MongoDB doesn't have **SQL-style JOINs**, but it provides **join-like features using aggregation**, mainly **$lookup**.

Insurance Schema (Sample)

customers

```
{
  _id: ObjectId("c1"),
  customerId: 101,
  name: "Rahul Sharma",
  city: "Delhi"
}
```

policies

```
{
  _id: ObjectId("p1"),
  policyId: 201,
  customerId: 101,
  policyType: "Health",
  premium: 15000
}
```

claims

```
{
  _id: ObjectId("cl1"),
  claimId: 301,
  policyId: 201,
  claimAmount: 50000,
  status: "Approved"
}
```

INNER JOIN (Customer ↔ Policy)

Customers **who have policies**

```
db.customers.aggregate([
  {
    $lookup: {
      from: "policies",
      localField: "customerId",
      foreignField: "customerId",
      as: "policyDetails"
    }
  },
  { $unwind: "$policyDetails" }
])
```

Similar to:

```
SELECT * FROM customers c
INNER JOIN policies p
ON c.customerId = p.customerId;
```

LEFT OUTER JOIN

All customers, even without policies

```
db.customers.aggregate([
  {
    $lookup: {
      from: "policies",
```

```
    localField: "customerId",
    foreignField: "customerId",
    as: "policies"
   }
  }
])
```

LEFT JOIN + FILTER (Customer with Health Policy)

```
db.customers.aggregate([
  {
   $lookup: {
    from: "policies",
    localField: "customerId",
    foreignField: "customerId",
    as: "policies"
   }
  },
  { $unwind: "$policies" },
  { $match: { "policies.policyType": "Health" } }
])
```

JOIN with Multiple Tables (Customer → Policy → Claim)

```
db.customers.aggregate([
  {
   $lookup: {
    from: "policies",
    localField: "customerId",
    foreignField: "customerId",
    as: "policy"
   }
  },
  { $unwind: "$policy" },
  {
   $lookup: {
    from: "claims",
    localField: "policy.policyId",
    foreignField: "policyId",
    as: "claims"
   }
  }
])
```

JOIN with Selected Columns (Projection)

```
db.customers.aggregate([
  {
   $lookup: {
    from: "policies",
    localField: "customerId",
    foreignField: "customerId",
    as: "policy"
   }
```

```
  },
  { $unwind: "$policy" },
  {
    $project: {
      name: 1,
      city: 1,
      "policy.policyType": 1,
      "policy.premium": 1
    }
  }
])
```

JOIN with Condition using Pipeline ($lookup Advanced)

Premium > 10,000 only

```
db.customers.aggregate([
  {
    $lookup: {
      from: "policies",
      let: { custId: "$customerId" },
      pipeline: [
        {
          $match: {
            $expr: {
              $and: [
                { $eq: ["$customerId", "$$custId"] },
                { $gt: ["$premium", 10000] }
              ]
            }
          }
        }
      ],
      as: "highPremiumPolicies"
    }
  }
])
```

FIND Customers with NO Policies (Anti-Join)

```
db.customers.aggregate([
  {
    $lookup: {
      from: "policies",
      localField: "customerId",
      foreignField: "customerId",
      as: "policies"
    }
  },
  { $match: { policies: { $size: 0 } } }
])
```

SQL JOIN	MongoDB Equivalent
INNER JOIN	$lookup + $unwind
LEFT JOIN	$lookup
JOIN + WHERE	$lookup + $match
JOIN 3 tables	Multiple $lookup
NOT EXISTS	$lookup + $size:0

Topic: Join-Like Operations using $lookup

Create Collections

use insuranceDB

Insert Sample Data

Customers

```
db.customers.insertMany([
  { customerId: 101, name: "Rahul Sharma", city: "Delhi" },
  { customerId: 102, name: "Anita Verma", city: "Mumbai" },
  { customerId: 103, name: "Suresh Kumar", city: "Chennai" }
])
```

Policies

```
db.policies.insertMany([
  { policyId: 201, customerId: 101, policyType: "Health", premium: 15000 },
  { policyId: 202, customerId: 101, policyType: "Life", premium: 25000 },
  { policyId: 203, customerId: 102, policyType: "Vehicle", premium: 8000 }
])
```

Claims

```
db.claims.insertMany([
  { claimId: 301, policyId: 201, claimAmount: 50000, status: "Approved" },
  { claimId: 302, policyId: 201, claimAmount: 20000, status: "Pending" },
  { claimId: 303, policyId: 203, claimAmount: 30000, status: "Rejected" }
])
```

LAB-1: List Customers with Their Policies
(INNER JOIN)
Display customer name, city, policy type and premium.
Solution

```
db.customers.aggregate([
  {
    $lookup: {
      from: "policies",
      localField: "customerId",
      foreignField: "customerId",
      as: "policy"
    }
  },
  { $unwind: "$policy" },
  {
    $project: {
      _id: 0,
      name: 1,
      city: 1,
      "policy.policyType": 1,
```

```
    "policy.premium": 1
   }
  }
 ])
```

LAB-2: List ALL Customers Including Those Without Policies (LEFT OUTER JOIN)

Show customers even if they don't have any policy.

