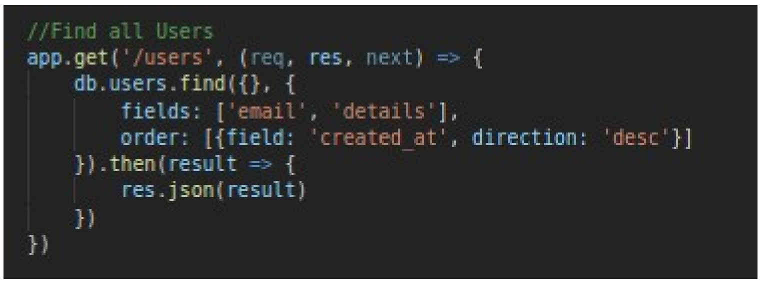
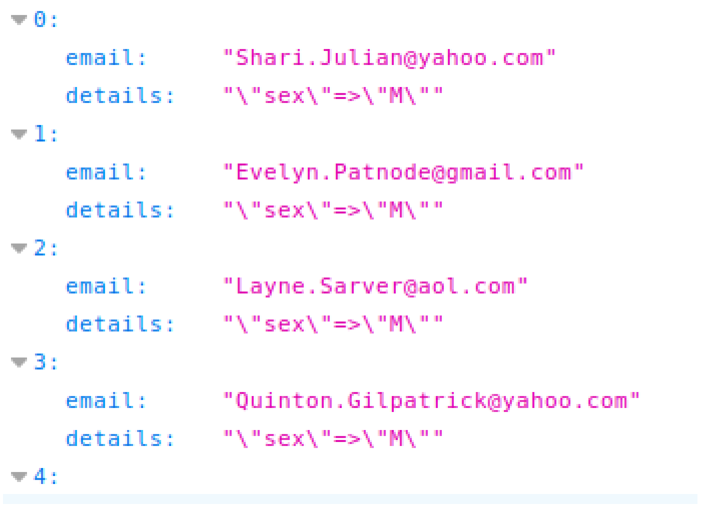
Enterprise Application Development: Lab 1

Problem Set 1:

1.a. GET /users

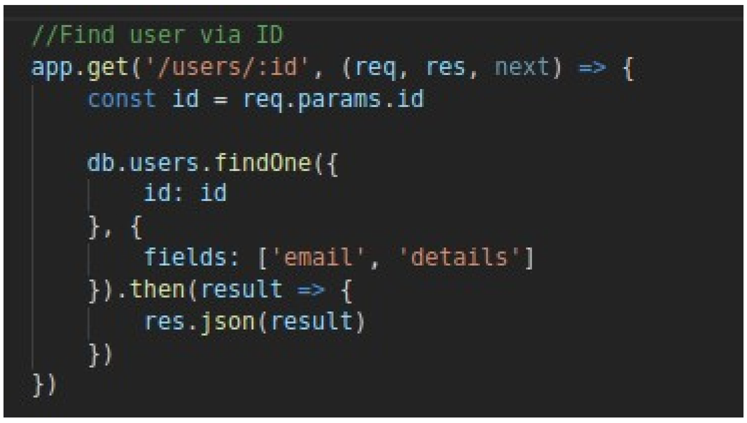


```
//Find all Users
app.get('/users', (req, res, next) => {
    db.users.find({}, {
        fields: ['email', 'details'],
        order: [{field: 'created_at', direction: 'desc'}]
    }).then(result => {
        res.json(result)
    })
})
```

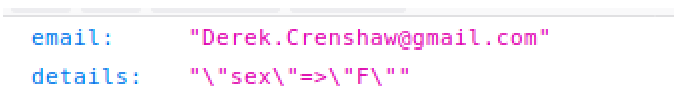


```
▼ 0:
    email:     "Shari.Julian@yahoo.com"
    details:   "\"sex\"=>\"M\""
▼ 1:
    email:     "Evelyn.Patnode@gmail.com"
    details:   "\"sex\"=>\"M\""
▼ 2:
    email:     "Layne.Sarver@aol.com"
    details:   "\"sex\"=>\"M\""
▼ 3:
    email:     "Quinton.Gilpatrick@yahoo.com"
    details:   "\"sex\"=>\"M\""
▼ 4:
```

1.b. GET /users/:id

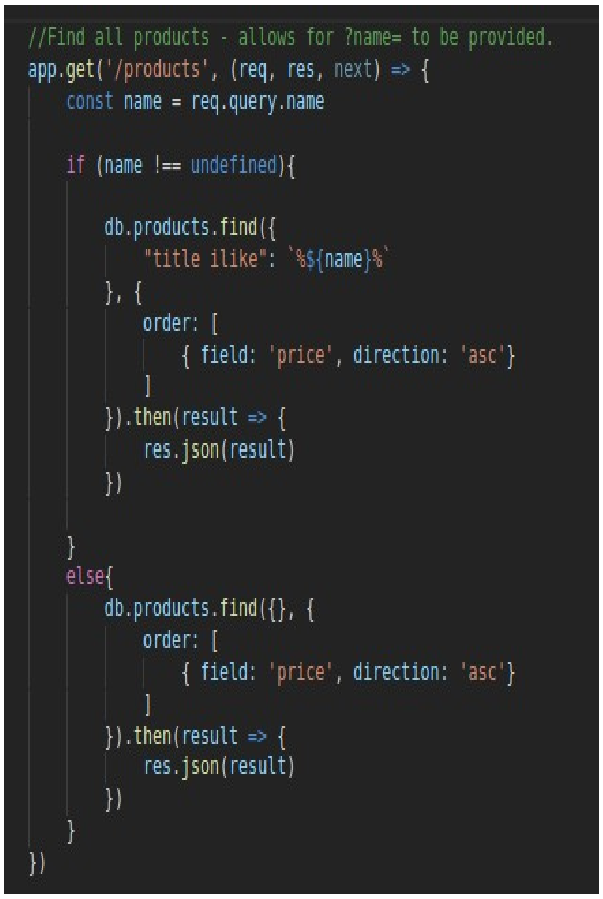


```
//Find user via ID
app.get('/users/:id', (req, res, next) => {
    const id = req.params.id

    db.users.findOne({
        id: id
    }, {
        fields: ['email', 'details']
    }).then(result => {
        res.json(result)
    })
})
```

