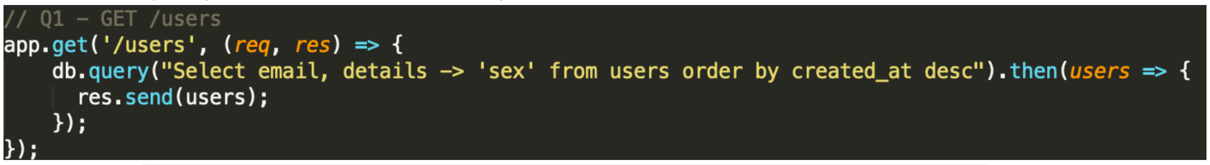
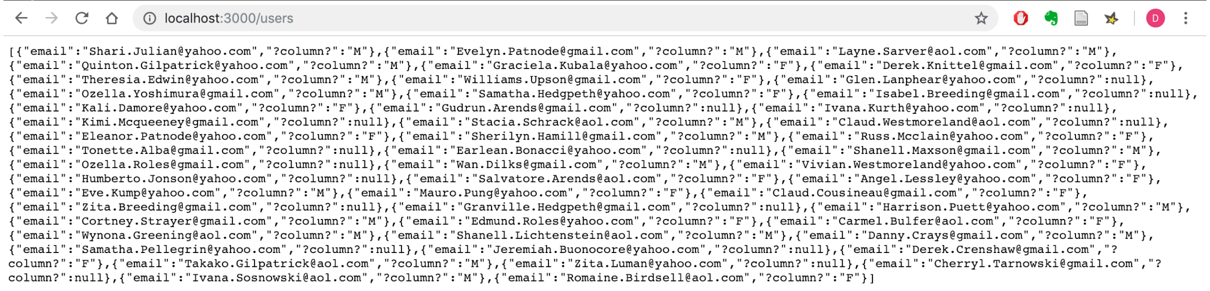
Lab Part 01:

Massive instance to connect to the database:

This is the query from Q1.1 an the output when tested in the browser.

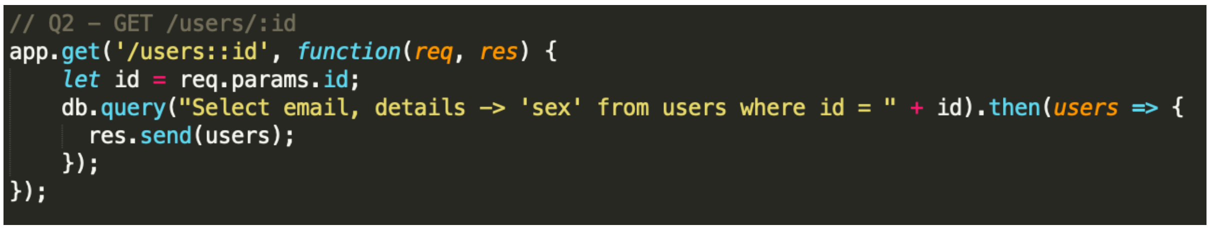


```
// Q1 - GET /users
app.get('/users', (req, res) => {
    db.query("Select email, details -> 'sex' from users order by created_at desc").then(users => {
      res.send(users);
    });
});
```

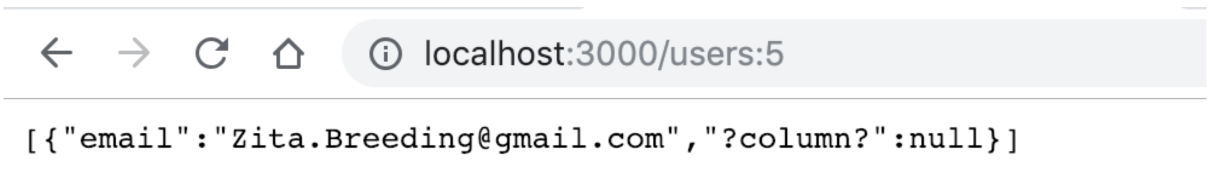


```
[{"email":"Shari.Julian@yahoo.com","?column?":"M"},{"email":"Evelyn.Patnode@gmail.com","?column?":"M"},{"email":"Layne.Sarver@aol.com","?column?":"M"},
{"email":"Quinton.Gilpatrick@yahoo.com","?column?":"M"},{"email":"Graciela.Kubala@yahoo.com","?column?":"F"},{"email":"Derek.Knittel@gmail.com","?column?":"F"},
{"email":"Theresia.Edwin@yahoo.com","?column?":"M"},{"email":"Williams.Upson@gmail.com","?column?":"F"},{"email":"Glen.Lanphear@yahoo.com","?column?":null},
{"email":"Ozella.Yoshimura@gmail.com","?column?":"M"},{"email":"Samatha.Hedgpeth@yahoo.com","?column?":"F"},{"email":"Isabel.Breeding@gmail.com","?column?":null},
{"email":"Kali.Damore@yahoo.com","?column?":"F"},{"email":"Gudrun.Arends@gmail.com","?column?":null},{"email":"Ivana.Kurth@yahoo.com","?column?":null},
{"email":"Kimi.Mcqueeney@gmail.com","?column?":null},{"email":"Stacia.Schrack@aol.com","?column?":"M"},{"email":"Claud.Westmoreland@aol.com","?column?":null},
{"email":"Eleanor.Patnode@yahoo.com","?column?":"F"},{"email":"Sherilyn.Hamill@gmail.com","?column?":"M"},{"email":"Russ.Mcclain@yahoo.com","?column?":"F"},
{"email":"Tonette.Alba@gmail.com","?column?":null},{"email":"Earlean.Bonacci@yahoo.com","?column?":null},{"email":"Shanell.Maxson@gmail.com","?column?":"M"},
{"email":"Ozella.Roles@gmail.com","?column?":null},{"email":"Wan.Dilks@gmail.com","?column?":"M"},{"email":"Vivian.Westmoreland@yahoo.com","?column?":"F"},
{"email":"Humberto.Jonson@yahoo.com","?column?":null},{"email":"Salvatore.Arends@aol.com","?column?":"F"},{"email":"Angel.Lessley@yahoo.com","?column?":"F"},
{"email":"Eve.Kump@yahoo.com","?column?":"M"},{"email":"Mauro.Pung@yahoo.com","?column?":"F"},{"email":"Claud.Cousineau@gmail.com","?column?":"F"},
{"email":"Zita.Breeding@gmail.com","?column?":null},{"email":"Granville.Hedgpeth@gmail.com","?column?":null},{"email":"Harrison.Puett@yahoo.com","?column?":"M"},
{"email":"Cortney.Strayer@gmail.com","?column?":"M"},{"email":"Edmund.Roles@yahoo.com","?column?":"F"},{"email":"Carmel.Bulfer@aol.com","?column?":"F"},
{"email":"Wynona.Greening@aol.com","?column?":"M"},{"email":"Shanell.Lichtenstein@aol.com","?column?":"M"},{"email":"Danny.Crays@gmail.com","?column?":"M"},
{"email":"Samatha.Pellegrin@yahoo.com","?column?":null},{"email":"Jeremiah.Buonocore@yahoo.com","?column?":null},{"email":"Derek.Crenshaw@gmail.com","?
column?":"F"},{"email":"Takako.Gilpatrick@aol.com","?column?":"M"},{"email":"Zita.Luman@yahoo.com","?column?":null},{"email":"Cherryl.Tarnowski@gmail.com","?
column?":null},{"email":"Ivana.Sosnowski@aol.com","?column?":"M"},{"email":"Romaine.Birdsell@aol.com","?column?":"F"}]
```

