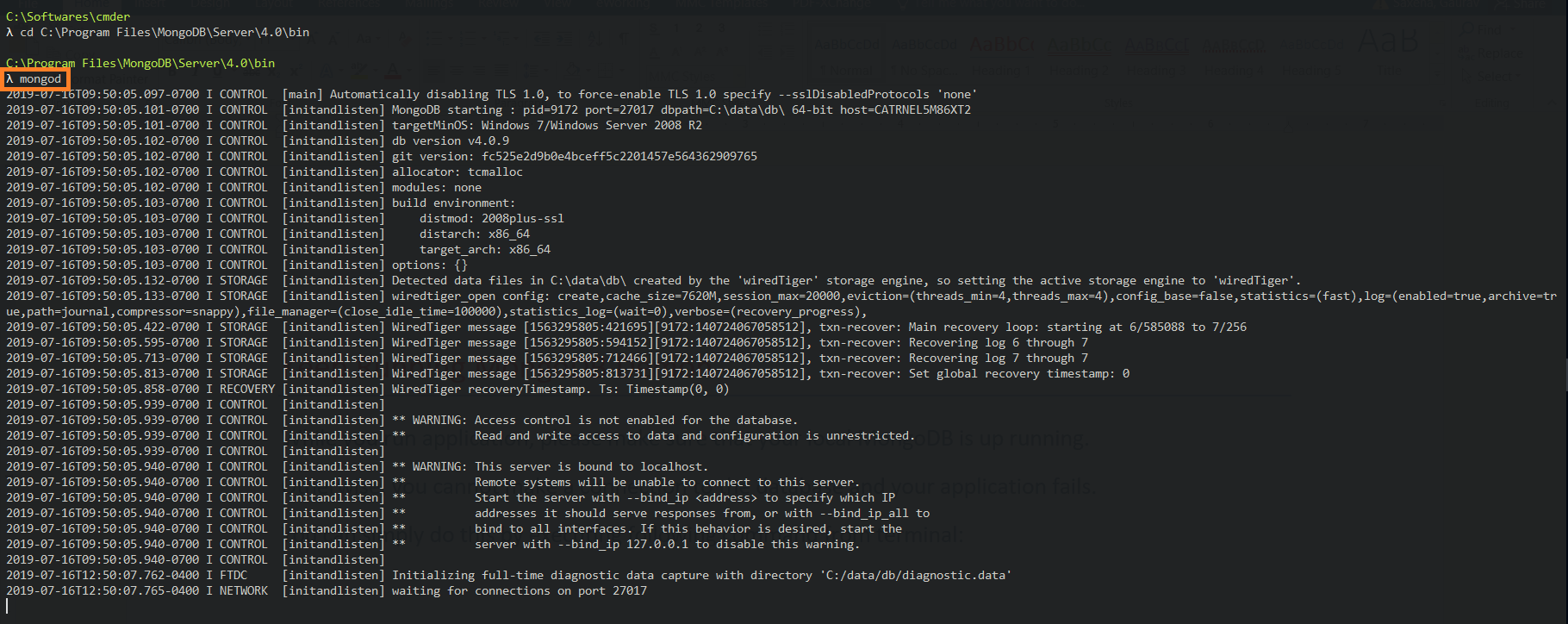
# Keep Running MongoDB Demon

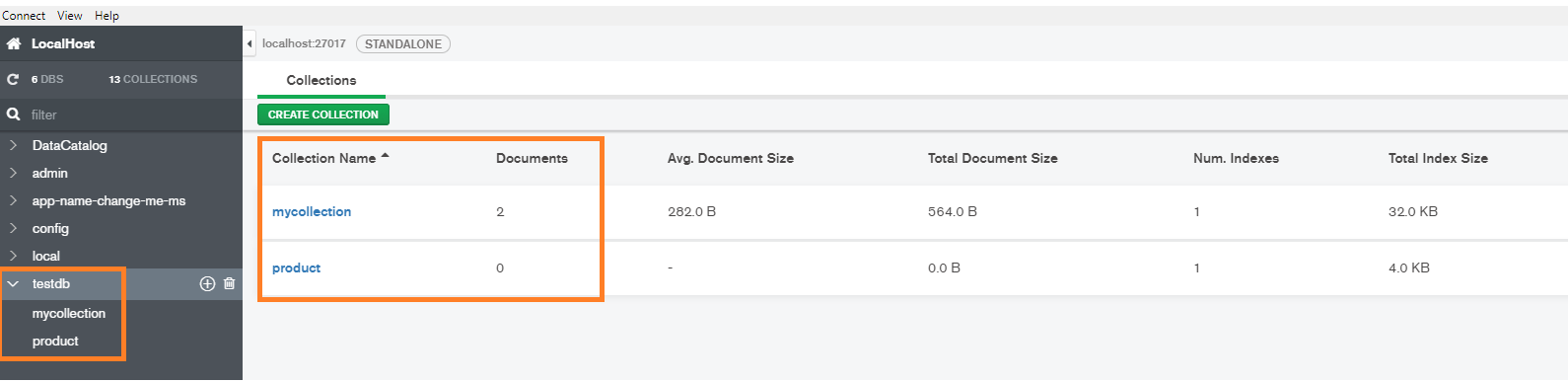
While you run application, please make sure that your local MongoDB is up running.

Otherwise, you cannot make a connection to the database and your application fails.

You can simply do this by executing following command from terminal:



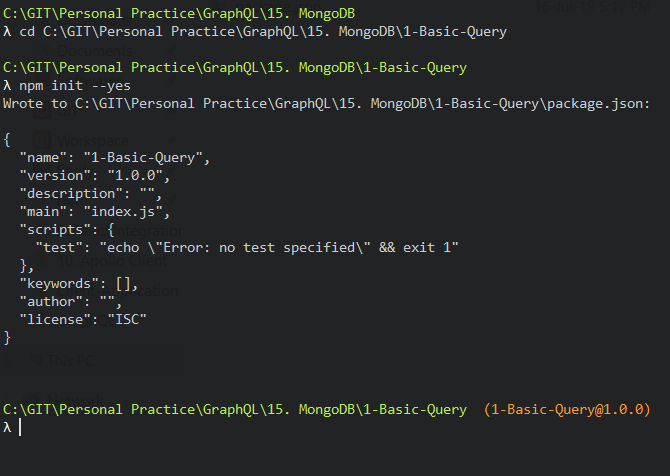
Check the DB and Collections in MongoDB



# Initialize Project and Install Packages

First of all, we need to create package.jsonfile to install dependencies.

Open your terminal and run this command. It will automatically create a package.json file for you.



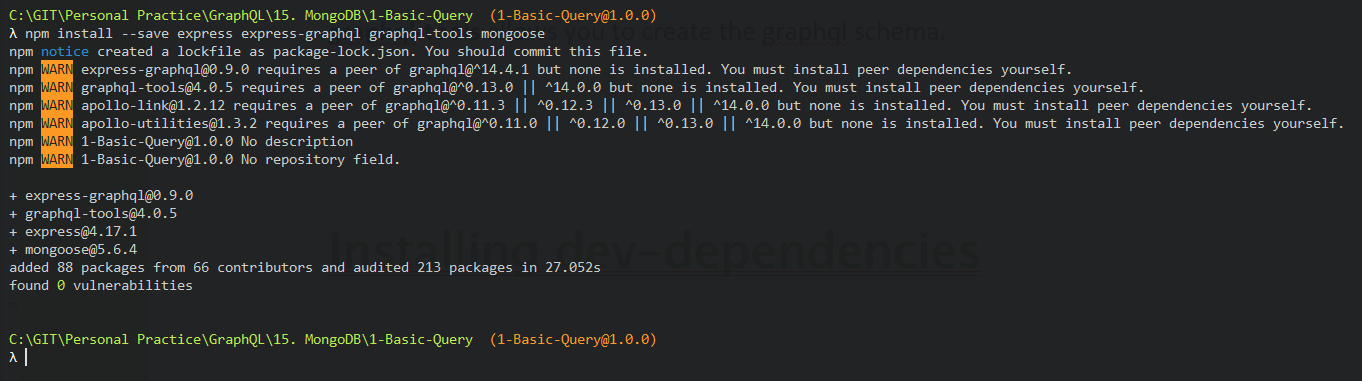
## Installing dependencies

We will be using express framework to build our APIs.

