

## Lab-14 : Aggregate Method - \$lookup (join in MongoDB)

**\$lookup** is an aggregation pipeline stage used to perform a **left outer join** between two collections.


---

### Collections (Tables):

Student	Department	Fees
id	id	id
name	dept_name	student_id
dept_id		amount
marks		status

### Syntax:

```
db.collectionName.aggregate(  
  [  
    {  
      $lookup: {  
        from: "foreignCollection",  
        localField: "field_from_current_collection",  
        foreignField: "field_from_foreign_collection",  
        as: "output_array_name"  
      }  
    }  
  ]  
)
```

 Collection to join with

## Examples:

### 1) Fetch each student with department details. (JOIN From Student Collection)

```
db.Student.aggregate([
  {
    $lookup:
      {
        from: "Department",
        localField: "dept_id",
        foreignField: "_id",
        as: "department_info"
      }
  }
])
```

#### NOTE:

**localField must match the field from the LEFT (starting) collection.**

When we start from **Student**:

Left collection – **Student**

Join with – **Department**

So:

**localField:** "dept\_id" - Student side

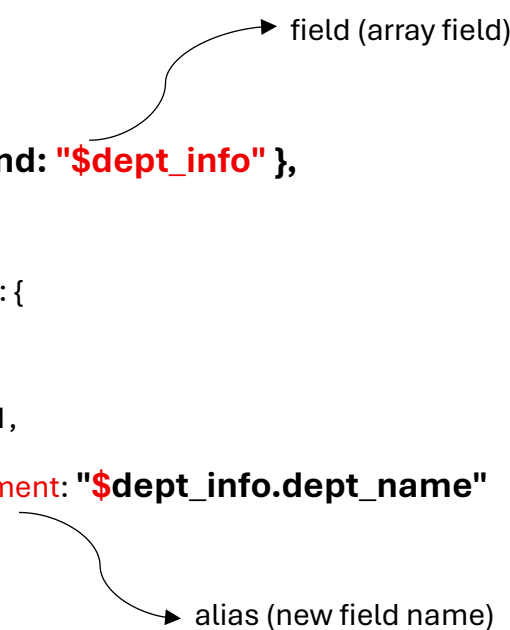
**foreignField:** "\_id" - Department side

### JOIN from Department collection:

```
db.Department.aggregate([
  {
    $lookup: {
      from: "Student",
      localField: "_id",
      foreignField: "dept_id",
      as: "student_info"
    }
  }
])
```

**2) Joins students with departments and displays only the student's name and their department name.**

```
db.Student.aggregate([
  {
    $lookup: {
      from: "Department",
      localField: "dept_id",
      foreignField: "_id",
      as: "dept_info"
    }
  },
  { $unwind: "$dept_info" },
  {
    $project: {
      _id: 0,
      name: 1,
      department: "$dept_info.dept_name"
    }
  }
])
```



**Use of unwind:** Used to break an array field into separate documents.

( **NOTE :** \$lookup always returns an **array** )

Without unwind	With unwind
<pre>dept_info: [   {     _id: 101,     dept_name: 'Computer Science'   } ]</pre>	<pre>dept_info: {   _id: 101,   dept_name: 'Computer Science' }</pre>

### 3) Finds students whose fee status is marked as “Unpaid”.

```
db.Student.aggregate([
  {
    $lookup: {
      from: "Fees",
      localField: "_id",
      foreignField: "student_id",
      as: "fee"
    }
  },
  { $unwind: "$fee" },
  { $match: { "fee.status": "Unpaid" } }
])
```

---

### 4) Fetch students with both department and fee details.

```
db.Student.aggregate([
  {
    $lookup: {
      from: "Department",
      localField: "dept_id",
      foreignField: "_id",
      as: "dept"
    }
  },
  {
```

```
{
  $lookup: {
    from: "Fees",
    localField: "_id",
    foreignField: "student_id",
    as: "fee"
  }
}
])
```

---

**5) Find students with no matching department.**

```
db.Student.aggregate([
  {
    $lookup: {
      from: "Department",
      localField: "dept_id",
      foreignField: "_id",
      as: "dept"
    }
  },
  { $match: { dept: { $size: 0 } } }
])
```

---

**6) Count students in each department.**

```
db.Student.aggregate([
  {
    $lookup: {
      from: "Department",
      localField: "dept_id",
      foreignField: "_id",
      as: "dept"
    }
  },
  { $unwind: "$dept" },
  {
    $group: {
      _id: "$dept.dept_name",
      totalStudents: { $sum: 1 }
    }
  }
])
```

---