ASSIGNMENT – MongoDB

MongoDB is a document-oriented NoSQL database used for high-volume data storage. mongosh (MongoDB Shell) is the interactive command-line interface that allows users to interact with MongoDB instances.

Download MongoDB Community Server :

1. Go to: https://www.mongodb.com/try/download/community
2. Choose:
   - **Version**: Latest stable
   - **Platform**: Windows x64
   - **Package**: **MSI**
3. Click **Download**.

Install MongoDB Community Server :

1. Open the downloaded .msi file.
2. In Setup:
   - Choose **Complete** installation.
   - Check the box: ✅*Install MongoDB as a Service*
   - Leave other defaults as-is.
3. Optionally: Check ✅*Install MongoDB Compass* (GUI).
4. Click **Install**.
5. Finish once setup is complete.

Download MongoDB Shell (mongosh) :

1. Visit: https://www.mongodb.com/try/download/shell
2. Choose:
   - **Version**: Latest
   - **Platform**: Windows x64
   - **Package**: MSI
3. Click **Download** and install using the wizard.

✅Once installed, mongosh will be available for use in the Command Prompt.

Set Environment Variables (Optional, but recommended) :

1. Press Win + S → Search **"Environment Variables"** → Click **"Edit the system environment variables"**.
2. In the System Properties window, click **Environment Variables**.
3. Under **System Variables**, select Path → Click **Edit** → Click **New**.

4. Add the following paths:
   - MongoDB Server:

     C:\Program Files\MongoDB\Server\<version>\bin

   - MongoDB Shell:

     C:\Program Files\MongoDB\mongosh\bin

5. Click OK on all windows.

Verify Installation :

Open **Command Prompt**, type the following commands:

mongod --version

mongosh --version

If installed correctly, you'll see version information for both.

Start MongoDB Shell (mongosh) :

In a new Command Prompt window:

mongosh

You'll get the prompt:

test>

Now you're ready to run MongoDB queries.

