IPL Auction Analysis Report

Group No: Group 15

Student Names: Sruthi Gandla & Tejesvani Muppara Vijayaram

Executive Summary:

The objective of the IPL Auction Analysis project is to transform the way that players are evaluated at the Indian Premier League (IPL) auctions. The project used a huge amount of data on player statistics, auction years, and bid amounts, with a primary goal of assisting franchise departments. The team carefully constructed conceptual and logical models in order to structure and arrange the data from the IPL auctions, capturing important correlations between bid amounts, auction dynamics, and player performance. Utilizing NoSQL queries with MongoDB and SQL queries with MySQL, the project carried out an extensive study that gives a detailed picture of player performance and aided in the process of making strategic decisions. The incorporation of visualizations based on Python helped the dissemination of analytical results, empowering relevant parties to arrive at well-informed conclusions.

The analysis's significant conclusions provide franchise departments with practical information that helps them match bids to players' skills and past results. By optimizing auction techniques and reducing the possibility of overpaying or undervaluing players, this data-driven approach cleared the path for an increasingly intelligent and sophisticated IPL auction ecosystem. The project's future plans call for ongoing model, query, and application interface improvement in response to changing player performance indicators. This will guarantee the project's continued relevance and influence on the IPL auctions' dynamics.

I. Introduction:

The Indian Premier League (IPL) is a highly popular and profitable Twenty20 cricket league in India, established by the Board of Control for Cricket in India (BCCI) in 2008. It occurs annually during the Indian cricket season, typically from March to May. The IPL operates on a franchise-based system, featuring city-based teams owned by various individuals or organizations, with ownership decided through bidding before each season. Teams build their squads by participating in a player auction, where players are bought and sold based on their cricketing skills and potential contributions to the team.

About IPL auction:

The IPL auction is the primary mechanism by which teams are formed for each season. It allows franchise owners to build their squads from scratch or make necessary adjustments to their existing teams. This process ensures that teams are competitive and balanced. Team owners must carefully evaluate their squad's needs, assess available players, and allocate their budgets wisely to build a well-rounded team that can compete effectively in various conditions.

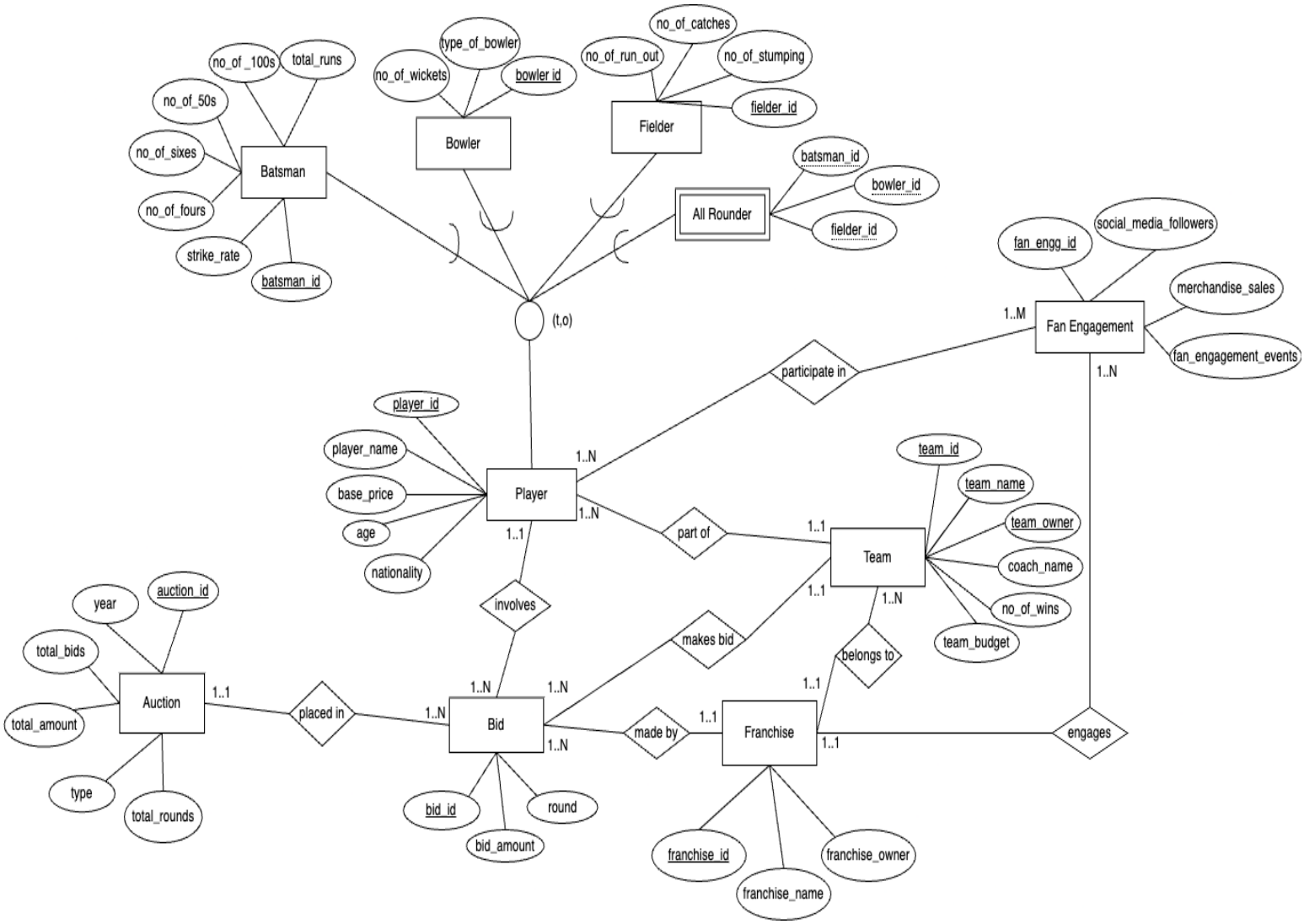
In our project, our primary focus is on assessing the strengths and weaknesses of cricket players. This analysis aims to provide valuable insights to potential bidders, helping them determine the appropriate value for each player quickly during the auction process. To facilitate this analysis, we have categorized every player into specific categories: batsman, bowler, fielder, all-rounder, and wicketkeeper.

