The Gentleman Coder

MongoDB

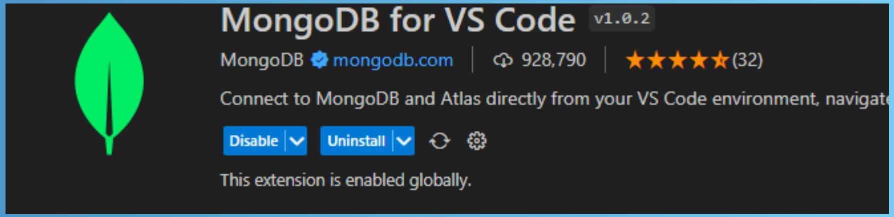
https://account.mongodb.com/account/register

www.justcoder.co.uk

VSc & Connect to the database

VSCode

You will need the MongoDB plugin to be able to connect to the DB.



MongoDB for VS Code v1.0.2

MongoDB mongodb.com | 928,790 | ★★★★★(32)

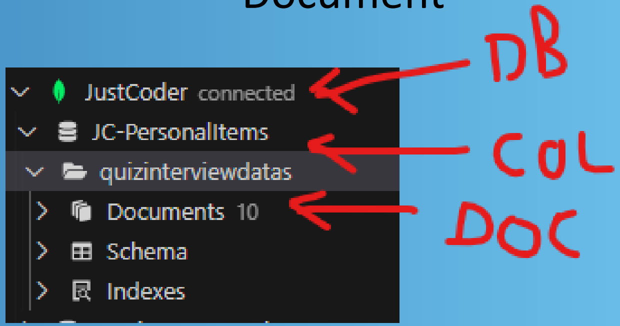
Connect to MongoDB and Atlas directly from your VS Code environment, navigate

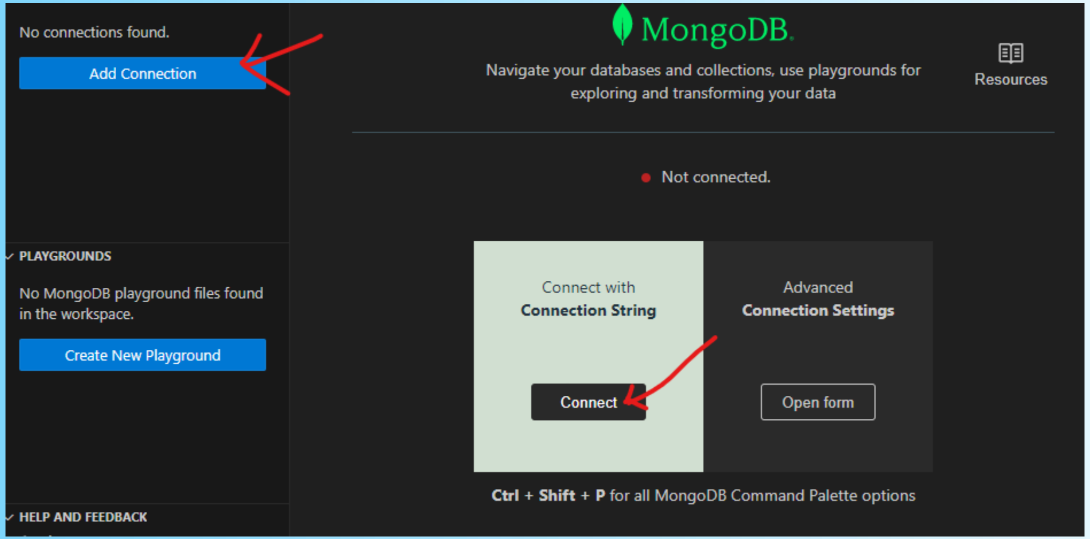
This extension is enabled globally.