MongoDB PRACTICE QUERIES :

```
test> use mydb

switched to db mydb

mydb> db.people.insertMany([

... {id: 1, user_id: 1001, status: "A", age: 25},

... {id: 2, user_id: 1002, status: "B", age: 30},

... {id: 3, user_id: 1003, status: "C", age: 50},

... {id: 4, user_id: 1004, status: "C", age: 50}
```

```
...
mydb> db.people.insertMany([
... {id: 1, user_id: 1001, status: "A", age: 25},
... {id: 2, user_id: 1002, status: "B", age: 30},
... {id: 3, user_id: 1003, status: "A", age: 50},
... {id: 4, user_id: 1004, status: "C", age:50}
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('687e0f332112325795eec4a9'),
    '1': ObjectId('687e0f332112325795eec4aa'),
    '2': ObjectId('687e0f332112325795eec4ab'),
    '3': ObjectId('687e0f332112325795eec4ac')
  }
}
mydb> db.people.find()
[
  {
    _id: ObjectId('687e0f332112325795eec4a9'),
    id: 1,
    user_id: 1001,
    status: 'A',
    age: 25
  },
  {
    _id: ObjectId('687e0f332112325795eec4aa'),
    id: 2,
    user_id: 1002,
    status: 'B',
    age: 30
  },
  {
    _id: ObjectId('687e0f332112325795eec4ab'),
    id: 3,
    user_id: 1003,
    status: 'A',
    age: 50
  },
  {
    _id: ObjectId('687e0f332112325795eec4ac'),
    id: 4,
    user_id: 1004,
    status: 'C',
    age: 50
  }
]
mydb> db.people.find({}, {id: 1, user_id: 1, status: "A"})
[
  {
    _id: ObjectId('687e0f332112325795eec4a9'),
    id: 1,
    user_id: 1001,
    status: 'A'
  },
  {
    _id: ObjectId('687e0f332112325795eec4aa'),
    id: 2,
    user_id: 1002,
    status: 'A'
  },
  {
    _id: ObjectId('687e0f332112325795eec4ab'),
    id: 3,
    user_id: 1003,
    status: 'A'
  },
  {
    _id: ObjectId('687e0f332112325795eec4ac'),
    id: 4,
    user_id: 1004,
    status: 'A'
  }
]
mydb> db.people.find({}, {id: 1, user_id: 1, status: 1})
[
  {
    _id: ObjectId('687e0f332112325795eec4a9'),
    id: 1,
    user_id: 1001,
    status: 'A'
  },
  {
    _id: ObjectId('687e0f332112325795eec4aa'),
    id: 2,
    user_id: 1002,
    status: 'B'
  },
  {
    _id: ObjectId('687e0f332112325795eec4ab'),
    id: 3,
    user_id: 1003,
    status: 'A'
  },
  {
    _id: ObjectId('687e0f332112325795eec4ac'),
    id: 4,
    user_id: 1004,
    status: 'C'
  }
]
mydb> mydb> mydb>

mydb> db.people.find({}, {user_id: 1, status: 1, _id: 0})
[
  { user_id: 1001, status: 'A' },
  { user_id: 1002, status: 'B' },
  { user_id: 1003, status: 'A' },
  { user_id: 1004, status: 'C' }
]
mydb> db.people.finf({ status: "A"})
TypeError: db.people.finf is not a function
mydb> db.people.find({ status: "A"})
[
  {
    _id: ObjectId('687e0f332112325795eec4a9'),
    id: 1,
    user_id: 1001,
    status: 'A',
    age: 25
  },
  {
    _id: ObjectId('687e0f332112325795eec4ab'),
    id: 3,
    user_id: 1003,
    status: 'A',
    age: 50
  }
]
mydb> db.people.find({status: "A"}, {user_id: 1, status: 1, _id: 0})
[ { user_id: 1001, status: 'A' }, { user_id: 1003, status: 'A' } ]
mydb> db.people.find({status: {$ne: "A"}})
[
  {
    _id: ObjectId('687e0f332112325795eec4aa'),
    id: 2,
    user_id: 1002,
    status: 'B',
    age: 30
  },
  {
    _id: ObjectId('687e0f332112325795eec4ac'),
    id: 4,
    user_id: 1004,
    status: 'C',
    age: 50
  }
]
mydb> db.people.find({status: "A", age: 50})
[
  {
    _id: ObjectId('687e0f332112325795eec4ab'),
    id: 3,
    user_id: 1003,
    status: 'A',
    age: 50
  }
]
mydb> db.people.find({$or:[{status:"A",age:50})
Uncaught:
SyntaxError: Unexpected token, expected "," (1:40)

> 1 | db.people.find({$or:[{status:"A",age:50})
    |                                        ^
  2 |

mydb> db.people.find({$or:[{status:"A"}, {age:50}]})
[
  {
    _id: ObjectId('687e0f332112325795eec4a9'),
    id: 1,
    user_id: 1001,
    status: 'A',
    age: 25
  },
  {
    _id: ObjectId('687e0f332112325795eec4ab'),
    id: 3,
    user_id: 1003,
    status: 'A',
    age: 50
  },
  {
    _id: ObjectId('687e0f332112325795eec4ac'),
    id: 4,
    user_id: 1004,
    status: 'C',
    age: 50
  }
]
mydb> version()
2.5.6
mydb> Get -History
ReferenceError: Get is not defined
mydb> Get-History
ReferenceError: Get is not defined
mydb> db.people.find({age: {$gt: 25}})
[
  {
