VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Big Data Analytics (23CS6PCBDA)

Submitted by:

Rani Aishwarya H S (1BM22CS217)

Under the Guidance of
Vikranth B.M.
Assistant Professor, BMSCE

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
March 2024 - June 2024
B. M. S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "**Big Data Analytics**" carried out by **Rani Aishwarya H S (1BM22CS217),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2024. The Lab report has been approved as it satisfies the academic requirements in respect of **Big Data Analytics –(23CS6PCBDA)** work prescribed for the said degree.

Vikranth B.M.
Associate Professor
Department of CSE
BMSCE, Bengaluru

Dr. Kavitha sooda
Professor and Head
Department of CSE
BMSCE, Bengaluru

Table Of Contents

Sl.no	Program details	Pg no
1	MongoDB- CRUD Operations Demonstration (Practice and Self Study)	5-9
2	Perform the DB operations using Cassandra.	10-17
3	Perform the DB operations using Cassandra	18-20
4	Execution of HDFS Commands for interaction with Hadoop Environment. (Minimum 10 commands to be executed)	21-23
5	Implement Wordcount program on Hadoop framework	24-27
6	a)Create a MapReduce program to find average temperature for each year from the NCDC data set. b) find the mean max temperature for every month.	28-30
7	For a given Text file, Create a Map Reduce program to sort the content in an alphabetic order listing only top 10 maximum occurrences of words.	31-32
8	Write a Scala program to print numbers from 1 to 100 using a for loop.	33-34
9	Using RDD and FlatMap count how many times each word appears in a file and write out a list of words whose count is strictly greater than 4 using Spark.	35-36

Course Outcomes

CO1: Apply the concepts of NoSQL, Hadoop, Spark for a given task

CO2: Analyse data analytic techniques for a given problem.

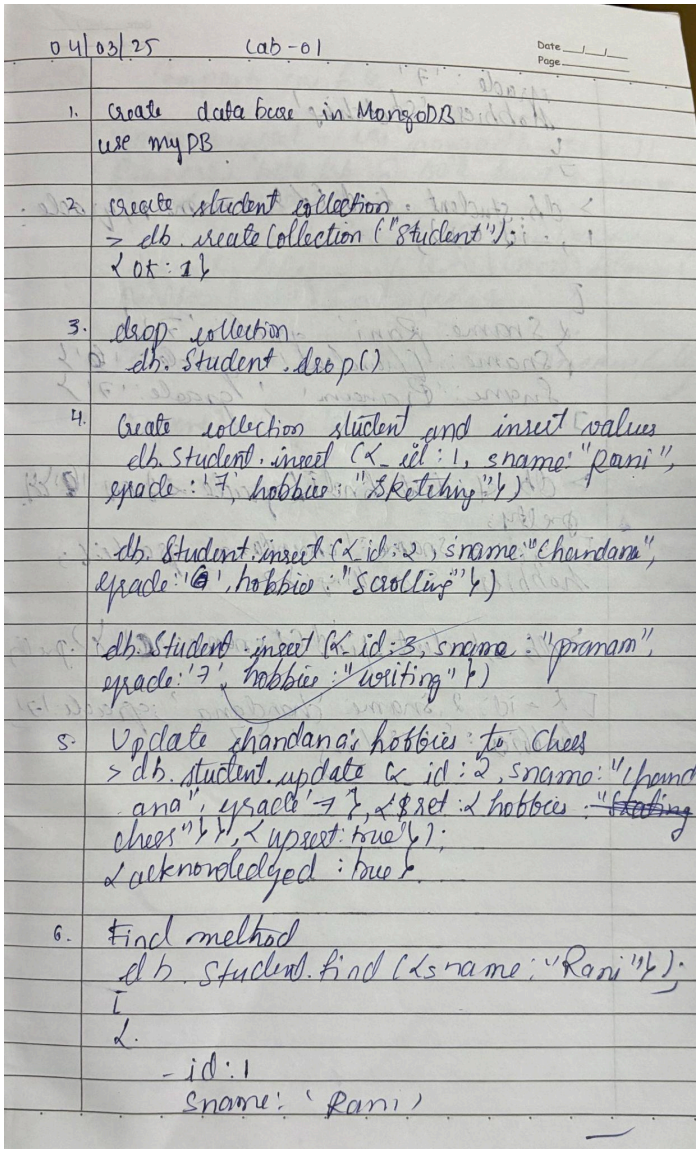
CO3: Conduct experiments using data analytics mechanisms for a given problem.

Program 1

MongoDB- CRUD Operations Demonstration (Practice and Self Study)

- Created a database named **myDB** and verified its existence.
- Created and dropped collections like **Student** and **Students**.
- Inserted student data into collections.
- Performed **upsert** to insert or update a student record.
- Used to find queries with various filters: by name, grade, hobbies, regex, etc.
- Retrieved specific fields while suppressing `_id`.
- Counted total documents and documents with specific criteria.
- Sorted records in ascending and descending order.
- Imported data from a CSV file and exported data to a CSV file.
- Used `save()` to insert or replace documents.
- Added, removed, and set fields to `null` in documents.
- Retrieved limited records and skipped initial entries.
- Created a **food** collection with arrays and queried arrays by value, index, size, etc.
- Updated specific elements in an array.
- Practiced query optimizations using `$in`, `$all`, `$ne`, `$regex`, `$slice`, and more.

Observation:



04/03/25 Lab-01

1. Create database in MongoDB
use myDB

2. create student collection
> db.createCollection("Student");
{ ok: 1 }

3. drop collection
db.Student.drop()

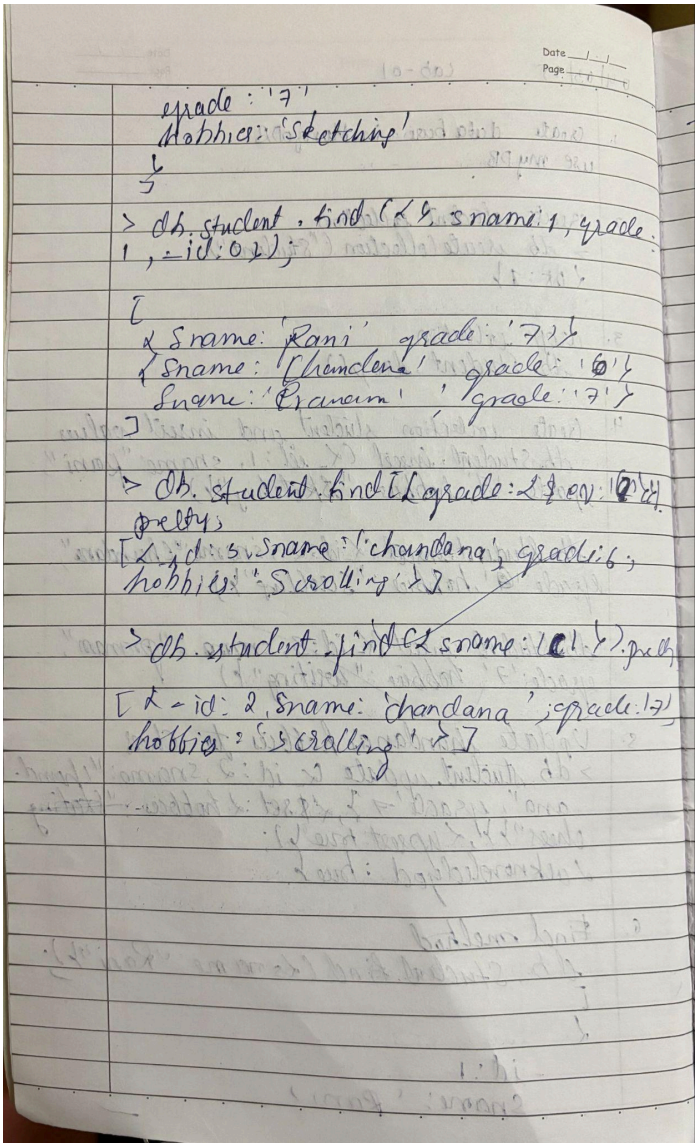
4. Create collection student and insert values
db.Student.insert({_id:1, sname:"Rani", grade:'7', hobbies:"Sketching"})

db.Student.insert({_id:2, sname:"chandana", grade:'6', hobbies:"Scrolling"})

db.Student.insert({_id:3, sname:"pranam", grade:'7', hobbies:"writing"})

5. Update chandana's hobbies to chess
> db.student.update({_id:2, sname:"chandana", grade:'7'}, {$set:{hobbies:"chess"}}, {upsert:true});
{ acknowledged: true }

6. Find method
db.Student.find({sname:"Rani"});
[
{
_id:1
sname:'Rani'

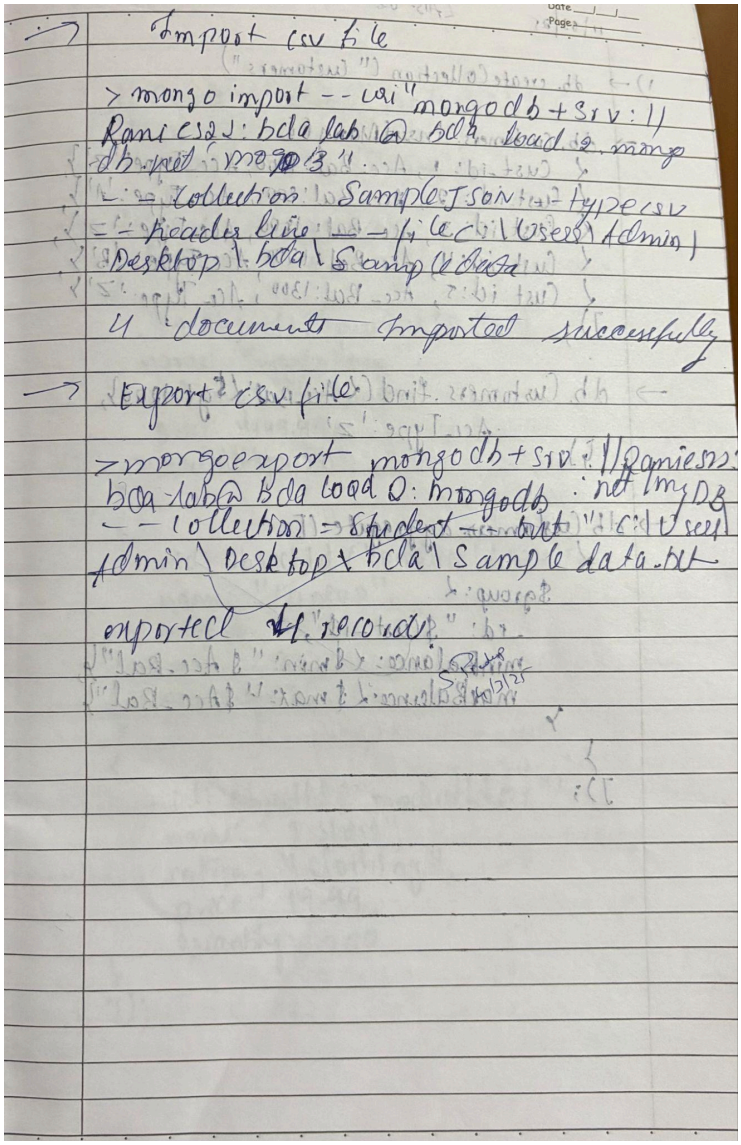


grade:'7'
Hobbies:'Sketching'
}

> db.student.find({}, {sname:1, grade:1, _id:0});
[
{sname:'Rani', grade:'7'}
{sname:'Chandana', grade:'6'}
{sname:'Pranam', grade:'7'}
]

> db.student.find({grade:{$eq:'6'}}).pretty();
[{_id:2, sname:'chandana', grade:6, hobbies:'Scrolling'}]

> db.student.find({sname:'Rani'}).pretty();
[{_id:2, sname:'chandana', grade:'7', hobbies:'Scrolling'}]



→ Import csv file

> mongoimport --uri "mongodb+srv://Ranics22:bda lab@bda load.2.mongodb.net/mydb" --collection=SampleJSON --type=csv --file="C:\Users\Admin\Desktop\bda\Sample data.txt"

