VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

Big Data Analytics (23CS6PCBDA)

Submitted by

Nehal A K (1BM22CS176)

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 Nehal A K (1BM22CS176), 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.

Prof. Vikranth Bm
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

Index Sheet

Course Outcome

CO1	Apply the concept of NoSQL, Hadoop or Spark for a given task
CO2	Analyze 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.

Github Link: https://github.com/nehxl31/BDA-LAB

LAB-1

MongoDB- CRUD Operations Demonstration

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". o Display only StudName and Grade without _id.
   - Find documents where Grade is 'VII'.
   - Find documents where Hobbies is either 'Chess' or 'Skating'.
   - Find documents where StudName starts with 'M'. o Find documents where StudName contains 'e'.
   - Count number of documents in Students collection. o Sort documents by StudName in descending order.

III. Import/Export

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

IV. Save/Update/Delete Field Operations 11. Use save() to insert or update a document.

12. Add a new field to an existing document.
13. Remove a field from an existing document.

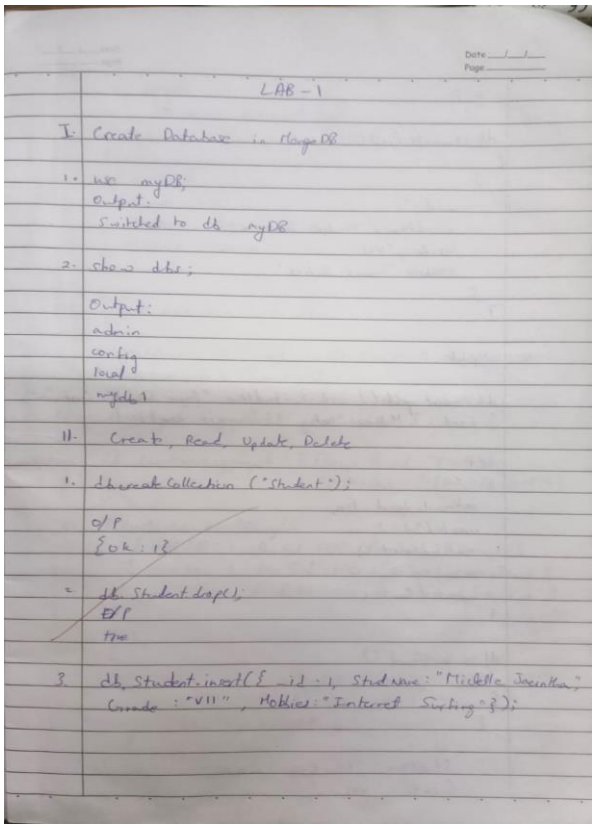
V. More FIND Operations

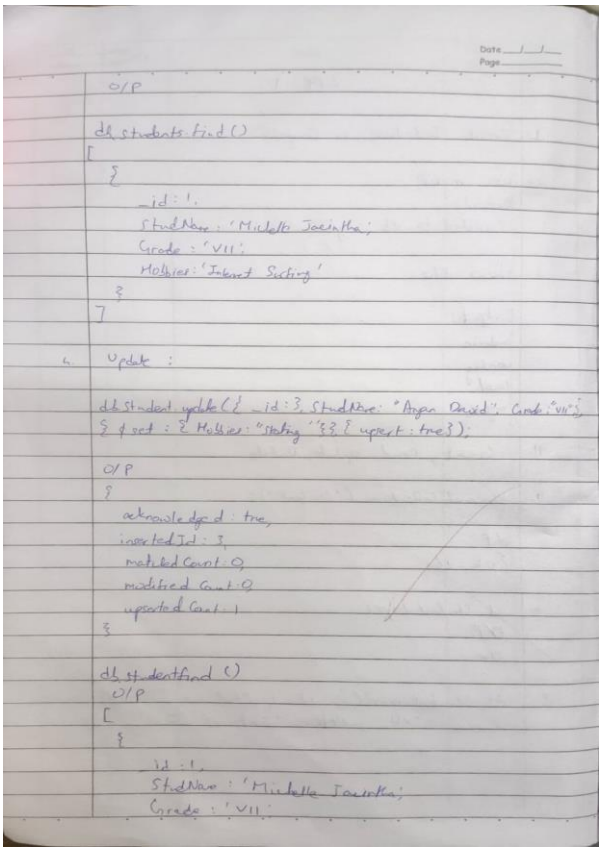
14. Display only StudName and Grade for _id:1, suppress _id.
15. Find documents where Grade is not 'VII'.
16. Find documents where StudName ends with 's'.

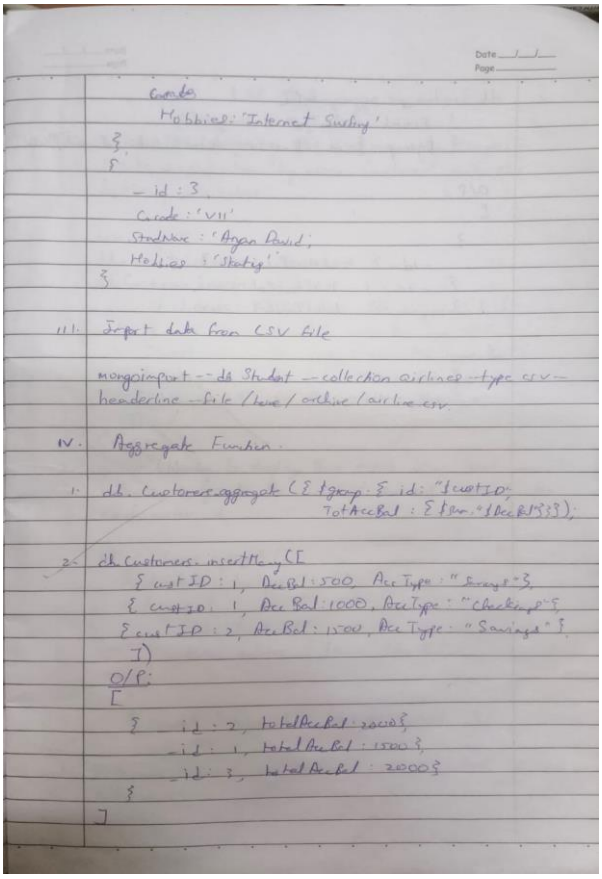
VI. NULL and Count Queries

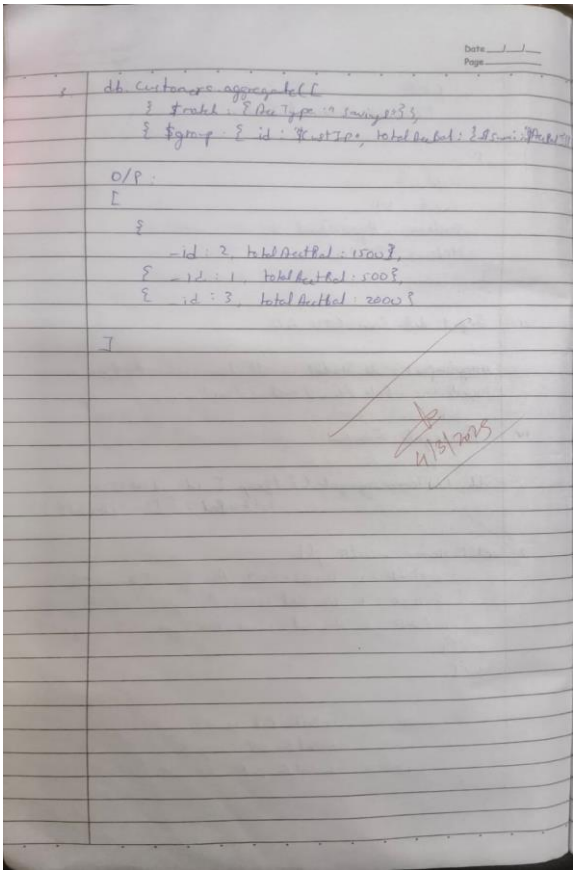
17. Set a field value to null.
18. Count all documents in Students.
19. Count documents with Grade: "VII".
20. Retrieve first 3 documents where Grade: "VII".
21. Sort documents in ascending order by StudName.

Observation Book:









Code & Output:

```
mydb> db.Students.insert({id:1,name:"Sam",Grade:"VII",Hobbies:["Cricket","Football"]});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68335bd5d735942126c59f35') }
}
mydb> db.Students.insertOne({id:2,name:"John",Grade:"VI",Hobbies:"Drawing"});
{
  acknowledged: true,
  insertedId: ObjectId('68335c08d735942126c59f36')
}
mydb> db.find()
```

```
mydb> db.Students.find()
\[
  {
    _id: ObjectId('68335bd5d735942126c59f35'),
    id: 1,
    name: 'Sam',
    Grade: 'VII',
    Hobbies: [ 'Cricket', 'Football' ]
  },
  {
    _id: ObjectId('68335c08d735942126c59f36'),
    id: 2,
    name: 'John',
    Grade: 'VI',
    Hobbies: 'Drawing'
  }
]
```

```
mydb> db.Students.insertMany([{_id: 3, name: "Sam", Grade: "VII", Hobbies: ["Cricket", "Football"]}, {_id: 4, name: "Alice", Grade: "VIII", Hobb
ies: ["Reading", "Swimming"] },{ _id: 5, name: "Bob", Grade: "VII", Hobbies: ["Chess", "Basketball"]}]);
{ acknowledged: true, insertedIds: { '0': 3, '1': 4, '2': 5 } }
mydb> db.Students.find()
[
  {
    _id: ObjectId('68335bd5d735942126c59f35'),
    name: 'Sam',
    Grade: 'VII',
    Hobbies: [ 'Cricket', 'Football' ]
  },
  {
    _id: ObjectId('68335c08d735942126c59f36'),
    name: 'John',
    Grade: 'VI',
    Hobbies: 'Drawing'
  },
  {
    _id: 3,
    name: 'Sam',
    Grade: 'VII',
    Hobbies: [ 'Cricket', 'Football' ]
  },
  {
    _id: 4,
    name: 'Alice',
    Grade: 'VIII',
    Hobbies: [ 'Reading', 'Swimming' ]
  },
  {
    _id: 5,
    name: 'Bob',
    Grade: 'VII',
    Hobbies: [ 'Chess', 'Basketball' ]
  }
```

```
mydb> db.Students.find({Grade:"VII",Hobbies:"Chess"})
[
  {
    _id: 5,
    name: 'Bob',
    Grade: 'VII',
    Hobbies: [ 'Chess', 'Basketball' ]
  }
]
```

```
mydb> db.Students.update({Grade:"VII",name:"Bob"},{$set:{Grade:"IX"}})
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
mydb> db.Students.find().sort({name:1})
[
  {
    _id: 4,
    name: 'Alice',
    Grade: 'VIII',
    Hobbies: [ 'Reading', 'Swimming' ]
  },
  {
    _id: 5,
    name: 'Bob',
    Grade: 'IX',
    Hobbies: [ 'Chess', 'Basketball' ]
  },
  {
    _id: ObjectId('68335c08d735942126c59f36'),
    name: 'John',
    Grade: 'VI',
    Hobbies: 'Drawing'
  },
  {
    _id: ObjectId('68335bd5d735942126c59f35'),
    name: 'Sam',
    Grade: 'VII',
    Hobbies: [ 'Cricket', 'Football' ]
  },
  {
    _id: 3,
    name: 'Sam',
    Grade: 'VII',
    Hobbies: [ 'Cricket', 'Football' ]
  }
]
```

```
mydb> db.Accounts.aggregate([{$group:{_id:"$cust_id",min_bal:{$min:"$acc_bal"},max_bal:{$max:"$acc_bal"}}}])
[
  { _id: 101, min_bal: 5000, max_bal: 5000 },
  { _id: 102, min_bal: 12000, max_bal: 12000 },
  { _id: 105, min_bal: 3000, max_bal: 3000 },
  { _id: 104, min_bal: 20000, max_bal: 20000 },
  { _id: 103, min_bal: 7500, max_bal: 7500 }
]
```