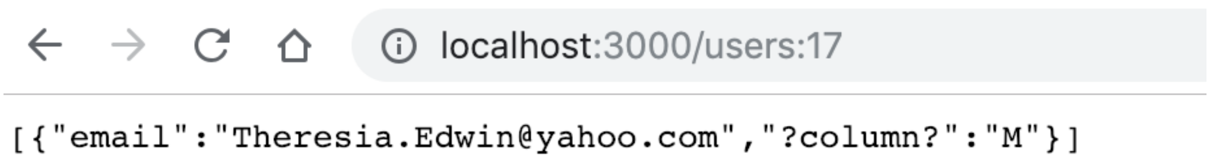
This is the query from Q1.2 and the output when tested in the browser.



```
// Q2 - GET /users/:id
app.get('/users::id', function(req, res) {
    let id = req.params.id;
    db.query("Select email, details -> 'sex' from users where id = " + id).then(users => {
      res.send(users);
    });
});
```

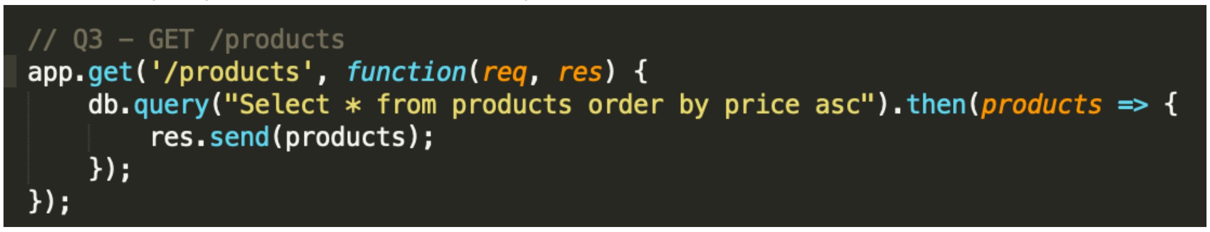


```
[{"email":"Zita.Breeding@gmail.com","?column?":null}]
```

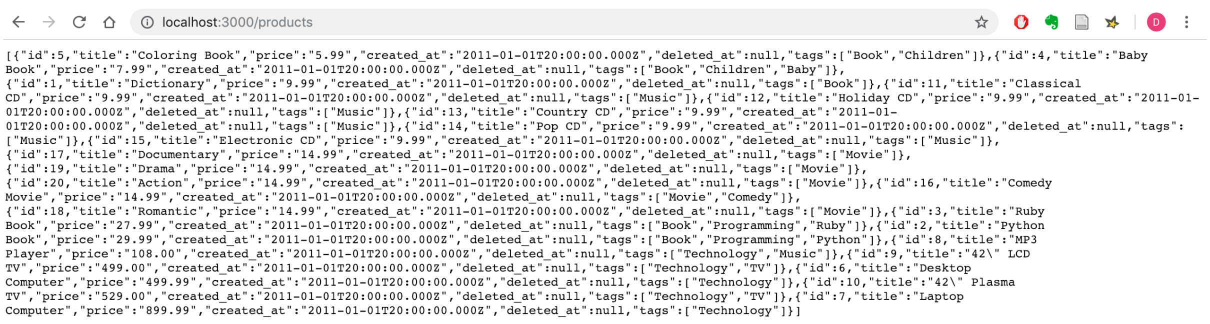


```
[{"email":"Theresia.Edwin@yahoo.com","?column?":"M"}]
```

This is the query from Q.1.3 and the output when tested in the browser.



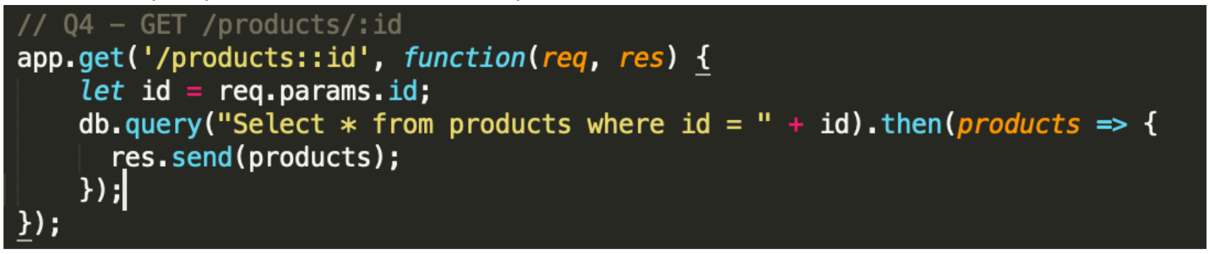
```
// Q3 - GET /products
app.get('/products', function(req, res) {
    db.query("Select * from products order by price asc").then(products => {
        res.send(products);
    });
});
```