4 documents imported successfully

→ Export csv file

>mongoexport mongodb+srv://Ranics22:bda lab@bda load 0.mongodb.net/mydb --collection=Student --out "C:\Users\Admin\Desktop\bda\Sample data.txt"

exported 4 records

Code with Output:

```
Atlas atlas-wanmtx-shard-0 [primary] Student> use Students
switched to db Students
Atlas atlas-wanmtx-shard-0 [primary] Students> show collections

Atlas atlas-wanmtx-shard-0 [primary] Students> db.students.insertMany([
...     { "Rollno": 10, "Name": "John", "Age": 20, "ContactNo": "1234567890", "Email-Id":
"john@example.com", "grade": "A", "hobby": "Reading" },
...     { "Rollno": 11, "Name": "Alice", "Age": 21, "ContactNo": "9876543210", "Email-Id":
 "alice@example.com", "grade":
 "B", "hobby": "Painting" },
...     { "Rollno": 12, "Name": "Bob", "Age": 22, "ContactNo": "2345678901", "Email-Id": "
bob@example.com", "grade": "C", "hobby": "Cooking" },
...     { "Rollno": 13, "Name": "Eve", "Age": 23, "ContactNo": "3456789012", "Email-Id": "
eve@example.com", "grade": "A"
 },
...     { "Rollno": 14, "Name": "Charlie", "Age": 24, "ContactNo": "4567890123", "Email-Id
": "charlie@example.com", "hobby": "Gardening" }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("661ce9dc76a00ff8cc51dae1"),
    '1': ObjectId("661ce9dc76a00ff8cc51dae2"),
    '2': ObjectId("661ce9dc76a00ff8cc51dae3"),
    '3': ObjectId("661ce9dc76a00ff8cc51dae4"),
    '4': ObjectId("661ce9dc76a00ff8cc51dae5")
  }
}
```

```
Atlas atlas-wanmtx-shard-0 [primary] Students> db.students.updateOne(
...     { "Rollno": 10 },
...     { $set: { "Email-Id": "john.doe@example.com" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
Atlas atlas-wanmtx-shard-0 [primary] Students> db.students.updateOne
...     { "Rollno": 11 },
...     { $set: { "Name": "Alicee" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
Atlas atlas-wanmtx-shard-0 [primary] Students> db.students.find({ "hobby": { $nin: ["Chess
", "Skating"] } })
[
  {
    _id: ObjectId("661ce9dc76a00ff8cc51dae1"),
    Rollno: 10,
    Name: 'John',
    Age: 20,
    ContactNo: '1234567890',
    'Email-Id': 'john.doe@example.com',
    grade: 'A',
    hobby: 'Reading'
  },
  {
    _id: ObjectId("661ce9dc76a00ff8cc51dae2"),
    Rollno: 11,
    Name: 'Alicee',
    Age: 21,
    ContactNo: '9876543210',
    'Email-Id': 'alice@example.com',
    grade: 'B',
    hobby: 'Painting'
  },
  {
    _id: ObjectId("661ce9dc76a00ff8cc51dae3"),
    Rollno: 12,
    Name: 'Bob',
    Age: 22,
    ContactNo: '2345678901',
    'Email-Id': 'bob@example.com',
    grade: 'C',
    hobby: 'Cooking'
  },
```

```
Atlas atlas-wanmtx-shard-0 [primary] Students> db.students.find({}, { "Name": 1, "grade":
{ $ifNull: ["$grade", "Not available"] }, "_id": 0 })
[
  { Name: 'John', grade: 'A' },
  { Name: 'Alicee', grade: 'B' },
  { Name: 'Bob', grade: 'C' },
  { Name: 'Eve', grade: 'A' },
  { Name: 'Charlie', grade: 'Not available' }
]
```

Program 2

Perform the following DB operations using Cassandra.

a) Create a keyspace by name Employee

b) Create a column family by name

Employee-Info with attributes Emp_Id Primary Key, Emp_Name, Designation, Date_of_Joining, Salary, Dept_Name

c) Insert the values into the table in batch

d) Update Employee name and Department of Emp-Id 121

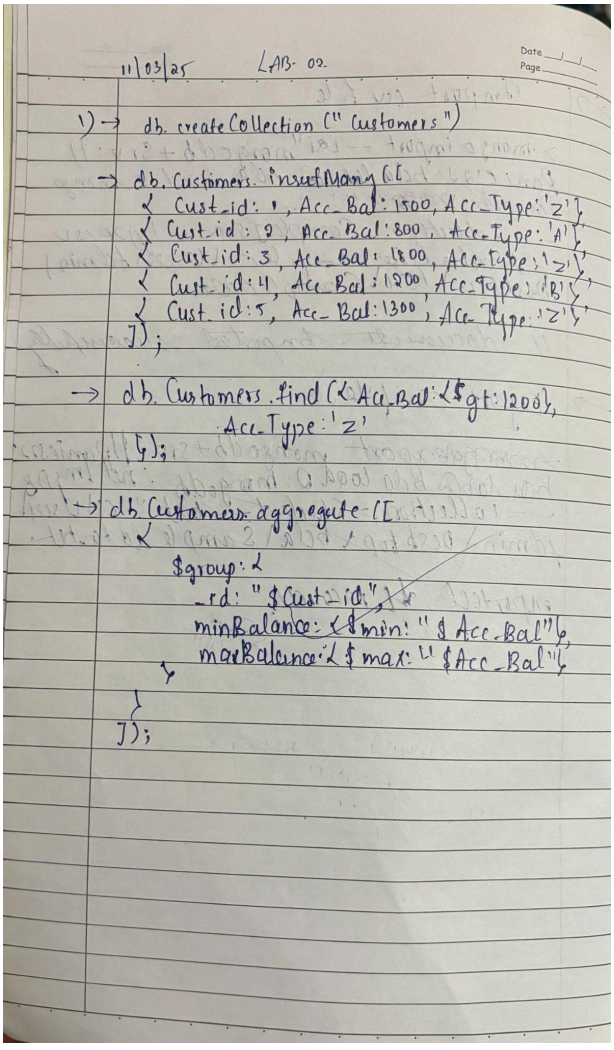
e) Sort the details of Employee records based on salary

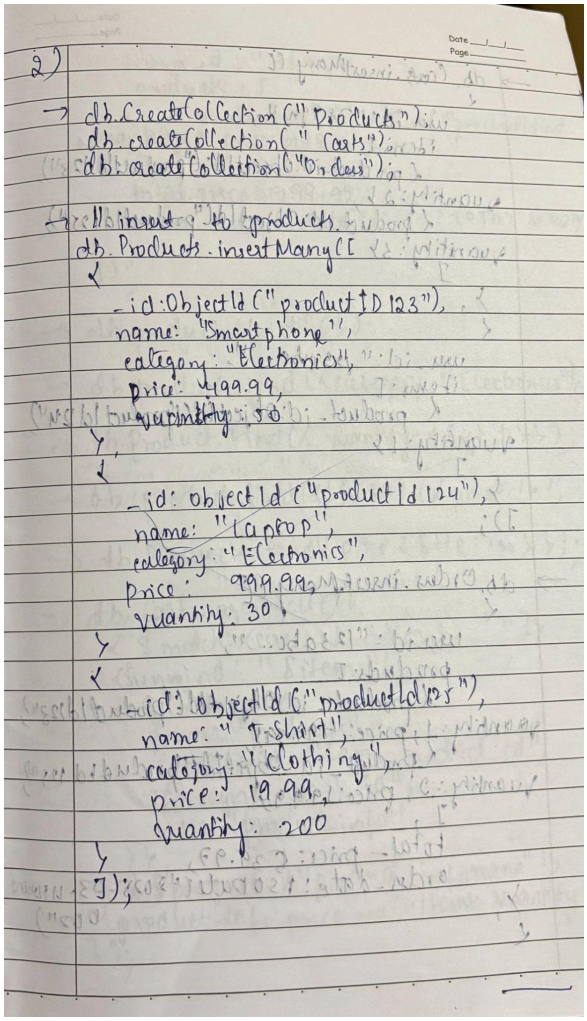
f) Alter the schema of the table Employee_Info to add a column Projects which stores a set of Projects done by the corresponding Employee.

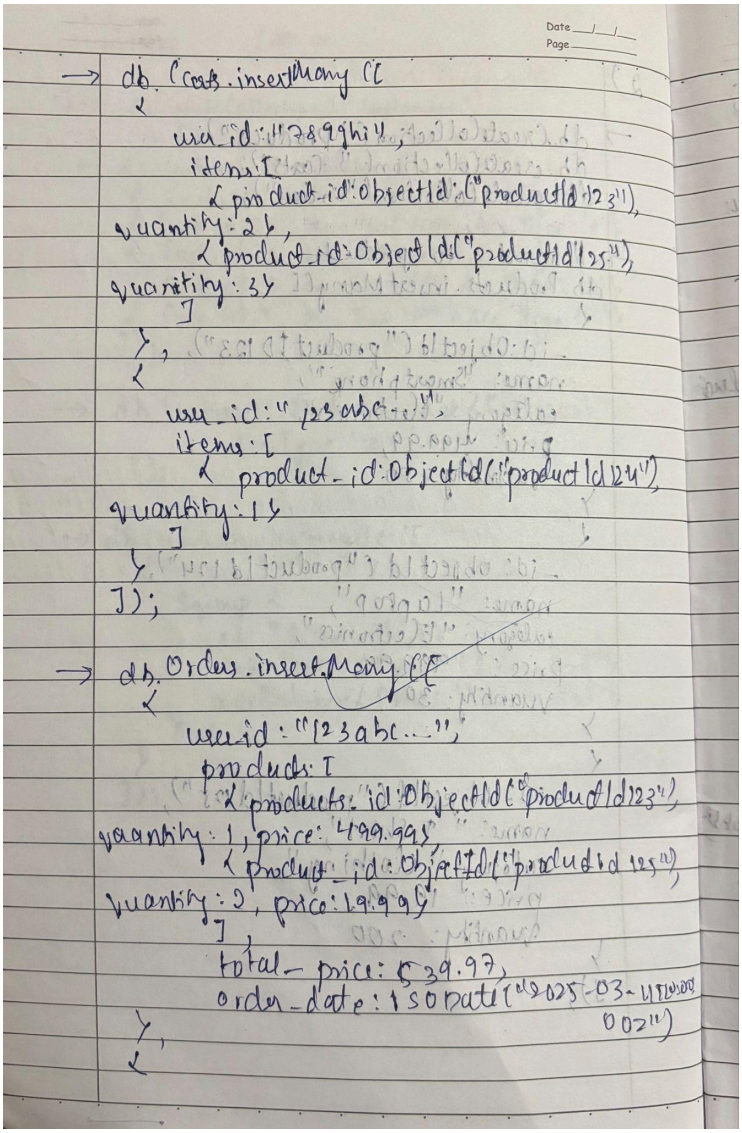
g) Update the altered table to add project names.

h) Create a TTL of 15 seconds to display the values ofEmployees.

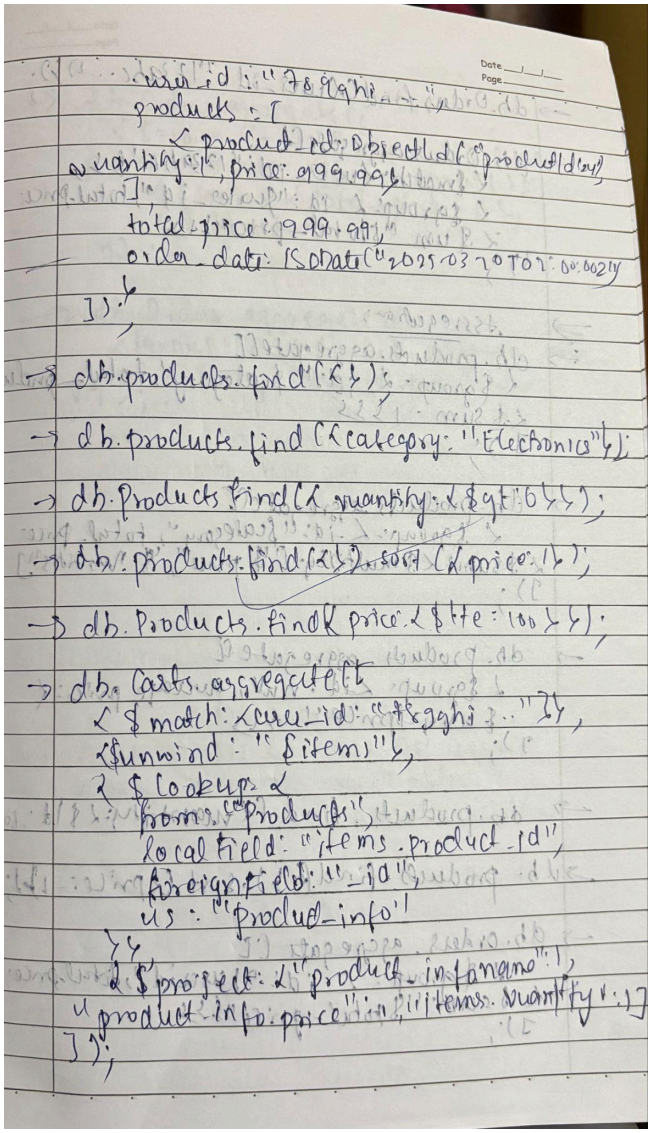
Observation:



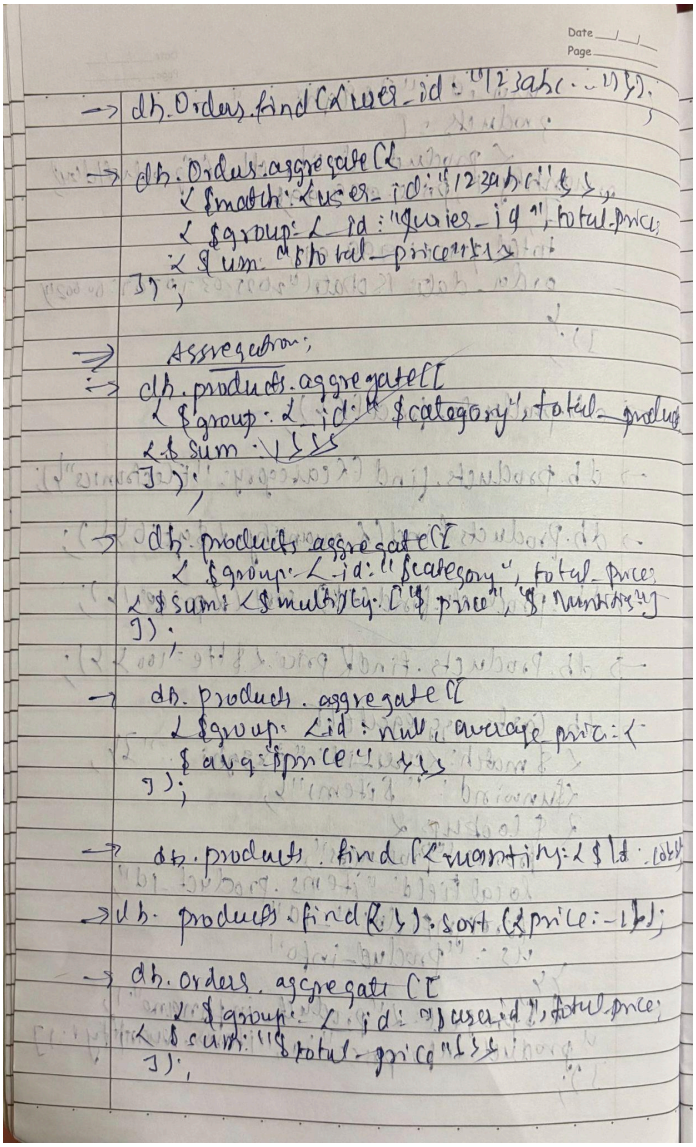




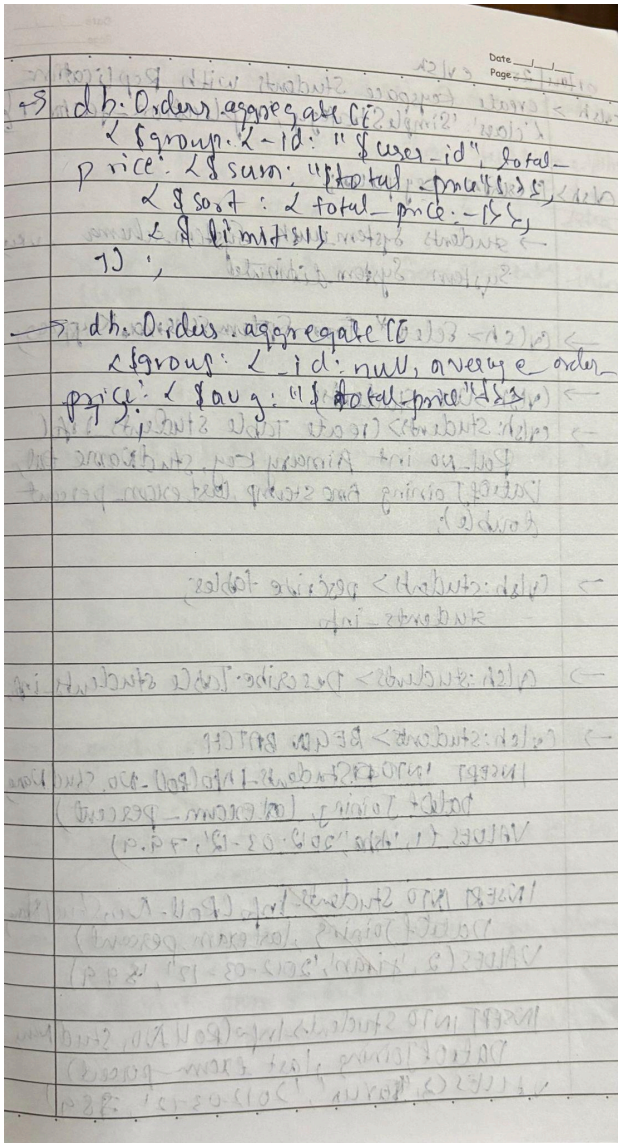
```
db.Carts.insertMany([
  {
    user_id: "789ghi",
    items: [
      { product_id: ObjectId("productId123"),
        quantity: 2 },
      { product_id: ObjectId("productId125"),
        quantity: 3 }
    ]
  },
  {
    user_id: "123abc...",
    items: [
      { product_id: ObjectId("productId124"),
        quantity: 1 }
    ]
  }
]);

db.Orders.insertMany([
  {
    user_id: "123abc...",
    products: [
      { products_id: ObjectId("productId123"),
        quantity: 1, price: 499.99 },
      { product_id: ObjectId("productId125"),
        quantity: 2, price: 19.99 }
    ],
    total_price: 539.97,
    order_date: ISODate("2025-03-11T00:00:00
                         00Z")
  },
  {
```



```
    user_id: "789ghi...",
    products: [
      { product_id: ObjectId("productId123"),
        quantity: 1, price: 999.99 }
    ],
    total_price: 999.99,
    order_date: ISODate("2025-03-10T01:00:00.002Z")
  }
]);

db.products.find({});

db.products.find({ category: "Electronics" });

db.Products.find({ quantity: { $gt: 0 } });

db.products.find({}).sort({ price: 1 });

db.Products.find({ price: { $lte: 100 } });

db.Carts.aggregate([
  { $match: { user_id: "789ghi..." } },
  { $unwind: "$items" },
  { $lookup: {
      from: "products",
      localField: "items.product_id",
      foreignField: "_id",
      as: "product_info"
  } },
  { $project: { "product_info.name": 1,
    "product_info.price": "$items.quantity": 1
  } }
]);
```



```
db.Orders.find({user_id: "123abc..."});

db.Orders.aggregate([
  {$match: {user_id: "123abc..."}},
  {$group: {_id: "$user_id", total_price:
  {$sum: "$total_price"}}}
]);

Aggregation:
db.products.aggregate([
  {$group: {_id: "$category", total_product
  {$sum: 1}}}
]);

db.products.aggregate([
  {$group: {_id: "$category", total_price:
  {$sum: {$multiply: ["$price", "$quantity"]}}}
]);

db.products.aggregate([
  {$group: {_id: null, average_price: {
  $avg: "$price"}}}
]);

db.products.find({quantity: {$lt: 10}});

db.products.find({}).sort({price: -1});

db.orders.aggregate([
  {$group: {_id: "$user_id", total_price:
  {$sum: "$total_price"}}}
]);
```



```
db.Orders.aggregate([
  {$group: {_id: "$user_id", total_
  price: {$sum: "$total_price"}}},
  {$sort: {total_price: -1}},
  {$limit: 1}
]);

db.Orders.aggregate([
  {$group: {_id: null, average_order_
  price: {$avg: "$total_price"}}}
]);
```

Code with Output:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Admin> mongosh
Current Mongosh Log ID: 67cff492cea8162357fa4213
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.4.0
Using MongoDB:          7.0.5
Using Mongosh:          2.4.0

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2025-03-05T10:38:37.702+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> use lab2
switched to db lab2
lab2> db.createCollection("Customers")
{ ok: 1 }
lab2> db.Customers.insertMany([
...     { Cust_id: "C001", Acc_Bal: 1500, Acc_Type: "Z" },
...     { Cust_id: "C002", Acc_Bal: 800, Acc_Type: "Z" },
...     { Cust_id: "C003", Acc_Bal: 2500, Acc_Type: "A" },
...     { Cust_id: "C004", Acc_Bal: 1800, Acc_Type: "Z" },
...     { Cust_id: "C005", Acc_Bal: 950, Acc_Type: "B" }
... ]);
...
lab2> db.Customers.insertMany([
...     { Cust_id: "001", Acc_Bal: 1500, Acc_Type: "S" },
...     { Cust_id: "002", Acc_Bal: 800, Acc_Type: "S" },
...     { Cust_id: "003", Acc_Bal: 2500, Acc_Type: "C" },
...     { Cust_id: "004", Acc_Bal: 1800, Acc_Type: "C" },
...     { Cust_id: "005", Acc_Bal: 950, Acc_Type: "S" }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67cff561cea8162357fa4214'),
    '1': ObjectId('67cff561cea8162357fa4215'),
    '2': ObjectId('67cff561cea8162357fa4216'),
    '3': ObjectId('67cff561cea8162357fa4217'),
    '4': ObjectId('67cff561cea8162357fa4218')
  }
}
lab2> db.find({Acc_Bal: {$gt: 1200}, Acc_Type: "S"})
TypeError: db.find is not a function
lab2> db.Customers.find({Acc_Bal: {$gt: 1200}, Acc_Type: "S"})
[
```

```
    price: 999,
    quantity: 30
  }
]
lab2_2> db.Products.find({price: {$lte : 100}})
[
  {
    _id: ObjectId('67cffb682f58b006f2fa421c'),
    product_id: 'P004',
    name: 'T-shirt',
    category: 'Clothing',
    price: 25,
    quantity: 200
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421d'),
    product_id: 'P005',
    name: 'Jeans',
    category: 'Clothing',
    price: 40,
    quantity: 150
  }
]
lab2_2> db.Carts.find({user_id: "123abc"})

lab2_2> db.Carts.insertMany([
...     {
...         user_id: "123abc",
...         cart_items: [
...             { product_id: "P001", quantity: 1 },
...             { product_id: "P004", quantity: 2 }
...         ]
...     },
...     {
...         user_id: "789ghi",
...         cart_items: [
...             { product_id: "P003", quantity: 1 },
...             { product_id: "P002", quantity: 1 }
...         ]
...     },
...     {
...         user_id: "456def",
...         cart_items: [
...             { product_id: "P005", quantity: 2 }
...         ]
...     },
...     {
...         user_id: "123abc",
...         cart_items: [
...             { product_id: "P004", quantity: 1 },
...             { product_id: "P003", quantity: 1 }
```

```
lab2_2> db.Carts.insertMany([
...     {
...         user_id: "123abc",
...         cart_items: [
...             { product_id: "P001", quantity: 1 },
...             { product_id: "P004", quantity: 2 }
...         ]
...     },
...     {
...         user_id: "789ghi",
...         cart_items: [
...             { product_id: "P003", quantity: 1 },
...             { product_id: "P002", quantity: 1 }
...         ]
...     },
...     {
...         user_id: "456def",
...         cart_items: [
...             { product_id: "P005", quantity: 2 }
...         ]
...     },
...     {
...         user_id: "123abc",
...         cart_items: [
...             { product_id: "P004", quantity: 1 },
...             { product_id: "P003", quantity: 1 }
...         ]
...     },
...     {
...         user_id: "789ghi",
...         cart_items: [
...             { product_id: "P002", quantity: 1 },
...             { product_id: "P005", quantity: 3 }
...         ]
...     }
... ]);
...
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67cffcab2f58b006f2fa4228'),
    '1': ObjectId('67cffcab2f58b006f2fa4229'),
    '2': ObjectId('67cffcab2f58b006f2fa422a'),
    '3': ObjectId('67cffcab2f58b006f2fa422b'),
    '4': ObjectId('67cffcab2f58b006f2fa422c')
  }
}
lab2_2> db.Carts.find({user_id: "123abc"})
[
```

```
mongosh mongodb://127.0.0.
}
lab2_2> db.Carts.find({user_id: "123abc"})
[
  {
    _id: ObjectId('67cffcab2f58b006f2fa4228'),
    user_id: '123abc',
    cart_items: [
      { product_id: 'P001', quantity: 1 },
      { product_id: 'P004', quantity: 2 }
    ]
  },
  {
    _id: ObjectId('67cffcab2f58b006f2fa422b'),
    user_id: '123abc',
    cart_items: [
      { product_id: 'P004', quantity: 1 },
      { product_id: 'P003', quantity: 1 }
    ]
  }
]
lab2_2> db.Orders.find({user_id: "123abc"})
[
  {
    _id: ObjectId('67cffb6c2f58b006f2fa421e'),
    user_id: '123abc',
    order_items: [
      { product_id: 'P001', quantity: 1, price: 699 },
      { product_id: 'P004', quantity: 2, price: 25 }
    ],
    total_price: 749,
    order_date: ISODate('2025-03-01T00:00:00.000Z')
  },
  {
    _id: ObjectId('67cffb6c2f58b006f2fa4221'),
    user_id: '123abc',
    order_items: [
      { product_id: 'P002', quantity: 1, price: 999 },
      { product_id: 'P004', quantity: 2, price: 25 }
    ],
    total_price: 1049,
    order_date: ISODate('2025-03-04T00:00:00.000Z')
  }
]
lab2_2> db.Orders.aggregate([
...     { $match: { user_id: "123abc..." } },
...     { $group: { _id: "$user_id", total_spent: { $sum: "$total_price" } } }
... ]);
...

lab2_2> db.Orders.aggregate([
...     { $match: { user_id: "123abc" } },
```

```
]
lab2_2> db.Orders.aggregate([
...     { $match: { user_id: "123abc..." } },
...     { $group: { _id: "$user_id", total_spent: { $sum: "$total_price" } } }
... ]);
...

lab2_2> db.Orders.aggregate([
...     { $match: { user_id: "123abc" } },
...     { $group: { _id: "$user_id", total_spent: { $sum: "$total_price" } } }
... ]);
[ { _id: '123abc', total_spent: 1798 } ]
lab2_2> db.Products.aggregate([
...     { $group: { _id: "$category", total_products: { $sum: 1 } } }
... ]);
...
[
  { _id: 'Electronics', total_products: 3 },
  { _id: 'Clothing', total_products: 2 }
]
lab2_2> db.Products.aggregate([
...     { $group: { category: "$category", total_products: { $sum: 1 } } }
... ]);
MongoServerError[Location40234]: The field 'category' must be an accumulator object
lab2_2> db.Products.aggregate([
...     { $group: { _id: "$category", total_price: { $sum: "price" } } }
... ]);
[
  { _id: 'Electronics', total_price: 0 },
  { _id: 'Clothing', total_price: 0 }
]
lab2_2> db.Products.aggregate([
...     { $group: { _id: "$category", total_price: { $sum: "price" } } }
... ]);
lab2_2> db.Products.aggregate([
...     { $group: { _id: "$category", total_price: { $sum: "$price" } } }
... ]);
[
  { _id: 'Electronics', total_price: 1848 },
  { _id: 'Clothing', total_price: 65 }
]
lab2_2> db.Products.aggregate([
...     { $group: { _id: NULL, avg_price: { $sum: "$price"} / {$sum: 1} } }
... ]);
ReferenceError: NULL is not defined
lab2_2> db.Products.aggregate([
...     { $group: { _id: null, avg_price: { $sum: "$price"} / {$sum: 1} } }
... ]);
...
MongoServerError[Location40234]: The field 'avg_price' must be an accumulator object
lab2_2> db.Products.aggregate([
```

