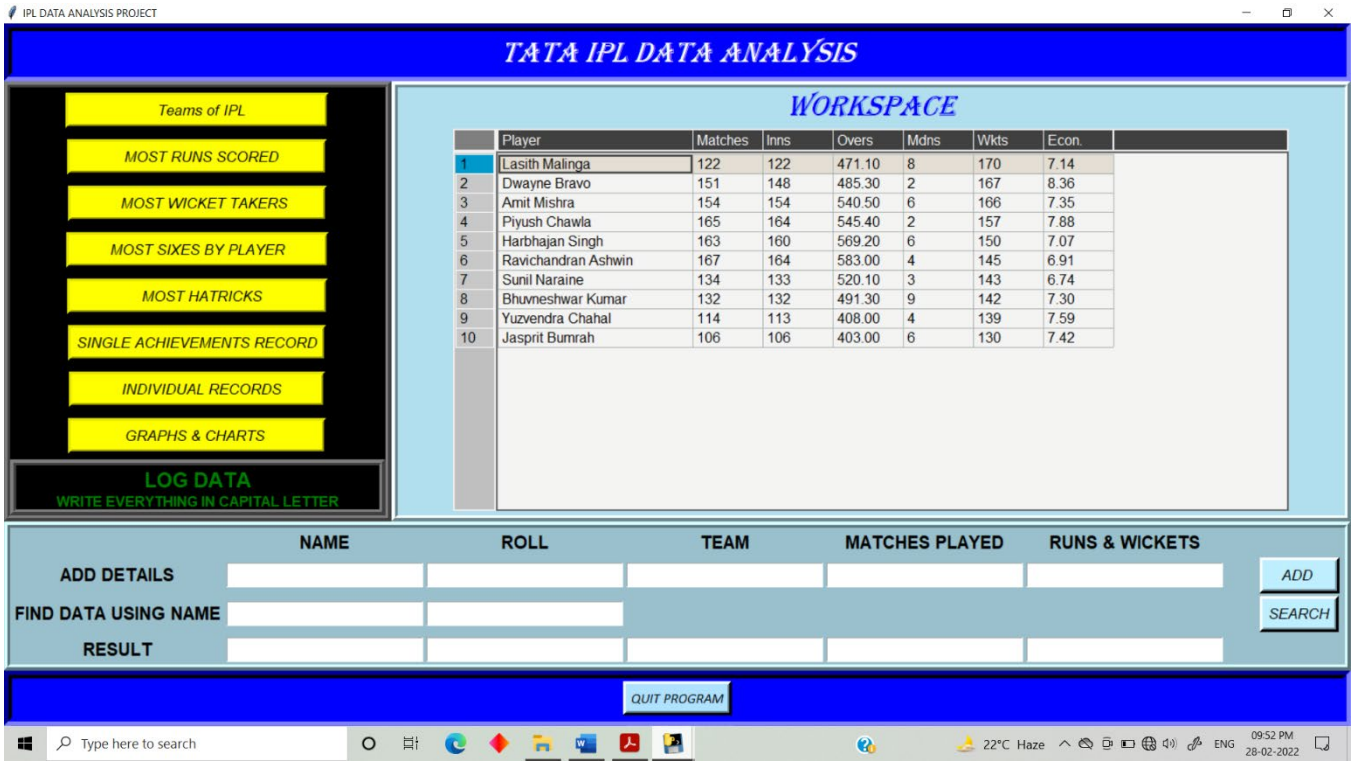
1. Teams of IPL:~



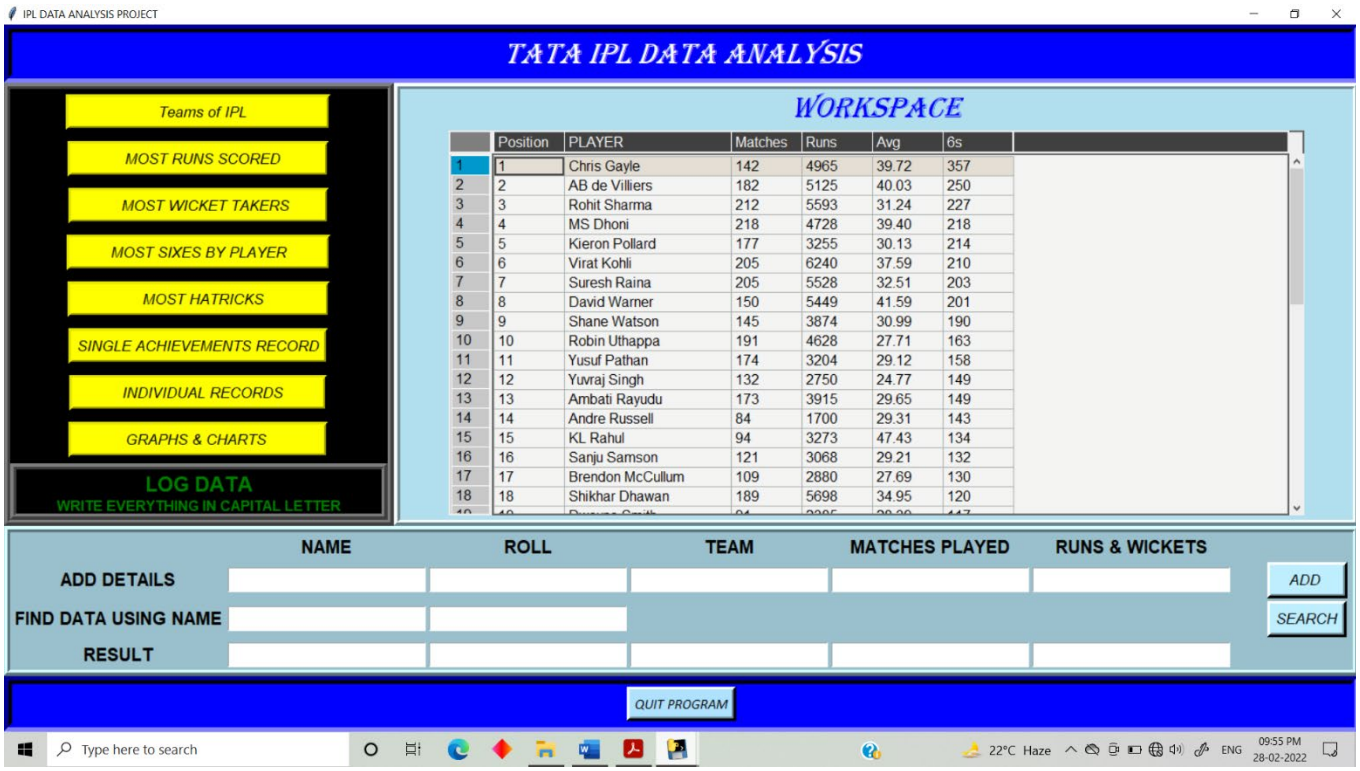
2. Most Runs Scored:-



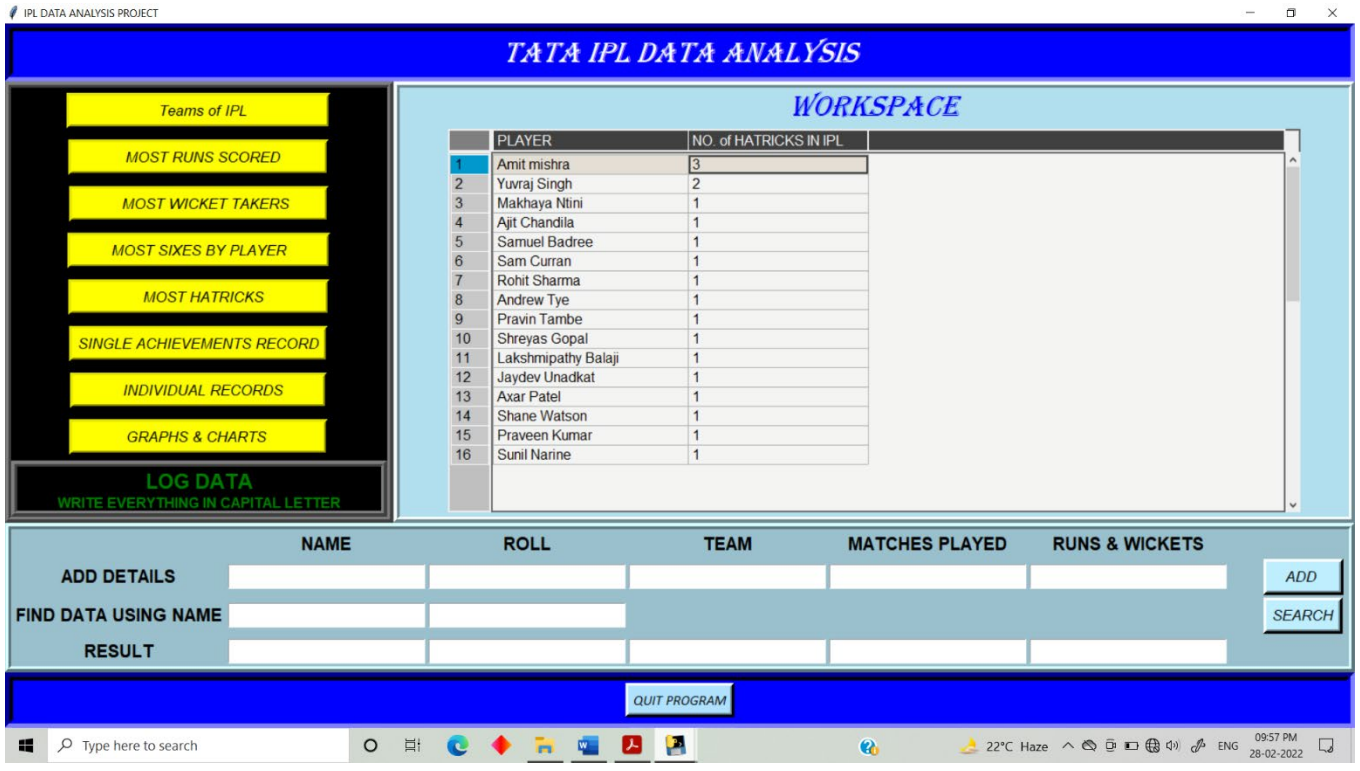
3. Most Wicket Takers:-



