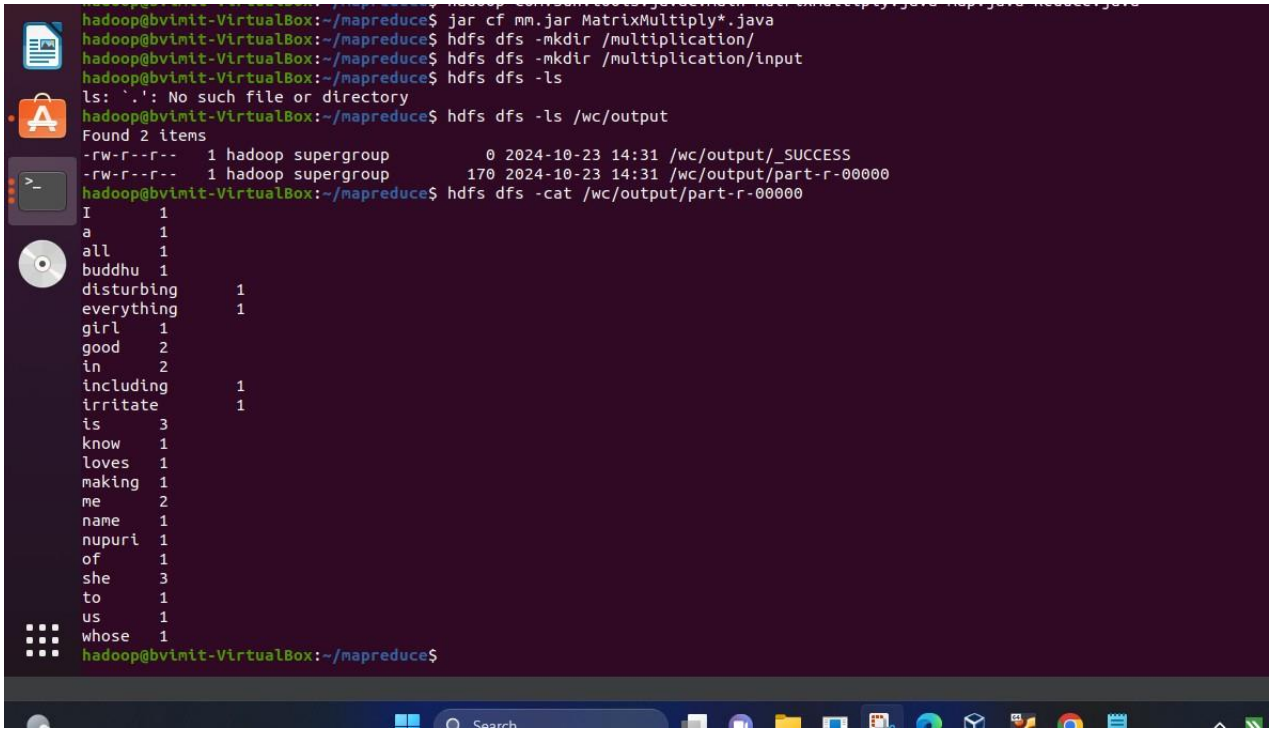
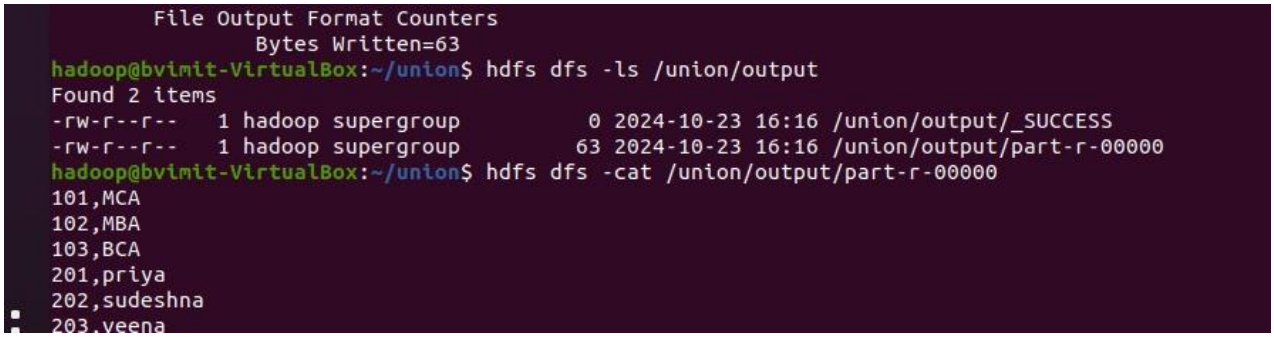
Map Reduce

1. Write a program in Map Reduce for Word Count operation



2. Write a program in Map Reduce for Union operation.



3. Write a program in Map Reduce for Intersection operation

```
hadoop@bvimit-VirtualBox:~/intersection$ hadoop com.sun.tools.javac.Main IntersectionOperation.java
hadoop@bvimit-VirtualBox:~/intersection$ jar cf io.jar IntersectionOperation*.class
hadoop@bvimit-VirtualBox:~/intersection$ hdfs dfs -ls /
Found 8 items
drwxr-xr-x   - hadoop supergroup          0 2024-10-29 19:02 /multiplication
drwxr-xr-x   - hadoop supergroup          0 2022-11-18 12:23 /test
drwxr-xr-x   - hadoop supergroup          0 2022-11-18 12:37 /test1
drwxr-xr-x   - hadoop supergroup          0 2022-11-22 10:51 /testing
drwxrwxr-x   - hadoop supergroup          0 2022-11-26 15:52 /tmp
drwxr-xr-x   - hadoop supergroup          0 2024-10-29 19:28 /union
drwxr-xr-x   - hadoop supergroup          0 2022-11-26 14:21 /user
drwxr-xr-x   - hadoop supergroup          0 2024-10-29 18:40 /wordcount
```

```
hadoop@bvimit-VirtualBox:~/intersection$ hdfs dfs -mkdir /io
hadoop@bvimit-VirtualBox:~/intersection$ hdfs dfs -mkdir /io/input
hadoop@bvimit-VirtualBox:~/intersection$ hdfs dfs -put data1 /io/input
hadoop@bvimit-VirtualBox:~/intersection$ hdfs dfs -put data2 /io/input
hadoop@bvimit-VirtualBox:~/intersection$ hadoop jar io.jar IntersectionOperation /io/input/data1 /io/input/data2 /io/output
2024-10-29 20:10:24,147 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-10-29 20:10:24,535 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute y
our application with ToolRunner to remedy this.
2024-10-29 20:10:24,562 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.staging/job_173020515002
7_0011
2024-10-29 20:10:24,829 INFO input.FileInputFormat: Total input files to process : 2
2024-10-29 20:10:25,343 INFO mapreduce.JobSubmitter: number of splits:2
2024-10-29 20:10:25,925 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1730205150027_0011
2024-10-29 20:10:25,925 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-10-29 20:10:26,097 INFO conf.Configuration: resource-types.xml not found
```

```
hadoop@bvimit-VirtualBox:~/intersection$ hdfs dfs -ls /io/output
Found 2 items
-rw-r--r--   1 hadoop supergroup          0 2024-10-29 20:10 /io/output/_SUCCESS
-rw-r--r--   1 hadoop supergroup         20 2024-10-29 20:10 /io/output/part-r-00000
hadoop@bvimit-VirtualBox:~/intersection$ hdfs dfs -cat /io/output/part-r-00000
banana
mango
orange
hadoop@bvimit-VirtualBox:~/intersection$
```

4. Write a program in Map Reduce for Matrix Multiplication

```
        File Output Format Counters
                Bytes Written=36
hadoop@bvimit-VirtualBox:~/matrixmultiply$ hdfs dfs -ls /multiply/output
Found 2 items
-rw-r--r--   1 hadoop supergroup          0 2024-10-23 15:57 /multiply/output/_SUCCESS
-rw-r--r--   1 hadoop supergroup         36 2024-10-23 15:57 /multiply/output/part-r-00000
hadoop@bvimit-VirtualBox:~/matrixmultiply$ hdfs dfs -cat /multiply/output/part-r-00000
0,0,19.0
0,1,22.0
1,0,43.0
1,1,50.0
hadoop@bvimit-VirtualBox:~/matrixmultiply$
```

Assignment 03

Q1) Demonstrate basic Commands in MongoDB

a. Create and drop a database

```
test> use student
switched to db student
student> show dbs
admin   40.00 KiB
config  72.00 KiB
local   40.00 KiB
student> db.createCollection("studentdetails")
{ ok: 1 }
student> db.createCollection("student_details")
{ ok: 1 }
student> db.createCollection("stud_details")
{ ok: 1 }
student> db.createCollection("studdetails")
{ ok: 1 }
student> show collections
stud_details
studdetails
student_details
studentdetails
student> show dbs
admin    40.00 KiB
config   72.00 KiB
local    40.00 KiB
student  32.00 KiB
student> db.dropDatabase("student")
{ ok: 1, dropped: 'student' }
student> show dbs
admin   40.00 KiB
config  72.00 KiB
local   40.00 KiB
student> show collections
```

b. Creating the Collection in MongoDB

```
student> db.createCollection("studentdetails")
{ ok: 1 }
student> db.createCollection("student_details")
{ ok: 1 }
student> db.createCollection("stud_details")
{ ok: 1 }
student> db.createCollection("studdetails")
{ ok: 1 }
student> show collections
stud_details
studdetails
student_details
studentdetails
```

c. Renaming a collection

```
student> db.createCollection("studentdetails")
{ ok: 1 }
student> db.createCollection("student_details")
{ ok: 1 }
student> db.createCollection("stud_details")
{ ok: 1 }
student> db.createCollection("studdetails")
{ ok: 1 }
student> db.studentdetails.renameCollection("STUDENTDETAILS")
{ ok: 1 }
student> show collections
stud_details
studdetails
student_details
STUDENTDETAILS
student>
```

d. MongoDB Insert Document (insert single document, insert multiple documents)

- **Inserting single document**

```
student> db.studentdetails.insertOne({
...   name: "Vinayak",
...   age: 21,
...   gender: "Male",
...   course: "Computer Science",
...   marks: {
...     math: 85,
...     science: 90,
...     english: 88
...   }
... })
{
  acknowledged: true,
  insertedId: ObjectId('66c7731f19270ad5ff2710bc')
}
student> db.studentdetails.find()
[
  {
    _id: ObjectId('66c7731f19270ad5ff2710bc'),
    name: 'Vinayak',
    age: 21,
    gender: 'Male',
    course: 'Computer Science',
    marks: { math: 85, science: 90, english: 88 }
  }
]
student>
```

