**Players Analysis For IPL Auction**

**Project Overview:**

This project involved analyzing the Indian Premier League (IPL) dataset to assist stakeholders in identifying key players to target for the upcoming IPL auction. Using SQL, we performed data extraction, transformation, and analysis to provide actionable insights for player selection.

**Objectives:**

1. To analyze historical IPL data to identify key players to target for the upcoming auction.
2. To help stakeholders make informed decisions based on comprehensive data analysis.
3. To create a set of criteria and recommendations for selecting players based on their performance metrics.

**Methodology:**

1. Data collection:

* Dataset : The IPL dataset used includes historical records of all matches played between teams, featuring detailed ball-by-ball data.
* Source: Data was sourced from Kaggle.

1. SQL Queries and Data Manipulation Techniques Used:
   1. Data Retrieval and Basic Queries:

* Objective: Extract raw data and essential statistics from the dataset.
* Technique: Utilized SELECT statements to pull specific fields such as player names, match dates, and performance metrics.
  1. Aggregation and Summarization:
* Objective**:** Summarize player performance metrics for comprehensive analysis.
* Technique**:** Employed aggregation functions such as SUM() and COUNT() to calculate total runs, wickets, and other cumulative statistics.
  1. Joining Tables:
* Objective**:** Combine related data from multiple tables for integrated insights.
* Technique**:** Used JOIN operations to merge different cte’s having player statistics.
  1. Subqueries for Comparative Analysis:
* Objective**:** Perform comparative analysis by isolating and evaluating subsets of data.
* Technique**:** Applied subqueries to filter and compare player performance across different conditions.
  1. Window Functions for Ranking:
* Objective: Rank players based on performance metrics to identify top performers.
* Technique: Implemented window functions such as RANK() , MAX() etc. to assign ranks based on aggregated performance data.

1. **Top 10 batsman with average more than 30 and Strike Rate greater than 130 (Player Should have played minimum 10 matches)**

Query:- with no\_of\_matches as (

select batsman,COUNT(distinct id) as no\_of\_matches

from ballsnew

group by batsman),

batting\_stats as

(select ballsnew.batsman,

SUM(batsman\_runs) as Total\_runs ,SUM(is\_wicket) as No\_of\_wickets,

COUNT(\*) as No\_of\_balls

from ballsnew

join no\_of\_matches

on ballsnew.batsman = no\_of\_matches.batsman

where no\_of\_matches>10

group by ballsnew.batsman),

avg\_sr as (

select batting\_stats.batsman,

case when No\_of\_wickets = 0 then Total\_runs

else round(cast(Total\_runs as float)/cast( No\_of\_wickets as float),2) end as Average,

round(cast (Total\_runs as float)/(cast (No\_of\_balls as float))\*100,2) as StrikeRate

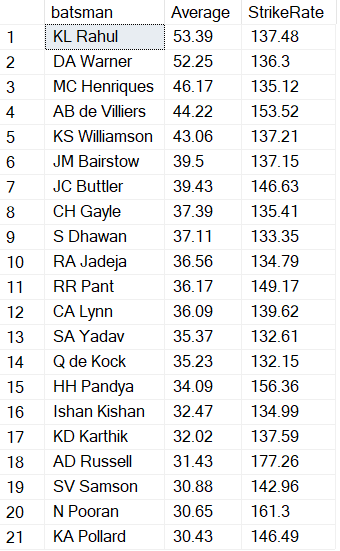
from batting\_stats)

select \*

from avg\_sr

where Average>=30 and StrikeRate>=130

**O/P:-**

****

1. **Top batsman with average more than 30 and Strike Rate greater than 140 in power plays (overs less than 6) (Player Should have played minimum 10 matches)**

Query:-

with no\_of\_matches as (

select batsman,COUNT(distinct id) as no\_of\_matches

from ballsnew

group by batsman),

batting\_stats as

(select ballsnew.batsman,

SUM(batsman\_runs) as Total\_runs ,SUM(is\_wicket) as No\_of\_wickets,

COUNT(\*) as No\_of\_balls

from ballsnew

join no\_of\_matches

on ballsnew.batsman = no\_of\_matches.batsman

where no\_of\_matches>=10 and overs<6

group by ballsnew.batsman),

avg\_sr as (

select batting\_stats.batsman,

case when No\_of\_wickets = 0 then Total\_runs

else round(cast(Total\_runs as float)/cast( No\_of\_wickets as float),2) end as Average,

round(cast (Total\_runs as float)/(cast (No\_of\_balls as float))\*100,2) as StrikeRate

