# st124738-Sonakul Kamnuanchai

**Transactions**

1.View project details description

db.projects.find({}, { \_id: 0 });

2. View task of project with detailed description.

db.projects.aggregate([

{

$match: { "mysql\_id": 20 }

},

{

$unwind: "$tasks\_id"

},

{

$lookup: {

from: "tasks",

localField: "tasks\_id",

foreignField: "\_id",

as: "task\_details"

}

},

{

$unwind: "$task\_details"

},

{

$project: {

task\_id: 1,

task\_name: "$task\_details.name",

severity: "$task\_details.severity",

estimated\_time: "$task\_details.estimated\_time",

status: "$task\_details.status",

created\_at: "$task\_details.created\_at",

updated\_at: "$task\_details.updated\_at"

}

}

]);

3. Change status on tasks (eg. In Progress, Completed).

db.tasks.updateMany(

{ "status": "Pending" },

{

$set: { "status": "Completed" }

}

);

**Queries**

1.The project manager wants to see team members based on performance by role.

db.employees.aggregate([

{

$match: {

"role.name": { $in: ["Frontend\_developer", "Backend\_developer", "Business\_analyst"] }

}

},

{

$sort: { "KPI.employee\_rating": -1 }

},

{

$project: {

\_id: 0,

employee\_name: { $concat: ["$first\_name", " ", "$last\_name"] },

employee\_rating: "$KPI.employee\_rating",

role\_name: "$role.name",

tasks\_completed: "$KPI.tasks\_completed",

tasks\_assigned: "$KPI.tasks\_assigned"

}

}

]);

2.The project manager wants to see top clients based on project count.

db.projects.aggregate([

{

$group: {

\_id: "$client\_id",

project\_count: { $sum: 1 }

}

},

{

$lookup: {

from: "clients",

localField: "\_id",

foreignField: "client\_id",

as: "client\_details"

}

},

{

$unwind: "$client\_details"

},

{

$project: {

client\_name: "$client\_details.name",

project\_count: 1

}

},

{

$sort: { project\_count: -1 }

}

]);

3.The project manager wants to see who is assigned to multiple projects.

db.tasks.aggregate([

{

$unwind: "$assignments"

},

{

$group: {

\_id: "$assignments.employee\_id",

projects\_assigned: { $addToSet: "$\_id" }

}

},

{

$match: {

"projects\_assigned.1": { $exists: true }

}

},

{

$lookup: {

from: "employees",

localField: "\_id",

foreignField: "\_id",

as: "employee\_details"

}

},

{

$unwind: "$employee\_details"

},

{

$project: {

\_id: 0,

employee\_name: {

$concat: ["$employee\_details.first\_name", " ", "$employee\_details.last\_name"]

},

project\_count: { $size: "$projects\_assigned" },

role\_name: "$employee\_details.role.name"

}

}

]);

4.The project manager wants to analyze where the project was cancelled and evaluate which employees performed badly.

db.projects.aggregate([

{ $match: { status: "Cancelled" } },

{

$match: {

$expr: {

$gt: ["$updated\_at", "$created\_at"]

}

}

},

{ $unwind: "$tasks\_id" },

{

$lookup: {

from: "tasks",

localField: "tasks\_id",

foreignField: "\_id",

as: "task\_details"

}

},

{ $unwind: "$task\_details" },

{ $unwind: "$task\_details.assignments"},

{

$lookup: {

from: "employees",

localField: "task\_details.assignments.employee\_id",

foreignField: "\_id",

as: "employee\_details"

}

},

{ $unwind: "$employee\_details" },

{

$project: {

project\_name: "$name",

employee\_name: {

$concat: ["$employee\_details.first\_name", " ", "$employee\_details.last\_name"]

},

completion\_percentage: "$task\_details.assignments.completion\_percentage",

role\_name: "$employee\_details.role.name",

\_id: 0

}

},

{ $sort: { completion\_percentage: 1 } } ]);

