**Step 1: Create the DYPIT Database**

Code:

use DYPIT

* **Explanation**: This command switches to the DYPIT database, and it will be created if it doesn’t exist.

**Step 2: Create Teachers and Students Collections and Insert Sample Data**

1. **Insert sample data into Teachers collection**:

Code:

db.Teachers.insertMany([

{ Tname: "Praveen", dno: 1, dname: "COMP", experience: "8 years", salary: 20000, date\_of\_joining: new Date("2015-06-20") },

{ Tname: "Rajesh", dno: 2, dname: "IT", experience: "5 years", salary: 18000, date\_of\_joining: new Date("2018-07-15") },

{ Tname: "Suresh", dno: 3, dname: "E&TC", experience: "6 years", salary: 15000, date\_of\_joining: new Date("2017-03-10") },

{ Tname: "Anita", dno: 1, dname: "COMP", experience: "4 years", salary: 22000, date\_of\_joining: new Date("2019-08-01") },

{ Tname: "Anjali", dno: 2, dname: "IT", experience: "3 years", salary: 16000, date\_of\_joining: new Date("2021-01-25") }

])

1. **Insert sample data into Students collection**:

Code:

db.Students.insertMany([

{ Sname: "John", roll\_no: 1, class: "FY" },

{ Sname: "xyz", roll\_no: 2, class: "SY" },

{ Sname: "Alice", roll\_no: 3, class: "TY" }

])

**Step 3: Queries for Various Tasks**

1. **Display the department-wise average salary**:

Code:

db.Teachers.aggregate([

{ $group: { \_id: "$dname", avgSalary: { $avg: "$salary" } } }

])

* + **Explanation**: This query groups the data by department name (dname) and calculates the average salary.

1. **Display the number of employees working in each department**:

Code:

db.Teachers.aggregate([

{ $group: { \_id: "$dname", employeeCount: { $sum: 1 } } }

])

* + **Explanation**: This query counts the number of employees in each department using $sum: 1.

1. **Display the department-wise total salary of departments having total salary >= 50,000**:

Code:

db.Teachers.aggregate([

{ $group: { \_id: "$dname", totalSalary: { $sum: "$salary" } } },

{ $match: { totalSalary: { $gte: 50000 } } }

])

* + **Explanation**: This query groups the data by department name, sums the salaries, and filters departments where the total salary is greater than or equal to 50,000.

**Step 4: Queries Using Operators (max, min, etc.)**

1. **Find the maximum and minimum salary in the Teachers collection**:

code:

db.Teachers.aggregate([

{ $group: { \_id: null, maxSalary: { $max: "$salary" }, minSalary: { $min: "$salary" } } }

])

* + **Explanation**: The $max and $min operators are used to find the maximum and minimum salary, respectively.

1. **Find the total experience of all teachers in years**:

Code:

db.Teachers.aggregate([

{

$project: {

experienceInYears: { $toInt: { $arrayElemAt: [ { $split: ["$experience", " "] }, 0 ] } }

}

},

{ $group: { \_id: null, totalExperience: { $sum: "$experienceInYears" } } }

])

* + **Explanation**: This query extracts the numeric experience from the string and sums it.

**Step 5: Creating Unique Index**

1. **Create a unique index on the roll\_no field in Students collection**:

Code:

db.Students.createIndex({ roll\_no: 1 }, { unique: true })

* + **Explanation**: This creates a unique index on roll\_no, ensuring no duplicate roll numbers are allowed.

**Step 6: Creating a Compound Index**

1. **Create a compound index on dname and salary fields in Teachers collection**:

Code:

db.Teachers.createIndex({ dname: 1, salary: -1 })

* + **Explanation**: This creates a compound index on dname (ascending order) and salary (descending order) to optimize queries that sort or filter by these fields.

**Step 7: Show All Indexes Created in DYPIT Database**

1. **Show all indexes created in Teachers collection**:

Code:

db.Teachers.getIndexes()

1. **Show all indexes created in Students collection**:

Code:

db.Students.getIndexes()

**Running the Commands**

* **Start MongoDB**: Ensure your MongoDB server is running using mongod.
* **Open MongoDB Shell**: Use mongo to open the shell and run these commands.
* **Verify Indexes**: Use getIndexes() to check that the indexes are created correctly.

These steps cover creating the database, inserting data, running aggregation queries, creating unique and compound indexes, and displaying all created indexes.

Top of Form

Bottom of Form