Lab Sheet 7: Using $lookup **Command in MongoDB**

---

1. Objectives:

- Understand how to perform **joins between collections** in MongoDB using $lookup.
- Learn **One-to-One** and **One-to-Many** relationships using aggregation pipelines.
- Practice combining data from multiple collections to generate meaningful results.

---

2. Prerequisites:

- Basic knowledge of MongoDB commands (insert, find, aggregate).
- MongoDB server and shell (or Compass) installed.

---

3. Theory:

$lookup **Overview:**

The $lookup stage in MongoDB performs a **left outer join** between two collections.

Syntax:

{ $lookup: { from: "<foreign_collection>", localField: "<field_in_local_collection>", foreignField: "<field_in_foreign_collection>", as: "<output_array_field>" } }

Explanation:

- from: The other collection to join with.
- localField: The field from the current (input) collection.
- foreignField: The field from the target (foreign) collection.
- as: The name of the new array field that stores the joined documents.

---

4. Learning Outcomes

After completing this lab, students will be able to:

- Implement joins using $lookup in MongoDB.
- Differentiate between one-to-one and one-to-many joins.
- Build multi-stage aggregation pipelines combining $lookup, $unwind, and $project.

5. Dataset Setup

Collection 1: students

db.students.insertMany([{ _id: 1, name: "Alice", dept_id: "D1" }, { _id: 2, name: "Bob", dept_id: "D2" }, { _id: 3, name: "Charlie", dept_id: "D1" }, { _id: 4, name: "David", dept_id: "D3" }])

Collection 2: departments

db.departments.insertMany([{ dept_id: "D1", dept_name: "Computer Science" }, { dept_id: "D2", dept_name: "Electronics" }, { dept_id: "D3", dept_name: "Mechanical" }])

---

6. Lab Exercises

Exercise 1: One-to-One Join

Display each student with their department name.

Query:

db.students.aggregate([{ $lookup: { from: "departments", localField: "dept_id", foreignField: "dept_id", as: "department_info" } }])

Exercise 2: Display Student and Department Name (Flatten Output)

Use $unwind to display department details in a flat format.

Query:

db.students.aggregate([{ $lookup: { from: "departments", localField: "dept_id", foreignField: "dept_id", as: "department_info" } }, { $unwind: "$department_info" }, { $project: { _id: 0, name: 1, dept_name: "$department_info.dept_name" } }])

Exercise 3: One-to-Many Join

Join a collection with multiple matching entries.

Create another collection:

Collection 3: marks

db.marks.insertMany([{ student_id: 1, subject: "DBMS", score: 88 }, { student_id: 1, subject: "AI", score: 92 }, { student_id: 2, subject: "Electronics", score: 75 }, { student_id: 3, subject: "DBMS", score: 80 }, { student_id: 4, subject: "Thermodynamics", score: 85 }])

Query:

db.students.aggregate([{ $lookup: { from: "marks", localField: "_id", foreignField: "student_id", as: "student_marks" } }])

Exercise 4: Nested Join (Chained $lookup**)**

Display student, department, and marks together.

Query:

db.students.aggregate([{ $lookup: { from: "departments", localField: "dept_id", foreignField: "dept_id", as: "department_info" } }, { $unwind: "$department_info" }, { $lookup: { from: "marks", localField: "_id", foreignField: "student_id", as: "marks_info" } }, { $project: { name: 1, "department_info.dept_name": 1, "marks_info.subject": 1, "marks_info.score": 1 } }])

Scenario 1: Employee–Department Database

Collections:

db.employees.insertMany([{ emp_id: 101, name: "Ravi", dept_id: "D1", salary: 50000 }, { emp_id: 102, name: "Meena", dept_id: "D2", salary: 65000 }, { emp_id: 103, name: "Kiran", dept_id: "D1", salary: 48000 }, { emp_id: 104, name: "Anu", dept_id: "D3", salary: 72000 }]);

db.departments.insertMany([{ dept_id: "D1", dept_name: "IT", manager: "Suresh" }, { dept_id: "D2", dept_name: "HR", manager: "Divya" }, { dept_id: "D3", dept_name: "Finance", manager: "Amit" }]);

Query 1:

Display all employees along with their department names and manager details.

db.employees.aggregate([{ $lookup: { from: "departments", localField: "dept_id", foreignField: "dept_id", as: "dept_info" } }, { $unwind: "$dept_info" }, { $project: { _id: 0, name: 1, dept_name: "$dept_info.dept_name", manager: "$dept_info.manager" } }]);

Scenario 2: Customer–Orders Database

Collections:

db.customers.insertMany([{ cust_id: 1, name: "Ananya", city: "Bangalore" }, { cust_id: 2, name: "Rohit", city: "Delhi" }, { cust_id: 3, name: "Kavya", city: "Pune" }]);
db.orders.insertMany([{ order_id: 201, cust_id: 1, amount: 3500, date: "2025-10-10" }, { order_id: 202, cust_id: 1, amount: 1200, date: "2025-10-09" }, { order_id: 203, cust_id: 2, amount: 4500, date: "2025-10-08" }]);

Query 2:

Retrieve each customer and the list of all orders placed by them.

db.customers.aggregate([{ $lookup: { from: "orders", localField: "cust_id", foreignField: "cust_id", as: "order_details" } }]);

6. Assignment Queries :

1. **Find departments that do not have any employees** (use $lookup + $match + $size: 0).
2. **List customers who have not placed any orders**.
3. **Find the average order value per customer** (combine $lookup, $unwind, $group).
4. **List doctors who have more than one patient.**
5. **Show students who are not enrolled in any course.**

7. Viva Questions

1. What is the purpose of $lookup in MongoDB?
2. Can $lookup perform inner joins? Explain.

3. What is the difference between $lookup and $unwind?
4. Can you join more than two collections using $lookup?
5. What happens if there is no match found in the foreign collection?