Episode - 12

## Logical DB Query & compound Indexes.

1) How you reference a objectId in Schema ?

-> the type of object Id is "mongoose. Schema. Types. ObjectId"

eg:
```
const userSchema = new mongoose. Schema ({

        userId: {
            type: mongoose. Schema. Types. ObjectId
        }
})
```

2) How to compare objectId with the string version of id?

-> there is function called "equals()" which takes string version id & compare it with object Id. whether both are equal (or) not

eg:
```
ObjectId. equals ( string id)
```

3) What are the types of middleware in Mongoose?

Mongoose middleware :

→ Mongoose middleware (also known as "hooks") allow us to run some logic before (or) after certain actions such as save, update, delete ~~are~~ in documents

→ we can also validate the documents using this middlewares.

Types of Mongoose middleware :

*) pre Middleware ⇒ Runs before certain action occurs

*) post middleware ⇒ Runs after certain action occurs.

Use cases of these Middleware :

*) Pre - Save Middleware : used for hashing password before saving to DB.

*) post - Save Middleware : Logging changes after saving a document

*) Pre - remove middleware : clean -up data before removing a doc.

*) post - remove middleware : notify that doc is deleted.

4) How to create a Mongoose Middleware?

→ we attach the pre (or) post middleware function in the Schema

→ this middleware fn accept 2 arg : 1st → action 2nd → fn

→ Arrow fn is not allowed ~~as~~ it does not have "this" keyword.

```
    Schema.type ("action", fn() { })
eg: userSchema.pre ("find", function (next) {
        this.where ({ isActive : true }) → find active users
        next()
    })
```

→ Don't forget to call next as it is a middleware .

4) what is Indexing and why we need it?

-> Indexing improves the efficiency of querying operation.

-> when a collection grows querying for single doc by scanning the larger data makes the querying Process slow & inefficient. thus Index allow us to quickly locate & retreive that single doc from the largest collection.

Index:

-> Indexes store a small portion of the collection's data in a way that makes search faster.

-> It is used for query optimization, sorting, uniqueness.

5) what are the type of Indexes in MongoDB?

*) Single field Index: created on single field.

*) Compound Index: created on multiple field

*) Multikey Index: Supports indexing array fields

*) Text Index: used for full-text search

*) Hashed Index: used for sharding in MongoDB

*) Unique Index: ensure that indexed fields have unique values.

6) How to create a Index?

-> we define index in the schema either by using "index()" method (or) as options in schema field definitions

eg: using index() method

```
const userSchema = new mongoose.Schema({
        name : String,
        age : Number
})
```

```
//compound index on name & age field
userSchema.index({name : 1, age: -1})
const user = mongoose.model("User", userschema) .
```

→ 1 → ascending order
→ -1 → descending order.

eg: Mongoose option

```
const userSchema = new. mongoose. Schema ({
        name: { type: String,
                index: true        → single field index
              },
        email: { type: String,
                 unique: true      → unique index.
               }
})
```

---