Express provides another **express-graphql**package to work with GraphQL.

We are going to use Mongoose ODM to interact with MongoDB database.

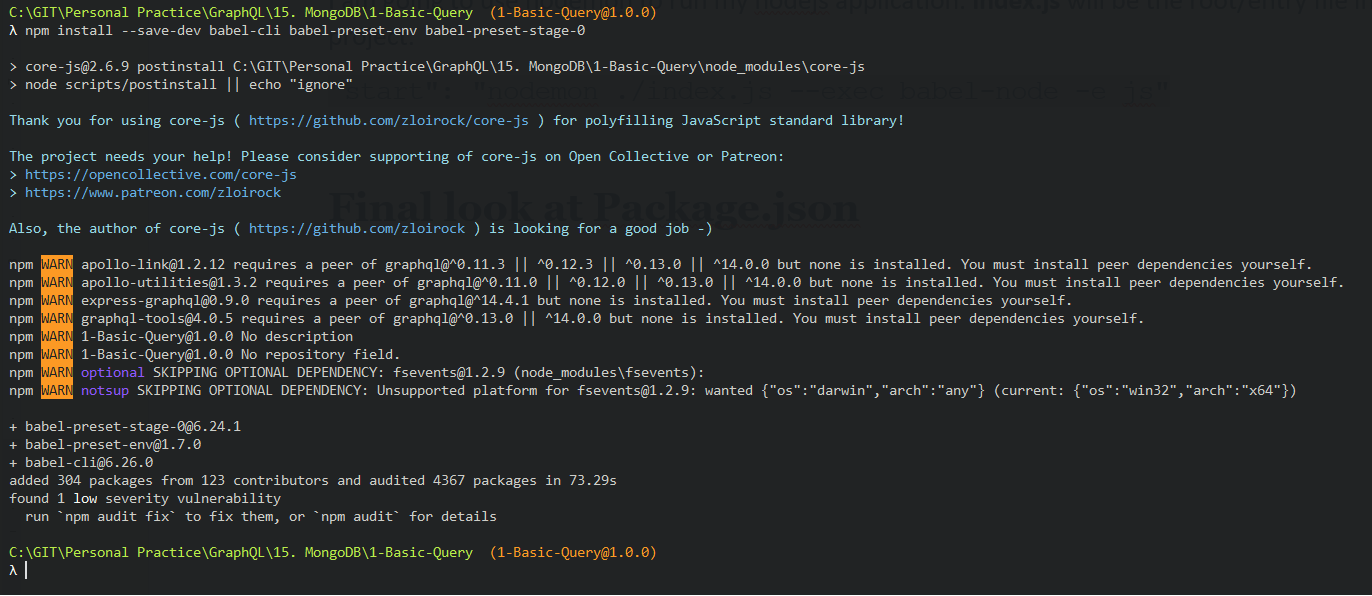
**graphql-tools** allows you to create the graphql schema.



## Installing dev-dependencies

We will be using ES6 in this example.

We need a babel to transpile our ES6 code to ES5.



Now you need to create a new **start**script in the package.json file to run the nodejs server.

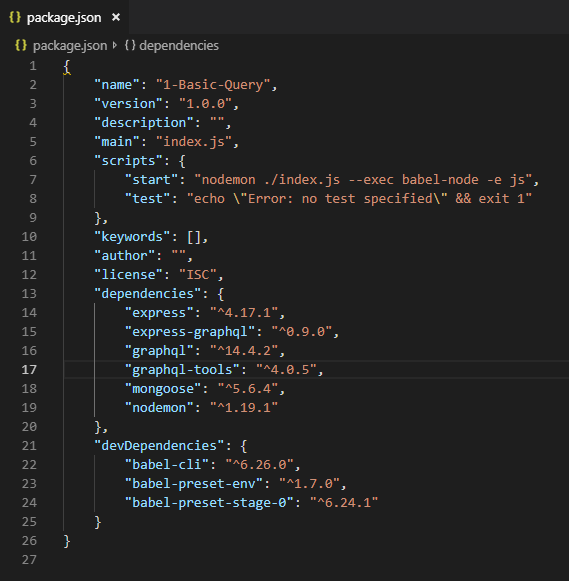
We are going to use nodemon to run my nodejs application.

**index.js**will be the root/entry file in our project.

Add below line to package.json

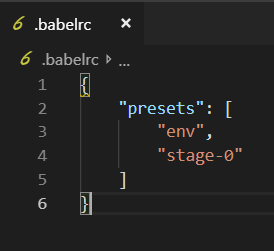
"start": "nodemon ./index.js --exec babel-node -e js"

## Final look at Package.json



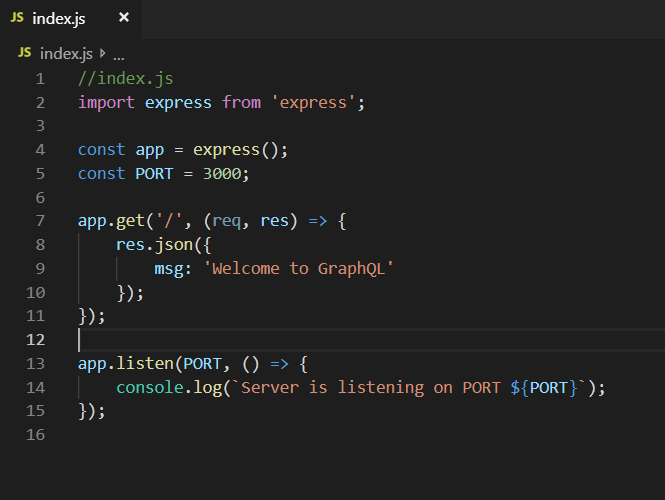
Creating .babelrc file

Open the project in your code editor and create new file .babelrc to in the root directory.



# Setting up ExpressJs Server

Create a new index.js file in the root directory of your project.

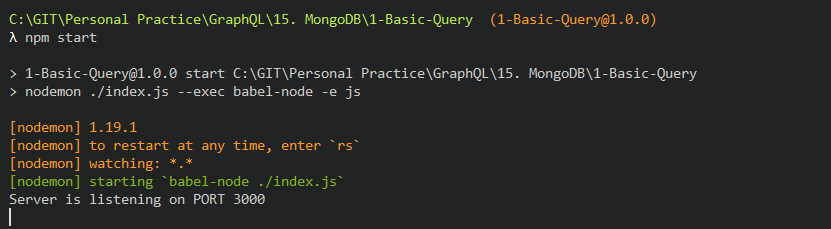


We have set up a basic express server.

Open your terminal and run this command to start the server.

## Running Server

We have set up a basic express server. Open your terminal and run this command to start the server.

