Balaji Paiyur Mohan(1001576836)
Sai Aishwarya Somesula(1001408525)

Project 2 DESIGN DOCUMENT

1) Install MongoDB Community Edition on Windows systems and optionally start MongoDB as a Windows service.

2) To start the mongo shell and connect to your MongoDB instance running on **localhost** with **default port**. At a prompt in a terminal window (or a command prompt for Windows), go to your mongodb installation dir:

```
cd <mongodb installation dir>
```

3) Now type mongo to start mongo shell.

4) When you run mongo without any arguments, the mongo shell will attempt to connect to the MongoDB instance running on the localhost interface on port 27017.

5) To display the database you are using, type `db`:

```
db
```

6) The operation should return `test`, which is the default database. To switch databases, issue the `use <db>` helper, as in the following example:

```
use <database>
```

7) To list the available databases, use the helper `show dbs`.

8) When you first store data in the database, such as by creating a collection, MongoDB creates the database. For example, the following creates the collection `myCollection` during the `insert()` operation:

```
db.myCollection.insert({x : 1})
```

9) The `db.collection.find()` method returns a cursor to the results; however, in the `mongo` shell, if the returned cursor is not assigned to a variable using the `var` keyword, then the cursor is automatically iterated up to 20 times to print up to the first 20 documents that match the query. The `mongo` shell will prompt `Type it` to iterate another 20 times.

10) To format the printed result, you can add the operation, as in the following:

```
db.myCollection.find().pretty()
```

11) To directly insert csv files directly as documents in collections, type:

```
mongoimport -d mydb -c things --type csv --file file.csv --headerline
```

`-d <database>`

Specifies the name of the database on which to run the **mongoimport**.

`-c <collection>`

Specifies the collection to import.

`--type <json|csv|tsv>`

Specifies the file type to import. The default format is JSON, but it's possible to import csv and tsv files.

`--file <filename>`

Specifies the location and name of a file containing the data to import. If you do not specify a file, `mongoimport` reads data from standard input.

`--headerline`

If using `--type csv` or `--type tsv`, uses the first line as field names. Otherwise, `mongoimport` will import the first line as a distinct document.

12) This will import the data into each collections. Now we can proceed to query the Database.

In this fashion, we are getting started with the project:

```
C:\Users\Balaji\Downloads>cd C:\Program Files\MongoDB\Server\4.0\bin

C:\Program Files\MongoDB\Server\4.0\bin>mongo
MongoDB shell version v4.0.1
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 4.0.1
Server has startup warnings:
2018-08-07T20:11:34.868-0500 I CONTROL  [initandlisten]
2018-08-07T20:11:34.868-0500 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-08-07T20:11:34.868-0500 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2018-08-07T20:11:34.868-0500 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
```

```
To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

> show dbs
admin         0.000GB
config        0.000GB
dbs           0.000GB
local         0.000GB
pymongo_test  0.000GB
> use dbs
switched to db dbs
>
```

```
C:\Program Files\MongoDB\Server\4.0\bin>mongoimport -d dbs -c players --type csv --file players-text.csv --headerline
2018-08-08T16:56:40.292-0500    connected to: localhost
2018-08-08T16:56:40.524-0500    imported 736 documents

C:\Program Files\MongoDB\Server\4.0\bin>mongoimport -d dbs -c TEAM --type csv --file world-cup-teams-text.csv --headerline
2018-08-08T17:00:08.038-0500    connected to: localhost
2018-08-08T17:00:08.290-0500    imported 32 documents

C:\Program Files\MongoDB\Server\4.0\bin>mongoimport -d dbs -c GOALS --type csv --file world-cup-goals-text.csv --headerline
2018-08-08T17:00:49.041-0500    connected to: localhost
2018-08-08T17:00:49.335-0500    imported 157 documents

C:\Program Files\MongoDB\Server\4.0\bin>mongoimport -d dbs -c STARTING_LINEUPS --type csv --file world-cup-starting-lineups-text.csv --headerline
2018-08-08T17:02:00.062-0500    connected to: localhost
2018-08-08T17:02:00.313-0500    imported 1408 documents

C:\Program Files\MongoDB\Server\4.0\bin>mongoimport -d dbs -c STADIUM --type csv --file stadiums-text.csv --headerline
2018-08-08T17:02:44.998-0500    connected to: localhost
2018-08-08T17:02:45.625-0500    imported 12 documents

C:\Program Files\MongoDB\Server\4.0\bin>mongoimport -d dbs -c GAME --type csv --file world-cup-schedule-results-text.csv --headerline
2018-08-08T17:04:36.077-0500    connected to: localhost
2018-08-08T17:04:36.325-0500    imported 64 documents
```

```
> show dbs
admin         0.000GB
config        0.000GB
dbs           0.000GB
local         0.000GB
pymongo_test  0.000GB
> use dbs
switched to db dbs
> show collections
GAME
GOALS
PLAYER
STADIUM
STARTING_LINEUPS
TEAM
>
```