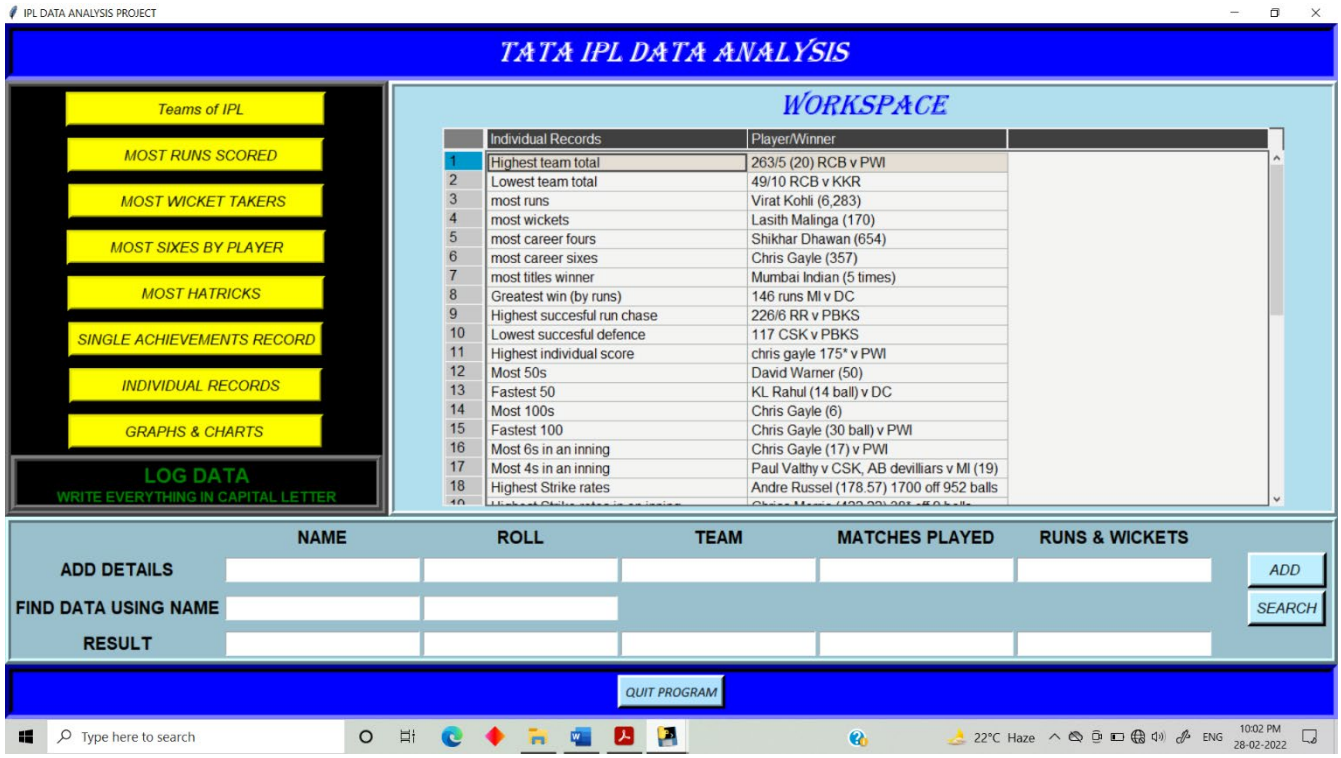
4. Most Sixes:-



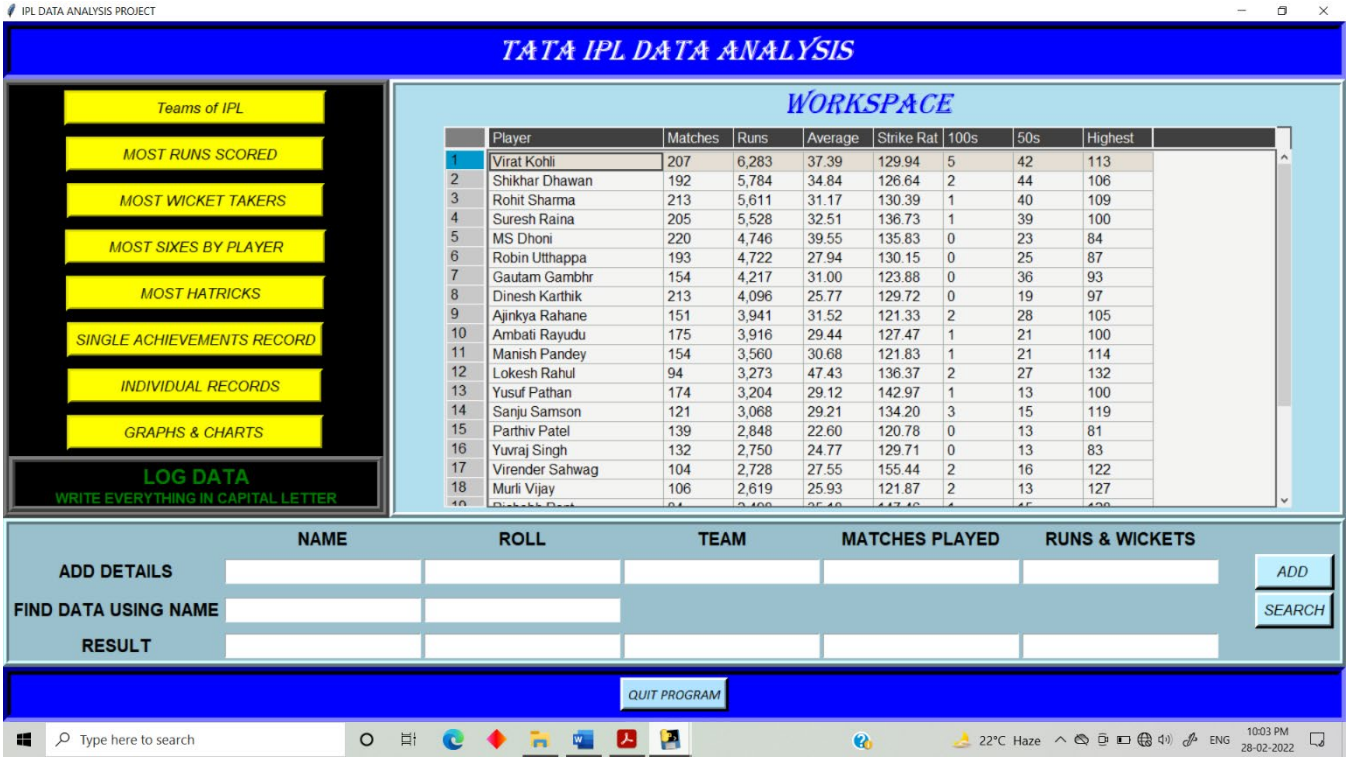
5. Most Hattrick's:-



6. Single Achievement Record:-

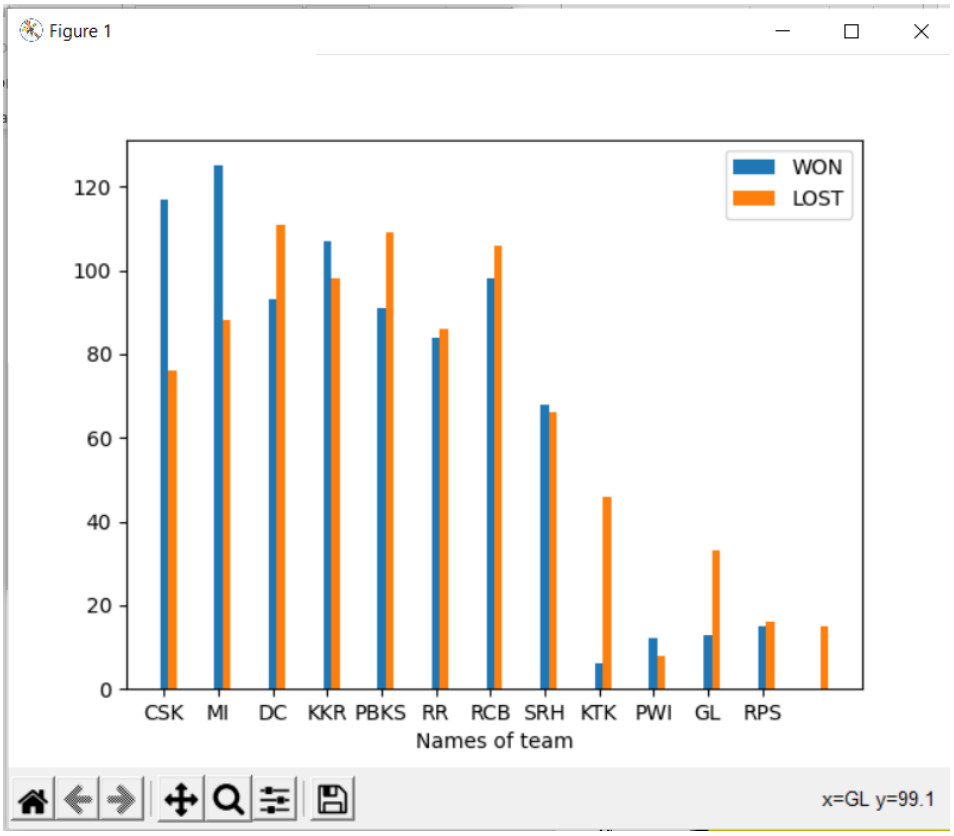