Section 1: Customer Collection Tasks

1. Create a collection named Customers with the following attributes: Cust_id, Acc_Bal, Acc_Type
2. Insert at least 5 records into the collection.
3. Write a query to display records where the total account balance is greater than 1200 and the account type is 'Z', grouped by each customer_id.
4. Determine the Minimum and Maximum account balance for each customer_id.

Section 2: E-Commerce Platform Schema & Queries

You are developing an e-commerce platform. Design a MongoDB schema to handle:

- Product information
- User carts
- Orders

Then, answer the following queries:

Product Queries

5. Retrieve all products.
6. Retrieve products in a specific category (e.g., Electronics).
7. Retrieve products with quantity greater than 0.
8. Retrieve products sorted by price in ascending order.
9. Retrieve products with price ≤ $100.

Cart and Order Queries

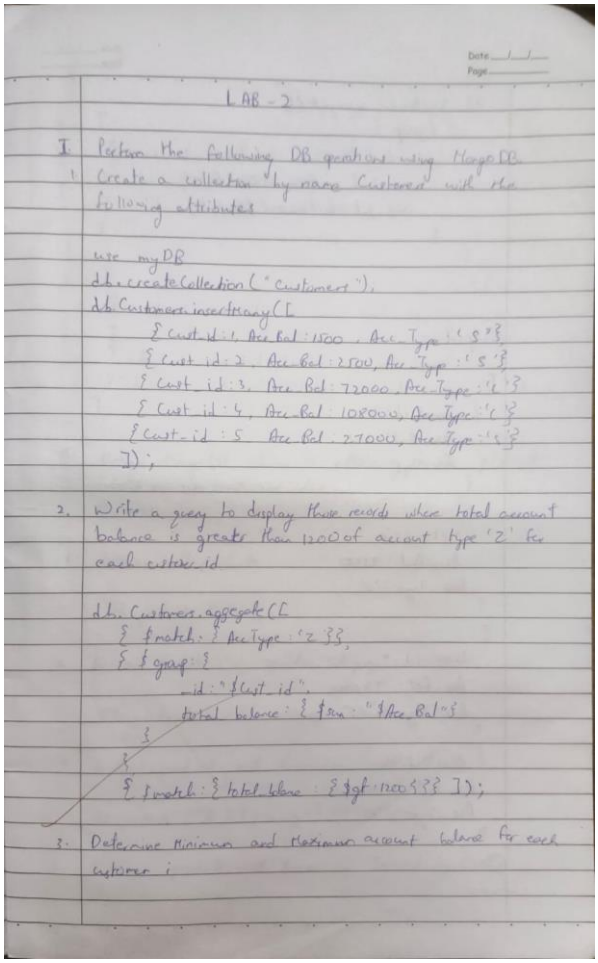
10. Retrieve products added to a user's cart (User ID: "789ghi...").
11. Retrieve orders placed by a user (User ID: "123abc...").
12. Retrieve total price of orders placed by a user (User ID: "123abc...").

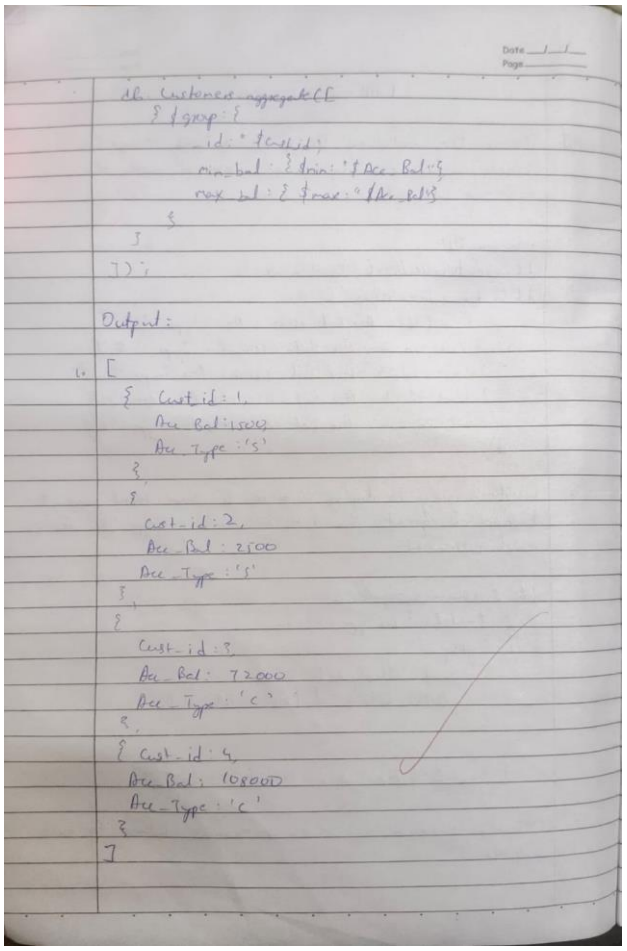
Section 3: Additional Aggregation Queries (Assignment-3 Design)

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

20. Find the average total price of orders.

Screenshot:





Additional queries

1. Total Number of products in each category

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

2. Total price of Products in Each Category

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

3. Find average of products

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

4. Find product with quantity less than 0

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

5. Sort Products by Price in Descending Order

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

6. Calculate Total price of Orders

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

7. Find user with Highest Total Price of Order

```
db.orders.aggregate([
  {
    $group: {
      _id: "$user_id",
      total_order: { $sum: "$total_price" }
    }
  },
  { $sort: { total_order_price: -1 } }
]);
```

8. Find Average Total Price of Order

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

2.
```
[
  { _id: 1, total_bal: 1500 }
  { _id: 2, total_bal: 2500 }
  { _id: 5, total_bal: 27000 }
]
```

3.
```
[
  { _id: 1, max_bal: 1000, min_bal: 1500 }
  { _id: 2, max_bal: 2500, min_bal: 2500 }
  { _id: 3, max_bal: 71000, min_bal: 71000 }
  { _id: 4, max_bal: 108000, min_bal: 108000 }
]
```

II. Design MongoDB schema to efficiently handle product information, user carts, and orders

Queries.

1. Retrieve all products

```
db.products.find();
```

2. Retrieve Products in a specific Category

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

3. Retrieve products with Quantity Greater than 0.

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

4. Retrieve products with Price in Ascending Order

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

5. Retrieve Products Added in a Specific Category (Electronics)

```
db.products.find({ price: { $lte: 1000 } });
```

6. Retrieve Orders Placed by a User (User with ID "123abc...")

```
db.users.aggregate([
  { $match: { _id: ObjectId("123abc...") } },
  { $lookup: {
      from: "orders",
      localField: "_id",
      foreignField: "user_id",
      as: "user_orders"
  } }
]);
```

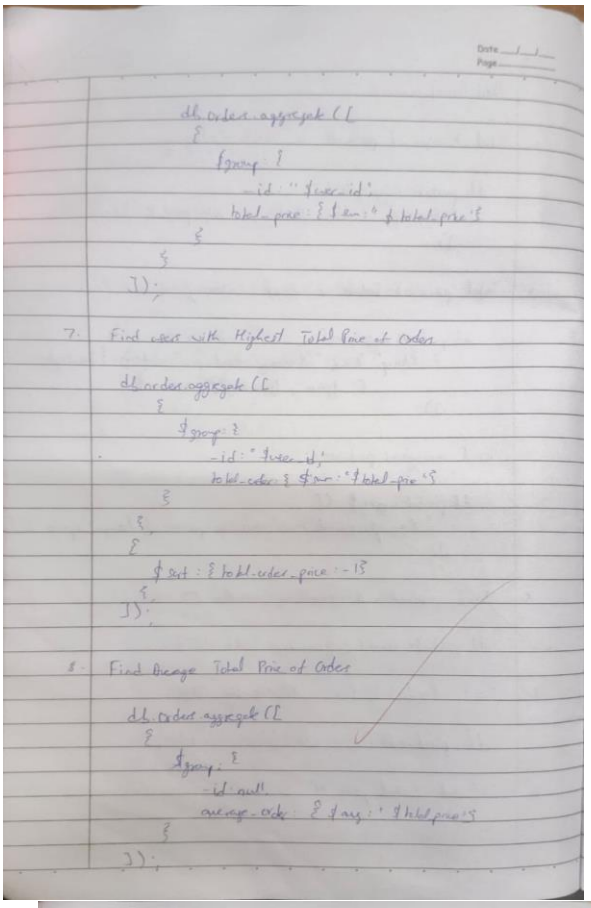
```
db.users.findOne({ _id: ObjectId("user1") }).cart;
```

```
db.products.find({ _id: ObjectId("product1") });
```

```
db.orders.find({ user_id: ObjectId("user1") });
```

7. Retrieve Total price of Orders placed by a User (ID "123abc...")

```
db.orders.aggregate([
  { $match: { user_id: ObjectId("user1") } },
  { $group: { _id: "$user_id", total_order: { $sum: "$total_price" } } }
]);
```