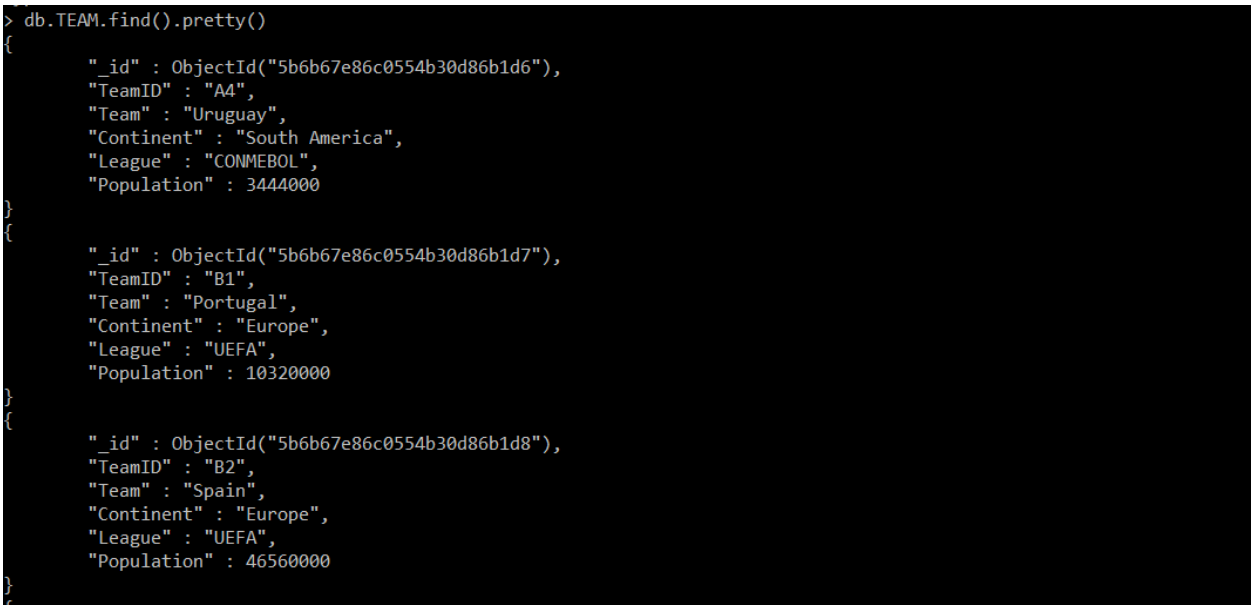
To view the data in the collections, use the helper:

```
db.TEAM.find()
```

To view in more organized way, type:

```
db.TEAM.find().pretty()
```

Below two pictures show the difference of views:



Requirement 1 can be implemented by typing the following code in the mongo shell:

Method 1:

```
db.TEAM.aggregate([{
        $lookup:{
          from: "GAME",
          localField: "TeamID",
          foreignField: "TeamID1",
          as: "TEAM_SCORES"
        }
},
```

```
{
        $lookup:{
          from:"GAME",
          localField:"TeamID",
          foreignField:"TeamID2",
          as:"TEAM_SCORES"
        }
}]).pretty()
```

Which will display results like this:

Method 2:

We can also try implementing function to gather team details in one collection by using forEach().

`cursor.forEach(`*function*`)`

Iterates the cursor to apply a JavaScript `function` to each document from the cursor.The `forEach()` method has the following prototype form:

```
db.collection.find().forEach(<function>)
```

The forEach() method has the following parameter: A JavaScript function to apply to each document from the cursor. The <function> signature includes a single argument that is passed the current document to process.

The following example invokes the forEach() method on the cursor returned by find() to print the name of each user in the collection:

```
db.TEAM.find().forEach(
... function (TEAM_SCORES) {
... TEAM_SCORES.GDate = db.GAME.findOne({"MatchDate":TEAM_SCORES.GDate})
... ... TEAM_SCORES.GCity = db.STADIUM.find({"SCity": TEAM_SCORES.GCity})
... ... TEAM_SCORES.GStadium = db.STADIUM.find({"SName":
TEAM_SCORES.GStadium})
... ... TEAM_SCORES.T1Name = db.GAME.find({"TeamID1": TEAM_SCORES.T1Name})
... ... TEAM_SCORES.T1Score = db.GAME.find({"Team1_Score":
TEAM_SCORES.T1Score})
... ... TEAM_SCORES.T2Name = db.GAME.find({"TeamID2": TEAM_SCORES.T2Name})
... ... TEAM_SCORES.T2Score = db.GAME.find({"Team2_Score":
TEAM_SCORES.T2Score})
... db.TEAM_SCORES.insert(TEAM_SCORES);
... }
... );
});
```

Wherein both of these codes perform the same operation of below SQL query:

```
Select G.GDate, S.SName, T1.TName, G.Score1, T2.TName, G.Score2
From TEAM AS T1, TEAM AS T2, GAME AS G, STADIUM AS S
Where G.TeamID1 = T1.TeamID AND G.TeamID2 = T2.TeamID
AND G.SId = S.SId
ORDER BY ASC;
```

To export resulting collections from both of these codes, we can use `mongoexport`:

Use mongoexport to dump a collection:

Export JSON File:

```
mongoexport --db <database-name> --collection <collection-name> --out
output.json
```