- **Inserting multiple document**

```
student> db.studentdetails.insertMany([
...   {
...     name: "Samiksha",
...     age: 22,
...     gender: "Female",
...     course: "Mechanical Engineering",
...     marks: {
...       math: 78,
...       science: 82,
...       english: 80
...     }
...   },
...   {
...     name: "Divya",
...     age: 23,
...     gender: "Male",
...     course: "Electronics",
...     marks: {
...       math: 88,
...       science: 85,
...       english: 90
...     }
...   },
...   {
...     name: "Deepika",
...     age: 21,
...     gender: "Female",
...     course: "Biotechnology",
...     marks: {
...       math: 79,
...       science: 89,
...       english: 87
...     }
...   },
```

```
...     {
...       name: "Sairaj",
...       age: 21,
...       gender: "Female",
...       course: "Biotechnology",
...       marks: {
...         math: 79,
...         science: 89,
...         english: 87
...       }
...     },
...     {
...       name: "Bhakti",
...       age: 21,
...       gender: "Female",
...       course: "Biotechnology",
...       marks: {
...         math: 79,
...         science: 89,
...         english: 87
...       }
...     },
...     {
...       name: "Purva",
...       age: 21,
...       gender: "Female",
...       course: "Biotechnology",
...       marks: {
...         math: 79,
...         science: 89,
...         english: 87
...       }
...     }
... ])
{
```

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('66c7758c422cdcdf1f2710bd'),
    '1': ObjectId('66c7758c422cdcdf1f2710be'),
    '2': ObjectId('66c7758c422cdcdf1f2710bf'),
    '3': ObjectId('66c7758c422cdcdf1f2710c0'),
    '4': ObjectId('66c7758c422cdcdf1f2710c1'),
    '5': ObjectId('66c7758c422cdcdf1f2710c2')
  }
}
student>
```

```
student> db.studentdetails.find()
[
  {
    _id: ObjectId('66c77579422cdcdf1f2710bc'),
    name: 'Vinayak',
    age: 21,
    gender: 'Male',
    course: 'Computer Science',
    marks: { math: 85, science: 90, english: 88 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710bd'),
    name: 'Samiksha',
    age: 22,
    gender: 'Female',
    course: 'Mechanical Engineering',
    marks: { math: 78, science: 82, english: 80 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710be'),
    name: 'Divya',
    age: 23,
    gender: 'Male',
    course: 'Electronics',
    marks: { math: 88, science: 85, english: 90 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710bf'),
    name: 'Deepika',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  },
```

```
  {
    _id: ObjectId('66c7758c422cdcdf1f2710c0'),
    name: 'Sairaj',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710c1'),
    name: 'Bhakti',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710c2'),
    name: 'Purva',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  }
]
student>
```

e. Querying all the documents in json format and Querying based on the criteria (find() method) [Use comparison operators, logical query operators]

- **Comparison Operator: -**

1. Equal to("$eq"):

```
student> db.studentdetails.find({ age: { $eq: 21 } })
[
  {
    _id: ObjectId('66c77579422cdcdf1f2710bc'),
    name: 'Vinayak',
    age: 21,
    gender: 'Male',
    course: 'Computer Science',
    marks: { math: 85, science: 90, english: 88 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710bf'),
    name: 'Deepika',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710c0'),
    name: 'Sairaj',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710c1'),
    name: 'Bhakti',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710c2'),
    name: 'Purva',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  }
]
student>
```

2. **Greater than("$gt"):**

```
student> db.studentdetails.find({ "marks.math": { $gt: 80 } })
[
  {
    _id: ObjectId('66c77579422cdcdf1f2710bc'),
    name: 'Vinayak',
    age: 21,
    gender: 'Male',
    course: 'Computer Science',
    marks: { math: 85, science: 90, english: 88 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710be'),
    name: 'Divya',
    age: 23,
    gender: 'Male',
    course: 'Electronics',
    marks: { math: 88, science: 85, english: 90 }
  }
]
student>
```

3. **Less than("$lt"):**

```
student> db.studentdetails.find({ age: { $lt: 23 } })
[
  {
    _id: ObjectId('66c77579422cdcdf1f2710bc'),
    name: 'Vinayak',
    age: 21,
    gender: 'Male',
    course: 'Computer Science',
    marks: { math: 85, science: 90, english: 88 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710bd'),
    name: 'Samiksha',
    age: 22,
    gender: 'Female',
    course: 'Mechanical Engineering',
    marks: { math: 78, science: 82, english: 80 }
  },
```

```
  {
    _id: ObjectId('66c7758c422cdcdf1f2710bf'),
    name: 'Deepika',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710c0'),
    name: 'Sairaj',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710c1'),
    name: 'Bhakti',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710c2'),
    name: 'Purva',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  }
]
student>
```

4. **Greater than equal to(“$gte”):**

```
student> db.studentdetails.find({ "marks.science": { $gte: 85 } })
[
  {
    _id: ObjectId('66c77579422cdcdf1f2710bc'),
    name: 'Vinayak',
    age: 21,
    gender: 'Male',
    course: 'Computer Science',
    marks: { math: 85, science: 90, english: 88 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710be'),
    name: 'Divya',
    age: 23,
    gender: 'Male',
    course: 'Electronics',
    marks: { math: 88, science: 85, english: 90 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710bf'),
    name: 'Deepika',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710c0'),
    name: 'Sairaj',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710c1'),
    name: 'Bhakti',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  },
```

```
  {
    _id: ObjectId('66c7758c422cdcdf1f2710c2'),
    name: 'Purva',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  }
]
student>
```

5. **Less than equal to(“$lte”):**

```
student> db.studentdetails.find({ "marks.english": { $lte: 88 } })
[
  {
    _id: ObjectId('66c77579422cdcdf1f2710bc'),
    name: 'Vinayak',
    age: 21,
    gender: 'Male',
    course: 'Computer Science',
    marks: { math: 85, science: 90, english: 88 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710bd'),
    name: 'Samiksha',
    age: 22,
    gender: 'Female',
    course: 'Mechanical Engineering',
    marks: { math: 78, science: 82, english: 80 }
  },
```

```
  {
    _id: ObjectId('66c7758c422cdcdf1f2710bf'),
    name: 'Deepika',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710c0'),
    name: 'Sairaj',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710c1'),
    name: 'Bhakti',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710c2'),
    name: 'Purva',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  }
]
student>
```

- **Logical Operators: -**

1. AND(“$and”):

```
student> db.studentdetails.find({
...   $and: [
...     { age: 21 },
...     { "marks.math": { $gt: 80 } }
...   ]
... })
[
  {
    _id: ObjectId('66c77579422cdcdf1f2710bc'),
    name: 'Vinayak',
    age: 21,
    gender: 'Male',
    course: 'Computer Science',
    marks: { math: 85, science: 90, english: 88 }
  }
]
student>
```

2. OR(“$or”):

```
student> db.studentdetails.find({
...   $or: [
...     { age: 21 },
...     { course: "Computer Science" }
...   ]
... })
[
  {
    _id: ObjectId('66c77579422cdcdf1f2710bc'),
    name: 'Vinayak',
    age: 21,
    gender: 'Male',
    course: 'Computer Science',
    marks: { math: 85, science: 90, english: 88 }
  },
```

```
  {
    _id: ObjectId('66c7758c422cdcdf1f2710bf'),
    name: 'Deepika',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710c0'),
    name: 'Sairaj',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710c1'),
    name: 'Bhakti',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710c2'),
    name: 'Purva',
    age: 21,
    gender: 'Female',
    course: 'Biotechnology',
    marks: { math: 79, science: 89, english: 87 }
  }
]
student>
```

3. IN(“$in”):

```
student> db.studentdetails.find({
...   course: { $in: ["Computer Science", "Mechanical Engineering"] }
... })
[
  {
    _id: ObjectId('66c77579422cdcdf1f2710bc'),
    name: 'Vinayak',
    age: 21,
    gender: 'Male',
    course: 'Computer Science',
    marks: { math: 85, science: 90, english: 88 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710bd'),
    name: 'Samiksha',
    age: 22,
    gender: 'Female',
    course: 'Mechanical Engineering',
    marks: { math: 78, science: 82, english: 80 }
  }
]
student>
```

4. NOR(“$nor”):

```
student> db.studentdetails.find({
...   $nor: [
...     { age: 21 },
...     { course: "Biotechnology" }
...   ]
... })
[
  {
    _id: ObjectId('66c7758c422cdcdf1f2710bd'),
    name: 'Samiksha',
    age: 22,
    gender: 'Female',
    course: 'Mechanical Engineering',
    marks: { math: 78, science: 82, english: 80 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710be'),
    name: 'Divya',
    age: 23,
    gender: 'Male',
    course: 'Electronics',
    marks: { math: 88, science: 85, english: 90 }
  }
]
student>
```

5. **NOT(“$not”):**

```
student> db.studentdetails.find({
...    age: { $not: { $eq: 21 } }
... })
[
  {
    _id: ObjectId('66c7758c422cdcdf1f2710bd'),
    name: 'Samiksha',
    age: 22,
    gender: 'Female',
    course: 'Mechanical Engineering',
    marks: { math: 78, science: 82, english: 80 }
  },
  {
    _id: ObjectId('66c7758c422cdcdf1f2710be'),
    name: 'Divya',
    age: 23,
    gender: 'Male',
    course: 'Electronics',
    marks: { math: 88, science: 85, english: 90 }
  }
]
student>
```

f. Update Document

- **UpdateOne:**

```
student> db.studentdetails.updateOne(
...   { name: "Vinayak" },
...   {
...     $set: { course: "Data Science", "marks.science": 95 }
...   }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
student> db.studentdetails.findOne({ name: "Vinayak" })
{
  _id: ObjectId('66c77579422cdcdf1f2710bc'),
  name: 'Vinayak',
  age: 21,
  gender: 'Male',
  course: 'Data Science',
  marks: { math: 85, science: 95, english: 88 }
}
student>
```

- **UpdateMany:**

```
student> db.studentdetails.updateMany(
...     { age: 21 },
...     { $set: { course: "Software Engineering" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 5,
  modifiedCount: 5,
  upsertedCount: 0
}
student> db.studentdetails.findOne({age:21 })
{
  _id: ObjectId('66c77579422cdcdf1f2710bc'),
  name: 'Vinayak',
  age: 21,
  gender: 'Male',
  course: 'Software Engineering',
  marks: { math: 85, science: 95, english: 88 }
}
student>
```

g. Delete document from a collection

- **DeleteOne:**

