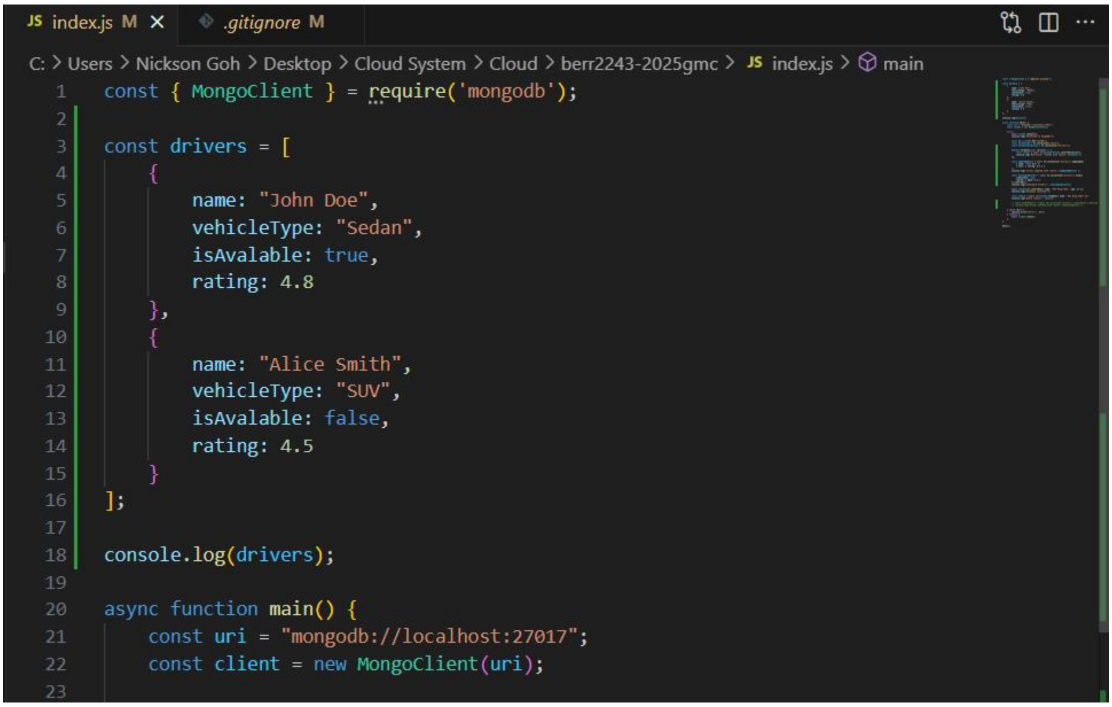
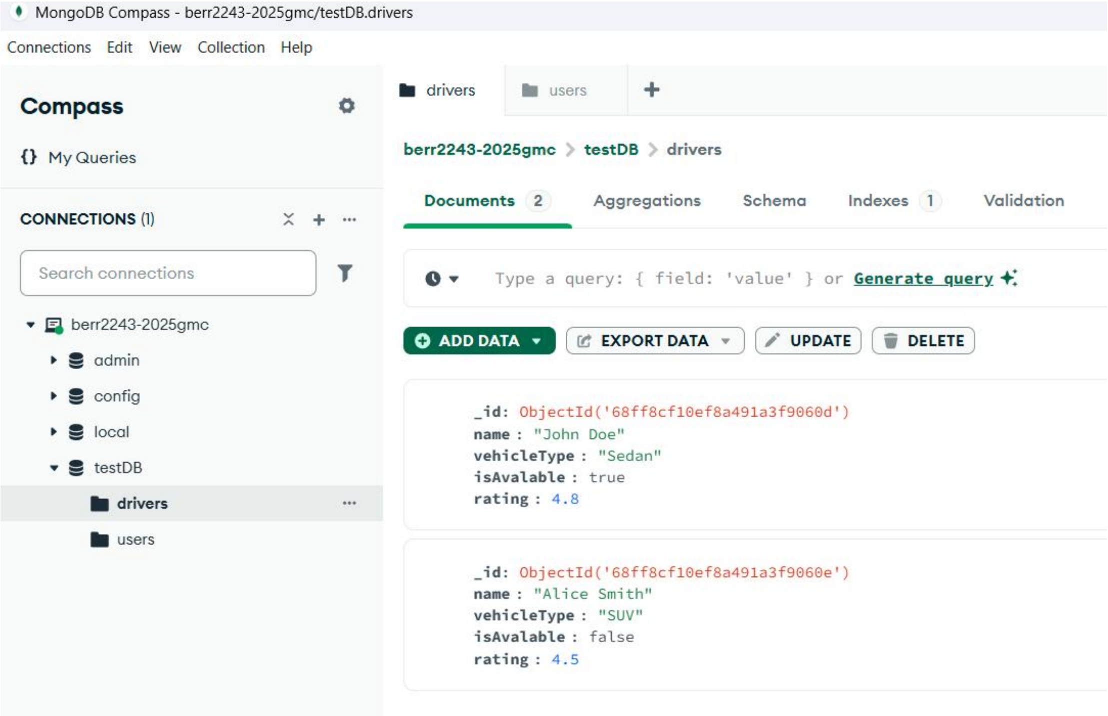
Task 1