5. The project manager wants to find employees in the role who are available and have the highest ratings, which can help with projects that are canceled.

db.employees.aggregate([

{

$match: {

"role.name": { $in: ["Backend\_developer", "Frontend\_developer", "Business\_analyst"] }

}

},

{

$sort: {"KPI.employee\_rating": -1 }

},

{

$group: {

\_id: "$role.name",

employee\_name: { $first: { $concat: ["$first\_name", " ", "$last\_name"] } },

employee\_rating: { $first: "$KPI.employee\_rating" },

role\_name: { $first: "$role.name" },

}

},

{

$project: {

\_id: 0,

role\_name: 1,

employee\_name: 1,

employee\_rating: 1,

}

}

]);

# st124784-Phue Pwint Thwe

1. View team members with certain Role (eg. PM, Developer)

db.employees.find({ "role.name": "Project Manager" })

2. View Task Progress and Assigned Team Members

db.tasks.aggregate([

{ $unwind: "$assignments" },

{

$lookup: {

from: "employees",

localField: "assignments.employee\_id",

foreignField: "\_id",

as: "Assigned\_Employee"

}

},

{ $unwind: "$Assigned\_Employee" },

{

$project: {

\_id: 1,

name: 1,

status: 1,

"assignments.completion\_percentage": 1,

"assignments.approval\_status": 1,

"Assigned\_Employee.first\_name": 1,

"Assigned\_Employee.last\_name": 1

}

}

])

3. View Feedbacks from Project Manager.

db.tasks.aggregate([

{ $unwind: "$assignments" },

{ $unwind: "$assignments.feedback" },

{

$lookup: {

from: "employees",

localField: "assignments.feedback.mysql\_id",

foreignField: "mysql\_id",

as: "PM\_Details"

}

},

{ $unwind: "$PM\_Details" },

{

$match: { "PM\_Details.role.name": "Project Manager" }

},

{

$project: {

\_id: 1,

name: 1,

"assignments.feedback.\_id": 1,

"assignments.feedback.date": 1,

"assignments.feedback.comments": 1,

"PM\_Details.first\_name": 1,

"PM\_Details.last\_name": 1,

"PM\_Details.email": 1Project Manager Checks Workload for Project Members.

}

}

])

1. Project Manager Checks Workload for Project Members.

db.tasks.aggregate([

{ $unwind: "$assignments" },

{

$lookup: {

from: "employees",

localField: "assignments.employee\_id",

foreignField: "\_id",

as: "Employee\_Details"

}

},

{ $unwind: "$Employee\_Details" },

{

$lookup: {

from: "employees",

localField: "assignments.feedback.mysql\_id",

foreignField: "mysql\_id",

as: "PM\_Details"

}

},

{ $unwind: "$PM\_Details" },

{ $match: { "PM\_Details.role.name": "Project Manager" } },

{

$group: {

\_id: "$assignments.employee\_id",

employee\_name: { $first: "$Employee\_Details.first\_name" },

employee\_lastname: { $first: "$Employee\_Details.last\_name" },

total\_assigned\_tasks: { $sum: 1 },

total\_estimated\_time: { $sum: "$estimated\_time" },

total\_completion: { $avg: "$assignments.completion\_percentage" }

}

},

{ $sort: { total\_assigned\_tasks: -1 } }

])

2. Check Team Member Workload Based on Task Priority and Estimated Time.

db.tasks.aggregate([

{ $unwind: "$assignments" },

{

$lookup: {

from: "employees",

localField: "assignments.employee\_id",

foreignField: "\_id",

as: "Employee\_Details"