7. Individual Record:-

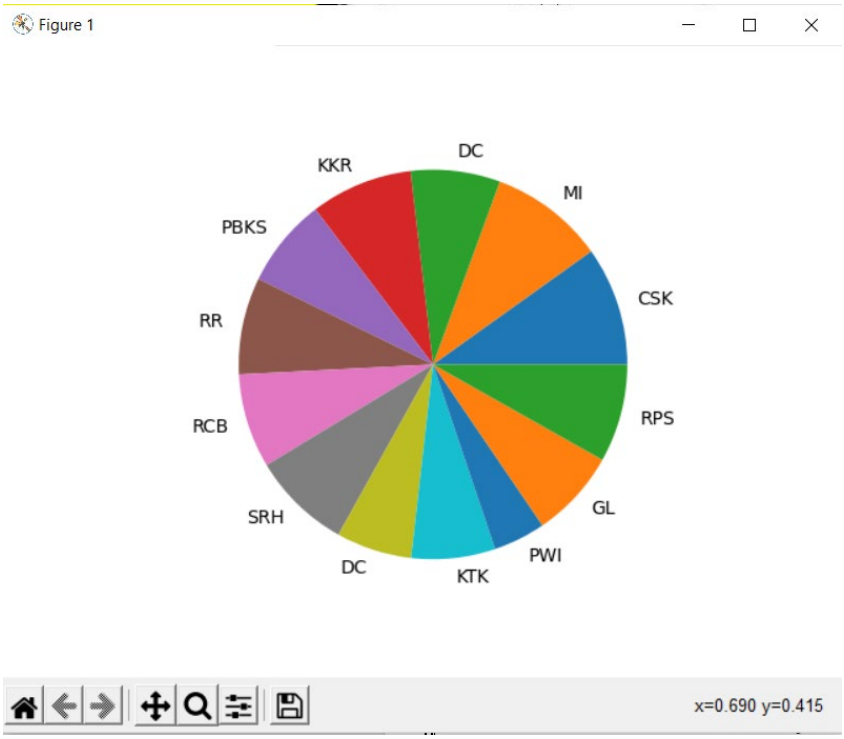


8. Graphs and Charts:-

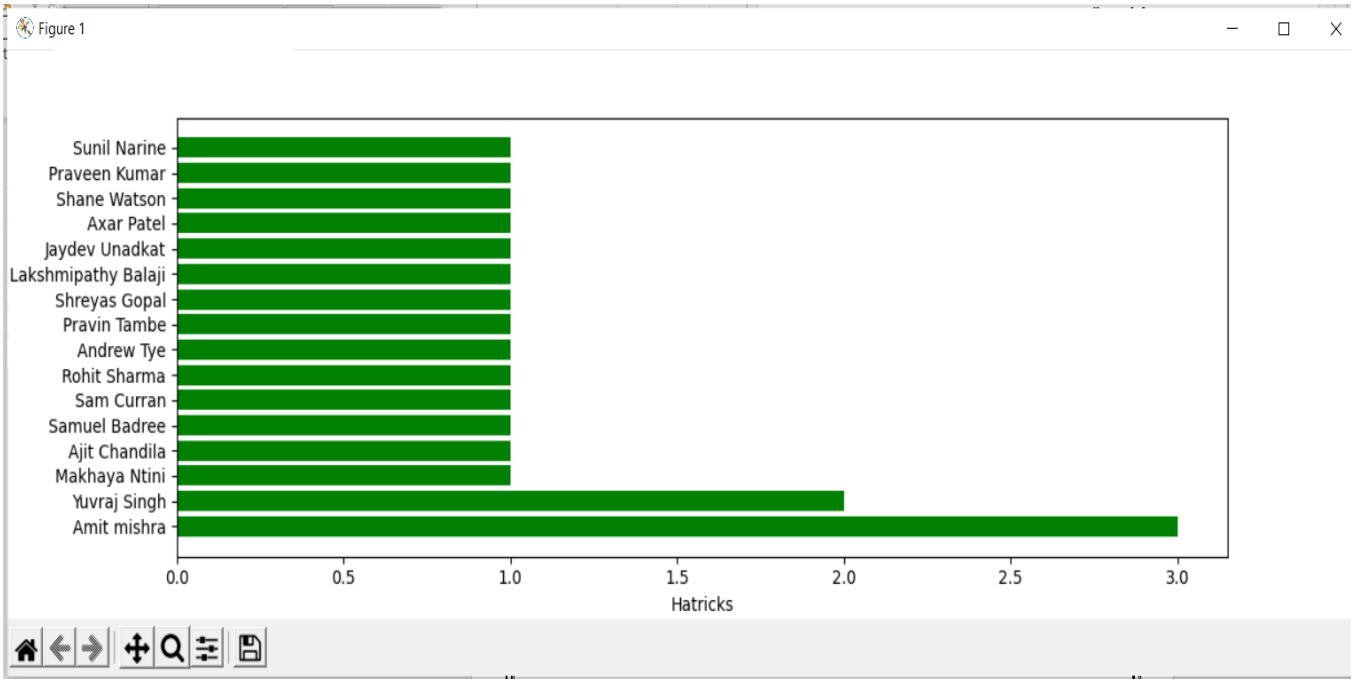
➢ Matches Won Lost



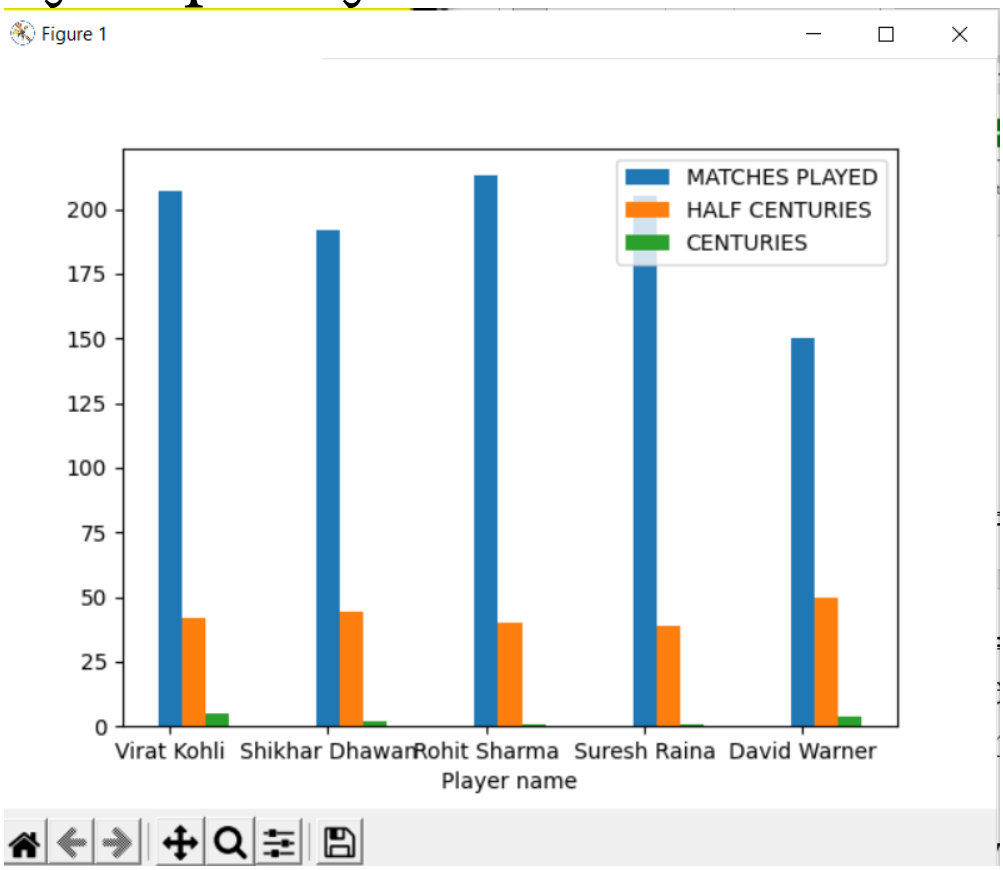
