**Task 1: CRUD Operations**

* **Create Database and Collections:**
* 1)Create a new MongoDB database named company.

Use company

* 2)Within the company database, create three collections: employees, financial\_details, and projects.
  + - db.createCollection('employees')
    - db.createCollection('financial\_details')
    - db.createCollection(‘projects')

**Insert Data:**

db.employees.insertMany([{ name: "John Doe", age: 30, position: "Manager", department: "HR" },

{ name: "Jane Smith", age: 25, position: "Developer", department: "Engineering" },

{ name: "Michael Johnson", age: 35, position: "Analyst", department: "Finance" },

{ name: "Emily Brown", age: 28, position: "Designer", department: "Marketing" },

{ name: "David Lee", age: 32, position: "Engineer", department: "Engineering" },

{ name: "Sarah Johnson", age: 29, position: "Marketing Manager", department: "Marketing" },

{ name: "Alex Rodriguez", age: 40, position: "Senior Developer", department: "Engineering" },

{ name: "Michelle White", age: 33, position: "Financial Analyst", department: "Finance" },

{ name: "Chris Evans", age: 27, position: "Graphic Designer", department: "Marketing" },

{ name: "Jessica Brown", age: 31, position: "Software Engineer", department: "Engineering" }]);

db.financial\_details.insertMany([

{ employee\_id: ObjectId(), salary: 60000, annual\_income: 72000, expenses: 10000 },

{ employee\_id: ObjectId(), salary: 70000, annual\_income: 84000, expenses: 12000 },

{ employee\_id: ObjectId(), salary: 80000, annual\_income: 96000, expenses: 15000 },

{ employee\_id: ObjectId(), salary: 55000, annual\_income: 66000, expenses: 8000 },

{ employee\_id: ObjectId(), salary: 65000, annual\_income: 78000, expenses: 11000 },

{ employee\_id: ObjectId(), salary: 75000, annual\_income: 90000, expenses: 13000 },

{ employee\_id: ObjectId(), salary: 85000, annual\_income: 102000, expenses: 16000 },

{ employee\_id: ObjectId(), salary: 72000, annual\_income: 86400, expenses: 14000 },

{ employee\_id: ObjectId(), salary: 78000, annual\_income: 93600, expenses: 17000 },

{ employee\_id: ObjectId(), salary: 68000, annual\_income: 81600, expenses: 13000 }

]);

db.projects.insertMany([

{ project\_name: "Website Redesign", employee\_id: ObjectId(), status: "In Progress" },

{ project\_name: "Financial Analysis Tool", employee\_id: ObjectId(), status: "Completed" },

{ project\_name: "SEO Optimization", employee\_id: ObjectId(), status: "Pending" },

{ project\_name: "Product Enhancement", employee\_id: ObjectId(), status: "In Progress" },

{ project\_name: "Market Research", employee\_id: ObjectId(), status: "Completed" },

{ project\_name: "Client Onboarding Portal", employee\_id: ObjectId(), status: "In Progress" },

{ project\_name: "Expense Tracking System", employee\_id: ObjectId(), status: "Completed" },

{ project\_name: "Customer Satisfaction Survey", employee\_id: ObjectId(), status: "Pending" },

{ project\_name: "Inventory Management App", employee\_id: ObjectId(), status: "In Progress" },

{ project\_name: "Training Program Development", employee\_id: ObjectId(), status: "Pending" }

]);

**Read Operation:**

* Write a query to retrieve all employees who are above 30 years old from the employees collection.
  + - db.employees.find({ age: { $gt: 30 } })

**Update Operation:**

* Choose an employee and update their position to a higher role in the employees collection.

db.employees.updateOne( { name: "John Doe" },

{ $set: { position: "Senior Manager" } })

**Delete Operation:**

* Delete a project from the projects collection based on the project name.
  + - db.projects.deleteOne({ project\_name: "Website Redesign" })

**Task 2: Aggregation**

* **Aggregation Pipeline:**
* Create an aggregation pipeline that calculates the average salary for employees in each position from the financial\_details collection.
* [
* { $lookup: {
* from: "employees",
* localField: "employee\_id",
* foreignField: "employee\_id",
* as: "employee"
* }},
* { $unwind: "$employee" },
* { $group: {
* \_id: "$employee.position",
* averageSalary: { $avg: "$salary" }
* }}
* ]

**Group and Count:**

* Write an aggregation query to group financial details by the employee's age from the financial\_details collection and count the number of employees in each age group.
* [
* { $lookup: {
* from: "employees",
* localField: "employee\_id",
* foreignField: "\_id",
* as: "employee"
* }},
* { $unwind: "$employee" },
* { $group: {
* \_id: "$employee.age",
* employeeCount: { $sum: 1 }
* }}
* ]
* **Sorting:**
* Sort the projects based on their status in descending order from the projects collection.
  + - [ { $sort: { status: -1 } }]

**Projecting Fields:**

* Create an aggregation query to project only the name and position fields for employees who have an annual income greater than a specified value from the employees and financial\_details collections.

[

* {
* $lookup: {
* from: "financial\_details",
* localField: "employee\_id",
* foreignField: "employee\_id",
* as: "financialDetails"
* }
* },
* {
* $match: {
* "financialDetails.annual\_income": { $gt: 50000 }
* }
* },
* {
* $project: {
* name: 1,
* position: 1
* }
* }
* ]

Task 3: Advance Aggregation (Use all below operation in single query)

* Lookup and Project:
* Use $lookup to join the employees collection with the projects collection based on the employee\_id. Project only the relevant fields from the projects collection.
* Group and Sum:
* Group the result from Task 1 by position and calculate the total number of projects each position is involved in using $group and $sum. Project the position and total projects.
* AddFields and Map:
* Use $addFields to create a new field named project\_names in the result from Task 2. This field should be an array containing the names of all projects for each position. You can use $map to iterate over the projects and extract their names.
* Reduce Operation:
* Implement a $reduce operation to calculate the total annual income of all employees across positions from the financial\_details collection. Project the result.
* Filter and Project:
* Create a query to filter out employees with annual expenses greater than a specified value from the financial\_details collection. Project only the necessary fields.

var specifiedValue = 50000; // Specify the value for annual expenses

[

{

$lookup: {

from: "projects",

localField: "employee\_id",

foreignField: "employee\_id",

as: "projects"

}

},

{

$project: {

position: 1,

projects: "$projects.project\_name"

}

},

{

$group: {

\_id: "$position",

total\_projects: { $sum: { $size: "$projects" } }

}

},

{

$addFields: {

project\_names: {

$map: {

input: "$projects",

as: "project",

in: "$$project"

}

}

}

},

{

$lookup: {

from: "financial\_details",

localField: "employee\_id",

foreignField: "employee\_id",

as: "financialDetails"

}

},

{

$project: {

total\_income: {

$reduce: {

input: "$financialDetails",

initialValue: 0,

in: { $sum: ["$$value", "$$this.annual\_income"] }

}

}

}

},

{

$match: {

"financialDetails.expenses": { $lt: 40000 }

}

},

{

$project: {

position: 1,

total\_projects: 1,

project\_names: 1,

total\_income: 1

}

}

]