**Team-3**

**Project Description**

**Provide a comprehensive and detailed explanation of all the implementation steps. Your document should be clear enough that anyone reading it can successfully implement the project without encountering any difficulties**.

**Project Description**:

"In this project, we embark on an in-depth exploration of IPL (Indian Premier League) cricket data spanning the years 2008 to 2019, using a combination of PySpark, Matplotlib, and data visualization techniques. Our primary goals are to assess team performance, determine season-wise rankings, analyze extra runs conceded by bowling teams, and recognize the top-performing batsmen in the IPL.

We begin by calculating the total runs scored by each team over the years, offering insights into the most consistent run-scorers in the league. We also identify the top team in each season, as determined by the points table, shedding light on the season's standout performers. Moreover, we analyze the extra runs conceded by bowling teams, pinpointing areas of improvement in terms of reducing extras. Finally, we showcase the top 5 batsmen with the most 'Player of the Match' awards, celebrating the individual brilliance that has graced IPL cricket over the years.

This project aims to deliver a comprehensive and data-driven perspective on IPL cricket, providing valuable insights and narratives through robust data analysis and engaging visualizations. It offers a captivating journey into the world of IPL cricket, revealing patterns, trends, and standout performers, both at the team and individual levels."

**Outcome Results Summary**

**A detailed discussion of the results achieved for each goal, which may include providing clear screenshots or snippets of your source code. Consider including metrics (data quality, 5Vs, latency, processing time, resource utilization, security, cost) in your discussion.**

**Here's a brief summary of the expected outcomes for each of your project goals:**

**First we have to create spark session**

from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()

**load the both csv files data into the note book for analysis**

**deliveries.csv**

ipl=spark.read.format('csv').option('header','true').load('deliveries.csv')

ipl.createOrReplaceTempView('game\_stats')

ipl.show()

**Matches.csv**

iplMatches=spark.read.format('csv').option('header','true').load('matches.csv')

iplMatches.createOrReplaceTempView('game\_stats')

iplMatches.show()

Here we use Matplotlib for visualization

**1. Team Performance - Total Runs Scored:**

we will calculate and present the total runs scored by each IPL team over the years. This will provide insights into the run-scoring performance of each team throughout the IPL history.

from pyspark.sql import SparkSession

from pyspark.sql.functions import count, col

# Create a Spark session

spark = SparkSession.builder.appName("TopWicketTakers").getOrCreate()

# Assuming 'cricket\_data' is your DataFrame with the specified columns

# If your DataFrame has a different name or structure, adjust the column names accordingly.

# Filter the DataFrame to include only dismissal records (wickets taken)

dismissal\_records = ipl.filter(col("dismissal\_kind").isNotNull())

# Group the data by 'bowler' and 'bowling\_team' and count the wickets taken by each bowler

wickets\_count = dismissal\_records.groupBy("bowler", "bowling\_team").agg(count("\*").alias("wickets"))

# Order the result in descending order by wickets taken

wickets\_count = wickets\_count.orderBy("wickets", ascending=False)

# Show the top 10 wicket-takers with their respective teams

wickets\_count.show(10)

A screenshot of a computer

Description automatically generated

The below is the visualization

**2. Player Performance - Top Run-Scorer and Top Wicket-Taker:**

we will identify and showcase the top run-scorer and the top wicket-taker in IPL history. This will highlight the most prolific batsman and bowler in the league.

**#code snippet for top wicket taker**

import seaborn as sns

import matplotlib.pyplot as plt

# Assuming 'wickets\_count' is your DataFrame with 'bowler' and 'wickets'

# If your DataFrame has a different name or structure, adjust the column names accordingly.

# Sort the data in descending order by wickets taken

wickets\_count = wickets\_count.orderBy("wickets", ascending=False).limit(10)

# Create a horizontal bar chart

plt.figure(figsize=(10, 6))

sns.barplot(x="wickets", y="bowler", data=wickets\_count.toPandas(), palette="Set3")

plt.xlabel("Wickets Taken")

plt.ylabel("Bowler")

plt.title("Top 10 Wicket-Takers")

# Display the bar chart

plt.show()

A screenshot of a computer

Description automatically generated

**#code snippet for top Run scorer**

from pyspark.sql import SparkSession

from pyspark.sql.functions import sum, desc

# Initialize a Spark session

spark = SparkSession.builder.appName("TopRunScorersWithTeam").getOrCreate()

# Assuming you have a DataFrame named 'matches' with the specified columns

# You can replace 'matches' with the actual name of your DataFrame

# Group by 'batsman' and 'batting\_team', and calculate the total runs scored by each batsman

top\_run\_scorers = ipl.groupBy("batsman", "batting\_team").agg(sum("batsman\_runs").alias("total\_runs"))

# Order the results by 'total\_runs' in descending order

top\_run\_scorers = top\_run\_scorers.orderBy(desc("total\_runs"))

# Show the top 10 run scorers with their team names

top10\_run\_scorers = top\_run\_scorers.limit(10)

top10\_run\_scorers.show()

A screenshot of a computer

Description automatically generated

**3. Toss Impact - Impact of Winning the Toss:**

we will investigate the impact of winning the toss on match outcomes. This analysis will help determine whether teams that win the toss have a statistically higher likelihood of winning the match.