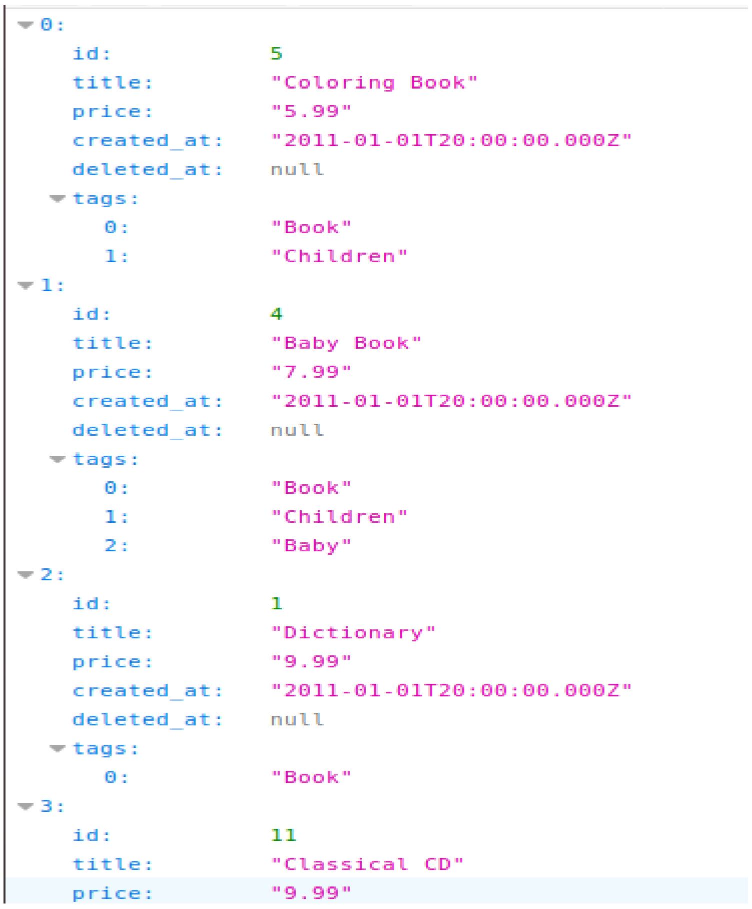


```
email:     "Derek.Crenshaw@gmail.com"
details:   "\"sex\"=>\"F\""
```

1.c. GET /products

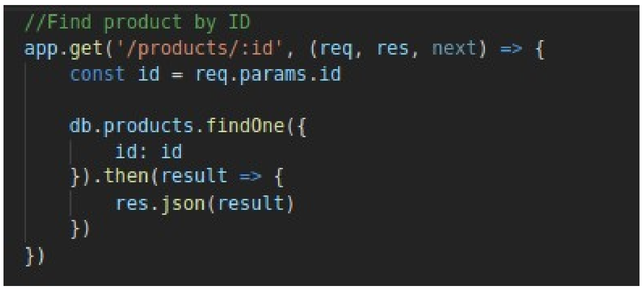


```
//Find all products - allows for ?name= to be provided.
app.get('/products', (req, res, next) => {
    const name = req.query.name

    if (name !== undefined){

        db.products.find({
            "title ilike": `%${name}%`
        }, {
            order: [
                { field: 'price', direction: 'asc'}
            ]
        }).then(result => {
            res.json(result)
        })

    }
    else{
        db.products.find({}, {
            order: [
                { field: 'price', direction: 'asc'}
            ]
        }).then(result => {
            res.json(result)
        })
    }
})
```

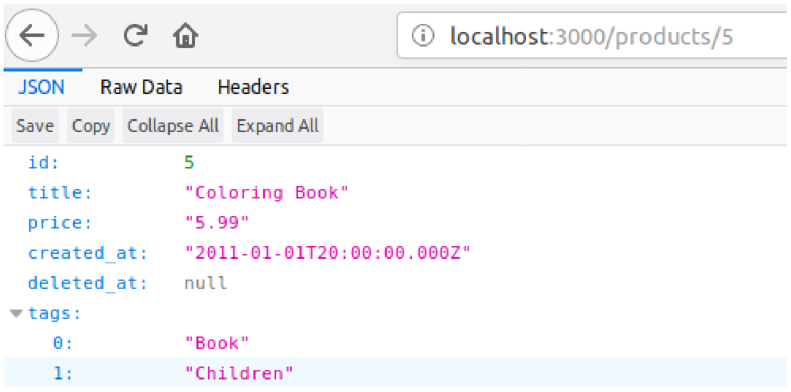


```
▼ 0:
    id:            5
    title:         "Coloring Book"
    price:         "5.99"
    created_at:    "2011-01-01T20:00:00.000Z"
    deleted_at:    null
  ▼ tags:
      0:           "Book"
      1:           "Children"
▼ 1:
    id:            4
    title:         "Baby Book"
    price:         "7.99"
    created_at:    "2011-01-01T20:00:00.000Z"
    deleted_at:    null
  ▼ tags:
      0:           "Book"
      1:           "Children"
      2:           "Baby"
▼ 2:
    id:            1
    title:         "Dictionary"
    price:         "9.99"
    created_at:    "2011-01-01T20:00:00.000Z"
    deleted_at:    null
  ▼ tags:
      0:           "Book"
▼ 3:
    id:            11
    title:         "Classical CD"
    price:         "9.99"
```

1.d. GET /products/:id

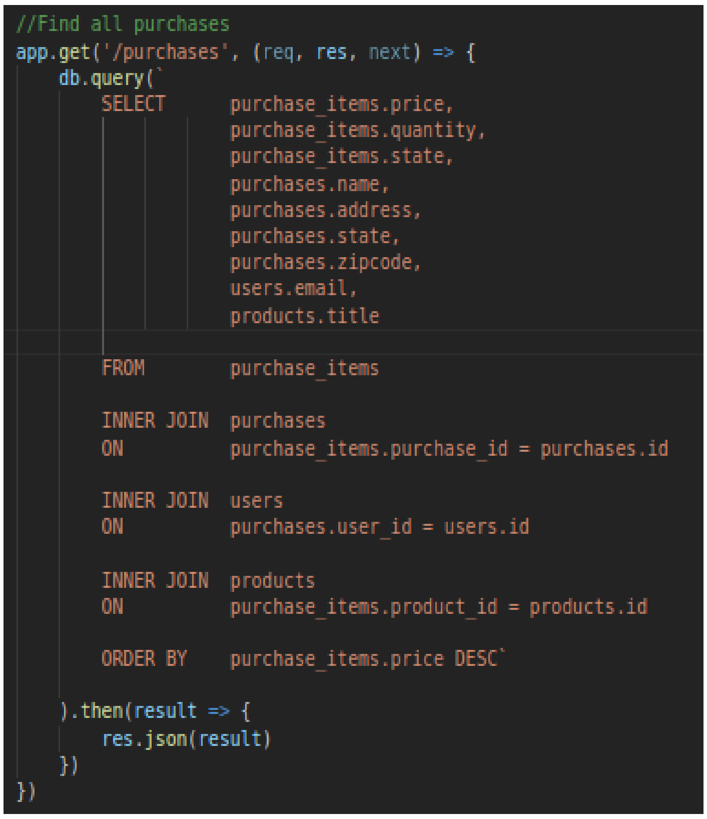


```
//Find product by ID
app.get('/products/:id', (req, res, next) => {
    const id = req.params.id

    db.products.findOne({
        id: id
    }).then(result => {
        res.json(result)
    })
})
```

