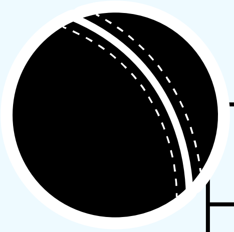
IE6700 DATA MANAGEMENT FOR ANALYTICS

Cricket Auction Player Performance Database Management System

Yash Bhadreshwara
Sparsh Marwah

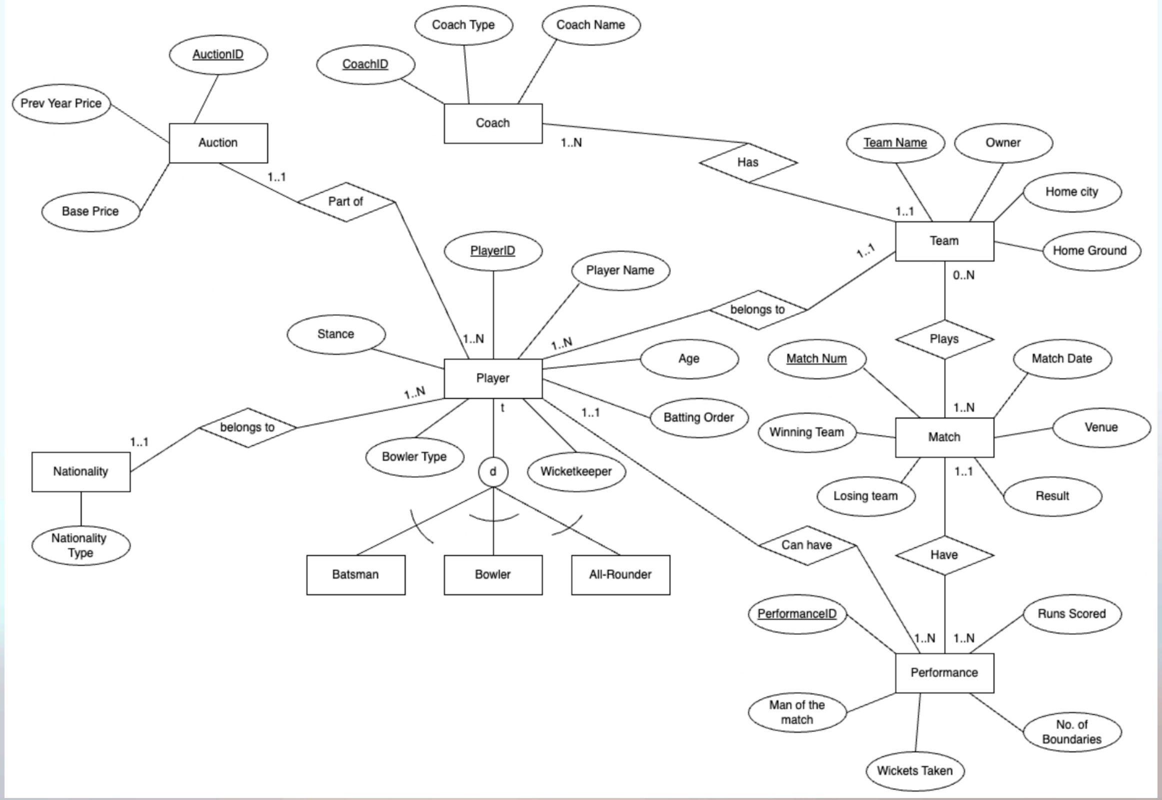
Group 6



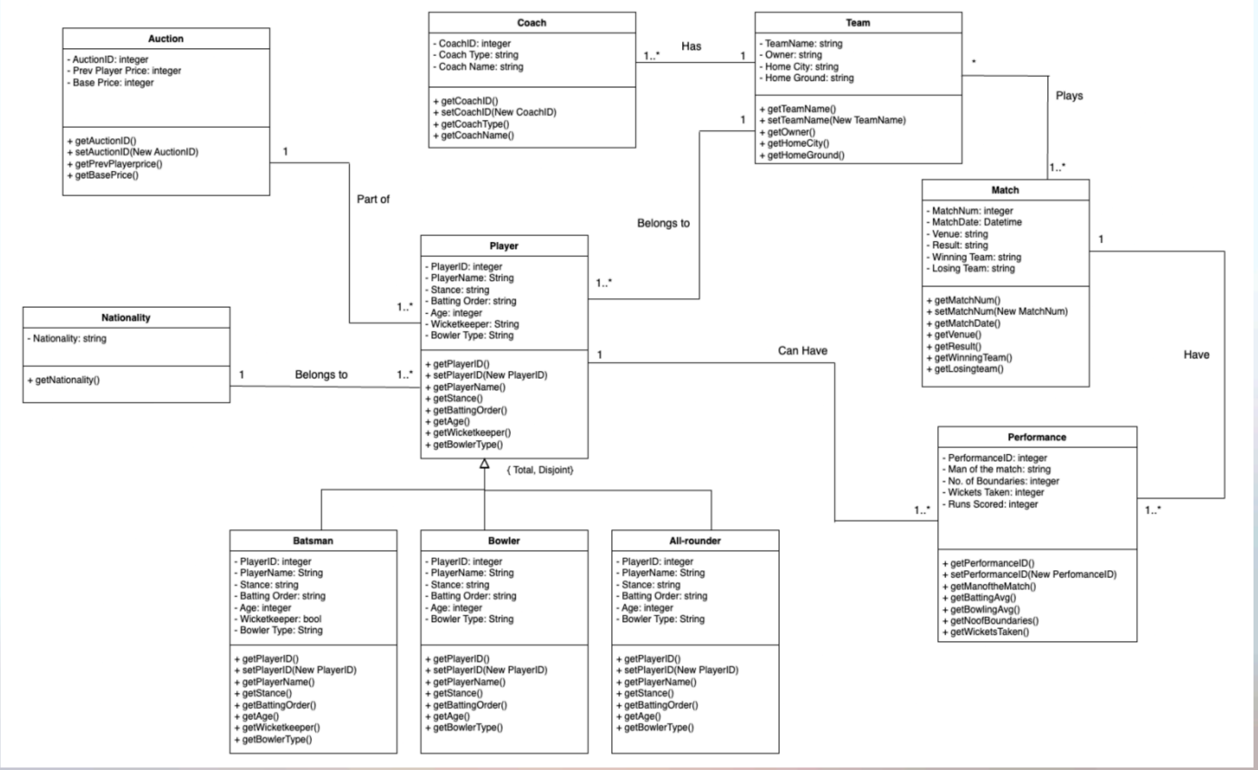
PROBLEM STATEMENT

In the world of professional cricket, team owners attend auctions to pick skilled players who can help their teams win. Choosing the right players during these auctions is super important because it directly affects how well the team does in games and competitions. To make smart choices, team owners and managers need to look at how well a player has done in the past, what role they play, and how they can help the team win. The main issue we are tackling is creating a strong system that uses data to help team owners pick the best players during these cricket auctions. We want this system to help team owners and managers make smart decisions that will make their cricket teams better and more competitive.

EER MODEL


AuctionID
Prev Year Price
Auction
Base Price
1..1
Part of
Coach Type
Coach Name
CoachID
Coach
1..N
Has
Team Name
Owner
Home city
1..1
Team
Home Ground
1..1
0..N
belongs to
PlayerID
Player Name
1..N
1..N
Stance
Player
Age
1..N
Batting Order
1..1
t
Plays
Match Num
Match Date
1..N
Winning Team
Match
Venue
Losing team
1..1
Result
Nationality
1..1
belongs to
Bowler Type
d
Wicketkeeper
Nationality Type
Batsman
Bowler
All-Rounder
Can have
Have
PerformanceID
Runs Scored
1..N
1..N
Performance
Man of the match
No. of Boundaries
Wickets Taken

UML



Auction

- AuctionID: integer
- Prev Player Price: integer
- Base Price: integer

+ getAuctionID()
+ setAuctionID(New AuctionID)
+ getPrevPlayerprice()
+ getBasePrice()

Coach

- CoachID: integer
- Coach Type: string
- Coach Name: string

+ getCoachID()
+ setCoachID(New CoachID)
+ getCoachType()
+ getCoachName()

Team

- TeamName: string
- Owner: string
- Home City: string
- Home Ground: string

+ getTeamName()
+ setTeamName(New TeamName)
+ getOwner()
+ getHomeCity()
+ getHomeGround()

Match

- MatchNum: integer
