VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Big Data Analytics(23CS6PCBDA)

Submitted by

Praneeta M Reddy (1BM22CS205)

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
Feb-2025 to June-2025

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(23CS6PCBDA)**" carried out by **Praneeta M Reddy (1BM22CS205),** 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 a **Big Data Analytics(23CS6PCBDA)** work prescribed for the said degree.

Pradeep Sadanand
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

Index Sheet

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 NCDC data set. b) find the mean max temperature for every month	25-31
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.	32-33
8	Write a Scala program to print numbers from 1 to 100 using for loop.	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.	35-37
10	Write a simple streaming program in Spark to receive text data streams on a particular port, perform basic text cleaning (like white space removal, stop words removal, lemmatization, etc.), and print the cleaned text on the screen. (Open Ended Question).	38-40

Github link: https://github.com/Praneeta205/BDA-6sem

Course Outcome

CO1	Apply the concept of NoSQL, Hadoop or Spark for a given task
CO2	Analyse big data analytics mechanisms that can be applied to obtain solution for a given problem.
CO3	Design and implement solutions using data analytics mechanisms for a given problem.

Lab 1:

Question: Perform basic CRUD (Create, Read, Update, Delete) operations in MongoDB.

Code with Output:

I. Create Database in MongoDB

1. Create a database named myDB.
2. Confirm the existence of your database.
3. List all databases.

II. CRUD Operations

4. Create a collection named Student.
5. Drop the Student collection.
6. Insert a document into Students collection.
7. Insert or update a document conditionally using upsert.
8. Perform the following FIND queries:
   - Find documents where StudName is "Aryan David".
   - Display only StudName and Grade without _id. o Find documents where Grade is 'VII'.
   - Find documents where Hobbies is either 'Chess' or 'Skating'.
   - Find documents where StudName starts with 'M'.
   - Find documents where StudName contains 'e'.
9. Count number of documents in Students collection.
   - Sort documents by StudName in descending order.

III. Import/Export

10. Import data from a CSV file into MongoDB.
11. Export data to a CSV file from MongoDB.

Code with output:

```
test> use mydb
switched to db mydb
mydb> // Create a collection named "Student"
... db.createCollection("Student")
...
{ ok: 1 }
mydb> // Drop the collection named "Student"
... db.Student.drop()
...
true
mydb> db.createCollection("Students")
{ ok: 1 }
mydb> db.Students.insert({
...   _id: 1,
...   StudName: "MichelleJacintha",
...   Grade: "VII",
...   Hobbies: "InternetSurfing"
... })
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{ acknowledged: true, insertedIds: { '0': 1 } }
mydb> db.Students.update(
...   { _id: 3, StudName: "AryanDavid", Grade: "VII" },
...   { $set: { Hobbies: "Chess" } },
...   { upsert: true }
... )
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: 3,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 1
}
```

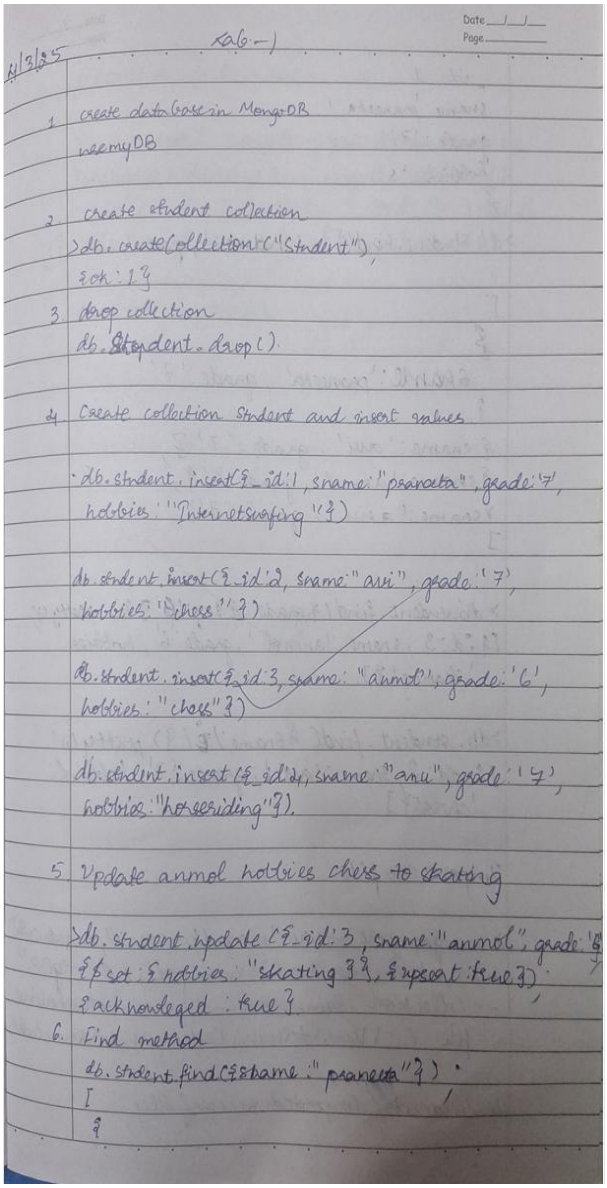
```
mydb> db.Students.insertMany([
...   {
...     _id: 2,
...     StudName: "Aryan David",
...     Grade: "VII",
...     Hobbies: "Skating"
...   },
...   {
...     _id: 4,
...     StudName: "Meera Joseph",
...     Grade: "IX",
...     Hobbies: "Reading"
...   },
...   {
...     _id: 5,
...     StudName: "Elena Thomas",
...     Grade: "VII",
...     Hobbies: "Skating"
...   }
... ])
...
{ acknowledged: true, insertedIds: { '0': 2, '1': 4, '2': 5 } }
mydb> db.Students.find({ StudName: "Aryan David" })
[
  { _id: 2, StudName: 'Aryan David', Grade: 'VII', Hobbies: 'Skating' }
]
mydb> db.Students.find({}, { StudName: 1, Grade: 1, _id: 0 })
[
  { StudName: 'MichelleJacintha', Grade: 'VII' },
  { Grade: 'VII', StudName: 'AryanDavid' },
  { StudName: 'Aryan David', Grade: 'VII' },
  { StudName: 'Meera Joseph', Grade: 'IX' },
  { StudName: 'Elena Thomas', Grade: 'VII' }
]
```

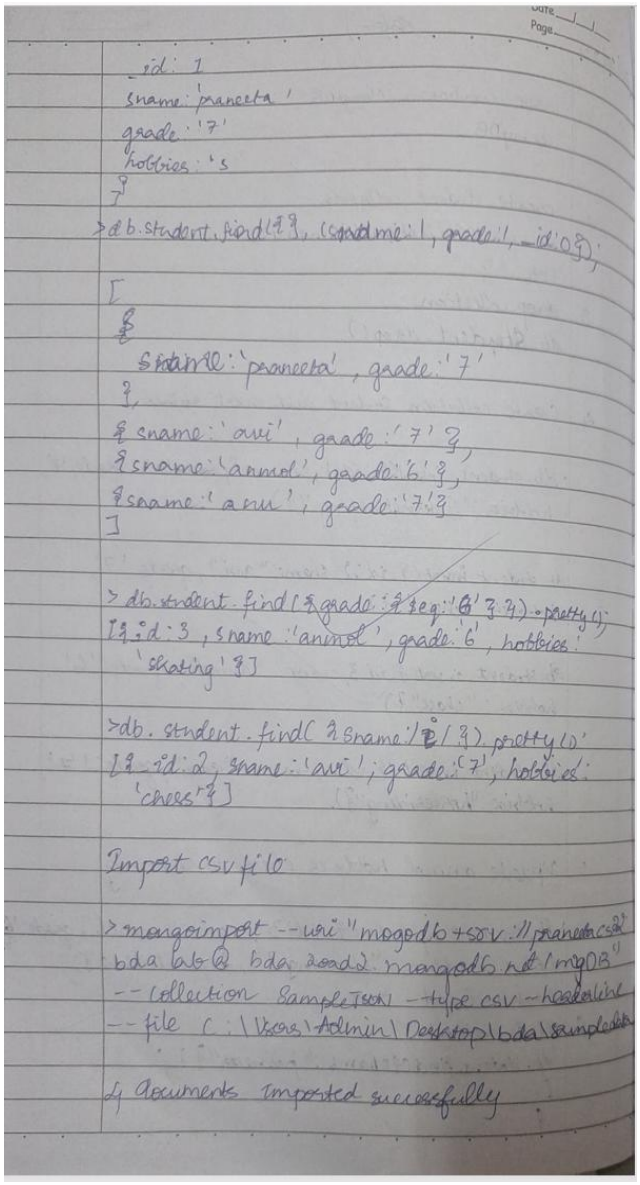
```
mydb> db.Students.find({ Grade: { $eq: "VII" } }).pretty()
[
  {
    _id: 1,
    StudName: 'MichelleJacintha',
    Grade: 'VII',
    Hobbies: 'InternetSurfing'
  },
  { _id: 3, Grade: 'VII', StudName: 'AryanDavid', Hobbies: 'Chess' },
  { _id: 2, StudName: 'Aryan David', Grade: 'VII', Hobbies: 'Skating' },
  {
    _id: 5,
    StudName: 'Elena Thomas',
    Grade: 'VII',
    Hobbies: 'Skating'
  }
]
mydb> db.Students.find({ Hobbies: { $in: ["Chess", "Skating"] } }).pretty()
[
  { _id: 3, Grade: 'VII', StudName: 'AryanDavid', Hobbies: 'Chess' },
  { _id: 2, StudName: 'Aryan David', Grade: 'VII', Hobbies: 'Skating' },
  {
    _id: 5,
    StudName: 'Elena Thomas',
    Grade: 'VII',
    Hobbies: 'Skating'
  }
]
mydb> db.Students.find({ StudName: /^M/ }).pretty()
[
  {
    _id: 1,
    StudName: 'MichelleJacintha',
    Grade: 'VII',
    Hobbies: 'InternetSurfing'
  },
  { _id: 4, StudName: 'Meera Joseph', Grade: 'IX', Hobbies: 'Reading' }
]
```

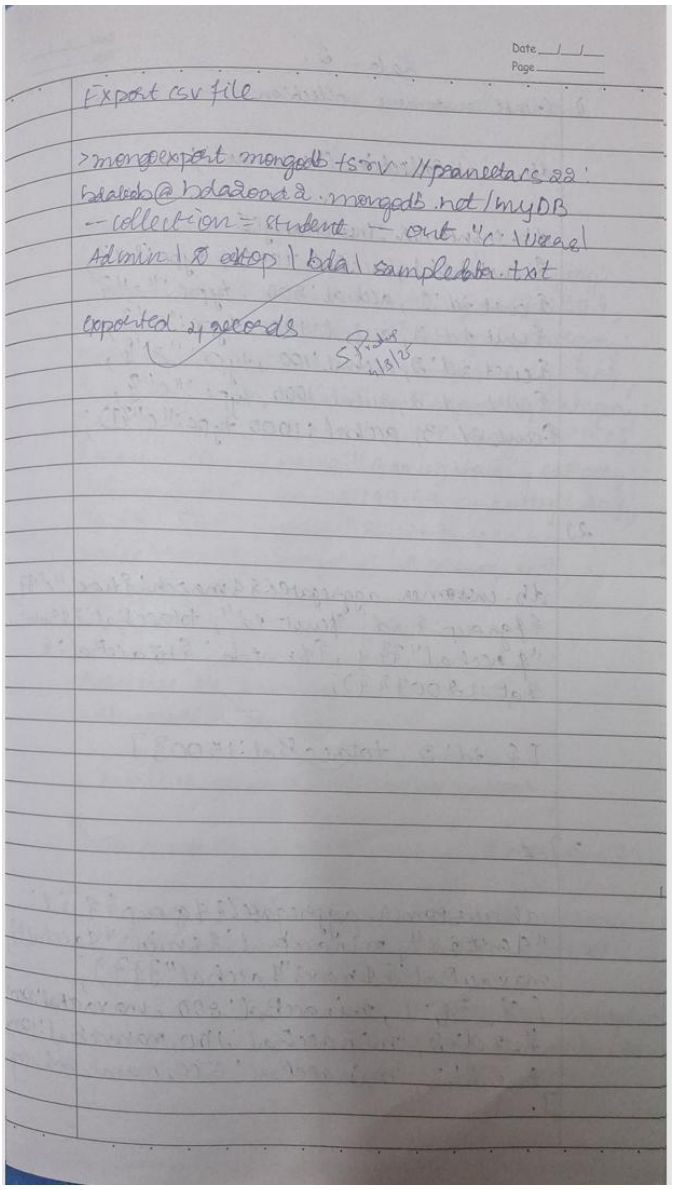
```
mydb> db.Students.find({ StudName: /e/ }).pretty()
[
  {
    _id: 1,
    StudName: 'MichelleJacintha',
    Grade: 'VII',
    Hobbies: 'InternetSurfing'
  },
  { _id: 4, StudName: 'Meera Joseph', Grade: 'IX', Hobbies: 'Reading' },
  {
    _id: 5,
    StudName: 'Elena Thomas',
    Grade: 'VII',
    Hobbies: 'Skating'
  }
]
mydb> db.Students.count()
DeprecationWarning: Collection.count() is deprecated. Use countDocuments or estimatedDocumentCount.
5
mydb> db.Students.find().sort({ StudName: -1 }).pretty()
[
  {
    _id: 1,
    StudName: 'MichelleJacintha',
    Grade: 'VII',
    Hobbies: 'InternetSurfing'
  },
  { _id: 4, StudName: 'Meera Joseph', Grade: 'IX', Hobbies: 'Reading' },
  {
    _id: 5,
    StudName: 'Elena Thomas',
    Grade: 'VII',
    Hobbies: 'Skating'
  },
  { _id: 3, Grade: 'VII', StudName: 'AryanDavid', Hobbies: 'Chess' },
  { _id: 2, StudName: 'Aryan David', Grade: 'VII', Hobbies: 'Skating' }
]
```

```
praneeta@LAPTOP-M1509RLF:~$ mongoexport --db=mydb --collection=Students --type=csv --fields=_id,StudName,Grade,Hobbies --out="/home/praneeta/stu.csv"
2025-05-27T01:10:27.322+0530    connected to: mongodb://localhost/
2025-05-27T01:10:27.323+0530    exported 5 records
praneeta@LAPTOP-M1509RLF:~$ cat stu.csv
_id,StudName,Grade,Hobbies
1,MichelleJacintha,VII,InternetSurfing
3,AryanDavid,VII,Chess
2,Aryan David,VII,Skating
4,Meera Joseph,IX,Reading
5,Elena Thomas,VII,Skating
praneeta@LAPTOP-M1509RLF:~$ nano ss.csv
praneeta@LAPTOP-M1509RLF:~$ mogoimport --db=mydb --collection=Students --type=csv--headerline--file="/home/praneeta/ss.csv"
mogoimport: command not found
praneeta@LAPTOP-M1509RLF:~$ mongoimport --db=mydb --collection=Students --type=csv--headerline--file="/home/praneeta/ss.csv"
2025-05-27T01:13:06.359+0530    error validating settings: unknown type csv--headerline--file=/home/praneeta/ss.csv
praneeta@LAPTOP-M1509RLF:~$ mongoimport --db=mydb --collection=Students --type=csv --headerline --file="/home/praneeta/ss.csv"
2025-05-27T01:13:51.554+0530    connected to: mongodb://localhost/
2025-05-27T01:13:51.557+0530    2 document(s) imported successfully. 0 document(s) failed to import.
```

Screenshot


Lab-1
1 create database in MongoDB
use myDB
2 create student collection
db.createCollection("Student")
3 drop collection
db.Student.drop()
4 Create collection Student and insert values
db.student.insert({_id:1, sname:"praneeta", grade:'7', hobbies:"Internetsurfing"})
db.student.insert({_id:2, sname:"avi", grade:'7', hobbies:"chess"})
db.student.insert({_id:3, sname:"anmol", grade:'6', hobbies:"chess"})
db.student.insert({_id:4, sname:"anu", grade:'7', hobbies:"horseriding"})
5 Update anmol hobbies chess to skating
db.student.update({_id:3, sname:"anmol", grade:'6'}, {$set:{hobbies:"skating"}}, {upsert:true})
{acknowledged:true}
6. Find method
db.student.find({sname:"praneeta"})


_id:1
sname:'praneeta'
grade:'7'
hobbies:'s'
db.student.find({}, {sname:1, grade:1, _id:0})
{sname:'praneeta', grade:'7'}
{sname:'avi', grade:'7'},
{sname:'anmol', grade:'6'},
{sname:'anu', grade:'7'}
db.student.find({grade:{$eq:'6'}}).pretty()
{_id:3, sname:'anmol', grade:'6', hobbies:'skating'}
db.student.find({sname:'avi'}).pretty()
{_id:2, sname:'avi', grade:'7', hobbies:'chess'}
Import csv file
mongoimport --uri "mongodb+srv://praneeta
--collection SampleJSON1 --type csv --headerline
--file C:\Users\Admin\Desktop\bda\sample
4 documents imported successfully



Section 1: Customer Collection Tasks

1. Create a collection named Customers with the following fields:
    - Cust_id (Customer ID)
    - Acc_Bal (Account Balance)
    - Acc_Type (Account Type)
2. Insert at least 5 records into the Customers collection.
3. Query: Display all customer records where:
    - Account Balance (Acc_Bal) is greater than 1200 AND
    - Account Type (Acc_Type) is 'Z'
    - Group the results by each Cust_id.
4. Determine the minimum and maximum account balance for each customer ID.