```
student> db.studentdetails.deleteOne({ name: "Purva" })
{ acknowledged: true, deletedCount: 1 }
student> db.studentdetails.findOne({name:"Purva" })
null
student>
```

- **DeleteMany**

```
student> db.studentdetails.deleteMany({ age: 21 })
{ acknowledged: true, deletedCount: 4 }
student> db.studentdetails.findOne({age:21 })
null
student>
```

Q2) Create a student Collection with the fields: (SRN, Sname, Degree, Sem, CGPA)

```
student> db.createCollection("student")
{ ok: 1 }
student> show collections
student
studentdetails
student>
```

```
student> db.student.insertMany([
...    { SRN: "001", Sname: "Vinayak", Degree: "MCA", Sem: 1, CGPA: 8.5 },
...    { SRN: "002", Sname: "Samiksa", Degree: "BCA", Sem: 2, CGPA: 7.2 },
...    { SRN: "003", Sname: "Divya", Degree: "MCA", Sem: 3, CGPA: 9.0 },
...    { SRN: "004", Sname: "Deepika", Degree: "BCA", Sem: 4, CGPA: 6.8 },
...    { SRN: "005", Sname: "Bhakti", Degree: "MCA", Sem: 2, CGPA: 7.9 },
...    { SRN: "006", Sname: "Sairaj", Degree: "MCA", Sem: 4, CGPA: 9.1 },
...    { SRN: "007", Sname: "Purva", Degree: "BCA", Sem: 3, CGPA: 6.2 }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('66c78c5e7bc2ba84222710bc'),
    '1': ObjectId('66c78c5e7bc2ba84222710bd'),
    '2': ObjectId('66c78c5e7bc2ba84222710be'),
    '3': ObjectId('66c78c5e7bc2ba84222710bf'),
    '4': ObjectId('66c78c5e7bc2ba84222710c0'),
    '5': ObjectId('66c78c5e7bc2ba84222710c1'),
    '6': ObjectId('66c78c5e7bc2ba84222710c2')
  }
}
student>
```

1. **Display all the documents**

```
student> db.student.find()
[
  {
    _id: ObjectId('66c78c5e7bc2ba84222710bc'),
    SRN: '001',
    Sname: 'Vinayak',
    Degree: 'MCA',
    Sem: 1,
    CGPA: 8.5
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710bd'),
    SRN: '002',
    Sname: 'Samiksa',
    Degree: 'BCA',
    Sem: 2,
    CGPA: 7.2
  },
```

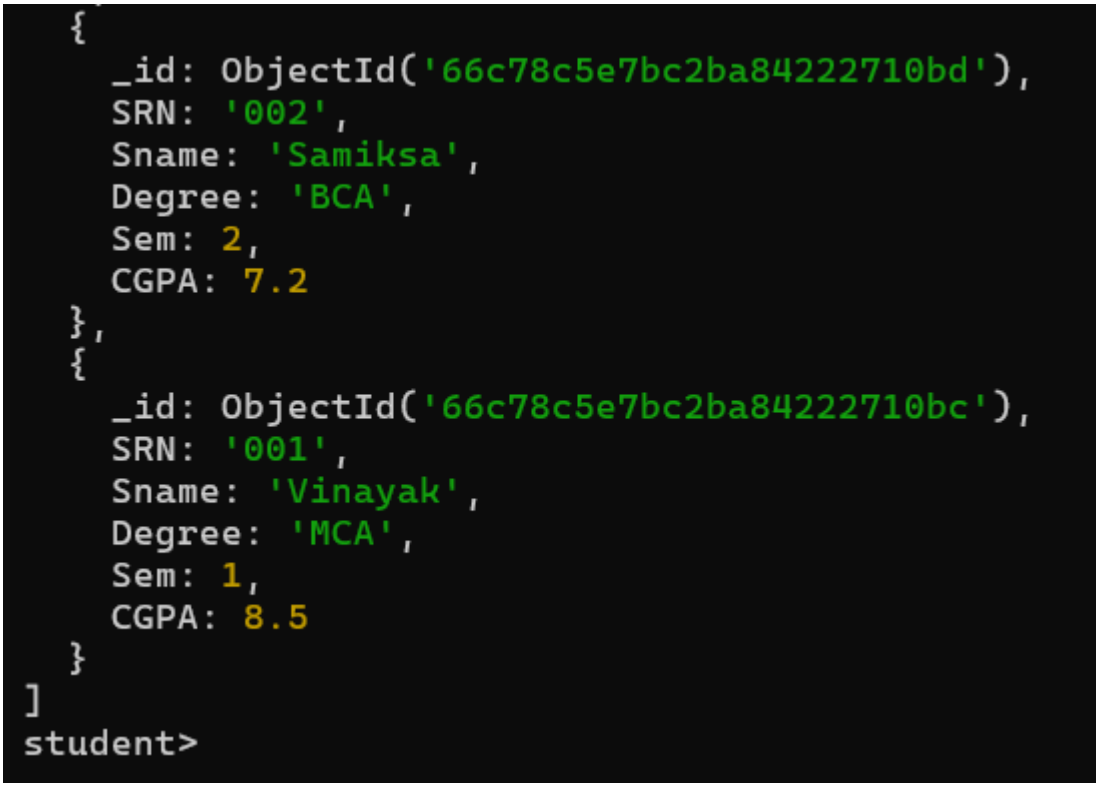
```
  {
    _id: ObjectId('66c78c5e7bc2ba84222710be'),
    SRN: '003',
    Sname: 'Divya',
    Degree: 'MCA',
    Sem: 3,
    CGPA: 9
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710bf'),
    SRN: '004',
    Sname: 'Deepika',
    Degree: 'BCA',
    Sem: 4,
    CGPA: 6.8
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710c0'),
    SRN: '005',
    Sname: 'Bhakti',
    Degree: 'MCA',
    Sem: 2,
    CGPA: 7.9
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710c1'),
    SRN: '006',
    Sname: 'Sairaj',
    Degree: 'MCA',
    Sem: 4,
    CGPA: 9.1
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710c2'),
    SRN: '007',
    Sname: 'Purva',
    Degree: 'BCA',
    Sem: 3,
    CGPA: 6.2
  }
]
student>
```

2. **Display all the students in MCA**

```
student> db.student.find({ Degree: "MCA" })
[
  {
    _id: ObjectId('66c78c5e7bc2ba84222710bc'),
    SRN: '001',
    Sname: 'Vinayak',
    Degree: 'MCA',
    Sem: 1,
    CGPA: 8.5
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710be'),
    SRN: '003',
    Sname: 'Divya',
    Degree: 'MCA',
    Sem: 3,
    CGPA: 9
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710c0'),
    SRN: '005',
    Sname: 'Bhakti',
    Degree: 'MCA',
    Sem: 2,
    CGPA: 7.9
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710c1'),
    SRN: '006',
    Sname: 'Sairaj',
    Degree: 'MCA',
    Sem: 4,
    CGPA: 9.1
  }
]
student>
```

3. **Display all the students in ascending order**

```
student> db.student.find().sort({ Sname: 1 })
[
  {
    _id: ObjectId('66c78c5e7bc2ba84222710c0'),
    SRN: '005',
    Sname: 'Bhakti',
    Degree: 'MCA',
    Sem: 2,
    CGPA: 7.9
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710bf'),
    SRN: '004',
    Sname: 'Deepika',
    Degree: 'BCA',
    Sem: 4,
    CGPA: 6.8
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710be'),
    SRN: '003',
    Sname: 'Divya',
    Degree: 'MCA',
    Sem: 3,
    CGPA: 9
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710c2'),
    SRN: '007',
    Sname: 'Purva',
    Degree: 'BCA',
    Sem: 3,
    CGPA: 6.2
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710c1'),
    SRN: '006',
    Sname: 'Sairaj',
    Degree: 'MCA',
    Sem: 4,
    CGPA: 9.1
  },
```


{
_id: ObjectId('66c78c5e7bc2ba84222710bd'),
SRN: '002',
Sname: 'Samiksa',
Degree: 'BCA',
Sem: 2,
CGPA: 7.2
},
{
_id: ObjectId('66c78c5e7bc2ba84222710bc'),
SRN: '001',
Sname: 'Vinayak',
Degree: 'MCA',
Sem: 1,
CGPA: 8.5
}
]
student>

4. **Display first 5 students**

```
student> db.student.find().limit(5).pretty()
[
  {
    _id: ObjectId('66c78c5e7bc2ba84222710bc'),
    SRN: '001',
    Sname: 'Vinayak',
    Degree: 'MCA',
    Sem: 1,
    CGPA: 8.5
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710bd'),
    SRN: '002',
    Sname: 'Samiksa',
    Degree: 'BCA',
    Sem: 2,
    CGPA: 7.2
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710be'),
    SRN: '003',
    Sname: 'Divya',
    Degree: 'MCA',
    Sem: 3,
    CGPA: 9
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710bf'),
    SRN: '004',
    Sname: 'Deepika',
    Degree: 'BCA',
    Sem: 4,
    CGPA: 6.8
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710c0'),
    SRN: '005',
    Sname: 'Bhakti',
    Degree: 'MCA',
    Sem: 2,
    CGPA: 7.9
  }
]
student>
```

5. **Display the degree of student “Rahul”**

```
]
student> db.student.find({ Sname: "Bhakti" }, { Degree: 1, _id: 0 })
[ { Degree: 'MCA' } ]
student>
```

6. **Display student details in descending order of percentage**

```
student> db.student.find().sort({ CGPA: -1 })
[
  {
    _id: ObjectId('66c78c5e7bc2ba84222710c1'),
    SRN: '006',
    Sname: 'Sairaj',
    Degree: 'MCA',
    Sem: 4,
    CGPA: 9.1
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710be'),
    SRN: '003',
    Sname: 'Divya',
    Degree: 'MCA',
    Sem: 3,
    CGPA: 9
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710bc'),
    SRN: '001',
    Sname: 'Vinayak',
    Degree: 'MCA',
    Sem: 1,
    CGPA: 8.5
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710c0'),
    SRN: '005',
    Sname: 'Bhakti',
    Degree: 'MCA',
    Sem: 2,
    CGPA: 7.9
  },
```

