**Report on Designing Fifa\_Worldcup in MongoDB**

**1. Data Extraction and Loading Program**

The Python program attached below utilizes the Pandas library for data manipulation and the PyMongo library for MongoDB interactions. The program extracts data from flat CSV files, structures it according to the defined schemas, and loads it into the MongoDB database.

**1.1 Steps for Implementation**

1. **Read the CSV file**: Use pandas to load the data from a CSV file.
2. **Preprocess the data**:
   * Change data types as needed (e.g., converting strings to integers or dates).
   * Handle missing values (e.g., filling, dropping, or replacing).

**Convert to MongoDB-compatible format**: Convert the pandas DataFrame into a format suitable for MongoDB insertion.

1. **Insert into MongoDB**: Use pymongo to insert the preprocessed data into a MongoDB collection.

**2. Document Schema Design**

**2.1 COUNTRY Document Schema**

The COUNTRY document schema is designed to hold essential information about each country participating in the FIFA World Cup. This schema captures various attributes that provide a comprehensive overview of the country and its football team. The details are structured as follows

* **Country Name (Cname)**:
* *Type*: String
* *Description*: The official name of the country.
* **Capital (Capital)**:
* *Type*: String
* *Description*: The capital city of the country.
* **Population (Population)**:
* *Type*: Integer
* *Description*: The total population of the country.
* **Manager (Manager)**:
* *Type*: String
* *Description*: The name of the national team manager.
* **Players (Players)**:
* *Type*: Array of Objects
* *Description*: A collection of players on the national team, with each player object containing the following fields:
* **Last Name (Lname)**: *String* - The last name of the player.
* **First Name (Fname)**: *String* - The first name of the player.
* **Height (Height)**: *Float* - The height of the player in centimeters.
* **Date of Birth (DOB)**: *Date* - The date when the player was born.
* **Is Captain (is\_Captain)**: *Boolean* - Indicates whether the player is a captain.
* **Position (Position)**: *String* - The playing position of the player (e.g., Forward, Defender).
* **Number of Yellow Cards (no\_Yellow\_cards)**: *Integer* - The total number of yellow cards received by the player.
* **Number of Red Cards (no\_Red\_cards)**: *Integer* - The total number of red cards received by the player.
* **Number of Goals (no\_Goals)**: *Integer* - The total number of goals scored by the player.
* **Number of Assists (no\_Assists)**: *Integer* - The total number of assists made by the player.
* **World Cup History (World\_Cup\_History)**:
* *Type*: Array of Objects
* *Description*: Historical data of World Cups won, structured as follows:
* **Year (Year)**: *Integer* - The year in which the World Cup was won.
* **Host (Host)**: *String* - The host country of that specific World Cup.

**2.2 STADIUM Document Schema**

* + - The **STADIUM** document schema captures information about the stadiums where FIFA World Cup matches were held. This schema includes:
    - **Stadium Name (Stadium)**:
    - *Type*: String
    - *Description*: The name of the stadium.
    - **City (City)**:
    - *Type*: String
    - *Description*: The city where the stadium is located.
    - **Matches (Matches)**:
    - *Type*: Array of Objects
    - *Description*: A list of matches played at the stadium, with each match object containing:
    - **Match Details (Match)**:
    - *Type*: Object
    - *Contains*:
    - **Team 1 Name (Team1)**: *String* - The name of the first team.
    - **Team 2 Name (Team2)**: *String* - The name of the second team.
    - **Team 1 Score (Team1Score)**: *Integer* - The score of the first team.
    - **Team 2 Score (Team2Score)**: *Integer* - The score of the second team.
    - **Match Date (Date)**: *Date* - The date when the match was played.

**2.3 Python Program:**

**[1]** Attached as **mongo\_preprocessing\_data\_file.py**

**import pandas as pd  
import numpy as np  
from pymongo import MongoClient  
  
# Load CSV files  
countries\_df = pd.read\_csv('/Users/nivasm/Documents/UTA ASSIGNMENTS/BIG DATA/ASSIGNMENT1/Country\_Modified.csv')  
disciplinary\_records\_df = pd.read\_csv('/Users/nivasm/Documents/UTA ASSIGNMENTS/BIG DATA/ASSIGNMENT1/Player\_Cards.csv')  
matches\_df = pd.read\_csv('/Users/nivasm/Documents/UTA ASSIGNMENTS/BIG DATA/ASSIGNMENT1/Match\_results.csv')  
past\_winners\_df = pd.read\_csv('/Users/nivasm/Documents/UTA ASSIGNMENTS/BIG DATA/ASSIGNMENT1/Worldcup\_History.csv')  
player\_statistics\_df = pd.read\_csv('/Users/nivasm/Documents/UTA ASSIGNMENTS/BIG DATA/ASSIGNMENT1/Player\_Assists\_Goals.csv')  
players\_df = pd.read\_csv('/Users/nivasm/Documents/UTA ASSIGNMENTS/BIG DATA/ASSIGNMENT1/Players.csv')  
  