    _id: ObjectId('687e0f332112325795eec4aa'),
    id: 2,
    user_id: 1002,
    status: 'B',
    age: 30
  },
  {
    _id: ObjectId('687e0f332112325795eec4ab'),
    id: 3,
    user_id: 1003,
    status: 'A',
    age: 50
  },
  {
    _id: ObjectId('687e0f332112325795eec4ac'),
    id: 4,
    user_id: 1004,
    status: 'C',
    age: 50
  }
]
mydb> db.people.find({age: {$lt: 25}})

mydb> db.people.find({age:{$gt:25, $lt:50}})
[
  {
    _id: ObjectId('687e0f332112325795eec4aa'),
    id: 2,
    user_id: 1002,
    status: 'B',
    age: 30
  }
]
mydb> db.people.find({user_id: /bc/})

mydb> db.people.find({status:"A"}).sort({user_id:1})
[
  {
    _id: ObjectId('687e0f332112325795eec4a9'),
    id: 1,
    user_id: 1001,
    status: 'A',
    age: 25
  },
  {
    _id: ObjectId('687e0f332112325795eec4ab'),
    id: 3,
    user_id: 1003,
    status: 'A',
    age: 50
  }
]
mydb> db.people.find({status:"A"}).sort({user_id: -1})
[
  {
    _id: ObjectId('687e0f332112325795eec4ab'),
    id: 3,
    user_id: 1003,
    status: 'A',
    age: 50
  },
  {
    _id: ObjectId('687e0f332112325795eec4a9'),
    id: 1,
    user_id: 1001,
    status: 'A',
    age: 25
  }
]
mydb> db.people.count()
DeprecationWarning: Collection.count() is deprecated. Use countDocuments or
estimatedDocumentCount.
4
mydb> db.people.countDocuments({})
4
mydb> db.people.count({user_id: {$exists:true}})
4
mydb> db.people.count({age:{$gt:30}})
2
mydb> db.people.findOne()
```

```
{
  _id: ObjectId('687e0f332112325795eec4a9'),
  id: 1,
  user_id: 1001,
  status: 'A',
  age: 25
}
mydb> db.people.findTwo()
TypeError: db.people.findTwo is not a function
mydb> db.people.findSecond()
TypeError: db.people.findSecond is not a function
mydb> db.people.find().limit(5).skip(10)

mydb> db.people.find().limit(3).skip(1)
[
  {
    _id: ObjectId('687e0f332112325795eec4aa'),
    id: 2,
    user_id: 1002,
    status: 'B',
    age: 30
  },
  {
    _id: ObjectId('687e0f332112325795eec4ab'),
    id: 3,
    user_id: 1003,
    status: 'A',
    age: 50
  },
  {
    _id: ObjectId('687e0f332112325795eec4ac'),
    id: 4,
    user_id: 1004,
    status: 'C',
    age: 50
  }
]
mydb> db.people.find().limit(4).skip(2)
[
  {
    _id: ObjectId('687e0f332112325795eec4ab'),
    id: 3,
    user_id: 1003,
    status: 'A',
    age: 50
  },
  {
    _id: ObjectId('687e0f332112325795eec4ac'),
    id: 4,
    user_id: 1004,
    status: 'C',
    age: 50
  }
]
mydb> db.people.find({status:"A"}).explain()
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'mydb.people',
    parsedQuery: { status: { '$eq': 'A' } },
    indexFilterSet: false,
    queryHash: '5D6543D9',
    planCacheShapeHash: '5D6543D9',
    planCacheKey: '405CB45D',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'COLLSCAN',
      filter: { status: { '$eq': 'A' } },
      direction: 'forward'
    },
    rejectedPlans: []
  },
  queryShapeHash: 'E4B71C1FC6A4544FA07E582BF9C76628598DFC2DA2B46B02BC8B86D1A595E8BD',
  command: { find: 'people', filter: { status: 'A' }, '$db': 'mydb' },
  serverInfo: {
    host: 'DESKTOP-VM2OBVF',
    port: 27017,
    version: '8.0.11',
    gitVersion: 'bed99f699da6cb2b74262aa6d473446c41476643'
  },
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
    internalQueryFrameworkControl: 'trySbeRestricted',
    internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
  },
  ok: 1
}
mydb> db.people.updateMany({age:{$gt:25}},{$set{status:"C"}})
Uncaught:
SyntaxError: Unexpected token, expected "," (1:41)

> 1 | db.people.updateMany({age:{$gt:25}},{$set{status:"C"}})
    |                                          ^
  2 |

mydb> db.people.updateMany({age:{$gt:25}},{$set:{status:"C"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 3,
  modifiedCount: 2,
  upsertedCount: 0
}
mydb> db.people.updateMany({status:"A"},{$inc:{age:3}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
mydb> db.people.deleteMany({status:"D"})

{ acknowledged: true, deletedCount: 0 }

mydb> db.people.deleteMany({status:"C"})

{ acknowledged: true, deletedCount: 3 }

mydb> db.people.deleteMany({})

{ acknowledged: true, deletedCount: 1 }

mydb> db.people.find()


mydb> db.people.deleteMany({})
```