**Step 1: Create the Database DYPIT**

Code:

use DYPIT

* **Explanation**: This command switches to the DYPIT database. If it doesn't exist, it will be created.

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

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

Code:

db.Teachers.insertMany([

{ Tname: "Praveen", dno: 1, dname: "COMP", experience: "8 years", salary: 12000, date\_of\_joining: new Date("2015-06-20") },

{ Tname: "Rajesh", dno: 2, dname: "IT", experience: "5 years", salary: 15000, date\_of\_joining: new Date("2018-07-15") },

{ Tname: "Suresh", dno: 3, dname: "E&TC", experience: "6 years", salary: 9000, date\_of\_joining: new Date("2017-03-10") },

{ Tname: "Anita", dno: 4, dname: "COMP", experience: "4 years", salary: 11000, date\_of\_joining: new Date("2019-08-01") }

])

1. **Insert sample data into the 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 to Fetch and Manipulate Data**

1. **Find the information about all teachers**:

Code:

db.Teachers.find().pretty()

1. **Find the average salary of teachers in the Computer department**:

Code:

db.Teachers.aggregate([

{ $match: { dname: "COMP" } },

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

])

* + **Explanation**: This aggregation query filters teachers in the "COMP" department and calculates the average salary.

1. **Find the minimum and maximum salary of E&TC department teachers**:

Code:

db.Teachers.aggregate([

{ $match: { dname: "E&TC" } },

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

])

1. **Find teachers from Computer, IT, and E&TC departments with a salary ≥ 10,000**:

Code:

db.Teachers.find({

dname: { $in: ["COMP", "IT", "E&TC"] },

salary: { $gte: 10000 }

}).pretty()

1. **Find student information with roll\_no = 2 or Sname = "xyz"**:

Code:

db.Students.find({

$or: [{ roll\_no: 2 }, { Sname: "xyz" }]

}).pretty()

**Step 4: Updating Documents**

1. **Update the experience of teacher "Praveen" to 10 years (or insert if not present)**:

Code:

db.Teachers.updateOne(

{ Tname: "Praveen" },

{ $set: { experience: "10 years" } },

{ upsert: true }

)

1. **Update the department of all teachers working in the IT department to COMP**:

Code:

db.Teachers.updateMany(

{ dname: "IT" },

{ $set: { dname: "COMP" } }

)

**Step 5: Projection and Using save() Method**

1. **Find teachers' names and their experience**:

Code:

db.Teachers.find({}, { Tname: 1, experience: 1, \_id: 0 }).pretty()

1. **Insert one entry into the Teachers collection using the save() method**:

Code:

db.Teachers.save({

Tname: "NewTeacher", dno: 5, dname: "Math", experience: "3 years", salary: 8000, date\_of\_joining: new Date("2022-09-10")

})

**Step 6: Aggregating Data**

1. **Find the total salary of all teachers**:

Code:

db.Teachers.aggregate([

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

])

**How to Run and Check Output**

1. **Start MongoDB Server**:
   * Make sure your MongoDB server is running. Use mongod to start it if needed.
2. **Open MongoDB Shell**:
   * Use mongo in a new terminal to open the MongoDB shell.
3. **Run each command step by step**:
   * Use .find().pretty() to display documents in a well-formatted way.
   * Use aggregate() for calculations like average, min, max, and total salary.
4. **Verify the Output**:
   * Check the output after each command to ensure it works as expected. Adjust your queries as needed based on your data.