**NoSQL Programs**

**PROGRAM: 6, 7,**

**Question:6 - Create a NoSQL Database Schema for a Student-Details Scenario**

**Aim:**

Creating NoSQL Schema for Student-Details Scenario using MongoDB.

**Algorithm:**

1. Start the program.
2. Open Command Prompt and type **‘mongo’** to start mongo shell.
3. Create a database using the command **“use studentdb”.**
4. Create collections using **‘createCollection()’** command.
5. Add the collection name and use **validator{}** to pass your schema.  
   **createCollection(“customerSales”, validator :{//schema here})**
6. Stop the program.

**PROGRAM:**

> use studentdb

//switched to db studentdb

> db.createCollection("studentInfo", {

    validator: {

       $jsonSchema: {

          bsonType: "object",

          required: [ "student", "department", "address" ],

          properties: {

            customer: {

                bsonType: "object",

                required: [ "student\_id", "student\_name", "student\_email", "student\_phone" ],

                properties: {

                   student\_id: {

                      bsonType: "string",

                      description: "must be a string and required"

                   },

                   student\_name: {

                      bsonType: "string",

                      description: "must be a string and is required"

                   },

                   student\_email: {

                     bsonType: "string",

                     description: "must be a string and is required"

                  },

                  student\_phone: {

                     bsonType: "string",

                     description: "must be a string and is required"

                  }

                }

             },

             department: {

                bsonType: "object",

                required: [ "dept\_name", "program\_name" ],

                properties: {

                   dept\_name: {

                      bsonType: "string",

                      description: "must be a string and is required"

                   },

                   program\_name: {

                      bsonType: "double",

                      description: "must be a double and is required"

                   }

                }

             },

             address: {

                bsonType: "object",

                required: [ "state", "city" ],

                properties: {

                   state: {

                      bsonType: "string",

                      description: "must be a string and is required"

                   },

                   city: {

                      bsonType: "string",

                      description: "must be a string and is required"

                   }

                }

             }

          }

       }

    }

 })

//{"ok" : 1}

> db.customerSale.insert( {

     student:{

        student\_id: "",

        student\_name: "",

        student\_email: "",

        student\_phone: ""

     },

     department:{

        dept\_name: "",

        program\_name: "",

     },

     address:{

        state:"",

        city:""

     }

 })

 //WriteResult({ "nInserted" : 1 })

//Result: Our schema for student scenario is tested and working successfully.

**Question 7. Create a NoSQL Database to perform sorting, indexing and limiting of records.**

**Aim:**

Creating and writing NoSQL queries to retrieve information from the database using MongoDB.

**Algorithim:**

1. Start the program.
2. Open command prompt and type **“mongo”** to start mongo shell.
3. Create a database using the command (“ **use companydb**”)
4. Create collection( “**empDetails**”)
5. Insert documents into your created collections **(empDetails**”)
6. Write the queries to perform **Indexing, Sorting** and **Limiting** of the records.
7. Stop the program.

PROGRAM:

//create the database

> use companydb

//create collection

db.createCollection("empDetails")

//insert some documents into your created collection

> db.empDetails.insertMany(

    [

        {

            First\_Name: "Radhika",

            Last\_Name: "Sharma",

            Date\_Of\_Birth: "1995-09-26",

            e\_mail: "radhika\_sharma.123@gmail.com",

            phone: "7034587542"

        },

        {

            First\_Name: "Rachel",

            Last\_Name: "Christopher",

            Date\_Of\_Birth: "1990-02-16",

            e\_mail: "Rachel\_Christopher.123@gmail.com",

            phone: "8543210009"

        },

        {

            First\_Name: "Fathima",

            Last\_Name: "Sheik",

            Date\_Of\_Birth: "1990-02-16",

            e\_mail: "Fathima\_Sheik.123@gmail.com",

            phone: "8702805432"

        },

        {

            First\_Name: "Daniel",

            Last\_Name: "Wellbeck",

            Date\_of\_Birth: "1996-10-21",

            e\_mail: "DanielWellbeck.123@gmail.com",

            phone: "9256900321"

        }

    ]

)

//WRITING QUERIES TO PERFORM SORTING, INDEXING AND LIMITING OF RECORDS

db.empDetails.find().sort({"First\_Name":1})

db.empDetails.find().sort({"First\_Name":-1})

db.empDetails.find({},{"First\_Name":1,\_id:0}).sort({"First\_Name":-1})

db.empDetails.find().limit(4)

db.empDetails.find().sort({"First\_Name":1}).limit(1).skip(1)

db.empDetails.find({},{"First\_Name":1,\_id:0}).limit(1)

db.empDetails.createIndex({"e\_mail":1})

db.empDetails.getIndexes()

db.empDetails.dropIndex({"e\_mail":1})