```
    '3': ObjectId('67cffb682f58b006f2fa421c'),
    '4': ObjectId('67cffb682f58b006f2fa421d')
  }
}
lab2_2> db.Orders.insertMany([
...     {
...         user_id: "123abc",
...         order_items: [
...             { product_id: "P001", quantity: 1, price: 699 },
...             { product_id: "P004", quantity: 2, price: 25 }
...         ],
...         total_price: 749,
...         order_date: new Date("2025-03-01")
...     },
...     {
...         user_id: "789ghi",
...         order_items: [
...             { product_id: "P003", quantity: 1, price: 150 },
...             { product_id: "P002", quantity: 1, price: 999 }
...         ],
...         total_price: 1149,
...         order_date: new Date("2025-03-02")
...     },
...     {
...         user_id: "456def",
...         order_items: [
...             { product_id: "P005", quantity: 1, price: 40 }
...         ],
...         total_price: 40,
...         order_date: new Date("2025-03-03")
...     },
...     {
...         user_id: "123abc",
...         order_items: [
...             { product_id: "P002", quantity: 1, price: 999 },
...             { product_id: "P004", quantity: 2, price: 25 }
...         ],
...         total_price: 1049,
...         order_date: new Date("2025-03-04")
...     },
...     {
...         user_id: "789ghi",
...         order_items: [
...             { product_id: "P001", quantity: 1, price: 699 }
...         ],
...         total_price: 699,
...         order_date: new Date("2025-03-05")
...     }
... ]);
...
{
```

```
... ]);
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67cffb6c2f58b006f2fa421e'),
    '1': ObjectId('67cffb6c2f58b006f2fa421f'),
    '2': ObjectId('67cffb6c2f58b006f2fa4220'),
    '3': ObjectId('67cffb6c2f58b006f2fa4221'),
    '4': ObjectId('67cffb6c2f58b006f2fa4222')
  }
}
lab2_2> db.UserCarts.insertMany([
...     {
...         user_id: "123abc",
...         cart_items: [
...             { product_id: "P001", quantity: 1 },
...             { product_id: "P004", quantity: 2 }
...         ]
...     },
...     {
...         user_id: "789ghi",
...         cart_items: [
...             { product_id: "P003", quantity: 1 },
...             { product_id: "P002", quantity: 1 }
...         ]
...     },
...     {
...         user_id: "456def",
...         cart_items: [
...             { product_id: "P005", quantity: 2 }
...         ]
...     },
...     {
...         user_id: "123abc",
...         cart_items: [
...             { product_id: "P004", quantity: 1 },
...             { product_id: "P003", quantity: 1 }
...         ]
...     },
...     {
...         user_id: "789ghi",
...         cart_items: [
...             { product_id: "P002", quantity: 1 },
...             { product_id: "P005", quantity: 3 }
...         ]
...     }
... ]);
...
{
  acknowledged: true,
```

```
...         ]
...     }
... ]);
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67cffb712f58b006f2fa4223'),
    '1': ObjectId('67cffb712f58b006f2fa4224'),
    '2': ObjectId('67cffb712f58b006f2fa4225'),
    '3': ObjectId('67cffb712f58b006f2fa4226'),
    '4': ObjectId('67cffb712f58b006f2fa4227')
  }
}
lab2_2> db.Products.find()
[
  {
    _id: ObjectId('67cffb682f58b006f2fa4219'),
    product_id: 'P001',
    name: 'Smartphone',
    category: 'Electronics',
    price: 699,
    quantity: 50
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421a'),
    product_id: 'P002',
    name: 'Laptop',
    category: 'Electronics',
    price: 999,
    quantity: 30
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421b'),
    product_id: 'P003',
    name: 'Headphones',
    category: 'Electronics',
    price: 150,
    quantity: 100
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421c'),
    product_id: 'P004',
    name: 'T-shirt',
    category: 'Clothing',
    price: 25,
    quantity: 200
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421d'),
    product_id: 'P005',
```

```
> 1 | db.Products.find({quantity: {"$gt" : 0})
    |                                        ^
  2 |

lab2_2> db.Products.find({quantity: {$gt : 0}})
[
  {
    _id: ObjectId('67cffb682f58b006f2fa4219'),
    product_id: 'P001',
    name: 'Smartphone',
    category: 'Electronics',
    price: 699,
    quantity: 50
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421a'),
    product_id: 'P002',
    name: 'Laptop',
    category: 'Electronics',
    price: 999,
    quantity: 30
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421b'),
    product_id: 'P003',
    name: 'Headphones',
    category: 'Electronics',
    price: 150,
    quantity: 100
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421c'),
    product_id: 'P004',
    name: 'T-shirt',
    category: 'Clothing',
    price: 25,
    quantity: 200
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421d'),
    product_id: 'P005',
    name: 'Jeans',
    category: 'Clothing',
    price: 40,
    quantity: 150
  }
]
lab2_2> db.Products.find().sort({price: 1})
[
  {
```

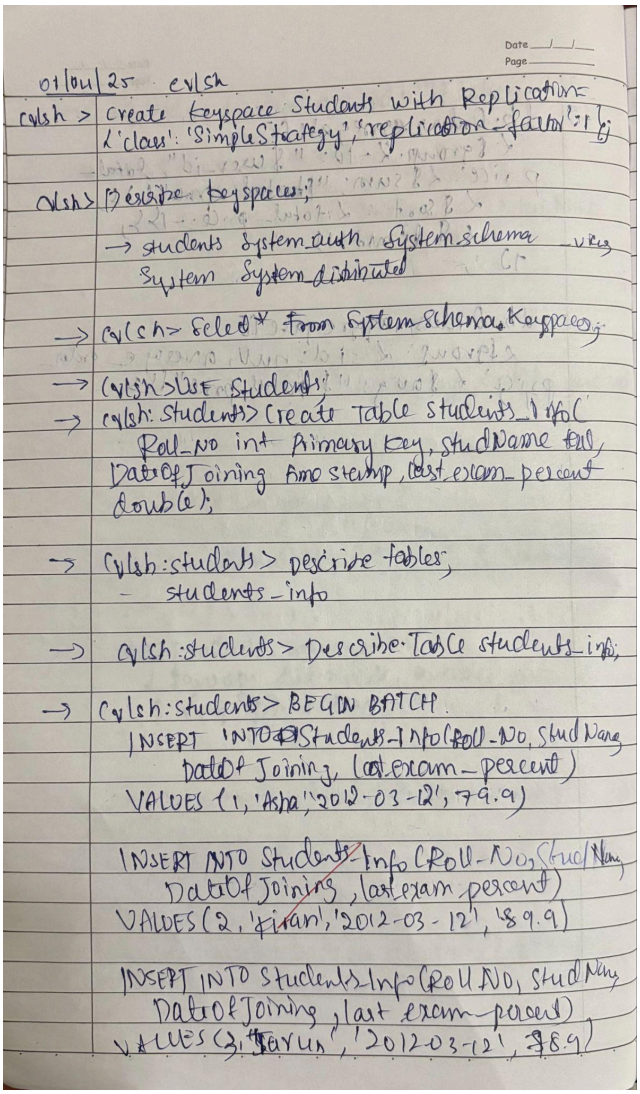
```
    quantity: 150
  }
]
lab2_2> db.Products.find().sort({price: 1})
[
  {
    _id: ObjectId('67cffb682f58b006f2fa421c'),
    product_id: 'P004',
    name: 'T-shirt',
    category: 'Clothing',
    price: 25,
    quantity: 200
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421d'),
    product_id: 'P005',
    name: 'Jeans',
    category: 'Clothing',
    price: 40,
    quantity: 150
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421b'),
    product_id: 'P003',
    name: 'Headphones',
    category: 'Electronics',
    price: 150,
    quantity: 100
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa4219'),
    product_id: 'P001',
    name: 'Smartphone',
    category: 'Electronics',
    price: 699,
    quantity: 50
  },
  {
    _id: ObjectId('67cffb682f58b006f2fa421a'),
    product_id: 'P002',
    name: 'Laptop',
    category: 'Electronics',
    price: 999,
    quantity: 30
  }
]
lab2_2> db.Products.find({price: {$lte : 100}})
[
  {
    _id: ObjectId('67cffb682f58b006f2fa421c'),
    product_id: 'P004',
```

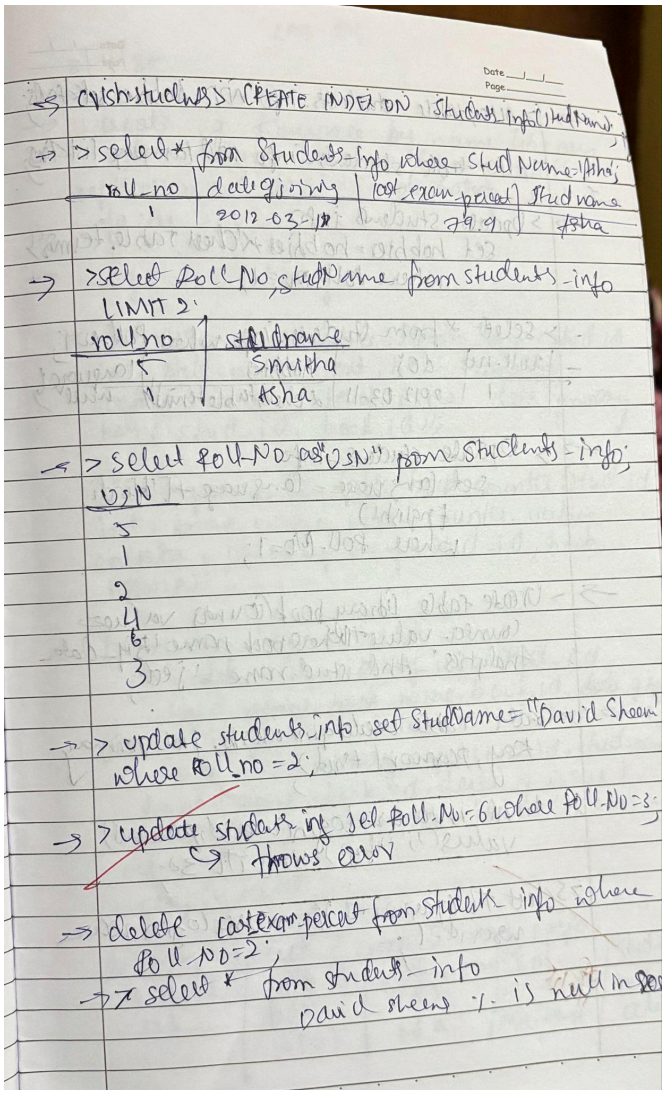
Program 3

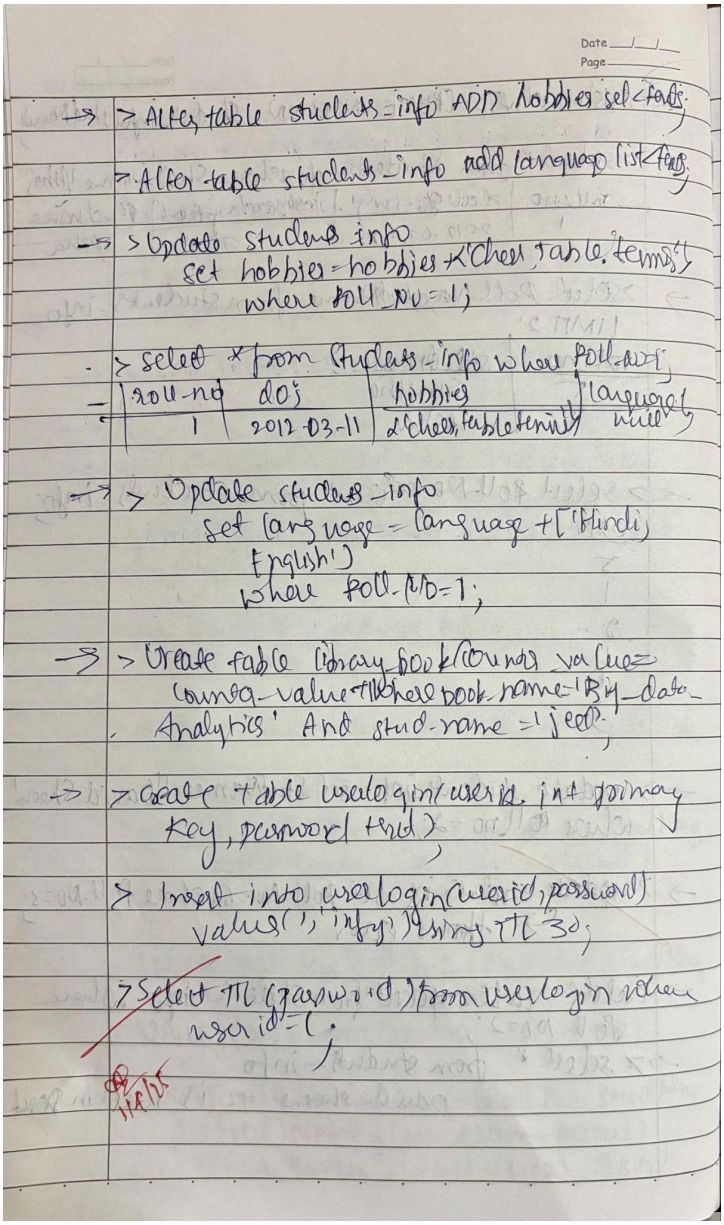
Perform the following DB operations using Cassandra.

a) Create a keyspace by name Library
b) Create a column family by name Library-Info with attributes
Stud_Id Primary Key,
Counter_value of type Counter,
Stud_Name, Book-Name, Book-Id,
Date_of_issue
c) Insert the values into the table in batch
d) Display the details of the table created and increase the value of the counter e)
Write a query to show that a student with id 112 has taken a book “BDA” 2 times. f)
Export the created column to a csv file
g) Import a given csv dataset from local file system into Cassandra column family

Observation:







Date __/__/__
Page _____

→ > Alter table students_info ADD hobbies set<text>;

> Alter table students_info add language list<text>;

→ > Update students_info
set hobbies = hobbies + {'Chess, Table tennis'}
where roll_no = 1;

> select * from students_info where roll_no=1;

roll_no	dob	hobbies	language
1	2012-03-11	{'Chess, Table tennis'}	null

→ > Update students_info
set language = language + ['Hindi', 'English']
where roll_no=1;

→ > Create table library_book(counter value counter, value+1 where book_name='By_data_Analytics' And stud_name='jeet');

→ > create table userlogin(user_id int primary key, password text);

> Insert into userlogin(user_id, password) values(1, 'info') using TTL 30;

> Select TTL (password) from userlogin where user_id=1;

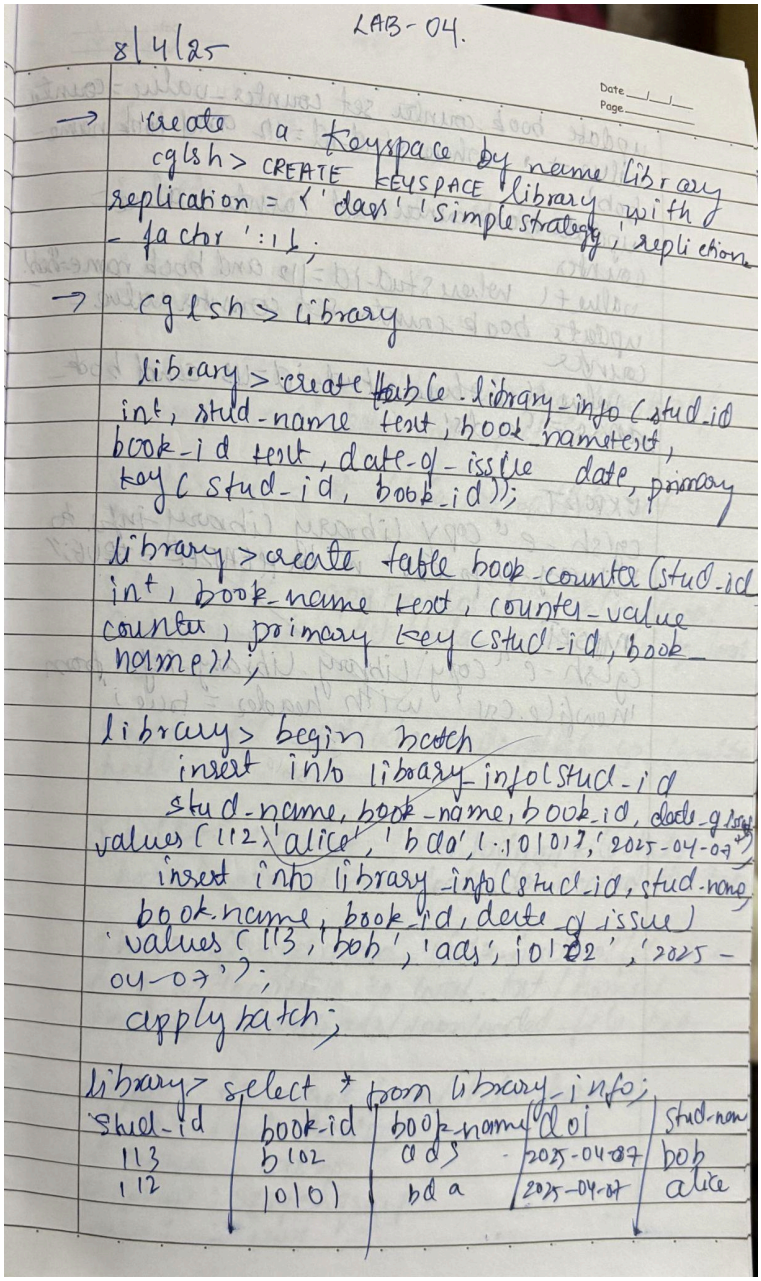
Code with Output:

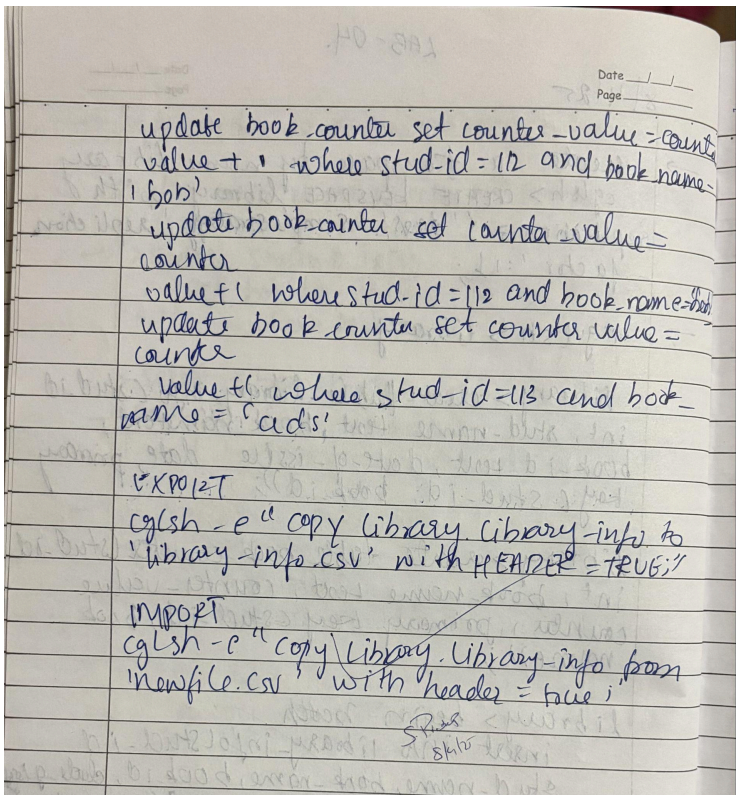
```
cqlsh> CREATE KEYSPACE Employee WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 1 };
cqlsh> CREATE TABLE Employee.Employee_Info (
   ...     Emp_Id int,
   ...     Salary DECIMAL,
   ...     Emp_Name TEXT,
   ...     Designation TEXT,
   ...     Date_of_Joining DATE,
   ...     Dept_Name TEXT,
   ...     PRIMARY KEY (Emp_Id, Salary)
   ... ) WITH CLUSTERING ORDER BY (Salary ASC);
cqlsh> BEGIN BATCH
   ... INSERT INTO Employee.Employee_Info (Emp_Id, Salary, Emp_Name, Designation, Date_of_Joining, Dept_Name) VALUES (121, 60000, 'John Doe', 'Developer', '2023-01-15
', 'IT');
   ... INSERT INTO Employee.Employee_Info (Emp_Id, Salary, Emp_Name, Designation, Date_of_Joining, Dept_Name) VALUES (122, 80000, 'Jane Smith', 'Manager', '2022-05-20
', 'HR');
   ... INSERT INTO Employee.Employee_Info (Emp_Id, Salary, Emp_Name, Designation, Date_of_Joining, Dept_Name) VALUES (123, 55000, 'Alice Johnson', 'Analyst', '2021-11
-10', 'Finance');
   ... APPLY BATCH;
cqlsh> UPDATE Employee.Employee_Info SET Emp_Name = 'Johnathan Doe', Dept_Name = 'Engineering' WHERE Emp_Id = 121 AND Salary = 60000;
cqlsh> SELECT * FROM Employee.Employee_Info WHERE Emp_Id = 121 ORDER BY Salary;

