CODE:

Matrix Multiplication:

```
import java.util.Scanner;

public class MatrixMul{
  public static void main(String args[]){

        int row1, col1, row2, col2;
        Scanner s = new Scanner(System.in);
        System.out.print("Enter number of rows in first matrix:");
        row1 = s.nextInt();
        System.out.print("Enter number of columns in first matrix:");
        col1 = s.nextInt();
        System.out.print("Enter number of rows in second matrix:");
        row2 = s.nextInt();
        System.out.print("Enter number of columns in second matrix:");
        col2 = s.nextInt();
        if (col1 != row2) {
        System.out.println("Matrix multiplication is not possible");
        }
        else {
        int a[][] = new int[row1][col1];
        int b[][] = new int[row2][col2];
        int c[][] = new int[row1][col2];
        System.out.println("Enter values for matrix A : \n");
        for (int i = 0; i < row1; i++) {
        for (int j = 0; j < col1; j++)
        a[i][j] = s.nextInt();
        }
        System.out.println("Enter values for matrix B : \n");
        for (int i = 0; i < row2; i++) {
        for (int j = 0; j < col2; j++)
        b[i][j] = s.nextInt();
        }

        System.out.println("Matrix multiplication is : \n");
        for(int i = 0; i < row1; i++) {
        for(int j = 0; j < col2; j++){
        c[i][j]=0;
        for(int k = 0; k < col1; k++){
        c[i][j] += a[i][k] * b[k][j];
        }
        System.out.print(c[i][j] + " ");
        }
        System.out.println();
        }
        }
```

```
  }
}
```

OUTPUT:

```
swpc-10@swpc10-H81M-S: ~/Desktop/BDA LA...
swpc-10@swpc10-H81M-S:~/Desktop/BDA LAB 5$ javac MatrixMul.java
swpc-10@swpc10-H81M-S:~/Desktop/BDA LAB 5$ java MatrixMul
Enter number of rows in first matrix:2
Enter number of columns in first matrix:2
Enter number of rows in second matrix:2
Enter number of columns in second matrix:2
Enter values for matrix A :

1
2
3
4
Enter values for matrix B :

4
3
2
1
Matrix multiplication is :

8 5
20 13
swpc-10@swpc10-H81M-S:~/Desktop/BDA LAB 5$
```

Aggregation in MongoDB:

- **Creating a database:**

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

vedantdhoke> db.college.insertOne({"name":"vedantdhoke","age":21,"id":1,"sec":"A","subject":["chemistry","geography"]})
{
  acknowledged: true,
  insertedId: ObjectId("633d93df9a9feb403f60d10a")
}
vedantdhoke> db.college.insertOne({"name":"rockybhai","age":31,"id":1,"sec":"B","subject":["english","geography"]})
{
  acknowledged: true,
  insertedId: ObjectId("633d93fe9a9feb403f60d10b")
}
vedantdhoke> db.college.insertOne({"name":"salmankhan","age":51,"id":2,"sec":"","subject":["english","geography"]})
{
  acknowledged: true,
  insertedId: ObjectId("633d94249a9feb403f60d10c")
}
vedantdhoke> db.college.insertOne({"name":"batman","age":51,"id":3,"sec":"","subject":["english","computergraphics"]})
{
  acknowledged: true,
  insertedId: ObjectId("633d94469a9feb403f60d10d")
}
vedantdhoke> db.college.insertOne({"name":"kratos","age":100,"id":4,"sec":"","subject":["english","computergraphics"]})
{
  acknowledged: true,
  insertedId: ObjectId("633d94649a9feb403f60d10e")
}
vedantdhoke>
```

mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