}

},

{ $unwind: "$Employee\_Details" },

{

$group: {

\_id: "$assignments.employee\_id",

employee\_name: { $first: "$Employee\_Details.first\_name" },

employee\_lastname: { $first: "$Employee\_Details.last\_name" },

total\_tasks\_assigned: { $sum: 1 },

total\_estimated\_time: { $sum: "$estimated\_time" },

high\_priority\_tasks: {

$sum: { $cond: [{ $eq: ["$severity", "High"] }, 1, 0] }

},

medium\_priority\_tasks: {

$sum: { $cond: [{ $eq: ["$severity", "Medium"] }, 1, 0] }

},

low\_priority\_tasks: {

$sum: { $cond: [{ $eq: ["$severity", "Low"] }, 1, 0] }

}

}

},

{ $sort: { total\_estimated\_time: -1 } }

])

3. Identify Team Member Roles with Highest or Lowest Performance.

db.employees.aggregate([

{

$lookup: {

from: "tasks",

localField: "\_id",

foreignField: "assignments.employee\_id",

as: "Task\_Details"

}

},

{ $unwind: "$Task\_Details" },

{ $unwind: "$Task\_Details.assignments" },

{

$match: {

"$expr": { "$eq": ["$Task\_Details.assignments.employee\_id", "$\_id"] }

}

},

{

$group: {

\_id: "$role.name",

Total\_Employees: { $sum: 1 },

Avg\_Employee\_Rating: { $avg: "$KPI.employee\_rating" },

Total\_Tasks\_Completed: {

$sum: { $cond: [{ $eq: ["$Task\_Details.assignments.completion\_percentage", 100] }, 1, 0] }

},

Total\_Tasks\_Assigned: { $sum: 1 }

}

},

{ $sort: { Avg\_Employee\_Rating: -1 } }

])

4. Identify Team Member Roles with Highest or Lowest Performance.

db.projects.aggregate([

{

$match: { status: "Ongoing" } // Filter only ongoing projects

},

{

$project: {

\_id: 1,

name: 1,

total\_tasks: { $size: "$tasks\_id" } // Count the number of tasks

}

},

{

$sort: { total\_tasks: -1 } // Sort by highest task count

}

])

5. Project manager checks the planned projects quarterly.

db.projects.aggregate([

{

$addFields: {

end\_date: { $dateFromString: { dateString: "$summary.end\_date" } },

quarter: { $ceil: { $divide: [{ $month: { $dateFromString: { dateString: "$summary.end\_date" } } }, 3] } },

year: { $year: { $dateFromString: { dateString: "$summary.end\_date" } } }

}

},

{

$match: {

status: "Planned",

year: { $gte: 2025 } // Match SQL filtering

}

},

{

$group: {

\_id: { year: "$year", quarter: "$quarter" },

Planned\_Projects: { $sum: 1 }

}

},

{ $sort: { "\_id.year": 1, "\_id.quarter": 1 } }

])

# st125166-Soe Htet Naing

-- 3 Transactions

-- insert/ view / update status of a certain project ID

db.projects.insertOne({

"\_id": ObjectId(),

"mysql\_id": 153,

"name": "New AI Development",

"status": "Ongoing",

"pm\_id": ObjectId("67cfc75e709c6ef6ee9ab34a"),

"summary": {

"description": "Developing an AI-based recommendation system",

"budget": 50000,

"start\_date": ISODate("2024-03-01T00:00:00Z"),

"end\_date": ISODate("2024-12-31T00:00:00Z")

},

"tasks\_id": [],

"created\_at": ISODate("2024-03-11T00:00:00Z"),

"updated\_at": ISODate("2024-03-11T00:00:00Z")

})

db.projects.findOne({ "\_id": ObjectId("67cfdb7e415076eb51418418") })

db.projects.updateOne(

{ "\_id": ObjectId("67cfdb7e415076eb51418418") },

{

"$set": {

"status": "Completed",

"updated\_at": ISODate("2024-03-11T00:00:00Z")

}

}

)

