Experiment:1

Where, AND, OR & CRUD

- ✓ Create the collection
- ✓ Lode the document into collection. students.csv.

Where

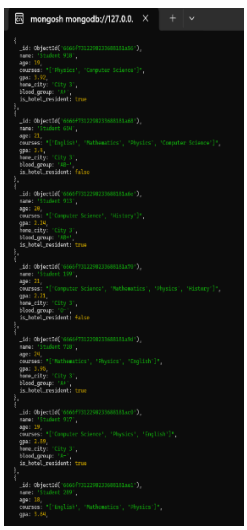
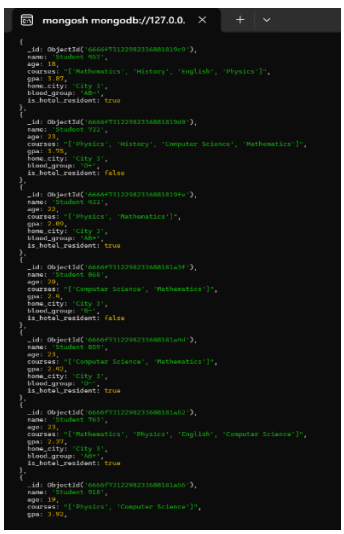
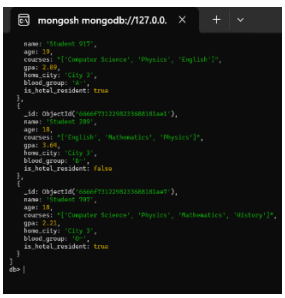
The `$where` operator in MongoDB allows you to use a JavaScript expression or function within your query to filter documents. It provides flexibility but has some drawbacks to consider.

```
// Find all students from "City 3"
db.students.find({ home_city: "City 3" });
```

- `db`: This refers to a database object that likely has been created elsewhere in the code.
- `.students`: This refers to a collection of documents within the database called “students”.
- `.find`: This is a method that is used to query the database collection.
- `({ home_city: "City 3" })`: This is a JavaScript object that specifies the query criteria. In this case, it’s looking for documents where the key `home_city` has a value of "City 3.

Output :

This would print the names of all students who live in "City 3", assuming each student document has a field called "name".

AND

The `$and` operator in MongoDB is used to perform a logical AND operation on multiple conditions within a query. It retrieves documents that satisfy **all** of the specified conditions.

```
// Find all students who are hotel residents OR have a GPA less than 3.
db.students.find({
  $or: [
    { is_hotel_resident: true },
    { gpa: { $lt: 3.0 } }
  ]
});
```

- `$and`: This keyword indicates the use of the AND operator.
- `[]`: This array contains all the individual conditions you want to apply.
- `{ condition1: value1 }`: This represents an individual condition. Here, `condition1` is the field name you're checking, and `value1` is the value you're comparing it to.
- You can add more conditions within the array, separated by commas.

Output :

```
db> db.student.find({ $and:[ {home_city:"City 5"},{blood_group:"A+"}]});
[
  {
    _id: ObjectId('6666f7312298233688181819c8'),
    name: 'Student 142',
    age: 24,
    courses: "['History', 'English', 'Physics', 'Computer Science']",
    gpa: 3.41,
    home_city: 'City 5',
    blood_group: 'A+',
    is_hotel_resident: false
  },
  {
    _id: ObjectId('6666f7312298233688181ae8'),
    name: 'Student 947',
    age: 20,
    courses: "['Physics', 'History', 'English', 'Computer Science']",
    gpa: 2.86,
    home_city: 'City 5',
    blood_group: 'A+',
    is_hotel_resident: true
  },
  {
    _id: ObjectId('6666f7312298233688181b5a'),
    name: 'Student 567',
    age: 22,
    courses: "['Computer Science', 'History', 'English', 'Mathematics']",
    gpa: 2.01,
    home_city: 'City 5',
    blood_group: 'A+',
    is_hotel_resident: true
  }
]
db>
```

In essence, this query targets students who are either hotel residents or have a GPA below 3.0, potentially returning a mix of both types of students.

OR

The `$or` operator in MongoDB is used to perform a logical OR operation on multiple conditions within a query. It retrieves documents that satisfy **at least one** of the specified conditions. Here's a breakdown of its usage:

```
// Find all students who are hotel residents OR have a GPA less than 3.
db.students.find({
  $or: [
    { is_hotel_resident: true },
    { gpa: { $lt: 3.0 } }
  ]
});
```

- `db`: This refers to a database object that likely has been created elsewhere in the code.
- `.students`: This refers to a collection of documents within the database called “students”.
- `.find`: This is a method that is used to query the database collection.
- `({ ... })`: This is a JavaScript object that specifies the query criteria. In this case, it’s looking for documents where either the `is_hotel_resident` field is true or the `gpa` field is less than 3.0, achieved using the `$or` operator.

Output :

This would print the names and GPAs of all students who are either hotel residents or have a GPA below 3.0, assuming each student document has fields named "name" and "gpa".

In essence, the output consists of documents from the `students` collection that satisfy at least one of the two conditions: being a hotel resident or having a GPA below 3.0.

CRUD

- C - Create / Insert
- R - Remove
- U - update
- D - Delete

Update

Given a collection you want to update new student update is used.

