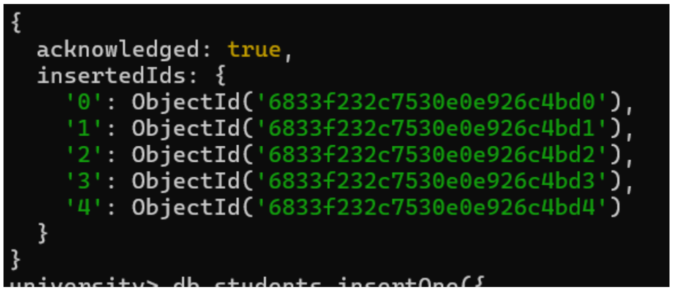
CRUD Operations

1. Insert a new student:

Code:

```
db.students.insertOne({
  name: "Ayaan",
  age: 21,
  gender: "Male",
  department: "Computer Science",
  courses: [
    { name: "MongoDB", score: 85 },
    { name: "Python", score: 90 }
  ],
  address: {
    city: "Hyderabad",
    state: "Telangana",
    pincode: 500032
  },
  enrollmentDate: ISODate("2024-08-01T00:00:00Z"),
  isActive: true
})
```



```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6833f232c7530e0e926c4bd0'),
    '1': ObjectId('6833f232c7530e0e926c4bd1'),
    '2': ObjectId('6833f232c7530e0e926c4bd2'),
    '3': ObjectId('6833f232c7530e0e926c4bd3'),
    '4': ObjectId('6833f232c7530e0e926c4bd4')
  }
}
university> db.students.insertOne({
```

2. Update score for Python course:

Code:

```
db.students.updateOne(
  { name: "divya", "courses.name": "Python" },
  { $set: { "courses.$.score": 95 } }
)
```

3. Delete student "mani":

Code:

```
db.students.deleteOne({ name: "mani" })
```

4. Find students in "Computer Science":

Code:

```
db.students.find({ department: "Computer Science" })
```

Query Operators

5. Age > 20:

code:

```
db.students.find({ age: { $gt: 20 } })
```

6. Enrollment between two dates:

Code:

```
db.students.find({
  enrollmentDate: {
    $gte: ISODate("2024-07-01T00:00:00Z"),
    $lte: ISODate("2024-09-30T00:00:00Z")
  }
})
```

7. Department: CS or Mathematics

Code:

```
db.students.find({

  department: { $in: ["Computer Science", "Mathematics"] }
})
```

8. Not in Mechanical:

Code:

```
db.students.find({
  department: { $ne: "Mechanical" }
})
```

9. Any course with score > 80:

Code:

```
db.students.find({
  "courses.score": { $gt: 80 }
})
```

Aggregation Framework

10. Group by department and count:

Code:

```
db.students.aggregate([
  { $group: { _id: "$department", count: { $sum: 1 } } }
])
```

11. Average age per department:

Code:

```
db.students.aggregate([
  { $group: { _id: "$department", avgAge: { $avg: "$age" } } }
])
```

12. Sort by total course score:

Code:

```
db.students.aggregate([
  {
    $project: {
      name: 1,
      totalScore: { $sum: "$courses.score" }
    }
  },
  { $sort: { totalScore: -1 } }
])
```

13. Filter active students before grouping:

Code:

```
db.students.aggregate([
  { $match: { isActive: true } },
  { $group: { _id: "$department", activeCount: { $sum: 1 } } }
])
```

14. Unique cities from address:

Code:

```
db.students.aggregate([
  { $group: { _id: "$address.city" } }
])
```

Projections

15. Only name, department, city:

Code:

```
db.students.find({}, {
  name: 1,
  department: 1,
  "address.city": 1,
  _id: 0
})
```

16. Exclude `_id`:

Code:

```
db.students.find({}, { _id: 0 })
```

17. Show name and total score:

Code:

```
db.students.aggregate([
  {
    $project: {
      name: 1,
      totalScore: { $sum: "$courses.score" }
    }
  }
])
```

Embedded Documents

18. Address.city = "Chennai":

Code:

```
db.students.find({ "address.city": "Chennai" })
```

19. Update address.pincode for "Meera":

Code:

```
db.students.updateOne(
  { name: "Meera" },
  { $set: { "address.pincode": 682005 } }
)
```

20. Add `landmark` to all addresses:

Code:

```
db.students.updateMany(
  {},
  { $set: { "address.landmark": "Near Main Road" } }
)
```

Array Operations

21. Add "Node.js" to "nithya":

Code:

```
db.students.updateOne(
  { name: "nithya" },
  { $push: { courses: { name: "Node.js", score: 88 } } }
)
```

22. Remove "MongoDB" from "divya":

Code:

```
db.students.updateOne(
  { name: "divya" },
  { $pull: { courses: { name: "MongoDB" } } }
)
```

23. Enrolled in both Python and MongoDB:

Code:

```
db.students.find({
  "courses.name": { $all: ["Python", "MongoDB"] }
})
```

24. $elemMatch for MongoDB score > 80:

Code:

```
db.students.find({
  courses: { $elemMatch: { name: "MongoDB", score: { $gt: 80 } } }
})
```