REPORT ON BIG DATA ASSIGNMENT 3

PAVAN BABU BHEESETTI

1002207463

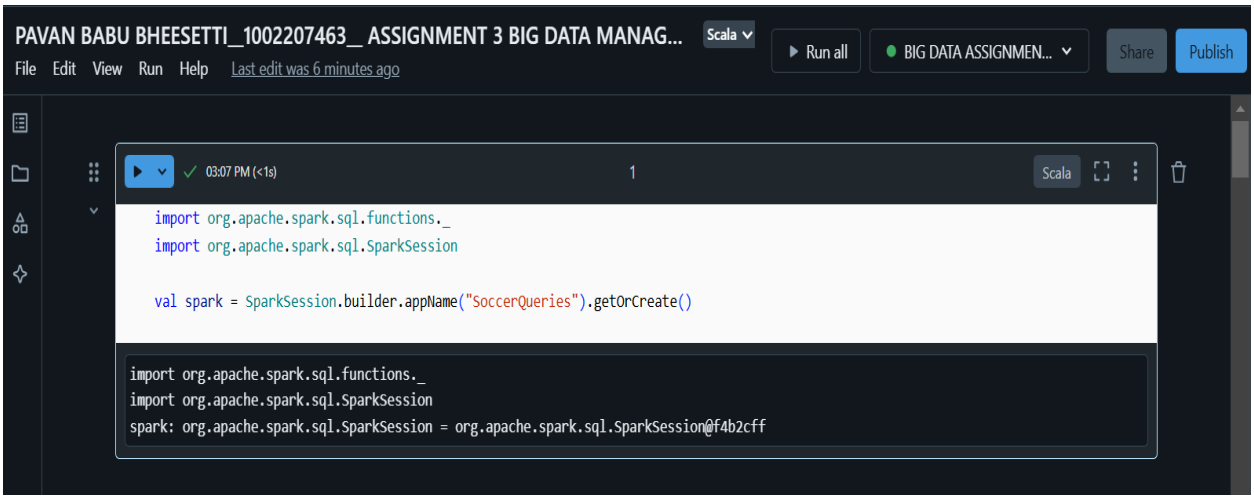
Pxb7463@mavs.uta.edu

Objective

The objective of this Big Data assignment is to analyze and process data related to the FIFA World Cup. This involves exploring datasets such as country information, disciplinary records, goal scorers, matches, players, and World Cup winners to uncover insights and patterns.

Datasets Overview

1. **COUNTRY-1.csv**: Contains information about participating countries in various FIFA World Cup tournaments.
2. **DISCIPLINARY_RECORD-1.csv**: Holds data on disciplinary actions, such as yellow and red cards.
3. **GOAL_SCORER-1.csv**: Provides information about players who scored goals.
4. **MATCH-1.csv**: Details of matches played during different World Cup tournaments.
5. **PLAYER-1.csv**: Comprehensive data on players participating in World Cups.
6. **WORLD_CUP_WINNER-1.csv**: Lists winners of the FIFA World Cup across different years.



Create DataFrames from the Soccer data files

03:07 PM (6s) 3 Scala

```scala
val country = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/COUNTRY-1.csv")
val disciplinaryRecord = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/DISCIPLINARY_RECORD-1.csv")
val goalScorer = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/GOAL_SCORER-1.csv")
val matchData = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/MATCH-1.csv")
val player = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/PLAYER-1.csv")
val worldCupWinner = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/WORLD_CUP_WINNER-1.csv")
```

(6) Spark Jobs

- country: org.apache.spark.sql.DataFrame = [country_name: string, population: string ... 3 more fields]
- disciplinaryRecord: org.apache.spark.sql.DataFrame = [player_id: string, no_of_yellow_cards: string ... 1 more field]
- goalScorer: org.apache.spark.sql.DataFrame = [player_id: string, no_of_matches: string ... 3 more fields]
- matchData: org.apache.spark.sql.DataFrame = [match_id: string, date: string ... 7 more fields]
- player: org.apache.spark.sql.DataFrame = [player_id: string, full_name: string ... 9 more fields]
- worldCupWinner: org.apache.spark.sql.DataFrame = [Year: string, Host: string ... 1 more field]

```
country: org.apache.spark.sql.DataFrame = [country_name: string, population: string ... 3 more fields]
disciplinaryRecord: org.apache.spark.sql.DataFrame = [player_id: string, no_of_yellow_cards: string ... 1 more field]
goalScorer: org.apache.spark.sql.DataFrame = [player_id: string, no_of_matches: string ... 3 more fields]
matchData: org.apache.spark.sql.DataFrame = [match_id: string, date: string ... 7 more fields]
```