 emp_id | salary | date_of_joining | dept_name   | designation | emp_name
--------+--------+-----------------+-------------+-------------+--------------
    121 |  60000 |      2023-01-15 | Engineering |   Developer | Johnathan Doe

(1 rows)
cqlsh> ALTER TABLE Employee.Employee_Info ADD Projects SET<TEXT>;
cqlsh> UPDATE Employee.Employee_Info SET Projects = {'Project A', 'Project B'} WHERE Emp_Id = 121 AND Salary = 60000;
cqlsh> INSERT INTO Employee.Employee_Info (Emp_Id, Salary, Emp_Name, Designation, Date_of_Joining, Dept_Name) VALUES (124, 30000, 'Temp Employee', 'Intern', '2023-10-
01', 'Temp Dept') USING TTL 15;
cqlsh> SELECT * FROM Employee.Employee_Info;

 emp_id | salary | date_of_joining | dept_name   | designation | emp_name      | projects
--------+--------+-----------------+-------------+-------------+---------------+--------------------------
    123 |  55000 |      2021-11-10 |     Finance |     Analyst | Alice Johnson |                     null
    122 |  80000 |      2022-05-20 |          HR |     Manager |    Jane Smith |                     null
    121 |  60000 |      2023-01-15 | Engineering |   Developer | Johnathan Doe | {'Project A', 'Project B'}

(3 rows)
cqlsh>
```

Program 4

Execution of HDFS Commands for interaction with Hadoop Environment. (Minimum 10 commands to be executed)

Observation:





Code with Output:

```
bmscecse@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.1.0 | Cassandra 4.1.4 | CQL spec 3.4.6 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE KEYSPACE Students WITH REPLICATION={
   ... 'class':'SimpleStrategy','replication_factor':1};
cqlsh> DESCRIBE KEYSPACES

students  system_auth         system_schema  system_views
system    system_distributed  system_traces  system_virtual_schema

cqlsh> SELECT * FROM system.schema_keyspaces;
InvalidRequest: Error from server: code=2200 [Invalid query] message="table schema_keyspaces does not exist"
cqlsh> use Students;
cqlsh:students> create table Students_info(Roll_No int Primary key,StudName text,DateOfJoining timestamp,last_exam_Percent double);
cqlsh:students> describe tables;

students_info

cqlsh:students> describe table students;
Table 'students' not found in keyspace 'students'
cqlsh:students> describe table students_info;

CREATE TABLE students.students_info (
    roll_no int PRIMARY KEY,
    dateofjoining timestamp,
    last_exam_percent double,
    studname text
) WITH additional_write_policy = '99p'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND memtable = 'default'
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND extensions = {}
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99p';
```

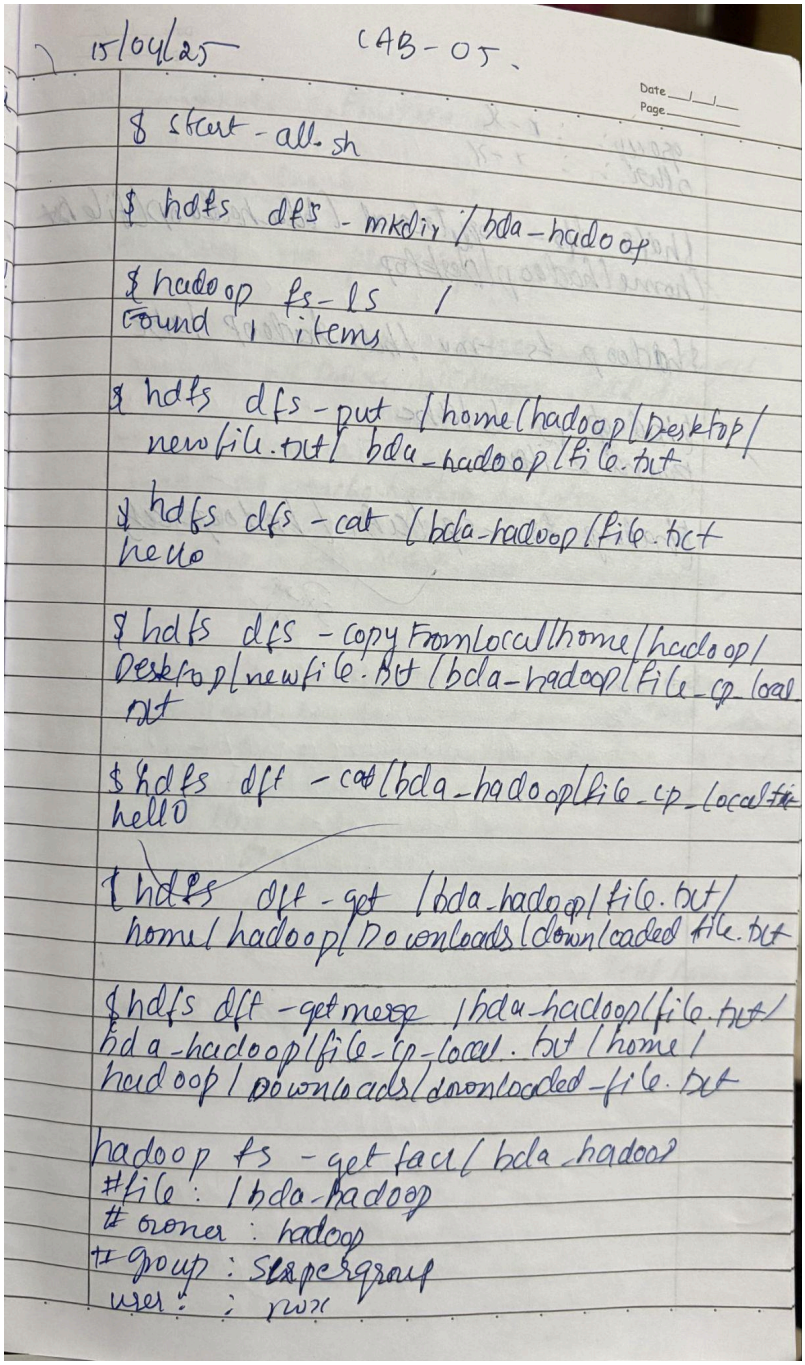
```
cqlsh:students> Begin batch insert into Students_info(Roll_no, StudName,DateOfJoining, last_exam_Percent) values(1,'Sadhana','2023-10-09', 98) insert into Students_info(Roll_no, StudName,DateOfJoining, last_exam_Percent) values(2,'Rutu','2023-10-10', 97) insert into Students_info(Roll_no, StudName,DateOfJoining, last_exam_Percent) values(3,'Rachana','2023-10-10', 97.5) insert into Students_info(Roll_no, StudName,DateOfJoining, last_exam_Percent) values(4,'Charu','2023-10-06', 96.5) apply batch;
cqlsh:students> select * from students_info;