For successful participation in the auction, bidders require access to comprehensive performance statistics from previous matches. For batsmen, crucial data includes the number of sixes and fours scored, the player's fastest run achievement, balls faced, and overall contribution to the team's batting performance. Bowlers' performance is evaluated based on data such as wickets taken, wide balls bowled, dot balls delivered, spin rate, handedness (left or right), and pace rate. Fielders' capabilities are assessed by considering factors like the number of catches taken, runs saved through fielding efforts, stumping proficiency, and other relevant metrics.

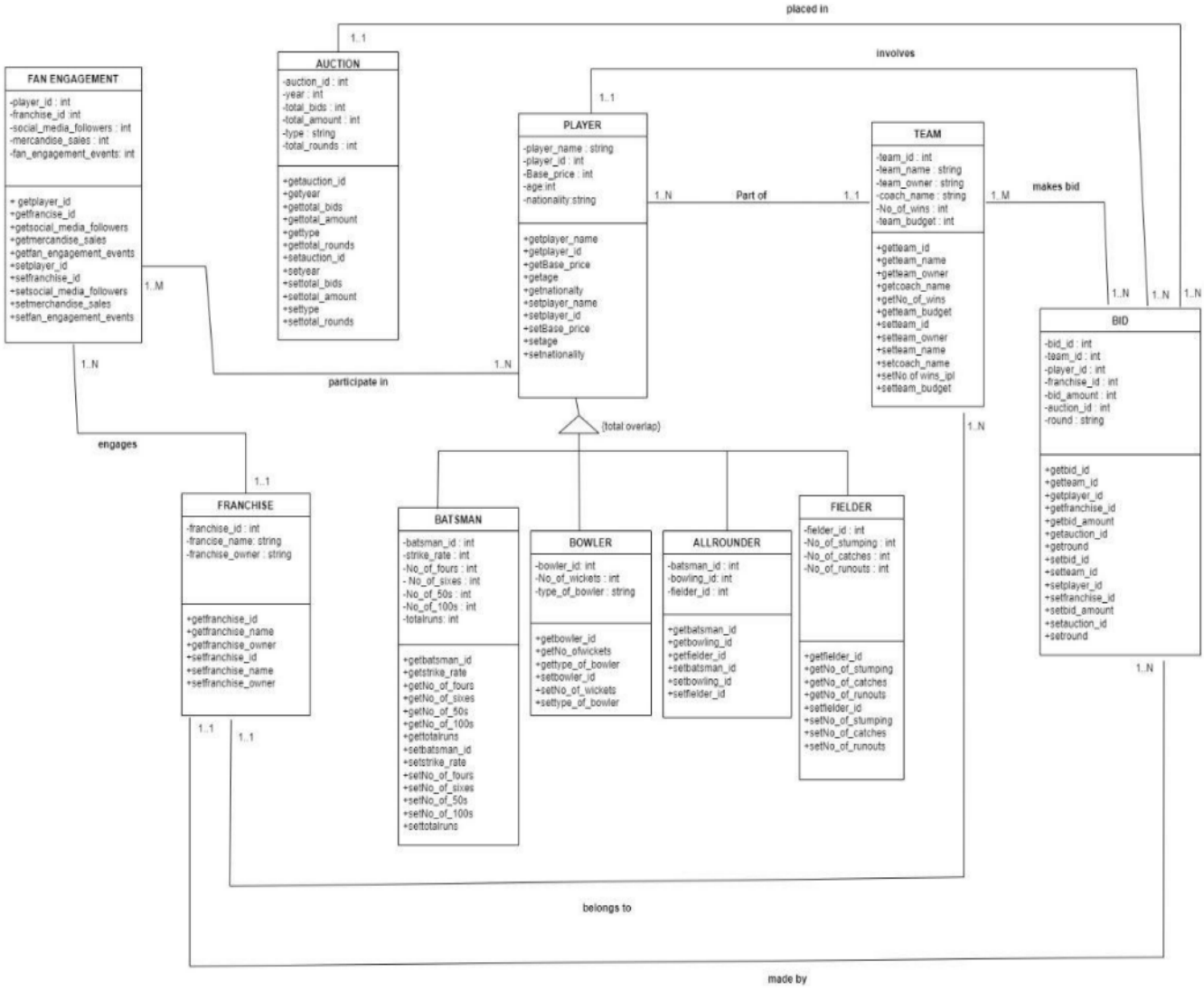
By providing a comprehensive and data-driven analysis of each player's performance and skills, we aim to empower bidders with the knowledge they need to make informed decisions during the auction process. This information allows them to efficiently assess a player's value and make swift decisions regarding choosing players for their teams.