index.js M
.gitignore M
C: > Users > Nickson Goh > Desktop > Cloud System > Cloud > berr2243-2025gmc > index.js > main

```js
const { MongoClient } = require('mongodb');

const drivers = [
    {
        name: "John Doe",
        vehicleType: "Sedan",
        isAvalable: true,
        rating: 4.8
    },
    {
        name: "Alice Smith",
        vehicleType: "SUV",
        isAvalable: false,
        rating: 4.5
    }
];

console.log(drivers);

async function main() {
    const uri = "mongodb://localhost:27017";
    const client = new MongoClient(uri);

```

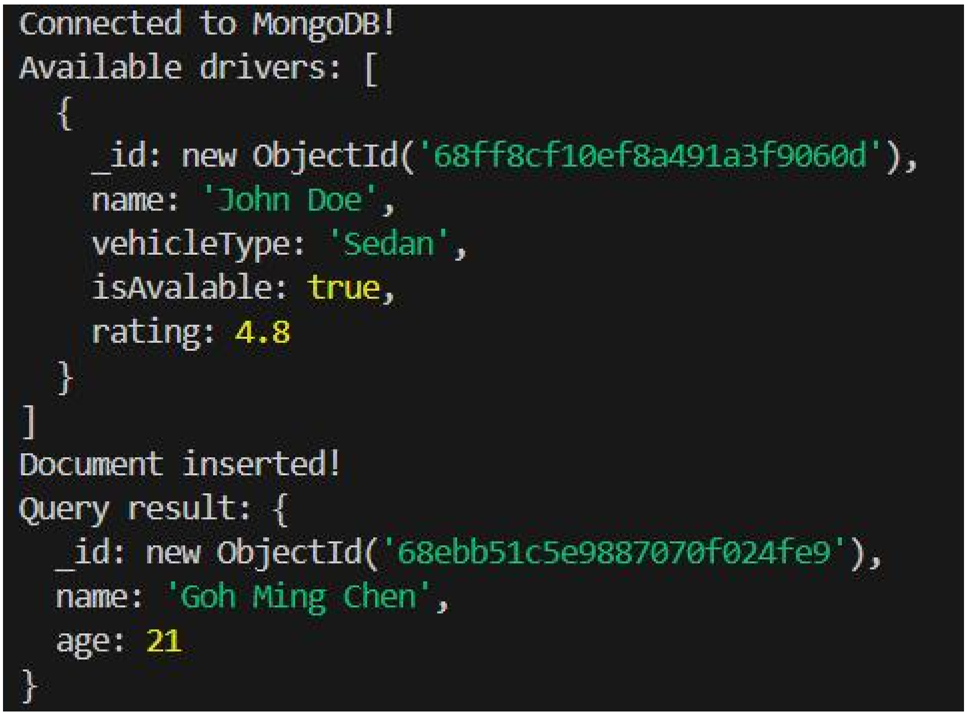
Task 3



```
_id: ObjectId('68ff8cf10ef8a491a3f9060d')
name : "John Doe"
vehicleType : "Sedan"
isAvalable : true
rating : 4.8
```