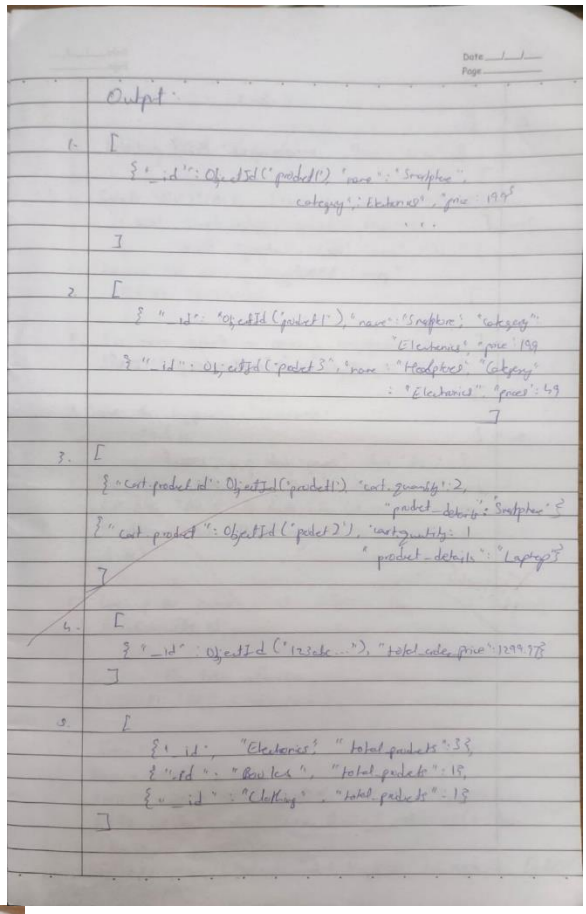
```
db.orders.aggregate([
  {
    $group: {
      _id: "$user_id",
      total_price: { $sum: "$total_price" }
    }
  }
]);
```

7. Find users with Highest Total Price of Orders

```
db.orders.aggregate([
  {
    $group: {
      _id: "$user_id",
      total_order: { $sum: "$total_price" }
    }
  },
  {
    $sort: { total_order_price: -1 }
  }
]);
```

8. Find Average Total Price of Orders

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



Output:

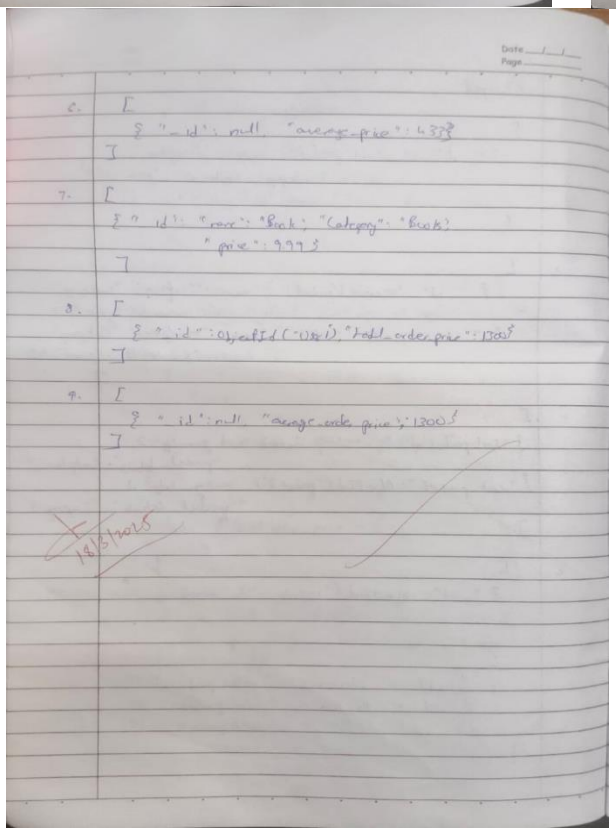
1.
```
[
  { "_id": ObjectId("product1"), "name": "Smartphone",
    category": "Electronics", "price": 199
    ...
]
```

2.
```
[
  { "_id": ObjectId("product1"), "name": "Smartphone", "category":
    "Electronics", "price": 199
  { "_id": ObjectId("product3"), "name": "Headphones", "Category"
    : "Electronics", "price": 49
]
```

3.
```
[
  { "cart_product_id": ObjectId("product1"), "cart_quantity": 2,
    "product_details": "Smartphone" }
  { "cart_product": ObjectId("product"), "cart_quantity": 1
    "product_details": "Laptop" }
]
```

4.
```
[
  { "_id": ObjectId("123abc..."), "total_order_price": 1299.97 }
]
```

5.
```
[
  { "_id": "Electronics", "total_products": 3 }
  { "id": "Books", "total_products": 15,
  { "_id": "Clothing", "total_products": 18
]
```



6.
```
[
  { "_id": null, "average_price": 437 }
]
```

7.
```
[
  { "_id": "price": "Book", "Category": "Books"
    "price": 999 }
]
```

8.
```
[
  { "_id": ObjectId("user1"), "total_order_price": 1200 }
]
```

9.
```
[
  { "_id": null, "average_order_price": 1200 }
]
```

18/8/2025

Code andOutput:

```
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.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] 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.Products.aggregate([
...   { $group: { _id: null, avgPrice: { $avg: "$price" } } }
... ])
[ { _id: null, avgPrice: 378 } ]
Atlas atlas-68mz5p-shard-0 [primary] ecommerceDB> db.Products.find({ quantity: { $lt: 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().sort({ price: -1 })
```

```
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>
```

LAB-2

Keyspace & Table Operations 1. Create a keyspace named Students.

2. Describe all existing keyspaces.
3. View keyspace details from the system schema.
4. Use the Students keyspace.
5. Create a table named Students_Info with columns:
    - Roll_No (Primary Key)
    - StudName
    - DateOfJoining
    - last_exam_Percent
6. List all tables in the current keyspace.
7. Describe the structure of the Students_Info table.

Insert & Query Data

8. Insert multiple records into Students_Info using batch insert.
9. View all data from the table.
10. Retrieve records where Roll_No is 1, 2, or 3.
11. Try querying non-primary key column and handle the error.
12. Create an index on StudName.
13. Query based on the indexed column (StudName = 'Asha').
14. Select only Roll_No and StudName with limit 2 rows.
15. Select Roll_No using an alias (USN).

Update & Delete Operations

16. Update StudName for a specific Roll_No.
17. Attempt to update a primary key (expect failure).
18. Delete a specific column (LastExamPercent) for a row.

19. Delete an entire row by Roll_No.

Working with Collections

20. Add a hobbies column of type set<text>.
21. Add a language column of type list<text>.
22. Update the hobbies set by adding elements.
23. Retrieve updated record with hobbies.
24. Update the language list by adding elements.
25. Remove an element from a set using - operator.

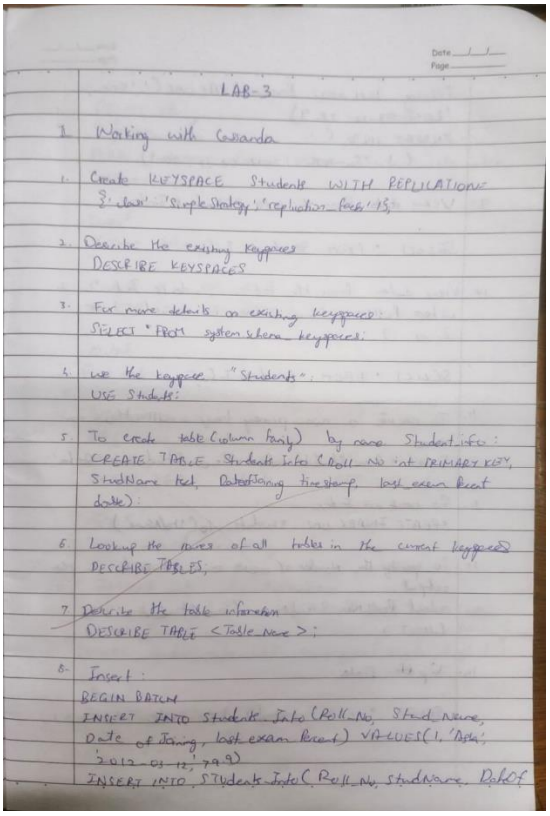
Counters

26. Create a table using a counter column.
27. Increment the counter value for a specific book-user pair.

Time To Live (TTL)

28. Create a table userlogin with TTL enabled on password.
29. Insert a record using TTL.
30. View TTL value of a specific column

Screenshot:



LAB-3

1. Working with Cassandra

1. Create KEYSPACE Students WITH REPLICATION= {'class': 'SimpleStrategy', 'replication_factor': 1};

2. Describe the existing keyspaces
DESCRIBE KEYSPACES

3. For more details on existing keyspaces:
SELECT * FROM system_schema.keyspaces;

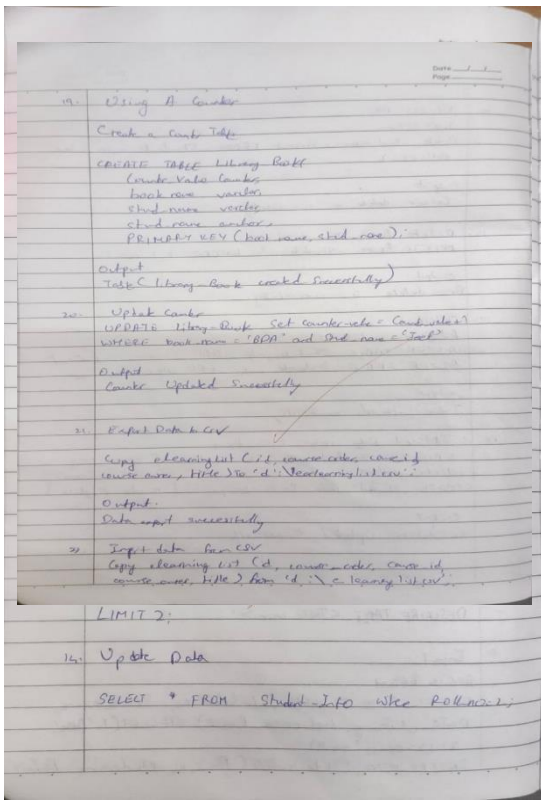
4. Use the keyspace "Students"
USE Students;

5. To create table (column family) by name Student_info:
CREATE TABLE Student_info (Roll_No int PRIMARY KEY, StudName text, DateOfJoining timestamp, last_exam_Percent double);

6. Lookup the names of all tables in the current keyspace
DESCRIBE TABLES;

7. Describe the table information
DESCRIBE TABLE <Table_Name>;

8. Insert:
BEGIN BATCH
INSERT INTO Student_Info(Roll_No, Stud_Name, Date_of_Joining, last_exam_Percent) VALUES(1, 'Ram', '2012-03-12', 79.9)
INSERT INTO STudent_Info(Roll_No, StudName, DoJ)