- MatchDate: Datetime
- Venue: string
- Result: string
- Winning Team: string
- Losing Team: string

+ getMatchNum()
+ setMatchNum(New MatchNum)
+ getMatchDate()
+ getVenue()
+ getResult()
+ getWinningTeam()
+ getLosingteam()

Nationality

- Nationality: string

+ getNationality()

Player

- PlayerID: integer
- PlayerName: String
- Stance: string
- Batting Order: string
- Age: integer
- Wicketkeeper: String
- Bowler Type: String

+ getPlayerID()
+ setPlayerID(New PlayerID)
+ getPlayerName()
+ getStance()
+ getBattingOrder()
+ getAge()
+ getWicketkeeper()
+ getBowlerType()

{ Total, Disjoint}

Performance

- PerformanceID: integer
- Man of the match: string
- No. of Boundaries: integer
- Wickets Taken: integer
- Runs Scored: integer

+ getPerformanceID()
+ setPerformanceID(New PerfomanceID)
+ getManoftheMatch()
+ getBattingAvg()
+ getBowlingAvg()
+ getNoofBoundaries()
+ getWicketsTaken()

Batsman

- PlayerID: integer
- PlayerName: String
- Stance: string
- Batting Order: string
- Age: integer
- Wicketkeeper: bool
- Bowler Type: String

+ getPlayerID()
+ setPlayerID(New PlayerID)
+ getPlayerName()
+ getStance()
+ getBattingOrder()
+ getAge()
+ getWicketkeeper()
+ getBowlerType()

Bowler

- PlayerID: integer
- PlayerName: String
- Stance: string
- Batting Order: string
- Age: integer
- Bowler Type: String

+ getPlayerID()
+ setPlayerID(New PlayerID)
+ getPlayerName()
+ getStance()
+ getBattingOrder()
+ getAge()
+ getBowlerType()

All-rounder

- PlayerID: integer
- PlayerName: String
- Stance: string
- Batting Order: string
- Age: integer
- Bowler Type: String

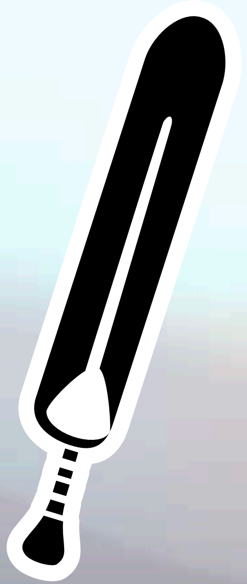
+ getPlayerID()
+ setPlayerID(New PlayerID)
+ getPlayerName()
+ getStance()
+ getBattingOrder()
+ getAge()
+ getBowlerType()

SCOPE OF ANALYTICS

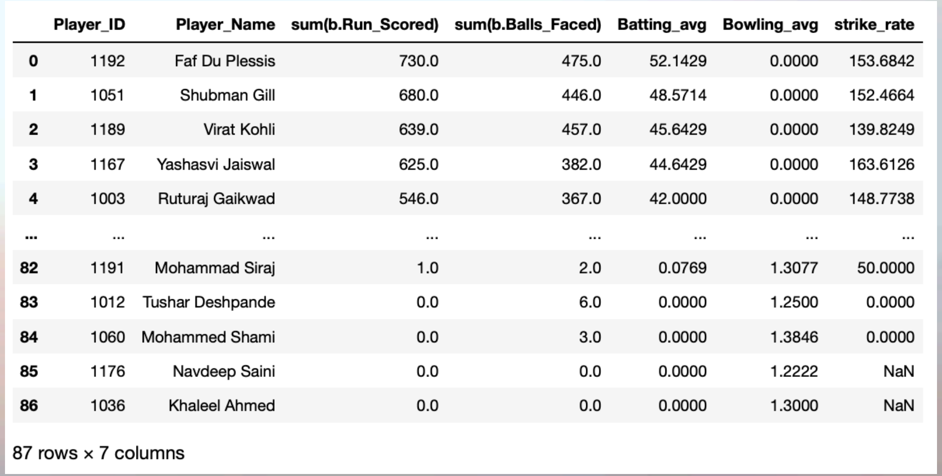
- This database can be used to track individual performances of all the players which can be used to coaches to concentrate on that particular player.
- Our database also can be used by training schools to train the students based on the previous player’s performance statistics.

