DATA TYPES : MongoDB Server stores data using the BSON format which supports some additional data types that are not available using the JSON format.

- Date
- Int32
- Decimal
- Timestamp

DATE :

- `Date()` method which returns the current date as a string.
- `new Date()` constructor which returns a `Date` object using the `ISODate()` wrapper.
- `ISODate()` constructor which returns a `Date` object using the `ISODate()` wrapper.

Int32 :

- If a number can be converted to a 32-bit integer, `mongosh` will store it as `Int32`. If not, `mongosh` defaults to storing the number as a `Double`. Numerical values that are stored as `Int32` in `mongosh` would have been stored by default as `Double` in the `mongo` shell.The Int32() constructor can be used to explicitly specify 32-bit integers.

```
test> db.types.insertOne(
...     {
...         "_id": 1,
...         "value": Int32("1"),
...         "expectedType": "Int32"
...     }
... )
{ acknowledged: true, insertedId: 1 }
test>
```

Long : Long() constructor can be used to explicitly specify a 64-binteger.

```
test> db.types.insertOne(
...     {
...         "_id": 3,
...         "value": Long("1"),
...         "expectedType": "Long"
...     }
... )
{ acknowledged: true, insertedId: 3 }
test>
```

Decimal128 : Decimal128() values are 128-bit decimal-based floating-point numbers that emulate decimal rounding with exact precision.

```
test> db.types.insertOne(
...     {
...         "_id": 5,
...         "value": Decimal128("1"),
...         "expectedType": "Decimal128"
...     }
... )
{ acknowledged: true, insertedId: 5 }
test>
```

Timestamp : MongoDB uses a BSON Timestamp internally in the oplog. The `Timestamp` type works similarly to the Java Timestamp type. Use the Date type for operations involving dates. A `Timestamp` signature has two optional parameters.

WHERE CLAUSE,AND,OR&CRUD

LOAD THE DOCUMENT

- Download the student csv from this link

- **Import the data to the collection created link**

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.**

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

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

AND : **Given a Collection you want to FILTER a subset based on multiple conditions.This operator is used to perform logical AND operation on the array of one or more expressions and select or retrieve only those documents that match all the given expression in the array.**

- **This operator performs short-circuit evaluation.**
- **If the first expression of $and operator evaluates to false, then MongoDB will not evaluate the remaining expressions in the array.**
- **You can also use AND operation implicitly with the help of comma(,).**

```
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.**
- **You can also use this operator with text queries, GeoSpatial queries, and sort operations.**

- **When MongoDB evaluating the clauses in the $or expression, it performs a collection scan.**

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


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