II. Conceptual Data Modeling

1. EER Diagram



2. UML Diagram



III. Mapping EER Model To Relational Model:

Note: Primary Keys are underlined and Foreign Keys are in *italics*

1. Player (player_id, player_name, base_price, age, nationality, *team_id*)
    a. player_id - Primary Key
    b. Team_id - Foreign key from Team relation - NOT NULL
2. Team (team_id, team_name, team_owner, coach_name, no_of_wins, team_budget, *franchise_id*)
    a. team_id - Primary Key
    b. franchise_id - Foreign Key from Franchise relation - NOT NULL
3. Franchise (franchise_id, franchise_name, franchise_owner)
    a. franchise_id - Primary Key
4. Bid (bid_id, *player_id, team_id, franchise_id, auction_id*, bid_amount, round)
    a. bid_id - Primary Key
    b. player_id - Foreign key from Player relation - NOT NULL
    c. team_id - Foreign key from Team relation -NOT NULL
    d. franchise_id - Foreign key from Franchise relation -NOT NULL
    e. auction_id - Foreign key from Auction relation -NOT NULL
5. Auction (auction_id, year, total_bids, total_amount, type, total_rounds)
    a. auction_id - Primary Key
6. Fan_Engagement (fan_engg_id, *franchise_id*, social_media_followers, merchandise_sales, fan_engagement_events)
    a. fan_engg_id - Primary key
    b. franchise_id - Foreign key from Franchise relation -NOT NULL
7. participate_in(*player_id, fan_engg_id*)
    a. player_id - Foreign key from Player relation - NOT NULL
    b. fan_engg_id - Foreign key from Fan Engagement relation - NOT NULL
8. Batsman(player_id, batsman_id, strike_rate, no.of_fours, no.of_sixes, no_of_50s, no_of_100s, total_runs)
    a. batsman_id - Primary Key
9. Bowler(player_id, bowler_id, no_of_wickets, type_of_bowler)
    a. bowler_id - Primary Key
10. Fielder(player_id, fielder_id, no_of_stumping, no_of_catches, no_of_run_out)
    a. Fielder_id - Primary Key
11. All Rounder(*player_id, batsman_id, bowler_id, fielder_id*)
    a. batsman_id - Foreign key from Player relation -NOT NULL
    b. b. bowler_id - Foreign key from Player relation -NOT NULL
    c. fielder-id - Foreign key from Player relation -NOT NULL

IV. Implementation of Relation Model via MySQL and NoSQL

MySQL Implementation:
The database was created in MySQL and the following queries were performed

Query1: Retrieve Top 10 batsmen who have the highest strike rate.

```
SELECT b.player_id, p.player_name, b.strike_rate
FROM batsman b, player p
WHERE b.player_id = p.player_id
ORDER BY strike_rate DESC
LIMIT 10;
```

player_id	player_name	strike_rate
9	Anthony Watson	149
68	John Shaw	147
153	Shane Freeman	147
108	Marc Smith	146
239	Timothy Jackson	144
242	Gregg Russell	144
12	James Tucker	143
87	Johnny Mack	142
235	Bobby Morgan	142
28	Richard Santiago	138

Query 2: Retrieve Top 10 Bowlers who have taken maximum no. of wickets.

```
SELECT b.player_id, p.player_name, b.no_of_wickets
FROM bowler b, player p
WHERE b.player_id = p.player_id
ORDER BY no_of_wickets DESC
LIMIT 10;
```

player_id	player_name	no_of_wickets
63	Timothy Reed DVM	499
14	Mark Nunez	496
133	Anthony Vasquez	495
184	Steven Thomas	483
229	Willie Bonilla	483
1	David Hayes	482
50	Bradley Thomas	482
47	Eric Davis	477
207	Earl Joyce	474
106	Daniel Rubio	473

Query 3: Retrieve the Top 10 Fielders in terms of their total wickets, which is the sum of no. of run outs, no. of catches, no. of stumping.

```
SELECT f.player_id, p.player_name, f.no_of_catches, f.no_of_stumping,
f.no_of_run_out, f.no_of_catches+f.no_of_stumping+f.no_of_run_out as
total_wickets
FROM fielder f, player p
WHERE f.player_id = p.player_id
ORDER BY total_wickets DESC
LIMIT 10;
```

player_id	player_name	no_of_catches	no_of_stumping	no_of_run_out	total_wickets
49	James Martin	46	9	20	75
55	Henry Davenport	50	6	18	74
98	Larry Pacheco	50	3	20	73
247	Matthew Thomas	47	5	18	70
28	Richard Santiago	48	8	12	68
74	Louis Salas	43	5	20	68
211	Matthew Ashley	39	10	18	67
225	Bradley Matthews	45	2	20	67
238	James Ortiz	43	8	16	67
35	Keith Murray	38	10	18	66

Query 4: Retrieving all the players who are well versed in batting, bowling and fielding which is termed as all rounders.

```
SELECT a.player_id, p.player_name, a.batsman_id, a.bowler_id, a.fielder_id
FROM all_rounder a
JOIN player p ON a.player_id = p.player_id
WHERE bowler_id IS NOT NULL
AND batsman_id IS NOT NULL
AND fielder_id IS NOT NULL;
```

player_id	player_name	batsman_id	bowler_id	fielder_id
1	David Hayes	1	1	1
3	Ricky Gordon	3	2	3
7	Douglas Lewis	5	5	6
9	Anthony Watson	6	6	8
12	James Tucker	8	8	11
13	Gregory Richard	9	9	12
14	Mark Nunez	10	10	13
16	Frank Franklin	11	12	15
27	Jason Lee	17	18	22
28	Richard Santiago	18	19	23

