PL ASSIGNMENT 7

Name :- Onkar Rajgonda Khot

PRN :- 21510094

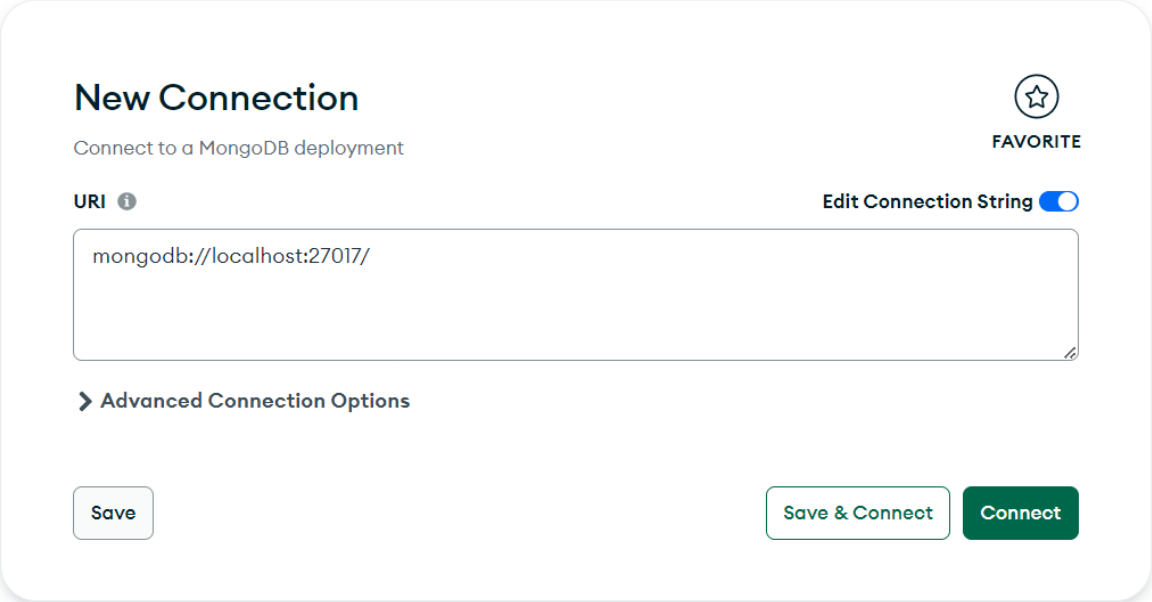
Batch:- T6

Perform following problem statements using MongoDB

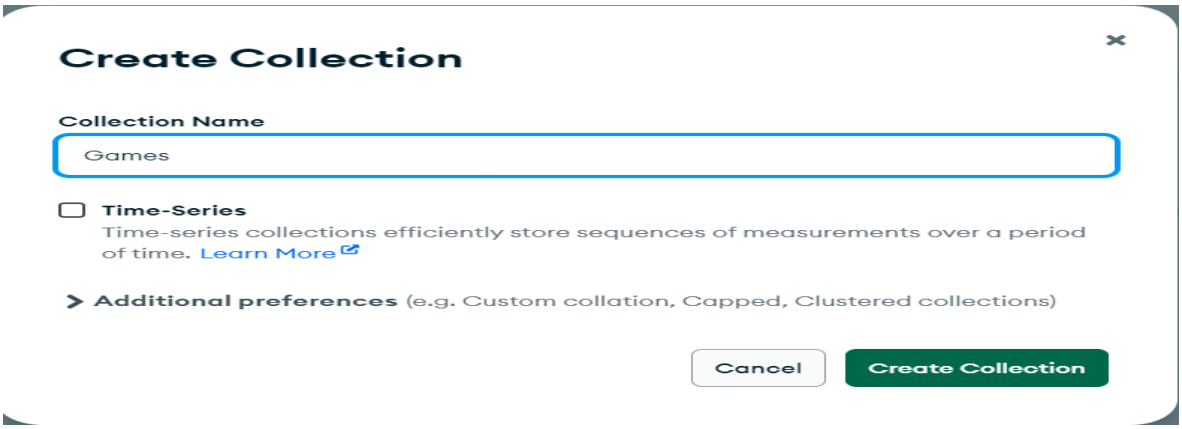
Problem Statement 1:

• First off, you need a database to connect to. MongoDB doesn't have a "create database" command. Instead, it is going to create one for you when you try to save something into it.

• Install and Connect to the mongoDB.