MongoDB

The structure is:

- Database
    - Collection
        - Document




No connections found.
Add Connection
PLAYGROUNDS
No MongoDB playground files found in the workspace.
Create New Playground
HELP AND FEEDBACK
MongoDB
Navigate your databases and collections, use playgrounds for exploring and transforming your data
Resources
Not connected.
Connect with Connection String
Connect
Advanced Connection Settings
Open form
Ctrl + Shift + P for all MongoDB Command Palette options

Connecting

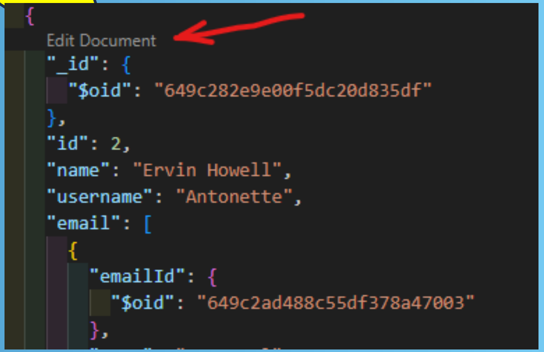
You can connect to a local instance you will need to download a local server for this: https://www.mongodb.com/try/download/community

Else we will connect to cloud I am using Atlas here and the connect string looks something like this: mongodb+srv://user:password@DBName.vwoam6e.mongodb.net/

Setting up Playground

The **Great** thing about MongoDB is "…", The **Bad** thing about MongoDB is "…"

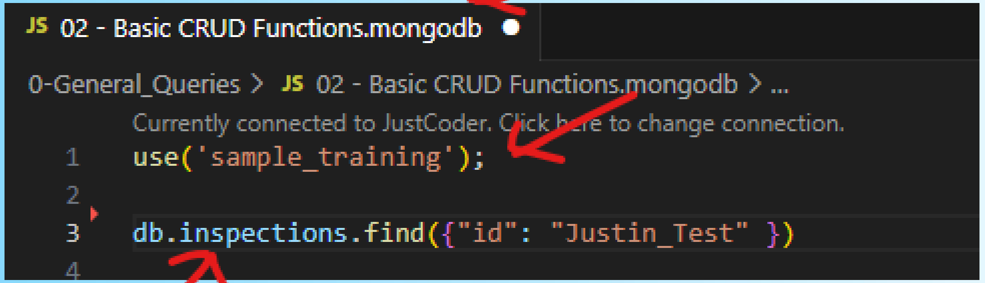
You can edit the DB directly in the result



```
{
  Edit Document
  "_id": {
    "$oid": "649c282e9e00f5dc20d835df"
  },
  "id": 2,
  "name": "Ervin Howell",
  "username": "Antonette",
  "email": [
    {
      "emailId": {
        "$oid": "649c2ad488c55df378a47003"
      },
```