SQL QUERY 1

```
sql1 = """
select a.Player_ID,a.Player_Name, sum(b.Run_Scored),sum(b.Balls_Faced),sum(b.Run_Scored)/
count(b.Match_Num) as Batting_avg,
sum(b.Wickets_Taken)/count(b.Match_Num) as Bowling_avg, (sum(b.Run_Scored)/sum(b.Balls_Faced))*100
as strike_rate
from Player_database.performances b
left join Player_database.players a
on b.Player_ID = a.Player_ID
group by 1,2
having count(match_num) >= 7
order by 5 desc
;"""

result_df1 = pd.read_sql_query(sql1, connection)
```



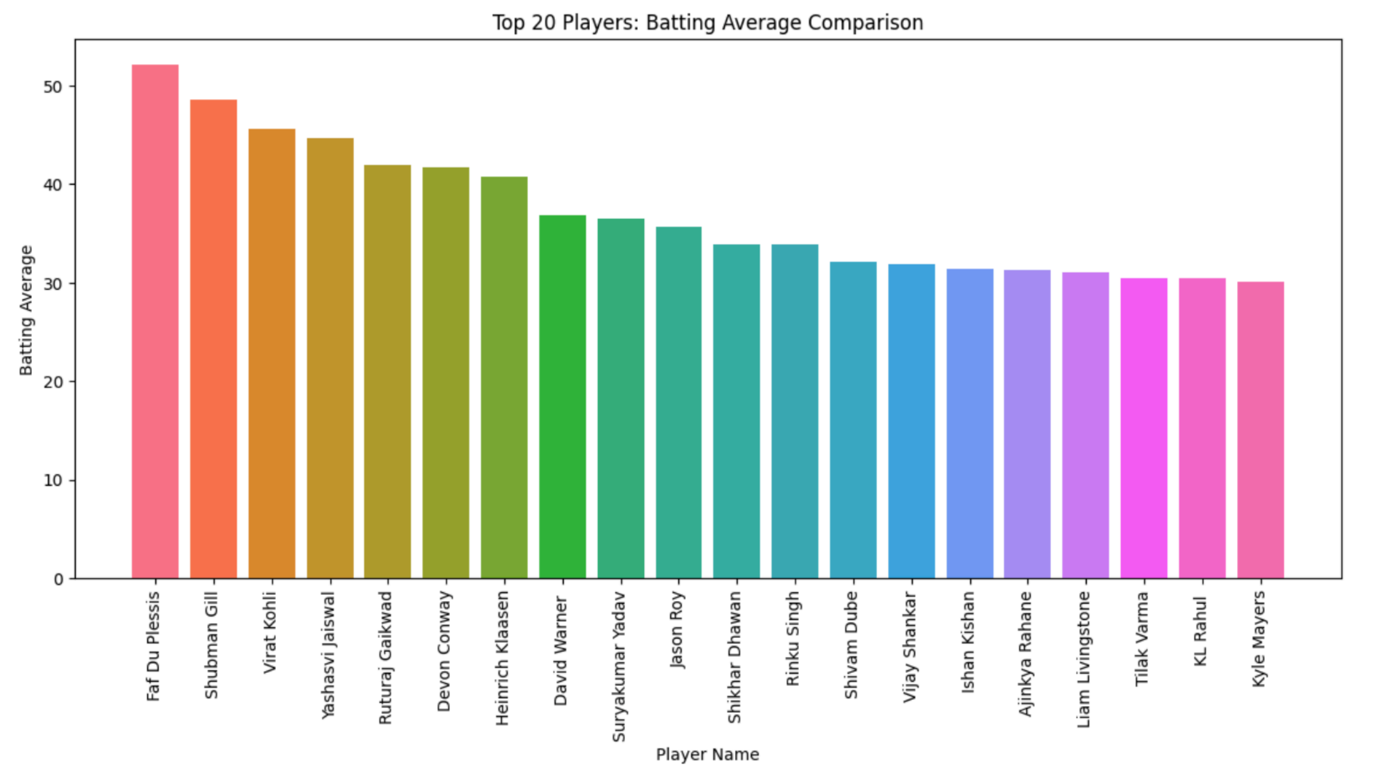
OUTPUT



	Player_ID	Player_Name	sum(b.Run_Scored)	sum(b.Balls_Faced)	Batting_avg	Bowling_avg	strike_rate
0	1192	Faf Du Plessis	730.0	475.0	52.1429	0.0000	153.6842
1	1051	Shubman Gill	680.0	446.0	48.5714	0.0000	152.4664
2	1189	Virat Kohli	639.0	457.0	45.6429	0.0000	139.8249
3	1167	Yashasvi Jaiswal	625.0	382.0	44.6429	0.0000	163.6126
4	1003	Ruturaj Gaikwad	546.0	367.0	42.0000	0.0000	148.7738
...	...	...	...	...	...	...	...
82	1191	Mohammad Siraj	1.0	2.0	0.0769	1.3077	50.0000
83	1012	Tushar Deshpande	0.0	6.0	0.0000	1.2500	0.0000
84	1060	Mohammed Shami	0.0	3.0	0.0000	1.3846	0.0000
85	1176	Navdeep Saini	0.0	0.0	0.0000	1.2222	NaN
86	1036	Khaleel Ahmed	0.0	0.0	0.0000	1.3000	NaN