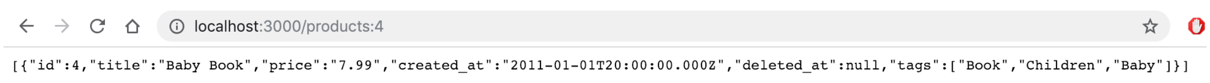
localhost:3000/products

```
[{"id":5,"title":"Coloring Book","price":"5.99","created_at":"2011-01-01T20:00:00.000Z","deleted_at":null,"tags":["Book","Children"]},{"id":4,"title":"Baby
Book","price":"7.99","created_at":"2011-01-01T20:00:00.000Z","deleted_at":null,"tags":["Book","Children","Baby"]},
{"id":1,"title":"Dictionary","price":"9.99","created_at":"2011-01-01T20:00:00.000Z","deleted_at":null,"tags":["Book"]},{"id":11,"title":"Classical
CD","price":"9.99","created_at":"2011-01-01T20:00:00.000Z","deleted_at":null,"tags":["Music"]},{"id":12,"title":"Holiday CD","price":"9.99","created_at":"2011-01-
01T20:00:00.000Z","deleted_at":null,"tags":["Music"]},{"id":13,"title":"Country CD","price":"9.99","created_at":"2011-01-
01T20:00:00.000Z","deleted_at":null,"tags":["Music"]},{"id":14,"title":"Pop CD","price":"9.99","created_at":"2011-01-01T20:00:00.000Z","deleted_at":null,"tags":
["Music"]},{"id":15,"title":"Electronic CD","price":"9.99","created_at":"2011-01-01T20:00:00.000Z","deleted_at":null,"tags":["Music"]},
{"id":17,"title":"Documentary","price":"14.99","created_at":"2011-01-01T20:00:00.000Z","deleted_at":null,"tags":["Movie"]},
{"id":19,"title":"Drama","price":"14.99","created_at":"2011-01-01T20:00:00.000Z","deleted_at":null,"tags":["Movie"]},
{"id":20,"title":"Action","price":"14.99","created_at":"2011-01-01T20:00:00.000Z","deleted_at":null,"tags":["Movie"]},{"id":16,"title":"Comedy
Movie","price":"14.99","created_at":"2011-01-01T20:00:00.000Z","deleted_at":null,"tags":["Movie","Comedy"]},
{"id":18,"title":"Romantic","price":"14.99","created_at":"2011-01-01T20:00:00.000Z","deleted_at":null,"tags":["Movie"]},{"id":3,"title":"Ruby
Book","price":"27.99","created_at":"2011-01-01T20:00:00.000Z","deleted_at":null,"tags":["Book","Programming","Ruby"]},{"id":2,"title":"Python
Book","price":"29.99","created_at":"2011-01-01T20:00:00.000Z","deleted_at":null,"tags":["Book","Programming","Python"]},{"id":8,"title":"MP3
Player","price":"108.00","created_at":"2011-01-01T20:00:00.000Z","deleted_at":null,"tags":["Technology","Music"]},{"id":9,"title":"42\" LCD
TV","price":"499.00","created_at":"2011-01-01T20:00:00.000Z","deleted_at":null,"tags":["Technology","TV"]},{"id":6,"title":"Desktop
Computer","price":"499.99","created_at":"2011-01-01T20:00:00.000Z","deleted_at":null,"tags":["Technology"]},{"id":10,"title":"42\" Plasma
TV","price":"529.00","created_at":"2011-01-01T20:00:00.000Z","deleted_at":null,"tags":["Technology","TV"]},{"id":7,"title":"Laptop
Computer","price":"899.99","created_at":"2011-01-01T20:00:00.000Z","deleted_at":null,"tags":["Technology"]}]
```

This is the query from Q1.4 and the output when tested in the browser.

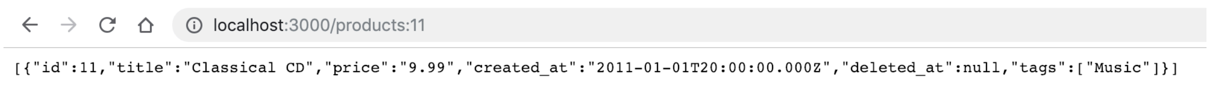


```
// Q4 - GET /products/:id
app.get('/products::id', function(req, res) {
    let id = req.params.id;
    db.query("Select * from products where id = " + id).then(products => {
      res.send(products);
    });
});
```



localhost:3000/products:4

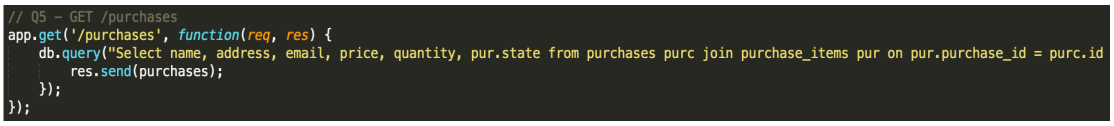
```
[{"id":4,"title":"Baby Book","price":"7.99","created_at":"2011-01-01T20:00:00.000Z","deleted_at":null,"tags":["Book","Children","Baby"]}]
```



localhost:3000/products:11

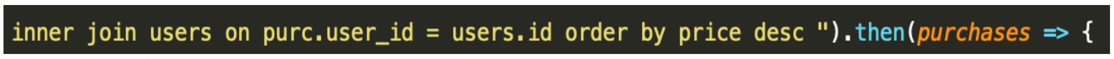
```
[{"id":11,"title":"Classical CD","price":"9.99","created_at":"2011-01-01T20:00:00.000Z","deleted_at":null,"tags":["Music"]}]
```

This is the query from Q1.5 and the output when tested in the browser.

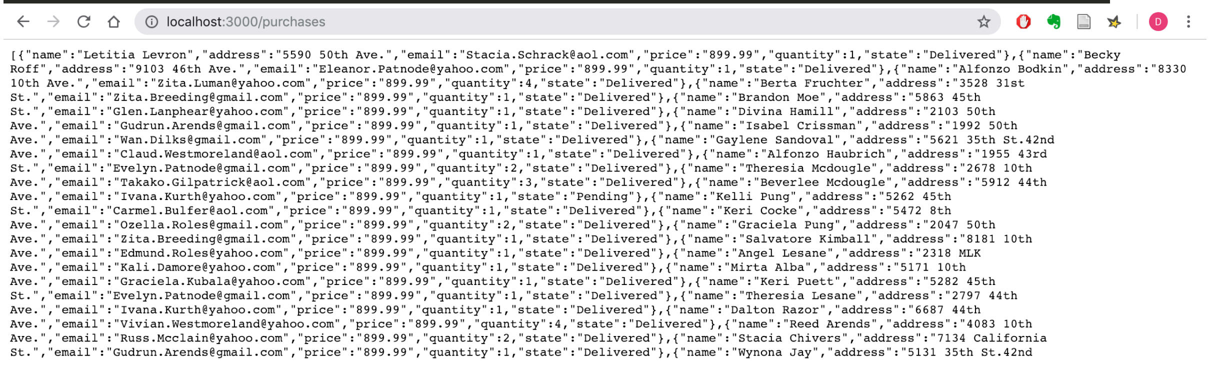


```
// Q5 - GET /purchases
app.get('/purchases', function(req, res) {
    db.query("Select name, address, email, price, quantity, pur.state from purchases purc join purchase_items pur on pur.purchase_id = purc.id
        res.send(purchases);
    });
});
```

+



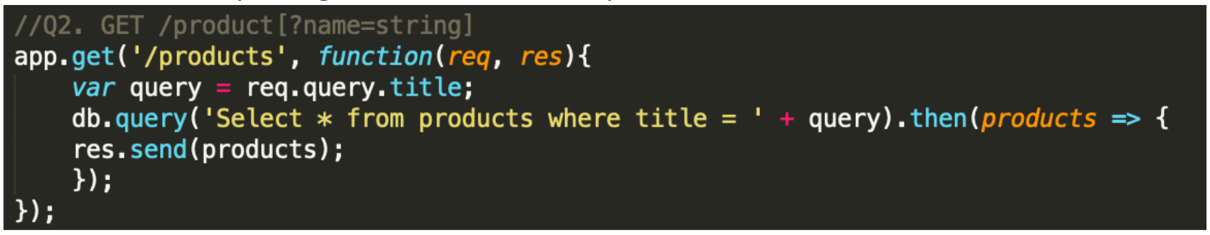
```
inner join users on purc.user_id = users.id order by price desc ").then(purchases => {
```