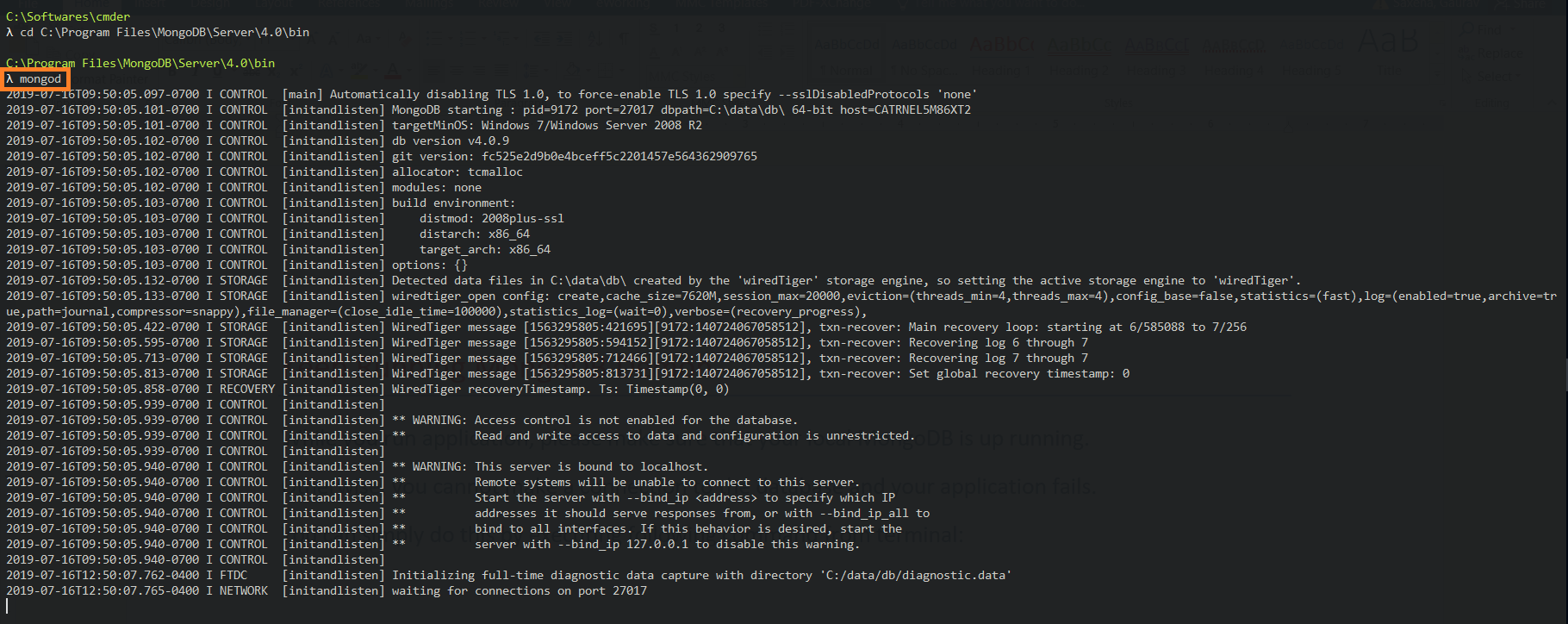


# Adding MongoDB to our Project

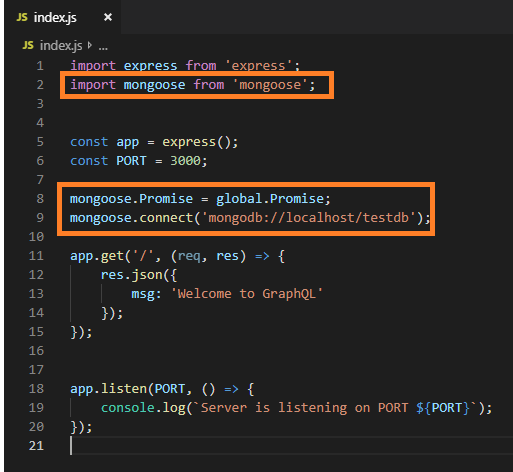
Let us integrate mongoose with Nodejs.

We have already installed mongoose package in our project. Make sure your MongoDB is running on your machine

Open your terminal and start the MongoDB server



**Adding Configuration for Mongoose**

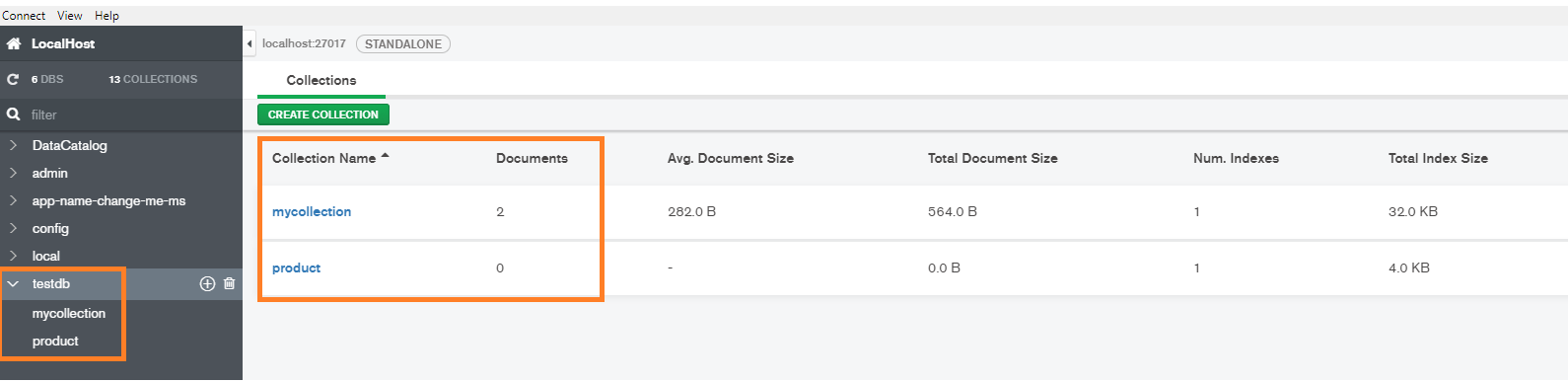


We have imported the mongoose package.

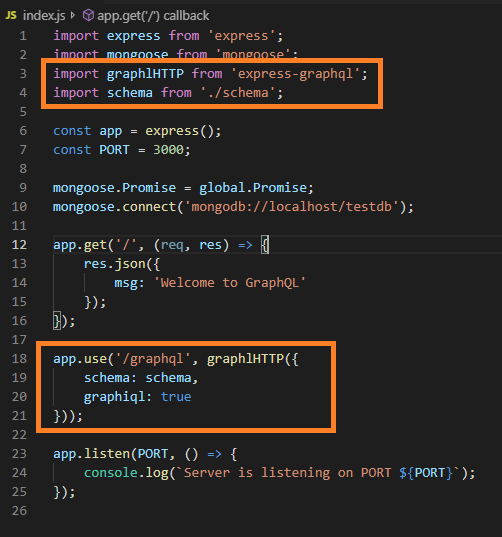
We want to use Promise with Mongodb so we have set the **mongoose.Promise**to **global Promise** object.

Mongoose allows us to connect with MongoDB using **connect()**method.

The name of my database is **testdb.**You can choose any database name.



# **Setting Up GraphQL Server**



Let see how to setup GraphQL server with Nodejs.

We are going to use the **express-graphql**package to interact with GraphQL.