87 rows × 7 columns

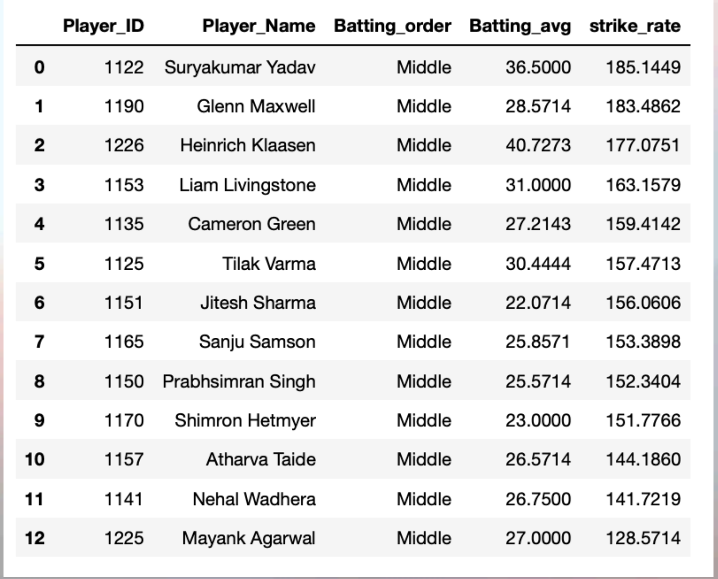
QUERY 1 VISUALIZATION



SQL QUERY 2

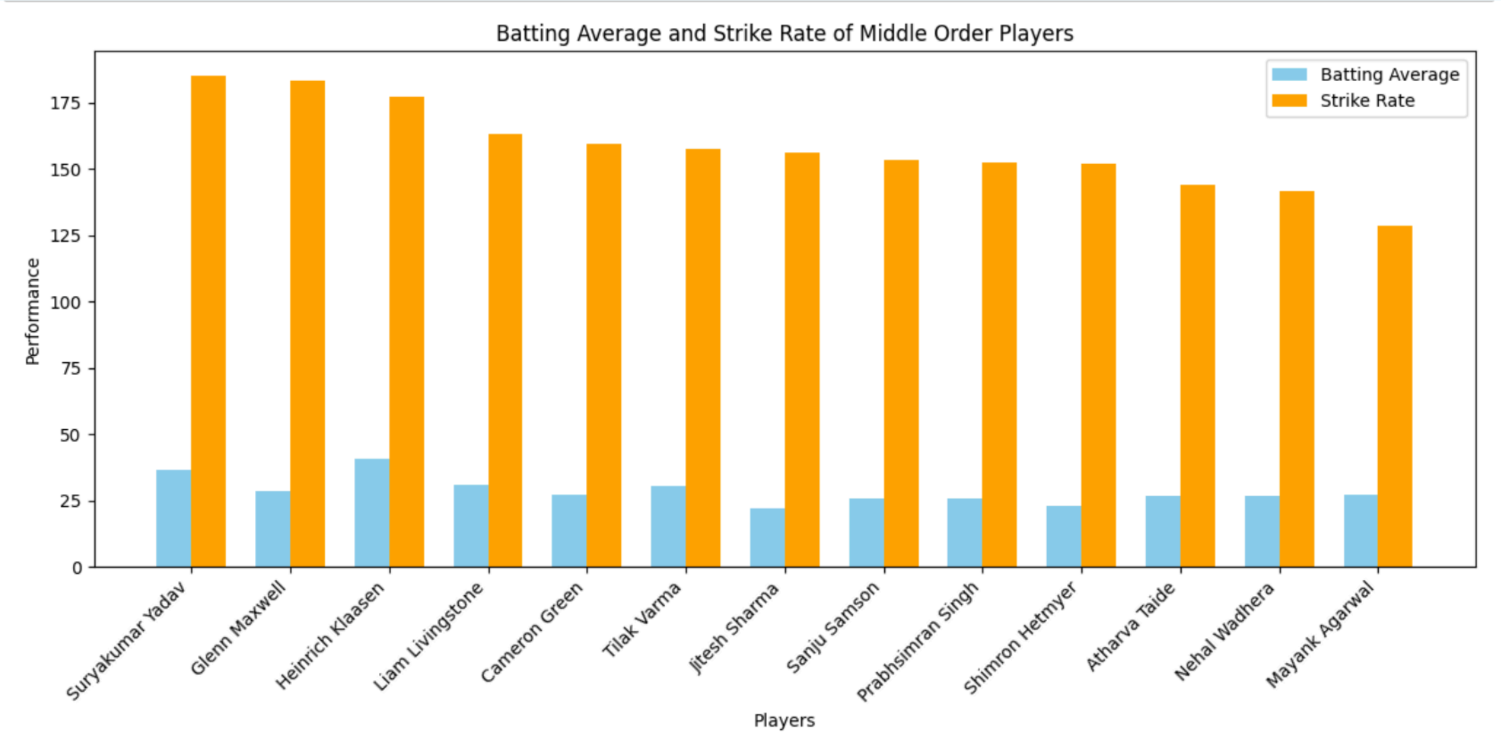
```
sql2 = """
select a.Player_ID,a.Player_Name,a.Batting_order,sum(b.Run_Scored)/count(b.Match_Num) as Batting_avg,
(sum(b.Run_Scored)/sum(b.Balls_Faced))*100 as strike_rate
from Player_database.performances b
left join Player_database.players a
on b.Player_ID = a.Player_ID
group by 1,2
having count(match_num) >= 7 and strike_rate >= 110 and Batting_avg >= 22 and Batting_order = 'Middle'
order by 5 desc
;"""

result_df2 = pd.read_sql_query(sql2, connection)
result_df2
```