# Function to convert all numpy data types to native Python data types  
def convert\_numpy\_to\_python(value):  
 if isinstance(value, (np.integer, np.int64)):  
 return int(value)  
 elif isinstance(value, (np.float64, np.float32)):  
 return float(value)  
 else:  
 return value  
country\_docs = []  
for \_, country in countries\_df.iterrows():  
 relevant\_players = []  
 country\_name = country['Country\_Name']  
 relevant\_country\_players = players\_df[players\_df['Country\_Name'] == country\_name]  
  
 for \_, player in relevant\_country\_players.iterrows():  
  
 disciplinary\_record = disciplinary\_records\_df[  
 disciplinary\_records\_df['Player\_id'] == player['Player\_id']].squeeze()  
  
 player\_stat = player\_statistics\_df[player\_statistics\_df['Player\_id'] == player['Player\_id']].squeeze()  
  
 relevant\_players.append({  
 'Lname': convert\_numpy\_to\_python(player['Lname']),  
 'Fname': convert\_numpy\_to\_python(player['Fname']),  
 'Height': convert\_numpy\_to\_python(player['Height']),  
 'DOB': convert\_numpy\_to\_python(player['DOB']),  
 'is\_Captain': convert\_numpy\_to\_python(player['Is\_captain']),  
 'Position': convert\_numpy\_to\_python(player['Position']),  
 'no\_Yellow\_cards': convert\_numpy\_to\_python(disciplinary\_record['No\_of\_Yellow\_cards']) if not disciplinary\_record.empty else 0,  
 'no\_Red\_cards': convert\_numpy\_to\_python(disciplinary\_record['No\_of\_Red\_cards']) if not disciplinary\_record.empty else 0,  
 'no\_Goals': convert\_numpy\_to\_python(player\_stat['Goals']) if not player\_stat.empty else 0,  
 'no\_Assists': convert\_numpy\_to\_python(player\_stat['Assists']) if not player\_stat.empty else 0  
 })  
  
 wc\_history = [  
 {  
 'Year': convert\_numpy\_to\_python(winner['Year']),  
 'Host': convert\_numpy\_to\_python(winner['Host'])  
 }  
 for \_, winner in past\_winners\_df.iterrows() if winner['Winner'] == country\_name  
 ]  
  
 country\_doc = {  
 'Cname': convert\_numpy\_to\_python(country['Country\_Name']),  
 'Capital': convert\_numpy\_to\_python(country['Capital']),  
 'Population': convert\_numpy\_to\_python(country['Population']),  
 'Manager': convert\_numpy\_to\_python(country['Manager']),  
 'Players': relevant\_players,  
 'World\_Cup\_History': wc\_history  
 }  
  
 country\_docs.append(country\_doc)  
  
stadium\_docs = []  
  
stadium\_grouped = matches\_df.groupby(['Stadium', 'Host\_city'])  
  
for (stadium, city), group in stadium\_grouped:  
 match\_list = group.apply(lambda match: {  
 'Match': {  
 'Team1': convert\_numpy\_to\_python(match['Team1']),  
 'Team2': convert\_numpy\_to\_python(match['Team2']),  
 'Team1Score': convert\_numpy\_to\_python(match['Team1\_score']),  
 'Team2Score': convert\_numpy\_to\_python(match['Team2\_score']),  
 'Date': convert\_numpy\_to\_python(match['Date'])  
 }  
 }, axis=1).tolist()  
  
 stadium\_doc = {  
 'Stadium': convert\_numpy\_to\_python(stadium),  
 'City': convert\_numpy\_to\_python(city),  
 'Matches': match\_list  
 }  
 stadium\_docs.append(stadium\_doc)  
  
client = MongoClient('mongodb://localhost:27017/')  
new\_db = client['Fifa\_WorldCup\_New']  
try:  
 new\_db.country\_documents.delete\_many({})  
 if country\_docs:  
 new\_db.country\_documents.insert\_many(country\_docs)  
 print("Country documents inserted into new database successfully.")  
 else:  
 print("No country documents to insert.")  
  