Section 2: E-Commerce Platform Schema & Queries

Design MongoDB Schema for the following:

- Products
  - Fields: product_id, name, category, price, quantity, etc.
- User Carts
  - Fields: user_id, products (array of product references with quantity)
- Orders
  - Fields: order_id, user_id, items, total_price, order_date, etc.

Product Queries

5. Retrieve all products from the product collection.
6. Retrieve products that belong to a specific category (e.g., 'Electronics').
7. Retrieve products where the quantity is greater than 0.
8. Retrieve products sorted by price in ascending order.
9. Retrieve products with a price less than or equal to $100.

Cart and Order Queries

10. Retrieve products added to a user's cart, where the user_id is "789ghi...".
11. Retrieve orders placed by a user, where the user_id is "123abc...".
12. Retrieve the total price of orders placed by user "123abc...".
13.

Section 3: Additional Aggregation Queries (Assignment-3 Design)

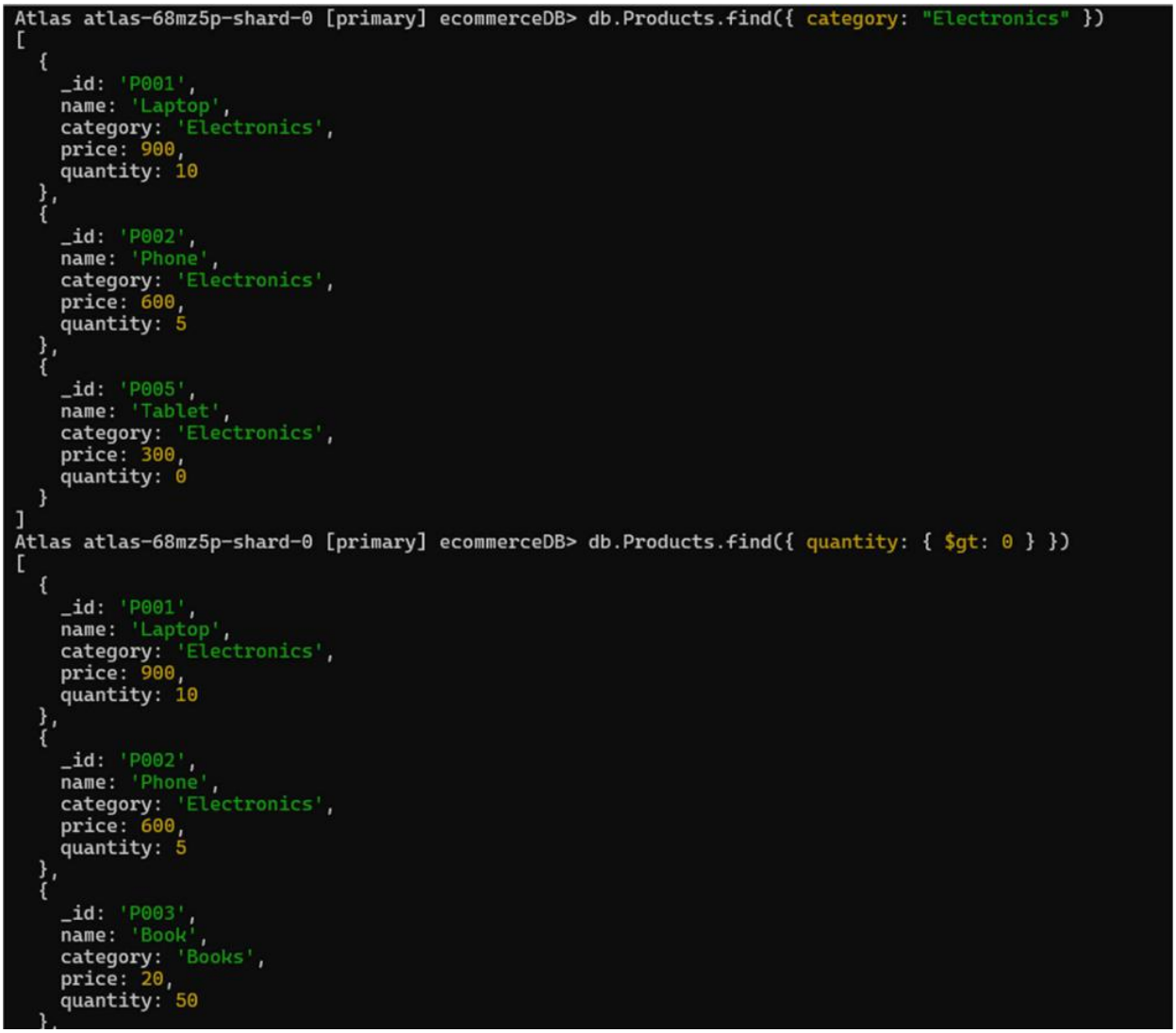
13. Calculate the total number of products available in each category.
14. Calculate the total price of products in each category.
15. Find the average price of all products.
16. Find products with a quantity less than 10.
17. Sort products by price in descending order.
18. Calculate the total price of orders placed by each user.
19. Find the user who has the highest total order value.
20. Find the average total price of all orders.

Code with Output:

```
Atlas atlas-68mz5p-shard-0 [primary] myDB> use ecommerceDB
switched to db ecommerceDB
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB> db.Customers.insertMany([
...   { Cust_id: "C001", Acc_Bal: 1000, Acc_Type: "Z" },
...   { Cust_id: "C002", Acc_Bal: 1300, Acc_Type: "Z" },
...   { Cust_id: "C003", Acc_Bal: 1500, Acc_Type: "S" },
...   { Cust_id: "C004", Acc_Bal: 700,  Acc_Type: "Z" },
...   { Cust_id: "C005", Acc_Bal: 1600, Acc_Type: "Z" }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('683429718d97f89e1f6c4bd0'),
    '1': ObjectId('683429718d97f89e1f6c4bd1'),
    '2': ObjectId('683429718d97f89e1f6c4bd2'),
    '3': ObjectId('683429718d97f89e1f6c4bd3'),
    '4': ObjectId('683429718d97f89e1f6c4bd4')
  }
}
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB> db.Customers.find({ Acc_BaAcc_Bal: { $gt: 1200 }, Acc_Type: "Z" })
[
  {
    _id: ObjectId('683429718d97f89e1f6c4bd1'),
    Cust_id: 'C002',
    Acc_Bal: 1300,
    Acc_Type: 'Z'
  },
  {
    _id: ObjectId('683429718d97f89e1f6c4bd4'),
    Cust_id: 'C005',
    Acc_Bal: 1600,
    Acc_Type: 'Z'
  }
]
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB> db.Customers.aggregate([
...   {
...     $group: {
...       _id: "$Cust_id",
...       MinBal: { $min: "$Acc_Bal" },
...       MaxBal: { $max: "$Acc_Bal" }
...     }
...   }
... ])
[
  { _id: 'C002', MinBal: 1300, MaxBal: 1300 },
  { _id: 'C001', MinBal: 1000, MaxBal: 1000 },
  { _id: 'C005', MinBal: 1600, MaxBal: 1600 },
  { _id: 'C004', MinBal: 700, MaxBal: 700 },
  { _id: 'C003', MinBal: 1500, MaxBal: 1500 }
```

```
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB> db.Products.insertMany([
...   { _id: "P001", name: "Laptop", category: "Electronics", price: 900, quantity: 10 },
...   { _id: "P002", name: "Phone", category: "Electronics", price: 600, quantity: 5 },
...   { _id: "P003", name: "Book", category: "Books", price: 20, quantity: 50 },
...   { _id: "P004", name: "Shoes", category: "Fashion", price: 70, quantity: 25 },
...   { _id: "P005", name: "Tablet", category: "Electronics", price: 300, quantity: 0 }
... ])
...
{
  acknowledged: true,
  insertedIds: { '0': 'P001', '1': 'P002', '2': 'P003', '3': 'P004', '4': 'P005' }
}
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB> db.Carts.insertMany([
...   {
...     user_id: "789ghi",
...     products: [
...       { product_id: "P001", quantity: 1 },
...       { product_id: "P003", quantity: 2 }
...     ]
...   }
... ])
...
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('683429bf8d97f89e1f6c4bd5') }
}
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB> db.Orders.insertMany([
...   {
...     user_id: "123abc",
...     products: [
...       { product_id: "P001", quantity: 1, price: 900 },
...       { product_id: "P004", quantity: 2, price: 70 }
...     ],
...     total_price: 1040
...   }
... ])
...
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('683429c38d97f89e1f6c4bd6') }
}
```

```
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB> db.Products.find()
[
  {
    _id: 'P001',
    name: 'Laptop',
    category: 'Electronics',
    price: 900,
    quantity: 10
  },
  {
    _id: 'P002',
    name: 'Phone',
    category: 'Electronics',
    price: 600,
    quantity: 5
  },
  {
    _id: 'P003',
    name: 'Book',
    category: 'Books',
    price: 20,
    quantity: 50
  },
  {
    _id: 'P004',
    name: 'Shoes',
    category: 'Fashion',
    price: 70,
    quantity: 25
  },
  {
    _id: 'P005',
    name: 'Tablet',
    category: 'Electronics',
    price: 300,
    quantity: 0
  }
]
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB> db.Products.find({ category: "Electronics" })
[
```



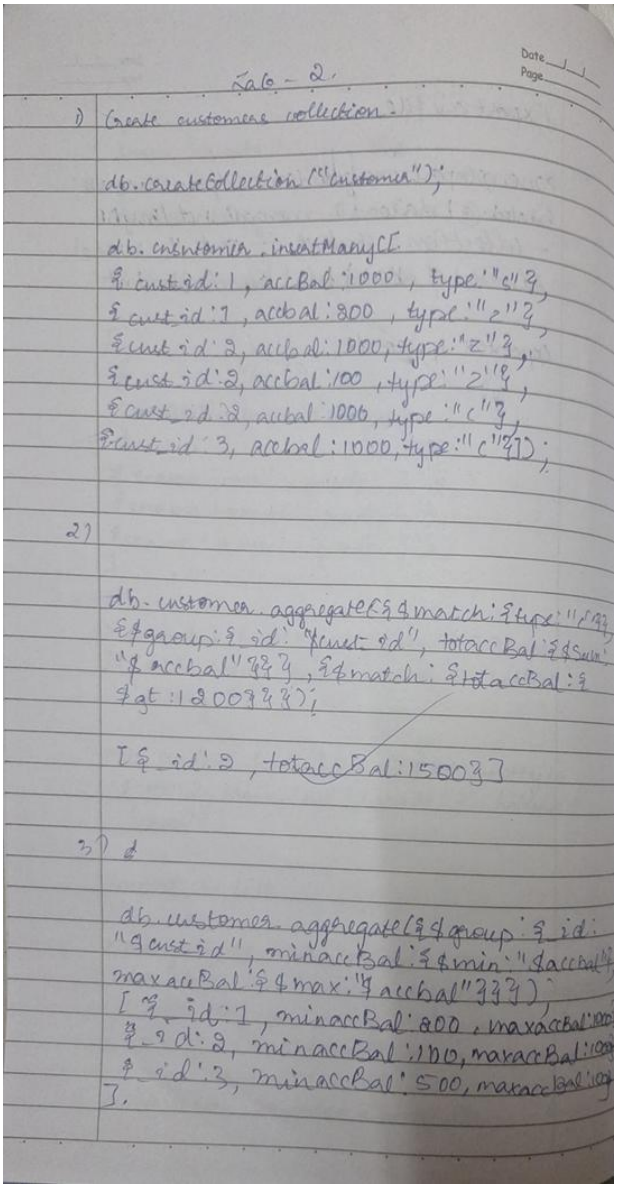
```
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB> db.Products.find({ category: "Electronics" })
[
  {
    _id: 'P001',
    name: 'Laptop',
    category: 'Electronics',
    price: 900,
    quantity: 10
  },
  {
    _id: 'P002',
    name: 'Phone',
    category: 'Electronics',
    price: 600,
    quantity: 5
  },
  {
    _id: 'P005',
    name: 'Tablet',
    category: 'Electronics',
    price: 300,
    quantity: 0
  }
]
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB> db.Products.find({ quantity: { $gt: 0 } })
[
  {
    _id: 'P001',
    name: 'Laptop',
    category: 'Electronics',
    price: 900,
    quantity: 10
  },
  {
    _id: 'P002',
    name: 'Phone',
    category: 'Electronics',
    price: 600,
    quantity: 5
  },
  {
    _id: 'P003',
    name: 'Book',
    category: 'Books',
    price: 20,
    quantity: 50
  },
```

```
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB> db.Products.find().sort({ price: 1 })
[
  {
    _id: 'P003',
    name: 'Book',
    category: 'Books',
    price: 20,
    quantity: 50
  },
  {
    _id: 'P004',
    name: 'Shoes',
    category: 'Fashion',
    price: 70,
    quantity: 25
  },
  {
    _id: 'P005',
    name: 'Tablet',
    category: 'Electronics',
    price: 300,
    quantity: 0
  },
  {
    _id: 'P002',
    name: 'Phone',
    category: 'Electronics',
    price: 600,
    quantity: 5
  },
  {
    _id: 'P001',
    name: 'Laptop',
    category: 'Electronics',
    price: 900,
    quantity: 10
  }
]
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB> db.Products.find({ price: { $lte: 100 } })
[
  {
    _id: 'P003',
    name: 'Book',
    category: 'Books',
    price: 20,
    quantity: 50
  },
  {
```

```
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB> db.Carts.find({ user_id: "789ghi" })
[
  {
    _id: ObjectId('683429bf8d97f89e1f6c4bd5'),
    user_id: '789ghi',
    products: [
      { product_id: 'P001', quantity: 1 },
      { product_id: 'P003', quantity: 2 }
    ]
  }
]
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB> db.Orders.find({ user_id: "123abc" })
[
  {
    _id: ObjectId('683429c38d97f89e1f6c4bd6'),
    user_id: '123abc',
    products: [
      { product_id: 'P001', quantity: 1, price: 900 },
      { product_id: 'P004', quantity: 2, price: 70 }
    ],
    total_price: 1040
  }
]
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB> db.Orders.aggregate([
...   { $match: { user_id: "123abc" } },
...   { $group: { _id: "$user_id", totalSpent: { $sum: "$total_price" } } }
... ])
[ { _id: '123abc', totalSpent: 1040 } ]
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB> db.Products.aggregate([
...   { $group: { _id: "$category", totalProducts: { $sum: 1 } } }
... ])
[
  { _id: 'Fashion', totalProducts: 1 },
  { _id: 'Books', totalProducts: 1 },
  { _id: 'Electronics', totalProducts: 3 }
]
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB> db.Products.aggregate([
...   { $group: { _id: "$category", totalPrice: { $sum: "$price" } } }
... ])
[
  { _id: 'Electronics', totalPrice: 1800 },
  { _id: 'Books', totalPrice: 20 },
  { _id: 'Fashion', totalPrice: 70 }
]
```

```
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB> db.Orders.aggregate([
...   { $group: { _id: "$user_id", totalSpent: { $sum: "$total_price" } } }
... ])
[ { _id: '123abc', totalSpent: 1040 } ]
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB> db.Orders.aggregate([
...   { $group: { _id: "$user_id", totalSpent: { $sum: "$total_price" } } },
...   { $sort: { totalSpent: -1 } },
...   { $limit: 1 }
... ])
[ { _id: '123abc', totalSpent: 1040 } ]
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB> db.Orders.aggregate([
...   { $group: { _id: null, avgOrderValue: { $avg: "$total_price" } } }
... ])
[ { _id: null, avgOrderValue: 1040 } ]
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB>
```

Screenshots



Lab - 2.