➢ Win% of Teams



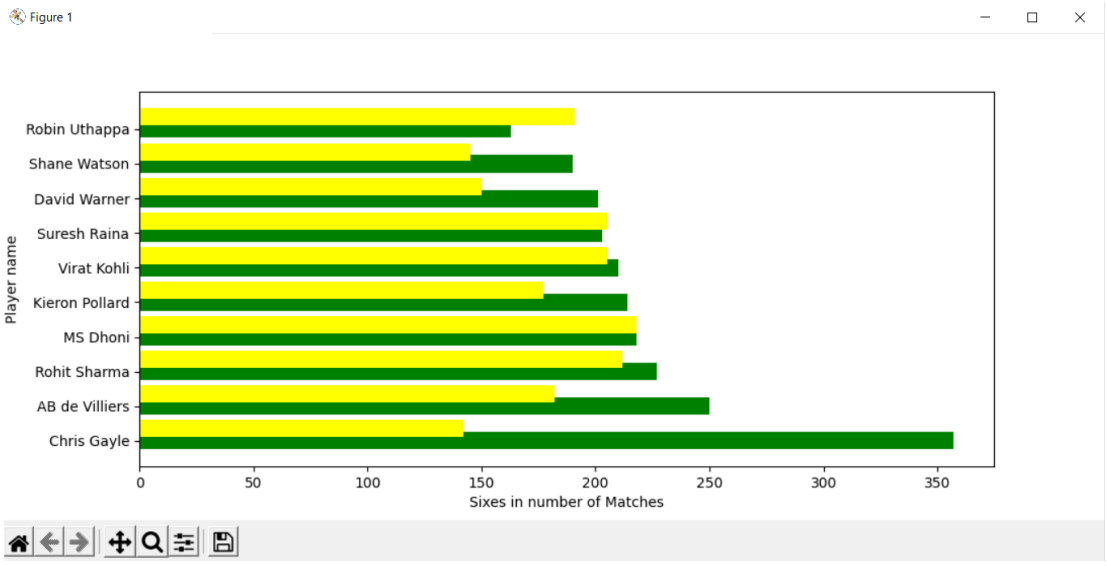
➢ Hattrick by Top Player



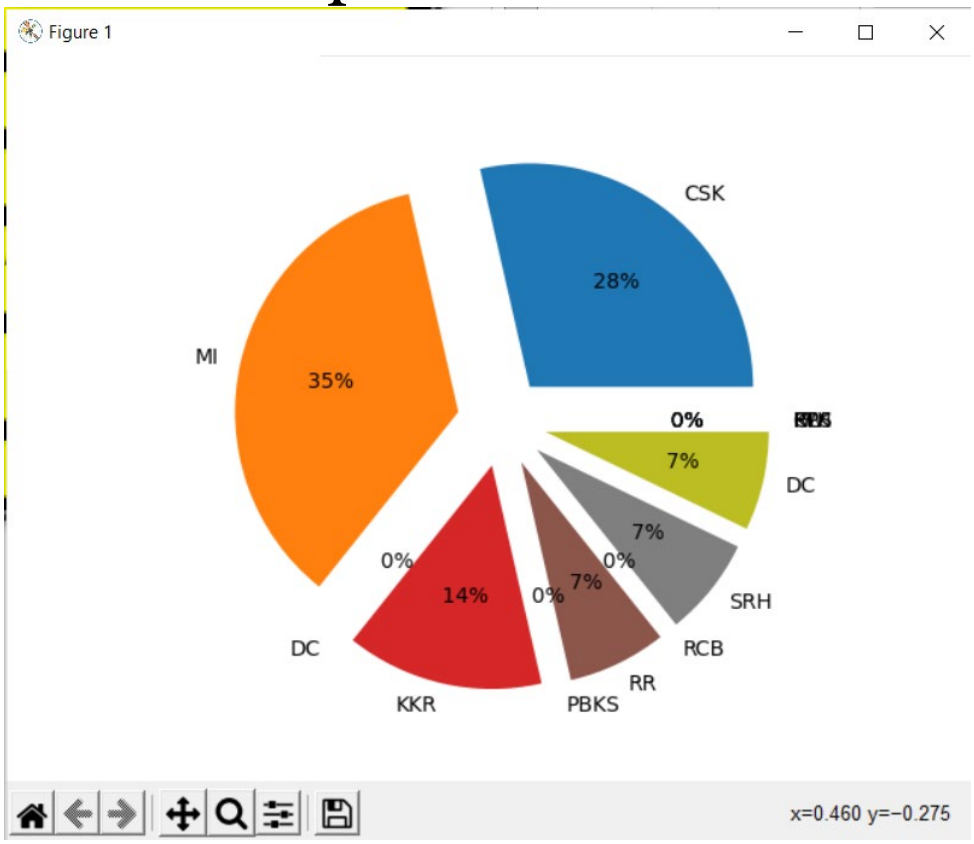
➢ Half Centuries and Centuries by Top Player



