Assignment-1

Name:- E Mokshagnya

Reg.No:- 22MIC7251

Slot:- F2

Q1.

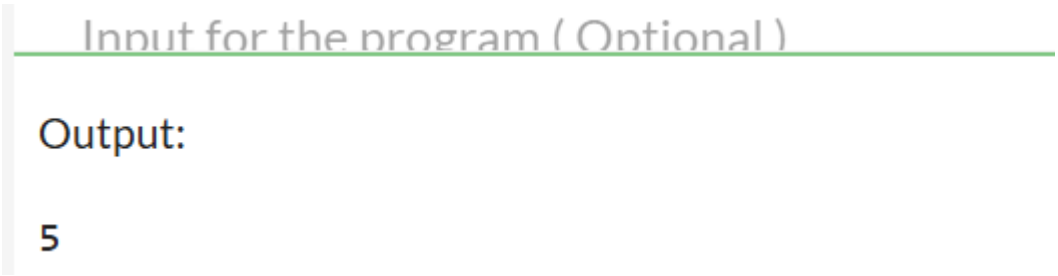
Consider a redis database that stores information about registered members of a **book club**. Implement the members collection using a Redis data structure with no duplicate values.

a. Insert 5 unique members into the members set (2 M)

Code:-

SADD members "Naruto" "Sasuke" "Luffy" "Goku" "Ichigo"

output:-



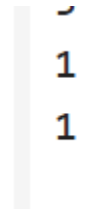
b. Move 2 members from members to premium_members (2 M)

Code:-

SMOVE members premium_members "Naruto"

SMOVE members premium_members "Luffy"

output:-



c. Create a new set all_members by merging members and premium_members (2 M)

Assignment-1

Code:-

SUNIONSTORE all_members members premium_members

output:-

```
Output:

5
```

d. Add 3 new members into all_members (2 M)

Code:-

SADD all_members "Tanjiro" "Levi" "Eren"

output:-

```
3
```

e. Remove 1 random member from both all_members and premium_members (2 M)

Code:-

SPOP all_members SPOP premium_members

output:-

```
Sasuke
Naruto
```

Q2.

Write Redis commands for the following:

a. Insert values 5, 10, 15, 20, 25, 10 into a list named stack (2 M)

Code:-

RPUSH stack 5 10 15 20 25 10

Assignment-1

output:-

```
Output:

6
```

b. Insert 50 between 15 and 20 (2 M)

Code:-

LINSERT stack AFTER 15 50

output:-

```
7
```

c. Remove the last 3 values from stack and insert them into a new list archive (2 M)

Code:-

RPOPLPUSH stack archive

RPOPLPUSH stack archive

RPOPLPUSH stack archive

output:-

```
10
25
20
```

d. Show the first 3 values from stack (2 M)

Code:-

LRANGE stack 0 2

Assignment-1

output:-

```
5
10
15
```

e. Update the last value in archive to 99 (2 M)

Code:-

LSET archive -1 99

output:-

```
OK
```

Q3.

Consider the following RDBMS relation of a **university exam system**:

ExamID	Student Name	Total Marks	Subjects (Marks)	Bonus (%)	Final Scores
201	Ravi	600	[90, 80, 70, 60, 100, 200]	[5, 0, 10, 5, 0, 0]	[94, 80, 77, 63, 100, 200]
202	Meena	550	[75, 85, 65, 120, 105, 100]	[0, 5, 0, 10, 0, 0]	[75, 89, 65, 132, 105, 100]

Answer the following using MongoDB:

a. Insert all records into a collection exams. (2 M)

Code:-

```
db.exams.insertMany([
 {
   ExamID: 201,
   StudentName: "Ravi",
   TotalMarks: 600,
```