19. Using A Counter

Create a Counter Table

CREATE TABLE Library_Book
Counter_Value Counter,
book_name varchar,
Stud_name varchar,
stud_name author,
PRIMARY KEY (book_name, stud_name);

Output
Table(Library_Book created Successfully)

20. Update Counter
UPDATE Library_Book SET counter_value = Counter_value+1
WHERE book_name = 'BBA' and Stud_name = 'Jeet';

Output
Counter Updated Successfully

21. Export Data to CSV

Copy elearninglist (id, course_code, course_id, course_name, title) to 'd:\e-learning\list.csv';

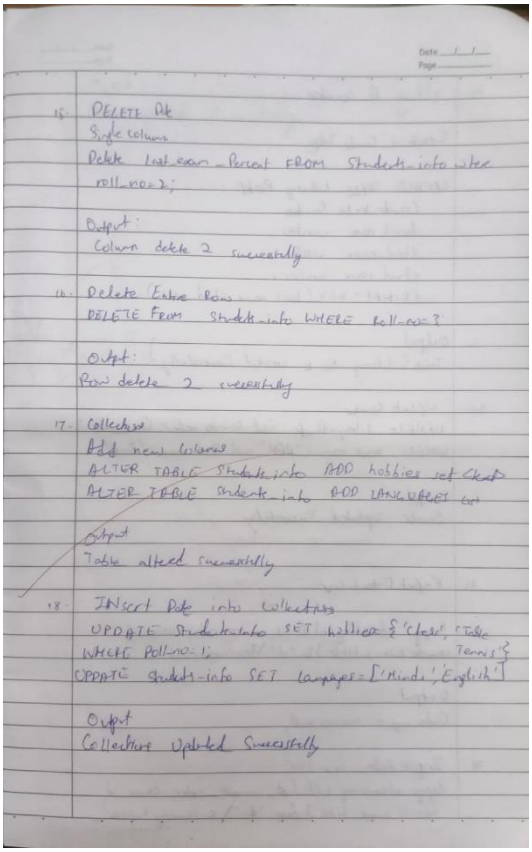
Output:
Data export successfully

22. Import data from CSV
Copy elearninglist (id, course_code, course_id, course_name, title) from 'd:\e-learning\list.csv';

LIMIT 2;

14. Update Data

SELECT * FROM Student_Info WHERE Roll_No=2;



15. DELETE Op
Single column
Delete Last_exam_Percent FROM Students_info where roll_no=2;

Output:
Column delete 2 successfully

16. Delete Entire Row
DELETE From Student_info WHERE Roll_no=?

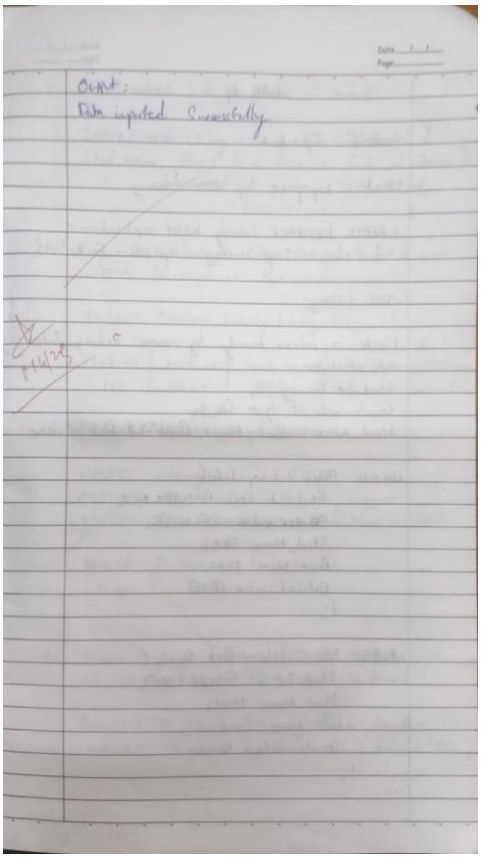
Output:
Row delete 2 successfully

17. Collection
Add new column
ALTER TABLE Student_info ADD hobbies set<text>
ALTER TABLE Student_info ADD LANGUAGES list

Output
Table altered Successfully

18. INSert Data into Collection
UPDATE Student_info SET hobbies = {'Cricket', 'Tennis'}
WHERE Roll_No=1;
UPDATE Student_info SET Languages = ['Hindi', 'English']

Output
Collection Updated Successfully



Output:
Data inputed Successfully

Code & Output:

```
ganajana@Anonymous:~$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | 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;

system_virtual_schema  system_schema  system_views  system_distributed  kitchen
students               system_auth    system        system_traces

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 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 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 crc_check_chance = 1.0
    AND default_time_to_live = 0
    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,'Asha','2012-03-12',79.9);
INSERT INTO Students_Info(Roll_No, StudName, DateOfJoining, last_exam_Percent) VALUES (2,'Krian','2012-03-12',89.9);
INSERT INTO Students_Info(Roll_No, StudName, DateOfJoining, last_exam_Percent) VALUES (3,'Tarun','2012-03-12',78.9);
INSERT INTO Students_Info(Roll_No, StudName, DateOfJoining, last_exam_Percent) VALUES (4,'Samrth','2012-03-12',90.9);
INSERT INTO Students_Info(Roll_No, StudName, DateOfJoining, last_exam_Percent) VALUES (5,'Smitha','2012-03-12',67.9);
INSERT INTO Students_Info(Roll_No, StudName, DateOfJoining, last_exam_Percent) VALUES (6,'Rohan','2012-03-12',56.9);
APPLY BATCH;
cqlsh:students> SELECT * FROM Students_Info;

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
       5 | 2012-03-12 00:00:00.000000+0000 |              67.9 |   Smitha
       1 | 2012-03-12 00:00:00.000000+0000 |              79.9 |     Asha
       2 | 2012-03-12 00:00:00.000000+0000 |              89.9 |    Krian
       4 | 2012-03-12 00:00:00.000000+0000 |              90.9 |   Samrth
       6 | 2012-03-12 00:00:00.000000+0000 |              56.9 |    Rohan
       3 | 2012-03-12 00:00:00.000000+0000 |              78.9 |    Tarun

(6 rows)
cqlsh:students> SELECT * FROM Students_Info WHERE Roll_No IN (1,2,3);

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
       1 | 2012-03-12 00:00:00.000000+0000 |              79.9 |     Asha
       2 | 2012-03-12 00:00:00.000000+0000 |              89.9 |    Krian
       3 | 2012-03-12 00:00:00.000000+0000 |              78.9 |    Tarun

(3 rows)
cqlsh:students> SELECT * FROM Students_Info WHERE StudName='Asha';
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may hav
ance. 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='Asha';

 roll_no | dateofjoining                   | last_exam_percent | studname
---------+---------------------------------+-------------------+----------
       1 | 2012-03-12 00:00:00.000000+0000 |              79.9 |     Asha

(1 rows)
```

```
cqlsh:students> SELECT Roll_No, StudName FROM Students_Info LIMIT 2;

 roll_no | studname
---------+----------
       5 |   Smitha
       1 |     Asha

(2 rows)
cqlsh:students> SELECT Roll_No AS USN FROM Students_Info;

 usn
-----
   5
   1
   2
   4
   6
   3

(6 rows)
cqlsh:students> UPDATE Students_Info SET StudName='David Sheen' WHERE Roll_No=2;
cqlsh:students> UPDATE Students_Info SET Roll_No=6 WHERE Roll_No=3;
InvalidRequest: Error from server: code=2200 [Invalid query] message="PRIMARY KEY part roll_no found in SET part"
cqlsh:students> DELETE last_exam_Percent FROM Students_Info WHERE Roll_No=2;
cqlsh:students> DELETE FROM Students_Info WHERE Roll_No=2;
cqlsh:students> ALTER TABLE Students_Info ADD hobbies set<text>;
cqlsh:students> ALTER TABLE Students_Info ADD language list<text>;
cqlsh:students> UPDATE Students_Info SET hobbies = hobbies + {'Chess','Table Tennis'} WHERE Roll_No=1;
cqlsh:students> SELECT * FROM Students_Info WHERE Roll_No=1;

 roll_no | dateofjoining                   | hobbies                  | language | last_exam_percent | studname
---------+---------------------------------+--------------------------+----------+-------------------+----------
       1 | 2012-03-12 00:00:00.000000+0000 | {'Chess', 'Table Tennis'} |     null |              79.9 |     Asha

(1 rows)
cqlsh:students> UPDATE Students_Info SET language = language + ['Hindi','English'] WHERE Roll_No=1;
cqlsh:students> UPDATE Students_Info SET hobbies = hobbies - {'Table Tennis'} WHERE Roll_No=1;
cqlsh:students> SELECT * FROM Students_Info;

 roll_no | dateofjoining                   | hobbies   | language                 | last_exam_percent | studname
---------+---------------------------------+-----------+--------------------------+-------------------+----------
       5 | 2012-03-12 00:00:00.000000+0000 |      null |                     null |              67.9 |   Smitha
       1 | 2012-03-12 00:00:00.000000+0000 | {'Chess'} | ['Hindi', 'English'] |              79.9 |     Asha
       4 | 2012-03-12 00:00:00.000000+0000 |      null |                     null |              90.9 |   Samrth
```

```
cqlsh:students> CREATE TABLE library_book (
  book_name varchar,
  stud_name varchar,
  counter_value counter,
  PRIMARY KEY(book_name, stud_name)
);
cqlsh:students> UPDATE library_book SET counter_value = counter_value + 1 WHERE book_name='Big data Analytics' AND stud_name='jeet'
cqlsh:students> SELECT * from library_book;

 book_name          | stud_name | counter_value
--------------------+-----------+---------------
 Big data Analytics |      jeet |             1

(1 rows)
cqlsh:students>
cqlsh:students> CREATE TABLE userlogin (userid int PRIMARY KEY, password text);
cqlsh:students> INSERT INTO userlogin(userid, password) VALUES (1,'infy') USING TTL 30;
cqlsh:students> SELECT TTL(password) FROM userlogin WHERE userid=1;

 ttl(password)
---------------
            22

(1 rows)
```

LAB– 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

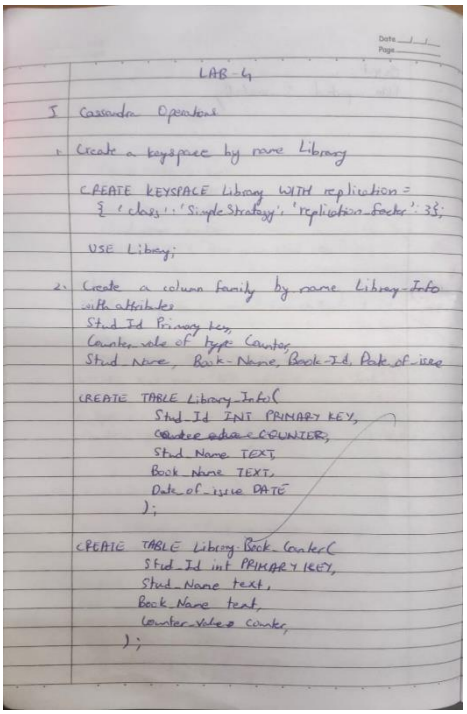
d) Display the details of the table created and increase the value of the counter

