-**Assignment**: Creating a Database Using MongoDB and Mongosh

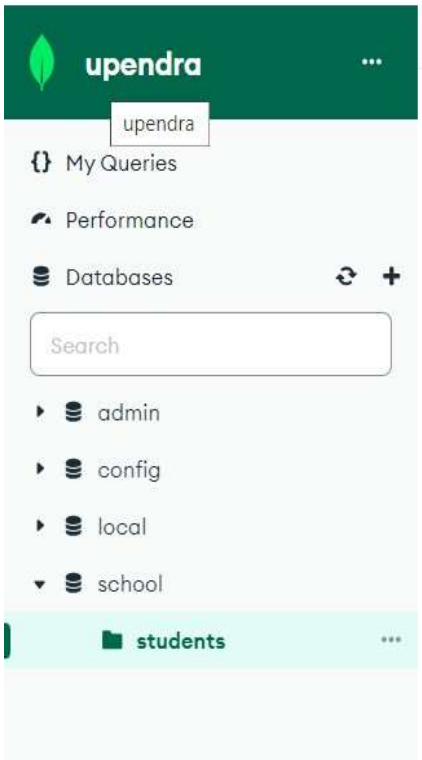
MongoDB: MongoDB is a source -available, cross-platform, document-oriented database program. Classified as a NoSQL database product, MongoDB utilizes JSON-like documents with optional schemas.

MongoDB is developed by MongoDB Inc.

ASSIGNMENT DETAILS:

→Database Setup: Created a new MongoDB database called “School”.

→Collection Creation: Created a collection named as “students” within the school database.



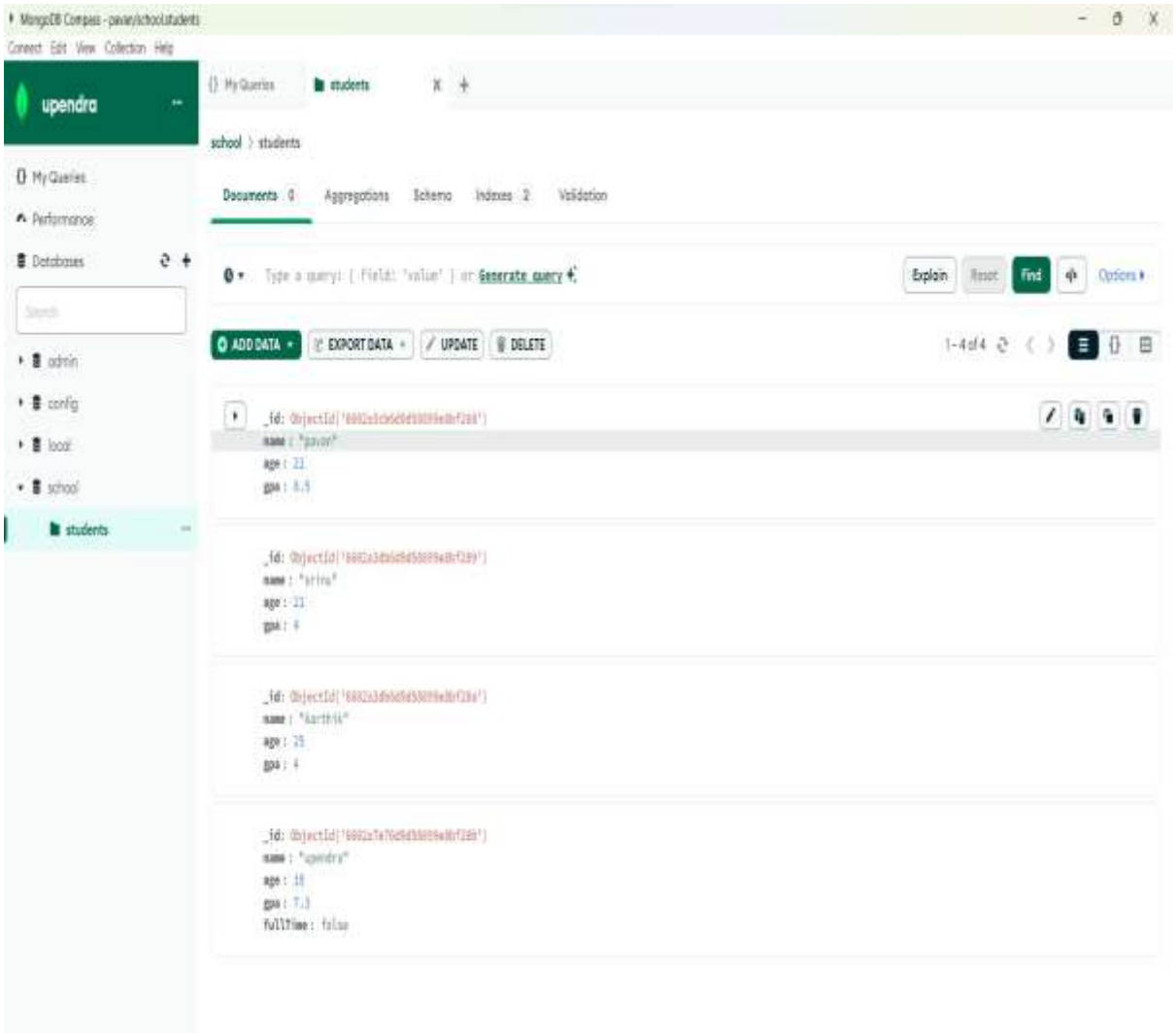
→ Document insertion: Inserted three documents into the Student collection, each representing a students with fields such as name, age and gpa.