1) Create customers collection:

db.createCollection("customers");

db.customers.insertMany([
{cust_id:1, accBal:1000, type:"c"},
{cust_id:1, accbal:800, type:"z"},
{cust_id:2, accbal:1000, type:"z"},
{cust_id:2, accbal:100, type:"z"},
{cust_id:2, aucbal:1000, type:"c"},
{cust_id:3, accbal:1000, type:"c"}]);

2)

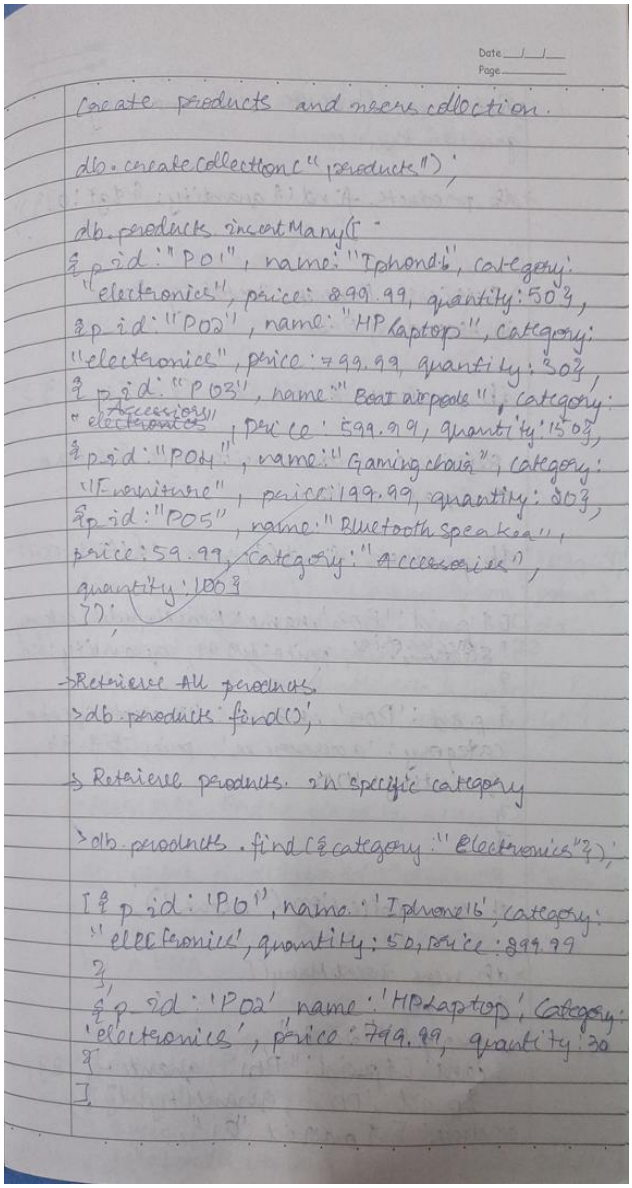
db.customer.aggregate([{$match:{type:"z"}},
{$group:{_id:"$cust_id", totaccBal:{$sum:"$accbal"}}},
{$match:{totaccBal:{$gt:1200}}}]);

[{_id:2, totaccBal:1500}]

3) a

db.customers.aggregate([{$group:{_id:"$cust_id", minaccBal:{$min:"$accbal"}, maxaccBal:{$max:"$accbal"}}}]);

[{_id:1, minaccBal:800, maxaccBal:...},
{_id:2, minaccBal:100, maxaccBal:...},
{_id:3, minaccBal:500, maxaccBal:...}].



Create products and users collection.

db.createCollection("products");

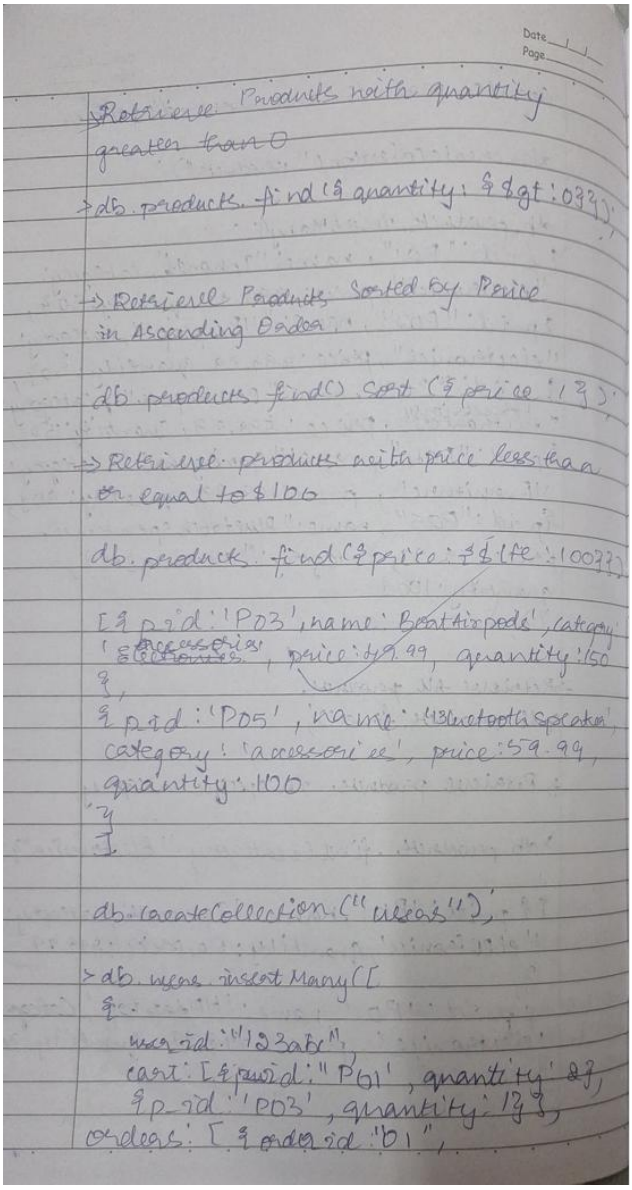
db.products.insertMany([
{p_id:"P01", name:"Iphone16", category:"electronics", price:899.99, quantity:50},
{p_id:"P02", name:"HP laptop", category:"electronics", price:799.99, quantity:30},
{p_id:"P03", name:"Boat airpods", category:"electronics", price:599.99, quantity:150},
{p_id:"P04", name:"Gaming chair", category:"Furniture", price:199.99, quantity:80},
{p_id:"P05", name:"Bluetooth speaker", price:59.99, category:"Accessories", quantity:100}
]);

->Retrieve all products.
>db.products.find();

->Retrieve products in specific category

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

[{p_id:'P01', name:'Iphone16', category:'electronics', quantity:50, price:899.99},
{p_id:'P02', name:'HP laptop', category:'electronics', price:799.99, quantity:30}
].



Retrieve Products with quantity greater than 0

> db.products.find({quantity: {$gt: 0}})

-> Retrieve Products Sorted by Price in Ascending Order

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

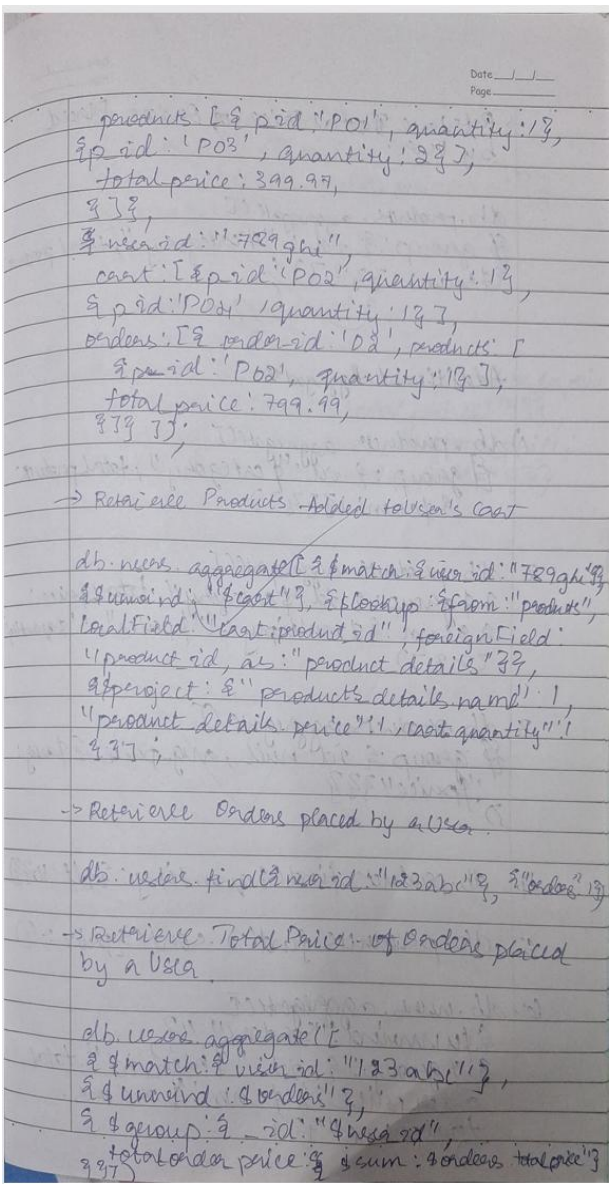
-> Retrieve products with price less than or equal to $100

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

[{ pid: 'P03', name: 'BeatAirpods', category: 'accessories', price: 69.99, quantity: 150 },
{ pid: 'P05', name: '13bluetooth speaker', category: 'accessories', price: 59.99, quantity: 100 }]

db.createCollection("users");

> db.users.insertMany([
{
user_id: "123abc",
cart: [{pid: "P01", quantity: 2},
{p_id: 'P03', quantity: 1}],
orders: [{order_id: 'O1',



products: [{pid: 'P01', quantity: 1},
{p_id: 'P03', quantity: 2}],
total_price: 399.97,
}]},
{user_id: "789ghi",
cart: [{p_id: 'P02', quantity: 1},
{p_id: 'P04' /quantity: 1}],
orders: [{order_id: 'O2', products: [
{pid: 'P02', quantity: 1}],
total_price: 799.99,
}]},

-> Retrieve Products Added to User's Cart

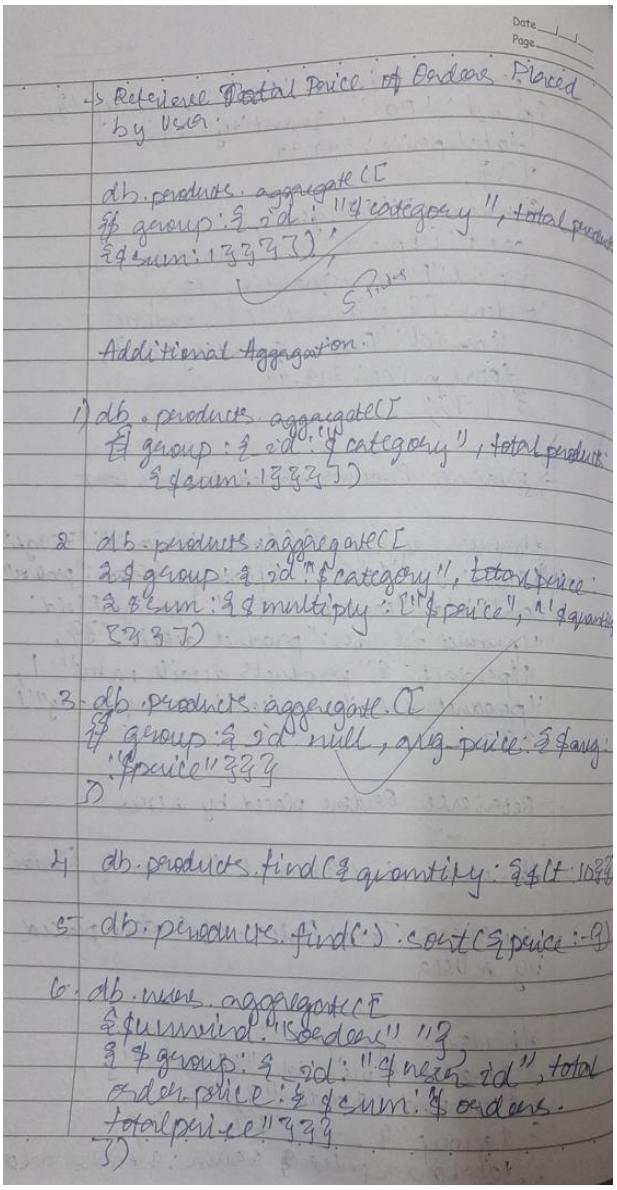
db.users.aggregate([{$match: {user_id: "789ghi"}},
{$unwind: "$cart"}, {$lookup: {from: "products",
localField: "cart.product_id", foreignField:
"product_id", as: "product details"}},
{$project: {"product's details.name": 1,
"product details.price": 1, cart.quantity": 1
}}]);

-> Retrieve Orders placed by a User.

db.users.find({user_id: "123abc"}, {"orders":1})

-> Retrieve Total Price of Orders placed by a User.

db.users.aggregate([
{$match: {user_id: "123abc"}},
{$unwind: "$orders"},
{$group: {_id: "$user_id",
total order price: {$sum: "$orders.total price"}}}])



5. Retrieve Total Price of Orders Placed by User.

```
db.products.aggregate([
  { $group: { _id: "$category", totalPrice: { $sum: "$price" } } }
])
```

Additional Aggregation:

1) db.products.aggregate([
   { $group: { _id: "$category", totalPrice: { $sum: "$price" } } }
])