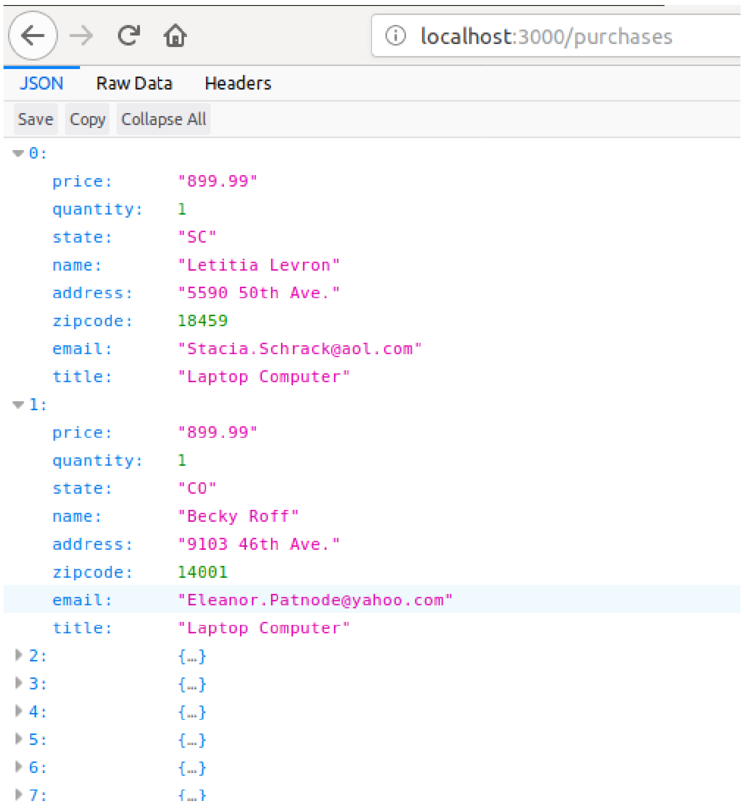


```
id:            5
title:         "Coloring Book"
price:         "5.99"
created_at:    "2011-01-01T20:00:00.000Z"
deleted_at:    null
▼ tags:
    0:         "Book"
    1:         "Children"
```

1.e. GET /purchases



```
//Find all purchases
app.get('/purchases', (req, res, next) => {
    db.query(`
        SELECT      purchase_items.price,
                    purchase_items.quantity,
                    purchase_items.state,
                    purchases.name,
                    purchases.address,
                    purchases.state,
                    purchases.zipcode,
                    users.email,
                    products.title

        FROM        purchase_items

        INNER JOIN  purchases
        ON          purchase_items.purchase_id = purchases.id

        INNER JOIN  users
        ON          purchases.user_id = users.id

        INNER JOIN  products
        ON          purchase_items.product_id = products.id

        ORDER BY    purchase_items.price DESC`

    ).then(result => {
        res.json(result)
    })
})
```



localhost:3000/purchases

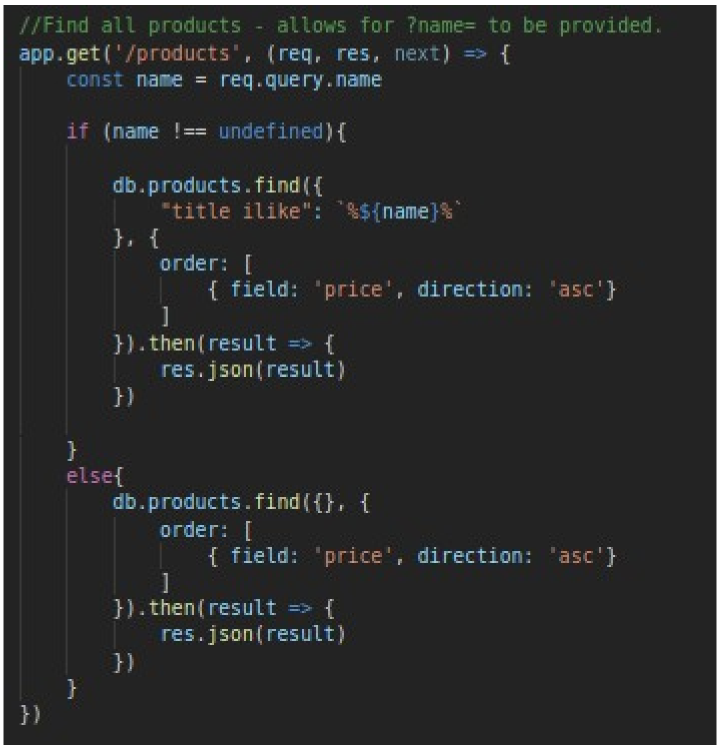
JSON   Raw Data   Headers

Save Copy Collapse All

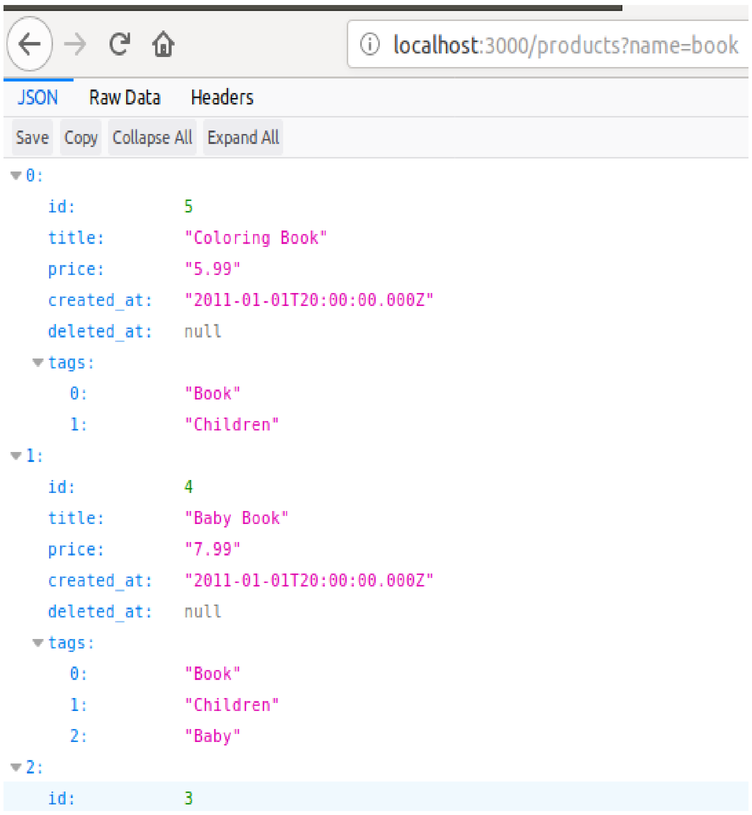
```
▼0:
    price:       "899.99"
    quantity:    1
    state:       "SC"
    name:        "Letitia Levron"
    address:     "5590 50th Ave."
    zipcode:     18459
    email:       "Stacia.Schrack@aol.com"
    title:       "Laptop Computer"
▼1:
    price:       "899.99"
    quantity:    1
    state:       "CO"
    name:        "Becky Roff"
    address:     "9103 46th Ave."
    zipcode:     14001
    email:       "Eleanor.Patnode@yahoo.com"
    title:       "Laptop Computer"
▶2:              {…}
▶3:              {…}
▶4:              {…}
▶5:              {…}
▶6:              {…}
▶7:              {…}
```