Query 5: Retrieve the player with the highest bid for each year along with the team that selected them.

```
SELECT a.year, p.player_name, t.team_name, b1.bid_amount
FROM bid b1, auction a, player p, team t
WHERE b1.auction_id = a.auction_id
AND b1.player_id = p.player_id
AND b1.team_id = t.team_id
AND b1.bid_amount = (
   SELECT MAX(b2.bid_amount)
   FROM bid b2
   WHERE b2.auction_id = b1.auction_id
)
ORDER BY b1.auction_id;
```

year	player_name	team_name	bid_amount
2018	Frank Franklin	Delhi Capitals	150000000.00
2019	Joshua Lyons	Mumbai Indians	148000000.00
2020	Anthony Vasquez	Mumbai Indians	149000000.00
2021	Gene Boyd	Chennai Super Kings	148000000.00
2021	Jonathan Rivera	Delhi Capitals	148000000.00
2022	John Turner	Delhi Capitals	149000000.00
2023	Marc King	Mumbai Indians	145000000.00

Query 6: Retrieve the average bid amount of each player

```
SELECT p.player_id, p.player_name, AVG(b.bid_amount)
FROM player p , bid b, auction a
WHERE p.player_id = b.player_id
AND a.auction_id = b.auction_id
GROUP BY b.player_id
ORDER BY player_id ASC;
```

player_id	player_name	AVG(b.bid_amount)
1	David Hayes	10949498.500000
2	Matthew Castillo	45543055.333333
3	Ricky Gordon	110290871.000000
4	Randall Jones	78651326.000000
5	David Sims	66058347.000000
7	Douglas Lewis	22392788.000000
8	Dennis Perry	110292465.000000
9	Anthony Watson	76426282.000000
10	James Barton	129000000.000000
11	Adrian Jackson	109000000.000000

Query 7:Retrieve the players who have played all the seasons.

```
SELECT player_id,player_name
FROM player p
WHERE NOT EXISTS
( SELECT *
FROM auction a
WHERE NOT EXISTS
(SELECT *
FROM bid b
WHERE p.player_id=b.player_id
AND a.auction_id=b.auction_id));
```

player_id	player_name
120	Vincent Welch
223	David Reid
NULL	NULL

Query 8: Retrieve the team with maximum number of wins in the overall IPL seasons

```
SELECT team_name, no_of_wins
FROM team
WHERE no_of_wins = (
                        SELECT MAX(no_of_wins)
                        FROM team
                );
```

team_name	no_of_wins
Mumbai Indians	5
Chennai Super Kings	5

Query 9: Retrieve the player who has played the highest no. of seasons.

```
SELECT  b.player_id,p.player_name  ,count(distinct
b.auction_id)
FROM bid b,player p
WHERE p.player_id=b.player_id
GROUP BY b.player_id
HAVING COUNT(b.player_id) >= ALL (SELECT count(*)
                                  FROM bid b
                                  GROUP BY b.player_id
                                  );
```

player_id	player_name	count(distinct b.auction_id)
120	Vincent Welch	6
223	David Reid	6

Query 10: Retrieve the players who have high demand and less demand for the corresponding auction years.

```
SELECT a.year, p.player_name, b.round_no,
   CASE
      WHEN b.round_no < 3 THEN 'High Demanded Player'
      WHEN b.round_no > 5 THEN 'Less Demanded Player'
      ELSE 'Normal Demanded Player'
   END AS demand_label
FROM auction a
JOIN bid b ON a.auction_id = b.auction_id
JOIN player p ON b.player_id = p.player_id
WHERE (b.round_no < 3 OR b.round_no > 5)
ORDER BY a.year desc, demand_label, b.round_no;
```

year	player_name	round_no	demand_label
2023	Jason Joyce	1	High Demanded Player
2023	Benjamin Vaughan	1	High Demanded Player
2023	Douglas Lewis	1	High Demanded Player
2023	Darryl Lee	1	High Demanded Player
2023	Mark Nunez	1	High Demanded Player
2023	Gregory Richard	1	High Demanded Player
2023	Arthur Brown	1	High Demanded Player
2023	Jacob Park	1	High Demanded Player

Query 11: Determine the players who are exclusively focused on batting and do not engage in bowling.

```
SELECT p.player_id, p.player_name
FROM batsman b
JOIN player p ON p.player_id=b.player_id
EXCEPT
SELECT p.player_id, p.player_name
FROM bowler bl
JOIN player p ON p.player_id=bl.player_id ;
```

player_id	player_name
2	Matthew Castillo
5	David Sims
10	James Barton
17	Richard Mosley
22	Connor Kim
30	Brent Young
33	Darryl Lee
38	Tony Diaz
45	Douglas Christensen DVM
51	Russell King
61	Scott Nelson
68	John Shaw
75	Nathan Sanchez

