Develop a MapReduce program to calculate the frequency of a given word in a given file.

Implement Matrix Multiplication using Map-Reduce

Develop a MapReduce program to find the grades of students.

The AI & DS Department of a college wants to digitize and manage student data using a NoSQL database. As a developer, build a application to manage student records. Each student record should include details such as roll number, name, branch, year, and marks.

You are required to:

1. Create a **database** named CollegeDB.
2. Create a **collection** named Students within the database.
3. Perform the following **CRUD operations** on the Students collection:
    - **Insert** multiple student records.
    - **Query** student records based on different criteria (e.g., all records, marks > 80).
    - **Update** existing student details (e.g., marks).
    - **Delete** a specific student record.

A company wants to shift its employee record system from spreadsheets to a NoSQL database. As a software engineer, you are required to create and manage employee records using MongoDB. Each employee's information should include Employee ID, Name, Department, Designation, and Salary.

You are required to:

1. Create a **database** named CompanyDB.
2. Create a **collection** named Employees within the database.
3. Perform the following **CRUD operations**:
    - **Insert** one or more employee records.
    - **Query** employee records based on different conditions.
    - **Update** employee details such as salary or designation.
    - **Delete** an employee record using a specific condition.

A college library wants to maintain a digital database of books instead of keeping handwritten registers. As a backend developer, you are assigned to develop a MongoDB-based system to manage books. Each book entry should include a Book ID, Title, Author, Category, and Availability status.

You are required to:

1. Create a **database** named LibraryDB.
2. Create a **collection** named Books.
3. Perform the following **CRUD operations**:
    - **Insert** multiple book records.
    - **Query** specific books using filter conditions.
    - **Update** book details such as availability status.
    - **Delete** a book record from the database.

An e-commerce startup wants to build a product catalog database to store and manage product information. As a database developer, your task is to use MongoDB to:

- Insert multiple products into the catalog.
- Query specific products using filters.
- Update and delete product information.
- Sort products by price.
- Project only selected fields from the data.
- Limit the number of results.
- Use aggregation to compute category-wise average prices.

A college wants to digitize student academic records. As a database developer, your task is to use MongoDB to:

- Create a database of student records.
- Perform CRUD operations on the data.
- Sort and filter records based on performance.
- Display selected fields.
- Limit the number of results.
- Use aggregation to compute average marks department-wise.

To connect Power BI to a dataset, perform data analysis, build meaningful visualizations, create an interactive dashboard, and compile insights into a story for effective decision-making.

The AI & DS Department of a college wants to digitize and manage student data using a NoSQL database. As a developer, build a application to manage student records. Each student record should include details such as roll number, name, branch, year, and marks.

You are required to:

1. Create a **database** named CollegeDB.
2. Create a **collection** named Students within the database.
3. Perform the following **CRUD operations** on the Students collection:
    - **Insert** multiple student records.
    - **Query** student records based on different criteria (e.g., all records, marks > 80).
    - **Update** existing student details (e.g., marks).
    - **Delete** a specific student record.

Solution:

Step 1: Install MongoDB

- For **Ubuntu**:

  sudo systemctl start mongodb
  sudo systemctl enable mongodb

Step 2: Start MongoDB Shell

mongosh

Step 3: Create Database and Collection

```
use CollegeDB          // Switch to (or create) database

db.createCollection("Students")   // Create collection
```

Step 4: Insert Documents

```
db.Students.insertOne({
  roll_no: "CSE1001",
  name: "Ravi Sharma",
  branch: "CSE",
  year: 2,
  marks: 88
})

db.Students.insertMany([
  {
```

```
  roll_no: "CSE1002",
  name: "Sneha Patil",
  branch: "CSE",
  year: 2,
  marks: 91
 },
 {
  roll_no: "CSE1003",
  name: "Amit Verma",
  branch: "IT",
  year: 1,
  marks: 76
 }
])
```

Step 5: Query Documents

- Find all students:

  ```
  db.Students.find().pretty()
  ```

- Find students with marks greater than 80:

  ```
  db.Students.find({ marks: { $gt: 80 } })
  ```

Step 6: Update Document

- Update marks of student with roll number "CSE1001":

  ```
  db.Students.updateOne(
    { roll_no: "CSE1001" },
    { $set: { marks: 90 } }
  )
  ```

Step 7: Delete Document

- Delete student with roll number "CSE1003":

  ```
  db.Students.deleteOne({ roll_no: "CSE1003" })
  ```

A company wants to shift its employee record system from spreadsheets to a NoSQL database. As a software engineer, you are required to create and manage employee records using MongoDB. Each employee's information should include Employee ID, Name, Department, Designation, and Salary.