In MongoDB, the `update` functionality is used to modify existing documents within a collection

```
// Find a student by name and update their GPA
db.students.updateOne({ name: "Alice Smith" }, { $set: { gpa: 3.8 } });
```

The second line is the actual code. It updates a document in the `students` collection of a MongoDB database. Here's a breakdown of the code:

- `db.students.updateOne({...})`: This part targets the `students` collection within the database `db` and uses the `updateOne` method to modify a single document.
  - `({ name: "Alice Smith" })`: This specifies the query criteria to identify the document to be updated. It searches for a document where the `name` field is equal to "Alice Smith".
- `, ($set: { gpa: 3.8 })`: This part defines the update operation to be performed on the matching document.
  - `$set`: This update operator indicates that we want to set a field value.
  - `{ gpa: 3.8 }`: This specifies that the field to be updated is `gpa` and its new value should be set to 3.8.

Output :

```
db> db.student.updateOne({name:"Alice Smith"},{$set:{gpa:3.8}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
db>
```

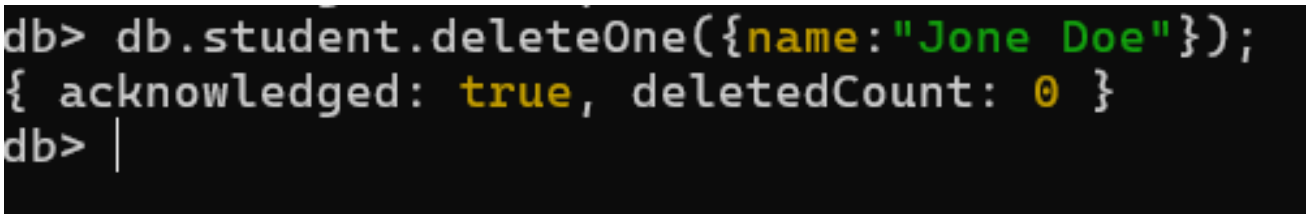
Delete

To delete particular detail delete is used.

```
// Delete a student by name
db.students.deleteOne({ name: "John Doe" });
```

- `db.students.deleteOne`:
  - `db`: This refers to a database object likely created elsewhere in your code. It represents the database you're connected to.
  - `.students`: Targets the collection named "students" within the database.
  - `.deleteOne`: This method is used to delete one document from the specified collection.
- `({ name: "John Doe" })`:
  - This JavaScript object defines the query criteria to identify the document to be deleted. It searches for a document where the field named "name" has the value "John Doe".

Output :



In essence, this code snippet removes the document from the "students" collection where the "name" field is equal to "John Doe".

Update Many

The `updateMany` method in MongoDB is used to update multiple documents within a collection that match a specified query criteria. It's a powerful tool for modifying data in bulk, especially when you need to update many documents based on certain conditions.

```
// Update all students with a GPA less than 3.0 by increasing it by 0.5
db.students.updateMany({ gpa: { $lt: 3.0 } }, { $inc: { gpa: 0.5 } });
```

- `db.students.updateMany`:
  - `db`: This refers to a database object likely created elsewhere in your code. It represents the database you're connected to.
  - `.students`: Targets the collection named "students" within the database.
  - `.updateMany`: This method is used to update multiple documents in the specified collection that match the filter criteria.
- `({ gpa: { $lt: 3.0 } })`:
  - This JavaScript object defines the query criteria to identify the documents that will be updated. It searches for documents where the field named "gpa" is less than (represented by <) 3.0.
- `, ($inc: { gpa: 0.5 })`:
  - This part defines the update operation to be performed on the matching documents.
  - `,`: This comma separates the query criteria from the update operation.
  - `$inc`: This update operator indicates that you want to increment the value of a field.
  - `{ gpa: 0.5 }`: This object specifies the field to update and the value to be added.
    - `gpa`: This is the field name you want to modify (GPA).
    - `0.5`: This is the value by which you want to increase the GPA.

Output :

```
db> db.studdent.updateMany({gpa:{$lt:3.0}},{$inc:{gpa:0.5}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
db>
```

In essence, this code snippet updates multiple documents in the "students" collection where the "gpa" field is less than 3.0. It uses the `updateMany` method and the `$inc` operator to increment the GPA by 0.5 for each matching student

Delete Many

To update delete many items at a time this deleteMany is used.

```
// Delete all students who are not hotel residents
db.students.deleteMany({ is_hotel_resident: false });
```

This line performs the deletion operation in MongoDB. Here's a breakdown:

- `db.students.deleteMany`:
  - `db`: This refers to a database object likely created elsewhere in your code. It represents the database you're connected to.
  - `.students`: Targets the collection named "students" within the database.
  - `.deleteMany`: This method is used to delete documents from the specified collection that match the filter criteria. Unlike `deleteOne`, it can delete multiple documents.
- `({ is_hotel_resident: false })`:
  - This JavaScript object defines the query criteria to identify the documents that will be deleted. It searches for documents where the field named "is_hotel_resident" has the value `false`.

Output :

```
db> db.student.deleteMany({ is_hostel_resident:false});
{ acknowledged: true, deletedCount: 0 }
db>
```

In essence, this code snippet removes all documents from the "students" collection where the "is_hotel_resident" field is set to `false`. This means it deletes all student documents that do not have hotel residency.