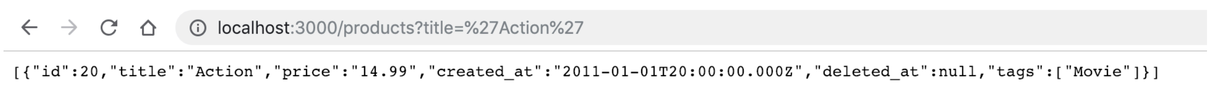
localhost:3000/purchases

```
[{"name":"Letitia Levron","address":"5590 50th Ave.","email":"Stacia.Schrack@aol.com","price":"899.99","quantity":1,"state":"Delivered"},{"name":"Becky
Roff","address":"9103 46th Ave.","email":"Eleanor.Patnode@yahoo.com","price":"899.99","quantity":1,"state":"Delivered"},{"name":"Alfonzo Bodkin","address":"8330
10th Ave.","email":"Zita.Luman@yahoo.com","price":"899.99","quantity":4,"state":"Delivered"},{"name":"Berta Fruchter","address":"3528 31st
St.","email":"Zita.Breeding@gmail.com","price":"899.99","quantity":1,"state":"Delivered"},{"name":"Brandon Moe","address":"5863 45th
St.","email":"Glen.Lanphear@yahoo.com","price":"899.99","quantity":1,"state":"Delivered"},{"name":"Divina Hamill","address":"2103 50th
Ave.","email":"Gudrun.Arends@gmail.com","price":"899.99","quantity":1,"state":"Delivered"},{"name":"Isabel Crissman","address":"1992 50th
Ave.","email":"Wan.Dilks@gmail.com","price":"899.99","quantity":1,"state":"Delivered"},{"name":"Gaylene Sandoval","address":"5621 35th St.42nd
Ave.","email":"Claud.Westmoreland@aol.com","price":"899.99","quantity":1,"state":"Delivered"},{"name":"Alfonzo Haubrich","address":"1955 43rd
St.","email":"Evelyn.Patnode@gmail.com","price":"899.99","quantity":2,"state":"Delivered"},{"name":"Theresia Mcdougle","address":"2678 10th
Ave.","email":"Takako.Gilpatrick@aol.com","price":"899.99","quantity":3,"state":"Delivered"},{"name":"Beverlee Mcdougle","address":"5912 44th
Ave.","email":"Ivana.Kurth@yahoo.com","price":"899.99","quantity":1,"state":"Pending"},{"name":"Kelli Pung","address":"5262 45th
St.","email":"Carmel.Bulfer@aol.com","price":"899.99","quantity":1,"state":"Delivered"},{"name":"Keri Cocke","address":"5472 8th
Ave.","email":"Ozella.Roles@gmail.com","price":"899.99","quantity":2,"state":"Delivered"},{"name":"Graciela Pung","address":"2047 50th
Ave.","email":"Zita.Breeding@gmail.com","price":"899.99","quantity":1,"state":"Delivered"},{"name":"Salvatore Kimball","address":"8181 10th
Ave.","email":"Edmund.Roles@yahoo.com","price":"899.99","quantity":1,"state":"Delivered"},{"name":"Angel Lesane","address":"2318 MLK
Ave.","email":"Kali.Damore@yahoo.com","price":"899.99","quantity":1,"state":"Delivered"},{"name":"Mirta Alba","address":"5171 10th
Ave.","email":"Graciela.Kubala@yahoo.com","price":"899.99","quantity":1,"state":"Delivered"},{"name":"Keri Puett","address":"5282 45th
St.","email":"Evelyn.Patnode@gmail.com","price":"899.99","quantity":1,"state":"Delivered"},{"name":"Theresia Lesane","address":"2797 44th
Ave.","email":"Ivana.Kurth@yahoo.com","price":"899.99","quantity":1,"state":"Delivered"},{"name":"Dalton Razor","address":"6687 44th
Ave.","email":"Vivian.Westmoreland@yahoo.com","price":"899.99","quantity":4,"state":"Delivered"},{"name":"Reed Arends","address":"4083 10th
Ave.","email":"Russ.Mcclain@yahoo.com","price":"899.99","quantity":2,"state":"Delivered"},{"name":"Stacia Chivers","address":"7134 California
St.","email":"Gudrun.Arends@gmail.com","price":"899.99","quantity":1,"state":"Delivered"},{"name":"Wynona Jay","address":"5131 35th St.42nd
```

Below is the Query String from Q2 and the output when tested in the browser.



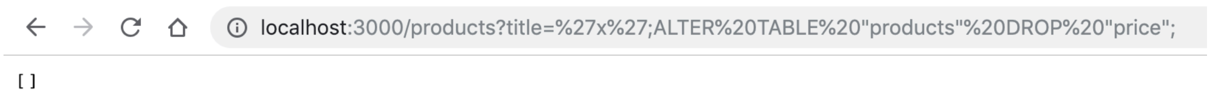
```
//Q2. GET /product[?name=string]
app.get('/products', function(req, res){
    var query = req.query.title;
    db.query('Select * from products where title = ' + query).then(products => {
    res.send(products);
    });
});
```



```
localhost:3000/products?title=%27Action%27
[{"id":20,"title":"Action","price":"14.99","created_at":"2011-01-01T20:00:00.000Z","deleted_at":null,"tags":["Movie"]}]
```

SQL INJECTION Q2.2

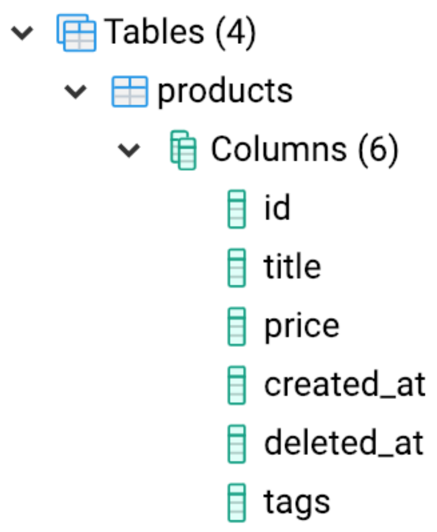
Below is the SQL Query used to implement an example of SQL Injection on a database. I decided to drop the 'price' column from the 'products' table.



```
localhost:3000/products?title=%27x%27;ALTER%20TABLE%20"products"%20DROP%20"price";
[]
```

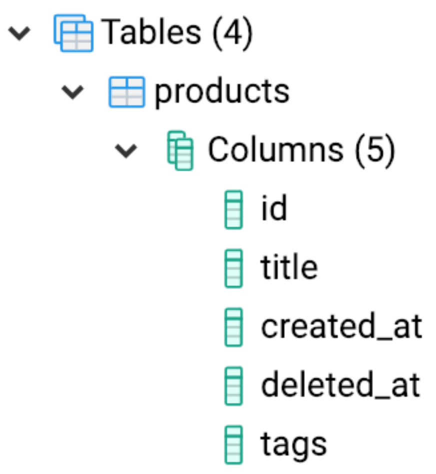
An example of before and after the query was ran through the browser.