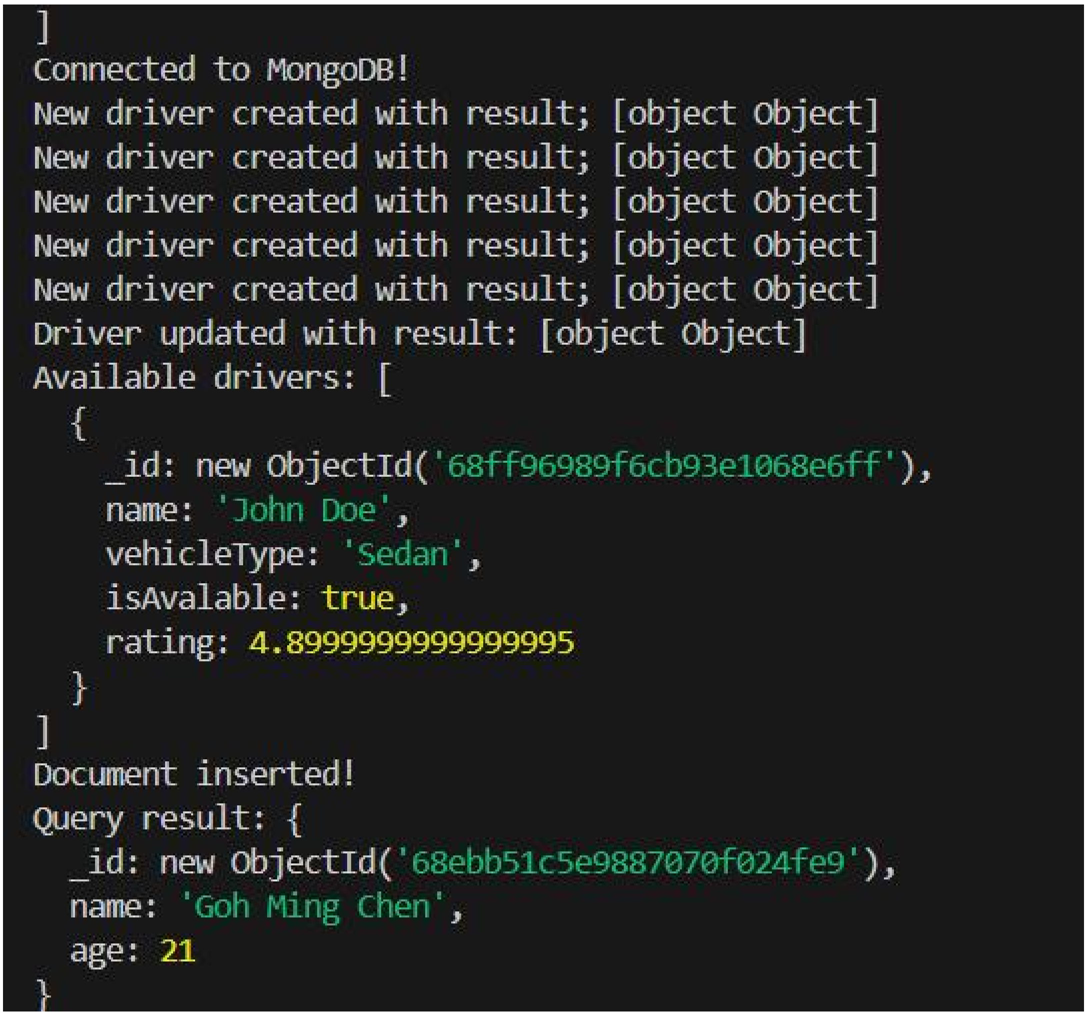
```
_id: ObjectId('68ff8cf10ef8a491a3f9060e')
name : "Alice Smith"
vehicleType : "SUV"
isAvalable : false
rating : 4.5
```