Problem 2 :

2.a. GET /products[?name=string]



```
//Find all products - allows for ?name= to be provided.
app.get('/products', (req, res, next) => {
    const name = req.query.name

    if (name !== undefined){

        db.products.find({
            "title ilike": `%${name}%`
        }, {
            order: [
                { field: 'price', direction: 'asc'}
            ]
        }).then(result => {
            res.json(result)
        })

    }
    else{
        db.products.find({}, {
            order: [
                { field: 'price', direction: 'asc'}
            ]
        }).then(result => {
            res.json(result)
        })
    }
})
```



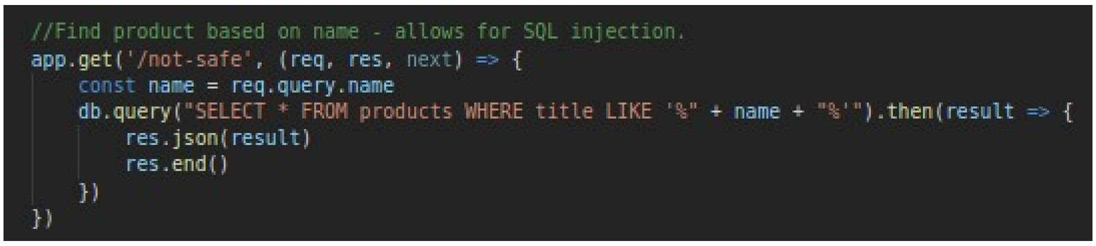
localhost:3000/products?name=book

JSON   Raw Data   Headers

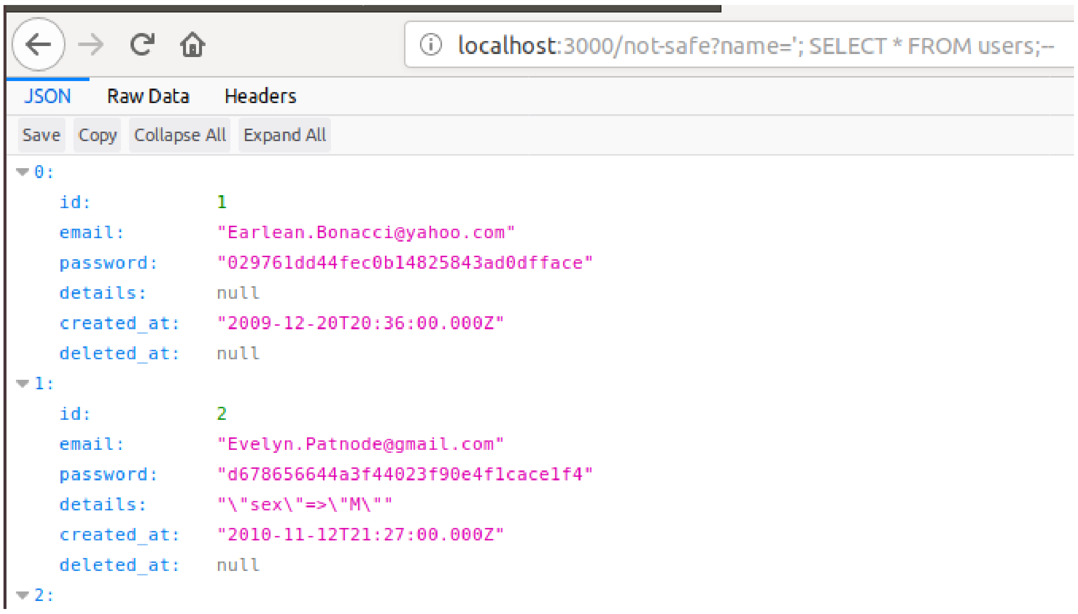
Save Copy Collapse All Expand All

```
▼0:
    id:           5
    title:        "Coloring Book"
    price:        "5.99"
    created_at:   "2011-01-01T20:00:00.000Z"
    deleted_at:   null
  ▼tags:
      0:          "Book"
      1:          "Children"
▼1:
    id:           4
    title:        "Baby Book"
    price:        "7.99"
    created_at:   "2011-01-01T20:00:00.000Z"
    deleted_at:   null
  ▼tags:
      0:          "Book"
      1:          "Children"
      2:          "Baby"
▼2:
    id:           3
```

2.b. SQL Injection (Bad way)



```
//Find product based on name - allows for SQL injection.
app.get('/not-safe', (req, res, next) => {
    const name = req.query.name
    db.query("SELECT * FROM products WHERE title LIKE '%" + name + "%'").then(result => {
        res.json(result)
        res.end()
    })
})
```