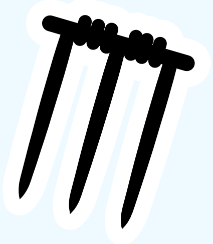
OUTPUT



	Player_ID	Player_Name	Batting_order	Batting_avg	strike_rate
0	1122	Suryakumar Yadav	Middle	36.5000	185.1449
1	1190	Glenn Maxwell	Middle	28.5714	183.4862
2	1226	Heinrich Klaasen	Middle	40.7273	177.0751
3	1153	Liam Livingstone	Middle	31.0000	163.1579
4	1135	Cameron Green	Middle	27.2143	159.4142
5	1125	Tilak Varma	Middle	30.4444	157.4713
6	1151	Jitesh Sharma	Middle	22.0714	156.0606
7	1165	Sanju Samson	Middle	25.8571	153.3898
8	1150	Prabhsimran Singh	Middle	25.5714	152.3404
9	1170	Shimron Hetmyer	Middle	23.0000	151.7766
10	1157	Atharva Taide	Middle	26.5714	144.1860
11	1141	Nehal Wadhera	Middle	26.7500	141.7219
12	1225	Mayank Agarwal	Middle	27.0000	128.5714

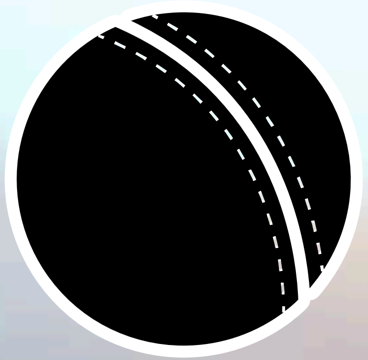
QUERY 2 VISUALIZATION



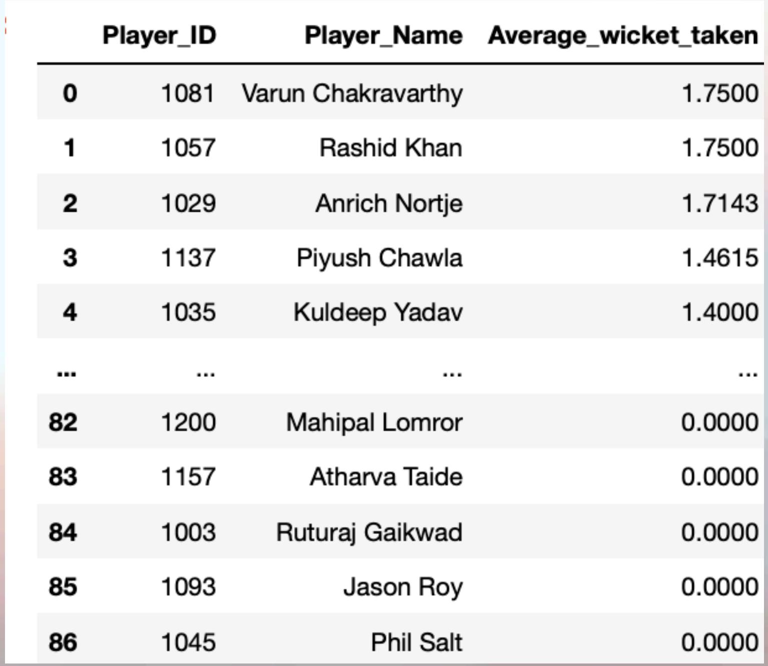


SQL QUERY 3

```
sql4 = """
select a.Player_ID,a.Player_Name,
sum(b.Wickets_Taken)/count(b.Match_Num) as Average_wicket_taken
from Player_database.performances b
left join Player_database.players a
on b.Player_ID = a.Player_ID
group by 1,2
having count(match_num) >= 7
order by 3 desc
;"""

result_df4 = pd.read_sql_query(sql4, connection)
result_df4
```