-- insert/ View/Update status of a certain project’s task

db.tasks.insertOne({

"\_id": ObjectId(),

"mysql\_id": 1001,

"name": "Develop Backend API",

"severity": "High",

"status": "In Progress",

"estimated\_time": 120,

"assignments": [

{

"\_id": ObjectId(),

"mysql\_id": 1001,

"employee\_id": ObjectId("67cfc75e709c6ef6ee9ab364"),

"assignment\_date": ISODate("2024-03-05T00:00:00Z"),

"completion\_percentage": 50,

"approval\_status": "Pending",

"feedback": [],

"time\_logs": []

}

],

"created\_at": ISODate("2024-03-11T00:00:00Z"),

"updated\_at": ISODate("2024-03-11T00:00:00Z")

})

db.tasks.findOne({ "\_id": ObjectId("67cfdef6415076eb51418419") })

db.tasks.updateOne(

{ "\_id": ObjectId("67cfdef6415076eb5141841a") },

{

"$set": {

"status": "Completed",

"updated\_at": ISODate("2024-03-11T00:00:00Z")

}

}

)

-- Insert/Update/View Team members information of project.

db.project\_teams.updateOne(

{ "project\_id": ObjectId("67cfdb7e415076eb51418418") },

{

"$push": {

"team\_members": {

"employee\_id": ObjectId("67cfc75e709c6ef6ee9ab35c"),

"end\_date": ISODate("2024-09-30T00:00:00Z")

}

}

},

db.project\_teams.find(

{ "project\_id": ObjectId("67cfdb7e415076eb51418418") },

{ "team\_members": 1, "\_id": 0 }

)

db.project\_teams.updateOne(

{

"project\_id": ObjectId("67cfdb7e415076eb51418418"),

"team\_members.employee\_id": ObjectId("67cfc75e709c6ef6ee9ab35c")

},

{

"$set": {

"team\_members.$.assigned\_date": ISODate("2024-03-15T00:00:00Z"),

"team\_members.$.end\_date": ISODate("2024-10-01T00:00:00Z")

}

}

)

-- 6 reports

-- Query average time taken for each project to complete tasks

db.projects.aggregate([

{

$match: { status: "Completed" } // Filter only completed projects

},

{

$lookup: {

from: "tasks",

localField: "tasks\_id",

foreignField: "\_id",

as: "tasks"

}

},

{

$unwind: "$tasks" // Flatten the tasks array

},

{

$unwind: "$tasks.assignments" // Flatten the assignments array

},

{

$unwind: "$tasks.assignments.time\_logs" // Flatten the time\_logs array

},

{

$group: {

\_id: "$\_id",

project\_name: { $first: "$name" },

total\_tasks: { $sum: 1 },

avg\_actual\_time: { $avg: "$tasks.assignments.time\_logs.actual\_time" }

}

},

{

$sort: { avg\_actual\_time: -1 } // Sort by highest average time taken

}

]);

-- Query activity log for each project by certain project manager.

db.projects.aggregate([

{

$lookup: {

from: "employees",

localField: "pm\_id",

foreignField: "\_id",

as: "project\_manager"

}

},

{

$unwind: "$project\_manager" // Expand the project\_manager array

},

{

$lookup: {

from: "tasks",

localField: "tasks\_id",

foreignField: "\_id",

as: "tasks"

}

},

{

$unwind: "$tasks" // Flatten tasks array

},

{

$unwind: "$tasks.assignments" // Expand assignments array

},

{

$unwind: "$tasks.assignments.time\_logs" // Expand time\_logs array

},

{

$group: {

\_id: {

pm\_id: "$pm\_id",

pm\_name: { $concat: ["$project\_manager.first\_name", " ", "$project\_manager.last\_name"] },

project\_id: "$\_id",

project\_name: "$name"

},

total\_tasks: { $sum: 1 },

total\_time\_logged: { $sum: "$tasks.assignments.time\_logs.actual\_time" },

latest\_update: { $max: "$tasks.assignments.time\_logs.log\_date" }

}

},

{

$sort: { "latest\_update": -1 } // Sort by most recent activity

}

]);