In MongoDB PowerShell:

```
school> db.students.insertOne({name:"pavan", age:21, gpa:8.5})
{
  acknowledged: true,
  insertedId: ObjectId('6602a3db6d9d58899e8bf208')
}
school> db.students.find()
[
  {
    _id: ObjectId('6602a3db6d9d58899e8bf208'),
    name: 'pavan',
    age: 21,
    gpa: 8.5
  }
]
school> db.students.insertMany([{name:"srinu", age:21, gpa:4},{name:"karthik", age:25, gpa:4}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6602a3db6d9d58899e8bf209'),
    '1': ObjectId('6602a3db6d9d58899e8bf20a')
  }
}
school> db.students.find()
[
  {
    _id: ObjectId('6602a3db6d9d58899e8bf208'),
    name: 'pavan',
    age: 21,
    gpa: 8.5
  },
  {
    _id: ObjectId('6602a3db6d9d58899e8bf209'),
    name: 'srinu',
    age: 21,
    gpa: 4
  },
```

```
  {
    _id: ObjectId('6602a3db6d9d58899e8bf20a'),
    name: 'karthik',
    age: 25,
    gpa: 4
  }
]
school> db.students.insertOne({name:"upendra", age:19,gpa:7.3,fullTime:false})
{
  acknowledged: true,
  insertedId: ObjectId('6602a7e76d9d58899e8bf20b')
}
```

In compass:



→Quering: Queries to retrieve: All students from the student collection. Students with age greater than or equal to 30.

→Comparision operator: Operators return data based on value comparisons

Less than($lt):

```
school> db.students.find({age:{$lt:20}})
[
  {
    _id: ObjectId('6602a7e76d9d58099e8bf28b'),
    name: 'upendra',
    age: 19,
    gpa: 7.3,
    fullTime: false
  }
]
school>
```

Greater than: ($gt)

```
school> db.students.find({age:{$gt:20}})
[
  {
    _id: ObjectId('6602a3cb6d9d58899e8bf208'),
    name: 'pavan',
    age: 21,
    gpa: 8.5
  },
  {
    _id: ObjectId('6602a3db6d9d58899e8bf209'),
    name: 'srinu',
    age: 21,
    gpa: 4
  },
  {
    _id: ObjectId('6602a3db6d9d58899e8bf20a'),
    name: 'karthik',
    age: 25,
    gpa: 4
  }
]
```

Greater than equal: ($gte)

```
school> db.students.find({age:{$gte:20}})
[
  {
    _id: ObjectId('6602a3cb6d9d58899e8bf208'),
    name: 'pavan',
    age: 21,
    gpa: 8.5
  },
  {
    _id: ObjectId('6602a3db6d9d58899e8bf209'),
    name: 'srinu',
    age: 21,
    gpa: 4
  },
  {
    _id: ObjectId('6602a3db6d9d58899e8bf20a'),
    name: 'karthik',
    age: 25,
    gpa: 4
  }
]
```

*Update Operator: Updated the job of a student with a specific name.