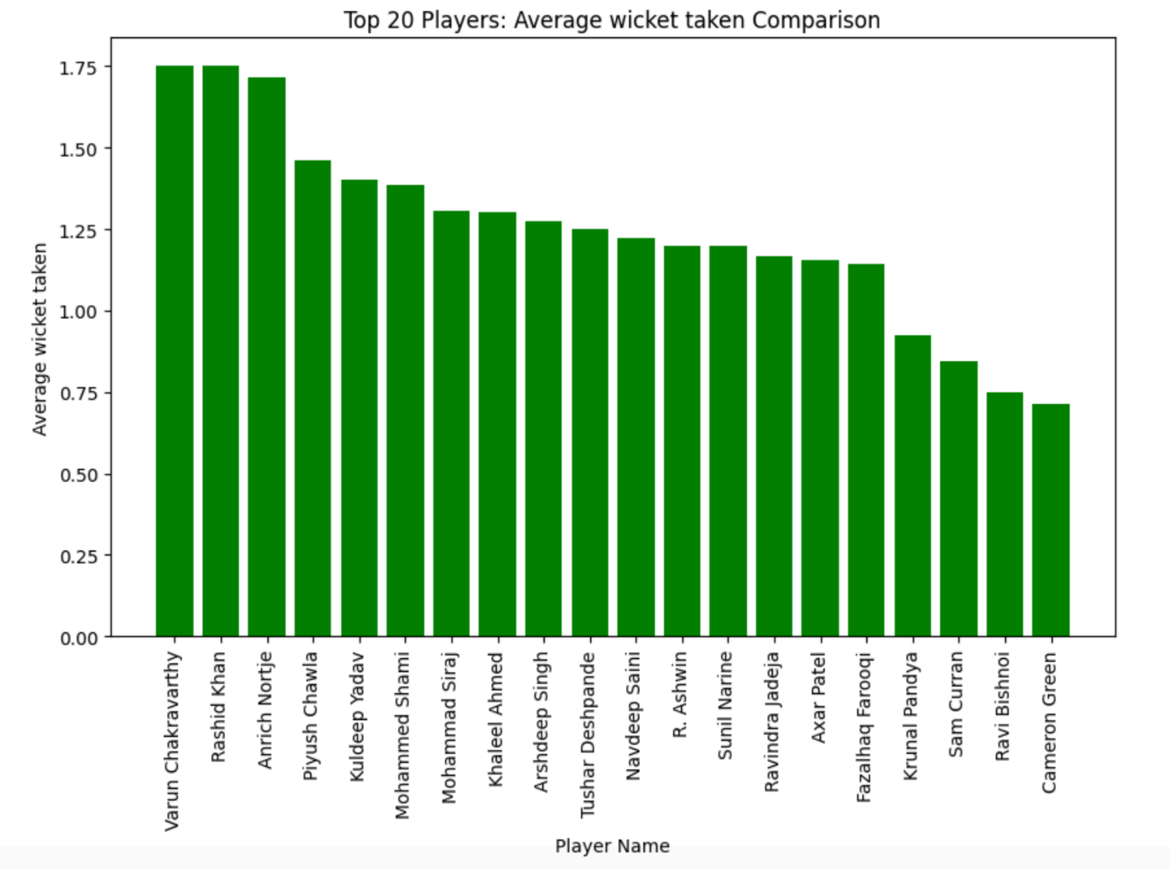


OUTPUT



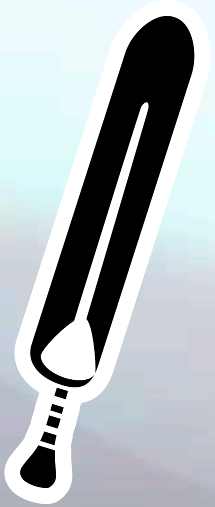
	Player_ID	Player_Name	Average_wicket_taken
0	1081	Varun Chakravarthy	1.7500
1	1057	Rashid Khan	1.7500
2	1029	Anrich Nortje	1.7143
3	1137	Piyush Chawla	1.4615
4	1035	Kuldeep Yadav	1.4000
...	...	...	...
82	1200	Mahipal Lomror	0.0000
83	1157	Atharva Taide	0.0000
84	1003	Ruturaj Gaikwad	0.0000
85	1093	Jason Roy	0.0000
86	1045	Phil Salt	0.0000

QUERY 3 VISUALIZATION

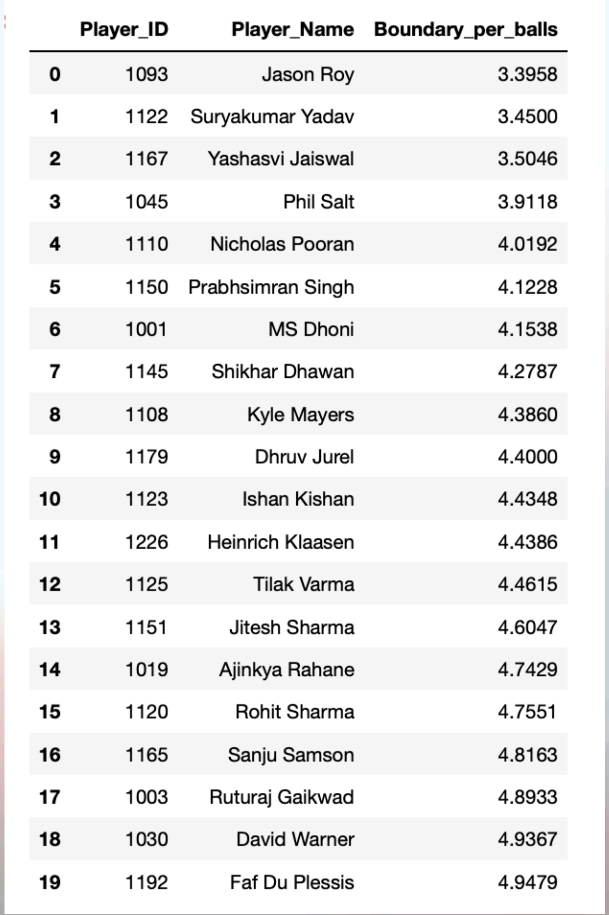


SQL QUERY 4

```
sql6 = """
select a.Player_ID,a.Player_Name,
sum(b.balls_faced)/sum(b.no_of_boundaries) as Boundary_per_balls
from Player_database.performances b
left join Player_database.batsmen a
on b.Player_ID = a.Player_ID
group by 1,2
having count(match_num) >= 7
order by 3
limit 20
;"""

result_df6 = pd.read_sql_query(sql6, connection)
result_df6
```