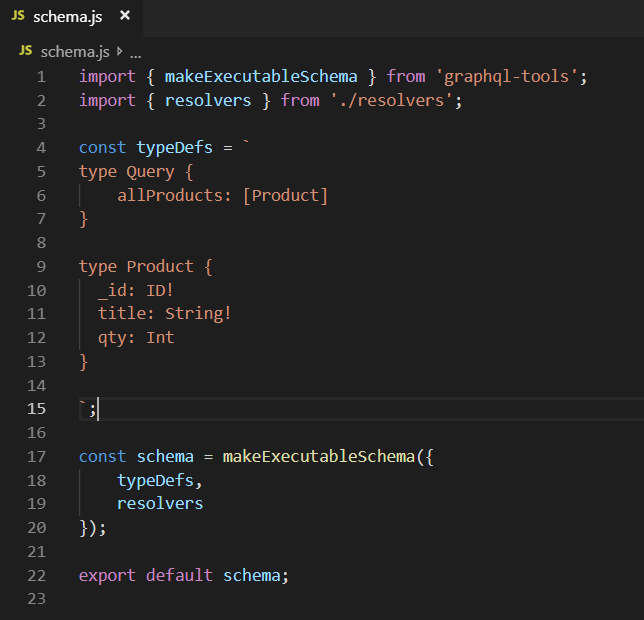
We have created the **graphql**middleware.

When you send the Graphql query or mutation request, then this middleware will be called and run the GraphQL operation.

You must provide **schema** in the **graphqlHTTP.**

**Creating GraphQL Schema**

Create a new schema.js file in the root directory.



We have defined the **typeDefs**in the**schema.js** file.

We have defined a new **Product**type in this file.

We also created a root Query type.

When you try to fetch all the products It will run the**allProducts** resolver against this Query.

**Creating allProducts Resolver**

Now we will need to create a new resolver file inside the root directory.

We are going to place all my resolvers inside the resolvers.js file.



When we try to execute the allProducts query, It will run this resolver and find all the products from the MongoDB database.

Resolvers also work with Promises so we can use **Async/Await**with resolver function.

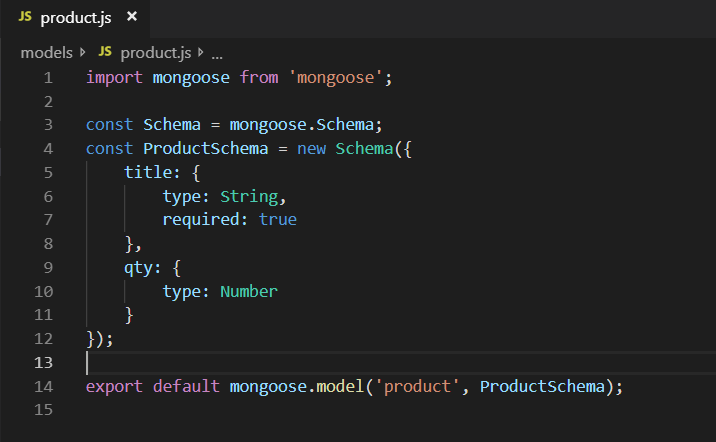
Each resolver function will return a Promise

## **Creating Product model**

We have imported the product model in resolvers.js, but have not create it.

Let’s create the Product model inside the models directory.

Create a new **model’s**directory and add new file product.js.

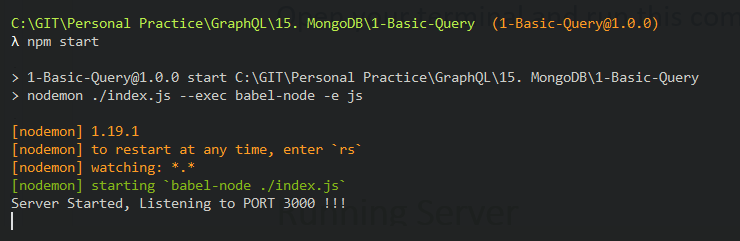


## Testing: Sending Query from GraphiQL

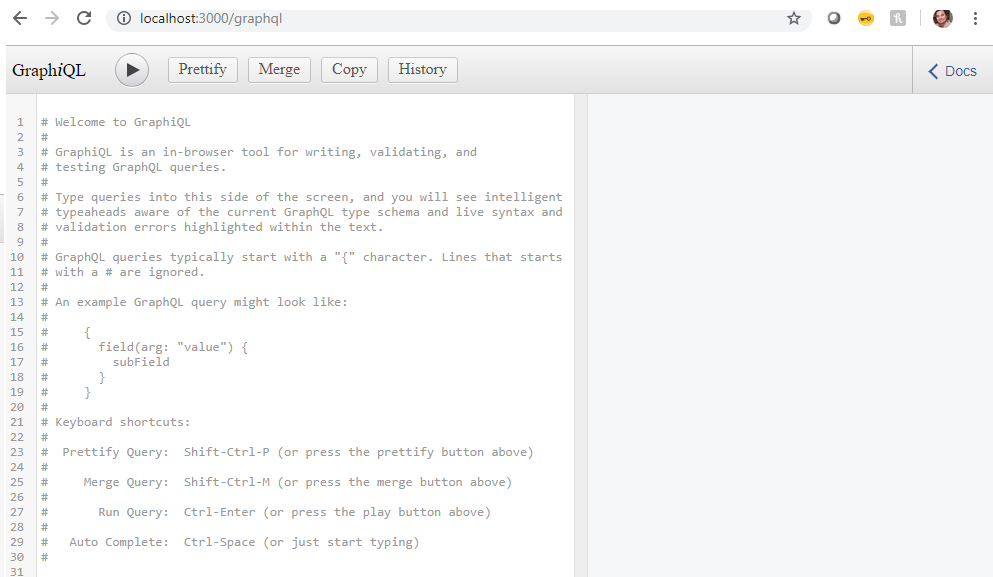
Now it’s time to run the Application by using npm start and go to this endpoint.

[**http://localhost:3000/graphql**](http://localhost:3000/graphql)**.**

**Start application**



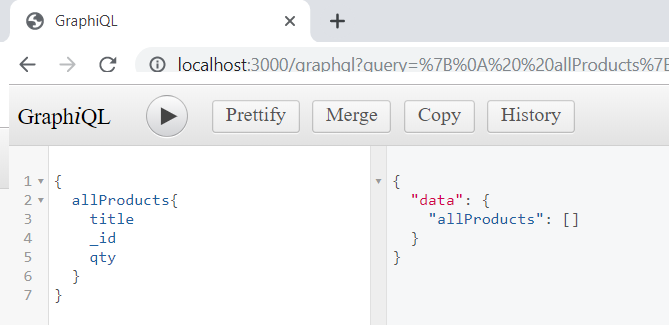
**Open browser and test**



**Testing query**

We are going to execute **allProducts** query to fetch all the products.

We don’t have any product in our database. We need mutation to create a product in our database.

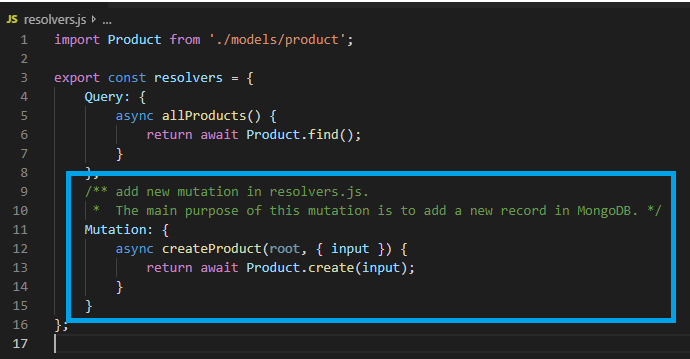


# Adding Product

**Step 1: Adding createProduct Mutation in Resolvers.js**

We are going to add new mutation in resolvers.js.

The main purpose of this mutation is to add a new record in MongoDB.