• Create a collection called 'games'. We're going to put some games in it



- Add 5 games to the database. Give each document the following properties: name, genre, rating (out of 100). If you make some mistakes and want to clean it out, use remove()on your collection.

```
>_MONGOSH
> use PLassignment
< switched to db PLassignment
> db.Games.insertMany([{name: 'game1', genre:'Action',rating: 97} ,{name :'game2' , genre :'Adventure', rating: 87}, {name :'ga
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("6548c351f611d8394e324d30"),
      '1': ObjectId("6548c351f611d8394e324d31"),
      '2': ObjectId("6548c351f611d8394e324d32"),
      '3': ObjectId("6548c351f611d8394e324d33"),
      '4': ObjectId("6548c351f611d8394e324d34")
    }
  }
```

- Write a query that returns all the games.

```
Databases
> db.Games.find()
< {
    _id: ObjectId("6548c351f611d8394e324d30"),
    name: 'game1',
    genre: 'Action',
    rating: 97
  }
  {
    _id: ObjectId("6548c351f611d8394e324d31"),
    name: 'game2',
    genre: 'Adventure',
    rating: 87
  }
  {
    _id: ObjectId("6548c351f611d8394e324d32"),
    name: 'game3',
    genre: 'Puzzle',
    rating: 95
  }
  {
    _id: ObjectId("6548c351f611d8394e324d33"),
    name: 'game4',
    genre: 'Sports',
    rating: 89
```

- Write a query to find one of your games by name without using limit(). Use the findOne method. Look how much nicer it's formatted!.

```
> db.Games.findOne({name:'game3'})
< {
    _id: ObjectId("6548c351f611d8394e324d32"),
    name: 'game3',
    genre: 'Puzzle',
    rating: 95
  }
```

- Write a query that returns the 3 highest rated games.

```
> db.Games.find().sort({rating: -1}).limit(3)
< {
    _id: ObjectId("6548c351f611d8394e324d30"),
    name: 'game1',
    genre: 'Action',
    rating: 97
  }
  {
    _id: ObjectId("6548c351f611d8394e324d32"),
    name: 'game3',
    genre: 'Puzzle',
    rating: 95
  }
  {
    _id: ObjectId("6548c351f611d8394e324d34"),
    name: 'game5',
    genre: 'Strategy',
    rating: 93
  }
```

• Update your two favourite games to have two achievements called ‘Game Master’ and ‘Speed Demon’, each under a single key. Show two ways to do this. Do the first using update()and do the second using save(). Hint: for save, you might want to query the object and store it in a variable first.

```
> db.Games.updateMany({name: {$in: ['game1' ,'game2']}},{$set: {achievements: ['Game MAster ', 'Speed Demon']}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 2,
    modifiedCount: 2,
    upsertedCount: 0
  }
```

- Write a query that returns all the games that have both the ‘Game Maser’ and the ‘Speed Demon’ achievements.

```
Atlas atlas-j6t81d-shard-0 [primary] test> db.games.updateMany( { name: { $in: ['Game 1', 'Game 2'] } }, { $set: { achievements: ['Game Master',
 'Speed Demon'] } } ,{multi:true})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 0,
  upsertedCount: 0
}
Atlas atlas-j6t81d-shard-0 [primary] test> db.games.find()
[
  {
    _id: ObjectId("6542336c4c21e157f1610a74"),
    name: 'Game 1',
    genre: 'Adventure',
    rating: 85,
    achievements: [ 'Game Master', 'Speed Demon' ]
  },
  {
    _id: ObjectId("6542336c4c21e157f1610a75"),
    name: 'Game 2',
    genre: 'Action',
    rating: 90,
    achievements: [ 'Game Master', 'Speed Demon' ]
  },
  {
    _id: ObjectId("6542336c4c21e157f1610a76"),
    name: 'Game 3',
    genre: 'Puzzle',
    rating: 75
  },
  {
    _id: ObjectId("6542336c4c21e157f1610a77"),
    name: 'Game 4',
```

- Write a query that returns only games that have achievements. Not all of your games should have achievements, obviously.

```
> db.Games.find({achievements:{$exists:true}})
< {
    _id: ObjectId("6548c351f611d8394e324d30"),
    name: 'game1',
    genre: 'Action',
    rating: 97,
    achievements: [
      'Game MAster ',
      'Speed Demon'
    ]
  }
  {
    _id: ObjectId("6548c351f611d8394e324d31"),
    name: 'game2',
    genre: 'Adventure',
    rating: 87,
    achievements: [
      'Game MAster ',
      'Speed Demon'
    ]
  }
```

Problem Statement 2:

MapReduce question:

- Write a reduce that calculates the total score from all games for each player and check the output.

```
games> db.games.mapReduce(function () {if(this.scores) for(var i=0; i<this.scores.length; i++){var player = this.scores[i].name;var score = this
.scores[i].score;emit(player, score);}}, function (key, values) {return Array.sum(values);}, {out: "player_scores"})
{ result: 'player_scores', ok: 1 }
games> db.player_scores.find()
[
  { _id: 'derrick', value: 2795 },
  { _id: 'bryan', value: 3439 },
  { _id: 'tim', value: 1764 }
]
```

Problem Statement 3:

REST API:

- Use the REST API to show all the game data stored in the db from the games collection.

- Output all of the available returned data in an html table in the following format:

Games

	_id ObjectId	name String	genre String	rating Int32	achievements Array
1	ObjectId('6548c351f611d8394e3…	"game1"	"Action"	97	[] 2 elements
2	ObjectId('6548c351f611d8394e3…	"game2"	"Adventure"	87	[] 2 elements
3	ObjectId('6548c351f611d8394e3…	"game3"	"Puzzle"	95	No field
4	ObjectId('6548c351f611d8394e3…	"game4"	"Sports"	89	No field
5	ObjectId('6548c351f611d8394e3…	"game5"	"Strategy"	93	No field