Task 4



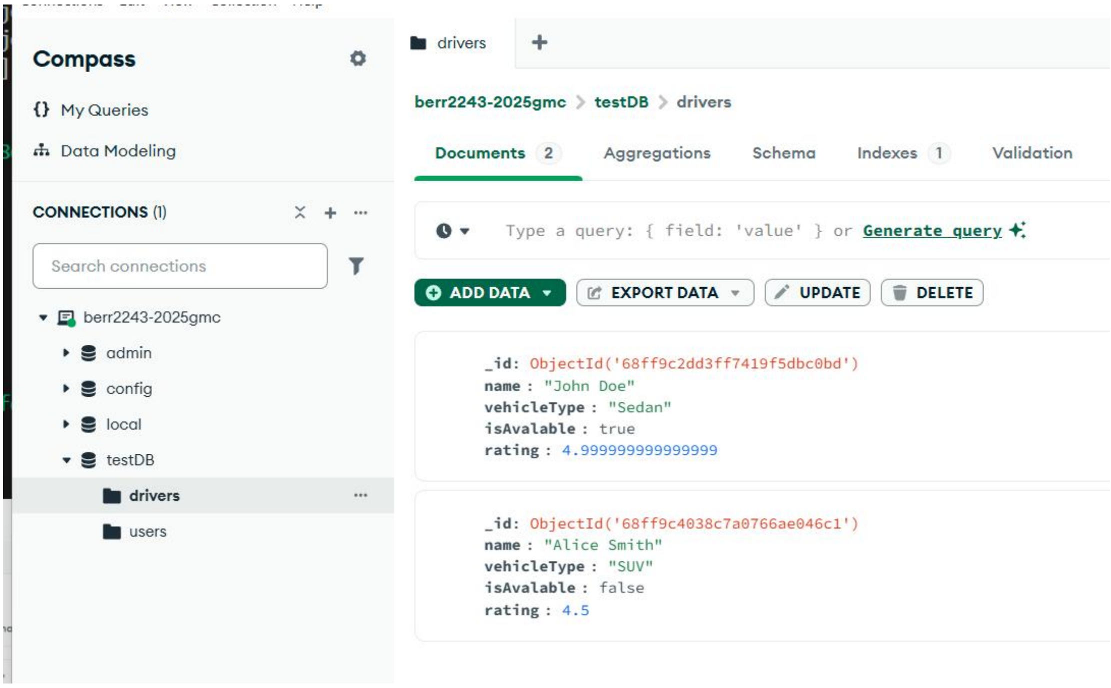
```
Connected to MongoDB!
Available drivers: [
  {
    _id: new ObjectId('68ff8cf10ef8a491a3f9060d'),
    name: 'John Doe',
    vehicleType: 'Sedan',
    isAvalable: true,
    rating: 4.8
  }
]
Document inserted!
Query result: {
  _id: new ObjectId('68ebb51c5e9887070f024fe9'),
  name: 'Goh Ming Chen',
  age: 21
}
```

Task 5 Increment 0.1 of John Doe’s rating from 4.8 to 4.89999…..95



```
]
Connected to MongoDB!
New driver created with result; [object Object]
New driver created with result; [object Object]
New driver created with result; [object Object]
New driver created with result; [object Object]
New driver created with result; [object Object]
Driver updated with result: [object Object]
Available drivers: [
  {
    _id: new ObjectId('68ff96989f6cb93e1068e6ff'),
    name: 'John Doe',
    vehicleType: 'Sedan',
    isAvalable: true,
    rating: 4.8999999999999995
  }
]
Document inserted!
Query result: {
  _id: new ObjectId('68ebb51c5e9887070f024fe9'),
  name: 'Goh Ming Chen',
  age: 21
}
```

