Practical 1: Mongo DB: Installation and Creation of database and Collection CRUD Document: Insert, Query, Update and Delete Document.

Steps to Install MongoDB

1. **Download MongoDB:**
    - Visit MongoDB Community Server.
    - Select your operating system and download the installer.
2. **Install MongoDB:**
    - Run the installer and select **Complete Installation**.
    - Choose to install MongoDB as a service (recommended).
3. **Set Up Environment Path:**
    - Add MongoDB's `bin` directory (e.g., `C:\Program Files\MongoDB\Server\<version>\bin`) to your system's PATH.
4. **Verify Installation:**
    - Open a terminal and run:
      mongod --version
      mongo --version
5. **Start MongoDB Server:**

Run the MongoDB server:

- mongod

Open a new terminal and connect using:

- mongosh

CRUD Operations in MongoDB with Output

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

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

PS C:\Users\Adarsh Nikam> mongosh
Current Mongosh Log ID: 6789d7469d338d5d35cb0ce1
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.3.8
Using MongoDB:          8.0.4
Using Mongosh:          2.3.8

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

------
   The server generated these startup warnings when booting
   2025-01-17T09:13:29.637+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

test> use myDatabase // Switch to (or create) a database
switched to db myDatabase
myDatabase> db.users.insertOne({
...     name: "Alice",
...     age: 25,
...     email: "alice@example.com"
... })
{
  acknowledged: true,
  insertedId: ObjectId('6789d8339d338d5d35cb0ce2')
}
myDatabase> // Find all documents

myDatabase> db.users.find()
[
  {
    _id: ObjectId('6789d8339d338d5d35cb0ce2'),
    name: 'Alice',
    age: 25,
    email: 'alice@example.com'
  }
]
myDatabase>

myDatabase> // Find with a condition
```

```
myDatabase> db.users.find({ age: { $gte: 18 } })
[
  {
    _id: ObjectId('6789d8339d338d5d35cb0ce2'),
    name: 'Alice',
    age: 25,
    email: 'alice@example.com'
  }
]
myDatabase> // Update one document

myDatabase> db.users.updateOne(
...     { name: "Alice" },
...     { $set: { age: 26 } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
myDatabase>

myDatabase> // Update multiple documents

myDatabase> db.users.updateMany(
...     { age: { $lt: 18 } },
...     { $set: { status: "minor" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
myDatabase> // Delete one document

myDatabase> db.users.deleteOne({ name: "Alice" })
{ acknowledged: true, deletedCount: 1 }
myDatabase>

myDatabase> // Delete multiple documents
```

```
myDatabase> db.users.deleteMany({ age: { $lt: 18 } })
{ acknowledged: true, deletedCount: 0 }
myDatabase> use school // Switch to or create a database
switched to db school
school>

school> db.students.insertMany([
...   { roll_no: 101, name: "Alice", age: 20, grade: "A" },
...   { roll_no: 102, name: "Bob", age: 21, grade: "B" },
...   { roll_no: 103, name: "Charlie", age: 22, grade: "A" },
...   { roll_no: 104, name: "Diana", age: 23, grade: "C" },
...   { roll_no: 105, name: "Eve", age: 20, grade: "B" }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6789d9cd9d338d5d35cb0ce3'),
    '1': ObjectId('6789d9cd9d338d5d35cb0ce4'),
    '2': ObjectId('6789d9cd9d338d5d35cb0ce5'),
    '3': ObjectId('6789d9cd9d338d5d35cb0ce6'),
    '4': ObjectId('6789d9cd9d338d5d35cb0ce7')
  }
}
school> // Find all students

school> db.students.find()
[
  {
    _id: ObjectId('6789d9cd9d338d5d35cb0ce3'),
    roll_no: 101,
    name: 'Alice',
    age: 20,
    grade: 'A'
  },
  {
    _id: ObjectId('6789d9cd9d338d5d35cb0ce4'),
    roll_no: 102,
    name: 'Bob',
    age: 21,
    grade: 'B'
  },
  {
    _id: ObjectId('6789d9cd9d338d5d35cb0ce5'),
```