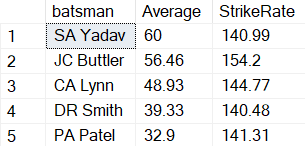
from batting\_stats)

select \*

from avg\_sr

where Average>=30 and StrikeRate>=140

order by Average desc, StrikeRate desc



1. **Top batsman with average more than 30 and Strike Rate greater than 180 in death overs (overs greater than 15) (Player Should have played minimum 10 matches)**

Query:-

with no\_of\_matches as (

select batsman,COUNT(distinct id) as no\_of\_matches

from ballsnew

group by batsman),

batting\_stats as

(select ballsnew.batsman,

SUM(batsman\_runs) as Total\_runs ,SUM(is\_wicket) as No\_of\_wickets,

COUNT(\*) as No\_of\_balls

from ballsnew

join no\_of\_matches

on ballsnew.batsman = no\_of\_matches.batsman

where no\_of\_matches>=10 and overs>14

group by ballsnew.batsman),

avg\_sr as (

select batting\_stats.batsman,

case when No\_of\_wickets = 0 then Total\_runs

else round(cast(Total\_runs as float)/cast( No\_of\_wickets as float),2) end as Average,

round(cast (Total\_runs as float)/(cast (No\_of\_balls as float))\*100,2) as StrikeRate

from batting\_stats)

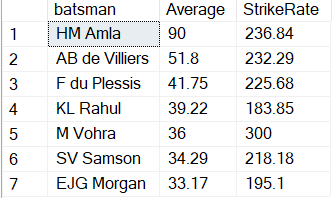
select \*

from avg\_sr

where Average>=30 and StrikeRate>=180

order by Average desc, StrikeRate desc

O/P:-



1. **Batsman who have scored 50s consecutively in more than 2 matches.**

**Query:-**

with cte as (

select id,batsman,SUM(batsman\_runs) as runs

from ballsnew

group by id, batsman),

cte2 as(

select \*,

ROW\_NUMBER()over(partition by batsman order by id) as rnk

from cte),

cte3 as (

select \*,rnk-

ROW\_NUMBER()over(partition by batsman order by id) as con

from cte2

where runs >=50),

cte4 as(

select \*,

count(\*) over (partition by batsman, con) as no\_

from cte3)

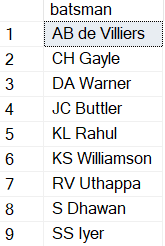
select distinct batsman

from cte4

where no\_ >=3

group by batsman,no\_

order by batsman



1. **Top 10 economical bowlers.**

**Query:-**

select top 10 bowler,cast(SUM(runs\_con)\*1.0/sum(total\_overs) as decimal(4,2)) as economy

from(

select id,bowler,SUM(total\_runs) as runs\_con ,COUNT(distinct(overs)) as total\_overs,count(distinct(id)) as matches

from ballsnew

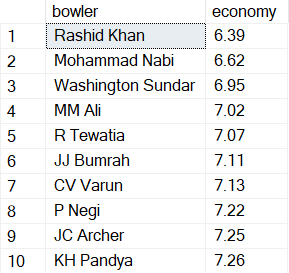
group by id, bowler)a

group by bowler

having SUM(matches)>10

order by economy

**O/P:-**

****

1. **Top 10 economical bowlers in death overs (Overs greater than 14).**

**Query:-**

select Top 10 bowler,cast(SUM(runs\_con)\*1.0/sum(total\_overs) as decimal(4,2)) as economy

from(

select id,bowler,SUM(total\_runs) as runs\_con ,COUNT(distinct(overs)) as total\_overs,count(distinct(id)) as matches

from ballsnew

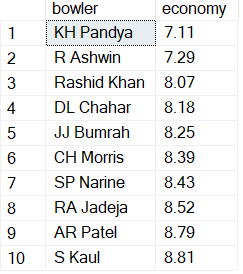
where overs>14

group by id, bowler)a

group by bowler

having SUM(matches)>10

order by economy



1. **Top 10 economical bowlers in power play overs (Overs less than 7).**

**Query:-**

select top 10 bowler,cast(SUM(runs\_con)\*1.0/sum(total\_overs) as decimal(4,2)) as economy

from(

select id,bowler,SUM(total\_runs) as runs\_con ,COUNT(distinct(overs)) as total\_overs,count(distinct(id)) as matches

from ballsnew

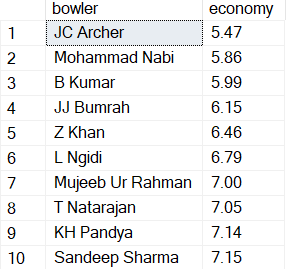
where overs<6

group by id, bowler)a

group by bowler

having SUM(matches)>10

order by economy



1. **List of players wo have taken hatricks.**

**Query:-**

with cte as(

select id,bowler,overs,ball-

ROW\_NUMBER()over(partition by bowler,id,overs order by ball) as wickets

from ballsnew

where is\_wicket =1)