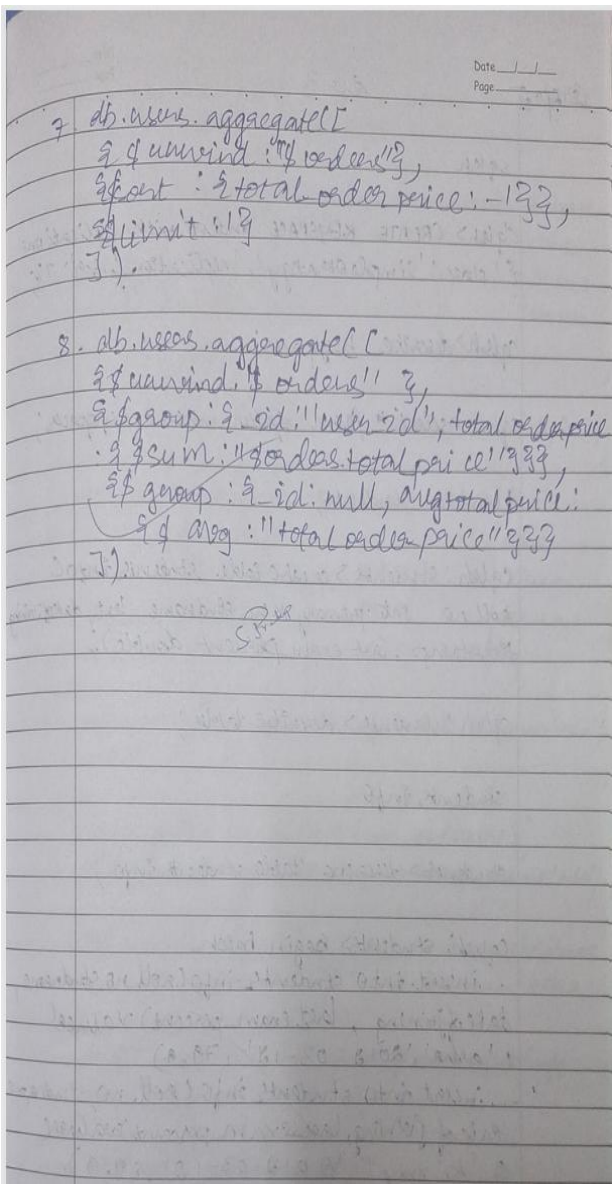
2. db.products.aggregate([
   { $group: { _id: "$category", totalPrice: { $sum: { $multiply: ["$price", "$quantity"] } } } }
])

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

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

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

6. db.users.aggregate([
   { $unwind: "$orders" },
   { $group: { _id: "$user_id", totalOrderPrice: { $sum: "$orders.totalPrice" } } }
])



7. db.users.aggregate([
   { $unwind: "$orders" },
   { $sort: { totalOrderPrice: -1 } },
   { $limit: 1 }
])

8. db.users.aggregate([
   { $unwind: "$orders" },
   { $group: { _id: "$user_id", totalOrderPrice: { $sum: "$orders.totalPrice" } } },
   { $group: { _id: null, avgTotalPrice: { $avg: "totalOrderPrice" } } }
])

Lab 2:

Question: Perform the following DB operations using Cassandra.

1. Create a keyspace by name Employee

2. Create a column family by name Employee-Info with attributes Emp_Id Primary Key, Emp_Name, Designation, Date_of_Joining, Salary, Dept_Name

3. Insert the values into the table in batch

4. Update Employee name and Department of Emp-Id 121

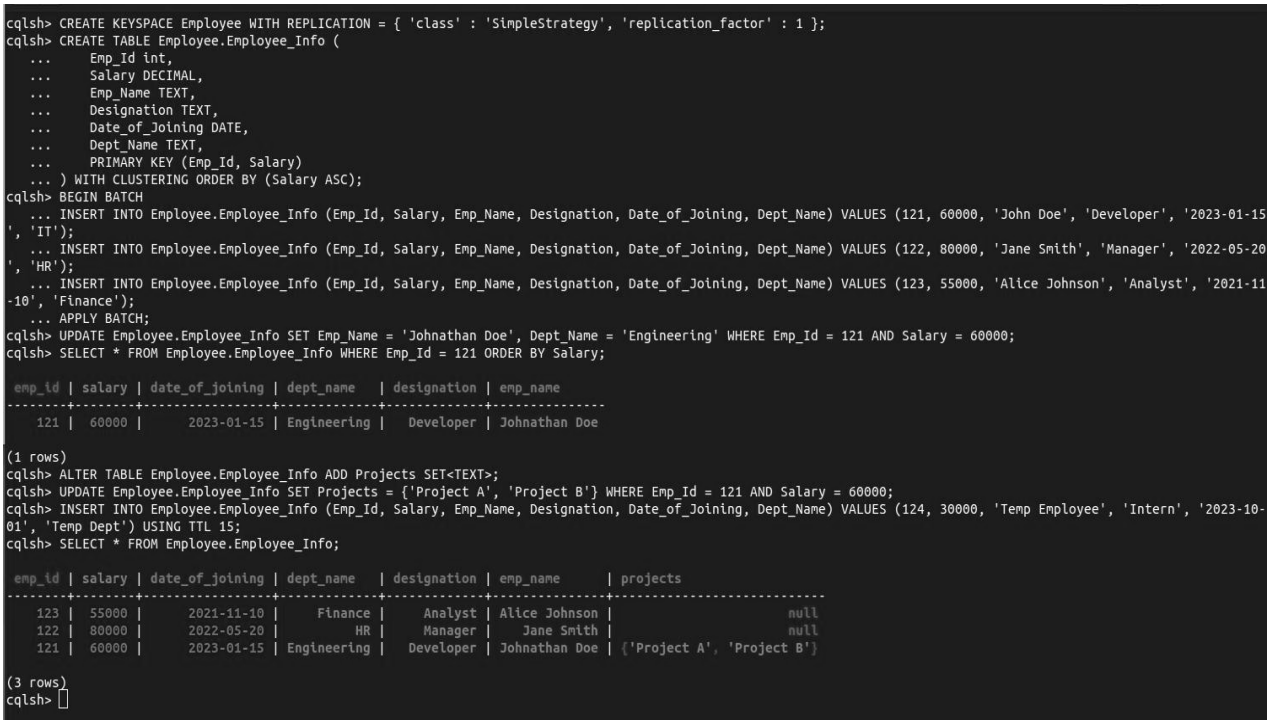
5. Sort the details of Employee records based on salary

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

7. Update the altered table to add project names.

8. Create a TTL of 15 seconds to display the values of Employees.

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>
```

Lab 3:

Perform the following DB operations using Cassandra.

1. Create a keyspace by name Library

2. 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

3. Insert the values into the table in batch

4. Display the details of the table created and increase the value of the counter

5. Write a query to show that a student with id 112 has taken a book “BDA” 2 times.

6. Export the created column to a csv file

7.Import a given csv dataset from local file system into Cassandra column family

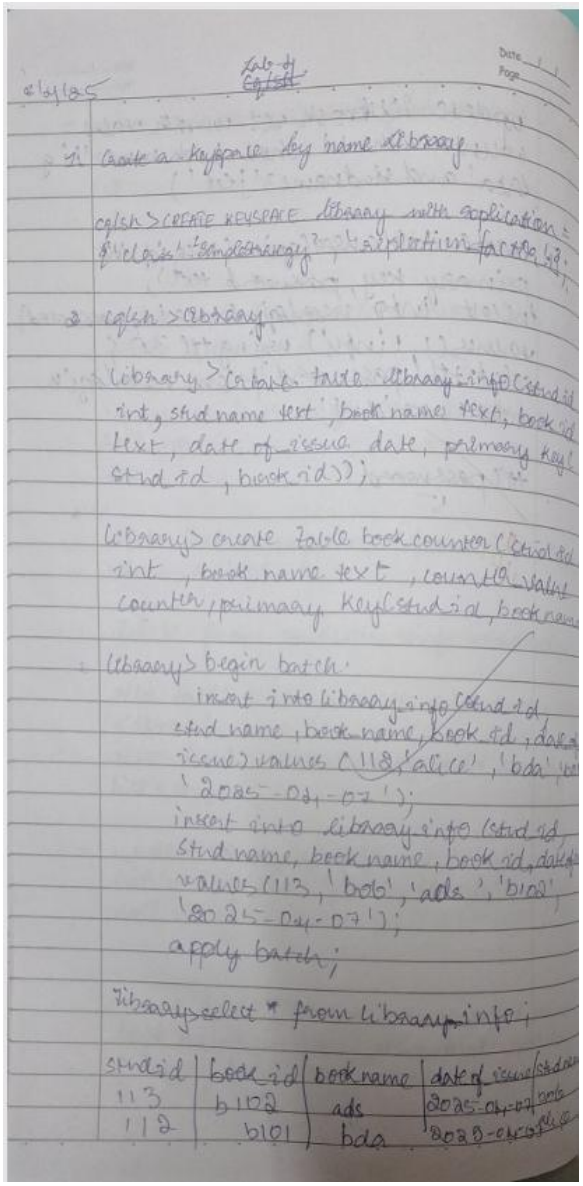
Code with Output:

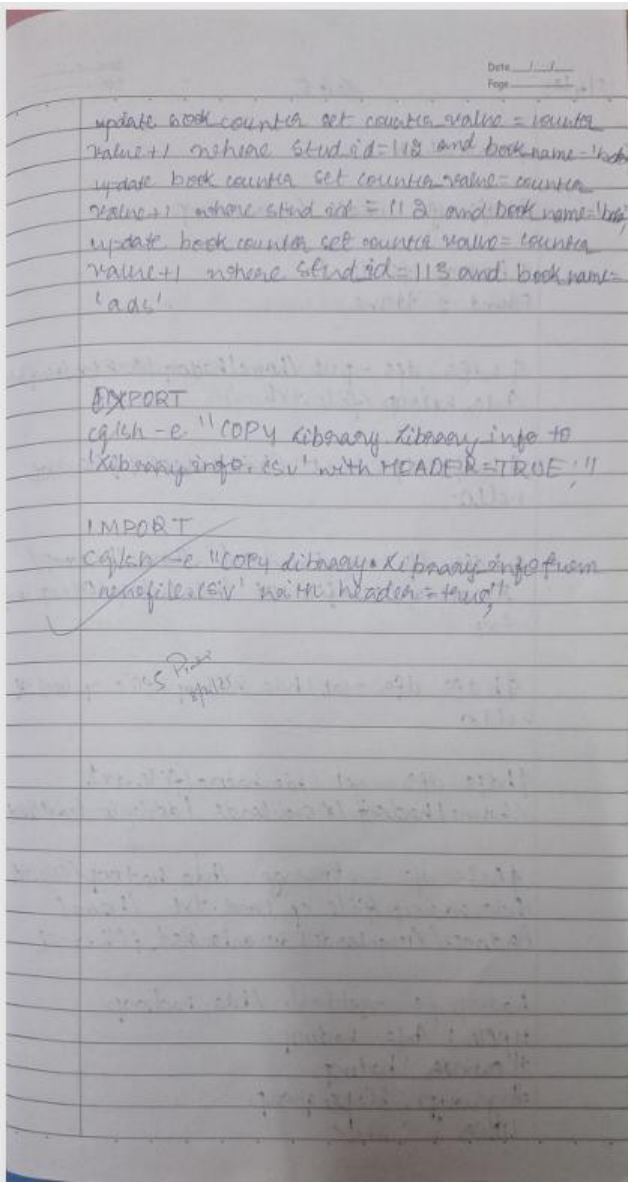
```
cqlsh> create keyspace lib with replication={'class':'SimpleStrategy','replication_factor':1};
cqlsh> select * from system_schema.keyspaces;

 keyspace_name      | durable_writes | replication
--------------------+----------------+-----------------------------------------------------------------------------------------
       system_auth |           True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '1'}
     system_schema |           True |                                {'class': 'org.apache.cassandra.locator.LocalStrategy'}
 system_distributed |           True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
            system |           True |                                {'class': 'org.apache.cassandra.locator.LocalStrategy'}
               lib |           True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '1'}
     system_traces |           True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '2'}
          students |           True | {'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '1'}
 cqlsh:lib> begin batch
        ... insert into lib_info(studid,bookid,stud_name,bookname,date_of_issue)val
 ues(1,101,'asha','bda','2012-03-12')
        ... insert into lib_info(studid,bookid,stud_name,bookname,date_of_issue)val
 ues(2,102,'ashi','ads','2012-03-12')
        ... apply batch;
 cqlsh:lib> select * from lib_info;

  studid | bookid | bookname | date_of_issue                   | stud_name
 --------+--------+----------+---------------------------------+-----------
       1 |    101 |      bda | 2012-03-11 18:30:00.000000+0000 |      asha
       2 |    102 |      ads | 2012-03-11 18:30:00.000000+0000 |      ashi

 (2 rows)
 cqlsh:lib> update lib_count set counter_val=counter_val+2 where studid='101' and bookname='bda';
 cqlsh:lib> select * from lib_count where studid=1 and bookid=101;

  studid | bookid | counter_val
 --------+--------+-------------
       1 |    101 |           2

 (1 rows)
```

```
praneeta@LAPTOP-M1509RLF:~$ cqlsh -e "copy lib.lib_info to 'libinfo.csv' with header=true;"
Using 15 child processes

Starting copy of lib.lib_info with columns [studid, bookid, bookname, date_of_issue, stud_name].
Processed: 3 rows; Rate:      16 rows/s; Avg. rate:      16 rows/s
3 rows exported to 1 files in 0.221 seconds.
```

```
praneeta@LAPTOP-M1509RLF:~$ cqlsh -e "copy lib.lib_info from 'l.csv' ;"
Using 15 child processes

Starting copy of lib.lib_info with columns [studid, bookid, bookname, date_of_issue, stud_name].
Processed: 1 rows; Rate:       2 rows/s; Avg. rate:       2 rows/s
1 rows imported from 1 files in 0.403 seconds (0 skipped).
```

Screenshots





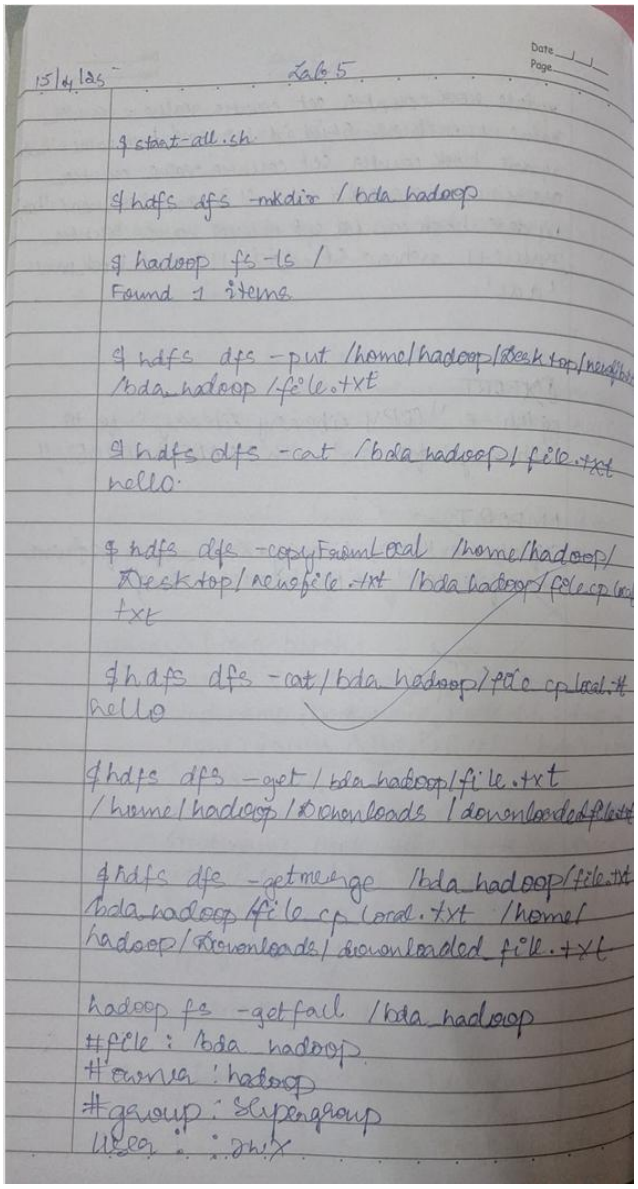
Lab 4:

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

Code with Output:

```
praneeta@LAPTOP-M1509RLF:~$ hdfs dfs -mkdir /bda_hadoop
praneeta@LAPTOP-M1509RLF:~$ hadoop fs -ls /
Found 5 items
drwxr-xr-x   - praneeta supergroup          0 2025-05-26 11:01 /bda
drwxr-xr-x   - praneeta supergroup          0 2025-05-26 16:51 /bda_hadoop
drwxr-xr-x   - praneeta supergroup          0 2025-05-26 16:42 /lhs
drwxr-xr-x   - praneeta supergroup          0 2025-05-26 11:45 /rgs
drwxr-xr-x   - praneeta supergroup          0 2025-05-26 16:03 /rig
praneeta@LAPTOP-M1509RLF:~$ hdfs dfs -put /home/praneeta/input.txt /bda_hadoop/text.txt
praneeta@LAPTOP-M1509RLF:~$ hdfs dfs -cat /bda_hadoop/text.txt
hi how are you
how is your job
how is your family
how is your brother
how is your sister
praneeta@LAPTOP-M1509RLF:~$ hdfs dfs -copyFromLocal /home/praneeta/input.txt /bda_hadoop/newtextcp.txt
praneeta@LAPTOP-M1509RLF:~$ hdfs dfs -cat /bda_hadoop/newtextcp.txt
hi how are you
how is your job
how is your family
how is your brother
how is your sister
praneeta@LAPTOP-M1509RLF:~$ hdfs dfs -get /bda_hadoop/text.txt /home/praneeta/newtext.txt
praneeta@LAPTOP-M1509RLF:~$ hdfs dfs -getmerge /bda_hadoop/text.txt /bda_hadoop/newtextcp.txt /home/praneeta/newtextmerge.txt
praneeta@LAPTOP-M1509RLF:~$ hadoop fs -getfacl /bda_hadoop
# file: /bda_hadoop
# owner: praneeta
# group: supergroup
user::rwx
group::r-x
other::r-x

praneeta@LAPTOP-M1509RLF:~$ hdfs dfs -copyToLocal /bda_hadoop/text.txt /home/praneeta
praneeta@LAPTOP-M1509RLF:~$ hadoop fs -mv /bda_hadoop /abc
praneeta@LAPTOP-M1509RLF:~$ hadoop fs -ls /abc
Found 2 items
-rw-r--r--   3 praneeta supergroup         89 2025-05-26 16:53 /abc/newtextcp.txt
-rw-r--r--   3 praneeta supergroup         89 2025-05-26 16:53 /abc/text.txt
praneeta@LAPTOP-M1509RLF:~$ hadoop fs -cp /abc/ /hadoop_lab
praneeta@LAPTOP-M1509RLF:~$
```

Screenshots:


15/4/25 Lab 5

$ 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

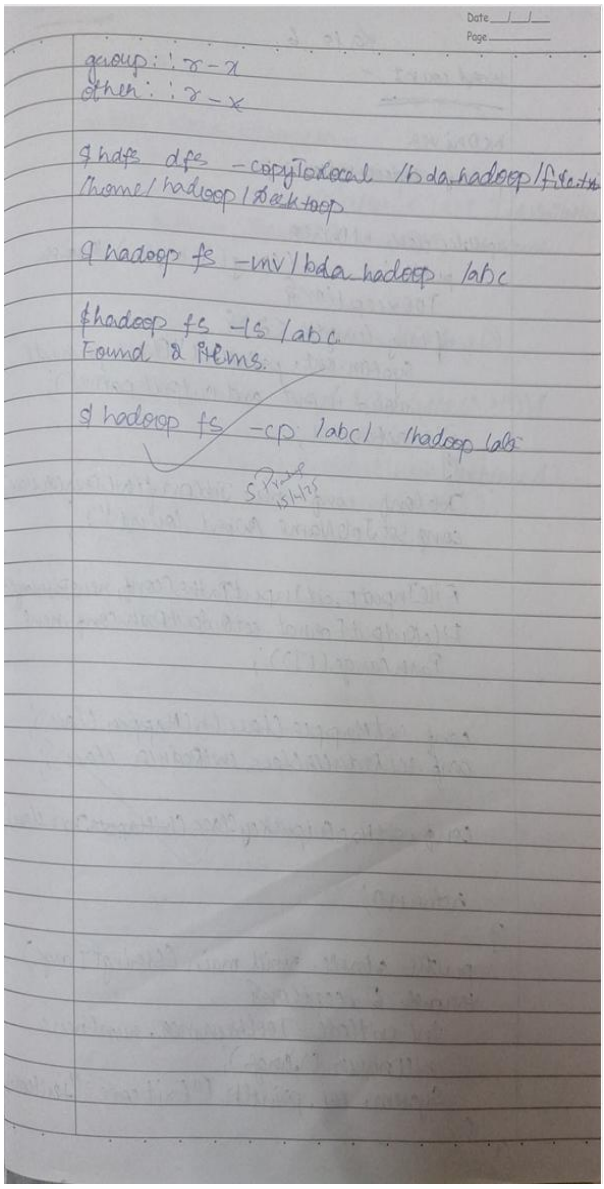
$ hdfs dfs -copyFromLocal /home/hadoop/Desktop/newfile.txt /bda_hadoop/filecp(local).txt

$ 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

5 Prof 15/4/25

Lab 5:

Question: Implement Wordcount program on Hadoop framework

Code with Output:

WCMapper.java

```
package wordcount;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class WCMapper extends MapReduceBase implements Mapper<LongWritable, Text, Text,
IntWritable> {

   // Map function
   public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output,
Reporter rep) throws IOException {
      String line = value.toString();
      // Splitting the line on spaces
      for (String word : line.split(" ")) {
         if (word.length() > 0) {
            output.collect(new Text(word), new IntWritable(1));
         }
      }
   }
}
```

WCReducer.java

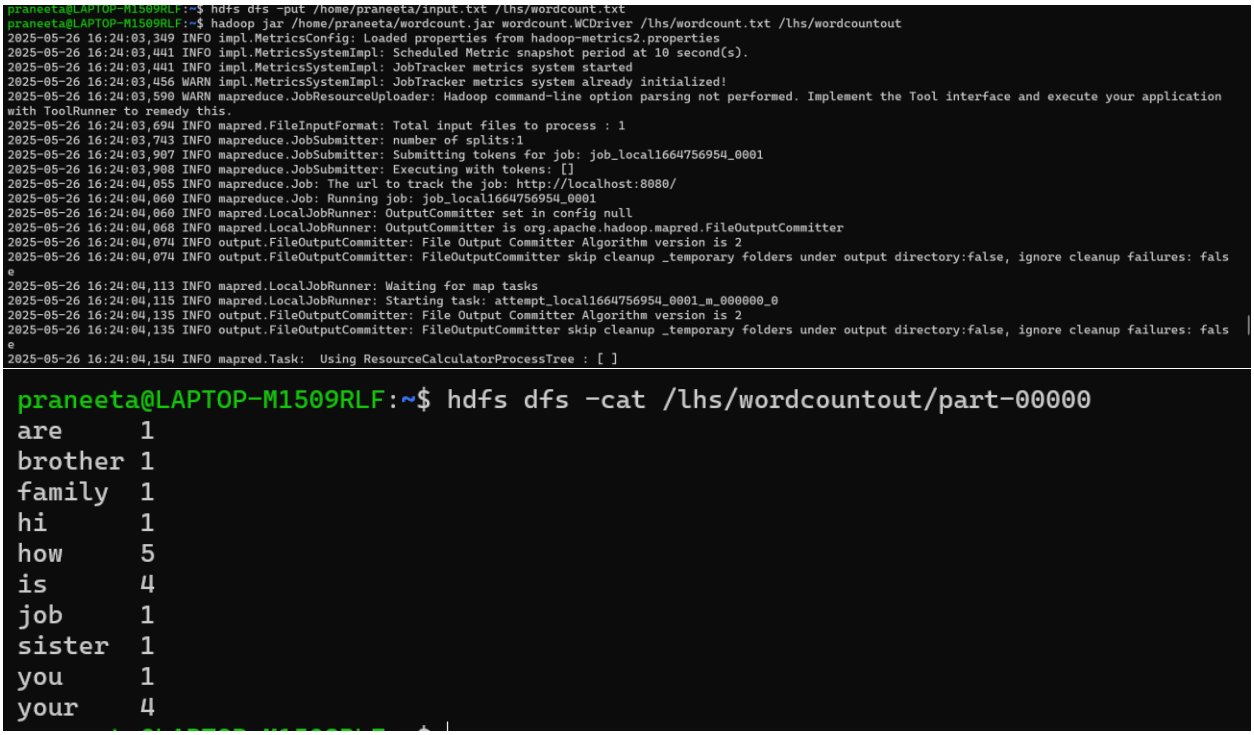
```
package wordcount;
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;

public class WCReducer extends MapReduceBase implements Reducer<Text, IntWritable, Text,
IntWritable> {
```

```
   // Reduce function
   public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable>
output, Reporter rep) throws IOException {
      int count = 0;
      // Counting the frequency of each word
      while (values.hasNext()) {
         IntWritable i = values.next();
         count += i.get();
      }
      output.collect(key, new IntWritable(count));
   }
}
```

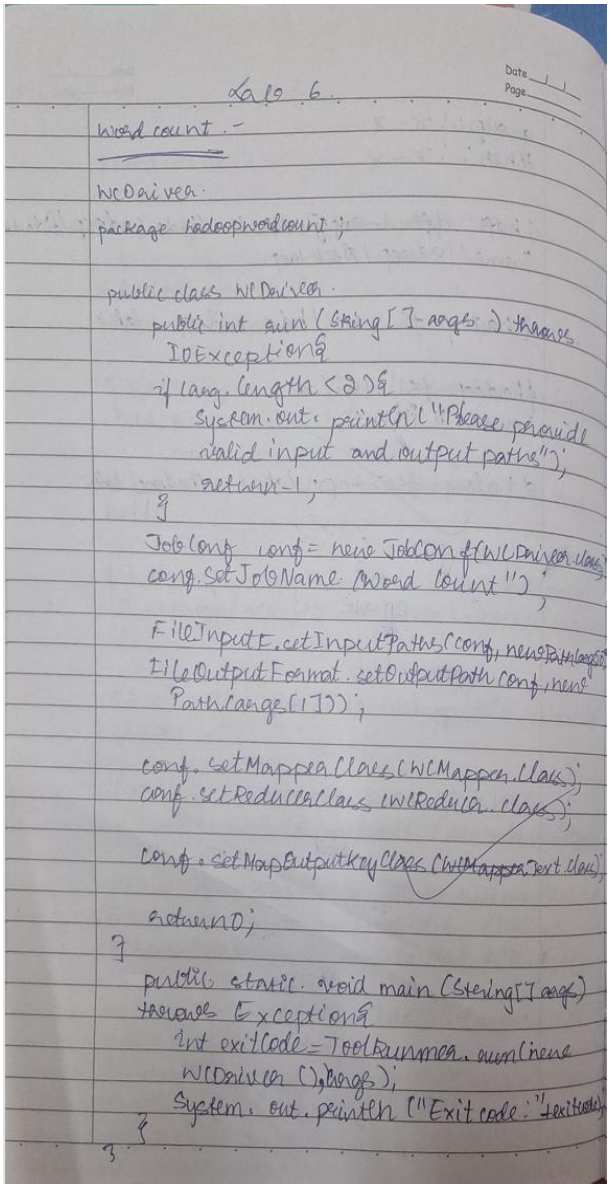
WCDriver.java

```
package wordcount;
import java.io.IOException;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class WCDriver extends Configured implements Tool {

   public int run(String[] args) throws IOException {
      if (args.length < 2) {
         System.out.println("Please provide valid input and output paths");
         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.setReducerClass(WCReducer.class);

      conf.setMapOutputKeyClass(Text.class);
      conf.setMapOutputValueClass(IntWritable.class);

      conf.setOutputKeyClass(Text.class);
```

```
        conf.setOutputValueClass(IntWritable.class);

        JobClient.runJob(conf);
        return 0;
    }

    // Main Method
    public static void main(String[] args) throws Exception {
        int exitCode = ToolRunner.run(new WCDriver(), args);
        System.out.println("Job Finished with exit code: " + exitCode);
    }
}
```



Screenshots:

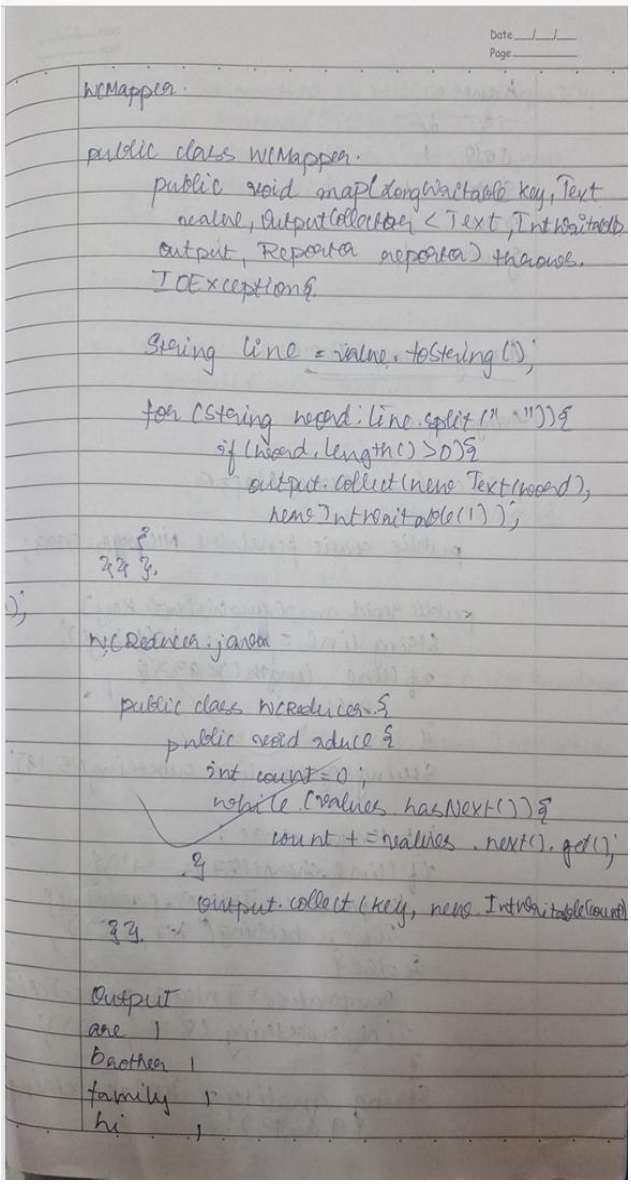


Lab 6.

Word count :-

WCDriver.

```
package hadoopwordcount;

public class WCDriver {
    public int run(String[] args) throws
    IOException{
    if (args.length < 2){
        System.out.println("Please provide
        valid input and output paths");
        return -1;
    }
    JobConf conf = new JobConf(WCDriver.class);
    conf.setJobName("Word Count");

    FileInputFormat.setInputPaths(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new
    Path(args[1]));

    conf.setMapperClass(WCMapper.class);
    conf.setReducerClass(WCReducer.class);

    conf.setMapOutputKeyClass(Text.class);

    return 0;
}
    public static void main(String[] args)
    throws Exception{
        int exitCode = ToolRunner.run(new
        WCDriver(), args);
        System.out.println("Exit code: "+exitCode);
    }
}
```



WCMapper.