 new\_db.stadium\_documents.delete\_many({})  
 if stadium\_docs:  
 new\_db.stadium\_documents.insert\_many(stadium\_docs)  
 print("Stadium documents inserted into new database successfully.")  
 else:  
 print("No stadium documents to insert.")  
except Exception as e:  
 print(f"An error occurred while inserting documents: {e}")  
  
print("COUNTRY and STADIUM documents uploaded to the new MongoDB database successfully.")**

**3. MongoDB Queries**

Once the data has been loaded into the MongoDB database, various queries can be executed to retrieve the stored documents. Below are some examples:

**3.1 Retrieve the list of country names that have won a world cup:**

**Query:**

db.country\_documents.find(

{ "World\_Cup\_History": { $exists: true, $ne: [] } },

{ "Cname": 1, "\_id": 0 }

)

O/P:

A screenshot of a computer code

Description automatically generated

**3.2 Retrieve the list of country names that have won a world cup and the number of world cups each has won in descending order:**

**Query:**

db.country\_documents.aggregate([

{ $match: { "World\_Cup\_History": { $exists: true, $ne: [] } } },

{

$project: {

Cname: 1,

number\_of\_world\_cups: { $size: "$World\_Cup\_History" }

}

},

{ $sort: { number\_of\_world\_cups: -1 } }

])

**O/P:**

**A screenshot of a computer program

Description automatically generated**

**A computer code with text

Description automatically generated with medium confidence**

**3.3 Retrieve the Capital of the countries in increasing order of country population for countries that have population more than 100 million:**

**Query:**

db.country\_documents.find(

{ Population: { $gt: 100 } },

{ Capital: 1, "\_id": 0 }

).sort({ Population: 1 })

**Output:**

**A screen shot of a computer code

Description automatically generated**

**3.4 Retrieve the Name of the stadium which has hosted a match where the number of goals scored by a single team was greater than 4:**

**Query:**

db.stadium\_documents.aggregate([

{

"$match": {

"$or": [

{"Matches.Match.Team1Score": {"$gt": 4}},

{"Matches.Match.Team2Score": {"$gt": 4}}

]

}

},

{

"$project": {

"Stadium": 1,

"\_id": 0

}

}

])

**Output:**

**A screenshot of a computer screen

Description automatically generated**

**3.5 Retrieve the names of all the cities which have the name of the Stadium starting with “Estadio”:**

**Query:**

db.stadium\_documents.find(

{"Stadium": {"$regex": "^Estadio"}},

{"City": 1, "\_id": 0}

)

**Output:**

A screen shot of a computer code

Description automatically generated

**3.6 Retrieve all stadiums and the number of matches hosted by each stadium**

**Query:**

db.stadium\_documents.aggregate([

{

"$project": {

"Stadium": 1,

"Number\_of\_Matches": {"$size": "$Matches"}

}

}

])

**Output:**

**A screen shot of a computer program

Description automatically generated**

****

**3.7 Retrieve the First Name, Last Name, and Date of Birth of Players whose heights are greater than 198 cms**

**Query:**

db.country\_documents.aggregate([

{ $unwind: "$Players" },

{ $match: { "Players.Height": { $gt: 198 } } },

{

$project: {

Fname: "$Players.Fname",

Lname: "$Players.Lname",

DOB: "$Players.DOB"

}

}

])

**Output:**

**A computer screen shot of text

Description automatically generated**

**3.8 Retrieve captains with more than 2 yellow cards or at least 1 red card**

**Query:**

db.country\_documents.aggregate([

{ $unwind: "$Players" },

{ $match: { $or: [{ "Players.no\_Yellow\_cards": { $gt: 2 } }, { "Players.no\_Red\_cards": { $gt: 1 } }] } },

{ $match: { "Players.is\_Captain": true } },

{

$project: {

Fname: "$Players.Fname",

Lname: "$Players.Lname",

Position: "$Players.Position",

No\_of\_Goals: "$Players.no\_Goals"

}

}

])

**Output:**

**A computer code on a blue background

Description automatically generated**

**4. Mongodb Screenshots:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**5. Conclusion**

The data extraction and loading process has been successfully implemented to create structured COUNTRY and STADIUM documents in MongoDB. This structured approach allows for efficient data retrieval and analysis, providing valuable insights into the FIFA World Cup and its participants.