You are required to:

1. Create a **database** named CompanyDB.
2. Create a **collection** named Employees within the database.
3. Perform the following **CRUD operations**:
    - **Insert** one or more employee records.
    - **Query** employee records based on different conditions.
    - **Update** employee details such as salary or designation.
    - **Delete** an employee record using a specific condition.

Solution:

Step 1: Installation (Ubuntu Example)

```
sudo systemctl start mongodb
sudo systemctl enable mongodb
```

Step 2: Start MongoDB Shell

```
mongosh
```

Step 3: Create Database and Collection

```
use CompanyDB
db.createCollection("Employees")
```

Step 4: Insert Records

```
db.Employees.insertOne({
  emp_id: "EMP001",
  name: "Priya Desai",
  department: "HR",
  designation: "Manager",
  salary: 65000
})

db.Employees.insertMany([
  {
    emp_id: "EMP002",
    name: "Ankit Verma",
    department: "IT",
    designation: "Developer",
    salary: 72000
  },
  {
    emp_id: "EMP003",
    name: "Ritika Shah",
    department: "Finance",
    designation: "Analyst",
```

```
    salary: 58000
  }
])
```

Step 5: Query Records

- Fetch all employees:

  ```
  db.Employees.find().pretty()
  ```

- Fetch employees with salary above ₹60,000:

  ```
  db.Employees.find({ salary: { $gt: 60000 } })
  ```

Step 6: Update Record

- Change designation of employee EMP002:

  ```
  db.Employees.updateOne(
    { emp_id: "EMP002" },
    { $set: { designation: "Senior Developer" } }
  )
  ```

Step 7: Delete Record

- Delete employee EMP003:

  ```
  db.Employees.deleteOne({ emp_id: "EMP003" })
  ```

A college library wants to maintain a digital database of books instead of keeping handwritten registers. As a backend developer, you are assigned to develop a MongoDB-based system to manage books. Each book entry should include a Book ID, Title, Author, Category, and Availability status.

You are required to:

1. Create a **database** named LibraryDB.
2. Create a **collection** named Books.
3. Perform the following **CRUD operations**:
   - **Insert** multiple book records.
   - **Query** specific books using filter conditions.
   - **Update** book details such as availability status.
   - **Delete** a book record from the database.

Solutions:

Step 1: Install MongoDB (on Ubuntu)

```
sudo systemctl start mongodb
sudo systemctl enable mongodb
```

Step 2: Start MongoDB Shell

```
mongosh
```

Step 3: Create Database and Collection

```
use LibraryDB
db.createCollection("Books")
```

Step 4: Insert Book Records

```
db.Books.insertMany([
  {
    book_id: "B001",
    title: "Database Systems",
    author: "Raghu Ramakrishnan",
    category: "Computer Science",
    available: true
  },
  {
    book_id: "B002",
    title: "Clean Code",
    author: "Robert C. Martin",
    category: "Programming",
    available: false
  },
  {
    book_id: "B003",
    title: "Introduction to Algorithms",
    author: "Cormen et al.",
    category: "Algorithms",
    available: true
  }
])
```

Step 5: Query Book Records

- Show all books:

  ```
  db.Books.find().pretty()
  ```

- Show available books:

  ```
  db.Books.find({ available: true })
  ```

Step 6: Update Book Availability

- Mark "B002" as available:

```
db.Books.updateOne(
  { book_id: "B002" },
  { $set: { available: true } }
)
```

Step 7: Delete a Book Record

- Remove book with book_id = "B003":

```
db.Books.deleteOne({ book_id: "B003" })
```

An e-commerce startup wants to build a product catalog database to store and manage product information. As a database developer, your task is to use MongoDB to:

- Insert multiple products into the catalog.
- Query specific products using filters.
- Update and delete product information.
- Sort products by price.
- Project only selected fields from the data.
- Limit the number of results.
- Use aggregation to compute category-wise average prices.