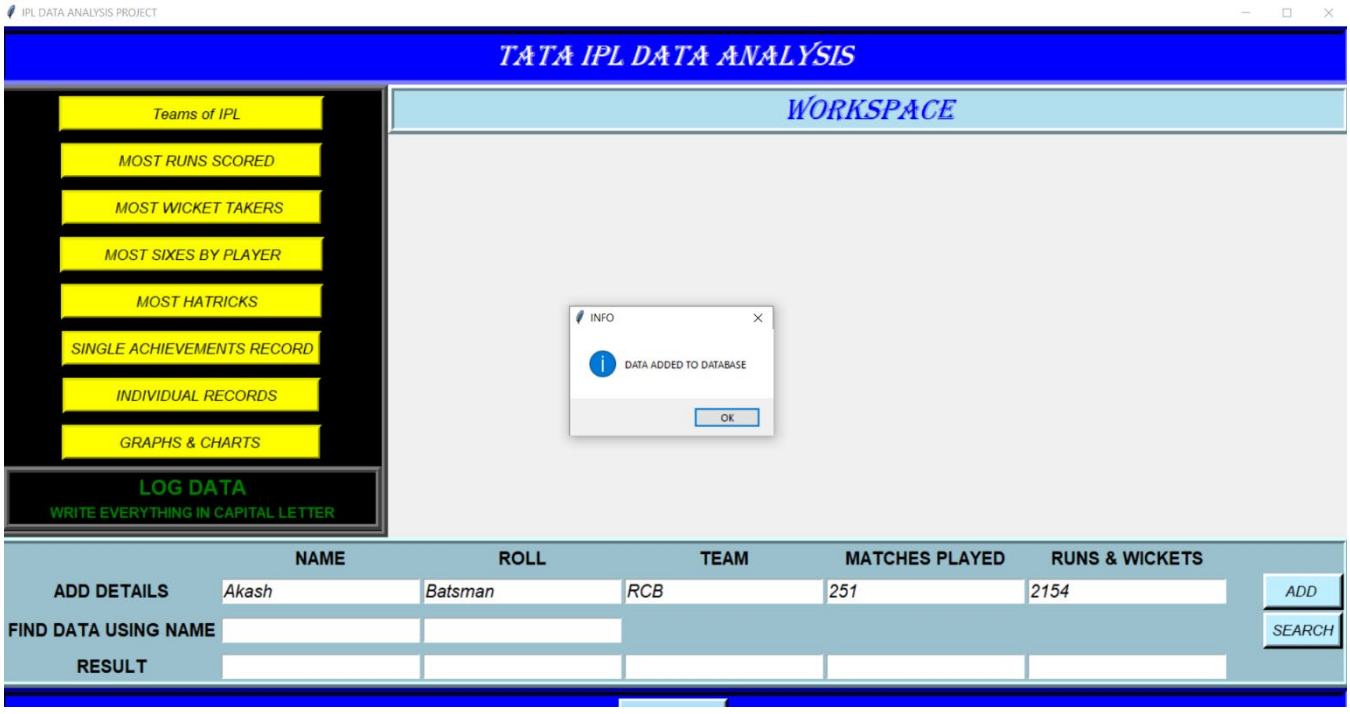
➢ Most sixes by top players



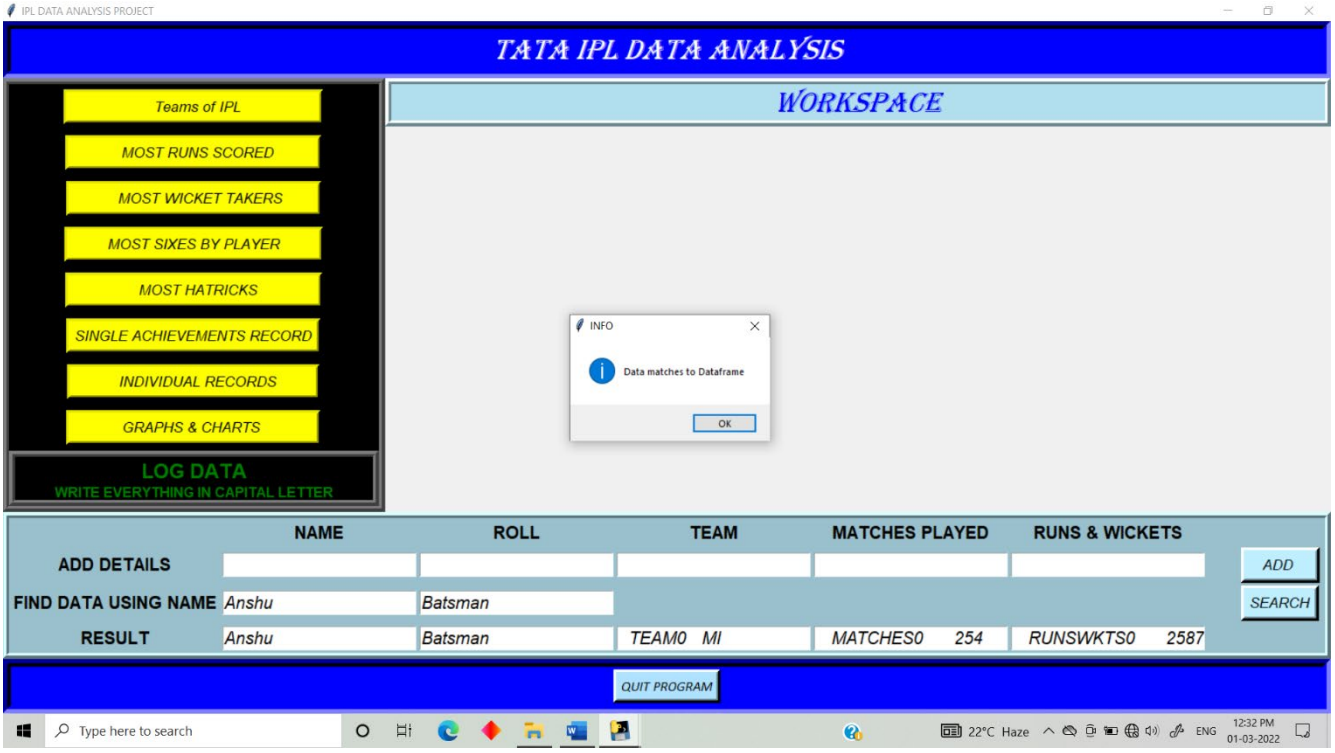
➢ The champion Team



9. ADD DATA:~



10. SEARCH DATA:~



PROGRAM CODE

```
from tkinter import *
from tkinter import messagebox
import pandas as pd
from pandastable import Table
import matplotlib.pyplot as plt
import numpy as np
import mysql.connector as sql

'''PANDAS SECTION'''

teams_of_ipl = pd.read_csv("Teams of ipl.csv")
most_runs = pd.read_csv("Runs maker.csv")
wicket_takers = pd.read_csv("Wicket takers.csv")
most_sixes = pd.read_csv("Most sixes.csv")
most_hatricks = pd.read_csv("Most hatricks.csv")
single_acheivement = pd.read_csv("Single achivement.csv")
individual_record = pd.read_csv("Individual record.csv")

'''GUI SECTION'''

gui = Tk()
gui.title('IPL DATA ANALYSIS PROJECT')
gui.geometry('500x500') # width x height
gui.minsize(1530, 810) # width , height
gui.maxsize(2000, 1500) # width , height

# Heading frame

frame_head = Frame(gui, bg="blue", borderwidth=9,
relief=SUNKEN)
frame_head.pack(fill="x")

label_head = Label(frame_head, text="TATA IPL DATA
ANALYSIS", font="Algerian 25 italic", bg="blue", fg="white")
label_head.pack(pady=3)
```

```
# Footer frame

frame_foot = Frame(gui, bg="blue", borderwidth=9,
relief=SUNKEN)
frame_foot.pack(side='bottom', fill="x")

quit_button = Button(frame_foot, text='QUIT PROGRAM',
command=gui.destroy, bg="lightskyblue1", fg="black",
                font="Calibri 12 italic", borderwidth=5,
relief=RAISED)
quit_button.pack(pady=3)

# Bottom frame

frame_bottom = Frame(gui, bg="lightblue3", borderwidth=7,
relief=RIDGE)
frame_bottom.pack(side='bottom', fill="y")

# Left side UP frame

frame_left_up = Frame(gui, bg="black", borderwidth=7,
relief=RIDGE)
frame_left_up.pack(fill=Y, side=LEFT)

# Workspace frame

frame_workspace = Frame(gui, bg="lightblue2",
borderwidth=7, relief=RIDGE)
canvas_b_workspace = Canvas(frame_workspace,
width=1125, height=480, scrollregion=(0, 0, 500, 500))

frame_workspace.pack(anchor=NE, fill="both")

work_label = Label(frame_workspace, text="Workspace",
bg="lightblue2", fg="blue", font="Algerian 25 italic")
work_label.pack()
```

```
# bottom frame labels textbox and buttons

label1 = Label(frame_bottom, text="ADD DETAILS",
bg="lightblue3", font="Aparajita 15 bold")
label1.grid(row=1, column=1, pady=1.5)

label2 = Label(frame_bottom, text="FIND DATA USING
NAME", bg="lightblue3", font="Aparajita 15 bold")
label2.grid(row=2, column=1, pady=1.5, padx=2)

label3 = Label(frame_bottom, text="RESULT",
bg="lightblue3", font="Aparajita 15 bold")
label3.grid(row=3, column=1, pady=2)

label4 = Label(frame_bottom, text="NAME", bg="lightblue3",
font="Aparajita 15 bold")
label4.grid(row=0, column=2, pady=1.5)

label5 = Label(frame_bottom, text="ROLL", bg="lightblue3",
font="Aparajita 15 bold")
label5.grid(row=0, column=3, pady=1.5)

label6 = Label(frame_bottom, text="TEAM", bg="lightblue3",
font="Aparajita 15 bold")
label6.grid(row=0, column=4, pady=1.5)

label7 = Label(frame_bottom, text="MATCHES PLAYED",
bg="lightblue3", font="Aparajita 15 bold")
label7.grid(row=0, column=5, pady=1.5)

label8 = Label(frame_bottom, text="RUNS & WICKETS",
bg="lightblue3", font="Aparajita 15 bold")
label8.grid(row=0, column=6, pady=1.5)
```

```
'''ADD details'''
name_entry = Entry(frame_bottom,
textvariable=StringVar, font="Abadi 15 italic")
roll_entry = Entry(frame_bottom, textvariable=StringVar,
font="Abadi 15 italic")
team_entry = Entry(frame_bottom, textvariable=StringVar,
font="Abadi 15 italic")
matches_played_entry = Entry(frame_bottom,
textvariable=IntVar, font="Abadi 15 italic")
runs_wicket_entry = Entry(frame_bottom,
textvariable=IntVar, font="Abadi 15 italic")

name_entry.grid(row=1, column=2, padx=2)
roll_entry.grid(row=1, column=3, padx=2)
team_entry.grid(row=1, column=4, padx=2)
matches_played_entry.grid(row=1, column=5, padx=2)
runs_wicket_entry.grid(row=1, column=6, padx=2)

'''Find data'''
name_entry2 = Entry(frame_bottom,
textvariable=StringVar, font="Abadi 15 italic")
roll_entry2 = Entry(frame_bottom, textvariable=StringVar,
font="Abadi 15 italic")

name_entry2.grid(row=2, column=2, padx=2)
roll_entry2.grid(row=2, column=3, padx=2)

'''RESULT'''
name_entry3 = Entry(frame_bottom,
textvariable=StringVar, font="Abadi 15 italic")
roll_entry3 = Entry(frame_bottom, textvariable=StringVar,
font="Abadi 15 italic")
team_entry3 = Entry(frame_bottom,
textvariable=StringVar, font="Abadi 15 italic")
matches_played_entry3 = Entry(frame_bottom,