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
       1 | 2023-10-08 18:30:00.000000+0000 |                98 |  Sadhana
       2 | 2023-10-09 18:30:00.000000+0000 |                97 |     Rutu
       4 | 2023-10-05 18:30:00.000000+0000 |              96.5 |    Charu
       3 | 2023-10-09 18:30:00.000000+0000 |              97.5 |  Rachana

(4 rows)
cqlsh:students> select * from students_info where roll_no in (1,2,3);

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
       1 | 2023-10-08 18:30:00.000000+0000 |                98 |  Sadhana
       2 | 2023-10-09 18:30:00.000000+0000 |                97 |     Rutu
       3 | 2023-10-09 18:30:00.000000+0000 |              97.5 |  Rachana

(3 rows)
cqlsh:students> select * from students_info where Studname='Charu';
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may have unpredictable performance. If you want to execute this query despite the performance unpredictability, use ALLOW FILTERING"
cqlsh:students> create index on Students_info(StudName);
cqlsh:students> select * from students_info where Studname='Charu';

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
       4 | 2023-10-05 18:30:00.000000+0000 |              96.5 |    Charu

(1 rows)
cqlsh:students> select Roll_no,StudName from students_info LIMIT 2;
```

```
(4 rows)
cqlsh:students> select * from students_info where roll_no in (1,2,3);

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
       1 | 2023-10-08 18:30:00.000000+0000 |                98 |  Sadhana
       2 | 2023-10-09 18:30:00.000000+0000 |                97 |     Rutu
       3 | 2023-10-09 18:30:00.000000+0000 |              97.5 |  Rachana

(3 rows)
cqlsh:students> select * from students_info where Studname='Charu';
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may have unpredictable performance. If you want to execute this query d
espite the performance unpredictability, use ALLOW FILTERING"
cqlsh:students> create index on Students_info(StudName);
cqlsh:students> select * from students_info where Studname='Charu';

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
       4 | 2023-10-05 18:30:00.000000+0000 |              96.5 |    Charu

(1 rows)
cqlsh:students> select Roll_no,StudName from students_info LIMIT 2;

 roll_no | studname
---------+----------
       1 |  Sadhana
       2 |     Rutu

(2 rows)
cqlsh:students> SELECT Roll_no as "USN" from Students_info;

 USN
-----
   1
   2
   4
   3
```

Program 5

Implement Wordcount program on Hadoop framework

Observation:


15/04/25 CAB-05.

$ start-all.sh

$ hdfs dfs -mkdir /bda-hadoop

$ hadoop fs -ls /
Found 1 items

$ hdfs dfs -put /home/hadoop/Desktop/newfile.txt /bda-hadoop/file.txt

$ hdfs dfs -cat /bda-hadoop/file.txt
hello

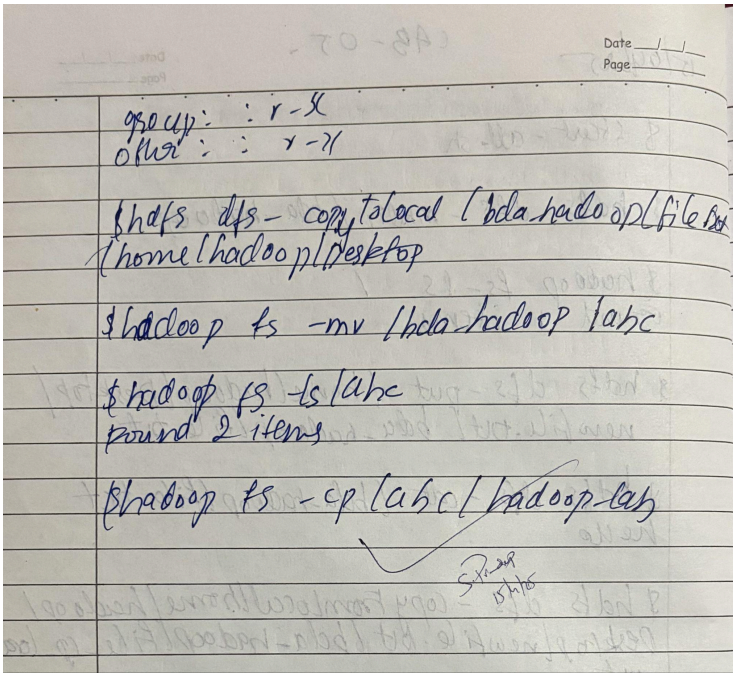
$ hdfs dfs -copyFromLocal /home/hadoop/Desktop/newfile.txt /bda-hadoop/file_cp_local.txt

$ hdfs dfs -cat /bda-hadoop/file_cp_local.txt
hello

hdfs dfs -get /bda-hadoop/file.txt /home/hadoop/Downloads/downloaded file.txt

$ hdfs dfs -getmerge /bda-hadoop/file.txt /bda-hadoop/file_cp_local.txt /home/hadoop/Downloads/downloaded_file.txt

hadoop fs -getfacl /bda-hadoop
#file: /bda-hadoop
owner: hadoop
group: supergroup
user: : rwx


group: : r-x
other: : r-x

$ hdfs dfs -copyToLocal /bda-hadoop/file.txt /home/hadoop/Desktop

$ hadoop fs -mv /bda-hadoop /abc

$ hadoop fs -ls /abc
Found 2 items

$ hadoop fs -cp /abc/ /hadoop-lab

Ubuntu - Eclipse.

Word count.

→ create JAR file & input text file

→ create three java classes into the project Named WCDriver, WCMapper, WCReducer.

→ import java.io.IOException;
import org.apache.hadoop.io.intWritable;
import org.apache.hadoop.io.Text
import org.apache.hadoop.mapred.MapReduceBase;
"
"
public class WCMapper extends MapReduceBase implements Mapper<LongWritable text, Text, IntWritable
public void map(LongWritable key, Text value, output collector <Text, IntWritable> output, Reporter rep) throws IOException
String line = value.toString();
for(String word: line.split("")){
if(word.length()>0){
output.collect(new Text(word), new IntWritable(1));
}
}
}

→ import java.IOException;
import java.io.IOException

public class WCReducer extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable
public void reduce(Text key, Iterator<IntWritable> value, output collector<Text, IntWritable> output), throws IOException
int count = 0;
while(value.hasNext()){
IntWritable i = value.next();
count += i.get();
}
output.collect(key, new IntWritable(count));
}

→ public class WCDriver extends Configured implements Tool{
public int run(String args[]) throws IOException
{
if(args.length<2){
System.out.println("please give valid inputs");
return -1;
}
JobConf conf = new JobConf(WCDriver.class);
FileInputFormat.setInputPaths(conf, new Path(args[0]));
FileOutputFormat.setOutputPath(conf, new Path(args[1]));
conf.setMapperClass(WCMapper.class);
conf.setMapOutputKeyClass(Text.class);
conf.setMapOutputValueClass(IntWritable.class);

Date __/__/__
Page

```
conf.setOutputValueClass(IterInt.class);
JobClient.runJob(conf);
return 0;
}
public static void main(String args[]) throws
Exception
{
    int exitCode = ToolRunner.run(new
WordMover(), args);
    System.out.println(exitCode);
}
```

Code with Output:

```
                HDFS: Number of bytes read=162
                HDFS: Number of bytes written=67
                HDFS: Number of read operations=15
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=4
                HDFS: Number of bytes read erasure-coded=0
        Map-Reduce Framework
                Map input records=5
                Map output records=20
                Map output bytes=161
                Map output materialized bytes=207
                Input split bytes=86
                Combine input records=0
                Combine output records=0
                Reduce input groups=10
                Reduce shuffle bytes=207
                Reduce input records=20
                Reduce output records=10
                Spilled Records=40
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=0
                Total committed heap usage (bytes)=1052770304
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=81
        File Output Format Counters
                Bytes Written=67
0
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ hadoop fs -ls /rgs/output/
Found 2 items
-rw-r--r--   1 hadoop supergroup          0 2025-05-06 15:50 /rgs/output/_SUCCESS
-rw-r--r--   1 hadoop supergroup         67 2025-05-06 15:50 /rgs/output/part-00000
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ hadoop fs -cat /rgs/output/part-00000
are     1
brother 1
family  1
hi      1
how     5
is      4
job     1
sister  1
ur      4
you     1
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ ~
bash: /home/hadoop: Is a directory
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$
```

Program 6

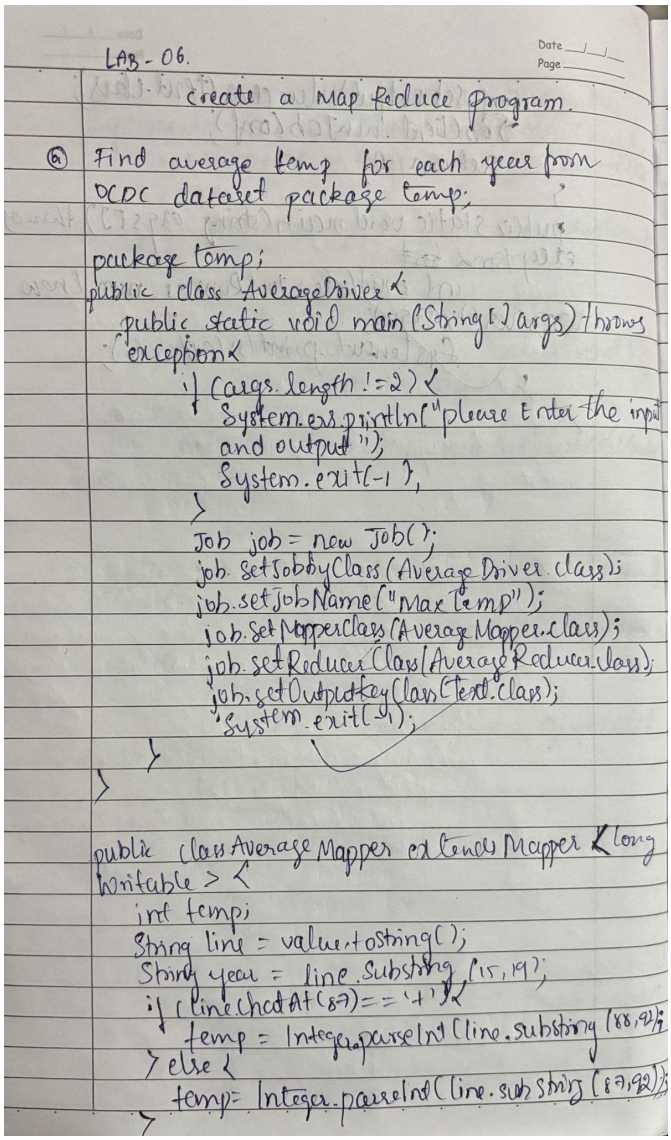
From the following link extract the weather data

https://github.com/tomwhite/hadoop-book/tree/master/input/ncdc/all

a) Create a MapReduce program to find average temperature for each year from the NCDC data set.

b) find the mean max temperature for every month

Observation:

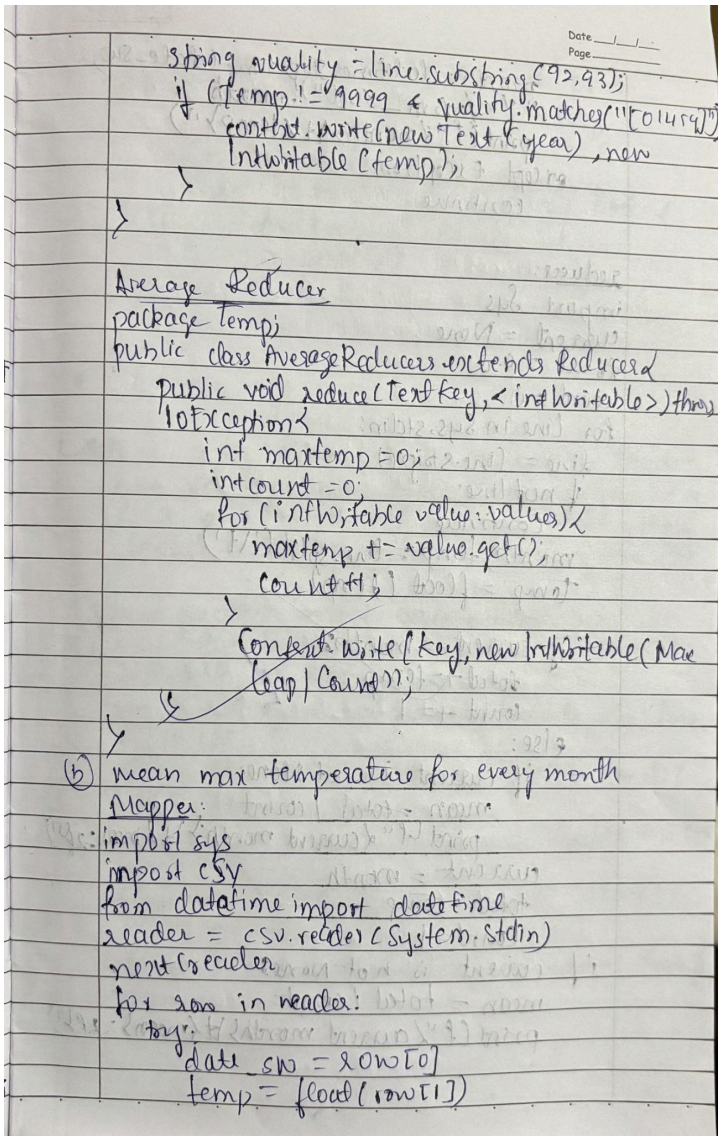


LAB - 06.

create a Map Reduce Program.

(a) Find average temp for each years from NCDC dataset package temp;