-- Query top high priority tasks with status pending of ongoing project with nearest due date

db.projects.aggregate([

{

$match: { "status": "Ongoing" } // Filter only Ongoing projects

},

{

$lookup: {

from: "tasks",

localField: "tasks\_id",

foreignField: "\_id",

as: "tasks"

}

},

{

$unwind: "$tasks" // Expand tasks array

},

{

$unwind: "$tasks.assignments" // Expand assignments array

},

{

$match: {

"tasks.severity": { $in: ["High", "Critical"] }, // Filter High/Critical priority tasks

"tasks.status": "Pending" // Filter only Pending tasks

}

},

{

$sort: { "tasks.assignments.due\_date": 1 } // Sort by nearest due date (ascending)

},

{

$limit: 10 // Get top 10 tasks

},

{

$project: {

\_id: 0,

project\_name: "$name",

project\_id: "$\_id",

task\_id: "$tasks.\_id",

task\_name: "$tasks.name",

priority: "$tasks.severity",

status: "$tasks.status",

assigned\_to: "$tasks.assignments.employee\_id",

due\_date: "$tasks.assignments.due\_date"

}

}

]);

-- Query task that needs to be assigned by each project manager .

db.projects.aggregate([

{

$lookup: {

from: "tasks",

localField: "tasks\_id",

foreignField: "\_id",

as: "tasks"

} // Join projects with tasks

},

{

$unwind: { path: "$tasks", preserveNullAndEmptyArrays: true } // Unwind tasks array

},

{

$match: {

"tasks.assignments": { $eq: [] } // Find tasks with no assignments

}

},

{

$lookup: {

from: "employees",

localField: "pm\_id",

foreignField: "\_id",

as: "project\_manager"

} // Get project manager details

},

{

$unwind: { path: "$project\_manager", preserveNullAndEmptyArrays: true }

},

{

$group: {

\_id: "$pm\_id",

project\_manager: { $first: { $concat: ["$project\_manager.first\_name", " ", "$project\_manager.last\_name"] } },

unassigned\_tasks: {

$push: {

project\_name: "$name",

task\_name: "$tasks.name",

task\_id: "$tasks.\_id"

}

}

}

}

]);

-- Query projects with most budgets and least budgets

db.projects.aggregate([

{

$project: {

name: 1,

budget: "$summary.budget"

} // Extract project name and budget from `summary`

},

{

$facet: {

most\_budget: [{ $sort: { budget: -1 } }, { $limit: 1 }],

least\_budget: [{ $sort: { budget: 1 } }, { $limit: 1 }]

} // Split results into "most budget" and "least budget"

}

]);

-- Query detailed overview of task progress and time metrics for total tasks of project

db.projects.aggregate([

{

$match: { \_id: ObjectId("67cfc765709c6ef6ee9ac5aa") } // Filter for a specific project

},

{

$lookup: {

from: "tasks",

localField: "tasks\_id",

foreignField: "\_id",

as: "tasks"

} // Join with `tasks` collection using task IDs stored in `tasks\_id`

},

{ $unwind: "$tasks" }, // Expand tasks array into separate documents

{ $unwind: { path: "$tasks.assignments", preserveNullAndEmptyArrays: true } }, // Expand assignments

{ $unwind: { path: "$tasks.assignments.time\_logs", preserveNullAndEmptyArrays: true } }, // Expand time logs

{

$group: {

\_id: "$\_id",

project\_name: { $first: "$name" },

total\_tasks: { $sum: 1 },

completed\_tasks: {

$sum: { $cond: [{ $eq: ["$tasks.status", "Completed"] }, 1, 0] }

},

in\_progress\_tasks: {

$sum: { $cond: [{ $eq: ["$tasks.status", "In Progress"] }, 1, 0] }

},

pending\_tasks: {

$sum: { $cond: [{ $eq: ["$tasks.status", "Pending"] }, 1, 0] }

},

total\_estimated\_time: { $sum: "$tasks.estimated\_time" },

total\_actual\_time: { $sum: "$tasks.assignments.time\_logs.actual\_time" },

avg\_actual\_time\_per\_task: { $avg: "$tasks.assignments.time\_logs.actual\_time" }