Playground

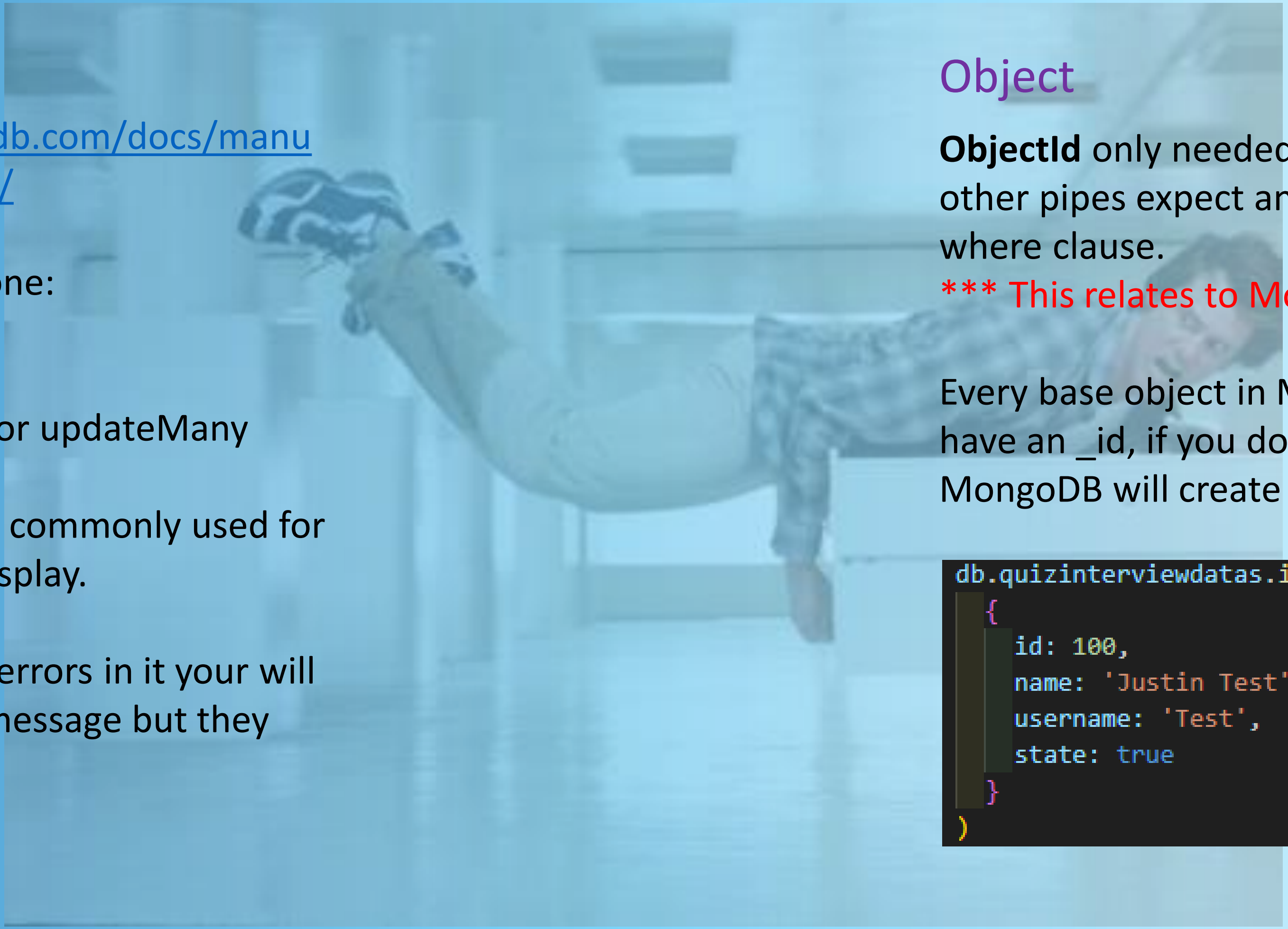
- File extension is .mongodb
- Declare the DB, then the collection and add the pipe.
- Any code you have written in the live environment you can just paste here (i.e. Node.js). I would often build my queries up here and then move to the node backend.
- The result is limited, to get all the data back you need to add '.toArray()' to the end of the pipe.
- You can edit the returned result of the DB directly in VSc



```
JS 02 - Basic CRUD Functions.mongodb
0-General_Queries > JS 02 - Basic CRUD Functions.mongodb > ...
Currently connected to JustCoder. Click here to change connection.
use('sample_training');

db.inspections.find({"id": "Justin_Test" })
```

Pipes/Methods and ObjectId



Pipes

https://www.mongodb.com/docs/manual/reference/method/

Find is the different one:

- find or findOne
- Normally
    - updateOne or updateMany

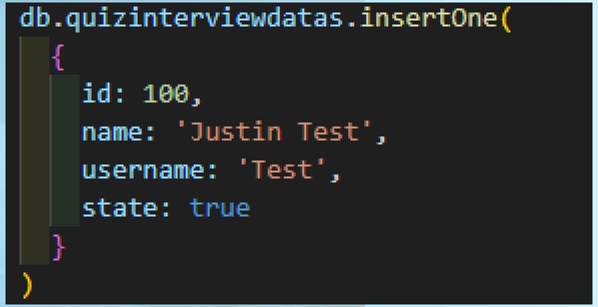
- Aggregation most commonly used for data collection/display.

- If your query has errors in it your will receive an error message but they are not the best.

Object

ObjectId only needed in aggregation, the other pipes expect an objectId in the where clause.
*** This relates to Mongoose ***

Every base object in MongoDB has to have an _id, if you do an insert without it MongoDB will create it for you.



```
db.quizinterviewdatas.insertOne(
  {
    id: 100,
    name: 'Justin Test',
    username: 'Test',
    state: true
  }
)
```