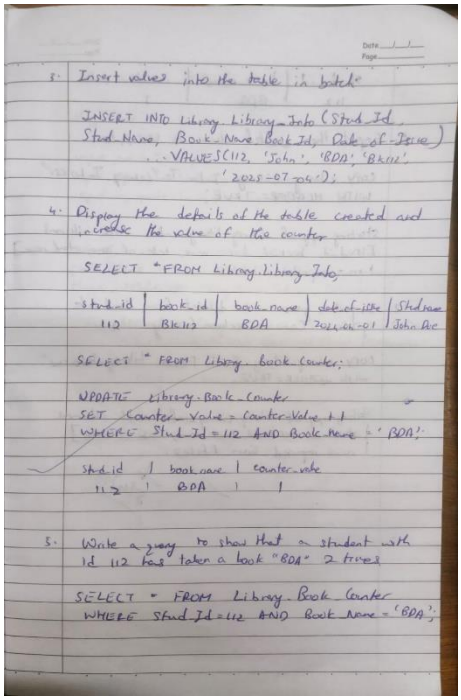
e) Write a query to show that a student with id 112 has taken a book “BDA” 2 times

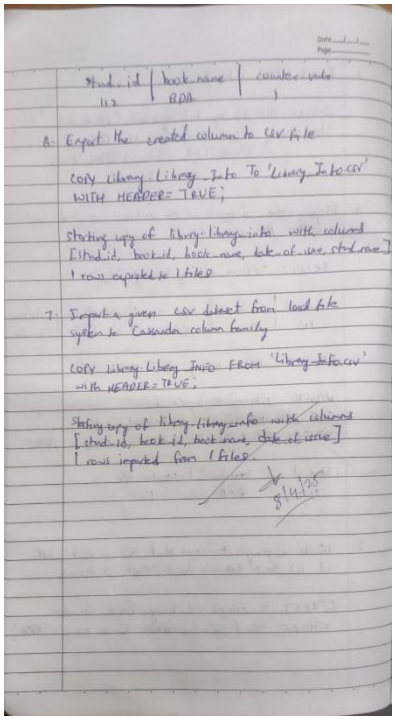
f) Export the created column to a CSV file

g) Import a given CSV dataset from local file system into Cassandra column family

Screenshot:







Code & Output:

```
ganajana@Anonymous:~$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 4.1.4 | CQL spec 3.4.6 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE KEYSPACE Library WITH REPLICATION = {
  'class': 'SimpleStrategy',
  'replication_factor': 1
};
AlreadyExists: Keyspace 'library' already exists
cqlsh> USE Library;
cqlsh:library> CREATE TABLE Library_Info (
  Stud_Id int,
  Stud_Name text,
  Book_Name text,
  Book_Id text,
  Date_of_Issue date,
  Counter_Value counter,
  PRIMARY KEY ((Stud_Id), Book_Name)
);
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot mix counter and non counter columns in the same table"
cqlsh:library> CREATE TABLE Library_Info (
  Stud_Id int,
  Book_Name text,
  Counter_Value counter,
  PRIMARY KEY ((Stud_Id), Book_Name)
);
```

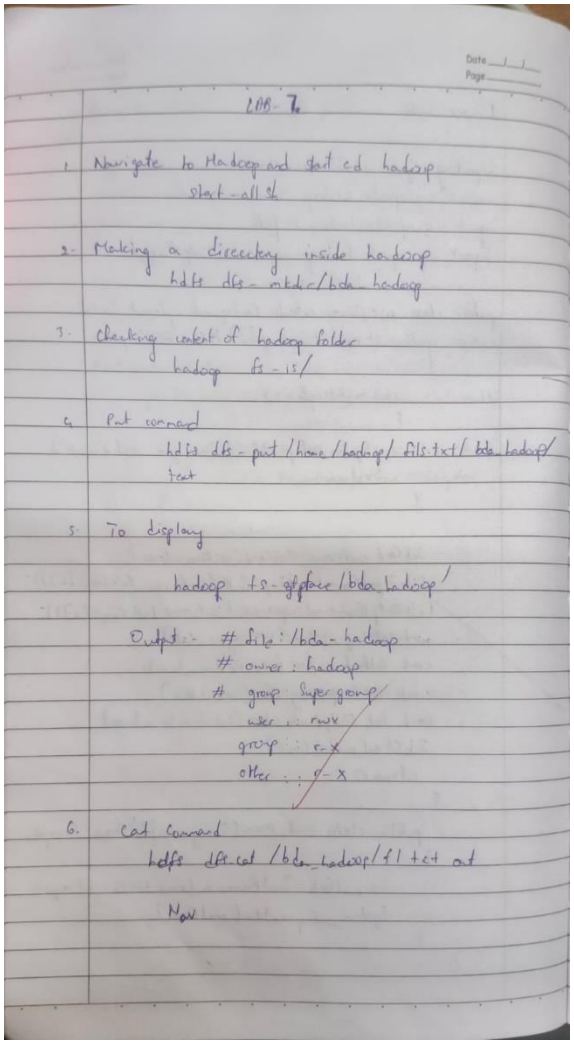
```
cqlsh:library> -- Student 112 takes the book "BDA"
UPDATE Library_Info
SET Counter_Value = Counter_Value + 1
WHERE Stud_Id = 112 AND Book_Name = 'BDA';

-- Student 112 takes the book "BDA" again
UPDATE Library_Info
SET Counter_Value = Counter_Value + 1
WHERE Stud_Id = 112 AND Book_Name = 'BDA';

-- Student 113 takes the book "ML"
UPDATE Library_Info
SET Counter_Value = Counter_Value + 1
WHERE Stud_Id = 113 AND Book_Name = 'ML';
cqlsh:library> -- View all counter data
SELECT * FROM Library_Info;

-- Check how many times student 112 took "BDA"
SELECT Counter_Value FROM Library_Info
WHERE Stud_Id = 112 AND Book_Name = 'BDA';

-- Get all books taken by student 112
SELECT * FROM Library_Info WHERE Stud_Id = 112;

 stud_id | book_name | counter_value
---------+-----------+---------------
     113 |        ML |             1
     112 |       BDA |             2

(2 rows)

 counter_value
---------------
             2

(1 rows)

 stud_id | book_name | counter_value
---------+-----------+---------------
     112 |       BDA |             2

(1 rows)
```

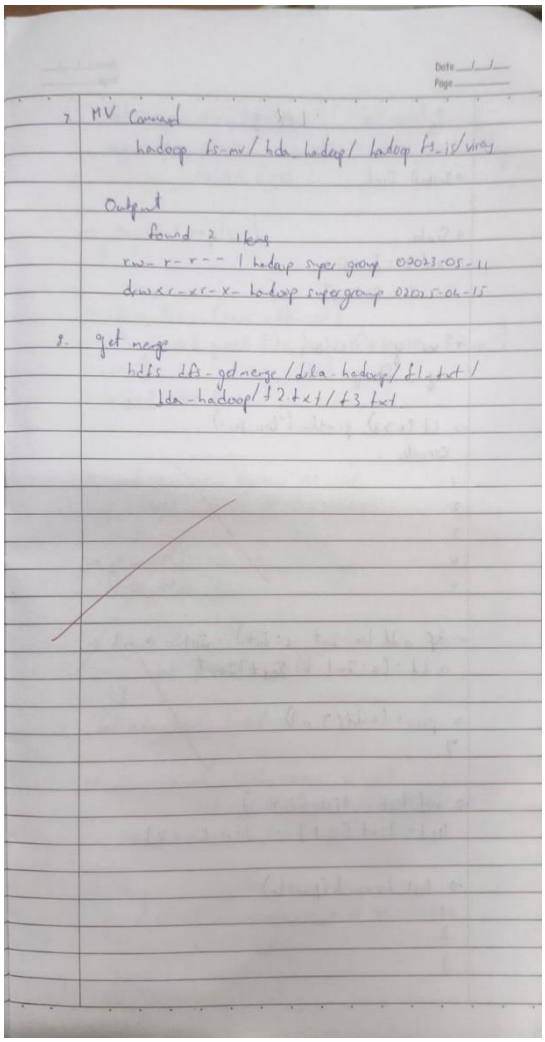
```
cqlsh:library> -- Delete record of student 113 for book "ML"
DELETE FROM Library_Info
WHERE Stud_Id = 113 AND Book_Name = 'ML';
cqlsh:library> COPY Library_Info (Stud_Id, Book_Name, Counter_Value)
TO 'D:\\library_info.csv';
Using 15 child processes

Starting copy of library.library_info with columns [stud_id, book_name, counter_value].
Processed: 1 rows; Rate:       0 rows/s; Avg. rate:       1 rows/s
1 rows exported to 1 files in 1.359 seconds.
cqlsh:library>
```

LAB- 4

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

Screenshot:





Code & Output:

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ cd ./Desktop/
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [bmscecse-HP-Elite-Tower-800-G9-Desktop-PC]
Starting resourcemanager
Starting nodemanagers
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -mkdir /Lab05
```

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hadoop fs -ls /Hadoop
ls: `/Hadoop': No such file or directory
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hadoop fs -ls /Lab05
```

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ touch test.txt
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ nano text.txt
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -put ./text.txt /Lab05/text.txt
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hadoop fs -ls /Lab05
Found 1 items
-rw-r--r--   1 hadoop supergroup         19 2024-05-13 14:33 /Lab05/text.txt
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -cat /Lab05/text.txt
Hello
How are you?
```

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hadoop fs -ls /Lab05
Found 2 items
-rw-r--r--   1 hadoop supergroup         15 2024-05-13 14:40 /Lab05/test.txt
-rw-r--r--   1 hadoop supergroup         19 2024-05-13 14:33 /Lab05/text.txt
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -getmerge /Lab05 /text.txt /Lab05 /test.txt ../
Downloads/Merged.txt
getmerge: `/text.txt': No such file or directory
getmerge: `/test.txt': No such file or directory
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -getmerge /Lab05/text.txt /Lab05/test.txt ../Do
wnloads/Merged.txt
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hadoop fs -getfacl /Lab05
# file: /Lab05
# owner: hadoop
# group: supergroup
user::rwx
group::r-x
other::r-x
```