,cte2 as(

select \*,

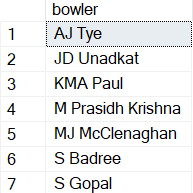
COUNT(\*)over ( partition by id,bowler, overs , wickets order by overs) as no\_of\_con\_wickets

from cte)

select bowler from cte2

where no\_of\_con\_wickets >2

group by bowler



1. **What is the least score defended by a bowler and his name?**

**Query:-**

with cte as(

select id,inning,max(overs) as last\_over from ballsnew

where inning =2

group by id,inning),

cte2 as(

select ballsnew.id,ballsnew.bowler,SUM(ballsnew.total\_runs)as total\_runs\_spend from ballsnew

join cte

on ballsnew.id=cte.id and ballsnew.inning=cte.inning and ballsnew.overs=cte.last\_over

group by ballsnew.id,ballsnew.bowler)

select TOP 1 bowler,total\_runs\_spend+result\_margin AS runs\_defended from cte2

join matchesnew

on cte2.id=matchesnew.id

where result = 'runs' and total\_runs\_spend+result\_margin<20

order by runs\_defended



1. **How many times less than 20 runs is defended by any bowler?**

**Query:-**

with cte as(

select id,inning,max(overs) as last\_over from ballsnew

where inning =2

group by id,inning),

cte2 as(

select ballsnew.id,ballsnew.bowler,SUM(ballsnew.total\_runs)as total\_runs\_spend from ballsnew

join cte

on ballsnew.id=cte.id and ballsnew.inning=cte.inning and ballsnew.overs=cte.last\_over

group by ballsnew.id,ballsnew.bowler)

select bowler,count(\*) as no\_of\_times from cte2

join matchesnew

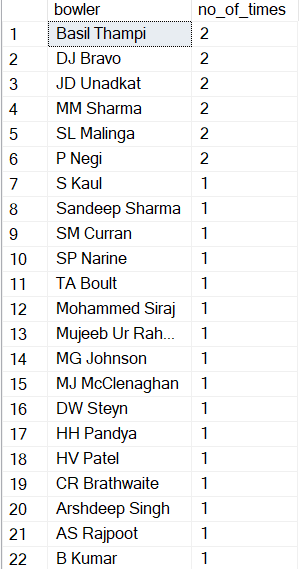
on cte2.id=matchesnew.id

where result = 'runs' and total\_runs\_spend+result\_margin<=20

group by bowler

order by no\_of\_times desc

**O/P:-**

****

1. **Best Allrounder. Player who has economy less than 8 and strike rate > 140 (Should have played atleast 10 matches)**

**Query:-**

with player\_batting\_stats as(

select batsman,cast(sum(batsman\_runs)\*1.0/sum(is\_wicket)as decimal(4,2)) as Average,cast(sum(batsman\_runs)\*1.0/count(\*)\*100 as decimal(6,2))as Strike\_rate, count(distinct(id)) as No\_of\_matches

from ballsnew

group by batsman

having sum(is\_wicket)>0),

bowling\_stats as(select bowler,cast(SUM(runs\_con)\*1.0/sum(total\_overs) as decimal(4,2)) as economy,SUM(matches) as no\_of\_matches

from(

select id,bowler,SUM(total\_runs) as runs\_con ,COUNT(distinct(overs)) as total\_overs,count(distinct(id)) as matches

from ballsnew

group by id, bowler)a

group by bowler)

,cte3 as(

select bowling\_stats.bowler,player\_batting\_stats.Average ,player\_batting\_stats.Strike\_rate,player\_batting\_stats.No\_of\_matches as Batting\_matches,

bowling\_stats.economy,bowling\_stats.no\_of\_matches as bowling\_matches

from player\_batting\_stats

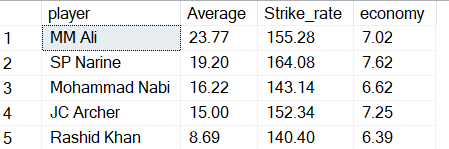
join bowling\_stats

on player\_batting\_stats.batsman= bowling\_stats.bowler)

select bowler as player, Average, Strike\_rate, economy from cte3

where economy<8 and Batting\_matches >=10 and bowling\_matches>=10 and Strike\_rate>= 140

order by Average desc, Strike\_rate desc

**O/P:- **