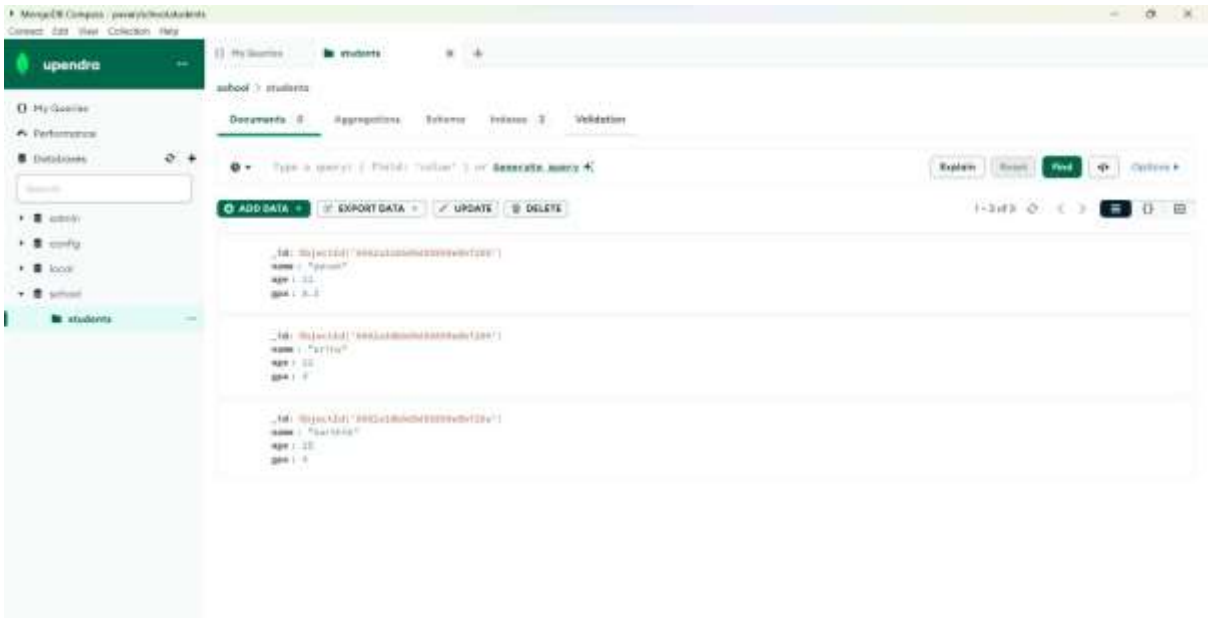
```
school> db.students.updateOne({name:"upendra"},{$set:{job:"student"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
school> db.students.find({name:"upendra"})
[
  {
    _id: ObjectId('6602a7e76d9d58899e8bf20b'),
    name: 'upendra',
    age: 19,
    gpa: 7.3,
    fullTime: false,
    job: 'student'
  }
]
```

```
school> db.students.updateOne({_id:ObjectId('6602a7e76d9d58899e8bf20b')},{$unset:{job:"student"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
school> db.students.find({name:"upendra"})
[
  {
    _id: ObjectId('6602a7e76d9d58899e8bf20b'),
    name: 'upendra',
    age: 19,
    gpa: 7.3,
    fullTime: false
  }
]
```

*Deletion Operator: Delete a user document based on a specific email address.

```
school> db.students.deleteOne({name:"upendra"})
{ acknowledged: true, deletedCount: 1 }
school>
```

In compass:



*Index Creation: Indexes support efficient execution of queries in MongoDB. Without indexes, MongoDB must scan every document in a collection to return query results. If an appropriate index exists for a query, MongoDB uses the index to limit the number of documents it must scan.

```
school> db.students.createIndex({name:1})
name_1
school> db.students.find({name:"pavan"}).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'school.students',
    indexFilterSet: false,
    parsedQuery: { name: { '$eq': 'pavan' } },
    queryHash: 'A2F088FD',
    planCacheKey: '41E45410',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { name: 1 },
        indexName: 'name_1',
        isMultiKey: false,
        multiKeyPaths: { name: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: { name: [ '["pavan", "pavan"]' ] }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 1,
    executionTimeMillis: 0,
    totalKeysExamined: 1,
    totalDocsExamined: 1,
    executionStages: {
      stage: 'FETCH',
```

```
      nReturned: 1,
      executionTimeMillisEstimate: 0,
      works: 2,
      advanced: 1,
      needTime: 0,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 1,
      docsExamined: 1,
      alreadyHasObj: 0,
      inputStage: {
        stage: 'IXSCAN',
        nReturned: 1,
        executionTimeMillisEstimate: 0,
        works: 2,
        advanced: 1,
        needTime: 0,
        needYield: 0,
        saveState: 0,
        restoreState: 0,
        isEOF: 1,
        keyPattern: { name: 1 },
        indexName: 'name_1',
        isMultiKey: false,
        multiKeyPaths: { name: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: { name: [ '["pavan", "pavan"]' ] },
        keysExamined: 1,
        works: 1,
        dupsTested: 0,
        dupsDropped: 0
      }
    }
  },
  command: { find: 'students', filter: { name: 'pavan' }, '$db': 'school' },
```

```
  serverInfo: {
    host: 'DESKTOP-RILC9AC',
    port: 27017,
    version: '7.0.7',
    gitVersion: 'cfb08e1ab7ef741b4abdd0638351b322514c45bd'
  },
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
    internalQueryFrameworkControl: 'trySbeRestricted'
  },
  ok: 1
}
```