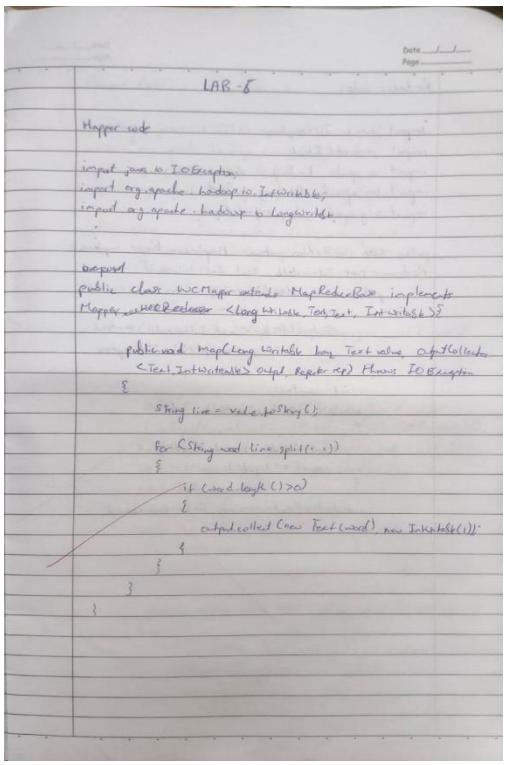
```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -copyToLocal /Lab05/text.txt ../Documents
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -copyToLocal /Lab05/test.txt ../Documents
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -cat /Lab05/text.txt
Hello
How are you?
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -mv /Lab05 /test_Lab05
```

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -ls /test_Lab05
Found 2 items
-rw-r--r--   1 hadoop supergroup         15 2024-05-13 14:40 /test_Lab05/test.txt
-rw-r--r--   1 hadoop supergroup         19 2024-05-13 14:33 /test_Lab05/text.txt
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -cp /test_Lab05/ /Lab05
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -ls /Lab05
Found 2 items
-rw-r--r--   1 hadoop supergroup         15 2024-05-13 14:51 /Lab05/test.txt
-rw-r--r--   1 hadoop supergroup         19 2024-05-13 14:51 /Lab05/text.txt
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~/Desktop$ hdfs dfs -ls /test_Lab05
Found 2 items
-rw-r--r--   1 hadoop supergroup         15 2024-05-13 14:40 /test_Lab05/test.txt
-rw-r--r--   1 hadoop supergroup         19 2024-05-13 14:33 /test_Lab05/text.txt
```

LAB - 5

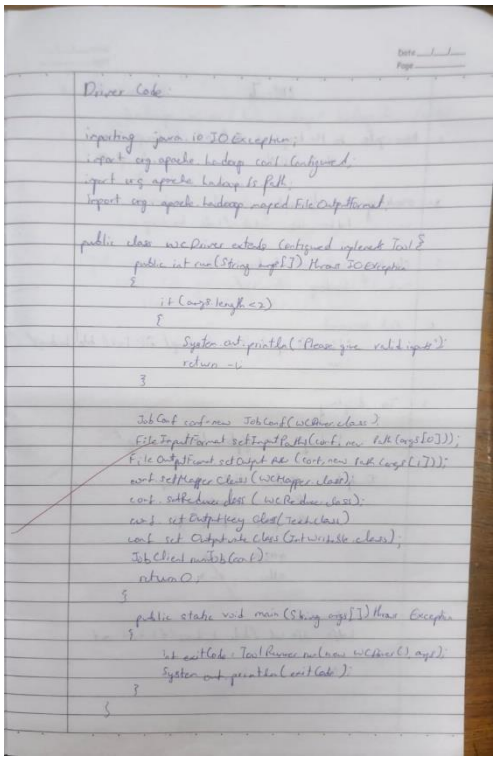
Implement Wordcount program on Hadoop framework

Screenshot:

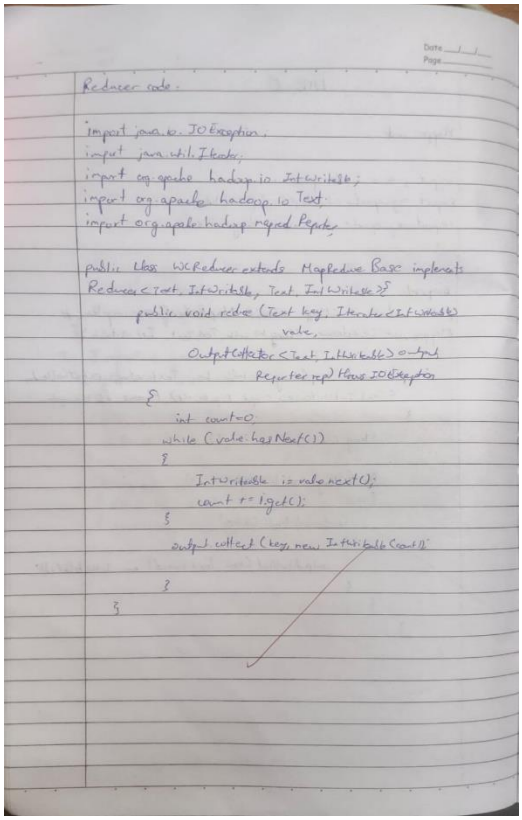

LAB-5

Mapper code

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;

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

    public void map(LongWritable key, Text value, OutputCollector
    <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));
            }
        }
    }
}
```


Driver Code:

```
importing java.io.IOException;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.mapred.FileOutputFormat;

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


Reducer code:

```
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.Reporter;

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

Code & Output:

```
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC: $ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [bmscecse-HP-Elite-Tower-600-G9-Desktop-PC]
Starting resourcemanager
Starting nodemanagers
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ nano mapper.py
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ nano reducer.py
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ chmod +x mapper.py
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ chmod +x reducer.py
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ hdfs dfs -mkdir /bdaa
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ hdfs dfs -ls /
Found 6 items
drwxr-xr-x   - hadoop supergroup          0 2025-04-15 14:32 /abc
drwxr-xr-x   - hadoop supergroup          0 2025-05-20 12:51 /bda
drwxr-xr-x   - hadoop supergroup          0 2025-05-21 13:36 /bdaa
drwxr-xr-x   - hadoop supergroup          0 2025-04-15 14:48 /hadoop_lab
drwxr-xr-x   - hadoop supergroup          0 2025-05-06 15:51 /rgs
drwxr-xr-x   - hadoop supergroup          0 2025-05-20 15:19 /word
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ nano input.txt
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ hdfs dfs -put input.txt /bdaa
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ hdfs dfs -ls /bdaa
Found 1 items
-rw-r--r--   1 hadoop supergroup       1243 2025-05-21 13:42 /bdaa/input.txt
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ hadoop jar /home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar \
  -input /bdaa/input.txt \
  -output /bdaa/output \
  -mapper mapper.py \
  -reducer reducer.py \
  -file /home/hadoop/mapper.py \
  -file /home/hadoop/reducer.py
```

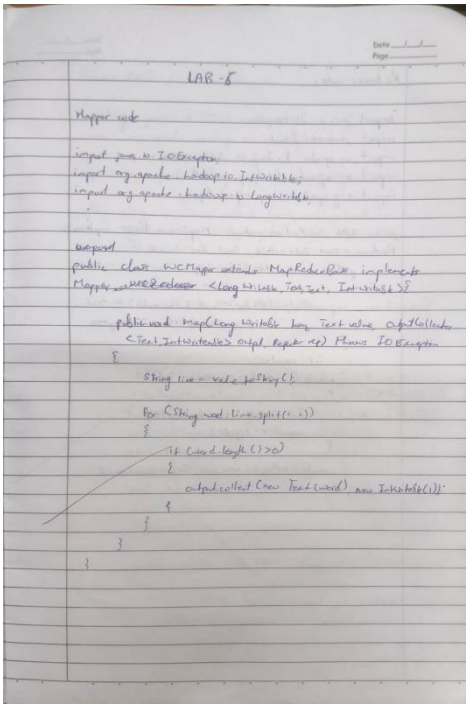
```
2025-05-21 13:45:40,632 INFO streaming.StreamJob: Output directory: /bdaa/output
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ hdfs dfs -ls /bdaa
Found 2 items
-rw-r--r--   1 hadoop supergroup       1243 2025-05-21 13:42 /bdaa/input.txt
drwxr-xr-x   - hadoop supergroup          0 2025-05-21 13:45 /bdaa/output
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ hdfs dfs -ls /bdaa/output
Found 2 items
-rw-r--r--   1 hadoop supergroup          0 2025-05-21 13:45 /bdaa/output/_SUCCESS
-rw-r--r--   1 hadoop supergroup       1190 2025-05-21 13:45 /bdaa/output/part-00000
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ hdfs dfs -cat /bdaa/output/part-00000
'Content        1
'lorem  1
(injected       1
1500s,  1
1960s   1
Aldus   1
English.        1
Ipsum   5
Ipsum.  1
It      3
Letraset        1
Lorem   7
Many    1
PageMaker       1
The     1
Various 1
Why     1
a       7
accident,       1
also    1
an      1
and     6
as      2
at      1
be      1
been    1
book.   1
but     1
by      2
```

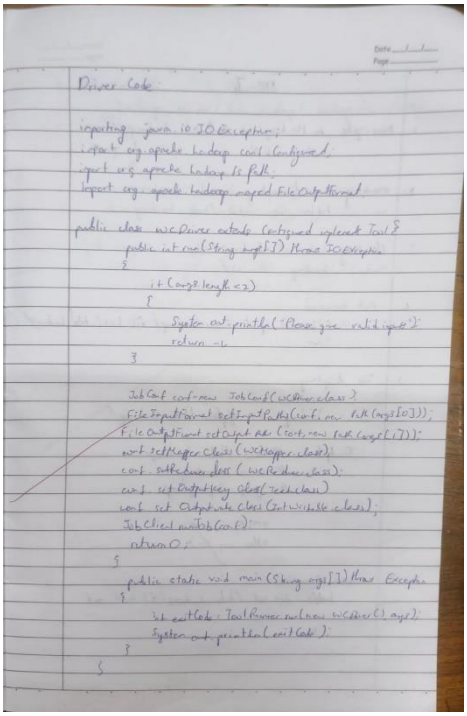
LAB – 6

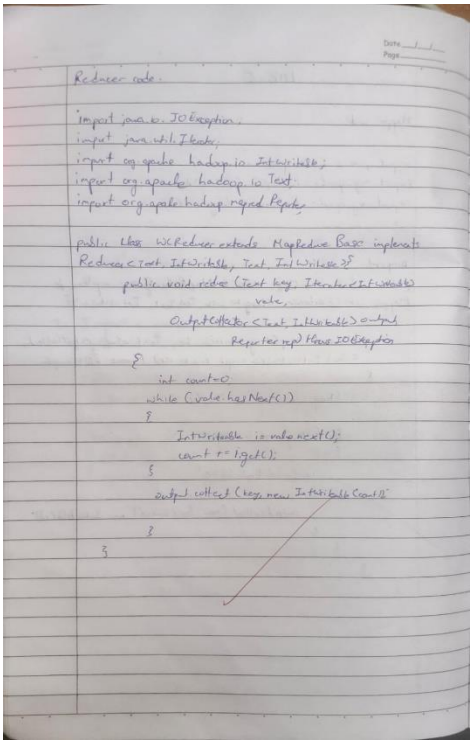
From the following link extract the weather data
https://github.com/tomwhite/hadoopbook/tree/master/input/ncdc/all

Create a Map Reduce program to
a) find average temperature for each year from NCDC data set.
b) find the mean max temperature for every month.