```
  {
    _id: ObjectId('66c78c5e7bc2ba84222710bd'),
    SRN: '002',
    Sname: 'Samiksa',
    Degree: 'BCA',
    Sem: 2,
    CGPA: 7.2
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710bf'),
    SRN: '004',
    Sname: 'Deepika',
    Degree: 'BCA',
    Sem: 4,
    CGPA: 6.8
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710c2'),
    SRN: '007',
    Sname: 'Purva',
    Degree: 'BCA',
    Sem: 3,
    CGPA: 6.2
  }
]
student>
```

7. **Display the number of of students in MCA**

```
]
student> db.student.countDocuments({ Degree: "MCA" })
4
student>
```

8. **Display all BCA students with CGPA greater than 6 but less than 8**

```
student> db.student.find({ Degree: "BCA", CGPA: { $gt: 6, $lt: 8 } })
[
  {
    _id: ObjectId('66c78c5e7bc2ba84222710bd'),
    SRN: '002',
    Sname: 'Samiksa',
    Degree: 'BCA',
    Sem: 2,
    CGPA: 7.2
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710bf'),
    SRN: '004',
    Sname: 'Deepika',
    Degree: 'BCA',
    Sem: 4,
    CGPA: 6.8
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710c2'),
    SRN: '007',
    Sname: 'Purva',
    Degree: 'BCA',
    Sem: 3,
    CGPA: 6.2
  }
]
student>
```

9. **Display all the students in MCA and in 4th Sem**

```
student> db.student.find({ Degree: "MCA", Sem: 4 })
[
  {
    _id: ObjectId('66c78c5e7bc2ba84222710c1'),
    SRN: '006',
    Sname: 'Sairaj',
    Degree: 'MCA',
    Sem: 4,
    CGPA: 9.1
  }
]
student>
```

10.Display all information where student name starts with “A”

```
student> db.student.find({ Sname: { $regex: /^S/ } })
[
  {
    _id: ObjectId('66c78c5e7bc2ba84222710bd'),
    SRN: '002',
    Sname: 'Samiksa',
    Degree: 'BCA',
    Sem: 2,
    CGPA: 7.2
  },
  {
    _id: ObjectId('66c78c5e7bc2ba84222710c1'),
    SRN: '006',
    Sname: 'Sairaj',
    Degree: 'MCA',
    Sem: 4,
    CGPA: 9.1
  }
]
student>
```

11.Display name and degree of the students whose name starts with “A”

```
student> db.student.find({ Sname: { $regex: /^D/ } }, { Sname: 1, Degree: 1, _id: 0 })
[
  { Sname: 'Divya', Degree: 'MCA' },
  { Sname: 'Deepika', Degree: 'BCA' }
]
student>
```

12.Display name and degree of all students

```
student> db.student.find({}, { Sname: 1, Degree: 1, _id: 0 })
[
  { Sname: 'Vinayak', Degree: 'MCA' },
  { Sname: 'Samiksa', Degree: 'BCA' },
  { Sname: 'Divya', Degree: 'MCA' },
  { Sname: 'Deepika', Degree: 'BCA' },
  { Sname: 'Bhakti', Degree: 'MCA' },
  { Sname: 'Sairaj', Degree: 'MCA' },
  { Sname: 'Purva', Degree: 'BCA' }
]
student>
```

Q3) Peform the following in MongoDB

Create an employee Collection with th fields: (eid, ename, dept, desig,salary,address{dno,street,locality,city})

```
student> use employee
switched to db employee
employee> db.createCollection("employee")
{ ok: 1 }
employee> show collections
employee
employee>
```

1. **Insert 10 documents**

```
employee> db.employee.insertMany([
...   { eid: "E001", ename: "Vinayak", dept: "IT", desig: "Developer", salary: 60000, address: { dno: "101", street: "Main St", locality: "Downtown", city: "Mumbai" } },
...   { eid: "E002", ename: "Samiksa", dept: "HR", desig: "Manager", salary: 90000, address: { dno: "102", street: "Second St", locality: "Midtown", city: "Navimumbai" }
},
...   { eid: "E003", ename: "Divya", dept: "Finance", desig: "Analyst", salary: 55000, address: { dno: "103", street: "Third St", locality: "Uptown", city: "Thane" } },
...   { eid: "E004", ename: "Deepika", dept: "IT", desig: "Developer", salary: 65000, address: { dno: "104", street: "Fourth St", locality: "Suburb", city: "Mumbai" } },
...   { eid: "E005", ename: "Bhakti", dept: "Sales", desig: "Executive", salary: 45000, address: { dno: "105", street: "Fifth St", locality: "Rural", city: "Navimumbai" }
 },
...   { eid: "E006", ename: "Purva", dept: "IT", desig: "Tester", salary: 50000, address: { dno: "106", street: "Sixth St", locality: "Downtown", city: "Thane" } },
...   { eid: "E007", ename: "Sairaj", dept: "Marketing", desig: "Manager", salary: 85000, address: { dno: "107", street: "Seventh St", locality: "Midtown", city: "Mumbai"
 } },
...   { eid: "E008", ename: "Yash", dept: "HR", desig: "Executive", salary: 40000, address: { dno: "108", street: "Eighth St", locality: "Suburb", city: "Navimumbai" } },

...   { eid: "E009", ename: "Pratik", dept: "Finance", desig: "Manager", salary: 95000, address: { dno: "109", street: "Ninth St", locality: "Uptown", city: "Thane" } },
...   { eid: "E010", ename: "Shubham", dept: "IT", desig: "Developer", salary: 70000, address: { dno: "110", street: "Tenth St", locality: "Downtown", city: "Mumbai" } }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('66c796e07bc2ba84222710c3'),
    '1': ObjectId('66c796e07bc2ba84222710c4'),
    '2': ObjectId('66c796e07bc2ba84222710c5'),
    '3': ObjectId('66c796e07bc2ba84222710c6'),
    '4': ObjectId('66c796e07bc2ba84222710c7'),
    '5': ObjectId('66c796e07bc2ba84222710c8'),
    '6': ObjectId('66c796e07bc2ba84222710c9'),
    '7': ObjectId('66c796e07bc2ba84222710ca'),
    '8': ObjectId('66c796e07bc2ba84222710cb'),
    '9': ObjectId('66c796e07bc2ba84222710cc')
  }
}
employee>
```

2. **Display the salary of "Rohan"**

```
employee> db.employee.find({ ename: "Yash" }, { salary: 1, _id: 0 })
[ { salary: 40000 } ]
employee>
```

3. **Display the city of employee "Ajit"**

```
employee> db.employee.find({ ename: "Deepika" }, { "address.city": 1, _id: 0 })
[ { address: { city: 'Mumbai' } } ]
employee>
```

4. **Update the salary of developers by 5000 increment**

```
employee> db.employee.find({ desig: "Developer" })
[
  {
    _id: ObjectId('66c796e07bc2ba84222710c3'),
    eid: 'E001',
    ename: 'Vinayak',
    dept: 'IT',
    desig: 'Developer',
    salary: 65000,
    address: {
      dno: '101',
      street: 'Main St',
      locality: 'Downtown',
      city: 'Mumbai'
    }
  },
```

5. **Add file age to employee “Ajit”**

```
employee> db.employee.find({ ename: "Purva" })
[
  {
    _id: ObjectId('66c796e07bc2ba84222710c8'),
    eid: 'E006',
    ename: 'Purva',
    dept: 'IT',
    desig: 'Tester',
    salary: 50000,
    address: {
      dno: '106',
      street: 'Sixth St',
      locality: 'Downtown',
      city: 'Thane'
    },
    age: 30
  }
]
employee>
```

6. **Remove all fields desig from “Rahul”**

```
employee> db.employee.updateOne(
...    { ename: "Divya" },
...    { $unset: { desig: "" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
employee> db.employee.find({ ename: "Divya" })
[
  {
    _id: ObjectId('66c796e07bc2ba84222710c5'),
    eid: 'E003',
    ename: 'Divya',
    dept: 'Finance',
    salary: 55000,
    address: {
      dno: '103',
      street: 'Third St',
      locality: 'Uptown',
      city: 'Thane'
    }
  }
]
employee>
```

7. **Display all employees from having designation “Manager” and salary 90000**

```
]
employee> db.employee.find({ desig: "Manager", salary: 90000 })
[
  {
    _id: ObjectId('66c796e07bc2ba84222710c4'),
    eid: 'E002',
    ename: 'Samiksa',
    dept: 'HR',
    desig: 'Manager',
    salary: 90000,
    address: {
      dno: '102',
      street: 'Second St',
      locality: 'Midtown',
      city: 'Navimumbai'
    }
  }
]
employee>
```

8. **Delete all documents where salary<2000**

```
employee> db.employee.deleteMany({ salary: { $lt: 55000 } })
{ acknowledged: true, deletedCount: 3 }
employee> db.employee.find({ ename: "Bhakti" })

employee> db.employee.find({ ename: "Purva" })

employee> db.employee.find({ ename: "Yash" })

employee>
```

Hive Assignment

Q1) Perform the following in Hive

1. **Create a Customer Database**

```
hive> create database Customer;
OK
Time taken: 0.034 seconds
```

2. **Create a table called Customer_Details (cid, cname, city, location, phone, pincode)**

```
hive> create table customer_details(
    > cid int,
    > cname string,
    > city string,
    > location string,
    > phone bigint,
    > pincode int) row format delimited fields terminated by ',';
OK
Time taken: 0.044 seconds
```

3. **Insert data from a .csv file**

```
hive> load data local inpath '/home/hadoop/hdfs_data/Customer_details.csv' into table customer_details;
Loading data to table userdb.customer_details
OK
Time taken: 0.122 seconds
```

G1

	A	B	C	D	E	F	G	H
1	101	Shravan	Roha	Raigad	9876754885	402109		
2	102	Parth	Palghar	Palghar	8756485851	401404		
3	103	Nilkanth	Panvel	Raigad	7564899551	410210		
4	104	Prathamesh	Roha	Raigad	7564854456	402109		
5	105	Sairaj	Satara	Satara	8545866547	400214		
6								
7								

```
hive> select * from customer_details;
OK
101     Shravan Roha    Raigad  9876754885      402109
102     Parth   Palghar Palghar 8756485851      401404
103     Nilkanth        Panvel  Raigad  7564899551      410210
104     Prathamesh      Roha    Raigad  7564854456      402109
105     Sairaj  Satara  Satara  8545866547      400214
Time taken: 0.065 seconds, Fetched: 5 row(s)
```

4. **Write a query to display all the customer name and location**