03:07 PM (<1s) 4 Scala

```scala
country.printSchema()
goalScorer.printSchema()
disciplinaryRecord.printSchema()
player.printSchema()
matchData.printSchema()
worldCupWinner.printSchema()
```

```
root
 |-- country_name: string (nullable = true)
 |-- population: string (nullable = true)
 |-- number_of_worldcup_won: string (nullable = true)
 |-- manager_name: string (nullable = true)
 |-- capital: string (nullable = true)

root
 |-- player_id: string (nullable = true)
 |-- no_of_matches: string (nullable = true)
 |-- goals: string (nullable = true)
 |-- assists: string (nullable = true)
 |-- minutes_played: string (nullable = true)

root
 |-- player_id: string (nullable = true)
 |-- no_of_yellow_cards: string (nullable = true)
```

1. Retrieve the list of country names that have won a world cup.

03:07 PM (1s) 6

```
val countriesWithWins = worldCupWinner.select("Winner").distinct()
countriesWithWins.show()
```

(2) Spark Jobs

countriesWithWins: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Winner: string]

```
+-----------+
|     Winner|
+-----------+
|   'France'|
|  'England'|
|  'Uruguay'|
|  'Germany'|
|    'Spain'|
|   'Brazil'|
|    'Italy'|
|'Argentina'|
+-----------+

countriesWithWins: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Winner: string]
```

2. Retrieve the list of country names that have won a world cup and the number of world cup each has won in descending order.

+ Code + Text

03:07 PM (3s) 8

```
val worldCupWins = worldCupWinner.groupBy("Winner")
                                 .count()
                                 .orderBy(desc("count"))
                                 .withColumnRenamed("count", "WorldCupWins")
worldCupWins.show()
```

(2) Spark Jobs

worldCupWins: org.apache.spark.sql.DataFrame = [Winner: string, WorldCupWins: long]

```
+-----------+------------+
|     Winner|WorldCupWins|
+-----------+------------+
|   'Brazil'|           5|
|  'Germany'|           4|
|    'Italy'|           4|
|  'Uruguay'|           2|
|'Argentina'|           2|
|   'France'|           1|
|  'England'|           1|
|    'Spain'|           1|
+-----------+------------+

worldCupWins: org.apache.spark.sql.DataFrame = [Winner: string, WorldCupWins: bigint]
```

3. List the Capital of the countries in increasing order of country population for countries that have population more than 100 million.

03:07 PM (2s) 10

```
val largePopulationCapitals = country.filter("Population > 100")
                                     .select("Capital", "Population")
                                     .orderBy("Population")
largePopulationCapitals.show()
```

(1) Spark Jobs

largePopulationCapitals: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Capital: string, Population: string]

```
+----------------+----------+
|         Capital|Population|
+----------------+----------+
|    'Mexico City'|     122.3|
|          'Tokyo'|    127.06|
|         'Moscow'|    142.46|
|          'Abuja'|     173.6|
|       'Brasilia'|     202.4|
|'Washington D.C.'|     318.9|
+----------------+----------+

largePopulationCapitals: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Capital: string, Population: string]
```

4. List the Name of the stadium which has hosted a match where the number of goals scored by a single team was greater than 4.