textvariable=IntVar, font="Abadi 15 italic")
runs_wicket_entry3 = Entry(frame_bottom,
textvariable=IntVar, font="Abadi 15 italic")
```

```
name_entry3.grid(row=3, column=2, padx=2, pady=2.5)
roll_entry3.grid(row=3, column=3, padx=2, pady=2.5)
team_entry3.grid(row=3, column=4, padx=2, pady=2.5)
matches_played_entry3.grid(row=3, column=5, padx=2,
pady=2.5)
runs_wicket_entry3.grid(row=3, column=6, padx=2,
pady=2.5)

# Function
# Function used in bottom frame
def add():
    mysql_connection = sql.connect(host='localhost',
    user='root', passwd='123456789',
    database="crickters_data")

    player = name_entry.get()
    roll = roll_entry.get()
    team = team_entry.get()
    matches = matches_played_entry.get()
    runs_wicket = runs_wicket_entry.get()

    cur = mysql_connection.cursor()
    query = f"insert into player_data
    values('{player}','{team}','{roll}',{matches},{runs_wicket})
    ;"
    cur.execute(query)
    mysql_connection.commit()
    mysql_connection.close()

    messagebox.showinfo('INFO', 'DATA ADDED TO
    DATABASE')

    name_entry.delete(0, END)
    roll_entry.delete(0, END)
    team_entry.delete(0, END)
    matches_played_entry.delete(0, END)
    runs_wicket_entry.delete(0, END)

    mysql_connection.close()
```

```
def search():

    # clear all result column
    name_entry3.delete(0, END)
    roll_entry3.delete(0, END)
    team_entry3.delete(0, END)
    matches_played_entry3.delete(0, END)
    runs_wicket_entry.delete(0, END)
    mysql_connection = sql.connect(host='localhost',
    user='root', passwd='123456789',
    database="crickters_data")

    # getting enter data
    name = name_entry2.get()
    roll = roll_entry2.get()

    # sql query
    df = pd.read_sql(f"select * from Player_data where
    PLAYER='{name}' and ROLL='{roll}';", mysql_connection)

    # inserting data from database
    Team = df.iloc[:, 1:2]
    Matches = df.iloc[:, 3:4]
    R_W = df.iloc[:, 4:]

    name_entry3.insert(0, name)
    roll_entry3.insert(0, roll)
    team_entry3.insert(0, Team)
    matches_played_entry3.insert(0, Matches)
    runs_wicket_entry3.insert(0, R_W)

    messagebox.showinfo('INFO', 'Data matches to
    Dataframe')


    mysql_connection.close()
```

```
# Functions used for graph
def g1():
    team = list(teams_of_ipl['Abbreviation'])
    won = list(teams_of_ipl['Won'])
    lost = list(teams_of_ipl['Lost'])

    plt.bar(team, won, 0.15, label='WON')
    plt.bar(np.arange(len(team)) + 0.15, lost, 0.15,
    label='LOST')

    plt.xlabel('Names of team')
    plt.legend()

    plt.show()

def g2():
    team = list(teams_of_ipl['Abbreviation'])
    won = list(teams_of_ipl['Win %'])

    plt.pie(won, labels=team)

    plt.show()

def g3():
    player = list(most_runs['Player Name'])
    matches = list(most_runs['Matches'])
    hc = list(most_runs['50s'])
    c = list(most_runs['100s'])

    plt.bar(player, matches, 0.15, label='MATCHES PLAYED')
    plt.bar(np.arange(len(player)) + 0.15, hc, 0.15,
    label='HALF CENTURIES')
    plt.bar(np.arange(len(player)) + 0.30, c, 0.15,
    label='CENTURIES')

    plt.legend()
    plt.xlabel('Player name')

    plt.show()
```

```
def g4():
    player = list(most_hatricks['PLAYER'])
    hatricks = list(most_hatricks['NO. of HATRICKS IN IPL'])

    plt.barh(player, hatricks, color='green')

    plt.ylabel('Player name')
    plt.xlabel('Hatricks')

    plt.show()

def g5():
    top10 = most_sixes.head(10)

    player = list(top10['PLAYER'])
    sixes = list(top10['6s'])
    matches = list(top10['Matches'])

    plt.barh(player, sixes, 0.5, color='green')
    plt.barh(np.arange(len(player)) + 0.35, matches, 0.5,
    color='yellow')

    plt.ylabel('Player name')
    plt.xlabel('Sixes in number of Matches')

    plt.show()

def g6():
    team = list(teams_of_ipl['Abbreviation'])
    titles = list(teams_of_ipl['Titles'])
     exp = [0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
     0.2]

    plt.pie(titles, labels=team, explode=exp,
    autopct='%3d%%')

    plt.show()
```

```
# Function used for creating table structure
def table(df):
    table_df = Table(canvas_b_workspace, dataframe=df,
    width=900, height=430)
    table_df.show()

# Functions used in buttons
def B1_workspace():

    canvas_b_workspace.pack(pady=5)

    table(teams_of_ipl)

def B2_workspace():

    canvas_b_workspace.pack(pady=5)

    table(most_runs)

def B3_workspace():

    canvas_b_workspace.pack(pady=5)

    table(wicket_takers)

def B4_workspace():

    canvas_b_workspace.pack(pady=5)