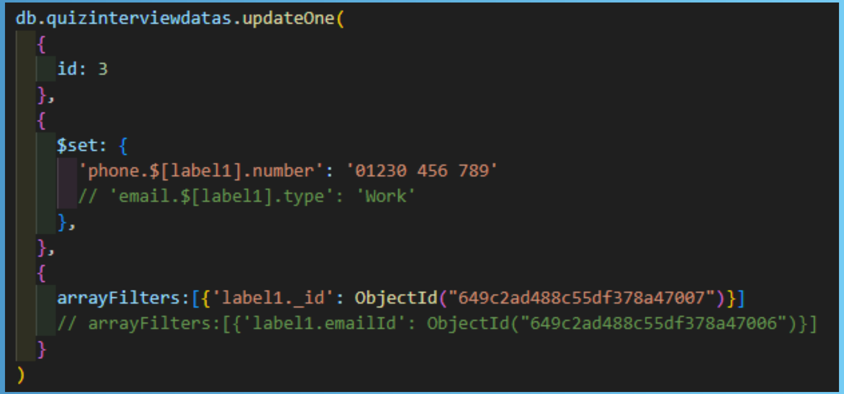
Array Filters

Nested Arrays

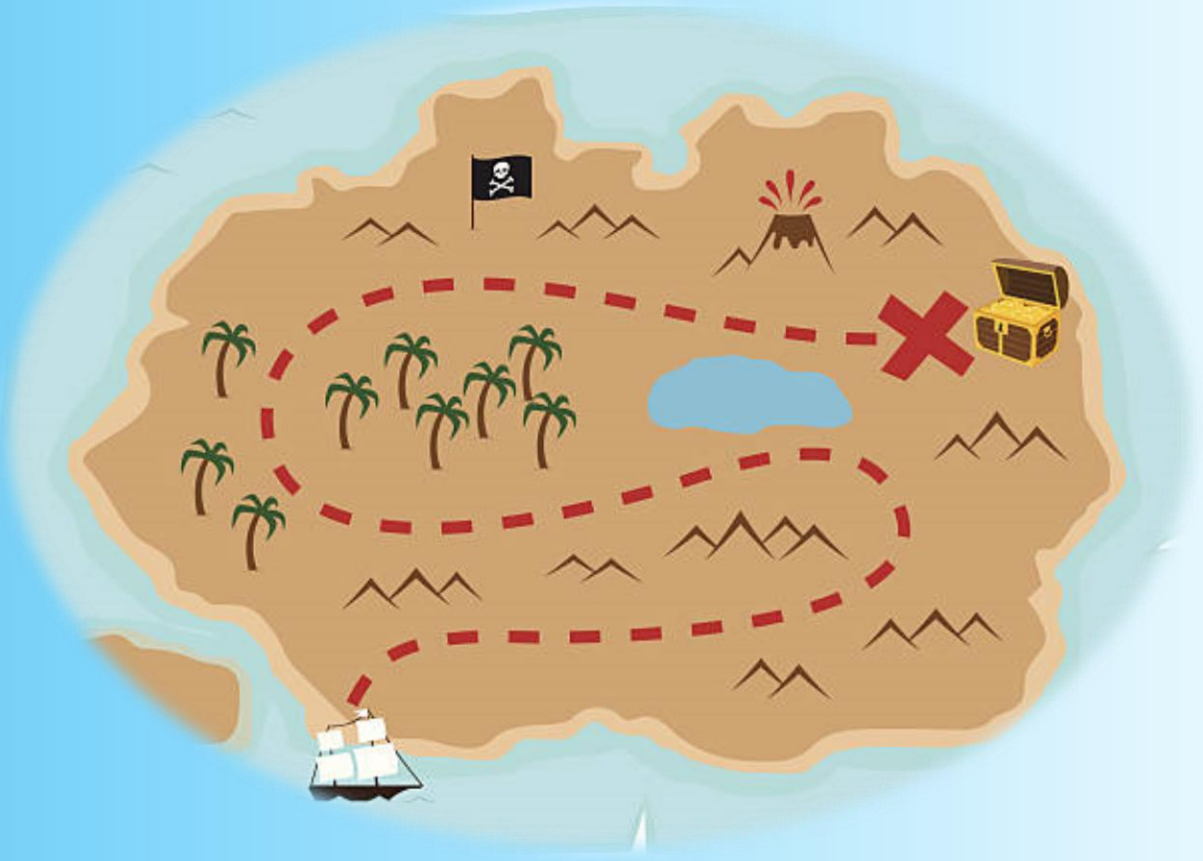
Biggest challenge I found in a document language is getting to the data you want to update as it can quite often end up nested.