}

},

{

$project: {

project\_name: 1,

total\_tasks: 1,

completed\_tasks: 1,

in\_progress\_tasks: 1,

pending\_tasks: 1,

total\_estimated\_time: 1,

total\_actual\_time: 1,

avg\_actual\_time\_per\_task: { $round: ["$avg\_actual\_time\_per\_task", 2] },

completion\_percentage: {

$round: [

{ $multiply: [{ $divide: ["$completed\_tasks", "$total\_tasks"] }, 100] },

2

]

}

}

}

]);

# st124784-Min Marn Ko

**Queries**

### 1.Identifying Tasks Past Deadlines with Responsible Team Members

db.tasks.aggregate([

{

$unwind: "$assignments" // Expand assignments array

},

{

$match: {

"assignments.due\_date": { $lt: new Date() }, // Find overdue tasks

"assignments.completion\_percentage": { $lt: 100 } // Ensure tasks are not completed

}

},

{

$lookup: {

from: "employees",

localField: "assignments.employee\_id",

foreignField: "\_id",

as: "Responsible\_Team\_Member"

}

},

{

$unwind: "$Responsible\_Team\_Member" // Ensure single employee per task entry

},

{

$project: {

\_id: 1,

name: 1,

status: 1,

"assignments.due\_date": 1,

"assignments.completion\_percentage": 1,

"Responsible\_Team\_Member.first\_name": 1,

"Responsible\_Team\_Member.last\_name": 1,

"Responsible\_Team\_Member.email": 1

}

},

{

$sort: { "assignments.due\_date": 1 } // Sort by closest overdue date first

}

])

### 2. Proportion of Task Status (Completed, In-Progress, Pending) Across Projects

db.tasks.aggregate([{

"$lookup": {

"from": "projects",

"localField": "\_id",

"foreignField": "tasks\_id",

"as": "project"} },

{

"$unwind": "$project"

},

{

"$group": {

"\_id": { "Project\_ID": "$project.\_id", "Project\_Name": "$project.name", "Status": "$status" },

"Count": { "$sum": 1 }

}

},{

"$group": {

"\_id": "$\_id.Project\_ID",

"Project\_Name": { "$first": "$\_id.Project\_Name" },

"Total\_Tasks": { "$sum": "$Count" },

"Status\_Counts": {

"$push": { "Status": "$\_id.Status", "Count": "$Count" }}

}

},{

"$unwind": "$Status\_Counts"

},{

"$project": {

"Project\_Name": 1,

"Status": "$Status\_Counts.Status",

"Count": "$Status\_Counts.Count",

"Proportion": {

"$round": [

{ "$divide": ["$Status\_Counts.Count", "$Total\_Tasks"] }, 2]}}},

{

"$sort": { "Project\_Name": 1, "Status": 1 }

}

])

### 3. Task Completion Rate by Priority Level

db.tasks.aggregate([

{

$group: {

\_id: "$severity", // Group by priority level (Low, Medium, High)

Total\_Tasks: { $sum: 1 },

Completed\_Tasks: {

$sum: { $cond: [{ $eq: ["$status", "Completed"] }, 1, 0] }

}

}

},

{

$project: {

\_id: 0,

Priority\_Level: "$\_id",

Total\_Tasks: 1,

Completed\_Tasks: 1,

Completion\_Rate: {

$round: [{ $multiply: [{ $divide: ["$Completed\_Tasks", "$Total\_Tasks"] }, 100] }, 2]