Screenshot:







Code & Output:

a) Find average temperature for each year from NCDC data set

```
  GNU nano 7.2                                                    map.py *
import sys
for line in sys.stdin:
        year=line[15:19]
        temp=line[87:92]
        if temp!="+9999":
                print(f"{year}\t{temp}")
```

```
  GNU nano 7.2                                                    red.py *
import sys

current_year = None
temp_sum = 0
count = 0

for line in sys.stdin:
    year, temp = line.strip().split('\t')
    temp = int(temp)

    if current_year == year:
        temp_sum += temp
        count += 1
    else:
        if current_year:
            print(f"{current_year}\t{temp_sum/count:.2f}")
        current_year = year
        temp_sum = temp
        count = 1

# print last year
if current_year:
    print(f"{current_year}\t{temp_sum/count:.2f}")
```

```
ganajana@Anonymous:~$ hdfs dfs -mkdir /weatherdata
ganajana@Anonymous:~$ git clone https://github.com/tomwhite/hadoop-book.git
fatal: destination path 'hadoop-book' already exists and is not an empty directory.
ganajana@Anonymous:~$ git clone https://github.com/tomwhite/hadoop-book.git
Cloning into 'hadoop-book'...
remote: Enumerating objects: 4969, done.
remote: Total 4969 (delta 0), reused 0 (delta 0), pack-reused 4969 (from 1)
Receiving objects: 100% (4969/4969), 2.85 MiB | 6.78 MiB/s, done.
Resolving deltas: 100% (1945/1945), done.
ganajana@Anonymous:~$ hdfs dfs -put hadoop-book/input/ncdc/all/*/weatherdata
put: `.': No such file or directory: `hdfs://localhost:9000/user/ganajana'
ganajana@Anonymous:~$ hdfs dfs -ls /
Found 1 items
drwxr-xr-x   - ganajana supergroup          0 2025-05-26 16:27 /weatherdata
ganajana@Anonymous:~$ hdfs dfs -put hadoop-book/input/ncdc/all/*/weatherdata
put: `.': No such file or directory: `hdfs://localhost:9000/user/ganajana'
ganajana@Anonymous:~$ hdfs dfs -put hadoop-book/input/ncdc/all/* /weatherdata
ganajana@Anonymous:~$ nano map.py
ganajana@Anonymous:~$ chmod +x map.py
ganajana@Anonymous:~$ nano red.py
ganajana@Anonymous:~$ chmod +x red.py
ganajana@Anonymous:~$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.4.1.jar -input /weatherdata -output /weatherdata/out -mapper map.py
-reducer red.py
```

```
Mean Temperature: 31.18
```

b) Find the mean max temperature for every month

```
 GNU nano 7.2                                                    map.py *
import sys

for line in sys.stdin:
    year = line[15:19]
    month = line[19:21]  # extract month from positions 19-20
    temp = line[87:92]
    if temp != "+9999":  # valid temperature
        print(f"{year}-{month}\t{temp}")
```

```
 GNU nano 7.2                                                    red.py *
import sys

current_year_month = None
temp_sum = 0
count = 0

for line in sys.stdin:
    year_month, temp = line.strip().split('\t')
    temp = int(temp)

    if current_year_month == year_month:
        temp_sum += temp
        count += 1
    else:
        if current_year_month:
            print(f"{current_year_month}\t{temp_sum/count:.2f}")
        current_year_month = year_month
        temp_sum = temp
        count = 1

# print last year-month
if current_year_month:
    print(f"{current_year_month}\t{temp_sum/count:.2f}")
```

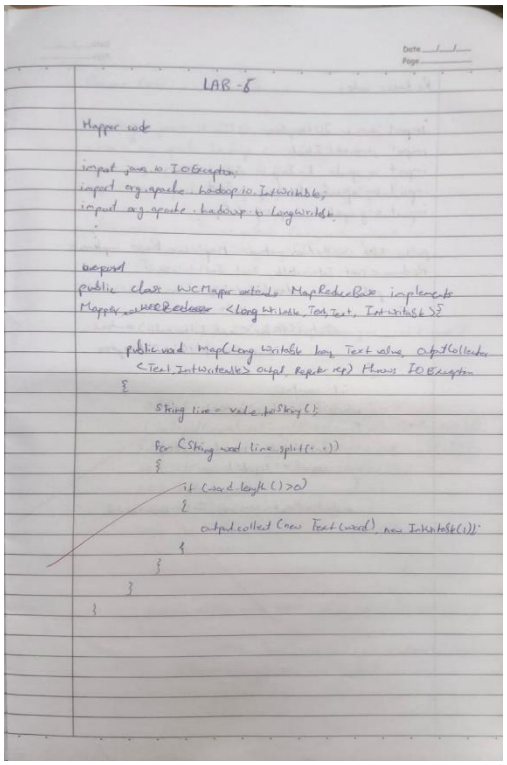
```
ganajana@Anonymous:~$ hdfs dfs -mkdir /weatherdata
ganajana@Anonymous:~$ git clone https://github.com/tomwhite/hadoop-book.git
fatal: destination path 'hadoop-book' already exists and is not an empty directory.
ganajana@Anonymous:~$ git clone https://github.com/tomwhite/hadoop-book.git
Cloning into 'hadoop-book'...
remote: Enumerating objects: 4969, done.
remote: Total 4969 (delta 0), reused 0 (delta 0), pack-reused 4969 (from 1)
Receiving objects: 100% (4969/4969), 2.85 MiB | 6.78 MiB/s, done.
Resolving deltas: 100% (1945/1945), done.
ganajana@Anonymous:~$ hdfs dfs -put hadoop-book/input/ncdc/all/*/weatherdata
put: `.': No such file or directory: `hdfs://localhost:9000/user/ganajana'
ganajana@Anonymous:~$ hdfs dfs -ls /
Found 1 items
drwxr-xr-x   - ganajana supergroup          0 2025-05-26 16:27 /weatherdata
ganajana@Anonymous:~$ hdfs dfs -put hadoop-book/input/ncdc/all/*/weatherdata
put: `.': No such file or directory: `hdfs://localhost:9000/user/ganajana'
ganajana@Anonymous:~$ hdfs dfs -put hadoop-book/input/ncdc/all/* /weatherdata
ganajana@Anonymous:~$ nano map.py
ganajana@Anonymous:~$ chmod +x map.py
ganajana@Anonymous:~$ nano red.py
ganajana@Anonymous:~$ chmod +x red.py
ganajana@Anonymous:~$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.4.1.jar -input /weatherdata -output /weatherdata/out -mapper map.py
-reducer red.py
```

```
Max Temperature: 33.50
```

LAB– 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.

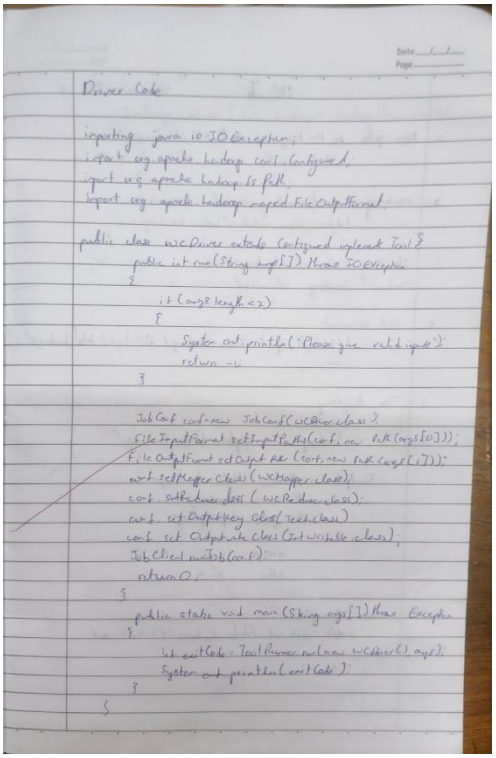
Screenshot:



LAB-5

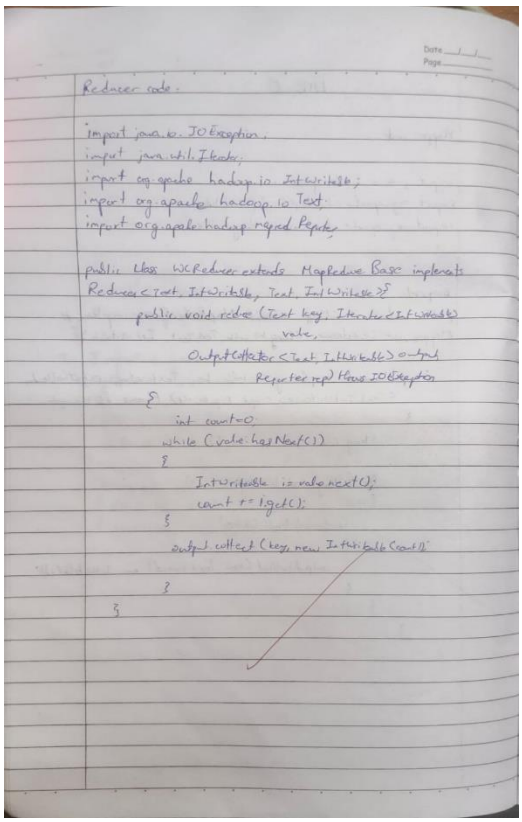
Mapper code

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;

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

    public void map(LongWritable key, Text value, OutputCollector
    <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));
            }
        }
    }
}
```



Driver Code:

```
importing java.io.IOException;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.mapred.FileOutputFormat;

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



Reducer code:

```
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.Reporter;

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

    }
}
```

Code & Output:

```
  GNU nano 7.2                                                    map1.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")
```

```
  GNU nano 7.2                                                    red1.py *
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}")
```