```
hive> select cname, location
    > from customer_details;
OK
Shravan Raigad
Parth   Palghar
Nilkanth        Raigad
Prathamesh      Raigad
Sairaj  Satara
Time taken: 0.072 seconds, Fetched: 5 row(s)
```

5. **Display all information where customer cust_code and cust_name**

```
hive> select cid, cname
    > from customer_details;
OK
101     Shravan
102     Parth
103     Nilkanth
104     Prathamesh
105     Sairaj
Time taken: 0.067 seconds, Fetched: 5 row(s)
hive>
```

6. **Display the customer information for area_code=a101**

```
hive> select *
    > from customer_details
    > where pincode = 402109;
OK
101     Shravan Roha    Raigad  9876754885      402109
104     Prathamesh      Roha    Raigad  7564854456      402109
Time taken: 0.079 seconds, Fetched: 2 row(s)
hive>
```

7. **Display customer details where cid is ‘C101’ or ‘C201’**

```
hive> select *
    > from customer_details
    > where cid in (101,102);
OK
101     Shravan Roha    Raigad  9876754885      402109
102     Parth   Palghar Palghar 8756485851      401404
Time taken: 0.078 seconds, Fetched: 2 row(s)
hive>
```

8. **Display the city wise customer count.**

```
Total MapReduce CPU Time Spent: 3 seconds 570 msec
OK
Palghar 1
Panvel  1
Roha    2
Satara  1
Time taken: 18.34 seconds, Fetched: 4 row(s)
```

9. **Display the customers from city Roha, Panvel , Palghar**

```
hive> select *
    > from customer_details
    > where city in ('Roha','Panvel','Palghar');
OK
101     Shravan Roha    Raigad  9876754885      402109
102     Parth   Palghar Palghar 8756485851      401404
103     Nilkanth        Panvel  Raigad  7564899551      410210
104     Prathamesh      Roha    Raigad  7564854456      402109
Time taken: 0.087 seconds, Fetched: 4 row(s)
hive>
```

10. **Rename the table to customer_New**

```
hive> alter table customer_details rename to customer_new;
OK
Time taken: 0.067 seconds
hive> select * from customer_new;
OK
101     Shravan Roha    Raigad  9876754885      402109
102     Parth   Palghar Palghar 8756485851      401404
103     Nilkanth        Panvel  Raigad  7564899551      410210
104     Prathamesh      Roha    Raigad  7564854456      402109
105     Sairaj  Satara  Satara  8545866547      400214
Time taken: 0.07 seconds, Fetched: 5 row(s)
hive>
```

11. **Rename the column 'location' to 'Region'**

```
hive> ALTER TABLE customer_new
    > CHANGE COLUMN
    > location Region STRING;
OK
Time taken: 0.083 seconds
hive> describe customer_new;
OK
cid                     string
cname                   string
city                    string
region                  string
phone                   bigint
pincode                 string
Time taken: 0.025 seconds, Fetched: 6 row(s)
hive>
```

Q2) Perform the following in HIVE:

1. **Create a Emp Database**

```
hive> create database empDB;
OK
Time taken: 0.111 seconds
hive> show databases;
OK
customerdb
default
empdb
mydb
Time taken: 0.032 seconds, Fetched: 4 row(s)
hive> use empDB;
OK
Time taken: 0.017 seconds
hive>
```

2. **Create a table called 'Employee'**

```
hive> create table employee(
    > eid STRING,
    > ename STRING,
    > designation STRING,
    > salary INT,
    > did STRING)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ',';
OK
Time taken: 0.088 seconds
hive> describe employee;
OK
eid                     string
ename                   string
designation             string
salary                  int
did                     string
Time taken: 0.027 seconds, Fetched: 5 row(s)
hive>
```

3. Add 10 employees who have joined the company to thedatabase. (eid,ename,designation,salary,did)

	A	B	C	D	E
1	E101	Vinayak	Manager	7000	D001
2	E102	Samiksha	Developer	5000	D002
3	E103	Sairaj	Analyst	6000	D003
4	E104	Divya	Manager	8000	D001
5	E105	Deepika	Developer	5500	D002
6	E106	Purva	Analyst	4500	D003
7	E107	Bhakti	Developer	5200	D002
8	E108	Shivanshu	Analyst	4800	D003
9	E109	Lokesh	Manager	7500	D001
10	E110	Shruti	Developer	6200	D002
11					

```
hive> load data local inpath '/home/hadoop/Documents/emp_details.csv'
    > into table employee;
Loading data to table empdb.employee
OK
Time taken: 0.173 seconds
hive> select * from employee;
OK
E101    Vinayak Manager 7000    D001
E102    Samiksha        Developer       5000    D002
E103    Sairaj  Analyst 6000    D003
E104    Divya   Manager 8000    D001
E105    Deepika Developer       5500    D002
E106    Purva   Analyst 4500    D003
E107    Bhakti  Developer       5200    D002
E108    Shivanshu       Analyst 4800    D003
E109    Lokesh  Manager 7500    D001
E110    Shruti  Developer       6200    D002
Time taken: 0.087 seconds, Fetched: 10 row(s)
hive>
```

4. Create Dept employee (did, dname)

```
hive> create table dept(
    > did STRING,
    > dname STRING)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ',';
OK
Time taken: 0.067 seconds
```

5. Insert details of 3 departments

	A	B
1	D001	HR
2	D002	IT
3	D003	Finance
4		
5		

```
hive> load data local inpath '/home/hadoop/Documents/dept.csv'
    > into table dept;
Loading data to table empdb.dept
OK
Time taken: 0.528 seconds
hive> select * from dept;
OK
D001    HR
D002    IT
D003    Finance
Time taken: 0.091 seconds, Fetched: 3 row(s)
hive>
```

6. Display the department wise employee count.

```
hive> select d.dname, COUNT(e.eid)
    > from employee e
    > JOIN dept d ON e.did=d.did
    > GROUP BY d.dname;
```

```
Finance 3
HR      3
IT      4
Time taken: 34.164 seconds, Fetched: 3 row(s)
hive>
```

7. Rename the table Employee to new_emp

```
hive> ALTER TABLE employee RENAME TO new_emp;
OK
Time taken: 0.113 seconds
hive> show tables;
OK
dept
new_emp
Time taken: 0.019 seconds, Fetched: 2 row(s)
hive>
```

8. **Rename the column ‘designation to ‘job_title’**

```
hive> ALTER TABLE new_emp CHANGE COLUMN
    > designation job_title STRING;
OK
```

```
hive> describe new_emp;
OK
eid                     string
ename                   string
job_title               string
salary                  int
did                     string
Time taken: 0.024 seconds, Fetched: 5 row(s)
hive>
```

9. **Display the number of employees present in “HR” dept andsalary greater than 5000.**

```
hive> select COUNT(e.eid) from new_emp e
    > JOIN dept d on e.did=d.did
    > WHERE d.dname = 'HR' and e.salary>5000;
```

```
Total MapReduce CPU Time Spent: 4 seconds 460 msec
OK
3
Time taken: 27.165 seconds, Fetched: 1 row(s)
hive>
```

Q3) Implement in HIVE

1. **Create a table Book (aid,aname,city,pname,btitle,price,rating)**

```
hive> create database bookDB;
OK
Time taken: 0.045 seconds
hive> show databases;
OK
bookdb
customerdb
default
empdb
mydb
Time taken: 0.019 seconds, Fetched: 5 row(s)
hive> create table book(
    > aid STRING,
    > aname STRING,
    > city STRING,
    > pname STRING,
    > btitle STRING,
    > price FLOAT,
    > rating FLOAT)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ',';
OK
Time taken: 0.056 seconds
hive> describe book;
OK
aid                     string
aname                   string
city                    string
pname                   string
btitle                  string
price                   float
rating                  float
Time taken: 0.027 seconds, Fetched: 7 row(s)
hive>
```

2. Load the data from a .csv file

	A	B	C	D	E
1	Vinayak	Mumbai	Publisher A	Data Structure	1000
2	Samiksha	Indore	Publisher B	Learning HTML	350
3	Sairaj	NaviMumbai	Publisher C	Core Java	200
4	Divya	Pune	Publisher A	Python Programming	650
5	Deepika	Nasik	Publisher B	Javascript Essentials	500
6	Purva	Amravati	Publisher C	Data Structure	150
7	Bhakti	Nagpur	Publisher A	Learning HTML	800
8	Shivanshu	Mumbai	Publisher B	Core Java	350
9	Lokesh	Hyderbad	Publisher C	Python Programming	450
10	Shruti	Indore	Publisher A	Javascript Essentials	250
11					

```
hive> load data local inpath '/home/hadoop/Documents/books.csv'
    > into table book;
Loading data to table empdb.book
OK
Time taken: 0.566 seconds
hive> select * from book;
OK
1       Vinayak Mumbai  Publisher A     Data Structure  1000.0  4.9
2       Samiksha        Indore  Publisher B     Learning HTML   350.0   4.7
3       Sairaj  NaviMumbai      Publisher C     Core Java       200.0   3.8
4       Divya   Pune    Publisher A     Python Programming      650.0   4.2
5       Deepika Nasik   Publisher B     Javascript Essentials   500.0   3.5
6       Purva   Amravati        Publisher C     Data Structure  150.0   4.1
7       Bhakti  Nagpur  Publisher A     Learning HTML   800.0   3.3
8       Shivanshu       Mumbai  Publisher B     Core Java       350.0   3.9
9       Lokesh  Hyderbad        Publisher C     Python Programming      450.0   1.5
10      Shruti  Indore  Publisher A     Javascript Essentials   250.0   2.0
Time taken: 0.089 seconds, Fetched: 10 row(s)
hive>
```

3. Display the name of author whose Rating is less than 2

```
hive> select aname from book
    > where rating<3.5;
OK
Bhakti
Lokesh
Shruti
Time taken: 0.159 seconds, Fetched: 3 row(s)
hive>
```

4. Display the publisher wise count of authors

```
hive> SELECT pname, COUNT(DISTINCT aname) AS author_count
    > FROM Book
    > GROUP BY pname;
```

```
Publisher A     4
Publisher B     3
Publisher C     3
Time taken: 27.926 seconds, Fetched: 3 row(s)
hive>
```

5. **Rename the table to Book_Details**

```
hive> ALTER TABLE Book RENAME TO Book_Details;
OK
Time taken: 0.117 seconds
hive> show tables;
OK
book_details
dept
new_emp
Time taken: 0.051 seconds, Fetched: 3 row(s)
hive>
```

6. **Display the name of the book having the highest price.**

```
hive> SELECT btitle
    > FROM Book_Details
    > ORDER BY price DESC
    > LIMIT 1;
```