Assignment-1

```
  Subjects: [90, 80, 70, 60, 100, 200],
  Bonus: [5, 0, 10, 5, 0, 0],
  FinalScores: [94, 80, 77, 63, 100, 200]
 },
 {
  ExamID: 202,
  StudentName: "Meena",
  TotalMarks: 550,
  Subjects: [75, 85, 65, 120, 105, 100],
  Bonus: [0, 5, 0, 10, 0, 0],
  FinalScores: [75, 89, 65, 132, 105, 100]
 }
]);
```

output:-

```
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('68da84d6e7f627a876f69fd5'),
      '1': ObjectId('68da84d6e7f627a876f69fd6')
    }
  }
```

b. Show all ExamID and Student Name where Total Marks > 580 (2 M)

Code:-

```
db.exams.find({ TotalMarks: { $gt: 580 } }, { ExamID: 1, StudentName: 1 });
```

output:-

```
> db.exams.find({ TotalMarks: { $gt: 580 } }, { ExamID: 1, StudentName: 1 });
< {
    _id: ObjectId('68da84d6e7f627a876f69fd5'),
    ExamID: 201,
    StudentName: 'Ravi'
  }
```

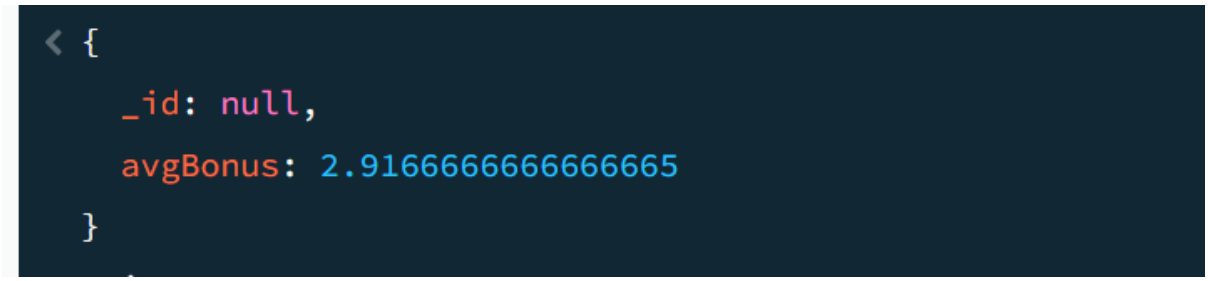
Assignment-1

c. Find the average bonus applied for all students. (2 M)

Code:-

```
db.exams.aggregate([
 { $unwind: "$Bonus" },
 { $group: { _id: null, avgBonus: { $avg: "$Bonus" } } }
]);
```

output:-

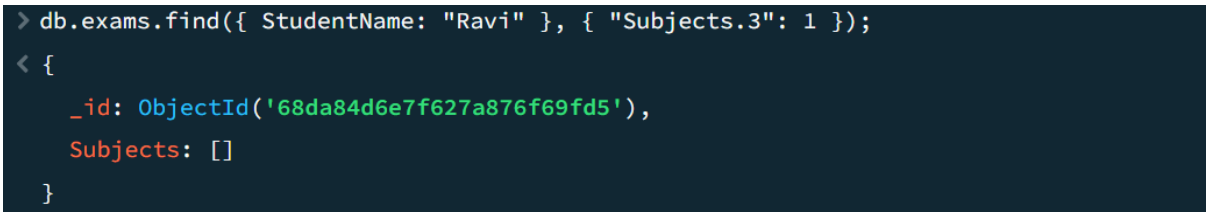


d. Show the 4th subject marks for Ravi (2 M)

Code:-

```
db.exams.find({ StudentName: "Ravi" }, { "Subjects.3": 1 });
```

output:-



e. Find the difference between Total Marks and average Final Scores for ExamID 201 (2 M)

Code:-

```
db.exams.aggregate([
 { $match: { ExamID: 201 } },
```

Assignment-1

```
 { $project: {
    ExamID: 1,
    diff: { $subtract: [ "$TotalMarks", { $avg: "$FinalScores" } ] }
 }}
]);
```

output:-

```
{
    _id: ObjectId('68da84d6e7f627a876f69fd5'),
    ExamID: 201,
    diff: 497.6666666666667
  }
```

Q4.