Before:



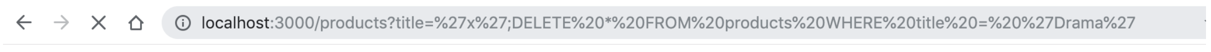
```
Tables (4)
  products
    Columns (6)
      id
      title
      price
      created_at
      deleted_at
      tags
```

After:



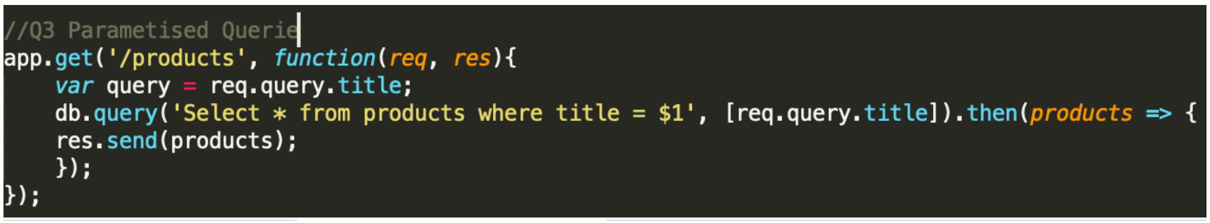
```
Tables (4)
  products
    Columns (5)
      id
      title
      created_at
      deleted_at
      tags
```

You could also delete a product from the products table by entering this into your browser:

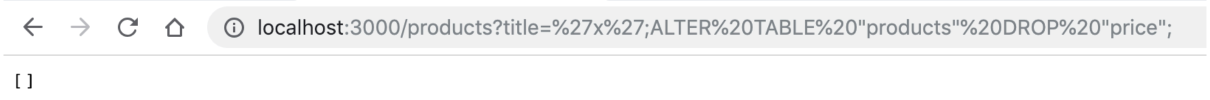


```
localhost:3000/products?title=%27x%27;DELETE%20*%20FROM%20products%20WHERE%20title%20=%20%27Drama%27
```

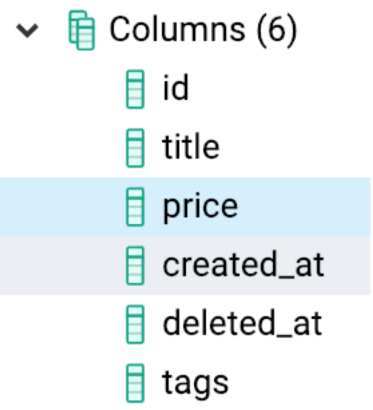
I had to re-download the ‘pgguide’ database as the price column was removed as of the last section. This is part of Q.3, using a parameterised query to protect against SQL injection. Also below is the output and proof that it worked correctly as the price column remained in the table.



```
//Q3 Parametised Querie
app.get('/products', function(req, res){
    var query = req.query.title;
    db.query('Select * from products where title = $1', [req.query.title]).then(products => {
    res.send(products);
    });
});
```



```
localhost:3000/products?title=%27x%27;ALTER%20TABLE%20"products"%20DROP%20"price";
[]
```

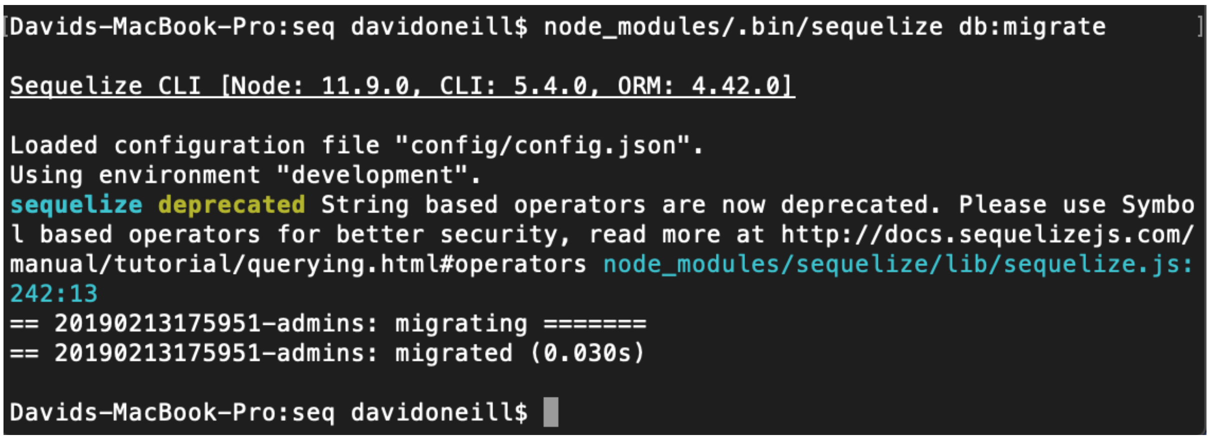


```
Columns (6)
    id
    title
    price
    created_at
    deleted_at
    tags
```

Q3.2 Stored Procedure:
Did not get the Stored Procedure section completed.

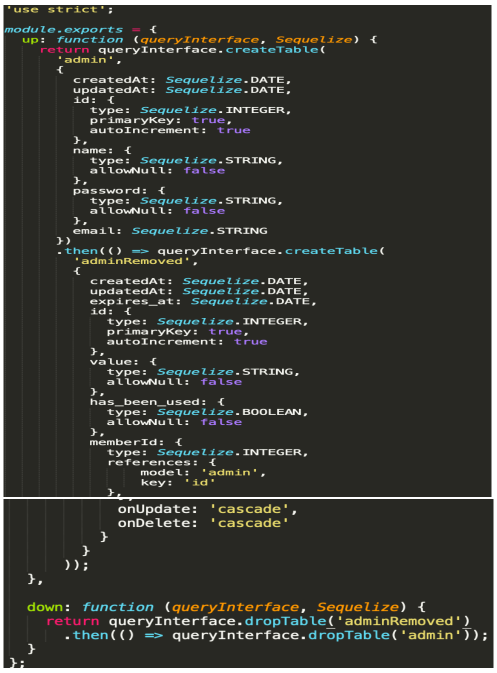
Next is Q.4. This is my migrations that I used. I decided to create 2 new tables called ‘admin’ and ‘adminRemoved’.

Below is the command being run in the terminal to begin the migration.