03:07 PM (2s) 12 Scala

```
val highScoringStadiums = matchData.filter(col("team1_score") > 4 || col("team2_score") > 4)
                                   .select("stadium")
                                   .distinct()
highScoringStadiums.show()
```

(2) Spark Jobs

highScoringStadiums: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [stadium: string]

```
+-----------------+
|          stadium|
+-----------------+
|'Estadio Mineirao'|
|'Arena Fonte Nova'|
+-----------------+

highScoringStadiums: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [stadium: string]
```

5. List the names of all the cities which have the name of the Stadium starting with “Estadio”.

03:07 PM (2s) 14 Scala

```scala
val estadioCities = matchData.filter(lower(col("stadium")).contains("estadio"))
                                .select("stadium", "host_city")
                                .distinct()
estadioCities.show(50, truncate = false)
```

(2) Spark Jobs

estadioCities: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [stadium: string, host_city: string]

```
+--------------------+----------------+
|stadium             |host_city       |
+--------------------+----------------+
|'Estadio Nacional'  |'Brasilia'      |
|'Estadio Mineirao'  |'Belo Horizonte'|
|'Estadio das Dunas' |'Natal'         |
|'Estadio do Maracana'|'Rio De Janerio'|
|'Estadio Beira-Rio' |'Porto Alegre'  |
|'Estadio Castelao'  |'Fortaleza'     |
+--------------------+----------------+

estadioCities: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [stadium: string, host_city: string]
```

6. List all stadiums and the number of matches hosted by each stadium.

03:07 PM (2s) 16 Scala

```scala
val stadiumMatchCount = matchData.groupBy("stadium")
                                .agg(count("match_id").alias("MatchCount"))
stadiumMatchCount.show()
```

(2) Spark Jobs

stadiumMatchCount: org.apache.spark.sql.DataFrame = [stadium: string, MatchCount: long]

```
+--------------------+----------+
|             stadium|MatchCount|
+--------------------+----------+
| 'Estadio Beira-Rio'|         5|
|  'Estadio Mineirao'|         6|
|  'Estadio Nacional'|         7|
| 'Estadio das Dunas'|         4|
|  'Arena Fonte Nova'|         6|
|  'Arena Da Baixada'|         4|
|'Estadio do Marac...|         7|
|'Arena de Sao Paulo'|         6|
|    'Arena Amazonia'|         4|
| 'Estadio Castelao'|         6|
|    'Arena Pantanal'|         4|
|   'Arena Pernamuco'|         5|
+--------------------+----------+

stadiumMatchCount: org.apache.spark.sql.DataFrame = [stadium: string, MatchCount: bigint]
```

7. List the First Name, Last Name and Date of Birth of Players whose heights are greater than 198 cms.

03:07 PM (1s) 18

```
val tallPlayers = player.filter(col("height").cast("int") > 198)
                        .select("fname", "lname", "dob")
tallPlayers.show()
```

(1) Spark Jobs

tallPlayers: org.apache.spark.sql.DataFrame = [fname: string, lname: string ... 1 more field]

```
+--------+----------+------------+
|   fname|     lname|         dob|
+--------+----------+------------+
|'FRASER'| 'FORSTER'|'1988-03-17'|
|   'LEE'|'BUMYOUNG'|'1989-04-02'|
+--------+----------+------------+

tallPlayers: org.apache.spark.sql.DataFrame = [fname: string, lname: string ... 1 more field]
```

8. List the Fname, Lname, Position and No of Goals scored by the Captain of a team who has more than 2 Yellow cards or 1 Red card.

03:07 PM (3s) 20 Scala

```
val playerWithGoals = player
  .join(goalScorer, "player_id")
  .join(disciplinaryRecord, "player_id")
  .filter("(no_of_yellow_cards > 2 OR no_of_red_cards > 1) AND is_captain = true")
  .select("fname", "lname", "position", "goals")

playerWithGoals.show()
```

(3) Spark Jobs

playerWithGoals: org.apache.spark.sql.DataFrame = [fname: string, lname: string ... 2 more fields]

```
+--------+-------+----------+-----+
|   fname|  lname|  position|goals|
+--------+-------+----------+-----+
|'THIAGO'|'SILVA'|'Defender'|    1|
+--------+-------+----------+-----+

playerWithGoals: org.apache.spark.sql.DataFrame = [fname: string, lname: string ... 2 more fields]
```

Insights

- **Goal Trends**: Average number of goals per match increased in recent years.
- **Disciplinary Patterns**: Teams with higher yellow cards often perform poorly in later stages.
- **Performance by Country**: Historical dominance by a few countries like Brazil and Germany.

Conclusion

This Big Data assignment demonstrates the effective use of Apache Spark for processing and analyzing large datasets. It highlights how Spark can be leveraged for schema validation, EDA, and generating insights in a scalable manner.