Using the same table data in Q3, answer the following in MongoDB:

a. Increase all bonus values by 3% (2 M)

Code:-

```
db.exams.updateMany({}, { $inc: { "Bonus.$[]": 3 } });
```

output:-

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
```

b. Update Total Marks of each student to be equal to the maximum Final Score (2 M)

Assignment-1

Code:-

```
db.exams.aggregate([
 { $project: { ExamID: 1, maxScore: { $max: "$FinalScores" } } }
]).forEach(doc => {
 db.exams.updateOne({ ExamID: doc.ExamID }, { $set: { TotalMarks: doc.maxScore } });
});
```

output:-

```
> db.exams.aggregate([
    { $project: { ExamID: 1, maxScore: { $max: "$FinalScores" } } }
  ]).forEach(doc => {
    db.exams.updateOne({ ExamID: doc.ExamID }, { $set: { TotalMarks: doc.maxScore } });
  });
```

c. Find the average of all final scores for ExamID 202 (2 M)

Code:-

```
db.exams.aggregate([
 { $match: { ExamID: 202 } },
 { $project: { avgFinal: { $avg: "$FinalScores" } } }
]);
```

output:-

```
< {
    _id: ObjectId('68da84d6e7f627a876f69fd6'),
    avgFinal: 94.33333333333333
  }
```

d. Find the student who got the lowest final score (2 M)

Code:-

```
db.exams.aggregate([
 { $unwind: "$FinalScores" },
```

Assignment-1

```
 { $sort: { FinalScores: 1 } },
 { $limit: 1 },
 { $project: { StudentName: 1, FinalScores: 1 } }
]);
```

output:-

```
< {
    _id: ObjectId('68da84d6e7f627a876f69fd5'),
    StudentName: 'Ravi',
    FinalScores: 63
  }
```

e. Show all student names sorted by Total Marks in descending order. (2 M)

Code:-

```
db.exams.find({}, { StudentName: 1 }).sort({ TotalMarks: -1 });
```

output:-

```
> db.exams.find({}, { StudentName: 1 }).sort({ TotalMarks: -1 });
< {
    _id: ObjectId('68da84d6e7f627a876f69fd5'),
    StudentName: 'Ravi'
  }
  {
    _id: ObjectId('68da84d6e7f627a876f69fd6'),
    StudentName: 'Meena'
  }
```

Q5.

Based on Q3 table data, answer the following in MongoDB:

a. Show student names where any subject mark is greater than 100 and Total Marks > 580 (2 M)

Code:-

Assignment-1

```
db.exams_q5.find(
 { TotalMarks: { $gt: 580 }, Subjects: { $gt: 100 } },
 { StudentName: 1, _id: 0 }
);
```

output:-

```
    { StudentName: 1, _id: 0 }
  );
< {
    StudentName: 'Ravi'
  }
ecommerce>
```

b. Add the average Final Score for each student as overall_score (2 M)

Code:-

```
db.exams_q5.aggregate([
 { $project: { ExamID: 1, overall_score: { $avg: "$FinalScores" } } }
]).forEach(doc => {
 db.exams_q5.updateOne(
  { ExamID: doc.ExamID },
  { $set: { overall_score: doc.overall_score } }
 );
});
```

output:-

```
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 3,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

Assignment-1

c. Remove the Bonus field for students where Total Marks < 560 (2 M)

Code:-

```
db.exams_q5.updateMany(
 { TotalMarks: { $lt: 560 } },
 { $unset: { Bonus: "" } }
);
```

output:-

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

d. Show all exams where Subjects array does not contain exactly 6 elements (2 M)

Code:-

```
db.exams_q5.find(
 { $expr: { $ne: [ { $size: "$Subjects" }, 6 ] } },
 { ExamID: 1, StudentName: 1, Subjects: 1, _id: 0 }
);
```

output:-

```
{
  ExamID: 203,
  StudentName: 'Tanjiro',
  Subjects: [
    80,
    90,
    100
  ]
}
```

Assignment-1

e. Create a unique index on ExamID (2 M)

Code:-

```
db.exams_q5.createIndex({ ExamID: 1 }, { unique: true });
```

output:-