Query 12: Determine the players' profit for the previous year by assessing the difference between the base price and the actual bid amount in the auction.

```
SELECT t.player_id,t.player_name,t.bid_amount, t.base_price,
            t.bid_amount - t.base_price as player_profit
FROM (SELECT p.player_id, p.player_name, p.base_price, max(b.bid_amount) as bid_amount
      FROM bid b, player p
      WHERE b.player_id = p.player_id
      AND b.auction_id = 6
```

player_id	player_name	bid_amount	base_price	player_profit
18	Marc King	145000000.00	858096.00	144141904.00
220	Derrick Reed	142000000.00	1016673.00	140983327.00
131	Justin Aguilar	141000000.00	3986888.00	137013112.00
194	Steven Jordan	137000000.00	2744498.00	134255502.00
63	Timothy Reed DVM	138000000.00	3878187.00	134121813.00
120	Vincent Welch	134000000.00	1147319.00	132852681.00
187	Benjamin Vaughan	140000000.00	8144844.00	131855156.00
13	Gregory Richard	135000000.00	3285888.00	131714112.00
49	James Martin	129000000.00	732263.00	128267737.00
3	Ricky Gordon	128000000.00	410981.00	127589019.00
127	Louis Brown	128000000.00	647247.00	127352753.00
17	Richard Mosley	129000000.00	3506617.00	125493383.00

```
        GROUP BY p.player_id) AS t
ORDER BY player_profit desc;
```

NoSQL Implementation

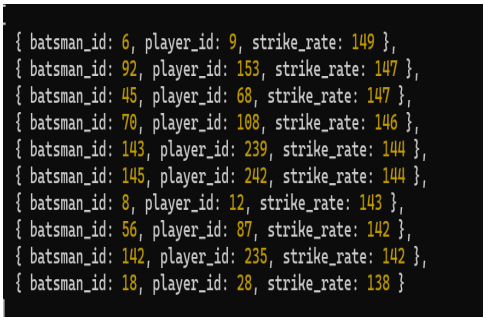
A new database connection in the name of "Project_IPLSchema" is created in MongoDB and all the necessary collections have been created for the NoSQL implementation. Following are the queries and corresponding outputs executed in Mongosh:

Query 1: Retrieving the player names of age greater than 24.

```
db.players.find({age:{$gte:24}},{player_name:1,age:1,_id:0})
```

Query 2: Retrieving Top 10 batsman who has highest strike rate.

```
db.batsman.find({}, {batsman_id:1, player_id:1, strike_rate:1, _id:0}).sort({strike_rate: -1}).limit(10)
```



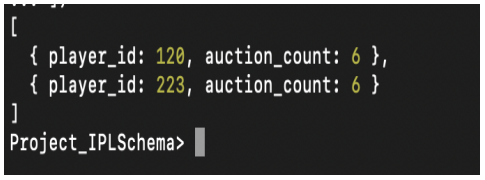
Query 3: Retrieve the player with the highest bid for each year along with the team that selected them.

```
db.bid.aggregate([
 {
  $lookup: {
    from: "auction",
    localField: "auction_id",
    foreignField: "auction_id",
    as: "auction",
  },
 },
 { $unwind: "$auction",
 },
 {
  $lookup: {
    from: "player",
    localField: "player_id",
    foreignField: "player_id",
    as: "player",
  },
 },
 { $unwind: "$player",
 },
 {
  $lookup: {
    from: "team",
    localField: "team_id",
    foreignField: "team_id",
    as: "team",
  },
 },
 { $unwind: "$team",
 },
 {
  $group: {
    _id: "$auction.auction_id",
    year: { $first: "$auction.year", },
```

```
    player_name: { $first: "$player.player_name", },
    team_name: { $first: "$team.team_name", },
    bid_amount: { $max: "$bid_amount", },
   },
  },
  {
   $sort: { _id: 1,},
  },
])
```

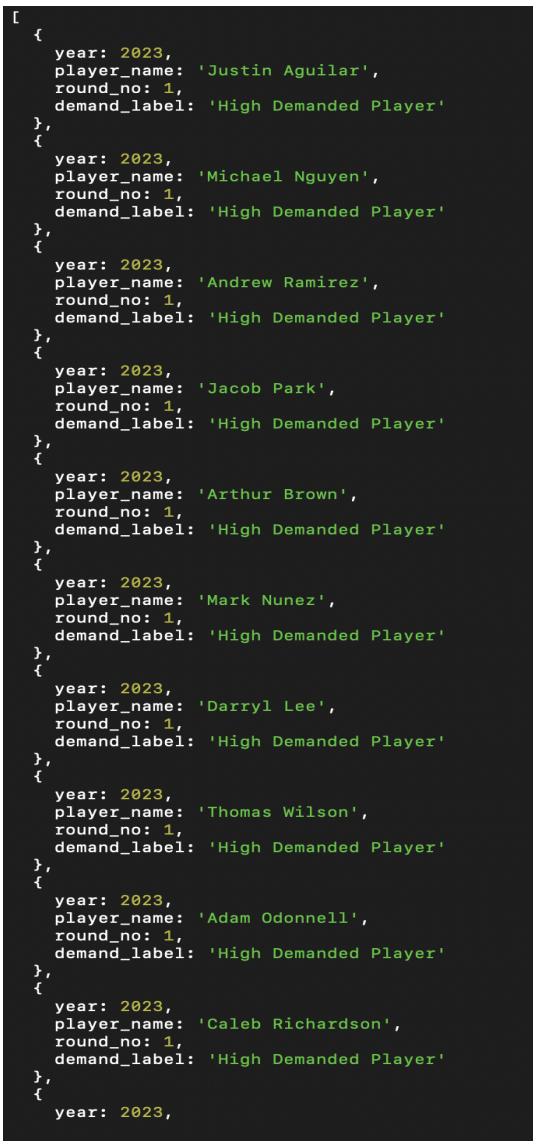
Query 4: Retrieving the player Names who has played the Max Number of seasons.

```
db.bid.aggregate([
  {
   $group: {
    _id: "$player_id",
     uniqueAuctionCount: { $addToSet: "$auction_id",
},
    totalCount: { $sum: 1 },
   },
  },
  {
   $addFields: {
    player_id: "$_id",
    auction_count: { $size: "$uniqueAuctionCount", },
   },
  },
  {
   $sort: { auction_count: -1 },
  },
  {
   $limit: 2,
  },
  {
   $project: { player_id: 1, auction_count: 1, _id: 0, },
  },
])
```