```
school> // Update the grade of a student with roll_no 105

school> db.students.updateOne(
...   { roll_no: 105 },
...   { $set: { grade: "A" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
school>

school> // Update grades for students younger than 22

school> db.students.updateMany(
...   { age: { $lt: 22 } },
...   { $set: { grade: "B+" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 3,
  modifiedCount: 3,
  upsertedCount: 0
}
school> // Delete a single student by roll number

school> db.students.deleteOne({ roll_no: 101 })
{ acknowledged: true, deletedCount: 1 }
school>

school> // Delete students with grade 'C'

school> db.students.deleteMany({ grade: "C" })
{ acknowledged: true, deletedCount: 1 }
school>
```

Practical 2: Develop a MapReduce program to calculate the frequency of a given word in a given file.

MapReduce is a programming model or pattern within the Hadoop framework that is used to access big data stored in the Hadoop File System (HDFS). It is a core component, integral to the functioning of the Hadoop framework.

```python
import os
from multiprocessing import Pool

# Create a sample file
def create_sample_file(file_path):
    content = """MapReduce is a programming model designed to process and generate large datasets efficiently. It works by splitting the data into smaller chunks, which are then proce
    with open(file_path, 'w') as file:
        file.write(content)

# Mapper function: counts occurrences of each word in a chunk
def mapper(chunk):
    word_count = {}
    for line in chunk.splitlines():
        words = line.split()
        for word in words:
            if word in word_count:
                word_count[word] += 1
            else:
                word_count[word] = 1
    return word_count

# Reducer function: combines dictionaries from all mappers
def reducer(mapped_results):
    final_counts = {}
    for result in mapped_results:
        for word, count in result.items():
            if word in final_counts:
                final_counts[word] += count
            else:
                final_counts[word] = count
    return final_counts
```

```python
# Function to split a file into chunks for parallel processing
def split_file(file_path, num_chunks):
    with open(file_path, 'r') as f:
        content = f.read()
    chunk_size = len(content) // num_chunks
    chunks = [content[i:i + chunk_size] for i in range(0, len(content), chunk_size)]
    return chunks

# Main function
if __name__ == "__main__":
    file_path = "sample.txt"
    create_sample_file(file_path)  # Create a sample file

    num_chunks = 4  # Number of chunks for parallel processing

    # Split the file into chunks
    chunks = split_file(file_path, num_chunks)

    # Use multiprocessing Pool to process chunks in parallel
    with Pool(processes=num_chunks) as pool:
        mapped_results = pool.map(mapper, chunks)

    # Reduce the results
    word_counts = reducer(mapped_results)

    # Print all word counts
    for word, count in word_counts.items():
        print(f"The word '{word}' appears {count} times in the file.")
```

Output:

```
The word 'MapReduce' appears 2 times in the file.
The word 'is' appears 4 times in the file.
The word 'a' appears 2 times in the file.
The word 'programming' appears 1 times in the file.
The word 'model' appears 2 times in the file.
The word 'designed' appears 1 times in the file.
The word 'to' appears 3 times in the file.
The word 'process' appears 2 times in the file.
The word 'and' appears 6 times in the file.
The word 'generate' appears 1 times in the file.
The word 'large' appears 2 times in the file.
The word 'datasets' appears 1 times in the file.
The word 'efficiently.' appears 1 times in the file.
The word 'It' appears 3 times in the file.
The word 'works' appears 1 times in the file.
The word 'by' appears 2 times in the file.
The word 'splitting' appears 1 times in the file.
The word 'the' appears 5 times in the file.
The word 'data' appears 3 times in the file.
The word 'into' appears 3 times in the file.
The word 'smaller' appears 2 times in the file.
The word 'chunks,' appears 1 times in the file.
The word 'which' appears 1 times in the file.
The word 'are' appears 3 times in the file.
The word 'then' appears 2 times in the file.
The word 'processed' appears 1 times in the file.
The word 'in' appears 2 times in the file.
The word 'parallel' appears 1 times in the file.
The word 'across' appears 1 times in the file.
The word 'multiple' appears 1 times in the file.
The word 'machines.' appears 1 times in the file.
The word 'The' appears 1 times in the file.
```