```
public class WCMapper
    public void map(LongWritable key, Text
    value, OutputCollector<Text, IntWritable>
    output, Reporter reporter) throws
    IOException{

    String line = value.toString();

    for (String word: line.split(" ")){
        if (word.length()>0){
            output.collect(new Text(word),
            new IntWritable(1));
        }
    }
};
```

WCReducer.java

```
public class WCReducer{
    public void reduce{
        int count=0;
        while (values.hasNext()){
            count += values.next().get();
        }
        output.collect(key, new IntWritable(count));
    }
}
```

Output

```
are    1
Brother    1
family    1
hi    1
```

Lab 6:

Question: From the following link extract the weather data https://github.com/tomwhite/hadoop-book/tree/master/input/ncdc/all Create a Map Reduce program to

a) find average temperature for each year from NCDC data set.

Code with Output:

AvgMapper.java

```
package avgtemp;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class AvgMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

   public static final int MISSING = 9999;

   @Override
   public void map(LongWritable key, Text value, Context context)
         throws IOException, InterruptedException {

      String line = value.toString();
      String year = line.substring(15, 19);
      int temperature;

      if (line.charAt(87) == '+') {
         temperature = Integer.parseInt(line.substring(88, 92));
      } else {
         temperature = Integer.parseInt(line.substring(87, 92));
      }

      String quality = line.substring(92, 93);

      if (temperature != MISSING && quality.matches("[01459]")) {
         context.write(new Text(year), new IntWritable(temperature));
      }
   }
}
```

AvgDriver.java

```
package avgtemp;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class AvgDriver {

   public static void main(String[] args) throws Exception {
      if (args.length != 2) {
         System.err.println("Please enter input and output paths");
         System.exit(-1);
      }

      Job job = Job.getInstance();
      job.setJarByClass(AvgDriver.class);
      job.setJobName("Average Temperature");

      FileInputFormat.addInputPath(job, new Path(args[0]));
      FileOutputFormat.setOutputPath(job, new Path(args[1]));

      job.setMapperClass(AvgMapper.class);
      job.setReducerClass(AvgReducer.class);

      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(IntWritable.class);

      System.exit(job.waitForCompletion(true) ? 0 : 1);
   }
}
```

AvgReducer.java

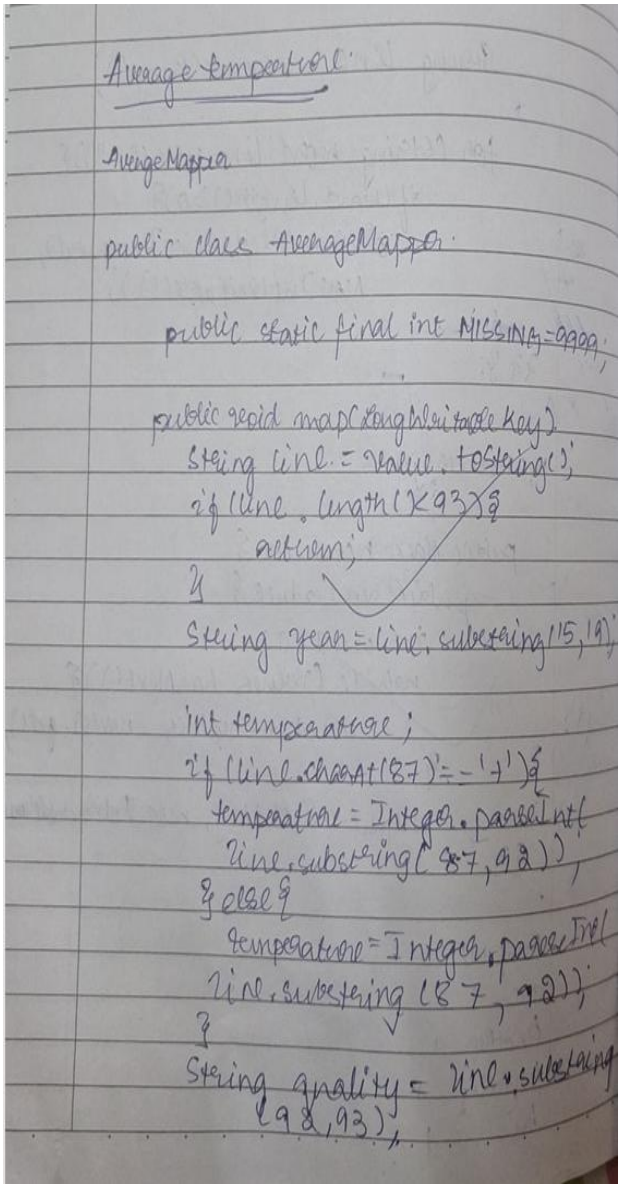
```
package avgtemp;



import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class AvgReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

   @Override
```

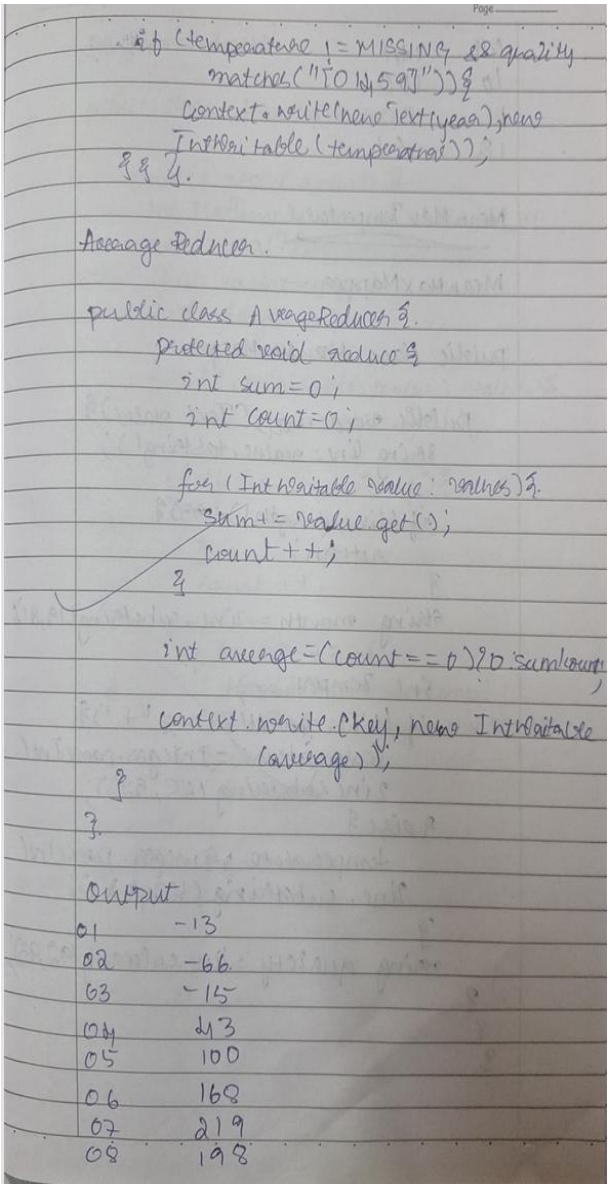
```
    public void reduce(Text key, Iterable<IntWritable> values, Context context)
            throws IOException, InterruptedException {

        int sumTemp = 0;
        int count = 0;

        for (IntWritable value : values) {
            sumTemp += value.get();
            count++;
        }

        int average = (count == 0) ? 0 : sumTemp / count;
        context.write(key, new IntWritable(average));
    }
}
```

```
praneeta@LAPTOP-M1509RLF:~$ hdfs dfs -put /home/praneeta/temperature.txt /lhs/temp.txt
praneeta@LAPTOP-M1509RLF:~$ hadoop jar /home/praneeta/avgtemp.jar avgtemp.AvgDriver /temp.txt /lhs/avgout
2025-05-26 16:31:44,734 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2025-05-26 16:31:44,804 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2025-05-26 16:31:44,805 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2025-05-26 16:31:44,897 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application
with ToolRunner to remedy this.
2025-05-26 16:31:44,966 INFO mapreduce.JobSubmitter: Cleaning up the staging area file:/tmp/hadoop/mapred/staging/praneeta601661354/.staging/job_local601661354_0001
Exception in thread "main" org.apache.hadoop.mapreduce.lib.input.InvalidInputException: Input path does not exist: hdfs://localhost:9000/temp.txt
        at org.apache.hadoop.mapreduce.lib.input.FileInputFormat.singleThreadedListStatus(FileInputFormat.java:340)
        at org.apache.hadoop.mapreduce.lib.input.FileInputFormat.listStatus(FileInputFormat.java:279)
        at org.apache.hadoop.mapreduce.lib.input.FileInputFormat.getSplits(FileInputFormat.java:404)
        at org.apache.hadoop.mapreduce.JobSubmitter.writeNewSplits(JobSubmitter.java:310)
        at org.apache.hadoop.mapreduce.JobSubmitter.writeSplits(JobSubmitter.java:327)
        at org.apache.hadoop.mapreduce.JobSubmitter.submitJobInternal(JobSubmitter.java:200)
        at org.apache.hadoop.mapreduce.Job$11.run(Job.java:1678)
        at org.apache.hadoop.mapreduce.Job$11.run(Job.java:1675)
        at java.base/java.security.AccessController.doPrivileged(Native Method)
```

```
praneeta@LAPTOP-M1509RLF:~$ hdfs dfs -ls /lhs/avgout
Found 2 items
-rw-r--r--   3 praneeta supergroup          0 2025-05-26 16:32 /lhs/avgout/_SUCCESS
-rw-r--r--   3 praneeta supergroup          8 2025-05-26 16:32 /lhs/avgout/part-r-00000
praneeta@LAPTOP-M1509RLF:~$ hdfs dfs -cat /lhs/avgout/part-r-00000
1901    46
```

ScreenShot





b) find the mean max temperature for every month

Code with output

MMDriver.java

```
package meanmaxtemp;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```

```
public class MMDriver {

   public static void main(String[] args) throws Exception {
      if (args.length != 2) {
         System.err.println("Please enter the input and output parameters");
         System.exit(-1);
      }

      Job job = Job.getInstance();
      job.setJarByClass(MMDriver.class);
      job.setJobName("Mean of Maximum Temperature (3-day blocks)");

      FileInputFormat.addInputPath(job, new Path(args[0]));
      FileOutputFormat.setOutputPath(job, new Path(args[1]));

      job.setMapperClass(MMMapper.class);
      job.setReducerClass(MMReducer.class);

      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(IntWritable.class);

      System.exit(job.waitForCompletion(true) ? 0 : 1);
   }
}
```

MMMapper.java

```
package meanmaxtemp;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class MMMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

   public static final int MISSING = 9999;

   @Override
   public void map(LongWritable key, Text value, Context context)
         throws IOException, InterruptedException {

      String line = value.toString();
      String month = line.substring(19, 21);
      int temperature;
```

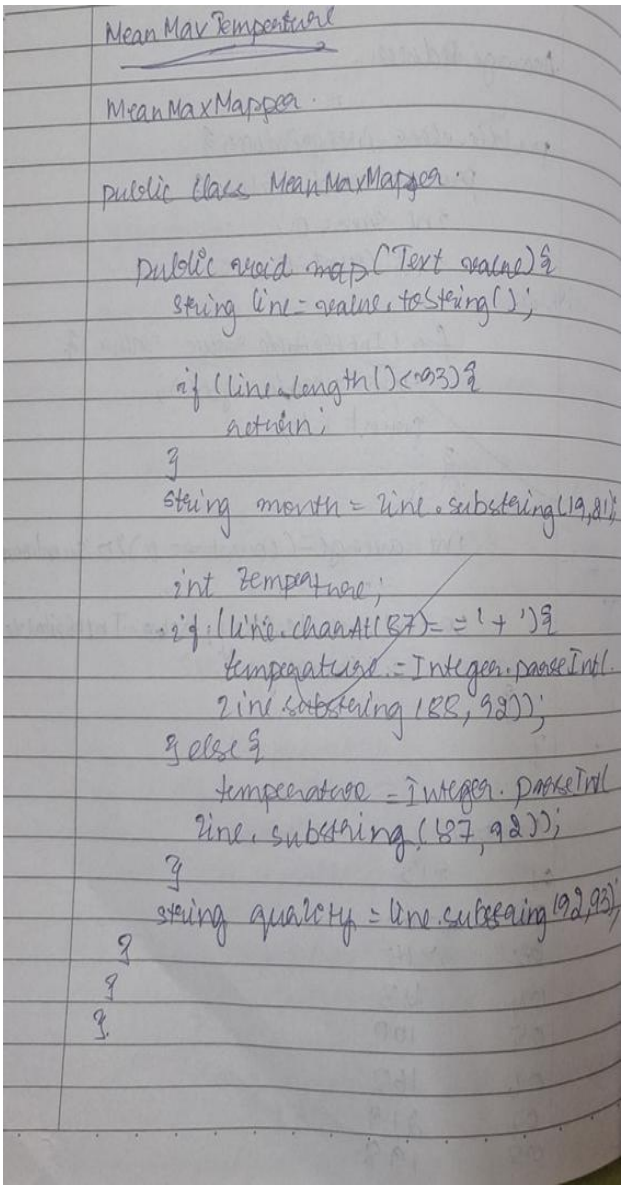
```
        if (line.charAt(87) == '+') {
            temperature = Integer.parseInt(line.substring(88, 92));
        } else {
            temperature = Integer.parseInt(line.substring(87, 92));
        }

        String quality = line.substring(92, 93);

        if (temperature != MISSING && quality.matches("[01459]")) {
            context.write(new Text(month), new IntWritable(temperature));
        }
    }
}
```

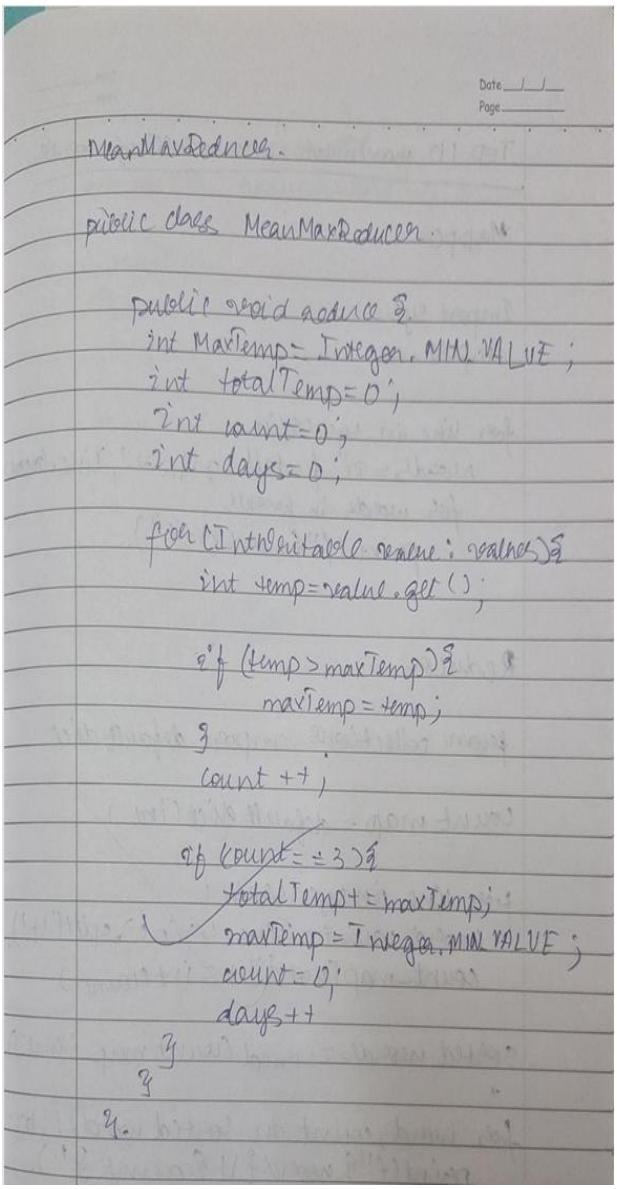
MMReducer.java

```
package meanmaxtemp;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class MMReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

    @Override
    public void reduce(Text key, Iterable<IntWritable> values, Context context)
            throws IOException, InterruptedException {

        int maxTemp = Integer.MIN_VALUE;
        int totalBlockMax = 0;
        int count = 0;
        int blockCount = 0;

        for (IntWritable value : values) {
            int temp = value.get();
            if (temp > maxTemp) {
                maxTemp = temp;
            }
            count++;

            if (count == 3) {
                totalBlockMax += maxTemp;
                maxTemp = Integer.MIN_VALUE;
                count = 0;
                blockCount++;
            }
        }
```

```
        // Avoid division by zero
        if (blockCount > 0) {
            int meanOfMax = totalBlockMax / blockCount;
            context.write(key, new IntWritable(meanOfMax));
        }
    }
}
```

Screenshot





Lab 7:

Question: 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.