Query 5: Retrieving the players who have high demand and less demand for the corresponding auction years.

```
db.auction.aggregate([
 {
  $lookup: {
    from: "bid",
    localField: "auction_id",
    foreignField: "auction_id",
    as: "bids",
  },
 },
 { $unwind: "$bids",
 },
 {
  $lookup: {
    from: "player",
    localField: "bids.player_id",
    foreignField: "player_id",
    as: "player_info",
  },
 },
 { $unwind: "$player_info",
 },
 {
  $match: {
    $or: [
      { "bids.round_no": { $lt: 3, }, },
      { "bids.round_no": { $gt: 5, }, },
    ],
  },
 },
 {
  $project: {
    _id: 0,
    year: "$year",
    player_name: "$player_info.player_name",
```



```
[
  {
    year: 2023,
    player_name: 'Justin Aguilar',
    round_no: 1,
    demand_label: 'High Demanded Player'
  },
  {
    year: 2023,
    player_name: 'Michael Nguyen',
    round_no: 1,
    demand_label: 'High Demanded Player'
  },
  {
    year: 2023,
    player_name: 'Andrew Ramirez',
    round_no: 1,
    demand_label: 'High Demanded Player'
  },
  {
    year: 2023,
    player_name: 'Jacob Park',
    round_no: 1,
    demand_label: 'High Demanded Player'
  },
  {
    year: 2023,
    player_name: 'Arthur Brown',
    round_no: 1,
    demand_label: 'High Demanded Player'
  },
  {
    year: 2023,
    player_name: 'Mark Nunez',
    round_no: 1,
    demand_label: 'High Demanded Player'
  },
  {
    year: 2023,
    player_name: 'Darryl Lee',
    round_no: 1,
    demand_label: 'High Demanded Player'
  },
  {
    year: 2023,
    player_name: 'Thomas Wilson',
    round_no: 1,
    demand_label: 'High Demanded Player'
  },
  {
    year: 2023,
    player_name: 'Adam Odonnell',
    round_no: 1,
    demand_label: 'High Demanded Player'
  },
  {
    year: 2023,
    player_name: 'Caleb Richardson',
    round_no: 1,
    demand_label: 'High Demanded Player'
  },
  {
    year: 2023,
```

```
      round_no: "$bids.round_no",
      demand_label: {
       $cond: {
        if: { $lt: ["$bids.round_no", 3], },
        then: "High Demanded Player",
        else: {
         $cond: {
          if: { $gt: ["$bids.round_no", 5], },
          then: "Less Demanded Player",
          else: "Normal Demanded Player",
         },
        },
       },
      },
     },
    },
    {
     $sort: { year: -1, demand_label: 1, round_no: 1,
     },
    },
  ])
```

Query 6: Retrieving the Top 10 Batsman who has scored Top 50's and Top 100's.

```
db.batsman.aggregate([
  {
   $sort: {
    no_of_100s: -1,
    no_of_50s: -1,
   },
  },
  {
   $lookup: {
    from: "player",
    localField: "player_id",
    foreignField: "player_id",
    as: "player_info",
```

```
  },
 },
 {
  $unwind: "$player_info",
 },
 {
  $project: {
    _id: 0,
    player_id: 1,
  player_name: "$player_info.player_name",
    no_of_50s: 1,
    no_of_100s: 1,
  },
 },
 {
  $limit: 10,
 },
])
```

```
[
  {
    player_id: 129,
    no_of_50s: 10,
    no_of_100s: 5,
    player_name: 'Daniel Jefferson'
  },
  {
    player_id: 83,
    no_of_50s: 9,
    no_of_100s: 5,
    player_name: 'James James'
  },
  {
    player_id: 113,
    no_of_50s: 9,
    no_of_100s: 5,
    player_name: 'Todd Green'
  },
  {
    player_id: 30,
    no_of_50s: 8,
    no_of_100s: 5,
    player_name: 'Brent Young'
  },
  {
    player_id: 31,
    no_of_50s: 8,
    no_of_100s: 5,
    player_name: 'Jonathan Bailey'
  },
  {
    player_id: 218,
    no_of_50s: 8,
    no_of_100s: 5,
    player_name: 'Daniel Singleton'
  },
  {
    player_id: 99,
    no_of_50s: 7,
    no_of_100s: 5,
    player_name: 'Logan White'
  },
  {
    player_id: 194,
    no_of_50s: 7,
    no_of_100s: 5,
    player_name: 'Steven Jordan'
  },
  {
    player_id: 90,
    no_of_50s: 5,
    no_of_100s: 5,
    player_name: 'Austin Medina'
  },
  {
    player_id: 198,
    no_of_50s: 5,
    no_of_100s: 5,
    player_name: 'Christopher Thomas'
  }
]
Project_IPLSchema>
```

V. Database Connection with Python

A database connection was established with Python from Jupyter Notebook to visualize and analyze the data. Below is the code snippet for database connection:

```
import mysql.connector
from mysql.connector import Error

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
try:
    connection = mysql.connector.connect(
        host = 'localhost',
        database = 'Project_IPLSchema',
        user = 'root',
        password = 'software@123',
        auth_plugin = 'mysql_native_password'
    )
    if connection.is_connected():
        db_Info = connection.get_server_info()
        print("Connected to MySQL Server version ", db_Info)
        cursor = connection.cursor()
        cursor.execute("select database();")
        record = cursor.fetchone()
        print("Your connected to database: ", record)

except Error as e:
    print("Error while connecting to MySQL", e)
```

```
Connected to MySQL Server version  8.0.34
Your connected to database:  ('project_iplschema',)
```