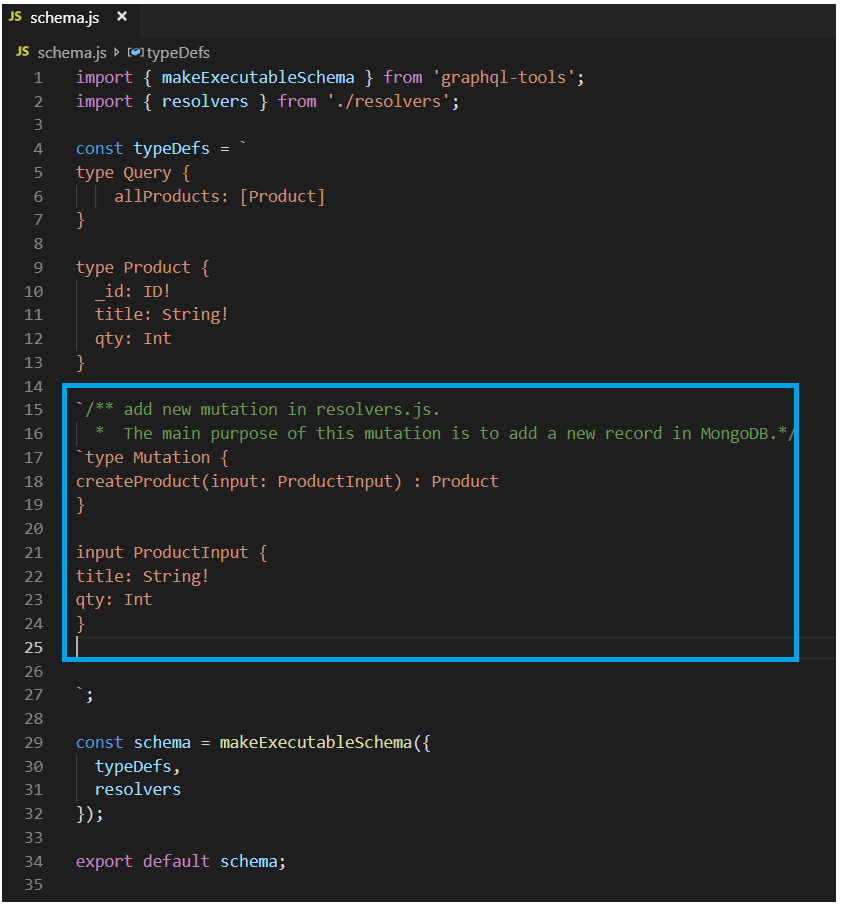


**Step 2: Adding createProduct Mutation in Schema.js**

This time we need to create a new input type which is **ProductInput.**

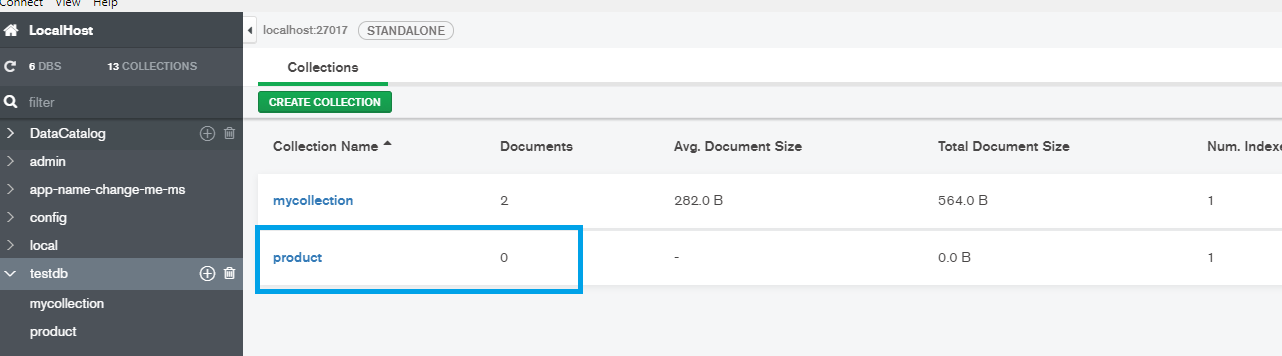
You need to pass the **input**argument while sending createProduct mutation.

The return type of this mutation is Product.