Solution

```
db.customers.aggregate([
 {
  $lookup: {
   from: "policies",
   localField: "customerId",
   foreignField: "customerId",
   as: "policies"
  }
 }
])
```

LAB-3: Customers Having Health Policies Only

Display customers who have Health policy.

Solution

```
db.customers.aggregate([
 {
  $lookup: {
   from: "policies",
   localField: "customerId",
   foreignField: "customerId",
   as: "policy"
  }
 },
 { $unwind: "$policy" },
 { $match: { "policy.policyType": "Health" } }
])
```

LAB-4: Customers with Premium Greater Than 20,000

Find customers whose policy premium is above 20,000.

Solution

```
db.customers.aggregate([
 {
  $lookup: {
   from: "policies",
   localField: "customerId",
   foreignField: "customerId",
   as: "policy"
  }
 },
 { $unwind: "$policy" },
 { $match: { "policy.premium": { $gt: 20000 } } }
])
```

LAB-5: Customers Without Any Policy
(ANTI JOIN)

List customers who do NOT have policies.

Solution

```
db.customers.aggregate([
 {
  $lookup: {
   from: "policies",
   localField: "customerId",
   foreignField: "customerId",
   as: "policies"
  }
 },
 { $match: { policies: { $size: 0 } } }
])
```

LAB-6: Customer → Policy → Claims
(MULTI-COLLECTION JOIN)

Show customer name, policy type, claim amount and claim status.

Solution

```
db.customers.aggregate([
 {
  $lookup: {
   from: "policies",
   localField: "customerId",
   foreignField: "customerId",
   as: "policy"
  }
 },
 { $unwind: "$policy" },
 {
  $lookup: {
   from: "claims",
   localField: "policy.policyId",
   foreignField: "policyId",
   as: "claims"
  }
 },
 { $unwind: "$claims" },
 {
  $project: {
   _id: 0,
   name: 1,
   "policy.policyType": 1,
   "claims.claimAmount": 1,
   "claims.status": 1
  }
 }
])
```

LAB-7: Policies Without Claims

Find policies which do not have any claims.

Solution

```
db.policies.aggregate([
 {
  $lookup: {
   from: "claims",
   localField: "policyId",
   foreignField: "policyId",
   as: "claims"
  }
 },
 { $match: { claims: { $size: 0 } } }
])
```

LAB-8: Approved Claims Only

Display customers whose claims are **Approved**.

Solution

```
db.customers.aggregate([
 {
  $lookup: {
   from: "policies",
   localField: "customerId",
   foreignField: "customerId",
   as: "policy"
  }
 },
 { $unwind: "$policy" },
 {
  $lookup: {
   from: "claims",
   localField: "policy.policyId",
   foreignField: "policyId",
   as: "claims"
  }
 },
 { $unwind: "$claims" },
 { $match: { "claims.status": "Approved" } }
])
```

LAB-9: Count Policies Per Customer

Show number of policies per customer.

Solution

```
db.customers.aggregate([
 {
  $lookup: {
   from: "policies",
   localField: "customerId",
   foreignField: "customerId",
   as: "policies"
  }
```

```
  },
  {
   $project: {
    name: 1,
    totalPolicies: { $size: "$policies" }
   }
  }
])
```

LAB-10: High-Premium Policies using $lookup Pipeline

Fetch customers with policies premium > 10,000 using advanced lookup.

Solution

```
db.customers.aggregate([
 {
  $lookup: {
   from: "policies",
   let: { custId: "$customerId" },
   pipeline: [
    {
     $match: {
      $expr: {
       $and: [
        { $eq: ["$customerId", "$$custId"] },
        { $gt: ["$premium", 10000] }
       ]
      }
     }
    }
   ],
   as: "highPremiumPolicies"
  }
 }
])
```

Observations

- $lookup + $unwind → INNER JOIN
- $lookup only → LEFT JOIN
- $size : 0 → NOT EXISTS
- Multiple $lookup → Multi-table JOIN
- $lookup + pipeline → WHERE inside JOIN