```
package temp;
public class AverageDriver {
  public static void main(String[] args) throws
  Exception{
    if (args.length != 2){
      System.out.println("please Enter the input
      and output ");
      System.exit(-1);
    }
    Job job = new Job();
    job.setJobByClass(AverageDriver.class);
    job.setJobName("Max temp");
    job.setMapperClass(AverageMapper.class);
    job.setReducerClass(AverageReducer.class);
    job.setOutputKeyClass(Text.class);
    System.exit(-1);
  }
}

public class AverageMapper extends Mapper<Long
Writable > {
  int temp;
  String line = value.toString();
  String year = line.substring(15,19);
  if (line.charAt(87) == '+'){
    temp = Integer.parseInt(line.substring(88,92));
  }else{
    temp= Integer.parseInt(line.substring(87,92));
  }
```



```
String quality = line.substring(92,93);
if (temp != 9999 & quality.matches("[01459]")
  context.write(new Text(year), new
  IntWritable(temp));
}
}

Average Reducer
package temp;
public class AverageReducer extends Reducer{
  Public void reduce(Text key, <intWritable>)throws
  IOException{
    int maxtemp = 0;
    int count = 0;
    for(IntWritable value: values){
      maxtemp += value.get();
      count++;
    }
    Context.write(key, new IntWritable(Max
    temp/count));
  }
}
```

(b) mean max temperature for every month

Mapper:

```
import sys
import csv
from datetime import datetime
reader = csv.reader(System.stdin)
next(reader)
for row in reader:
  try:
    date_str = row[0]
    temp = float(row[1])
```

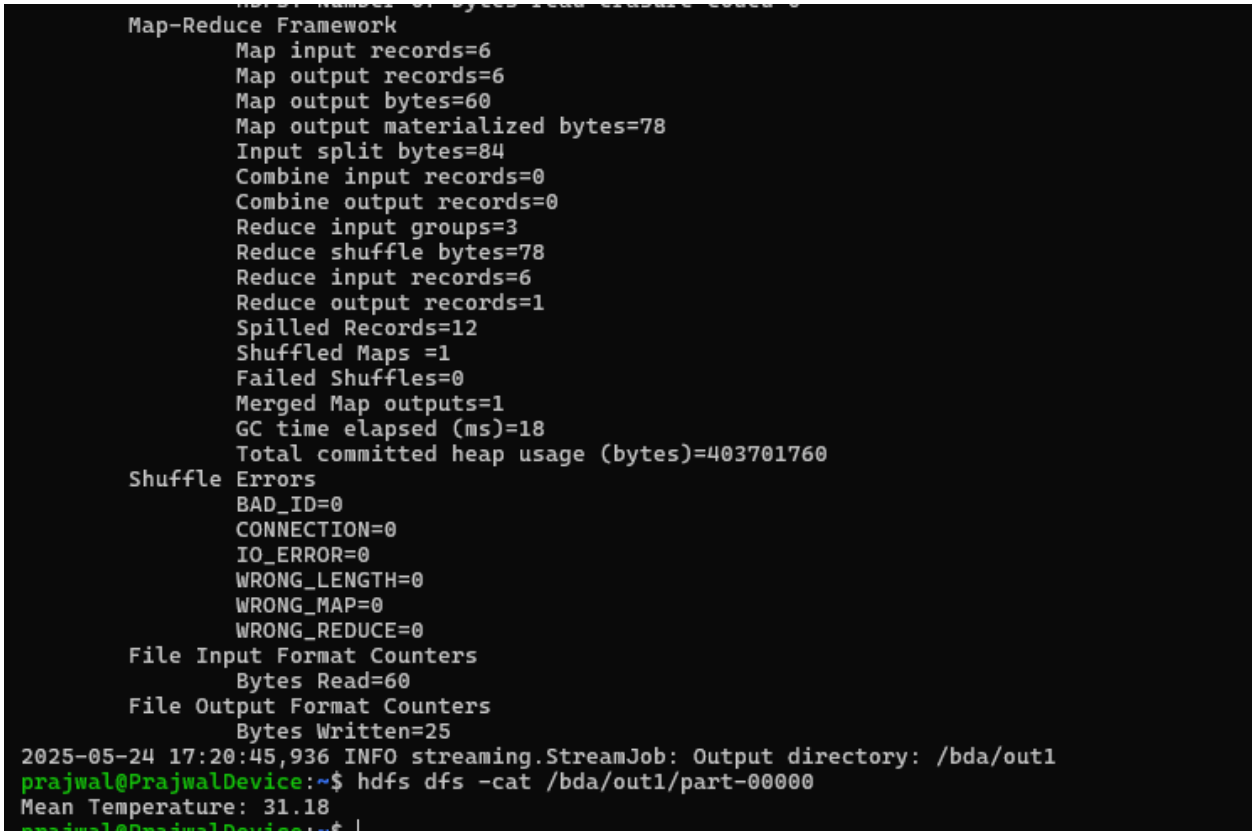
```
        month = datetime.strptime(date_str,
        "%Y-%m-%d").month
        print(f"{month}\t{temp}")
    except Exception:
        continue
```

reducer:

```
import sys
current = None
total = 0.0
count = 0
for line in sys.stdin:
    line = line.strip()
    if not line:
        continue
    month, temp = line.split('\t')
    temp = float(temp)

    if current == month:
        total += temp
        count += 1
    else:
        if current is not None:
            mean = total / count
            print(f"{current month}\t{mean:.2f}")
        current = month
        total = temp
        count = 1
if current is not None:
    mean = total / count
    print(f"{current month}\t{mean:.2f}")
```

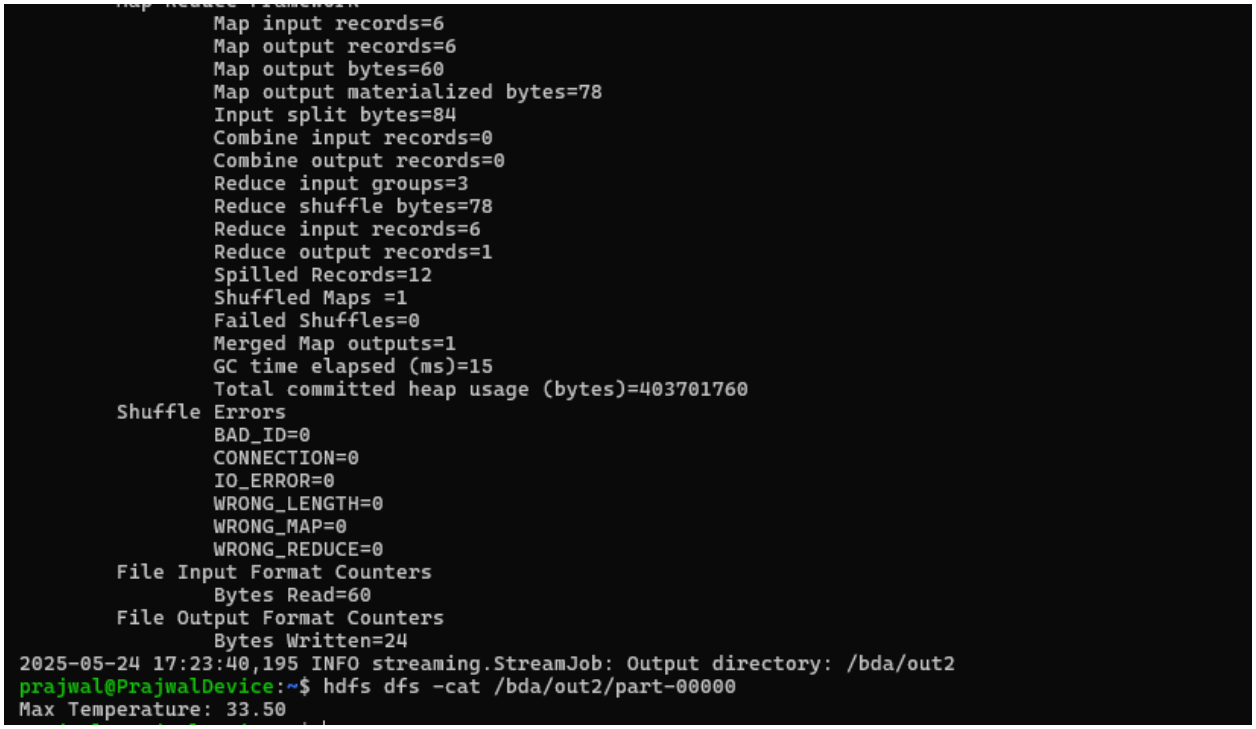
Code with Output:

a) Average temperature



```
        Map-Reduce Framework
                Map input records=6
                Map output records=6
                Map output bytes=60
                Map output materialized bytes=78
                Input split bytes=84
                Combine input records=0
                Combine output records=0
                Reduce input groups=3
                Reduce shuffle bytes=78
                Reduce input records=6
                Reduce output records=1
                Spilled Records=12
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=18
                Total committed heap usage (bytes)=403701760
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=60
        File Output Format Counters
                Bytes Written=25
2025-05-24 17:20:45,936 INFO streaming.StreamJob: Output directory: /bda/out1
prajwal@PrajwalDevice:~$ hdfs dfs -cat /bda/out1/part-00000
Mean Temperature: 31.18
```

b) Maximum temperature

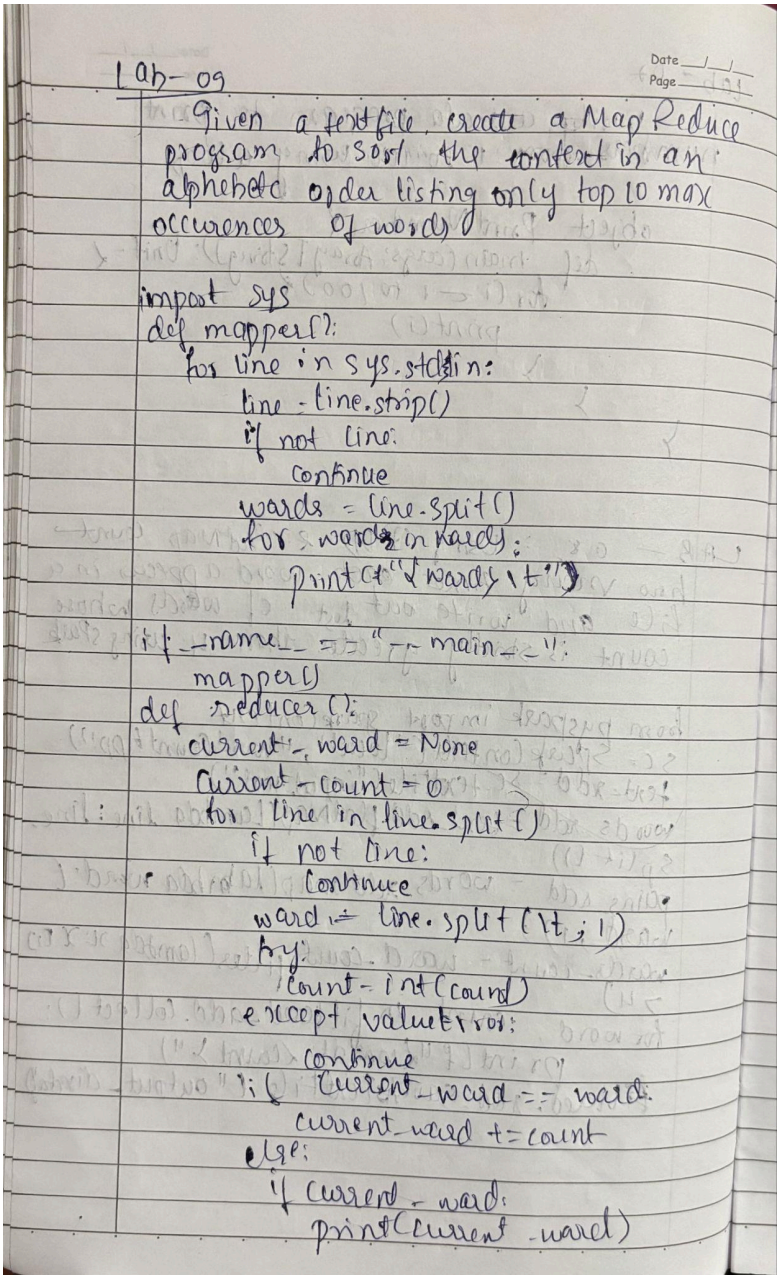