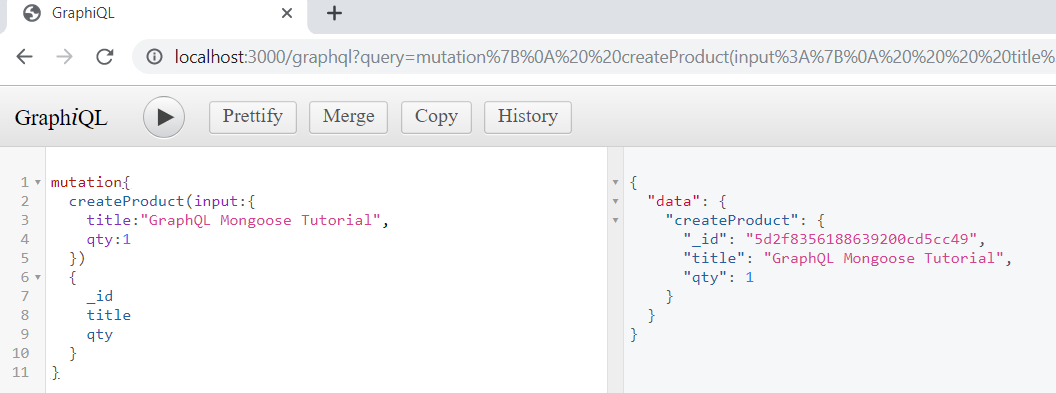
****

## Testing: Sending Mutation Query from GraphiQL

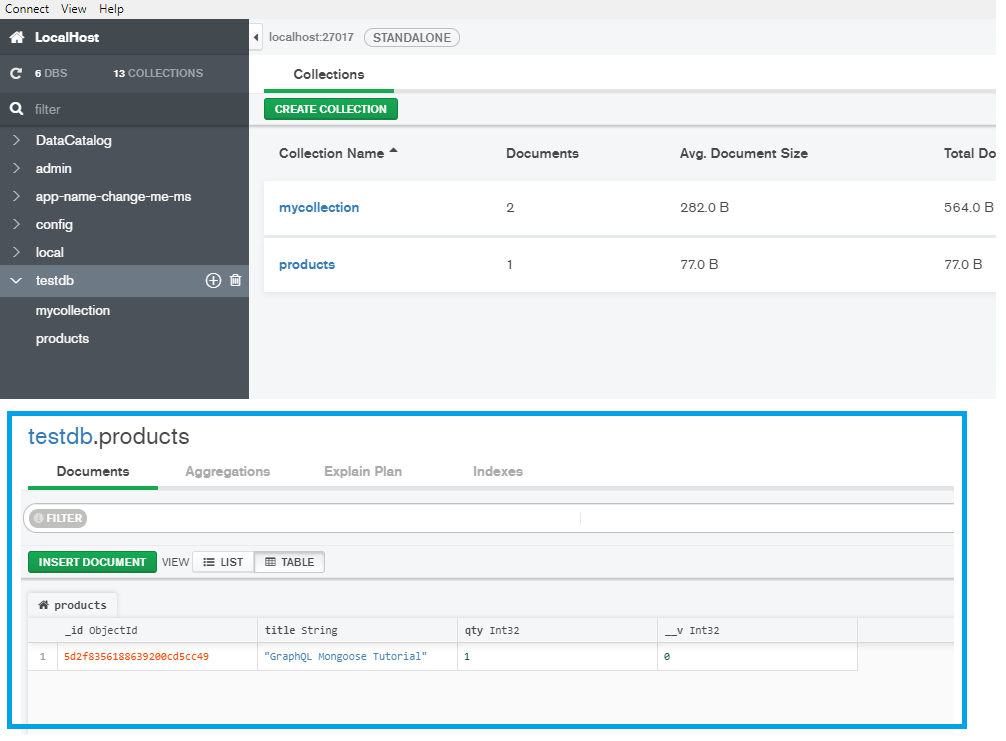
**Before MongoDB**

****

**On Browser**

****

**After MongoDB**

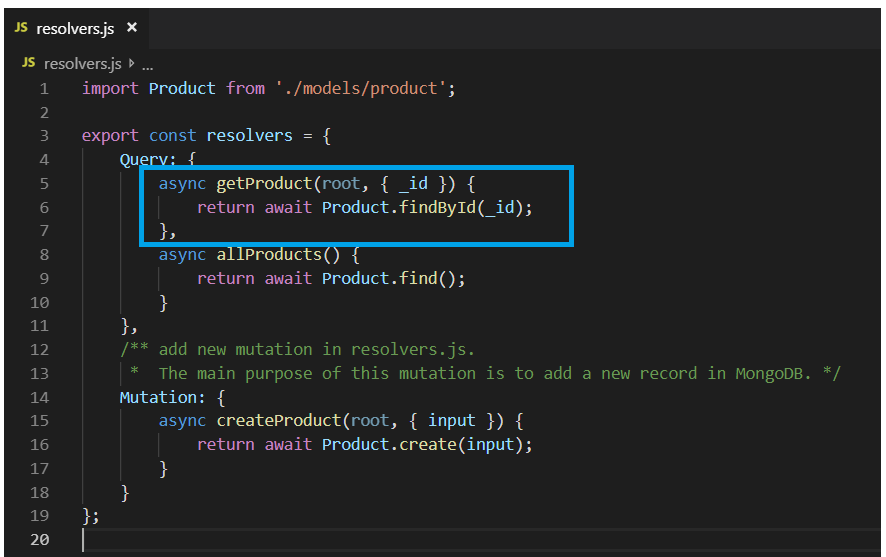
****

## Testing: Sending Mutation Query from GraphiQL



# Find Product

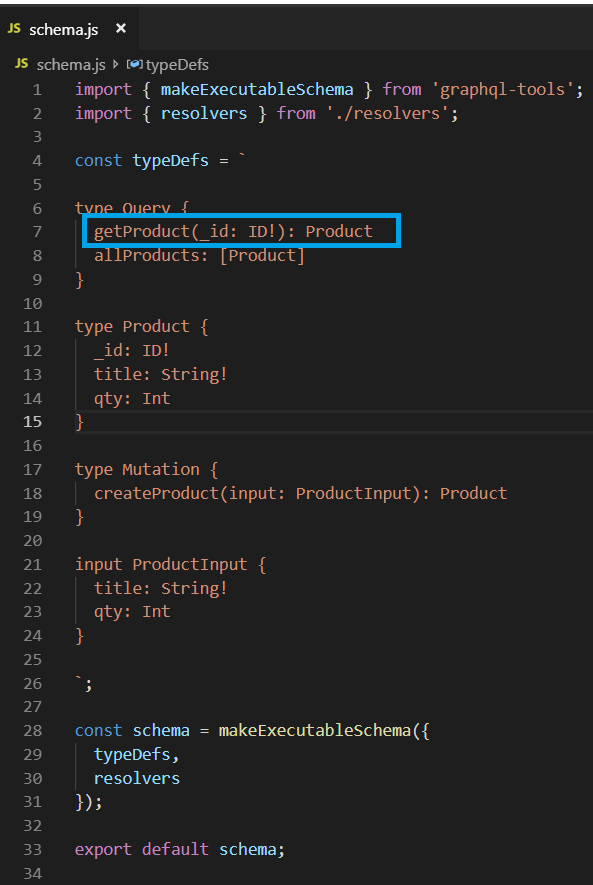
**Step 1: Adding getProduct Query in Resolvers.js**

****

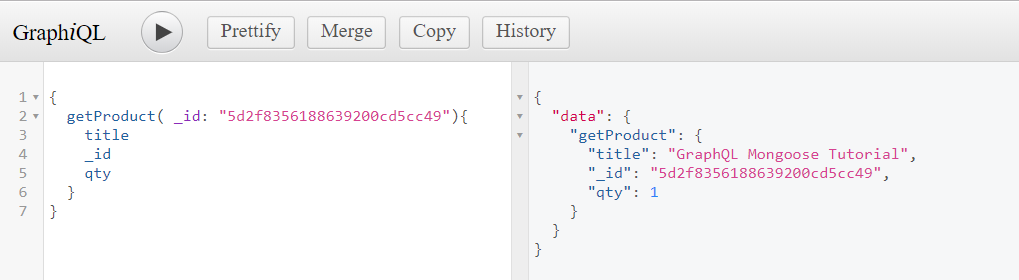
**Step 2: Adding getProduct Query in Schema.js**

We are going to create a new **getProduct** type in schema.js file.

We want to get Product by id. This getProduct also accepts an id argument, and the return type of the getProduct is Product.

****

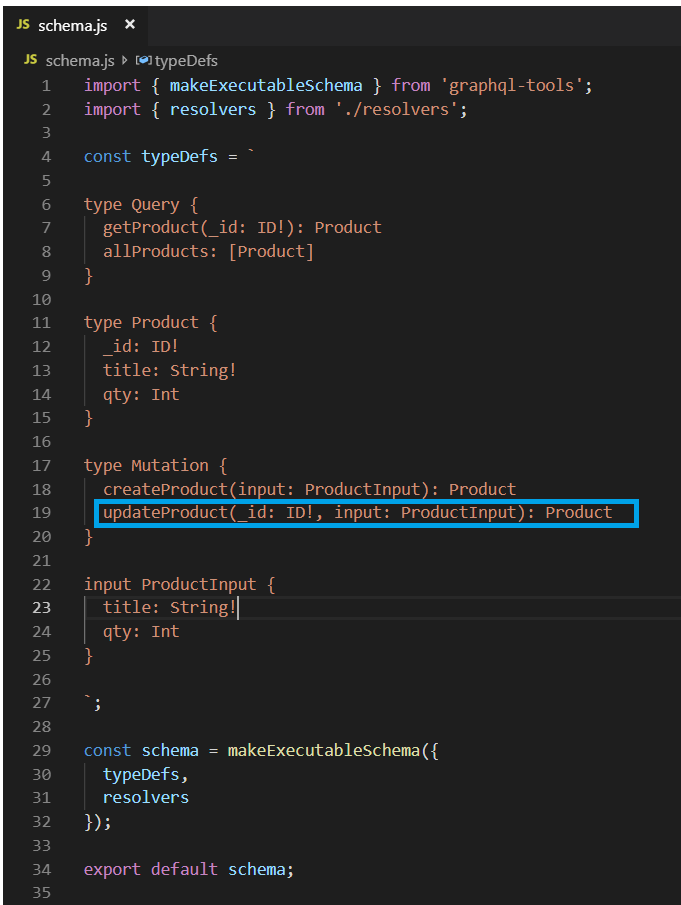
## Testing: Sending getProduct Query from GraphiQL