    table(most_sixes)

def B5_workspace():

    canvas_b_workspace.pack(pady=5)

    table(most_hatricks)
```

```
def B6_workspace():

    canvas_b_workspace.pack(pady=5)

    table(single_acheivement)

def B7_workspace():

    canvas_b_workspace.pack(pady=5)

    table(individual_record)

def BG_charts():
    graph = Tk()
    graph.title('CHARTS AND GRAPHS')
    graph.geometry('340x360')
    graph.maxsize(340, 360)

    frame_graph = Frame(graph, bg="black", borderwidth=7,
    relief=RIDGE)
    frame_graph.pack(fill=X)

    b1 = Button(frame_graph, text="            Matches Won
Lost           ", font="Cavolini 13 italic",
            borderwidth=5, relief=RAISED, bg="yellow",
fg="green", command=g1)
    b1.pack(pady=5)

    b2 = Button(frame_graph, text="           WIN% OF
TEAMS              ", font="Cavolini 13 italic",
            borderwidth=5, relief=RAISED, bg="yellow",
fg="green", command=g2)
    b2.pack(pady=5)

    b4 = Button(frame_graph, text="     HATRICKS BY TOP
PLAYER      ", font="Cavolini 13 italic",
            borderwidth=5, relief=RAISED, bg="yellow",
fg="green", command=g4)
    b4.pack(pady=5)
```

```
    b3 = Button(frame_graph, text="HALF CENTURIES AND
CENTURIES\nBY TOP PLAYER", font="Cavolini 13 italic",
                borderwidth=5, relief=RAISED, bg="yellow",
fg="green", command=g3)
    b3.pack(pady=5)

    b5 = Button(frame_graph, text="        MOST SIXES BY
PLAYER        \nIN MATCHES", font="Cavolini 13 italic",
                borderwidth=5, relief=RAISED, bg="yellow",
fg="green", command=g5)
    b5.pack(pady=5)

    b6 = Button(frame_graph, text="          THE CHAMPION
TEAM          ", font="Cavolini 13 italic",
                borderwidth=5, relief=RAISED, bg="yellow",
fg="green", command=g6)
    b6.pack(pady=5)

    graph.mainloop()

# Buttons of ADD and SEARCH

button_add = Button(frame_bottom, text="   ADD    ",
font="Cavolini 13 italic", borderwidth=5, relief=RAISED,
                    bg="lightblue1", command=add)
button_add.grid(row=1, column=7, padx=40)

button_search = Button(frame_bottom, text="SEARCH",
font="Cavolini 13 italic", borderwidth=5, relief=RAISED,
                       bg="lightblue1", command=search)
button_search.grid(row=2, column=7, padx=40, pady=3)
```

```
# Buttons of left side

button_1 = Button(frame_left_up, text="
Teams of IPL                ", font="Cavolini 13 italic",
              borderwidth=5, relief=RAISED, bg="yellow1",
command=B1_workspace)
button_2 = Button(frame_left_up, text="          MOST
RUNS SCORED         ", font="Cavolini 13 italic",
              borderwidth=5, relief=RAISED, bg="yellow1",
command=B2_workspace)
button_3 = Button(frame_left_up, text="         MOST
WICKET TAKERS       ", font="Cavolini 13 italic",
              borderwidth=5, relief=RAISED, bg="yellow1",
command=B3_workspace)
button_4 = Button(frame_left_up, text="       MOST SIXES
BY PLAYER        ", font="Cavolini 13 italic",
              borderwidth=5, relief=RAISED, bg="yellow1",
command=B4_workspace)
button_5 = Button(frame_left_up, text="              MOST
HATRICKS             ", font="Cavolini 13 italic",
              borderwidth=5, relief=RAISED, bg="yellow1",
command=B5_workspace)
button_6 = Button(frame_left_up, text="SINGLE
ACHIEVEMENTS RECORD", font="Cavolini 13 italic",
              borderwidth=5, relief=RAISED, bg="yellow1",
command=B6_workspace)
button_7 = Button(frame_left_up, text="         INDIVIDUAL
RECORDS        ", font="Cavolini 13 italic",
              borderwidth=5, relief=RAISED, bg="yellow1",
command=B7_workspace)
button_graph = Button(frame_left_up, text="
GRAPHS & CHARTS             ", font="Cavolini 13 italic",
                 borderwidth=5, relief=RAISED,  bg="yellow1",
command=BG_charts)
```

```
button_1.pack(pady=6, padx=8)
button_2.pack(pady=6, padx=8)
button_3.pack(pady=6, padx=8)
button_4.pack(pady=6, padx=8)
button_5.pack(pady=6, padx=8)
button_6.pack(pady=6, padx=8)
button_7.pack(pady=6, padx=8)
button_graph.pack(pady=6, padx=8)

# Left side Bottom frame

frame_left_down = Frame(frame_left_up, bg="black",
borderwidth=7, relief=RIDGE)
frame_left_down.pack(fill=Y)

log_label = Label(frame_left_down, text="LOG DATA",
bg="black", fg="green",
                font="Aparajita 17 bold")
log_label.pack()

log_label1 = Label(frame_left_down, text="WRITE
EVERYTHING IN CAPITAL LETTER", bg="black", fg="green",
                 font="Aparajita 12 bold")
log_label1.pack(padx=43)

gui.mainloop()


input('press any key to exit')
```