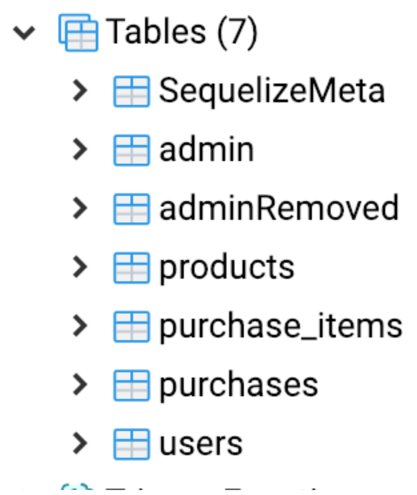
```
Davids-MacBook-Pro:seq davidoneill$ node_modules/.bin/sequelize db:migrate

Sequelize CLI [Node: 11.9.0, CLI: 5.4.0, ORM: 4.42.0]

Loaded configuration file "config/config.json".
Using environment "development".
sequelize deprecated String based operators are now deprecated. Please use Symbo
l based operators for better security, read more at http://docs.sequelizejs.com/
manual/tutorial/querying.html#operators node_modules/sequelize/lib/sequelize.js:
242:13
== 20190213175951-admins: migrating =======
== 20190213175951-admins: migrated (0.030s)

Davids-MacBook-Pro:seq davidoneill$
```

The code from my migration file:



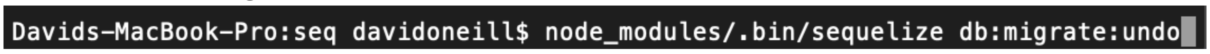
```
'use strict';

module.exports = {
  up: function (queryInterface, Sequelize) {
    return queryInterface.createTable(
      'admin',
      {
        createdAt: Sequelize.DATE,
        updatedAt: Sequelize.DATE,
        id: {
          type: Sequelize.INTEGER,
          primaryKey: true,
          autoIncrement: true
        },
        name: {
          type: Sequelize.STRING,
          allowNull: false
        },
        password: {
          type: Sequelize.STRING,
          allowNull: false
        },
        email: Sequelize.STRING
      })
      .then(() => queryInterface.createTable(
        'adminRemoved',
        {
          createdAt: Sequelize.DATE,
          updatedAt: Sequelize.DATE,
          expires_at: Sequelize.DATE,
          id: {
            type: Sequelize.INTEGER,
            primaryKey: true,
            autoIncrement: true
          },
          value: {
            type: Sequelize.STRING,
            allowNull: false
          },
          has_been_used: {
            type: Sequelize.BOOLEAN,
            allowNull: false
          },
          memberId: {
            type: Sequelize.INTEGER,
            references: {
                model: 'admin',
                key: 'id'
            },
            onUpdate: 'cascade',
            onDelete: 'cascade'
          }
        }
      ));
  },

  down: function (queryInterface, Sequelize) {
    return queryInterface.dropTable('adminRemoved')
      .then(() => queryInterface.dropTable('admin'));
  }
};
```

The new tables after being added to the database:



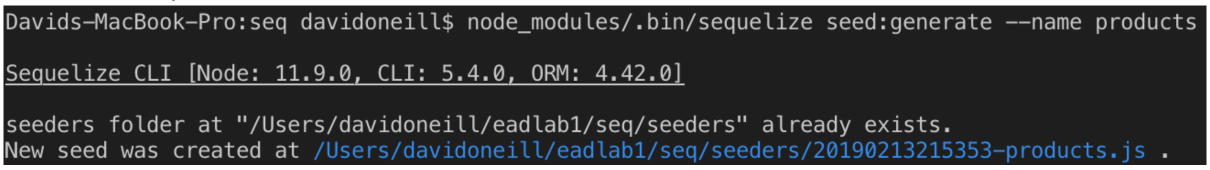
```
Tables (7)
  SequelizeMeta
  admin
  adminRemoved
  products
  purchase_items
  purchases
  users
```

You can undo a migration with the below command:



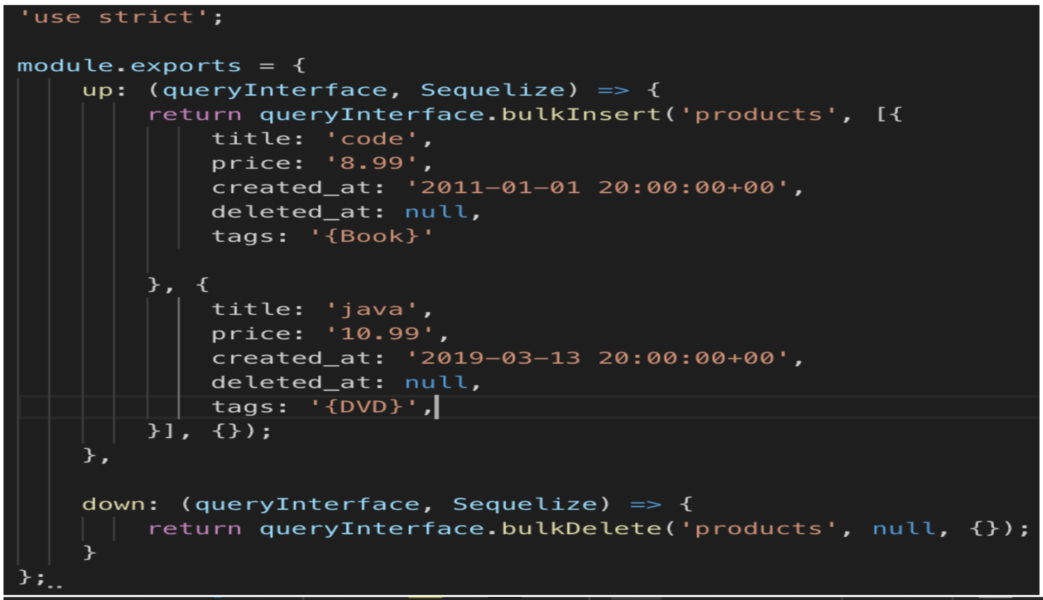
```
Davids-MacBook-Pro:seq davidoneill$ node_modules/.bin/sequelize db:migrate:undo
```

For Q.5 I added test data to my tables by using Seeds. Below I generated new seeds for each table. Below is the products table example. The rest of them can be found in /eadlab1/seq/seeders'.

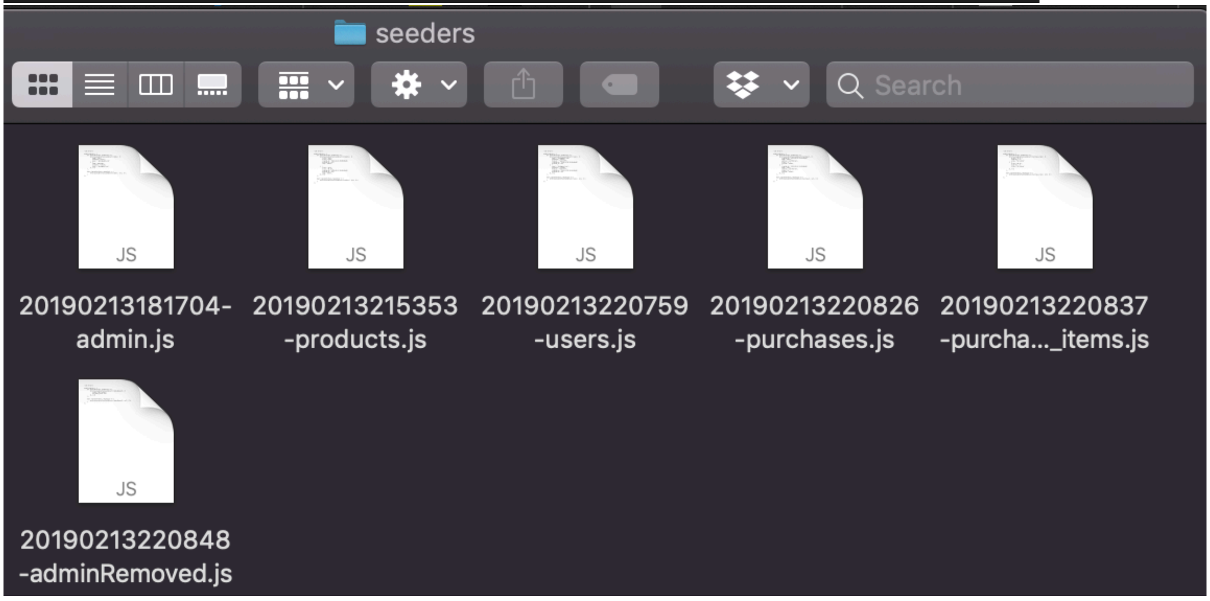


```
Davids-MacBook-Pro:seq davidoneill$ node_modules/.bin/sequelize seed:generate --name products

Sequelize CLI [Node: 11.9.0, CLI: 5.4.0, ORM: 4.42.0]

seeders folder at "/Users/davidoneill/eadlab1/seq/seeders" already exists.
New seed was created at /Users/davidoneill/eadlab1/seq/seeders/20190213215353-products.js .
```