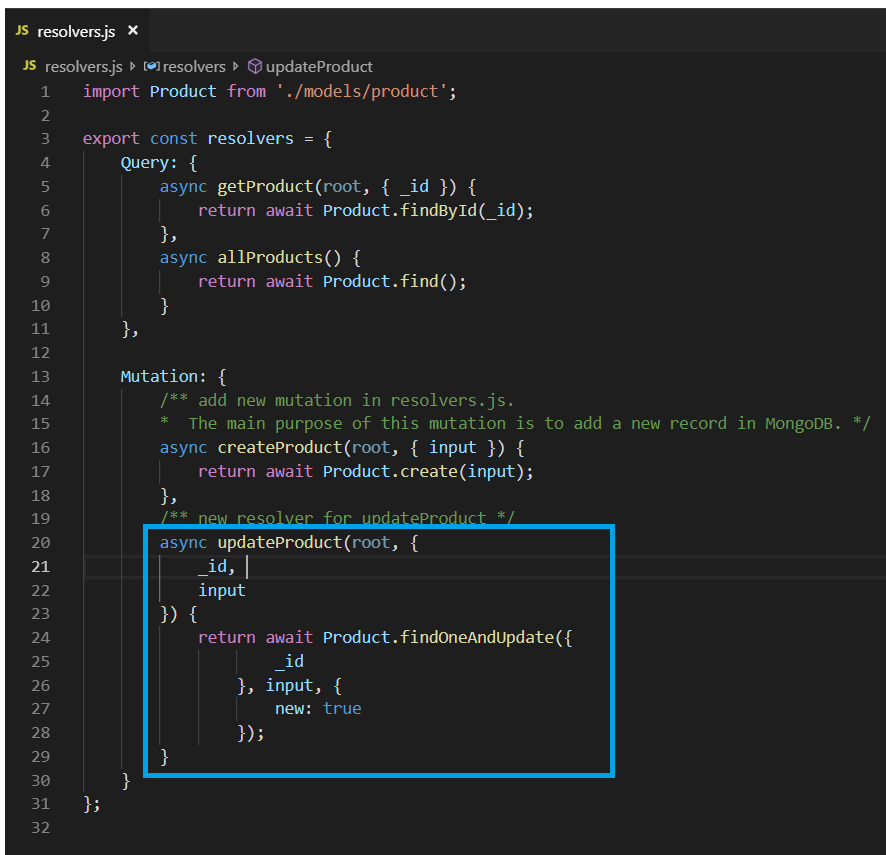
# Update Product

**Step 1: Adding updateProduct Mutation in Schema.js**

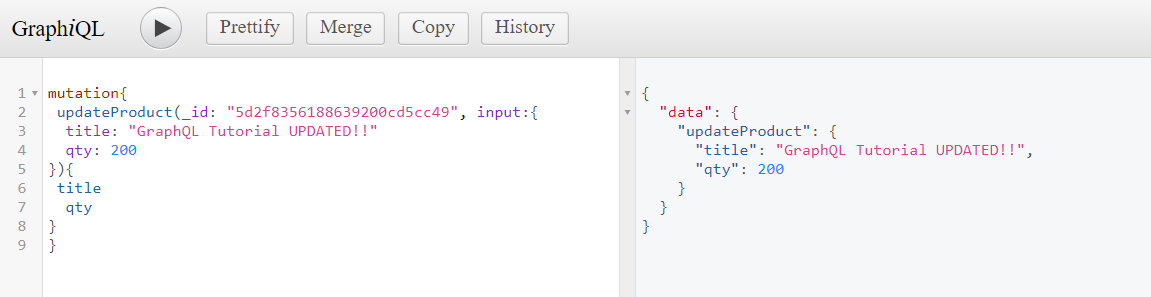
We need to create an updateProduct mutation in the schema.js file



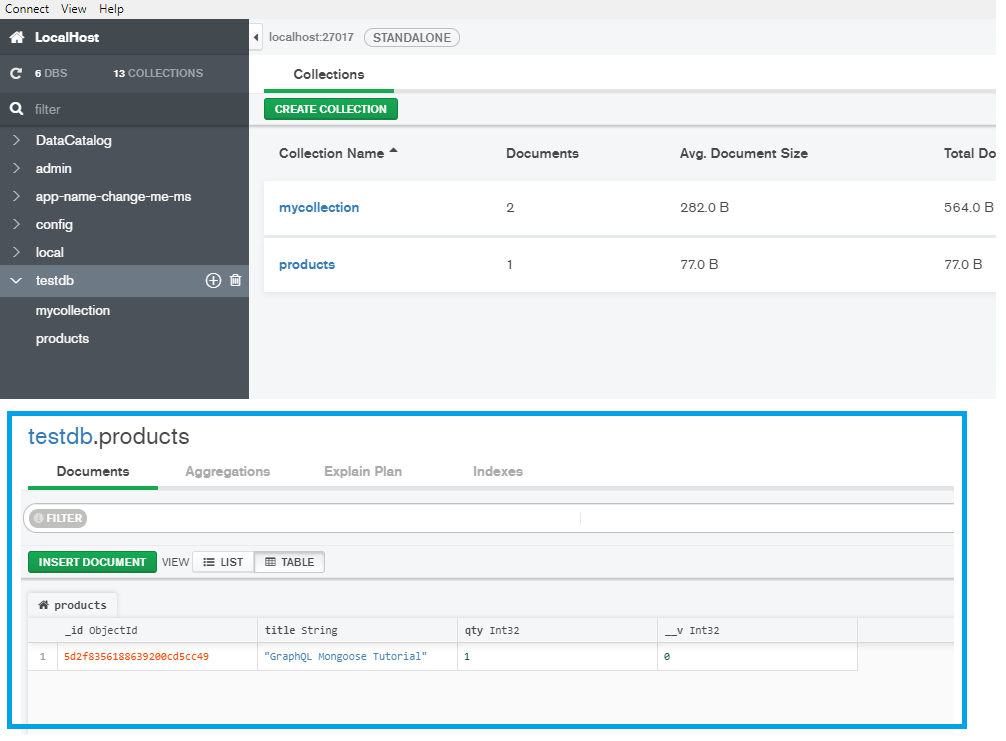
**Step 2: Adding updateProduct  Mutation in Resolver.js**



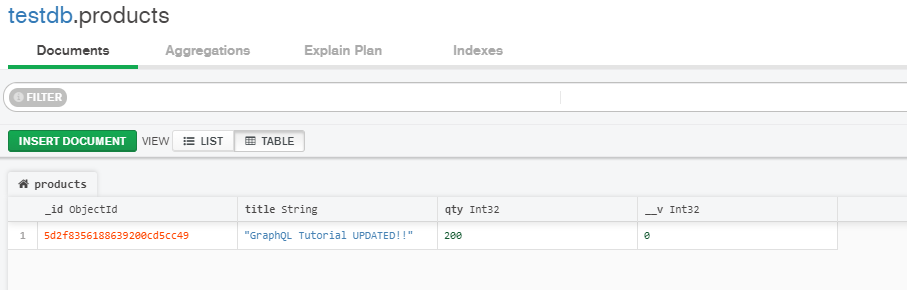
## Testing: Sending updateProduct  Mutation from GraphiQL