localhost:3000/not-safe?name='; SELECT * FROM users;--

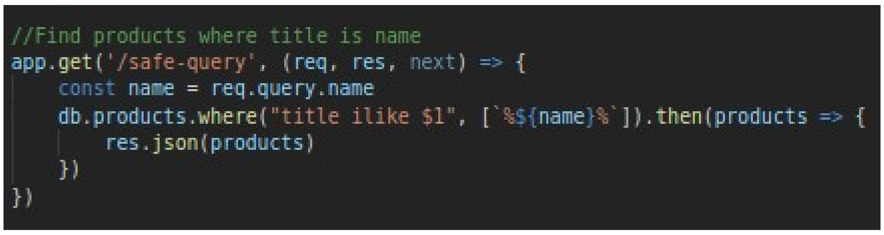
JSON Raw Data Headers

Save Copy Collapse All Expand All

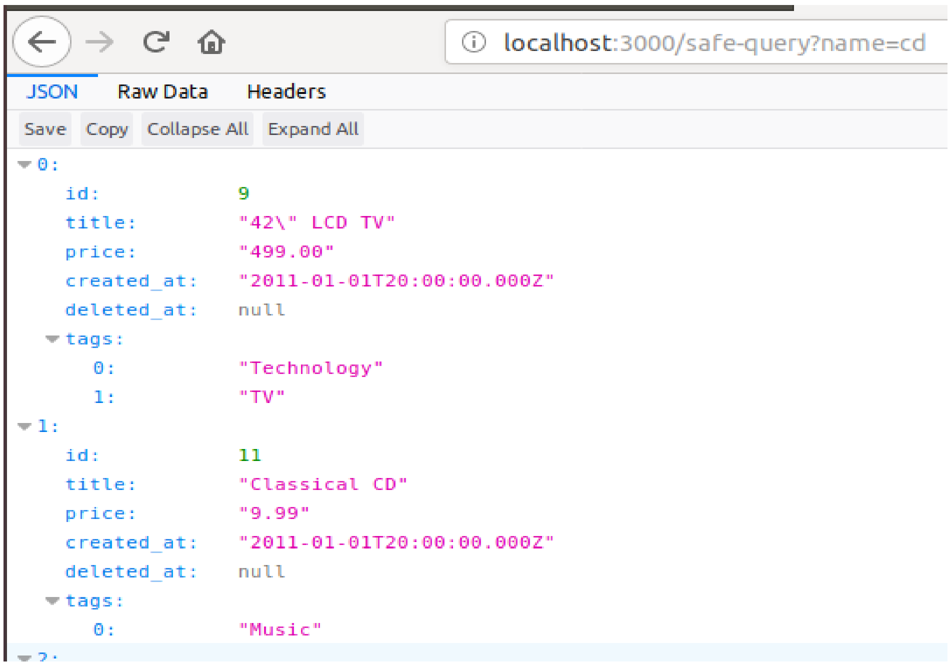
```
▼0:
   id:           1
   email:        "Earlean.Bonacci@yahoo.com"
   password:     "029761dd44fec0b14825843ad0dfface"
   details:      null
   created_at:   "2009-12-20T20:36:00.000Z"
   deleted_at:   null
▼1:
   id:           2
   email:        "Evelyn.Patnode@gmail.com"
   password:     "d678656644a3f44023f90e4f1cace1f4"
   details:      "\"sex\"=>\"M\""
   created_at:   "2010-11-12T21:27:00.000Z"
   deleted_at:   null
▼2:
```

Problem 3:

3.a. Using a parameterised query



```
//Find products where title is name
app.get('/safe-query', (req, res, next) => {
    const name = req.query.name
    db.products.where("title ilike $1", [`%${name}%`]).then(products => {
        res.json(products)
    })
})
```



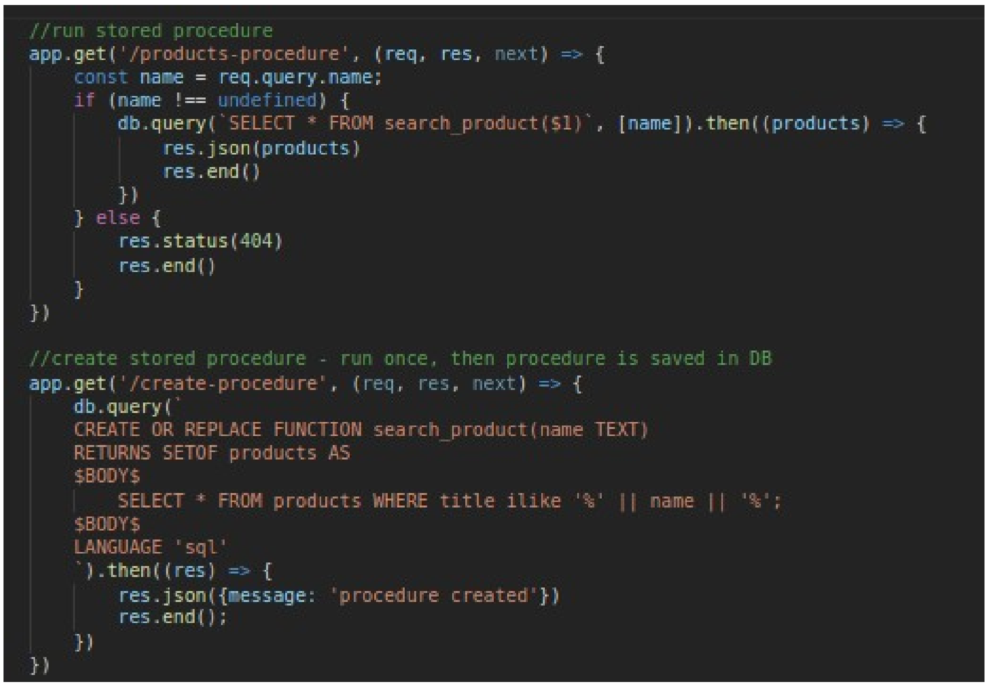
localhost:3000/safe-query?name=cd

JSON Raw Data Headers

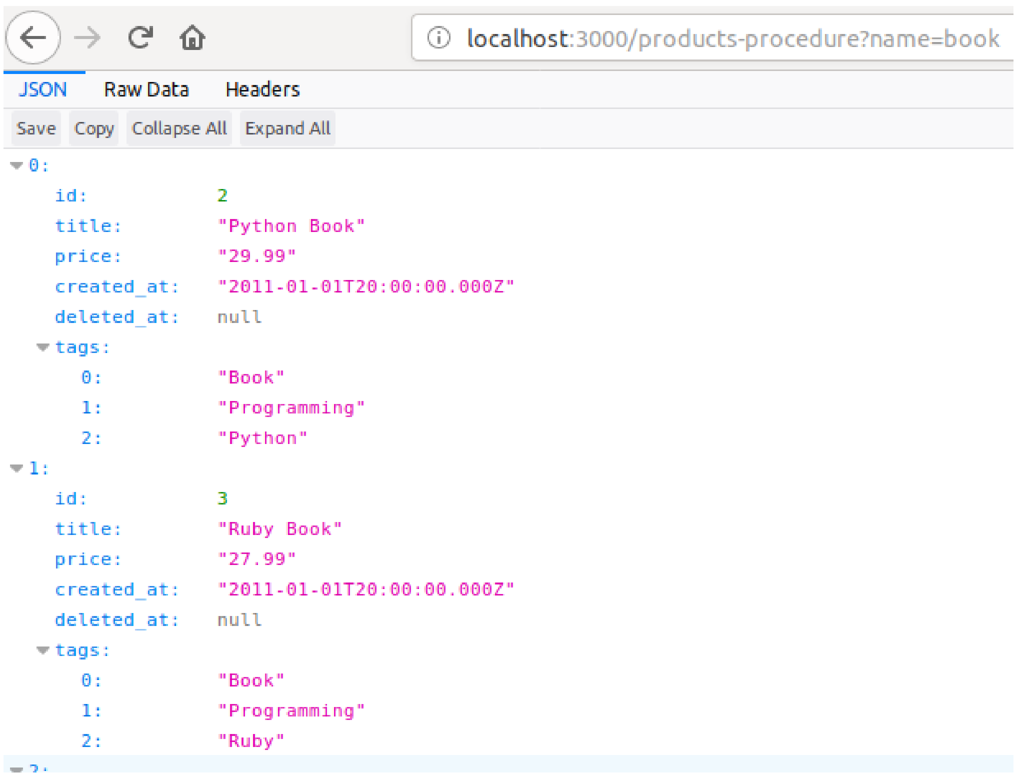
Save Copy Collapse All Expand All

```
▼0:
   id:           9
   title:        "42\" LCD TV"
   price:        "499.00"
   created_at:   "2011-01-01T20:00:00.000Z"
   deleted_at:   null
 ▼tags:
     0:          "Technology"
     1:          "TV"
▼1:
   id:           11
   title:        "Classical CD"
   price:        "9.99"
   created_at:   "2011-01-01T20:00:00.000Z"
   deleted_at:   null
 ▼tags:
     0:          "Music"
▼2:
```