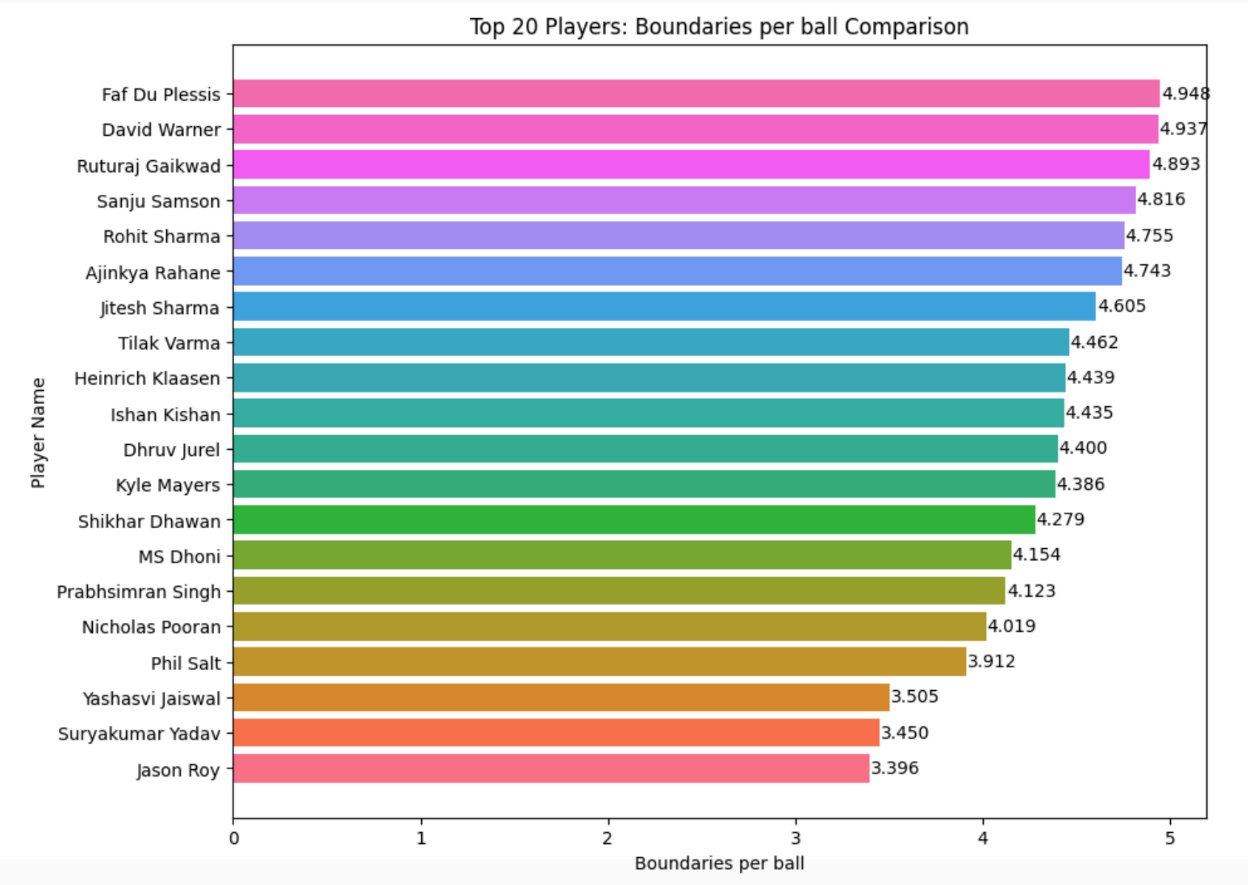


OUTPUT

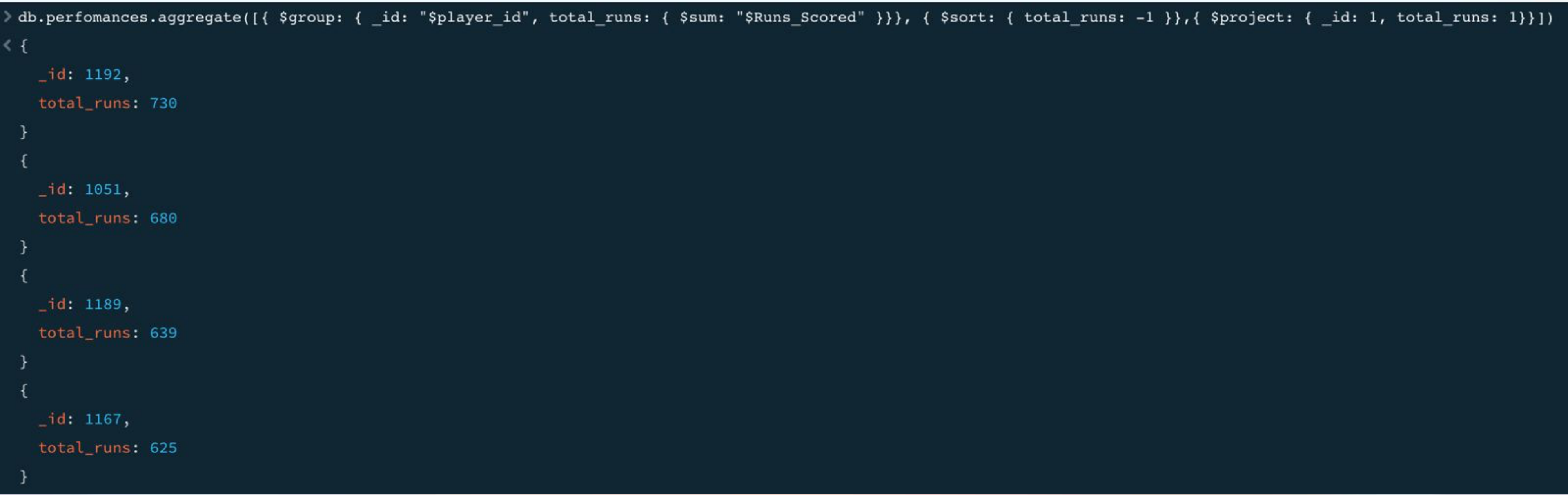


	Player_ID	Player_Name	Boundary_per_balls
0	1093	Jason Roy	3.3958
1	1122	Suryakumar Yadav	3.4500
2	1167	Yashasvi Jaiswal	3.5046
3	1045	Phil Salt	3.9118
4	1110	Nicholas Pooran	4.0192
5	1150	Prabhsimran Singh	4.1228
6	1001	MS Dhoni	4.1538
7	1145	Shikhar Dhawan	4.2787
8	1108	Kyle Mayers	4.3860
9	1179	Dhruv Jurel	4.4000
10	1123	Ishan Kishan	4.4348
11	1226	Heinrich Klaasen	4.4386
12	1125	Tilak Varma	4.4615
13	1151	Jitesh Sharma	4.6047
14	1019	Ajinkya Rahane	4.7429
15	1120	Rohit Sharma	4.7551
16	1165	Sanju Samson	4.8163
17	1003	Ruturaj Gaikwad	4.8933
18	1030	David Warner	4.9367
19	1192	Faf Du Plessis	4.9479