The following visualizations were performed on the data:

Chart 1: The Social Media influence of different Franchise

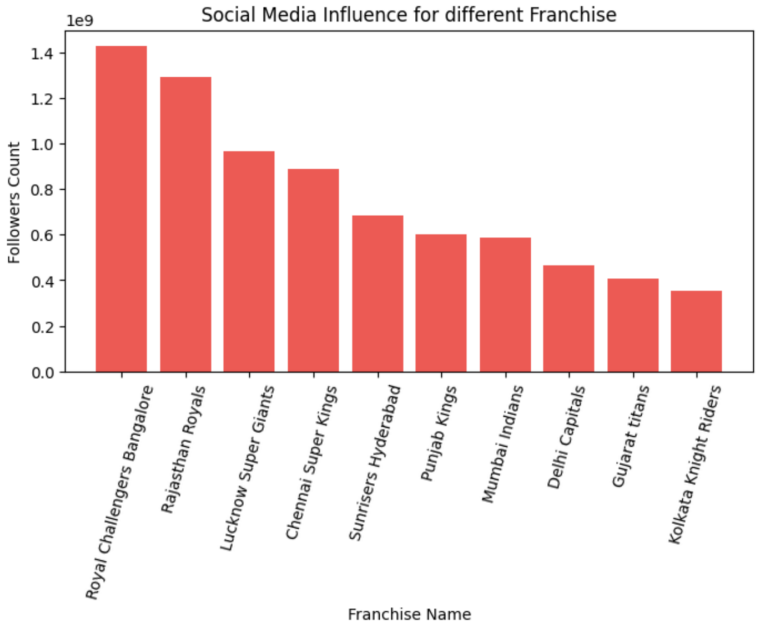


Chart 2: Distribution of Player by number of seasons

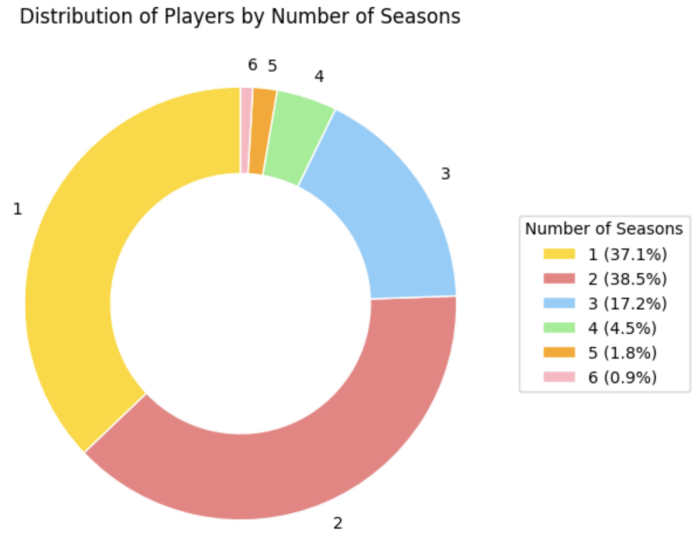


Chart 3: Total Amount Spent in IPL Auction Over the Years

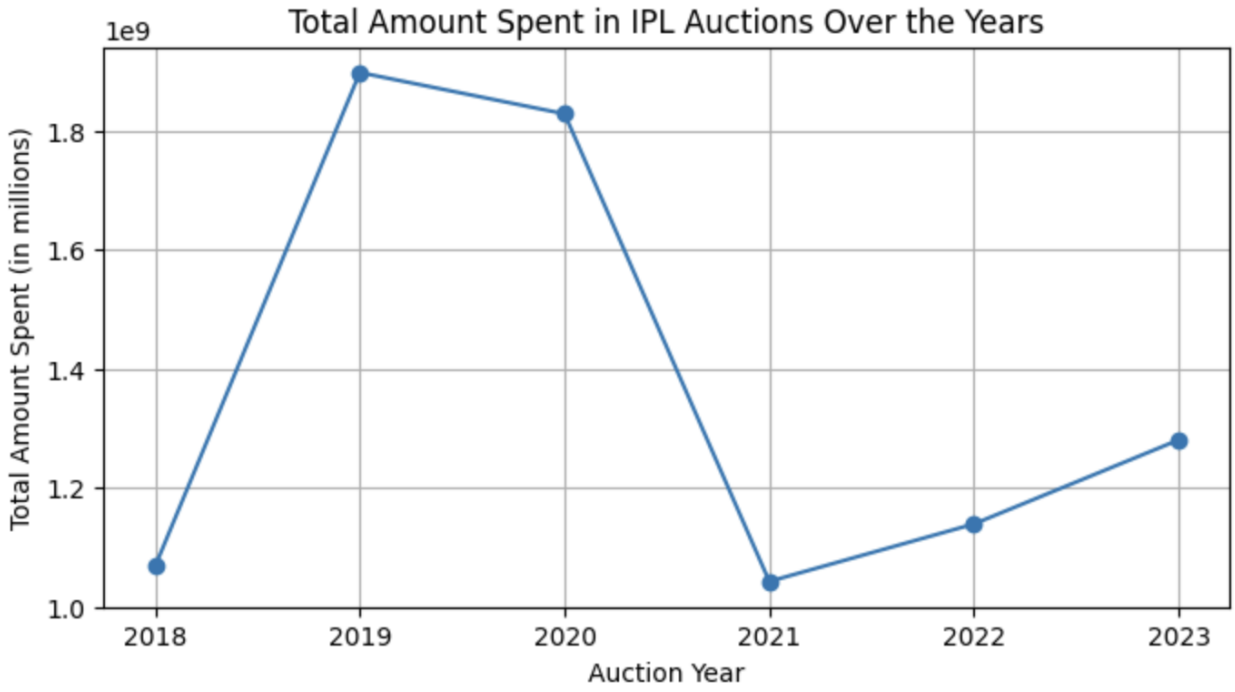


Chart 4: Distribution of Players as Batsmen and Bowlers

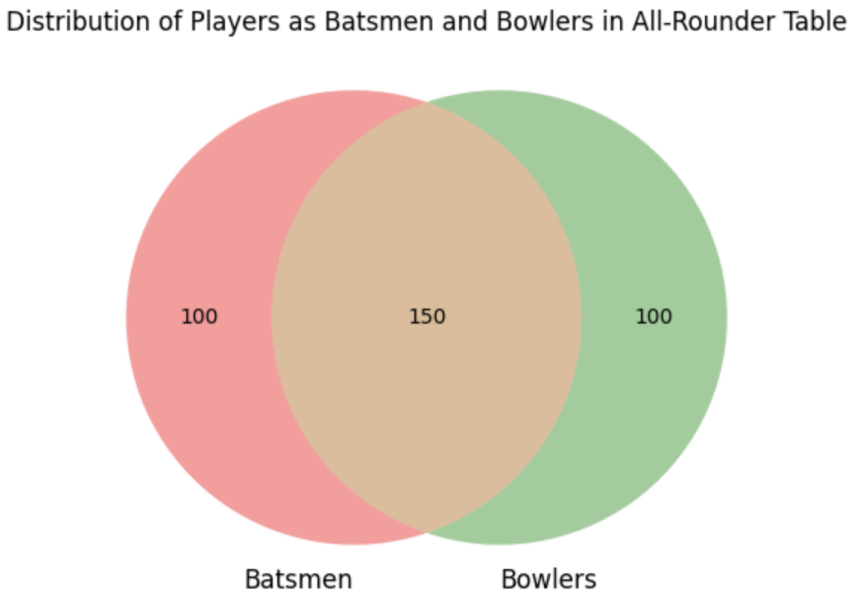
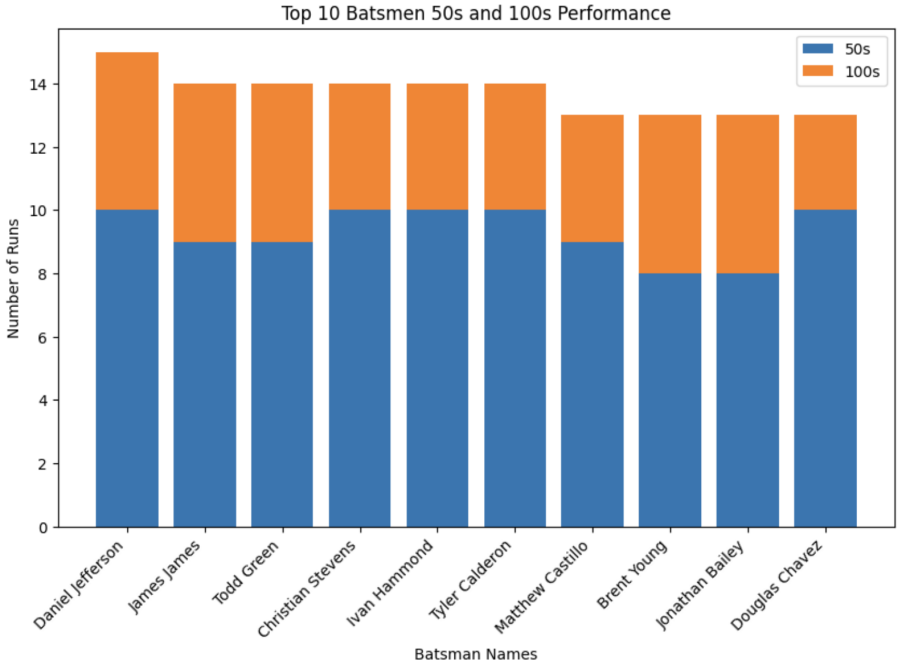


Chart 5: Top 10 Batsmen with Highest Number of 100's and 50's



VI. Summary and Recommendation

To sum up, our player analysis efforts have yielded valuable insights into IPL seasons by extracting crucial data and identifying top performers in fielding, bowling, and batting. Through the comparison of players' bid amounts from previous auctions with their performance, we've identified players with high and low demand, providing crucial information for strategic team-building. Utilizing both NoSQL and SQL queries, we were able to swiftly assess team capabilities by analyzing the most victories during an entire IPL season. The deployment of a Python application enhanced our ability to visualize and comprehend important metrics, such as the total amount spent in IPL auctions over time and the top 10 batters with the best scores.

Based on past performance data, these visualizations help franchises determine appropriate bid amounts and make well-informed decisions about player valuation. In order to help franchises with their strategic planning for upcoming IPL seasons, we plan to develop a user experience (UX) interface that will feature a player dashboard that highlights strengths and weaknesses derived from our thorough analysis.