3.b. Using a Stored Procedure

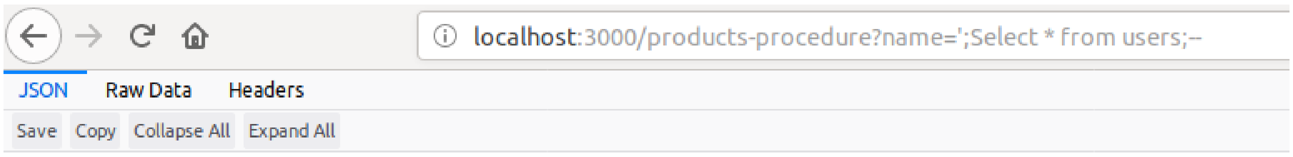


```
//run stored procedure
app.get('/products-procedure', (req, res, next) => {
    const name = req.query.name;
    if (name !== undefined) {
        db.query(`SELECT * FROM search_product($1)`, [name]).then((products) => {
            res.json(products)
            res.end()
        })
    } else {
        res.status(404)
        res.end()
    }
})

//create stored procedure - run once, then procedure is saved in DB
app.get('/create-procedure', (req, res, next) => {
    db.query(`
    CREATE OR REPLACE FUNCTION search_product(name TEXT)
    RETURNS SETOF products AS
    $BODY$
        SELECT * FROM products WHERE title ilike '%' || name || '%';
    $BODY$
    LANGUAGE 'sql'
    `).then((res) => {
        res.json({message: 'procedure created'})
        res.end();
    })
})
```



localhost:3000/products-procedure?name=book

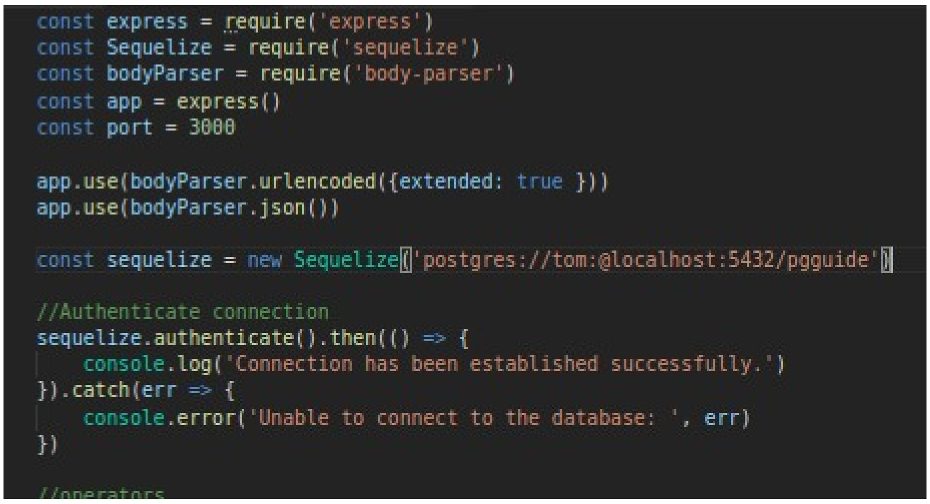
```
▼ 0:
    id:           2
    title:        "Python Book"
    price:        "29.99"
    created_at:   "2011-01-01T20:00:00.000Z"
    deleted_at:   null
  ▼ tags:
      0:          "Book"
      1:          "Programming"
      2:          "Python"
▼ 1:
    id:           3
    title:        "Ruby Book"
    price:        "27.99"
    created_at:   "2011-01-01T20:00:00.000Z"
    deleted_at:   null
  ▼ tags:
      0:          "Book"
      1:          "Programming"
      2:          "Ruby"
▼ 2:
```



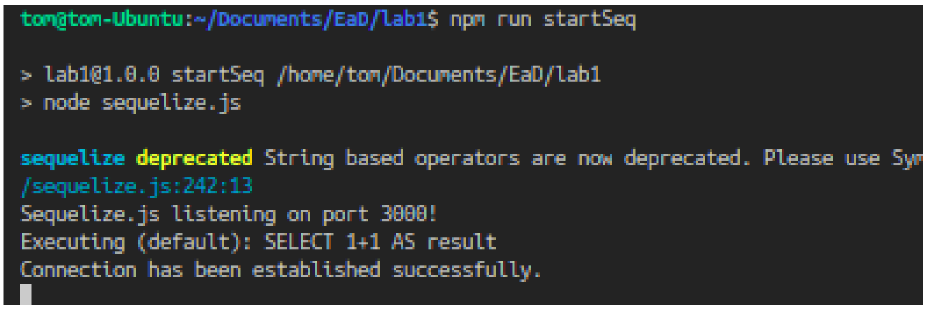
localhost:3000/products-procedure?name=';Select * from users;--

Problem 4:

4.a. Sequelize ORM



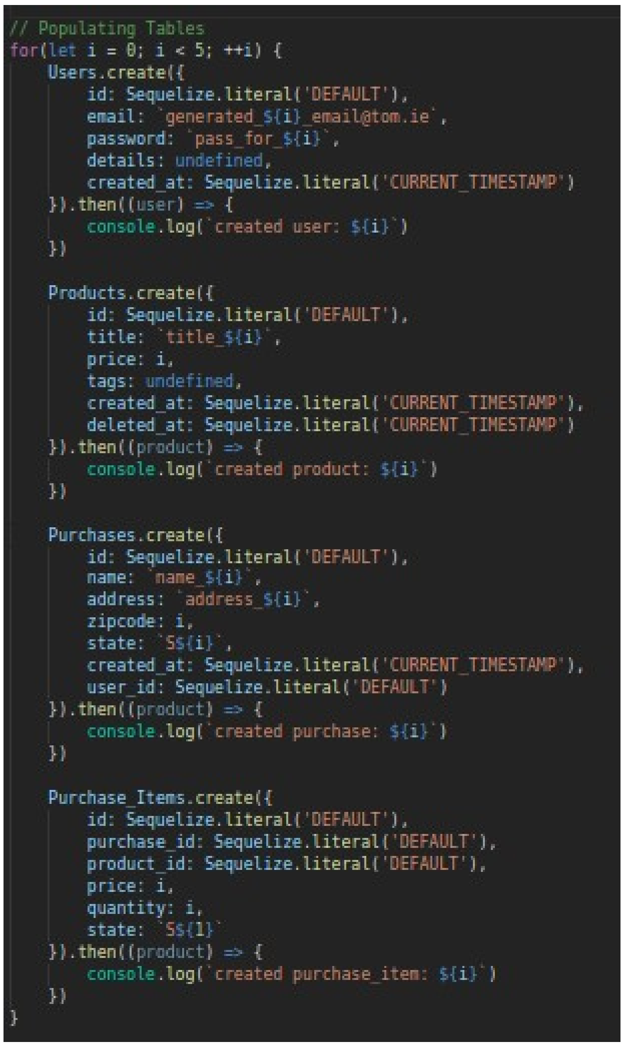
```
const express = require('express')
const Sequelize = require('sequelize')
const bodyParser = require('body-parser')
const app = express()
const port = 3000

app.use(bodyParser.urlencoded({extended: true }))
app.use(bodyParser.json())

const sequelize = new Sequelize('postgres://tom:@localhost:5432/pgguide')

//Authenticate connection
sequelize.authenticate().then(() => {
    console.log('Connection has been established successfully.')
}).catch(err => {
    console.error('Unable to connect to the database: ', err)
})

//operators
```