QUERY 4 VISUALIZATION

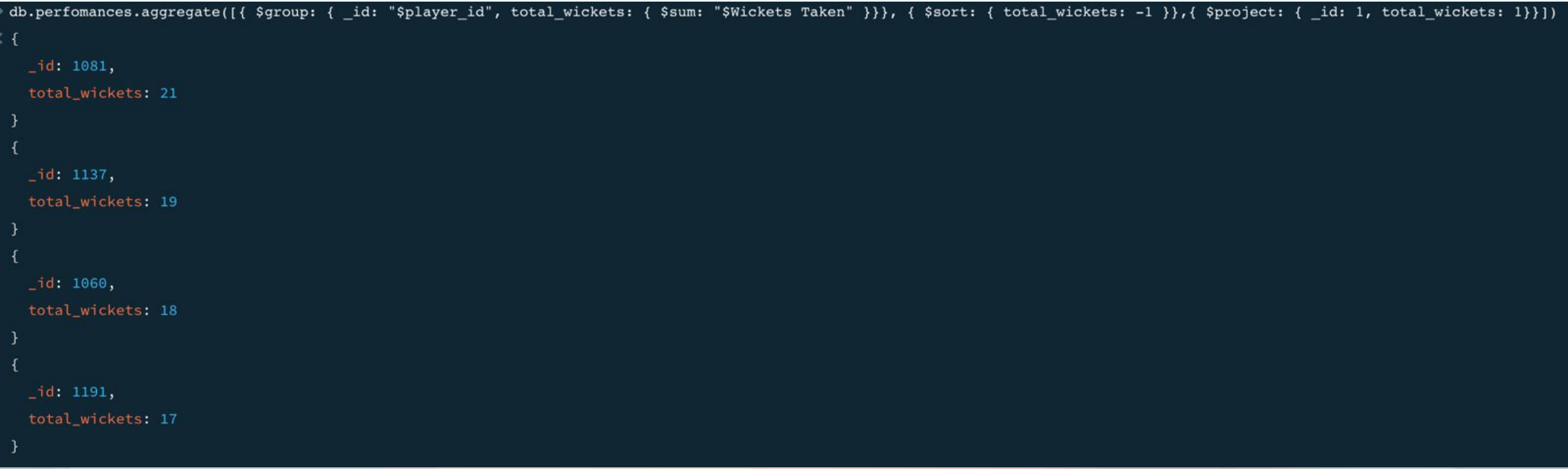


NO SQL QUERY 1



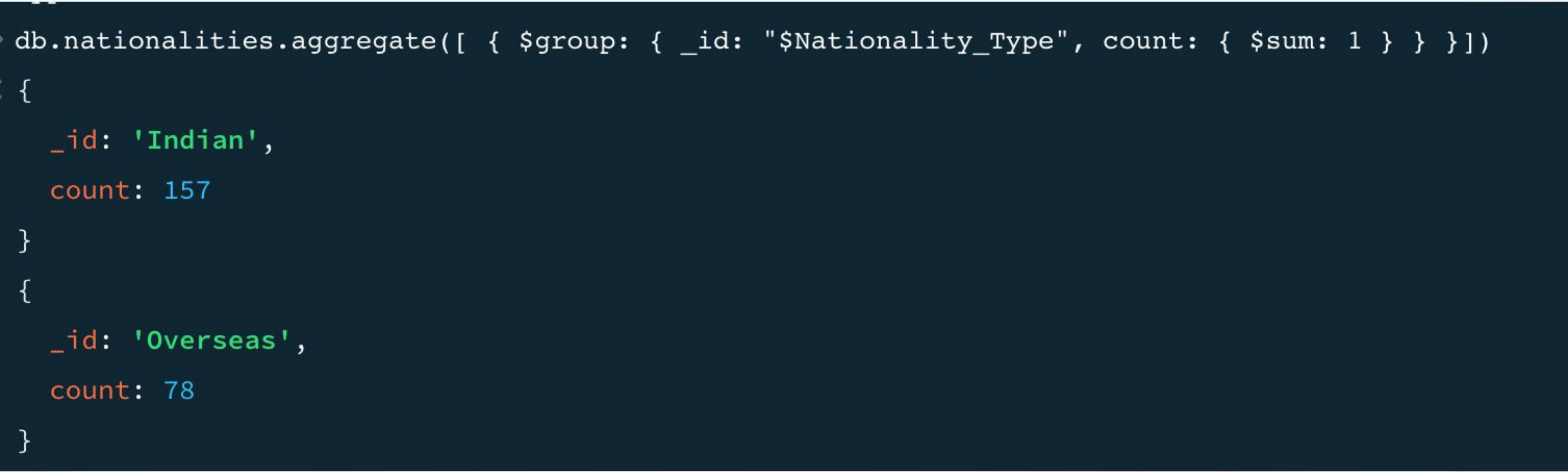
```
> db.perfomances.aggregate([{ $group: { _id: "$player_id", total_runs: { $sum: "$Runs_Scored" }}}, { $sort: { total_runs: -1 }},{ $project: { _id: 1, total_runs: 1}}])
< {
    _id: 1192,
    total_runs: 730
  }
  {
    _id: 1051,
    total_runs: 680
  }
  {
    _id: 1189,
    total_runs: 639
  }
  {
    _id: 1167,
    total_runs: 625
  }
```

NO SQL QUERY 2

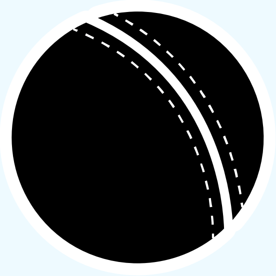


```
db.perfomances.aggregate([{ $group: { _id: "$player_id", total_wickets: { $sum: "$Wickets Taken" }}}, { $sort: { total_wickets: -1 }},{ $project: { _id: 1, total_wickets: 1}}])
{
  _id: 1081,
  total_wickets: 21
}
{
  _id: 1137,
  total_wickets: 19
}
{
  _id: 1060,
  total_wickets: 18
}
{
  _id: 1191,
  total_wickets: 17
}
```

NO SQL QUERY 3



```
db.nationalities.aggregate([ { $group: { _id: "$Nationality_Type", count: { $sum: 1 } } }])
{
  _id: 'Indian',
  count: 157
}
{
  _id: 'Overseas',
  count: 78
}
```



THANK YOU