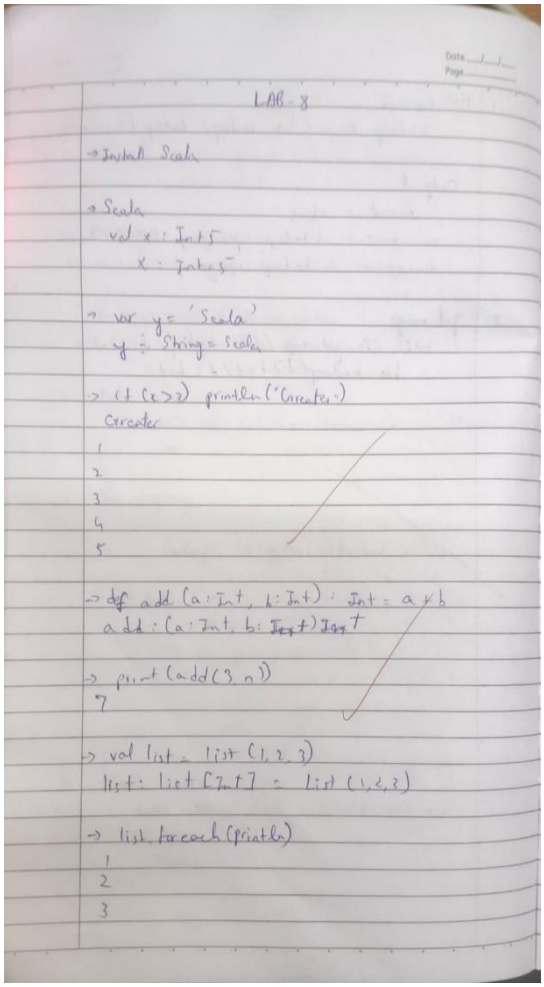
```
ganajana@Anonymous:~$ jps
4484 SecondaryNameNode
4277 DataNode
5461 NodeManager
6730 NameNode
7885 Jps
5326 ResourceManager
ganajana@Anonymous:~$ nano map1.py
ganajana@Anonymous:~$ chmod +x map1.py
ganajana@Anonymous:~$ nano red1.py
ganajana@Anonymous:~$ chmod +x red1.py
ganajana@Anonymous:~$ nano inpp.txt
ganajana@Anonymous:~$ hdfs dfs -mkdir /topp
ganajana@Anonymous:~$ hdfs dfs -put inpp.txt /topp
ganajana@Anonymous:~$ hdfs dfs -ls /topp
Found 1 items
-rw-r--r--   3 ganajana supergroup         66 2025-05-26 16:57 /topp/inpp.txt
ganajana@Anonymous:~$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.4.1.jar -input /topp/inpp.txt -output /topp/out -mapper map1.py -red
ucer red1.py
```

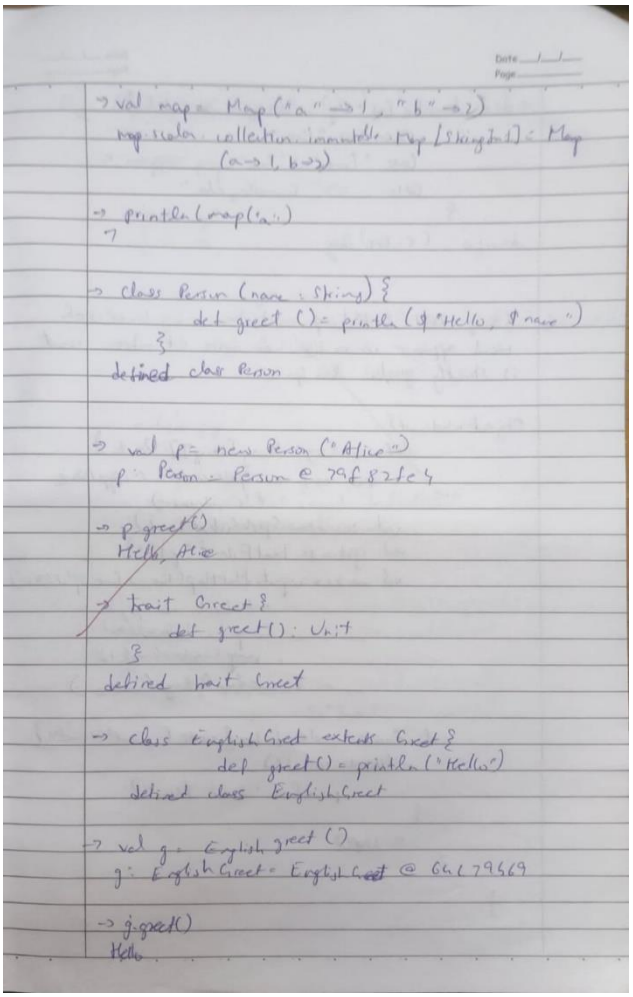
```
ganajana@Anonymous:~$ hdfs dfs -cat /topp/outt/part-00000
bye     2
hello   2
you     2
are     1
fine    1
gaj     1
great   1
hi      1
how     1
im      1
ganajana@Anonymous:~$
```

LAB – 8

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

Screenshot:





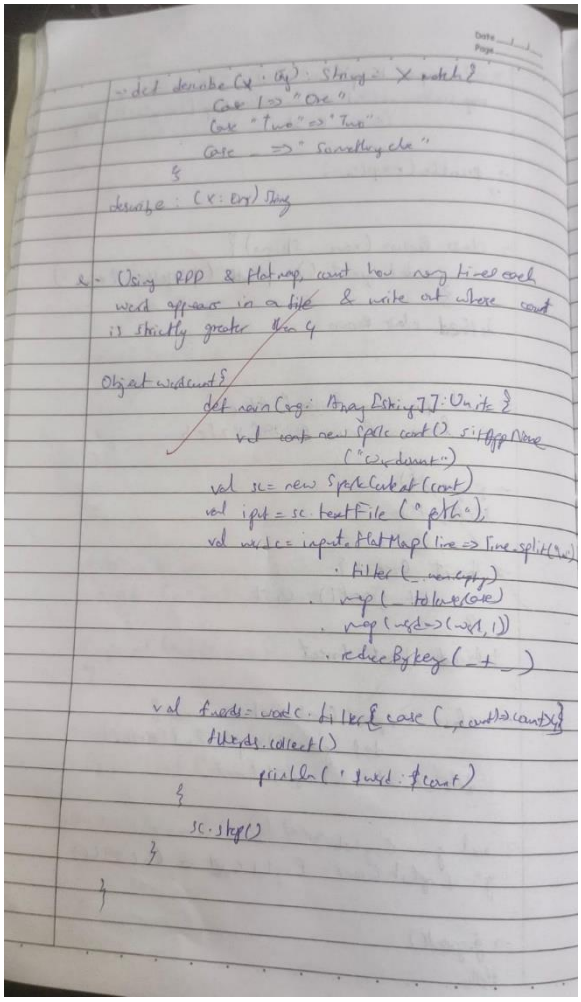
Code & Output:

```
  GNU nano 7.2                                                cry.scala *
object cry{
        def main(args:Array[String]):Unit={
                for(i<-1 to 100){
                        print(i + " ")
                }
        }
}
```

LAB – 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.

Screenshot:



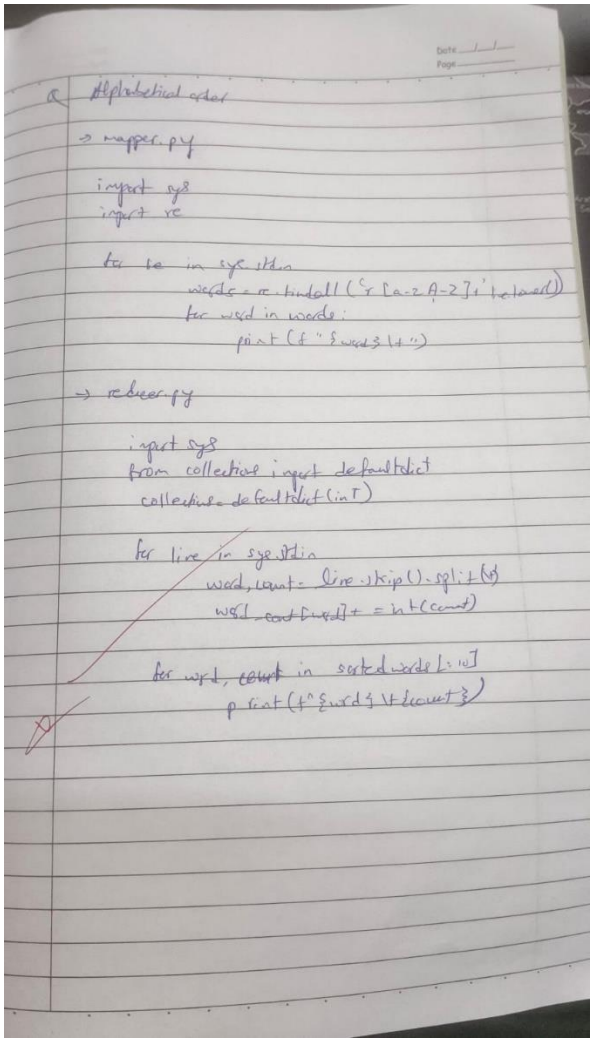
Code & Output:

```
ganajana@Anonymous:~$ spark-shell
25/05/26 07:34:04 WARN Utils: Your hostname, Anonymous resolves to a loopback address: 127.0.1.1; using 10.255.255.254 instead (on interface lo)
25/05/26 07:34:04 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).
25/05/26 07:34:10 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
25/05/26 07:34:10 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
Spark context Web UI available at http://10.255.255.254:4041
Spark context available as 'sc' (master = local[*], app id = local-1748244851075).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.1
      /_/

Using Scala version 2.12.18 (OpenJDK 64-Bit Server VM, Java 1.8.0_452)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val input = sc.textFile("file:///home/ganajana/sample.txt")
input: org.apache.spark.rdd.RDD[String] = file:///home/ganajana/sample.txt MapPartitionsRDD[1] at textFile at <console>:23

scala> val words = textFile.flatMap(line => line.split(" "))
<console>:22: error: not found: value textFile
       val words = textFile.flatMap(line => line.split(" "))
                   ^

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

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

scala> val counts = pairs.reduceByKey((a, b) => a + b)
counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:23

scala> val result = counts.filter { case (_, count) => count > 4 }
result: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[5] at filter at <console>:23

scala> result.collect().foreach(println)
(gaj,8)
(hello,6)
```

LAB – 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).

Screenshot:



Code & Output:

```
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  |
+------+
```