```
vedantdhoke> db.college.find().pretty()
[
  {
    _id: ObjectId("633d93df9a9feb403f60d10a"),
    name: 'vedantdhoke',
    age: 21,
    id: 1,
    sec: 'A',
    subject: [ 'chemistry', 'geography' ]
  },
  {
    _id: ObjectId("633d93fe9a9feb403f60d10b"),
    name: 'rockybhai',
    age: 31,
    id: 1,
    sec: 'B',
    subject: [ 'english', 'geography' ]
  },
  {
    _id: ObjectId("633d94249a9feb403f60d10c"),
    name: 'salmankhan',
    age: 51,
    id: 2,
    sec: '',
    subject: [ 'english', 'geography' ]
  },
  {
    _id: ObjectId("633d94469a9feb403f60d10d"),
    name: 'batman',
    age: 51,
    id: 3,
    sec: '',
    subject: [ 'english', 'computergraphics' ]
  },
  {
    _id: ObjectId("633d94649a9feb403f60d10e"),
    name: 'kratos',
    age: 100,
    id: 4,
    sec: '',
    subject: [ 'english', 'computergraphics' ]
  }
]
```

- **Displaying the total number of students in one section only:**

  In this example, for taking a count of the number of students in section B we first filter the documents using the $match operator, and then we use the $count accumulator to count the total number of documents that are passed after filtering from the $match.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

vedantdhoke> db.college.aggregate([{$match:{sec:"B"}},{$count:"Total student in sec :B"}])
[ { 'Total student in sec :B': 1 } ]
vedantdhoke>
```

- **Displaying the total number of students in both the sections and maximum age from both section:**

  In this example, we use $group to group, so that we can count for every other section in the documents, here $sum sums up the document in each group and $max accumulator is applied on age expression which will find the maximum age in each document.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

vedantdhoke> db.college.aggregate([{$group:{_id:"$sec",total_st:{$sum:1},max_age:{$max:"$age"}}}])
[
  { _id: 'B', total_st: 1, max_age: 31 },
  { _id: '', total_st: 3, max_age: 100 },
  { _id: 'A', total_st: 1, max_age: 21 }
]
vedantdhoke>
```

- **Displaying details of students whose age is greater than 30 using match stage:**

  In this example, we display students whose age is greater than 30. So we use the $match operator to filter out the documents.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
vedantdhoke> db.college.aggregate([{$match:{age:{$gt:30}}}])
[
  {
    _id: ObjectId("633d93fe9a9feb403f60d10b"),
    name: 'rockybhai',
    age: 31,
    id: 1,
    sec: 'B',
    subject: [ 'english', 'geography' ]
  },
  {
    _id: ObjectId("633d94249a9feb403f60d10c"),
    name: 'salmankhan',
    age: 51,
    id: 2,
    sec: '',
    subject: [ 'english', 'geography' ]
  },
  {
    _id: ObjectId("633d94469a9feb403f60d10d"),
    name: 'batman',
    age: 51,
    id: 3,
    sec: '',
    subject: [ 'english', 'computergraphics' ]
  },
  {
    _id: ObjectId("633d94649a9feb403f60d10e"),
    name: 'kratos',
    age: 100,
```

- **Sorting the students on the basis of age:**

  In this example, we are using the $sort operator to sort in ascending order we provide ‘age’:1 if we want to sort in descending order we can simply change 1 to -1 i.e. ‘age’:-1.

mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

```
vedantdhoke> db.college.aggregate([{'$sort':{'age':1}}])
[
  {
    _id: ObjectId("633d93df9a9feb403f60d10a"),
    name: 'vedantdhoke',
    age: 21,
    id: 1,
    sec: 'A',
    subject: [ 'chemistry', 'geography' ]
  },
  {
    _id: ObjectId("633d93fe9a9feb403f60d10b"),
    name: 'rockybhai',
    age: 31,
    id: 1,
    sec: 'B',
    subject: [ 'english', 'geography' ]
  },
  {
    _id: ObjectId("633d94249a9feb403f60d10c"),
    name: 'salmankhan',
    age: 51,
    id: 2,
    sec: '',
    subject: [ 'english', 'geography' ]
  },
  {
    _id: ObjectId("633d94469a9feb403f60d10d"),
    name: 'batman',
    age: 51,
    id: 3,
    sec: '',
    subject: [ 'english', 'computergraphics' ]
  },
  {
    _id: ObjectId("633d94649a9feb403f60d10e"),
    name: 'kratos',
    age: 100,
    id: 4,
    sec: '',
    subject: [ 'english', 'computergraphics' ]
  }
]
vedantdhoke>
```

