WHERE CLAUSE AND OR CRUD:

WHERE: Given a Collection you want to FILTER a subset based on a condition. That is the place WHERE is used.For queries that cannot be done any other way, there are "$where" clauses, which allow you to execute arbitrary JavaScript as part of your query. This allows you to do (almost) anything within a query. For security, use of "$where" clauses should be highly restricted or eliminated. End users should never be allowed to execute arbitrary "$where" clauses.

EXAMPLE:

```
db.products.find({
  $where: function() {
    // Calculate total price with discount
    const discount = this.price * this.discount / 100;
    const totalPrice = this.price - discount;

    // Return documents where total price is greater than 100
    return totalPrice > 100;
  }
})
```

```
test> db.stu.find({gpa:{$gt:3.5}}).count();
124
```

```
test> db.stu.find({home_city:"City 3"}).count();
34
```

AND:

In MongoDB, the `$and` operator is a powerful tool for constructing queries that select documents meeting all of a specified set of conditions

- **Logical AND:** It performs a logical AND operation, ensuring that a document is included in the results only if it satisfies **every** condition within the `$and` array.

```
test> db.stu.find({
... $and:[
...     {home_city:"City 5"},
...     {blood_group:"A+"}
... ]
... });
[
  {
    _id: ObjectId('6655e91dee1dcfb73e7398db'),
    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('6655e91eee1dcfb73e7399fb'),
    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('6655e91eee1dcfb73e739a6d'),
    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
  }
]
```

OR: Given a Collection you want to FILTER a subset based on multiple conditions but Any One is Sufficient.

● You can use this operator in methods like find(), update(), etc. according to your requirements.

```
test> db.stu.find({ $or: [ { is_hostel_resident: true }, { gpa: { $lt: 3.0 } } ] }).count()
261
test>
```

CRUD OPERATIONS:

C→CREATION / INSERT

R→REMOVE

U→UPDATE

D→DELETE

CREATE: This operation refers to adding a new piece of data to the storage system. For instance, creating a new user account in a database or adding a new document to a folder.

```
test> const studentData = {
...       "name":"Alice Smith",
...       "age":22,
...       "courses":["Mathematics","Computer Science","English"],
...       "gpa":3.8,
...       "home_city":"New York",
...       "blood_group":"A+",
...       "is_hotel_resident":false
...       };

test> db.stu.insertOne(studentData);
{
  acknowledged: true,
  insertedId: ObjectId('665b529e49389824aecdcdf7')
}
test>
```

READ: This operation involves retrieving existing data from the storage system. This could be fetching a specific user's information, displaying a list of files in a directory, or searching for data based on certain

```
test> db.stu.insertOne(studentData);
{
  acknowledged: true,
  insertedId: ObjectId('665b529e49389824aecdcdf7')
}
test> db.stu.insertOne({"name":"Bhagya"});
{
  acknowledged: true,
  insertedId: ObjectId('665b549349389824aecdcdf8')
}
test> db.stu.find({"name":"Bhagya"});
[ { _id: ObjectId('665b549349389824aecdcdf8'), name: 'Bhagya' } ]
test> db.stu.find().count();
505
test>
```

UPDATE: This operation allows you to modify existing data within the storage system. Editing a user's profile details, changing the content of a document, or updating the status of an order are all examples of update operations.

```
test> db.stu.updateOne({name:"Alice Smith"},{$set:{gpa:3.8}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
test>
```

DELETE: This operation permanently removes data from the storage system. Deleting unwanted files, removing inactive user accounts, or purging outdated records from a database are all examples of delete operations.

```
test> db.stu.deleteOne({name:"John Doe"});
{ acknowledged: true, deletedCount: 0 }
test>
```