```
OK
Data Structure
Time taken: 20.029 seconds, Fetched: 1 row(s)
hive>
```

7. **Display the authors from city Mumbai, Delhi or Chennai**

```
hive> SELECT aname
    > FROM Book_Details
    > WHERE city IN ('Mumbai', 'Indore', 'Hyderabad');
OK
Vinayak
Samiksha
Shivanshu
Shruti
Time taken: 0.111 seconds, Fetched: 4 row(s)
hive>
```

8. **Rename the column 'aname' to 'Author_Name'**

```
hive> ALTER TABLE Book_Details CHANGE COLUMN aname Author_Name STRING;
OK
Time taken: 0.052 seconds
hive> describe book_details;
OK
aid                     string
author_name             string
city                    string
pname                   string
btitle                  string
price                   float
rating                  float
Time taken: 0.031 seconds, Fetched: 7 row(s)
hive>
```

9. **Create a view Author_View for all the authors in the city Pune**

```
hive> CREATE VIEW Author_View AS
    > SELECT *
    > FROM Book_Details
    > WHERE city = 'Pune';
OK
Time taken: 0.184 seconds
```

10. **Describe the view.**

```
hive> DESCRIBE Author_View;
OK
aid                     string
author_name             string
city                    string
pname                   string
btitle                  string
price                   float
rating                  float
Time taken: 0.021 seconds, Fetched: 7 row(s)
hive>
```

11. **Display the contents of the view.**

```
hive> SELECT * FROM Author_View;
OK
4       Divya   Pune    Publisher A     Python Programming      650.0   4.2
Time taken: 0.13 seconds, Fetched: 1 row(s)
hive>
```

Q4) Implement in HIVE Partition

1. **Create a table student with a partition on dname with the details (rno,name,marks,subject,dname)**

```
hive> CREATE TABLE student (
    >     rno INT,
    >     name STRING,
    >     marks INT,
    >     subject STRING
    > )
    > PARTITIONED BY (dname STRING)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > ;
OK
Time taken: 0.118 seconds
hive> describe student;
OK
rno                     int
name                    string
marks                   int
subject                 string
dname                   string

# Partition Information
# col_name              data_type               comment
dname                   string
Time taken: 0.155 seconds, Fetched: 9 row(s)
hive>
```

2. **Insert data for 5 students for two departments MCA and MBA**

	A	B	C	D
1	1	Vinayak	49	CSDF
2	2	Samiksha	47	DSCC
3	3	Sairaj	35	EH
4	4	Divya	31	STQA
5	5	Deepika	43	MC
6				

	A	B	C	D
1	1	Bhakti	38	Finance
2	2	Shruti	40	Marketing
3	3	Purva	35	Sales
4	4	Shivanshu	33	Economics
5	5	Lokesh	29	Hospitality
6				
7				

```
hive> load data local inpath '/home/hadoop/Documents/MCA.csv'
    > into table student
    > PARTITION (dname='MCA');
Loading data to table empdb.student partition (dname=MCA)
OK
Time taken: 0.761 seconds
hive> load data local inpath '/home/hadoop/Documents/MBA.csv'
    > into table student
    > PARTITION (dname='MBA');
Loading data to table empdb.student partition (dname=MBA)
OK
Time taken: 0.24 seconds
hive> select * from student;
OK
1       Bhakti  38      Finance MBA
2       Shruti  40      Marketing       MBA
3       Purva   35      Sales   MBA
4       Shivanshu       33      Economics       MBA
5       Lokesh  29      Hospitality     MBA
1       Vinayak 49      CSDF    MCA
2       Samiksha        47      DSCC    MCA
3       Sairaj  35      EH      MCA
4       Divya   31      STQA    MCA
5       Deepika 43      MC      MCA
Time taken: 0.207 seconds, Fetched: 10 row(s)
hive>
```

3. **Display the contents of each partition.**

```
hive> SELECT * FROM student WHERE dname = 'MCA';
OK
1       Vinayak 49      CSDF    MCA
2       Samiksha        47      DSCC    MCA
3       Sairaj  35      EH      MCA
4       Divya   31      STQA    MCA
5       Deepika 43      MC      MCA
Time taken: 0.243 seconds, Fetched: 5 row(s)
hive>
    >
    > SELECT * FROM student WHERE dname = 'MBA';
OK
1       Bhakti  38      Finance MBA
2       Shruti  40      Marketing       MBA
3       Purva   35      Sales   MBA
4       Shivanshu       33      Economics       MBA
5       Lokesh  29      Hospitality     MBA
Time taken: 0.11 seconds, Fetched: 5 row(s)
hive>
```

4. **Display students having marks more than 60 in MCA department.**

```
hive> SELECT * FROM student
    > WHERE dname = 'MCA' AND marks > 40;
OK
1       Vinayak 49      CSDF    MCA
2       Samiksha        47      DSCC    MCA
5       Deepika 43      MC      MCA
Time taken: 0.119 seconds, Fetched: 3 row(s)
hive> SELECT * FROM student
    > WHERE dname = 'MCA' AND marks < 40;
OK
3       Sairaj  35      EH      MCA
4       Divya   31      STQA    MCA
Time taken: 0.109 seconds, Fetched: 2 row(s)
hive>
```

5. **Add a new partition for the department MSc**

```
hive> ALTER TABLE student ADD PARTITION (dname='MSc');
OK
Time taken: 0.09 seconds
hive> show partitions student;
OK
dname=MBA
dname=MCA
dname=MSc
Time taken: 0.038 seconds, Fetched: 3 row(s)
hive>
```

6. **Perform the following built in functions – lower, upper, ltrim,reverse**

```
hive> SELECT LOWER(name) AS lower_name FROM student;
OK
bhakti
shruti
purva
shivanshu
lokesh
vinayak
samiksha
sairaj
divya
deepika
Time taken: 0.119 seconds, Fetched: 10 row(s)
```

```
    > SELECT UPPER(name) AS upper_name FROM student;
OK
BHAKTI
SHRUTI
PURVA
SHIVANSHU
LOKESH
VINAYAK
SAMIKSHA
SAIRAJ
DIVYA
DEEPIKA
Time taken: 0.093 seconds, Fetched: 10 row(s)
```

```
    > SELECT LTRIM(name) AS trimmed_name FROM student;
OK
Bhakti
Shruti
Purva
Shivanshu
Lokesh
Vinayak
Samiksha
Sairaj
Divya
Deepika
Time taken: 0.103 seconds, Fetched: 10 row(s)
```

```
    > SELECT REVERSE(name) AS reversed_name FROM student;
OK
itkahB
iturhS
avruP
uhsnavihS
hsekoL
kayaniV
ahskimaS
jariaS
ayviD
akipeeD
Time taken: 0.087 seconds, Fetched: 10 row(s)
hive>
```

PIG Assignment

Q1) Perform the following PIG

1. **Create a .csv file to store customer details- id,name ,item_purchased, quantity,phone,city.**

	A	B	C	D	E	F
1	1	Vinayak	Mouse	150	9876543119	Mumbai
2	2	Samiksha	Keyboard	50	9876543118	Pune
3	3	Sairaj	Monitor	200	9876543104	Mumbai
4	4	Lokesh	Mouse	500	9876543107	Pune
5	5	Divya	Laptop	2500	9876543123	Banglore
6	6	Deepika	Mouse	1800	9876543121	Mumbai
7	7	Purva	Headphone	100	9876543112	Delhi
8	8	Bhakti	Mouse	300	9876543111	Mumbai
9	9	Shivanshu	Keyboard	450	9876543113	Delhi
10	10	Jitesh	Monitor	50	9876543128	Mumbai
11						

2. **Create a relation CUSTOMER to store the details of this .csv file**

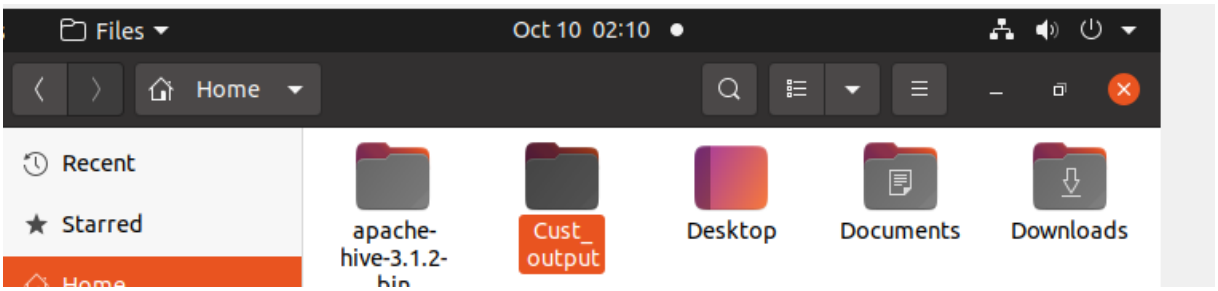
```
 since yarn.timeline-service.enabled set to false
grunt> Customer = LOAD '/home/hadoop/Documents/customer.csv' USING PigStorage('
,') AS (id:int, name:chararray, item_purchased:chararray, quantity:int, phone:c
hararray, city:chararray);
```

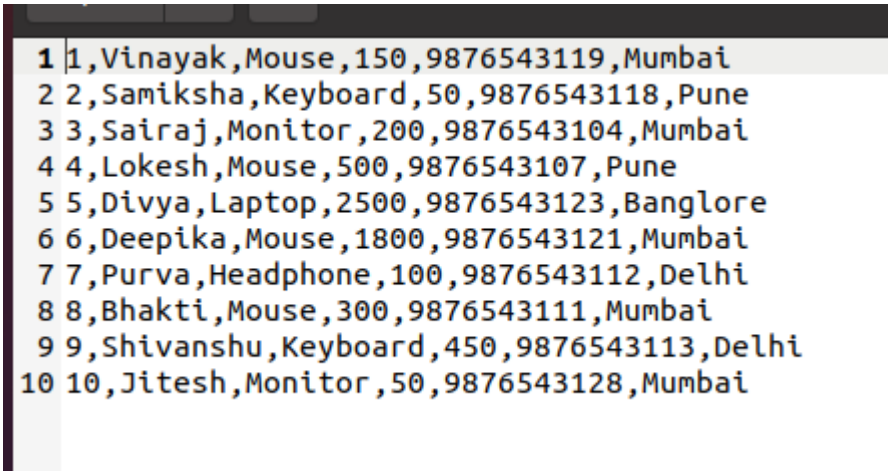
3. **Display the contents of this relation on screen**