- ArrayFilters will find any matching ID with the key name you declare.
- Used in all pipes except aggregation.
- You can use them multiple times in one query for nth time something is nested.

arr1.$[label1].arr2.$[label2].arr3.$[label3].arr4'



```
db.quizinterviewdatas.updateOne(
  {
    id: 3
  },
  {
    $set: {
      'phone.$[label1].number': '01230 456 789'
      // 'email.$[label1].type': 'Work'
    },
  },
  {
    arrayFilters:[{'label1._id': ObjectId("649c2ad488c55df378a47007")}]
    // arrayFilters:[{'label1.emailId': ObjectId("649c2ad488c55df378a47006")}]
  }
)
```



Aggregation Pipeline

Methods to flatten the data

- The main principle I used was just to bring back the data you need in as flatter shape as you can. Else it is just wasted bandwidth for data that never gets used.
- This will mean less code on the front end and easier to work with.
- **$unwind –** This is used to flatten the array, you will get a row per entry in the array.
- **$project –** A lot like using the 'as' operator in SQL when renaming columns of data. This is helpful for nested array items as you can remove the need to use dot notation. *Once renamed you can refer to the new name in the rest of the pipeline.*

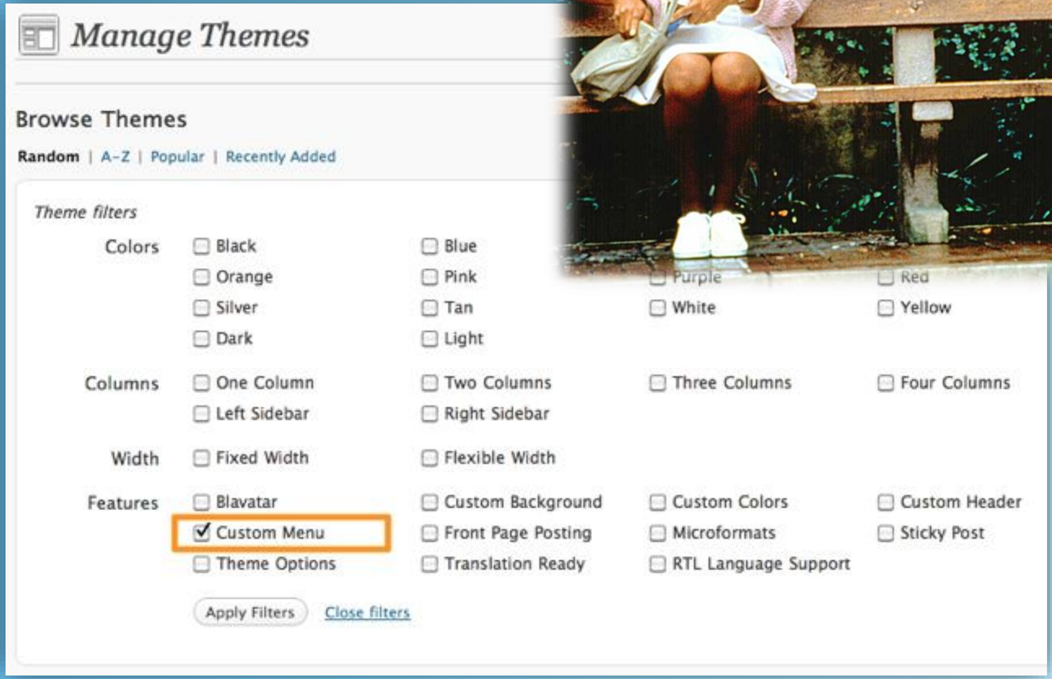
- **$group –** A lot like a 'subquery' in SQL where you need a calculation on the data i.e. sum.
- **$lookup –** A lot like a 'JOIN' in SQL, connecting 2 or more collections together.

Dynamic Aggregation Query

Life is like a box of chocolates

- This is where I wanted the ability to be able to filter the data sometimes and sometimes not.
- I sent an object that contained the data I wanted to filter. You then allocate this to a variable containing an mongoDB expression.
- You can then put the variable inline in the aggregation.
- I used this a lot in reporting where it was large datasets. I could get the user to provide these filter items, pass it to the BE, process on the BE and only bring back what is needed.