Solutions:

Step 1: Install MongoDB

(Same as previous exercises)

Step 2: Start MongoDB Shell

mongosh

Step 3: Create Database and Collection

```
db.createCollection("Products")
```

Step 4: Insert Product Documents

```
db.Products.insertMany([
  {
    product_id: "P001",
    name: "Wireless Mouse",
```

```
    category: "Electronics",
    price: 599,
    stock: 120,
    rating: 4.3
  },
  {
    product_id: "P002",
    name: "Bluetooth Speaker",
    category: "Electronics",
    price: 1299,
    stock: 60,
    rating: 4.6
  },
  {
    product_id: "P003",
    name: "Yoga Mat",
    category: "Fitness",
    price: 799,
    stock: 45,
    rating: 4.1
  },
  {
    product_id: "P004",
    name: "Laptop Stand",
    category: "Accessories",
    price: 999,
    stock: 35,
    rating: 4.0
  }
])
```

Step 5: Basic Queries

- Find all products:

  ```
  db.Products.find().pretty()
  ```

- Find products in Electronics category:

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

Step 6: Update Document

- Increase stock of P003 by 10:

  ```
  db.Products.updateOne(
    { product_id: "P003" },
    { $inc: { stock: 10 } }
  ```

)

Step 7: Delete Document

- Delete product with ID P004:

  db.Products.deleteOne({ product_id: "P004" })

Step 8: Sort Products by Price (Descending)

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

Step 9: Projection (Show only product name and price)

db.Products.find({}, { name: 1, price: 1, _id: 0 })

Step 10: Limit Results (Top 2 by rating)

db.Products.find().sort({ rating: -1 }).limit(2)

Step 11: Aggregation – Average Price per Category

```
db.Products.aggregate([
  {
    $group: {
      _id: "$category",
      average_price: { $avg: "$price" }
    }
  }
])
```

A college wants to digitize student academic records. As a database developer, your task is to use MongoDB to:

- Create a database of student records.
- Perform CRUD operations on the data.
- Sort and filter records based on performance.
- Display selected fields.
- Limit the number of results.
- Use aggregation to compute average marks department-wise.

Solutions:

Step 1: Install MongoDB

(Same as earlier problems.)

Step 2: Start MongoDB Shell

```
mongosh
```

Step 3: Create Database and Collection

```
use CollegeDB
db.createCollection("Students")
```

Step 4: Insert Student Records

```
db.Students.insertMany([
  {
    roll_no: "S001",
    name: "Ananya Joshi",
    department: "Computer",
    semester: 4,
    marks: 85,
    grade: "A"
  },
  {
    roll_no: "S002",
    name: "Raj Mehta",
    department: "Mechanical",
    semester: 4,
    marks: 78,
    grade: "B"
  },
  {
    roll_no: "S003",
    name: "Sneha Patil",
    department: "Computer",
    semester: 4,
    marks: 92,
    grade: "A+"
  },
  {
    roll_no: "S004",
    name: "Ishaan Kulkarni",
    department: "Electronics",
    semester: 4,
    marks: 69,
    grade: "C"
  }
])
```

Step 5: Query Operations

- Show all student records:

```
db.Students.find().pretty()
```

- Show Computer department students:

```
db.Students.find({ department: "Computer" })
```

Step 6: Update Operation

- Update grade of S002 to "B+":

```
db.Students.updateOne(
  { roll_no: "S002" },
  { $set: { grade: "B+" } }
)
```

Step 7: Delete Operation

```
db.Students.deleteOne({ roll_no: "S004" })
```

Step 8: Sorting

- Sort students by marks (descending):

```
db.Students.find().sort({ marks: -1 })
```

Step 9: Projection

- Show only names and marks:

```
db.Students.find({}, { name: 1, marks: 1, _id: 0 })
```

Step 10: Limit

- Show top 2 students:

```
db.Students.find().sort({ marks: -1 }).limit(2)
```

Step 11: Aggregation

- Find average marks per department:

```
db.Students.aggregate([
  {
    $group: {
      _id: "$department",
```

```
      avg_marks: { $avg: "$marks" }
    }
  }
])
```