```
ine.util.MapRedUtil - Total input paths to process : 1
(1,Vinayak,Mouse,150,9876543119,Mumbai)
(2,Samiksha,Keyboard,50,9876543118,Pune)
(3,Sairaj,Monitor,200,9876543104,Mumbai)
(4,Lokesh,Mouse,500,9876543107,Pune)
(5,Divya,Laptop,2500,9876543123,Banglore)
(6,Deepika,Mouse,1800,9876543121,Mumbai)
(7,Purva,Headphone,100,9876543112,Delhi)
(8,Bhakti,Mouse,300,9876543111,Mumbai)
(9,Shivanshu,Keyboard,450,9876543113,Delhi)
(10,Jitesh,Monitor,50,9876543128,Mumbai)
grunt>
```

4. **Store this relation data in local file system**

```
Details at logfile: /home/hadoop/pig_1728505386263.log
grunt> STORE Customer INTO '/home/hadoop/Cust_output' USING PigStorage(',');
2024-10-10 02:08:47,342 [main] INFO  org.apache.hadoop.conf.Configuration.depre
```





```
1 1,Vinayak,Mouse,150,9876543119,Mumbai
2 2,Samiksha,Keyboard,50,9876543118,Pune
3 3,Sairaj,Monitor,200,9876543104,Mumbai
4 4,Lokesh,Mouse,500,9876543107,Pune
5 5,Divya,Laptop,2500,9876543123,Banglore
6 6,Deepika,Mouse,1800,9876543121,Mumbai
7 7,Purva,Headphone,100,9876543112,Delhi
8 8,Bhakti,Mouse,300,9876543111,Mumbai
9 9,Shivanshu,Keyboard,450,9876543113,Delhi
10 10,Jitesh,Monitor,50,9876543128,Mumbai
```

5. **Display details of customers whose city is ‘Mumbai’;**

```
Details at logfile: /home/hadoop/pig_1728505386263.log
grunt> Customer_Mumbai = FILTER Customer BY city=='Mumbai';
2024-10-10 02:17:35,813 [main] INFO  org.apache.hadoop.conf.Co
grunt> DUMP Customer_Mumbai;
```

```
2024-10-10 02:18:06,140 [main] INFO  org.ap
(1,Vinayak,Mouse,150,9876543119,Mumbai)
(3,Sairaj,Monitor,200,9876543104,Mumbai)
(6,Deepika,Mouse,1800,9876543121,Mumbai)
(8,Bhakti,Mouse,300,9876543111,Mumbai)
(10,Jitesh,Monitor,50,9876543128,Mumbai)
grunt>
```

6. **Display id, name and city of all customers.**

```
grunt> Customer_Details = FOREACH Customer Generate id, name, city;
grunt> DUMP Customer_Details;
```

```
(1,Vinayak,Mumbai)
(2,Samiksha,Pune)
(3,Sairaj,Mumbai)
(4,Lokesh,Pune)
(5,Divya,Banglore)
(6,Deepika,Mumbai)
(7,Purva,Delhi)
(8,Bhakti,Mumbai)
(9,Shivanshu,Delhi)
(10,Jitesh,Mumbai)
grunt>
```

7. **Separate the contents of customer relation for quantity < 200 and >= 2000 to cust1 and cust2 respectively.**

```
grunt> cust1 = FILTER Customer BY quantity < 200;
grunt> cust2 = FILTER Customer BY quantity >= 2000;
grunt> DUMP cust1;
```

```
(1,Vinayak,Mouse,150,9876543119,Mumbai)
(2,Samiksha,Keyboard,50,9876543118,Pune)
(7,Purva,Headphone,100,9876543112,Delhi)
(10,Jitesh,Monitor,50,9876543128,Mumbai)
grunt>
```

```
(5,Divya,Laptop,2500,9876543123,Banglore)
grunt>
```

8. **Display the details of customers from city ‘Mumbai’ who purchased ‘Mouse’.**

```
grunt> Mumbai_Mouse = FILTER Customer BY city == 'Mumbai' AND item_purchased ==
'Mouse';
grunt> dump Mumbai_Mouse;
```

```
(1,Vinayak,Mouse,150,9876543119,Mumbai)
(6,Deepika,Mouse,1800,9876543121,Mumbai)
(8,Bhakti,Mouse,300,9876543111,Mumbai)
grunt>
```

Q2) Perform the following in PIG

1. **Create emp.txt file with 6 records, file with following fields- eno, name, city, salary,did**

```
1,Vinayak,Chennai,50000,101
2,Samiksha,Mumbai,60000,102
3,Sairaj,Chennai,55000,101
4,Lokesh,Delhi,70000,103
5,Divya,Chennai,50000,101
6,Deepika,Delhi,80000,103
7,Jitesh,Mumbai,90000,102
8,Shivanshu,Pune,60000,101
9,Purva,Pune,85000,103
10,Bhakti,Bangalore,75000,102
```

2. **Create dept.txt file with three departments sales', 'IT','Marketing' with the fields- did, dname, location**

```
101,Sales,Chennai
102,IT,Mumbai
103,Marketing,Delhi
```

3. **. Create a relation Employee for the data given in emp .txt**

```
grunt> Employee = LOAD '/home/hadoop/emp' USING PigStorage(',') AS (eno:int, nam
e:chararray, city:chararray, salary:int, did:int);
2024-10-11 12:24:27,586 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
grunt> DUMP Employee;
```

```
(1,Vinayak,Chennai,50000,101)
(2,Samiksha,Mumbai,60000,102)
(3,Sairaj,Chennai,55000,101)
(4,Lokesh,Delhi,70000,103)
(5,Divya,Chennai,50000,101)
(6,Deepika,Delhi,80000,103)
(7,Jitesh,Mumbai,90000,102)
(8,Shivanshu,Pune,60000,101)
(9,Purva,Pune,85000,103)
(10,Bhakti,Bangalore,75000,102)
grunt>
```

4. **Create a relation Department and insert 5 records.**

```
grunt> Department = LOAD '/home/hadoop/dept' USING PigStorage(',') AS (did:int,
dname:chararray, location:chararray);
2024-10-11 12:26:40,009 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
grunt> Dump Department;
```

```
(101,Sales,Chennai)
(102,IT,Mumbai)
(103,Marketing,Delhi)
grunt>
```

5. **Display all the employees from city "Chennai"**

```
grunt> ChennaiEmployees = FILTER Employee BY city == 'Chennai';
grunt> DUMP ChennaiEmployees;
```

```
(1,Vinayak,Chennai,50000,101)
(3,Sairaj,Chennai,55000,101)
(5,Divya,Chennai,50000,101)
grunt>
```

6. **Display name of employees with their department name**

```
grunt> EmployeeDept = JOIN Employee BY did, Department BY did;
grunt> EmployeeNameDept = FOREACH EmployeeDept GENERATE Employee::name, Departme
nt::dname;
grunt> DUMP EmployeeNameDept;
```

```
(Shivanshu,Sales)
(Divya,Sales)
(Sairaj,Sales)
(Vinayak,Sales)
(Bhakti,IT)
(Jitesh,IT)
(Samiksha,IT)
(Purva,Marketing)
(Deepika,Marketing)
(Lokesh,Marketing)
grunt>
```

7. **Sort the employee details according to their name in descending**

```
grunt> SortedEmployee = ORDER Employee BY name DESC;
grunt> DUMP SortedEmployee;
```

```
(1,Vinayak,Chennai,50000,101)
(8,Shivanshu,Pune,60000,101)
(2,Samiksha,Mumbai,60000,102)
(3,Sairaj,Chennai,55000,101)
(9,Purva,Pune,85000,103)
(4,Lokesh,Delhi,70000,103)
(7,Jitesh,Mumbai,90000,102)
(5,Divya,Chennai,50000,101)
(6,Deepika,Delhi,80000,103)
(10,Bhakti,Bangalore,75000,102)
grunt>
```

8. **Display total number of employees.**

```
grunt> TotalEmployees = FOREACH (GROUP Employee ALL) GENERATE COUNT(Employee);
grunt> DUMP TotalEmployees;
```

```
(10)
grunt>
```

9. **Display the department wise employee count.**

```
grunt> DeptWiseCount = FOREACH (GROUP Employee BY did) GENERATE group, COUNT(Emp
loyee);
grunt> DUMP DeptWiseCount;
```

```
(101,4)
(102,3)
(103,3)
grunt>
```

10. **Display employee and their department details**

```
grunt> EmployeeDetails = JOIN Employee BY did, Department BY did;
grunt> DUMP EmployeeDetails;
```

```
(8,Shivanshu,Pune,60000,101,101,Sales,Chennai)
(5,Divya,Chennai,50000,101,101,Sales,Chennai)
(3,Sairaj,Chennai,55000,101,101,Sales,Chennai)
(1,Vinayak,Chennai,50000,101,101,Sales,Chennai)
(10,Bhakti,Bangalore,75000,102,102,IT,Mumbai)
(7,Jitesh,Mumbai,90000,102,102,IT,Mumbai)
(2,Samiksha,Mumbai,60000,102,102,IT,Mumbai)
(9,Purva,Pune,85000,103,103,Marketing,Delhi)
(6,Deepika,Delhi,80000,103,103,Marketing,Delhi)
(4,Lokesh,Delhi,70000,103,103,Marketing,Delhi)
grunt>
```

11. **Perform Left Outer Join**

```
grunt> LeftJoin = JOIN Employee BY did LEFT OUTER, Department BY did;
grunt> DUMP LeftJoin;
```

```
(8,Shivanshu,Pune,60000,101,101,Sales,Chennai)
(5,Divya,Chennai,50000,101,101,Sales,Chennai)
(3,Sairaj,Chennai,55000,101,101,Sales,Chennai)
(1,Vinayak,Chennai,50000,101,101,Sales,Chennai)
(10,Bhakti,Bangalore,75000,102,102,IT,Mumbai)
(7,Jitesh,Mumbai,90000,102,102,IT,Mumbai)
(2,Samiksha,Mumbai,60000,102,102,IT,Mumbai)
(9,Purva,Pune,85000,103,103,Marketing,Delhi)
(6,Deepika,Delhi,80000,103,103,Marketing,Delhi)
(4,Lokesh,Delhi,70000,103,103,Marketing,Delhi)
grunt>
```

Q3) . Perform the following operations in PIG

1. **Create student.txt file with 10 records, file with following fields- Sid, sname, Saddress,cid**