This is the Seeders code used for the products table, the rest can be found in the project folder containing seeders for every table.

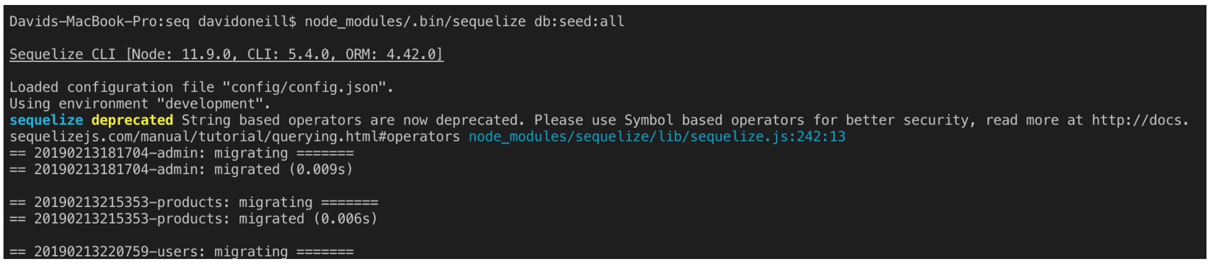


```
'use strict';

module.exports = {
    up: (queryInterface, Sequelize) => {
        return queryInterface.bulkInsert('products', [{
            title: 'code',
            price: '8.99',
            created_at: '2011-01-01 20:00:00+00',
            deleted_at: null,
            tags: '{Book}'

        }, {
            title: 'java',
            price: '10.99',
            created_at: '2019-03-13 20:00:00+00',
            deleted_at: null,
            tags: '{DVD}',
        }], {});
    },

    down: (queryInterface, Sequelize) => {
        return queryInterface.bulkDelete('products', null, {});
    }
};
```



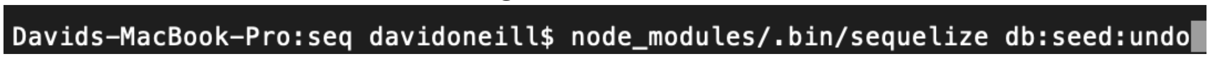
```
seeders

20190213181704-admin.js
20190213215353-products.js
20190213220759-users.js
20190213220826-purchases.js
20190213220837-purcha..._items.js
20190213220848-adminRemoved.js
```

To run the seeds you must run the following command.



```
Davids-MacBook-Pro:seq davidoneill$ node_modules/.bin/sequelize db:seed:all

Sequelize CLI [Node: 11.9.0, CLI: 5.4.0, ORM: 4.42.0]

Loaded configuration file "config/config.json".
Using environment "development".
sequelize deprecated String based operators are now deprecated. Please use Symbol based operators for better security, read more at http://docs.
sequelizejs.com/manual/tutorial/querying.html#operators node_modules/sequelize/lib/sequelize.js:242:13
== 20190213181704-admin: migrating =======
== 20190213181704-admin: migrated (0.009s)

== 20190213215353-products: migrating =======
== 20190213215353-products: migrated (0.006s)

== 20190213220759-users: migrating =======
```

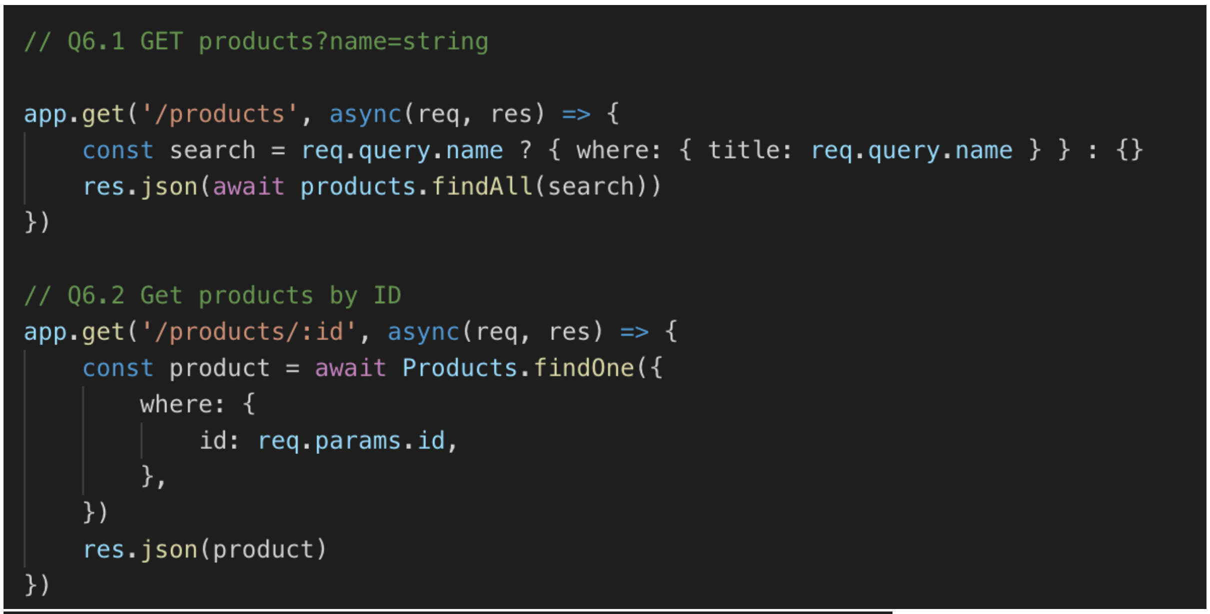
You can undo a seed with the following command:



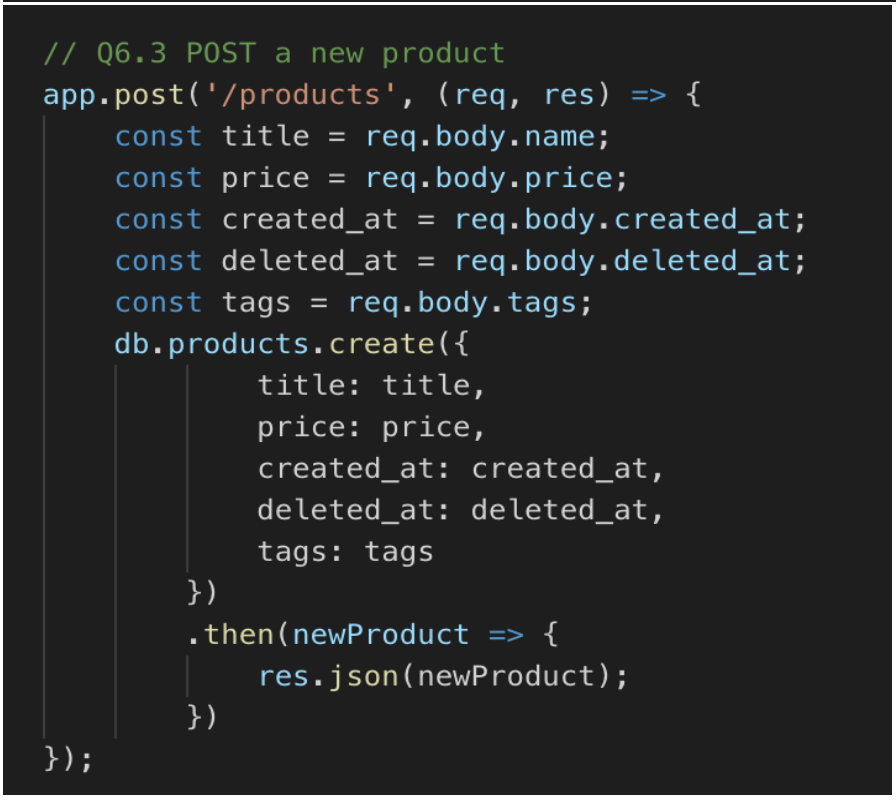
```
Davids-MacBook-Pro:seq davidoneill$ node_modules/.bin/sequelize db:seed:undo
```

You must be careful when using the command 'db:seed:undo:all' as it will remove all of the data from your database.

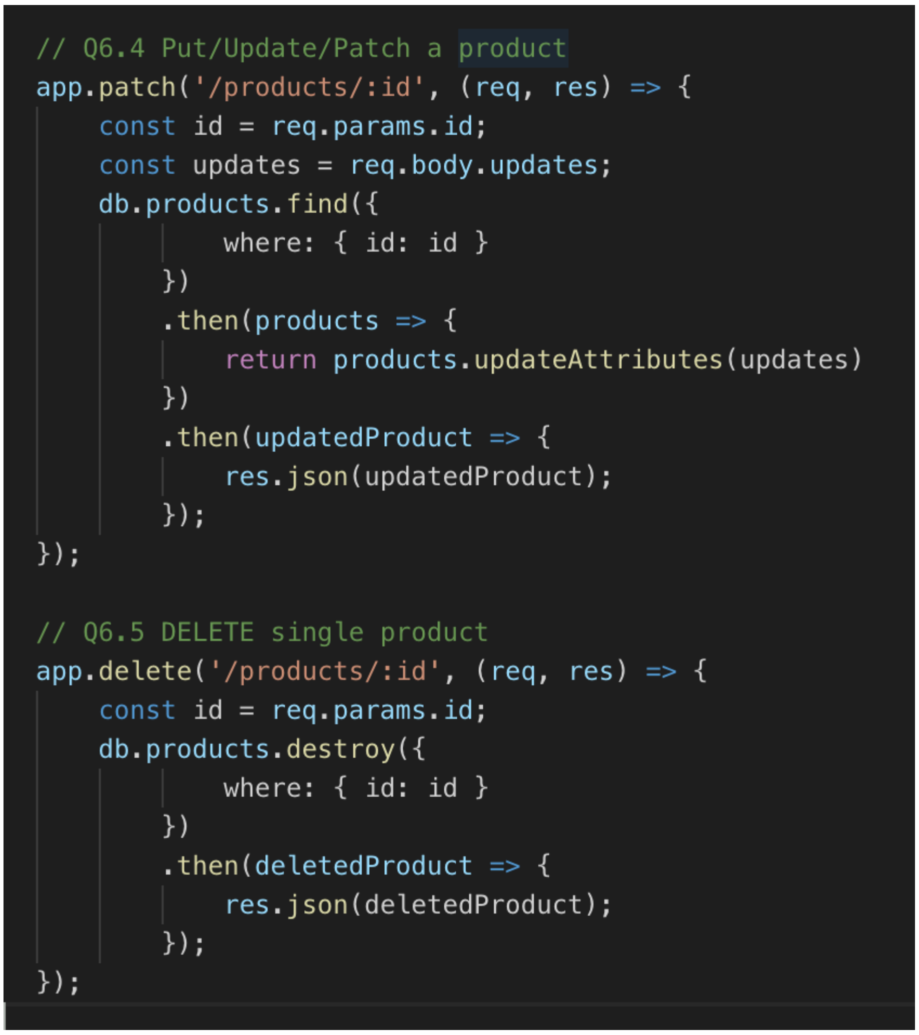
Here is my Q.6 code. I kept getting reference errors when trying to run this code so I could not retrieve the data or execute the code but this is the code that would have returned my desired results if I got it working in time.



```
// Q6.1 GET products?name=string

app.get('/products', async(req, res) => {
    const search = req.query.name ? { where: { title: req.query.name } } : {}
    res.json(await products.findAll(search))
})

// Q6.2 Get products by ID
app.get('/products/:id', async(req, res) => {
    const product = await Products.findOne({
        where: {
            id: req.params.id,
        },
    })
    res.json(product)
})
```



```
// Q6.3 POST a new product
app.post('/products', (req, res) => {
    const title = req.body.name;
    const price = req.body.price;
    const created_at = req.body.created_at;
    const deleted_at = req.body.deleted_at;
    const tags = req.body.tags;
    db.products.create({
            title: title,
            price: price,
            created_at: created_at,
            deleted_at: deleted_at,
            tags: tags
        })
        .then(newProduct => {
            res.json(newProduct);
        })
});
```



```
// Q6.4 Put/Update/Patch a product
app.patch('/products/:id', (req, res) => {
    const id = req.params.id;
    const updates = req.body.updates;
    db.products.find({
            where: { id: id }
        })
        .then(products => {
            return products.updateAttributes(updates)
        })
        .then(updatedProduct => {
            res.json(updatedProduct);
        });
});

// Q6.5 DELETE single product
app.delete('/products/:id', (req, res) => {
    const id = req.params.id;
    db.products.destroy({
            where: { id: id }
        })
        .then(deletedProduct => {
            res.json(deletedProduct);
        });
});
```