A screenshot of a computer

Description automatically generated

**4. Rank - Top Team in Each Season:**

we will determine and display the top team in the points table for each IPL season. This will offer insights into the season-wise performance of IPL teams.

**#code snippet**

from pyspark.sql import SparkSession

from pyspark.sql.functions import count, desc, rank # Add this import statement

from pyspark.sql.window import Window

# Initialize a Spark session

spark = SparkSession.builder.appName("SeasonTopWinningTeams").getOrCreate()

# Load the "matches.csv" dataset into a PySpark DataFrame

iplMatches = spark.read.csv("matches.csv", header=True, inferSchema=True)

# Group and aggregate the data to find the season-wise top-winning teams

season\_top\_winners = iplMatches.groupBy("Season", "winner").agg(count("winner").alias("wins"))

window\_spec = Window.partitionBy("Season").orderBy(desc("wins"))

season\_top\_winners = season\_top\_winners.withColumn("rank", rank().over(window\_spec))

top\_winners = season\_top\_winners.filter(col("rank") == 1).select("Season", "winner", "wins").orderBy("Season")

top\_winners.show()

A screenshot of a computer

Description automatically generatedanna

**5. Extra Runs - Total Extra Runs Given:**

we will calculate and analyze the total extra runs conceded by each bowling team over the years. This analysis will help identify which teams have struggled with extras and may suggest areas for improvement.

from pyspark.sql import SparkSession

from pyspark.sql.functions import sum

# Create a Spark session

spark = SparkSession.builder.appName("ExtraRunsAnalysis").getOrCreate()

# Assuming 'ipl' is your DataFrame with the specified columns

# If your DataFrame has a different name or structure, adjust the column names accordingly.

# Group the data by 'batting\_team' and sum the 'extra\_runs' to calculate the total extra runs for each team

extra\_runs\_by\_team = ipl.groupBy('batting\_team').agg(sum('extra\_runs').alias('total\_extra\_runs'))

# Order the result in descending order based on total extra runs

extra\_runs\_by\_team = extra\_runs\_by\_team.orderBy('total\_extra\_runs', ascending=False)

# Show the result

extra\_runs\_by\_team.show()

A screenshot of a computer

Description automatically generated

**6. Player of the Match - Top Batsmen:**

we will showcase the top 5 batsmen with the most "Player of the Match" awards, recognizing and celebrating their standout performances in IPL matches**.**

Your project aims to deliver comprehensive insights into team and player performance, match outcomes related to winning the toss, and areas for improvement in terms of extra runs conceded. These goals will provide a rich and data-driven exploration of IPL cricket history.

from pyspark.sql import SparkSession

from pyspark.sql.functions import count, desc

# Initialize a Spark session

spark = SparkSession.builder.appName("TopBatsmenPlayerOfTheMatch").getOrCreate()

# Assuming you have a DataFrame named 'matches' with the specified columns

# You can replace 'matches' with the actual name of your DataFrame

# Group by 'player\_of\_match' and calculate the number of times each player received the award

top\_batsmen\_player\_of\_match = iplMatches.groupBy("player\_of\_match").agg(count("\*").alias("awards"))

# Order the results by 'awards' in descending order

top\_batsmen\_player\_of\_match = top\_batsmen\_player\_of\_match.orderBy(desc("awards"))

# Show the top 5 batsmen with the most "Player of the Match" awards

top5\_batsmen\_player\_of\_match = top\_batsmen\_player\_of\_match.limit(10)

top5\_batsmen\_player\_of\_match.show()

A screenshot of a computer

Description automatically generated

**OutcomeConclusion**

**A conclusion to your project.**

**Conclusion:**

In this ambitious IPL (Indian Premier League) cricket data analysis project, we delved deep into the world of cricket, leveraging the power of PySpark and data analysis tools to extract a wealth of insights. Our primary goals, spanning team and player performance, the impact of winning the toss, season-wise rankings, extra runs analysis, and recognition of standout batsmen, have provided valuable revelations. We have successfully calculated and ranked total runs scored by each team, celebrated the top run-scorer and wicket-taker in IPL history, and investigated the dynamic impact of winning the toss. Our analysis revealed the champions of each season and showcased areas for improvement in reducing extra runs. Moreover, we paid tribute to the exceptional batsmen who have consistently shone in IPL matches. This project underscores the prowess of data analytics in unraveling the narratives within the IPL dataset, offering a deeper understanding of cricket's nuances. While our project goals are met, the IPL dataset remains an unexplored treasure trove of insights, promising continued adventures in the world of data-driven cricket analysis. The project not only serves cricket enthusiasts and teams but also underscores the enduring power of data-driven decision-making in sports.

**7.Citations**

Provide all the necessary citations for the sources you utilized to finish this project successfully. Save all your work to your GitHub repo and provide the URL

* Data Source: <https://www.kaggle.com/datasets/ramjidoolla/ipl-data-set?select=deliveries.csv>

We are using blow technologies

1. Jupyter Notebook
2. Pyspark
3. Matplot library--- for visualization

The below is the jupyter Notebook link.

The below is the GitHub link to view data sets and code Snippets