Code with Output:

Mapper.py

```
  GNU nano 6.2                                                  maxocc_mapper.py
#!/usr/bin/env python3
import sys
import re

for line in sys.stdin:
    words = re.findall(r'\w+', line.lower())  #
    for word in words:
        print(f"{word}\t1")
```

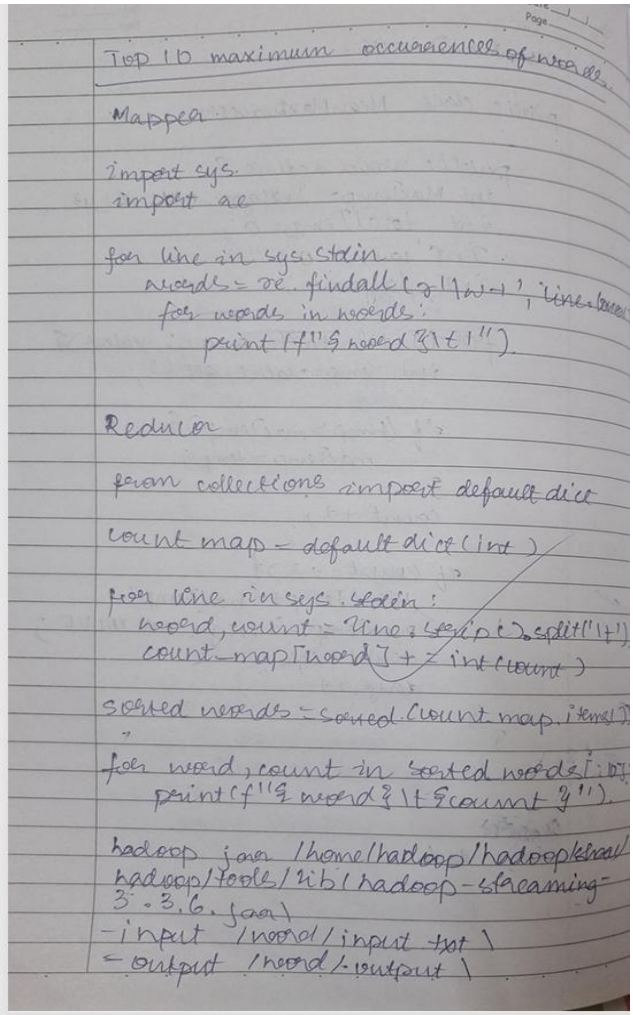
Reducer.py

```
  GNU nano 6.2                                                  naxocc_reducer.py
#!/usr/bin/env python3
import sys

from collections import defaultdict

count_map = defaultdict(int)

# Read input from mapper (word\t1)
for line in sys.stdin:
    word, count = line.strip().split('\t')
    count_map[word] += int(count)

# Sort by count desc, then word asc
sorted_words = sorted(count_map.items(), key=lambda x: (-x[1], x[0]))

# Output top 10
for word, count in sorted_words[:10]:
    print(f"{word}\t{count}")
```

```
praneeta@LAPTOP-M1509RLF:~$ nano maxocc_mapper.py
praneeta@LAPTOP-M1509RLF:~$ nano naxocc_reducer.py
praneeta@LAPTOP-M1509RLF:~$ nano maxocc_mapper.py
praneeta@LAPTOP-M1509RLF:~$ hdfs dfs -mkdir /lhs
praneeta@LAPTOP-M1509RLF:~$ hdfs dfs -put /home/praneeta/occ.txt /lhs/input.txt
praneeta@LAPTOP-M1509RLF:~$ chmod +x /home/praneeta/maxocc_mapper.py /home/praneeta/maxocc_reducer.py
chmod: cannot access '/home/praneeta/maxocc_reducer.py': No such file or directory
praneeta@LAPTOP-M1509RLF:~$ chmod +x /home/praneeta/maxocc_mapper.py /home/praneeta/naxocc_reducer.py
praneeta@LAPTOP-M1509RLF:~$ hadoop jar /home/praneeta/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar   -input /lhs/input.txt   -output /lhs/output   -mapper /
home/praneeta/maxocc_mapper.py   -reducer /home/praneeta/naxocc_reducer.py   -file /home/praneeta/manocc_mapper.py   -file /home/praneeta/naxocc_reducer.py
2025-05-26 16:16:57,167 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
File: file:/home/praneeta/manocc_mapper.py does not exist.
Try -help for more information
Streaming Command Failed!
praneeta@LAPTOP-M1509RLF:~$ hadoop jar /home/praneeta/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar   -input /lhs/input.txt   -output /lhs/output   -mapper /
home/praneeta/maxocc_mapper.py   -reducer /home/praneeta/naxocc_reducer.py   -file /home/praneeta/maxocc_mapper.py   -file /home/praneeta/naxocc_reducer.py
2025-05-26 16:17:11,838 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [/home/praneeta/maxocc_mapper.py, /home/praneeta/naxocc_reducer.py] [] /tmp/streamjob5831022467006698228.jar tmpDir=null
2025-05-26 16:17:12,965 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2025-05-26 16:17:13,061 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2025-05-26 16:17:13,062 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2025-05-26 16:17:13,077 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2025-05-26 16:17:13,338 INFO mapred.FileInputFormat: Total input files to process : 1
2025-05-26 16:17:13,401 INFO mapreduce.JobSubmitter: number of splits:1
2025-05-26 16:17:13,595 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local468980523_0001
2025-05-26 16:17:13,595 INFO mapreduce.JobSubmitter: Executing with tokens: []
2025-05-26 16:17:13,818 INFO mapred.LocalDistributedCacheManager: Localized file:/home/praneeta/maxocc_mapper.py as file:/tmp/hadoop-praneeta/mapred/local/job_local4689
80523_0001_55328529-5bc5-4f90-86ea-7e0a1b71ddca/maxocc_mapper.py
2025-05-26 16:17:13,845 INFO mapred.LocalDistributedCacheManager: Localized file:/home/praneeta/naxocc_reducer.py as file:/tmp/hadoop-praneeta/mapred/local/job_local468
```

```
2025-05-26 16:17:16,849 INFO streaming.StreamJob: Output directory: /lhs/output
praneeta@LAPTOP-M1509RLF:~$ hdfs dfs -cat /lhs/output/part-00000
the     10
ipsum   8
lorem   8
a       7
it      7
of      7
and     6
is      4
content 3
has     3
praneeta@LAPTOP-M1509RLF:~$
```

Screenshot



Lab 8:

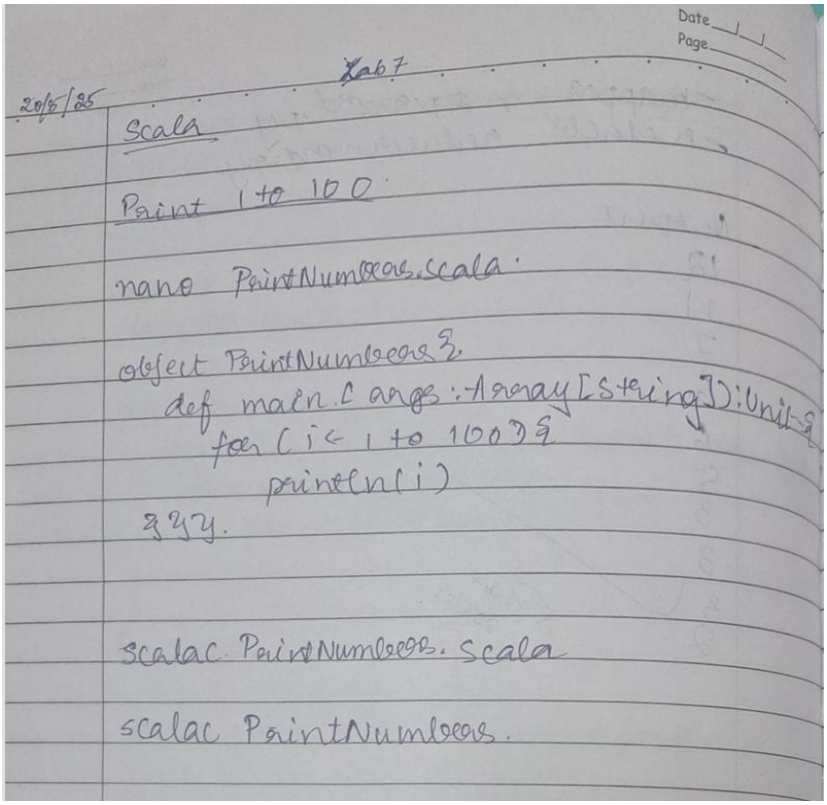
Question: Write a Scala program to print numbers from 1 to 100 using for loop.

Code with Output:

```
  GNU nano 6.2                                                    printnumbers.scala
object printnumbers{
        def main(args:Array[String]):Unit={
                for(i<-1 to 100){
                        print(i+" ")
                }
        }
}
```

```
praneeta@LAPTOP-M1509RLF:~$ nano printnumbers.scala
praneeta@LAPTOP-M1509RLF:~$ scala printnumbers.scala
warning: 1 deprecation (since 2.13.0); re-run with -deprecation for details
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59
60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 praneeta@LAPTOP-M1509RLF:~$
```

Screenshot

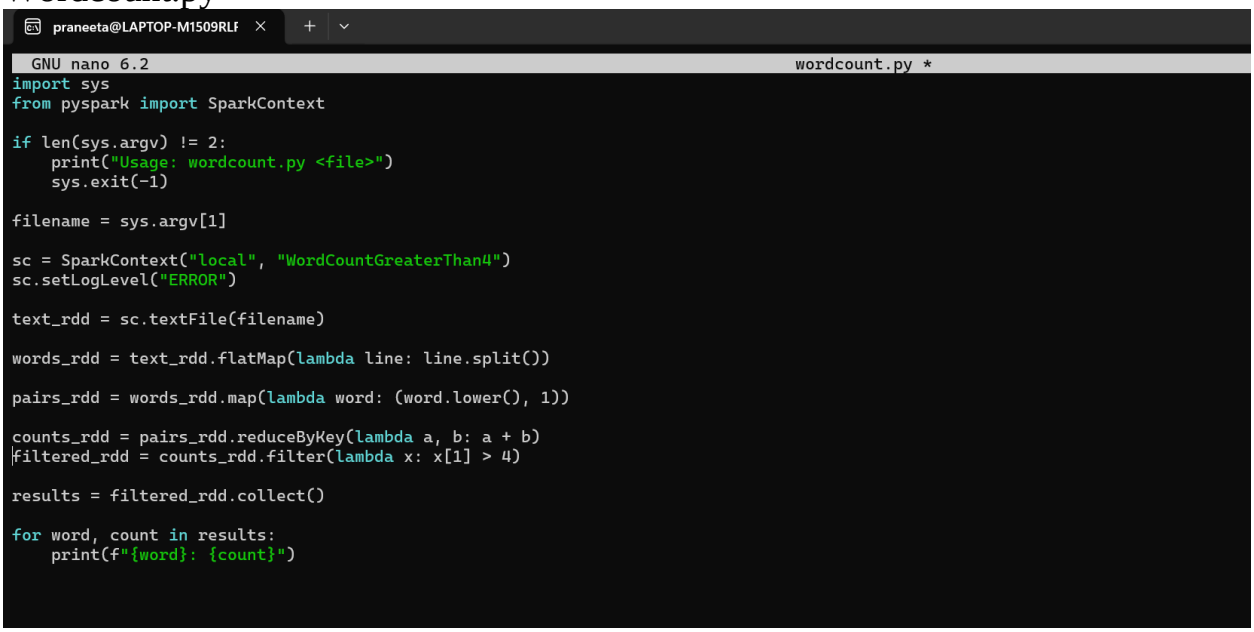


Lab 9:

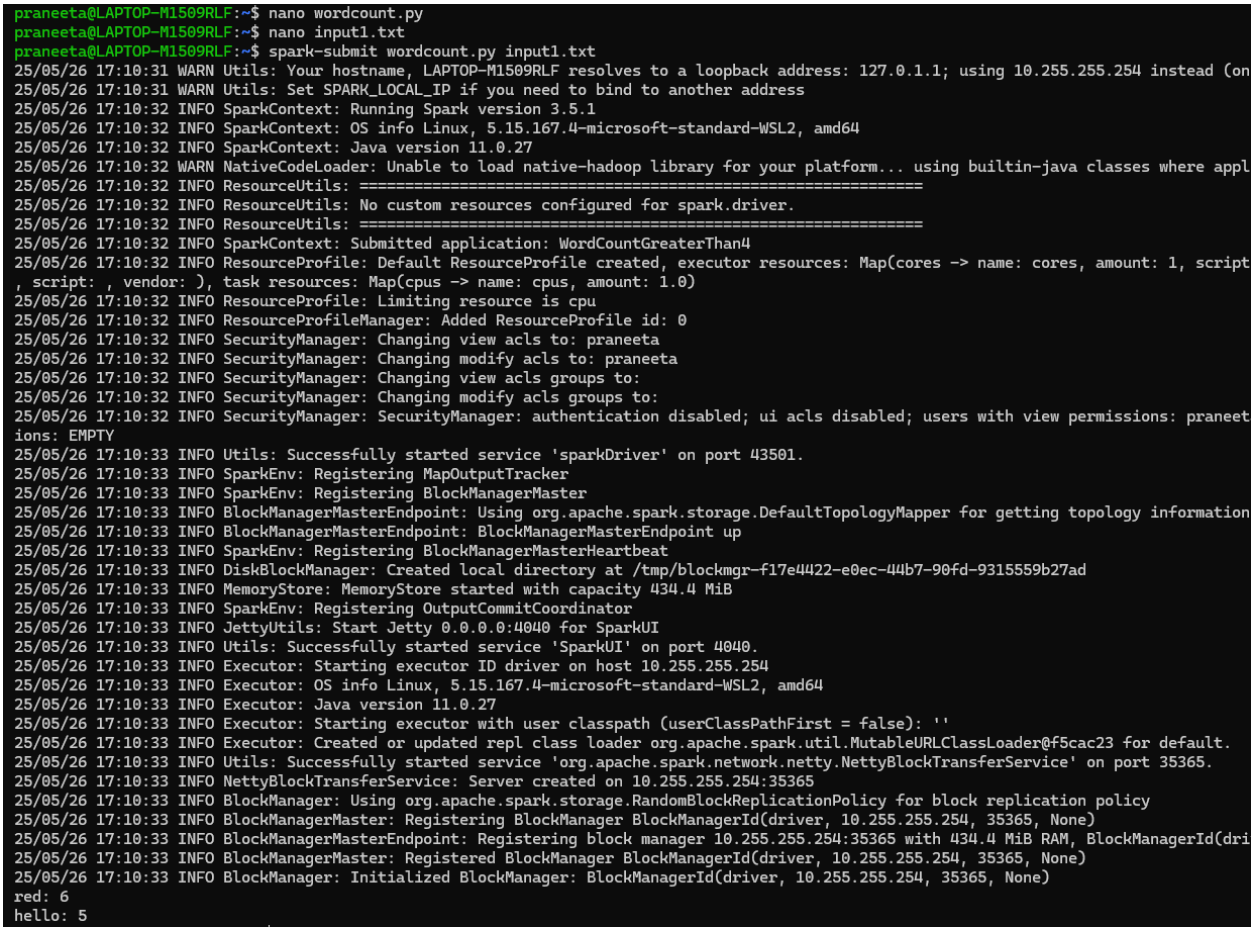
Question: Using RDD and FlatMap count how many times each word appears in a file and write out a list of words whose count is strictly greater than 4 using Spark.