USER MANUAL

How to install Software:

Minimum Hardware Requirement

- Intel Pentium Celeron or similar processer-based PC.
- 8GB RAM and 512 MB HDD space are desirable.
- Standard (O devices like Keyboard and Molise etc. A printer is needed for hand-copy reports.

Software Requirement

- Windows 7 to Windows 10 desirable.
- Python 3.72 or higher should be installed with matplotlib, pymysql, Pandas, and PandasTable.
- MySQL Ver 6.1 or above with office Database must be present in the machine.

Database Installation

The software project is distributed with a backup copy of CSV files. No dummy records are present in the tables for testing proposes, which can be filled by inserting real data.
Note: The PC must have a MySQL server with the user (root) and password-123456789 If the root password is any other password, it can be changed by running MySQL Server Instance Configure Wizard.

Start > Program > MySQL > MSQL Server Instance Configure Wizard

To create a MySQL, database named crickters_data, simply follow the following steps:-

Step 1: Open MySQL and type the following command to create the database named Library.

mysql>create database crickters_data;

Step 2: Select the table by using use command.

mysql> use crickters_data;

Step 3: Create the table by using the given command in SQL.

mysql> create table player_data (PLAYER CHAR(30), TEAM CHAR(5), ROLL VARCHAR(10), MATCHES INT, RUNSWKTS INT);

This will create a database with required tables.

BIBLIOGRAPHY

- Informatics Practices with Python (for Class XII) by Sumita Arora.
- Informatics Practices with Python (for Class XI) by Sumita Arora.
- www.python.org
- www.mysql.org
- www.tutorialspoint.com

Other than my IP teacher "**ASHUTOSH SHARMA**" also helped me in creating my software