Filter Example:



Manage Themes

Browse Themes

Random | A-Z | Popular | Recently Added

Theme filters

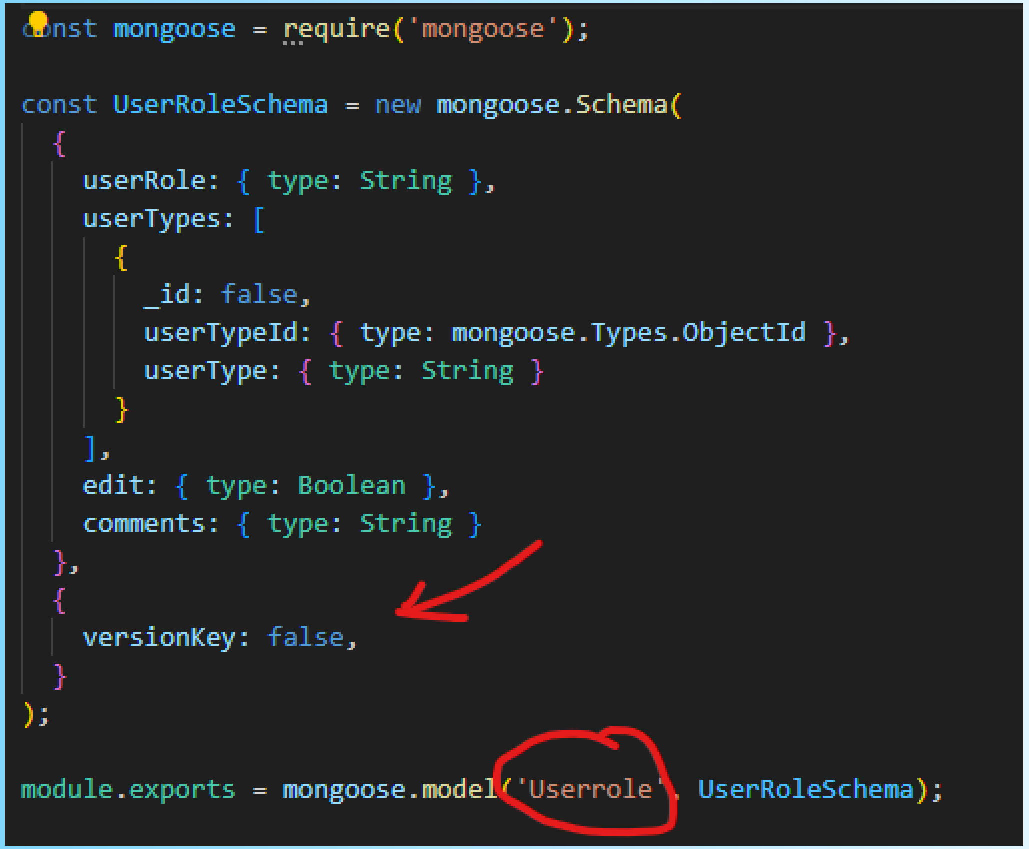
Colors	☐ Black	☐ Blue		
	☐ Orange	☐ Pink	☐ Purple	☐ Red
	☐ Silver	☐ Tan	☐ White	☐ Yellow
	☐ Dark	☐ Light		
Columns	☐ One Column	☐ Two Columns	☐ Three Columns	☐ Four Columns
	☐ Left Sidebar	☐ Right Sidebar		
Width	☐ Fixed Width	☐ Flexible Width		
Features	☐ Blavatar	☐ Custom Background	☐ Custom Colors	☐ Custom Header
	[✓ Custom Menu]	☐ Front Page Posting	☐ Microformats	☐ Sticky Post
	☐ Theme Options	☐ Translation Ready	☐ RTL Language Support	

Apply Filters Close filters

Mongoose & Principles

Some quirks of Mongoose

- You only need to use the 'ObjectId' method in the aggregation pipeline. All the others you can just used a string.
- To connect the collection to the model:
  - Use a capital letter for the first letter and none of the rest.
  - Drop the 's' off the end of the collection name.
- The version of the model can be tracked but I always stopped this by setting the versionkey: false



```
const mongoose = require('mongoose');

const UserRoleSchema = new mongoose.Schema(
  {
    userRole: { type: String },
    userTypes: [
      {
        _id: false,
        userTypeId: { type: mongoose.Types.ObjectId },
        userType: { type: String }
      }
    ],
    edit: { type: Boolean },
    comments: { type: String }
  },
  {
    versionKey: false,
  }
);

module.exports = mongoose.model('Userrole', UserRoleSchema);
```

https://github.com/JC-YouTube-Videos/mongoDB_VSc