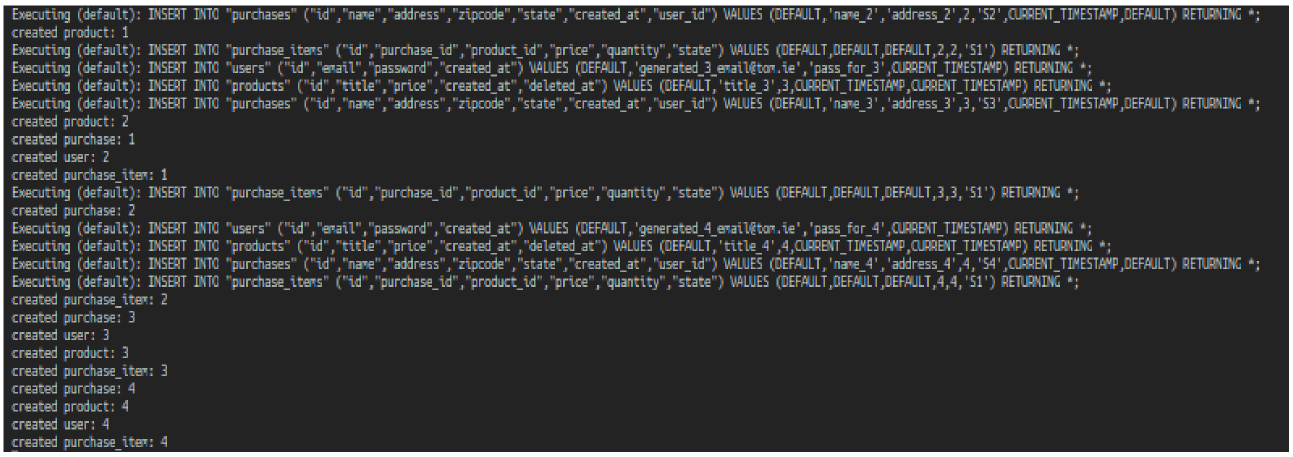
```
tom@tom-Ubuntu:~/Documents/EaD/lab1$ npm run startSeq

> lab1@1.0.0 startSeq /home/tom/Documents/EaD/lab1
> node sequelize.js

sequelize deprecated String based operators are now deprecated. Please use Sy
/sequelize.js:242:13
Sequelize.js listening on port 3000!
Executing (default): SELECT 1+1 AS result
Connection has been established successfully.
```

Problem 5:

5.a. Populate the Database with some additional test data for all methods.



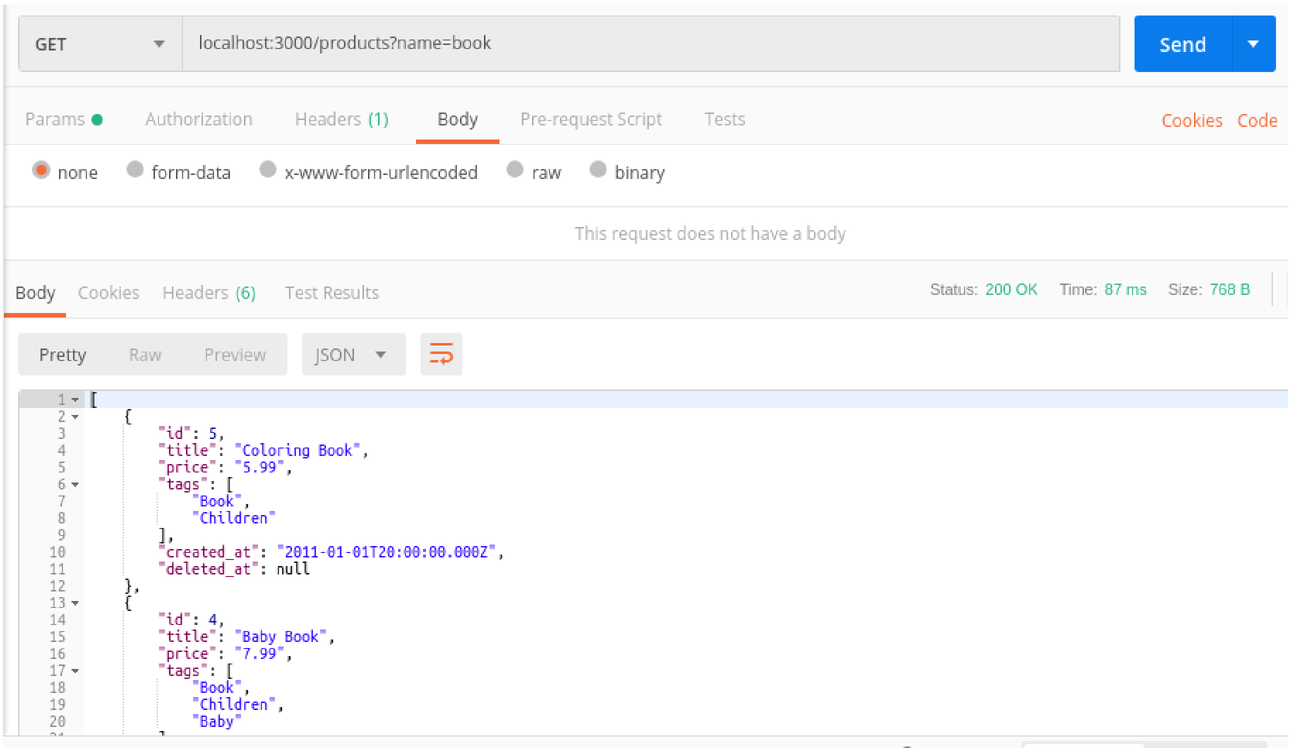
```
// Populating Tables
for(let i = 0; i < 5; ++i) {
    Users.create({
        id: Sequelize.literal('DEFAULT'),
        email: `generated_${i}_email@tom.ie`,
        password: `pass_for_${i}`,
        details: undefined,
        created_at: Sequelize.literal('CURRENT_TIMESTAMP')
    }).then((user) => {
        console.log(`created user: ${i}`)
    })

    Products.create({
        id: Sequelize.literal('DEFAULT'),
        title: `title_${i}`,
        price: i,
        tags: undefined,
        created_at: Sequelize.literal('CURRENT_TIMESTAMP'),
        deleted_at: Sequelize.literal('CURRENT_TIMESTAMP')
    }).then((product) => {
        console.log(`created product: ${i}`)
    })

    Purchases.create({
        id: Sequelize.literal('DEFAULT'),
        name: `name_${i}`,
        address: `address_${i}`,
        zipcode: i,
        state: `S${i}`,
        created_at: Sequelize.literal('CURRENT_TIMESTAMP'),
        user_id: Sequelize.literal('DEFAULT')
    }).then((product) => {
        console.log(`created purchase: ${i}`)
    })

    Purchase_Items.create({
        id: Sequelize.literal('DEFAULT'),
        purchase_id: Sequelize.literal('DEFAULT'),
        product_id: Sequelize.literal('DEFAULT'),
        price: i,
        quantity: i,
        state: `S${1}`
    }).then((product) => {
        console.log(`created purchase_item: ${i}`)
    })
}
```



```
Executing (default): INSERT INTO "purchases" ("id","name","address","zipcode","state","created_at","user_id") VALUES (DEFAULT,'name_2','address_2',2,'S2',CURRENT_TIMESTAMP,DEFAULT) RETURNING *;
created product: 1
Executing (default): INSERT INTO "purchase_items" ("id","purchase_id","product_id","price","quantity","state") VALUES (DEFAULT,DEFAULT,DEFAULT,2,2,'S1') RETURNING *;
Executing (default): INSERT INTO "users" ("id","email","password","created_at") VALUES (DEFAULT,'generated_3_email@tom.ie','pass_for_3',CURRENT_TIMESTAMP) RETURNING *;
Executing (default): INSERT INTO "products" ("id","title","price","created_at","deleted_at") VALUES (DEFAULT,'title_3',3,CURRENT_TIMESTAMP,CURRENT_TIMESTAMP) RETURNING *;
Executing (default): INSERT INTO "purchases" ("id","name","address","zipcode","state","created_at","user_id") VALUES (DEFAULT,'name_3','address_3',3,'S3',CURRENT_TIMESTAMP,DEFAULT) RETURNING *;
created product: 2
created purchase: 1
created user: 2
created purchase_item: 1
Executing (default): INSERT INTO "purchase_items" ("id","purchase_id","product_id","price","quantity","state") VALUES (DEFAULT,DEFAULT,DEFAULT,3,3,'S1') RETURNING *;
created purchase: 2
Executing (default): INSERT INTO "users" ("id","email","password","created_at") VALUES (DEFAULT,'generated_4_email@tom.ie','pass_for_4',CURRENT_TIMESTAMP) RETURNING *;
Executing (default): INSERT INTO "products" ("id","title","price","created_at","deleted_at") VALUES (DEFAULT,'title_4',4,CURRENT_TIMESTAMP,CURRENT_TIMESTAMP) RETURNING *;
Executing (default): INSERT INTO "purchases" ("id","name","address","zipcode","state","created_at","user_id") VALUES (DEFAULT,'name_4','address_4',4,'S4',CURRENT_TIMESTAMP,DEFAULT) RETURNING *;
Executing (default): INSERT INTO "purchase_items" ("id","purchase_id","product_id","price","quantity","state") VALUES (DEFAULT,DEFAULT,DEFAULT,4,4,'S1') RETURNING *;
created purchase_item: 2
created purchase: 3
created user: 3
created product: 3
created purchase_item: 3
created purchase: 4
created product: 4
created user: 4
created purchase_item: 4
```