- **Displaying details of a student having the largest age in the section – B:**

  In this example, first, we only select those documents that have section B, so for that, we use the $match operator then we sort the documents in descending order using $sort by setting ‘age’:-1 and then to only show the topmost result we use $limit.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

vedantdhoke> db.college.aggregate([{$match:{sec:"B"}},{'$sort':{'age':-1}},{$limit:1}])
[
  {
    _id: ObjectId("633d93fe9a9feb403f60d10b"),
    name: 'rockybhai',
    age: 31,
    id: 1,
    sec: 'B',
    subject: [ 'english', 'geography' ]
  }
]
vedantdhoke>
```

- **Displaying distinct names and ages (non-repeating):**

  Here, we use a distinct() method that finds distinct values of the specified field (i.e., name).

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

vedantdhoke> db.college.distinct("name")
[ 'batman', 'kratos', 'rockybhai', 'salmankhan', 'vedantdhoke' ]
vedantdhoke>
```

- **Counting the total numbers of documents:**

  Here, we use count() to find the total number of the document, unlike find() method it does not find all the documents, rather it counts them and returns a number.

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

vedantdhoke> db.college.count()
DeprecationWarning: Collection.count() is deprecated. Use countDocuments or estimatedDocumentCount.
5
vedantdhoke>
```

- **Joins:**Mongodb is not a relational database but you can write a function to perform join operations.

mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

```
vedantdhokelibrary> db.authors.insert([
...     {
...         _id: 'a1',
...         name: { first: 'orlando', last: 'becerra' },
...         age: 27
...     },
...     {
...         _id: 'a2',
...         name: { first: 'mayra', last: 'sanchez' },
...         age: 21
...     }
... ]);
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{ acknowledged: true, insertedIds: { '0': 'a1', '1': 'a2' } }
vedantdhokelibrary> db.categories.insert([
...     {
...         _id: 'c1',
...         name: 'sci-fi'
...     },
...     {
...         _id: 'c2',
...         name: 'romance'
...     }
... ]);
{ acknowledged: true, insertedIds: { '0': 'c1', '1': 'c2' } }
vedantdhokelibrary> db.books.insert([
...     {
...         _id: 'b1',
...         name: 'Groovy Book',
...         category: 'c1',
...         authors: ['a1']
...     },
...     {
...         _id: 'b2',
...         name: 'Java Book',
...         category: 'c2',
...         authors: ['a1','a2']
...     },
... ]);
{ acknowledged: true, insertedIds: { '0': 'b1', '1': 'b2' } }
vedantdhokelibrary> db.lendings.insert([
...     {
...         _id: 'l1',
...         book: 'b1',
...         date: new Date('01/01/11'),
...         lendingBy: 'jose'
...     },
...     {
```

mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

```
...     {
...         _id: 'l2',
...         book: 'b1',
...         date: new Date('02/02/12'),
...         lendingBy: 'maria'
...     }
... ]);
{ acknowledged: true, insertedIds: { '0': 'l1', '1': 'l2' } }
vedantdhokelibrary> db.books.find().forEach(
...     function (newBook) {
...         newBook.category = db.categories.findOne( { "_id": newBook.category } )
...         newBook.lendings = db.lendings.find( { "book": newBook._id  } ).toArray
...         newBook.authors = db.authors.find( { "_id": { $in: newBook.authors }  }
...         db.booksReloaded.insert(newBook);
...     }
... );
```

```
vedantdhokelibrary> db.booksReloaded.find().pretty()
[
  {
    _id: 'b1',
    name: 'Groovy Book',
    category: { _id: 'c1', name: 'sci-fi' },
    authors: [
      {
        _id: 'a1',
        name: { first: 'orlando', last: 'becerra' },
        age: 27
      }
    ],
    lendings: [
      {
        _id: 'l1',
        book: 'b1',
        date: ISODate("2010-12-31T18:30:00.000Z"),
        lendingBy: 'jose'
      },
      {
        _id: 'l2',
        book: 'b1',
        date: ISODate("2012-02-01T18:30:00.000Z"),
        lendingBy: 'maria'
      }
    ]
  },
  {
    _id: 'b2',
    name: 'Java Book',
    category: { _id: 'c2', name: 'romance' },
    authors: [
      {
        _id: 'a1',
        name: { first: 'orlando', last: 'becerra' },
        age: 27
      },
      { _id: 'a2', name: { first: 'mayra', last: 'sanchez' }, age: 21 }
    ],
    lendings: []
  }
]
```