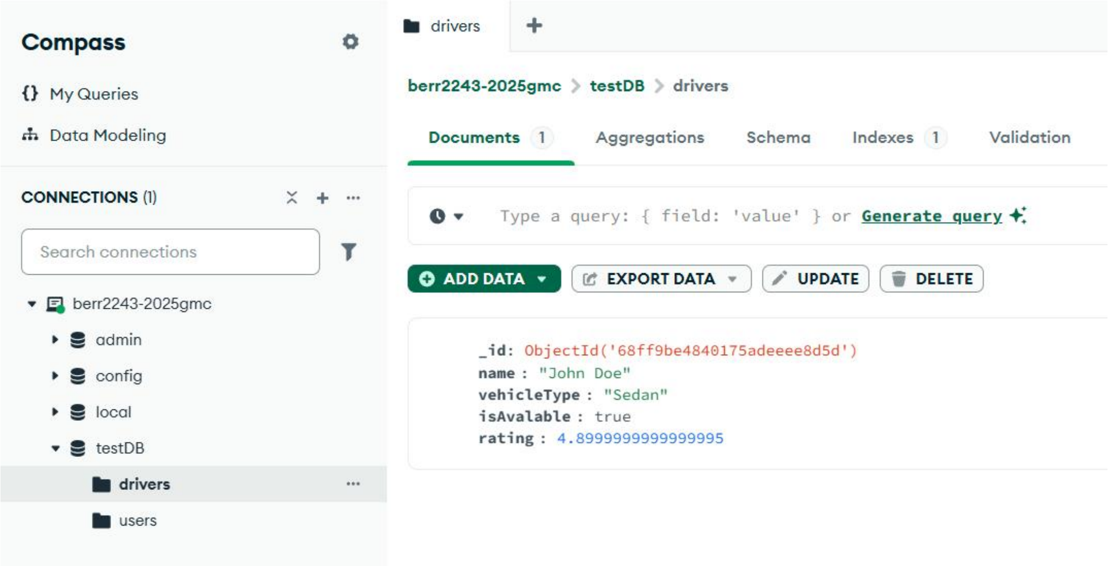
Task 6 Delete
Before



```
_id: ObjectId('68ff9c2dd3ff7419f5dbc0bd')
name : "John Doe"
vehicleType : "Sedan"
isAvalable : true
rating : 4.999999999999999
```

```
_id: ObjectId('68ff9c4038c7a0766ae046c1')
name : "Alice Smith"
vehicleType : "SUV"
isAvalable : false
rating : 4.5
```

After Delete



```
_id: ObjectId('68ff9be4840175adeeee8d5d')
name : "John Doe"
vehicleType : "Sedan"
isAvalable : true
rating : 4.8999999999999995
```

Exercise

1. Explain what is CRUD operations and how it is relates to the mongo functions in the exercise.

Operation	Description	MongoDB Function Used
Create	Adds new data into a collection	insertOne(), insertMany()
Read	Retrieves data from a collection	find(), findOne()
Update	Modifies existing data	updateOne(), updateMany()
Delete	Removes data from a collection	deleteOne(), deleteMany()

In Task 3, we used insertOne() : to insert/create new drivers.

In Task 4, we used find() to read and then find available drivers.

In Task 5, we used updateOne() to update existing data of a drivers.

In Task 6, we used deleteOne(): to delete a existing drivers.

2. Identify all the mongo operators used in the exercise, then explain the usage for Each.

Operator	Usage	Example	Description
$gte	Query Filter	{ rating: { $gte: 4.5 } }	Stands for “greater than or equal to”. Used to find drivers with a rating ≥ 4.5.
$inc	Update Modifier	{ $inc: { rating: 0.1 } }	Increases a field value by the specified amount. Used to increase John Doe’s rating by 0.1.

3. Replace the mongo functions in Task 5 to updateMany instead of updateOne, compare the diFerence based on the result in console and the mongo compass.

updateOne() affects only the first match, while updateMany() affects all matching documents.

4. Replace the mongo functions in Task 6 to deleteMany instead of deleteOne, compare the diFerence based on the result in console and the mongo compass.

deleteOne() removes only the first driverrs that matches the filter, while deleteMany() removes all matching drivers.