```
                Map input records=6
                Map output records=6
                Map output bytes=60
                Map output materialized bytes=78
                Input split bytes=84
                Combine input records=0
                Combine output records=0
                Reduce input groups=3
                Reduce shuffle bytes=78
                Reduce input records=6
                Reduce output records=1
                Spilled Records=12
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=15
                Total committed heap usage (bytes)=403701760
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=60
        File Output Format Counters
                Bytes Written=24
2025-05-24 17:23:40,195 INFO streaming.StreamJob: Output directory: /bda/out2
prajwal@PrajwalDevice:~$ hdfs dfs -cat /bda/out2/part-00000
Max Temperature: 33.50
```

Program 7

For a given Text file, Create a Map Reduce program to sort the content in an alphabetic order listing only top 10 maximum occurrences of words.

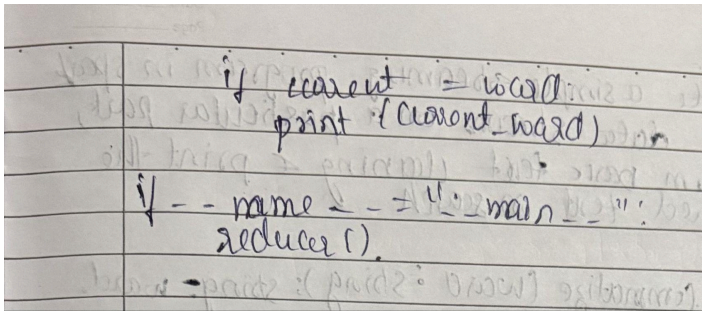
Observation:



Lab-09

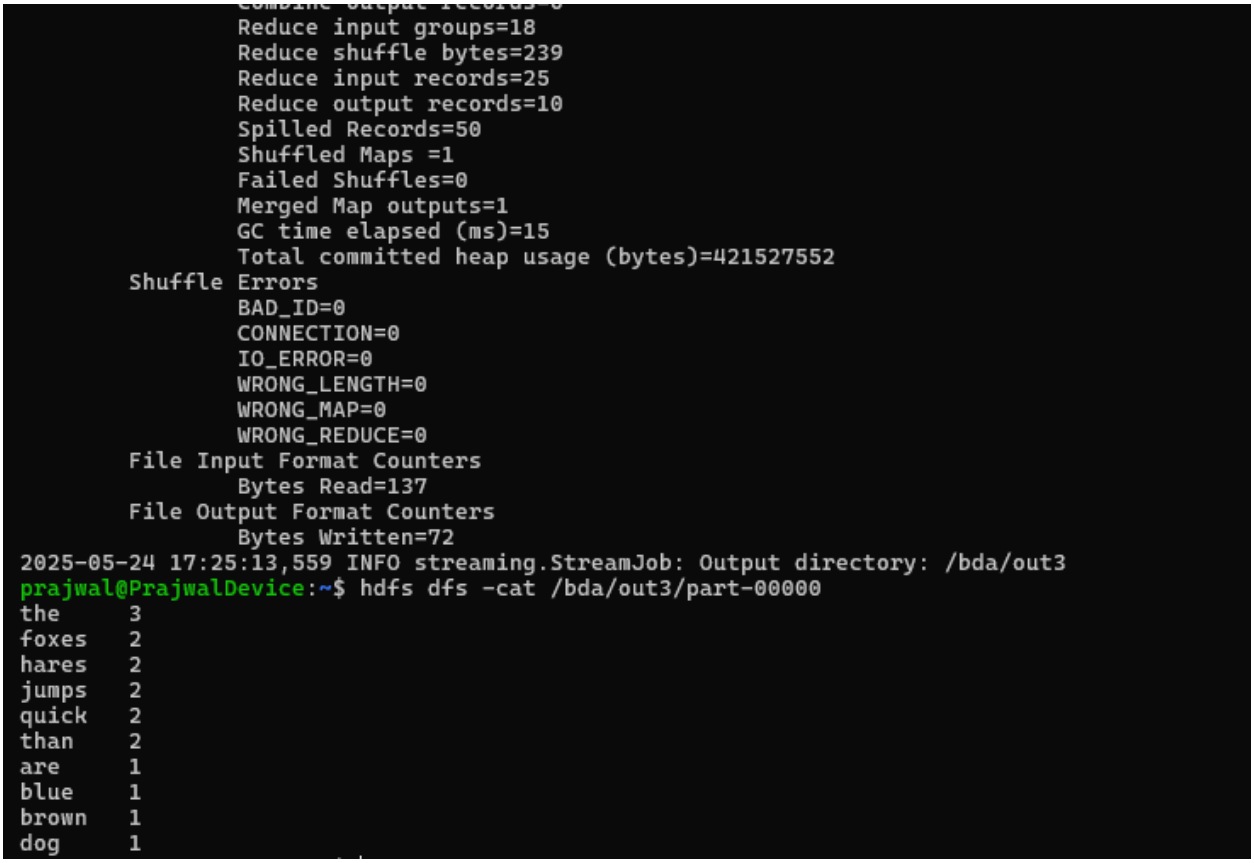
Given a text file, create a Map Reduce program to sort the content in an alphabetic order listing only top 10 max occurences of words.

```
import sys
def mapper():
    for line in sys.stdin:
        line = line.strip()
        if not line:
            continue
        words = line.split()
        for words in words:
            print("%s\t1" % words)
if __name__ == "__main__":
    mapper()
def reducer():
    current_word = None
    current_count = 0
    for line in line.split()
        if not line:
            continue
        word = line.split('\t', 1)
        try:
            count = int(count)
        except ValueError:
            continue
        if current_word == word:
            current_word += count
        else:
            if current_word:
                print(current_word)
```



```
    if current_word
        print(current_word)
if __name__ == "__main__":
    reducer()
```

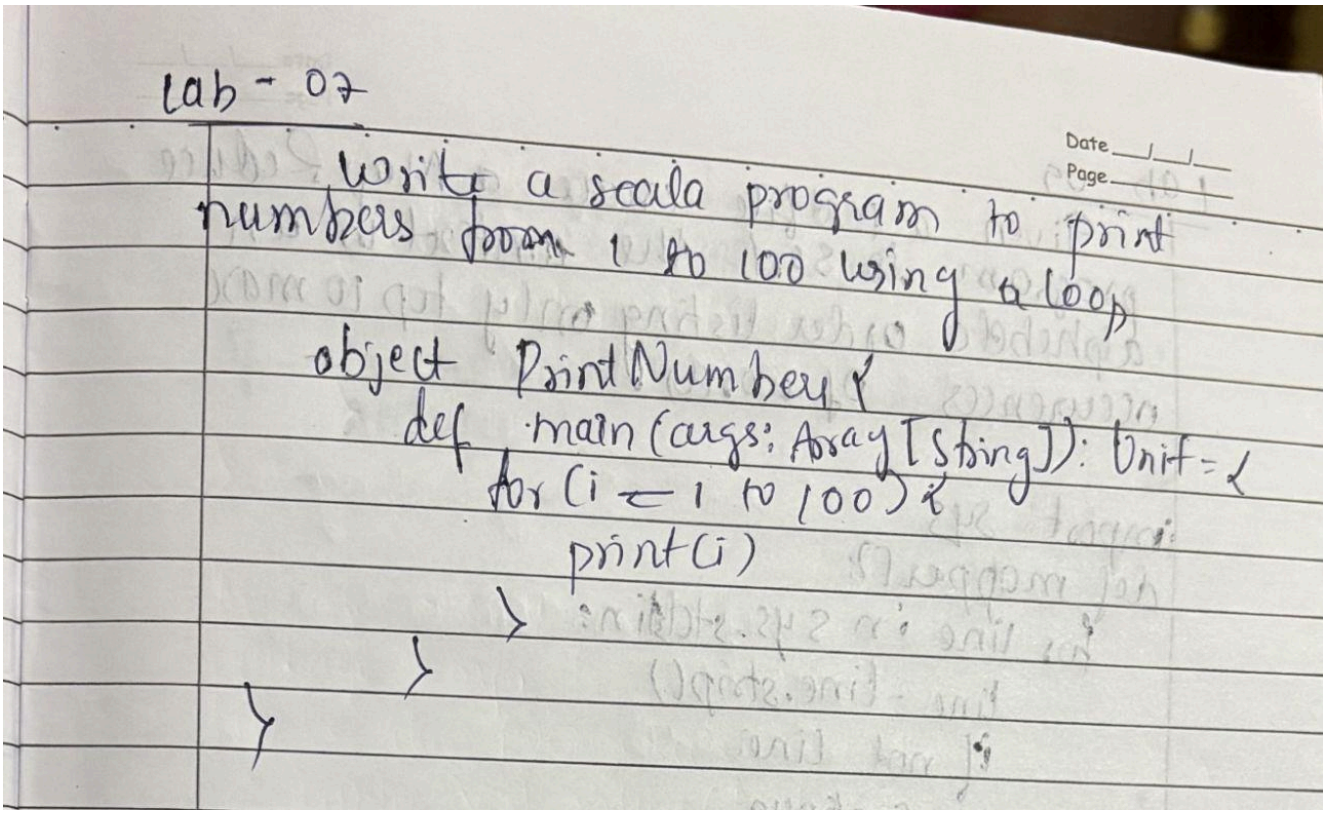
Code with Output:



Program 8

Write a Scala program to print numbers from 1 to 100 using for loop.

Observation:

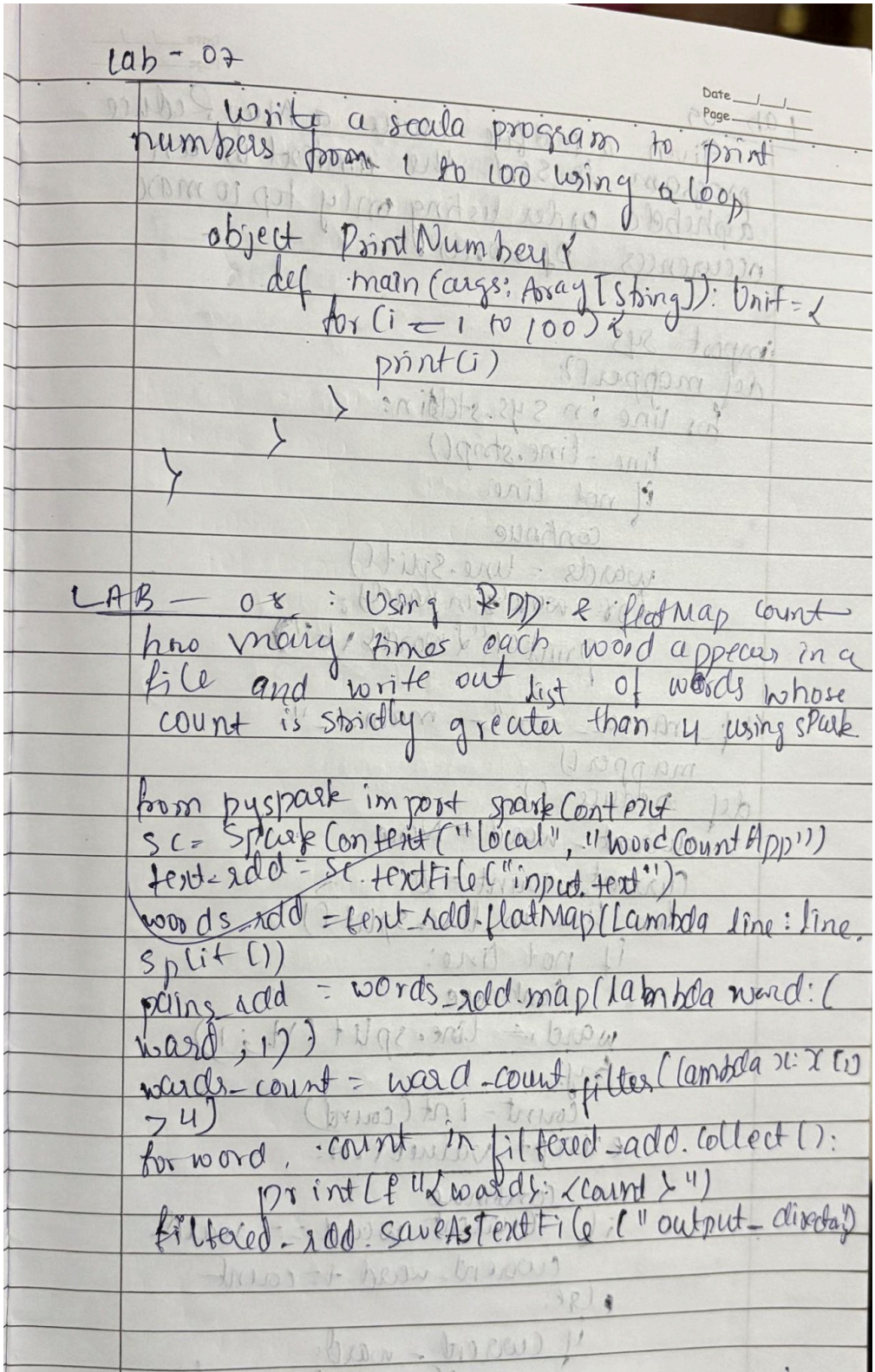


Code with Output:

Program 9

Using RDD and FlatMap count how many times each word appears in a file and write out a list of words whose count is strictly greater than 4 using Spark

Observation:



lab - 07

Write a scala program to print numbers from 1 to 100 using a loop

```
object PrintNumbers {
  def main(args: Array[String]): Unit = {
    for (i <- 1 to 100) {
      println(i)
    }
  }
}
```

LAB - 08 : Using RDD & flatMap count how many times each word appear in a file and write out list of words whose count is strictly greater than 4 using spark.

```
from pyspark import SparkContext
sc = SparkContext("local", "WordCountApp")
text_rdd = sc.textFile("input.text")
words_rdd = text_rdd.flatMap(lambda line: line.split())
pairs_rdd = words_rdd.map(lambda word: (word, 1))
words_count = word_count.filter(lambda x: x[1] > 4)
for word, count in filtered_rdd.collect():
    print("{} words: {}".format(count > 4))
filtered_rdd.saveAsTextFile("output_directory")
```

Code with Output:

```
prajwal@PrajwalDevice:~$ spark-shell
25/05/24 17:41:38 WARN Utils: Your hostname, PrajwalDevice resolves to a loopback address: 127.0.1.1; using 10.255.255.254 instead (on interface lo)
25/05/24 17:41:38 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/05/24 17:41:46 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Spark context Web UI available at http://10.255.255.254:4040
Spark context available as 'sc' (master = local[*], app id = local-1748088707553).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.5
      /_/

Using Scala version 2.12.18 (OpenJDK 64-Bit Server VM, Java 21.0.7)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val file=sc.text25/05/24 17:42:00 WARN GarbageCollectionMetrics: To enable non-built-in garbage collector(s) List(G1 Concurrent GC), users should configure it(them) to spark.e
 spark.eventLog.gcMetrics.oldGenerationGarbageCollectors
val file=sc.textFile("il.txt")
file: org.apache.spark.rdd.RDD[String] = il.txt MapPartitionsRDD[1] at textFile at <console>:23

scala> val words=file.flatMap(line=>line.split("\\W+"))
words: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at flatMap at <console>:23

scala> val wordpairs=words.map(word=>(word.toLowerCase,1))
wordpairs: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[3] at map at <console>:23

scala> val wordc=wordpairs.reduceByKey(_+_)
wordc: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:23

scala> val fil2=wordc.filter{case(word,count)=>count>2}
fil2: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[5] at filter at <console>:23

scala> fil2.collect().foreach(println)
```