Before MongoDB

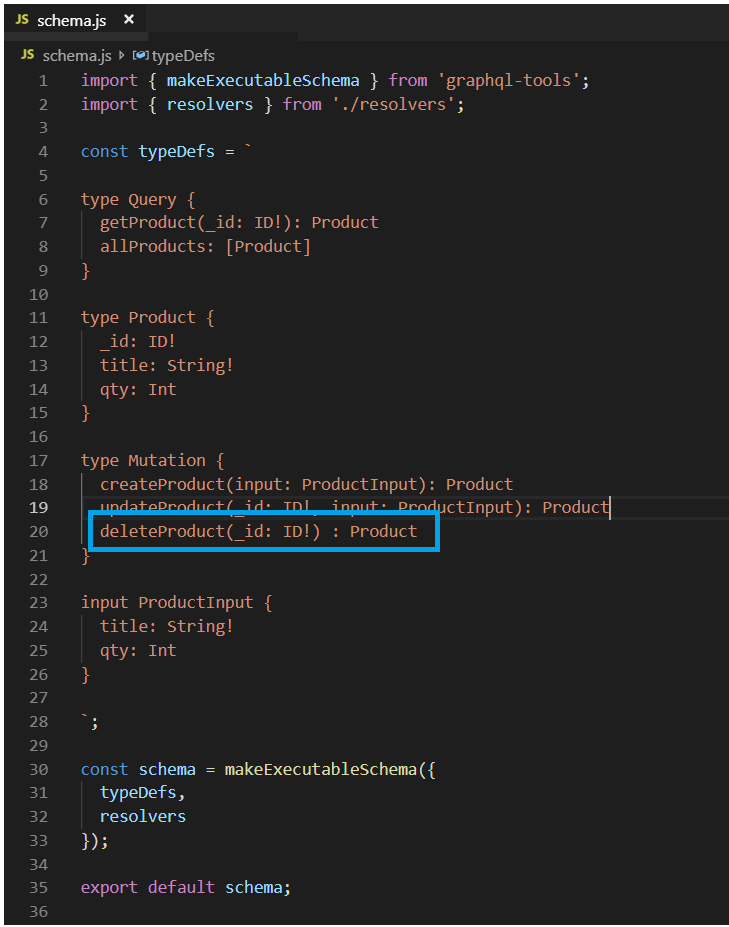
****

**After MongoDB**

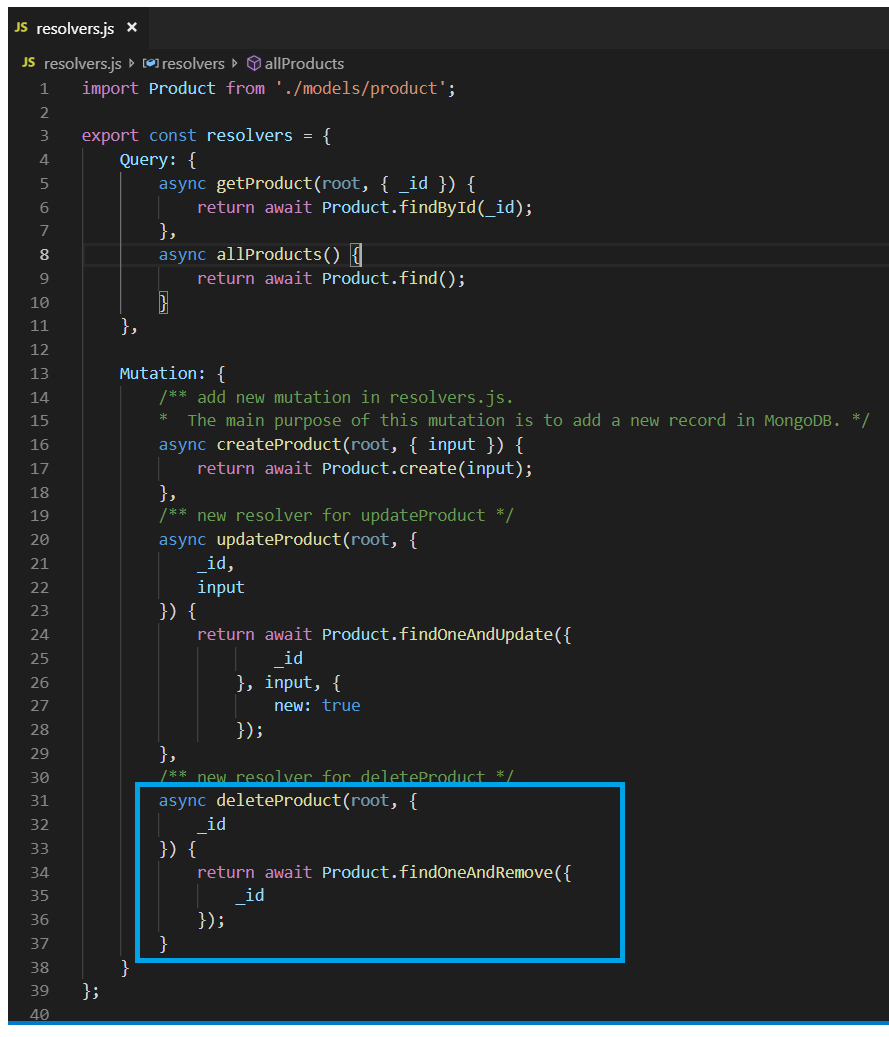


# Delete Product

**Step 1: Adding deleteProduct Mutation in Schema.js**



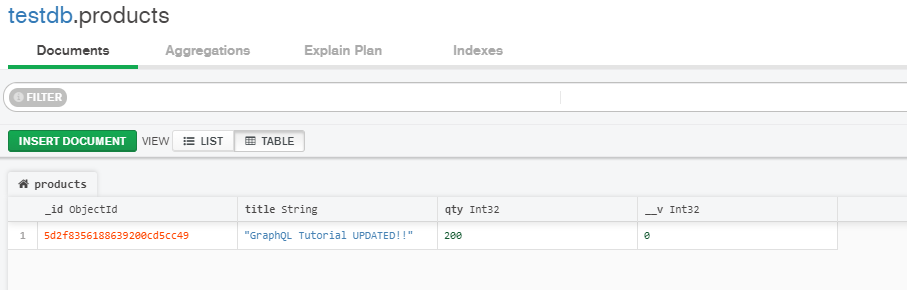
**Step 2: Adding deleteProduct  Mutation in Resolver.js**



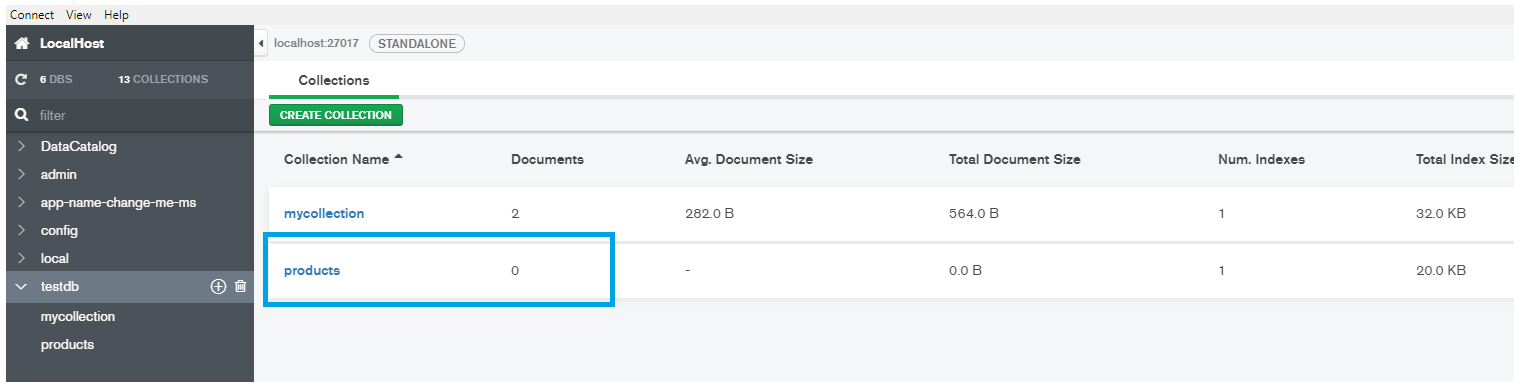
## Testing: Sending deleteProduct  Mutation from GraphiQL



**Before MongoDB**



After MongoDB



<https://medium.com/@haidermalik504/building-apis-with-graphql-nodejs-and-mongoose-64655c062dd2>