}

}

},

{ $sort: { Completion\_Rate: -1 } } // Sort by highest completion rate

])

### 4. Longest Time Taken to Complete Tasks

db.tasks.aggregate([

{

$unwind: "$assignments" // Expand assignments array

},

{

$match: {

"assignments.completion\_percentage": 100 // Only fully completed tasks

}

},

{

$addFields: {

time\_taken: {

$subtract: ["$assignments.completion\_date", "$assignments.assignment\_date"]

}

}

},

{

$project: {

\_id: 1,

name: 1,

status: 1,

"assignments.assignment\_date": 1,

"assignments.completion\_date": 1,

time\_taken\_days: { $divide: ["$time\_taken", 1000 \* 60 \* 60 \* 24] } // Convert milliseconds to days

}

},

{ $sort: { time\_taken\_days: -1 } } // Sort by longest time taken first

])

### 5. Delayed Projects Beyond Planned Deadlines

db.projects.aggregate([

{

$match: {

"summary.end\_date": { "$exists": true, "$ne": null }

}

},

{

$addFields: {

parsed\_end\_date: {

$cond: {

if: { $eq: [{ $type: "$summary.end\_date" }, "string"] },

then: { $dateFromString: { dateString: "$summary.end\_date", format: "%Y-%m-%d" } },

else: "$summary.end\_date"

}

}

}

},

{

$match: {

parsed\_end\_date: { "$lt": new Date() }, // Projects past their deadline

status: { "$nin": ["Completed", "On Hold"] } // Exclude completed and on-hold projects

}

},

{

$project: {

\_id: 0,

Project\_Name: "$name",

Status: "$status",

PM\_ID: "$pm\_id",

Deadline: "$summary.end\_date",

Parsed\_Deadline: "$parsed\_end\_date"

}

},

{ $sort: { Parsed\_Deadline: 1 } } // Sort by earliest missed deadline

])

### 1. View/Update feedbacks of team members ( eg. Comments )

db.tasks.aggregate([

{

$unwind: "$assignments" // Expand assignments array

},

{

$unwind: "$assignments.feedback" // Expand feedback array inside assignments

},

{

$lookup: {

from: "employees",

localField: "assignments.employee\_id",

foreignField: "\_id",

as: "Employee"

}

},

{

$unwind: "$Employee"

},

{

$project: {

\_id: 0,

Task\_Name: "$name",

Employee\_Name: { $concat: ["$Employee.first\_name", " ", "$Employee.last\_name"] },

"assignments.feedback.feedback\_id": 1,

"assignments.feedback.date": 1,

"assignments.feedback.comments": 1

}

},

{ $sort: { "assignments.feedback.date": -1 } } // Sort feedbacks by latest date

])

### 2. View activity log of each task

db.tasks.aggregate([

{

$unwind: "$assignments" // Expand assignments array

},

{

$lookup: {

from: "employees",

localField: "assignments.employee\_id",

foreignField: "\_id",

as: "Assigned\_Employee"

}

},

{ $unwind: "$Assigned\_Employee" },

{

$project: {

\_id: 0,

Task\_ID: "$\_id",

Task\_Name: "$name",

Status: "$status",

"assignments.assignment\_date": 1,

"assignments.completion\_percentage": 1,

"assignments.approval\_status": 1,

Assigned\_To: { $concat: ["$Assigned\_Employee.first\_name", " ", "$Assigned\_Employee.last\_name"] },

"assignments.feedback": 1,

"assignments.time\_logs": 1

}

},

{ $sort: { "assignments.assignment\_date": 1 } } // Sort logs by assignment date

])

### 3. Create/update/remove/view clients information.

db.clients.insertOne({

client\_id: ObjectId(),

name: "Acme Corp",

contact\_person: "John Doe",

address: "123 Business Street, New York, NY",

phone: "123-456-7890",

email: "johndoe@acmecorp.com",

projects: [] // Initially, no projects assigned

})