Code with Output:

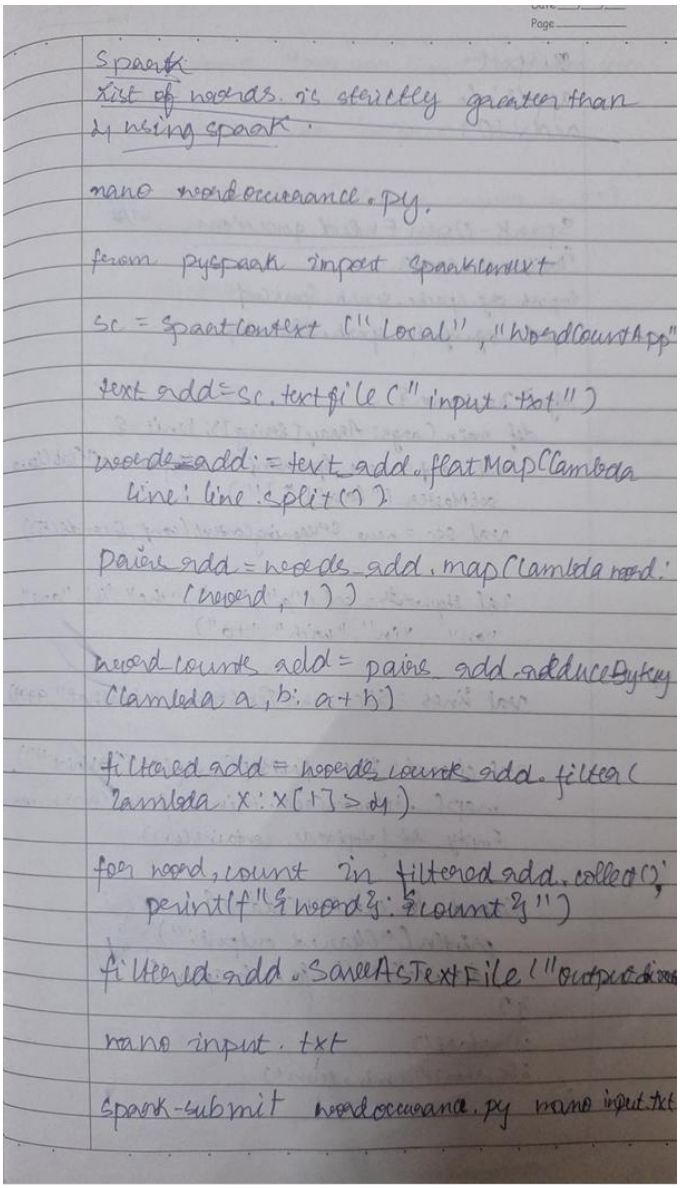
Wordcount.py



Input.txt


praneeta@LAPTOP-M1509RLF:~$ nano wordcount.py
praneeta@LAPTOP-M1509RLF:~$ nano input1.txt
praneeta@LAPTOP-M1509RLF:~$ spark-submit wordcount.py input1.txt
25/05/26 17:10:31 WARN Utils: Your hostname, LAPTOP-M1509RLF resolves to a loopback address: 127.0.1.1; using 10.255.255.254 instead (on
25/05/26 17:10:31 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
25/05/26 17:10:32 INFO SparkContext: Running Spark version 3.5.1
25/05/26 17:10:32 INFO SparkContext: OS info Linux, 5.15.167.4-microsoft-standard-WSL2, amd64
25/05/26 17:10:32 INFO SparkContext: Java version 11.0.27
25/05/26 17:10:32 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where appl
25/05/26 17:10:32 INFO ResourceUtils: ==
25/05/26 17:10:32 INFO ResourceUtils: No custom resources configured for spark.driver.
25/05/26 17:10:32 INFO ResourceUtils: ==
25/05/26 17:10:32 INFO SparkContext: Submitted application: WordCountGreaterThan4
25/05/26 17:10:32 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, amount: 1, script
, script: , vendor:), task resources: Map(cpus -> name: cpus, amount: 1.0)
25/05/26 17:10:32 INFO ResourceProfile: Limiting resource is cpu
25/05/26 17:10:32 INFO ResourceProfileManager: Added ResourceProfile id: 0
25/05/26 17:10:32 INFO SecurityManager: Changing view acls to: praneeta
25/05/26 17:10:32 INFO SecurityManager: Changing modify acls to: praneeta
25/05/26 17:10:32 INFO SecurityManager: Changing view acls groups to:
25/05/26 17:10:32 INFO SecurityManager: Changing modify acls groups to:
25/05/26 17:10:32 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: praneet
ions: EMPTY
25/05/26 17:10:33 INFO Utils: Successfully started service 'sparkDriver' on port 43501.
25/05/26 17:10:33 INFO SparkEnv: Registering MapOutputTracker
25/05/26 17:10:33 INFO SparkEnv: Registering BlockManagerMaster
25/05/26 17:10:33 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
25/05/26 17:10:33 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
25/05/26 17:10:33 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
25/05/26 17:10:33 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-f17e4422-e0ec-44b7-90fd-9315559b27ad
25/05/26 17:10:33 INFO MemoryStore: MemoryStore started with capacity 434.4 MiB
25/05/26 17:10:33 INFO SparkEnv: Registering OutputCommitCoordinator
25/05/26 17:10:33 INFO JettyUtils: Start Jetty 0.0.0.0:4040 for SparkUI
25/05/26 17:10:33 INFO Utils: Successfully started service 'SparkUI' on port 4040.
25/05/26 17:10:33 INFO Executor: Starting executor ID driver on host 10.255.255.254
25/05/26 17:10:33 INFO Executor: OS info Linux, 5.15.167.4-microsoft-standard-WSL2, amd64
25/05/26 17:10:33 INFO Executor: Java version 11.0.27
25/05/26 17:10:33 INFO Executor: Starting executor with user classpath (userClassPathFirst = false): ''
25/05/26 17:10:33 INFO Executor: Created or updated repl class loader org.apache.spark.util.MutableURLClassLoader@f5cac23 for default.
25/05/26 17:10:33 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 35365.
25/05/26 17:10:33 INFO NettyBlockTransferService: Server created on 10.255.255.254:35365
25/05/26 17:10:33 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy
25/05/26 17:10:33 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, 10.255.255.254, 35365, None)
25/05/26 17:10:33 INFO BlockManagerMasterEndpoint: Registering block manager 10.255.255.254:35365 with 434.4 MiB RAM, BlockManagerId(dri
25/05/26 17:10:33 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, 10.255.255.254, 35365, None)
25/05/26 17:10:33 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, 10.255.255.254, 35365, None)
red: 6
hello: 5

Screenshot


Spark
List of words. is strictly greater than
4 using spark.
nano wordoccurance.py.
from pyspark import SparkContext
sc = SparkContext ("local", "WordCountApp"
text_rdd=sc.textFile("input.txt")
words_rdd = text_rdd.flatMap(lambda
line: line.split())
pairs_rdd = words_rdd.map(lambda word:
(word, 1))
word_counts_rdd = pairs_rdd.reduceByKey
(lambda a, b: a+b)
filtered_rdd = words_count_rdd.filter(
lambda x: x[1] > 4).
for word, count in filtered_rdd.collect():
print(f"{word}: {count}")
filtered_rdd.saveAsTextFile("output
nano input.txt
spark-submit wordoccurance.py nano input.txt

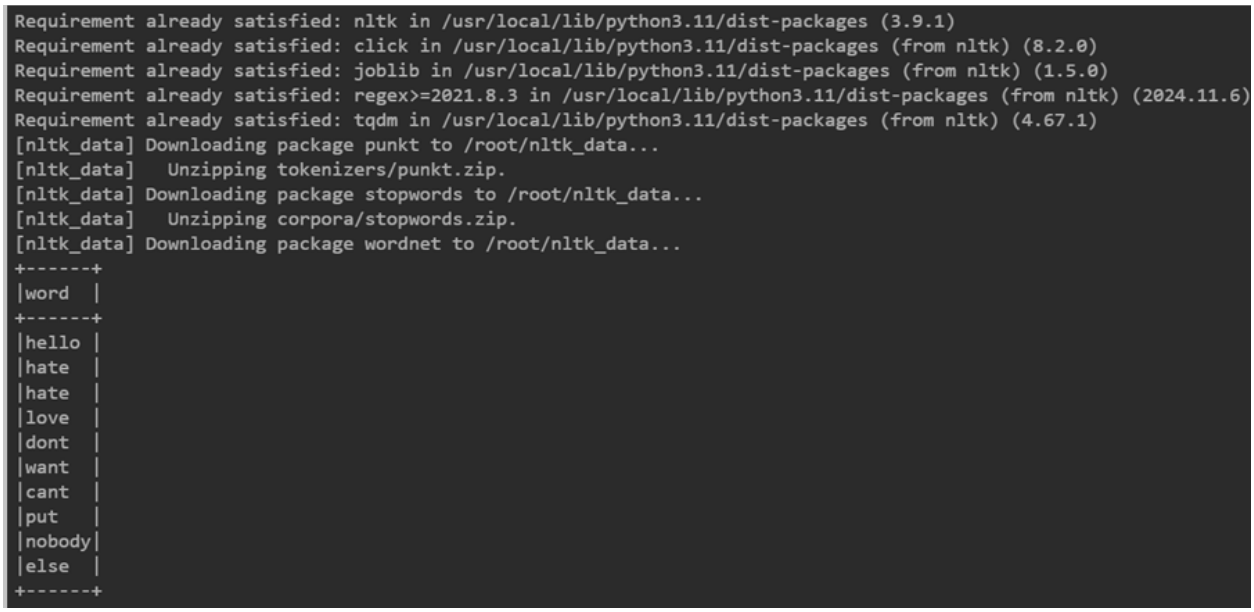
Lab10:

Question: Write a simple streaming program in Spark to receive text data streams on a particular port, perform basic text cleaning (like white space removal, stop words removal, lemmatization, etc.), and print the cleaned text on the screen. (Open Ended Question).

Code with output

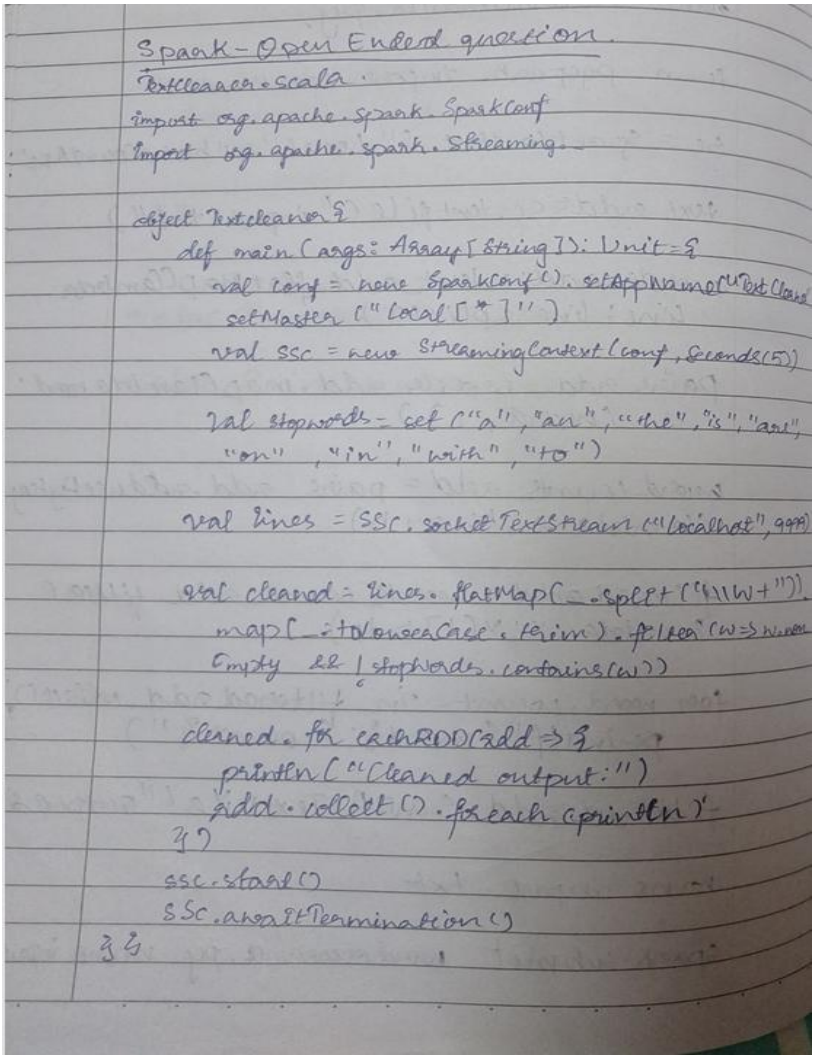
```
# Install NLTK and download required data (run once)
!pip install nltk

import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, lower, regexp_replace, split, explode, udf
from pyspark.sql.types import ArrayType, StringType
from pyspark.ml.feature import StopWordsRemover
from nltk.stem import WordNetLemmatizer

# Initialize SparkSession
spark = SparkSession.builder.appName("TextProcessing").getOrCreate()

# Define your input lines
lines = [
    "Hello, I hate you.",
    "I hate that I love you.",
    "Don't want to, but I can't put",
    "nobody else above you."
]
# Create DataFrame from lines
df = spark.createDataFrame(lines, "string").toDF("value")

# Step 1: Lowercase and remove punctuation
df_clean = df.select(regexp_replace(lower(col("value")), "[^a-zA-Z\\s]", "").alias("cleaned"))

# Step 2: Tokenize the cleaned text
df_tokens = df_clean.select(split(col("cleaned"), "\\s+").alias("tokens"))

# Step 3: Remove stop words
remover = StopWordsRemover(inputCol="tokens", outputCol="filtered")
df_filtered = remover.transform(df_tokens)
```

```
# Step 4: Lemmatization using NLTK WordNetLemmatizer with UDF
lemmatizer = WordNetLemmatizer()

def lemmatize_words(words):
    return [lemmatizer.lemmatize(word) for word in words]

lemmatize_udf = udf(lemmatize_words, ArrayType(StringType()))

df_lemmatized = df_filtered.withColumn("lemmatized", lemmatize_udf(col("filtered")))

# Step 5: Explode the lemmatized words and show results
df_lemmatized.select(explode(col("lemmatized")).alias("word")).show(truncate=False)
```



```
Requirement already satisfied: nltk in /usr/local/lib/python3.11/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (from nltk) (8.2.0)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from nltk) (1.5.0)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.11/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from nltk) (4.67.1)
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
+------+
|word  |
+------+
|hello |
|hate  |
|hate  |
|love  |
|dont  |
|want  |
|cant  |
|put   |
|nobody|
|else  |
+------+
```

Screenshot


Spark - Open Ended question.
TextCleaner.scala.
import org.apache.spark.SparkConf
import org.apache.spark.streaming._
object TextCleaner {
def main(args: Array[String]): Unit = {
val conf = new SparkConf().setAppName("TextCleaner
setMaster("local[*]")
val ssc = new StreamingContext(conf, Seconds(5))
val stopwords = set("a", "an", "the", "is", "are",
"on", "in", "with", "to")
val lines = ssc.socketTextStream("localhost", 9999)
val cleaned = lines.flatMap(_.split("\\W+"))
map(_.toLowerCase.trim).filter(w => w.nonEm
Empty && !stopwords.contains(w))
cleaned.foreachRDD(rdd => {
println("Cleaned output:")
rdd.collect().foreach(println)
})
ssc.start()
ssc.awaitTermination()
}}