Labsheet -09

1. Create Database & Collection

```
Use College DB
db. Create Collection ("Students")
db. Students . insert many ( [
   {name :'Alice', Course: 'MCA', year : 1, Marks : 85},
   { name : 'Bob', Course : 'MBA' , year:2, Marks : 65},
   {name : 'Charlie', Course: 'MCA', year : 2, Marks : 92},
    { name : 'David', Course : 'BCA', Year: 3, Marks : 78},
    {name : 'Eva', Course : 'MCA', Year:1, Marks : 55}
  ])
```

2. Querying in Mongo DB

(a) Retrive all MCA Student

```
db. Students find ({Course: 'MCA'})
```

(b) Student with Marks>75

```
db. Students find ({Marks: {$ gt : 75}})
```

(C) Update a Student's Marks

```
db . Student . update one (
   {name : 'Eva'},
   {$ Set: {marks: 68}}
  }
```

(d) Delete a record by name

```
db. Student deleteone ({name: 'Bob'})
```

3. Redis - Key - Value Store -

(a) Set & Retrive Key - value Pairs

```
SET Student 1 "Alice"
GET Student 2
```

(b) Update & delete key.

```
SET Student 3 "Alice Updated"
DEL Student 1
```

(C) Use Hash to Store Student Details

```
HSET Student : 1 id 1 name 'Alice' marks 90
HGET ALL Student : 1
```

4. Cassandra - Column - Family Store

Create keyspace & Table.

```
CREATE KEYSPACE COMPANY DB
WITH replication = {'Class': Sir ple Strategy', 'replication.factor:2}:
 Use Company DB
  CREATE TABLE Employees (
        emp-id INT Primary key,
        emp-name TEXT,
        department TEXT,
        Salary DECIMAL.
   ):
```

- Insert & Retrive Data

```
INSERT INTO Employees (emp-id, emp-name, department, Salary)
 VALUES (1, 'John', 'HR', 5000);
INSERT INTO Employees (emp-id, emp-name, depatment, Salary)
 VALUES (2, 'Mary', 'BIT', 7000);
INSERT INTO Employees ('emp-id, emp-name, department, Salary)
 VALUES( 3, 'Steve', 'Finance', 6000);
 SELECT * FROM Employees :
```

5. Neo 45 Graph Database

- Create Nodes & Relationships

Open Neo4j Browser → Run cypher quaries

```
CREATE (a: Person {name: 'Alice'})
CREATE (b: Person {name: 'Bob'})
CREATE (C: Person {name: 'Charlie'})
CREATE (a) -[: friends - with] → (b)
CREATE (b) - [: Friends - with] → (c)
```

Retrive All Friend of Alice

```
MATCH (a: Person: {name: 'Alice'}) - [: Friends - with] →
                                                  (friend)

Return friend. name .
```

6. Comparison Experiment : Mongo DB VS Redis

Features	Mongo DB (Document Model)	Redis (key-value Model)
• Data Format	JSON-like documents	Simple key-value Pairs
• Structure	Flexible schema	key-value (String, hash)
• Query	d.b Student. find ({ marks: ({ $gt: 75}})	GET Student : 1
• Speed	Slower for single lookup	Extremely Fast for Small data
• Use Case	Complax queries, flexible document	Caching, sessions, quick lookups.

7. Case Identification

Example : Social Media Application

Use case : Users, Post, likes, Comments, & Relationships

Best Model : Graph Database (Neo4j)

8. Handling Unstructured Data (MongoDB)

Example JSON Document :

```
db.Products . Insert one ( {
 Product_id : 1,
  name : 'Smart phone',
  brand : 'Techx',
  Prince : 500000,
  Specifications : {
   RAM : '8GB'
   Storage : '128 GB'
   Battery : '5000 mah"
 }
})
```

8/10