```
 1 1,Vinayak,Delhi,101
 2 2,Samiksha,Delhi,102
 3 3,Sairaj,Mumbai,101
 4 4,Deepika,Bangalore,103
 5 5,Yash,Delhi,101
 6 6,Divya,Hyderabad,102
 7 7,Lokesh,Pune,103
 8 8,Purva,Delhi,101
 9 9,Bhakti,Mumbai,102
10 10,Shivanshu,Bangalore,103
```

2. **Create course.txt file for ‘Java’, ADBMS’ and ’BDAV’ courses, with the fields- cid ,cname,fees**

```
1 101,Java,15000
2 102,ADBMS,12000
3 103,BDAV,13000
```

3. **Load the above file details into the relations STUDENT and COURSE.**

```
grunt> STUDENT = LOAD 'student.txt' USING PigStorage(',') AS (Sid:int, sname:cha
rarray, Saddress:chararray, cid:int);
2024-10-11 12:47:28,812 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
grunt> DUMP STUDENT;
```

```
(1,Vinayak,Delhi,101)
(2,Samiksha,Delhi,102)
(3,Sairaj,Mumbai,101)
(4,Deepika,Bangalore,103)
(5,Yash,Delhi,101)
(6,Divya,Hyderabad,102)
(7,Lokesh,Pune,103)
(8,Purva,Delhi,101)
(9,Bhakti,Mumbai,102)
(10,Shivanshu,Bangalore,103)
grunt>
```

```
grunt> COURSE = LOAD 'course' USING PigStorage(',')  AS (cid:int, cname:chararra
y, fees:int);
2024-10-11 12:49:25,687 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
grunt> DUMP COURSE;
```

```
(101,Java,15000)
(102,ADBMS,12000)
(103,BDAV,13000)
grunt>
```

4. **Display course wise student count.**

```
grunt> COURSE_WISE_COUNT = GROUP STUDENT BY cid;
grunt> STUDENT_COUNT = FOREACH COURSE_WISE_COUNT GENERATE group AS cid, COUNT(ST
UDENT) AS student_count;
grunt> DUMP STUDENT_COUNT;
```

```
(101,4)
(102,3)
(103,3)
grunt>
```

5. **Display the student name and the course applied by each student**

```
grunt> JOINED = JOIN STUDENT BY cid, COURSE BY cid;
grunt> STUDENT_COURSE = FOREACH JOINED GENERATE STUDENT::sname AS student_name,
COURSE::cname AS course_name;
grunt> DUMP STUDENT_COURSE;
```

```
(Purva,Java)
(Yash,Java)
(Sairaj,Java)
(Vinayak,Java)
(Bhakti,ADBMS)
(Divya,ADBMS)
(Samiksha,ADBMS)
(Shivanshu,BDAV)
(Lokesh,BDAV)
(Deepika,BDAV)
grunt>
```

6. **Display sname and their cname.**

```
grunt> SNAME_CNAME = FOREACH JOINED GENERATE STUDENT::sname AS sname, COURSE::cn
ame AS cname;
grunt> DUMP SNAME_CNAME;
```

```
(Purva,Java)
(Yash,Java)
(Sairaj,Java)
(Vinayak,Java)
(Bhakti,ADBMS)
(Divya,ADBMS)
(Samiksha,ADBMS)
(Shivanshu,BDAV)
(Lokesh,BDAV)
(Deepika,BDAV)
grunt>
```

7. **Write a Pig script to perform the following operations:**

a. **Display the contents of STUDENT and COURSE relation**

```
(1,Vinayak,Delhi,101)
(2,Samiksha,Delhi,102)
(3,Sairaj,Mumbai,101)
(4,Deepika,Bangalore,103)
(5,Yash,Delhi,101)
(6,Divya,Hyderabad,102)
(7,Lokesh,Pune,103)
(8,Purva,Delhi,101)
(9,Bhakti,Mumbai,102)
(10,Shivanshu,Bangalore,103)
grunt>
```

```
(101,Java,15000)
(102,ADBMS,12000)
(103,BDAV,13000)
grunt>
```

b. Display the sid and sname who lives in “Delhi”

```
grunt> DELHI_STUDENTS = FILTER STUDENT BY Saddress == 'Delhi';
grunt> DUMP DELHI_STUDENTS;
```

```
(1,Vinayak,Delhi,101)
(2,Samiksha,Delhi,102)
(5,Yash,Delhi,101)
(8,Purva,Delhi,101)
grunt>
```

c. Display student details in ascending order of their name

```
grunt> SORTED_STUDENTS = ORDER STUDENT BY sname ASC;
grunt> DUMP SORTED_STUDENTS;
```

```
(9,Bhakti,Mumbai,102)
(4,Deepika,Bangalore,103)
(6,Divya,Hyderabad,102)
(7,Lokesh,Pune,103)
(8,Purva,Delhi,101)
(3,Sairaj,Mumbai,101)
(2,Samiksha,Delhi,102)
(10,Shivanshu,Bangalore,103)
(1,Vinayak,Delhi,101)
(5,Yash,Delhi,101)
grunt>
```

Apache Spark Lab Assignment

Q1. Create the following Text File and perform the operations:

1. Student_details(sid,sname,course,did,dname)

```
scala> val stud_det=spark.read.csv("/home/hadoop/Documents/stud.csv")
stud_det: org.apache.spark.sql.DataFrame = [_c0: string, _c1: string ... 3 more fields]

scala> stud_det.printSchema()
root
 |-- _c0: string (nullable = true)
 |-- _c1: string (nullable = true)
 |-- _c2: string (nullable = true)
 |-- _c3: string (nullable = true)
 |-- _c4: string (nullable = true)
```

2. Create a dataframe to read the text file

```
scala> val stud_det=spark.read.csv("/home/hadoop/Documents/stud.csv")
stud_det: org.apache.spark.sql.DataFrame = [_c0: string, _c1: string ... 3 more fields]

scala> stud_det.printSchema()
root
 |-- _c0: string (nullable = true)
 |-- _c1: string (nullable = true)
 |-- _c2: string (nullable = true)
 |-- _c3: string (nullable = true)
 |-- _c4: string (nullable = true)
```

3. Display the schema of the dataframe

```
scala> stud_det.show
+---+-------+---+---+-------+
|_c0|    _c1|_c2|_c3|    _c4|
+---+-------+---+---+-------+
|  1| vedika|MCA|  1| BVIMIT|
|  2|    ved| CS|  2| MVIMSR|
|  3|    dev| IT|  3|BHARATI|
|  4|sanjana|MCA|  1| BVIMIT|
|  5|  sneha|MBA|  1| BVIMIT|
+---+-------+---+---+-------+
```

4. Create a view “Stud_View” using the above dataframe

```
scala> stud_det.createOrReplaceTempView("BVI")
```

5. Display student name,dname from the above view

```
scala> val stud_deta = spark.sql("Select * from BVI where _c3>1")
stud_deta: org.apache.spark.sql.DataFrame = [_c0: string, _c1: string ... 3 more fields]

scala> stud_deta.show()
+---+---+---+---+-------+
|_c0|_c1|_c2|_c3|    _c4|
+---+---+---+---+-------+
|  2|ved| CS|  2| MVIMSR|
|  3|dev| IT|  3|BHARATI|
+---+---+---+---+-------+
```

6. Describe the structure of the view

```
scala> stud_deta.show()
+---+---+---+---+-------+
|_c0|_c1|_c2|_c3|    _c4|
+---+---+---+---+-------+
|  2|ved| CS|  2| MVIMSR|
|  3|dev| IT|  3|BHARATI|
+---+---+---+---+-------+
```

Q2. Process the following in Apache Spark:

1. Create dataframe from json file which contains student data

```
scala> val df=spark.read.json("/home/hadoop/Documents/stud_data.json")
df: org.apache.spark.sql.DataFrame = [_corrupt_record: string]

scala> val df1=spark.read.option("multiline", "true").json("/home/hadoop/Documents/stud_data.json")
df1: org.apache.spark.sql.DataFrame = [Name: string, course: string ... 2 more fields]

scala> df1.show()
+------+------+-----+------+
|  Name|course|marks|rollno|
+------+------+-----+------+
|   ved|   MCA|   10|     1|
|   dev|   MBA|   15|     2|
|vedika|    IT|   20|     3|
+------+------+-----+------+
```

2. Print the schema in a tree format

```
scala> df1.printSchema()
root
 |-- Name: string (nullable = true)
 |-- course: string (nullable = true)
 |-- marks: long (nullable = true)
 |-- rollno: long (nullable = true)
```

3. Select only the "name" column

```
scala> df1.select("Name").show()
+------+
|  Name|
+------+
|   ved|
|   dev|
|vedika|
+------+
```

4.Count students by their course

```
scala> df1.groupBy("course").count().show()
+------+-----+
|course|count|
+------+-----+
|   MCA|    1|
|    IT|    1|
|   MBA|    1|
+------+-----+
```

5.Display students having marks less than 50

```
scala> df1.filter(df1.col("marks") < 15).show()
+----+------+-----+------+
|Name|course|marks|rollno|
+----+------+-----+------+
| ved|   MCA|   10|     1|
+----+------+-----+------+
```

Q3. Process the following in Apache Spark:

1. Consider the Employee.json file and save each of the following output in csv file.

```
scala> val data=spark.read.option("multiline", "true").json("/home/hadoop/Documents/emp.json")
data: org.apache.spark.sql.DataFrame = [Name: string, id: bigint ... 1 more field]
```

2. Displays the content of the DataFrame to stdout

```
scala> data.show()
+------+---+------+
|  Name| id|salary|
+------+---+------+
|   ved|  1| 50000|
|   dev|  2| 15000|
|vedika|  3| 70000|
+------+---+------+
```

3. Print the schema in a tree format

```
scala> data.printSchema()
root
 |-- Name: string (nullable = true)
 |-- id: long (nullable = true)
 |-- salary: long (nullable = true)
```

4. Select only the “salary" column.

```
scala> data.select(("salary")).show()
+------+
|salary|
+------+
| 50000|
| 15000|
| 70000|
+------+
```

5. Register the DataFrame as a SQL temporary view and display all information

```
scala> data.createOrReplaceTempView("emp1")
```

6. Using the same dataframe display employeeid and employee_name from the view

```
scala> val emp = spark.sql("Select Name,id from emp1")
emp: org.apache.spark.sql.DataFrame = [Name: string, id: bigint]


scala> emp.show()
+------+---+
|  Name| id|
+------+---+
|   ved|  1|
|   dev|  2|
|vedika|  3|
+------+---+
```

Q4. Implement Word count program in Spark