Problem 6:

6.a. GET /products[?name=string]


GET localhost:3000/products?name=book Send

Params Authorization Headers (1) Body Pre-request Script Tests Cookies Code

[✓ none] form-data x-www-form-urlencoded raw binary

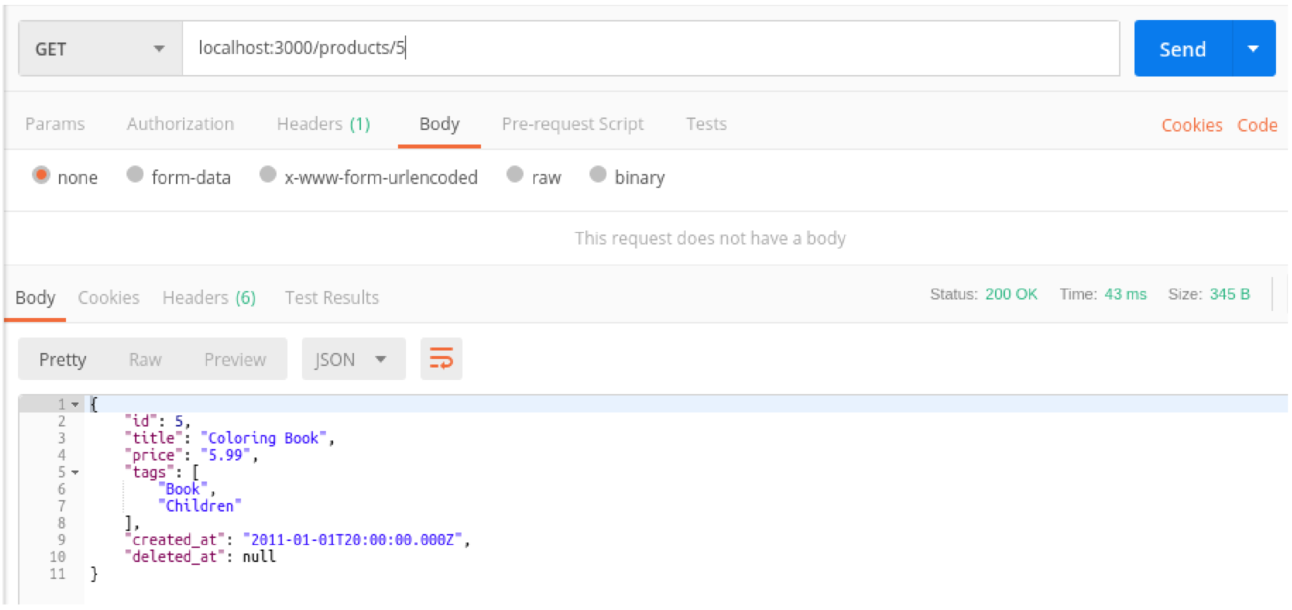
This request does not have a body

Body Cookies Headers (6) Test Results Status: 200 OK Time: 87 ms Size: 768 B

Pretty Raw Preview JSON

```
[
    {
        "id": 5,
        "title": "Coloring Book",
        "price": "5.99",
        "tags": [
            "Book",
            "Children"
        ],
        "created_at": "2011-01-01T20:00:00.000Z",
        "deleted_at": null
    },
    {
        "id": 4,
        "title": "Baby Book",
        "price": "7.99",
        "tags": [
            "Book",
            "Children",
            "Baby"
```

6.b. GET /products/:id


GET localhost:3000/products/5 Send

Params Authorization Headers (1) Body Pre-request Script Tests Cookies Code

[✓ none] form-data x-www-form-urlencoded raw binary

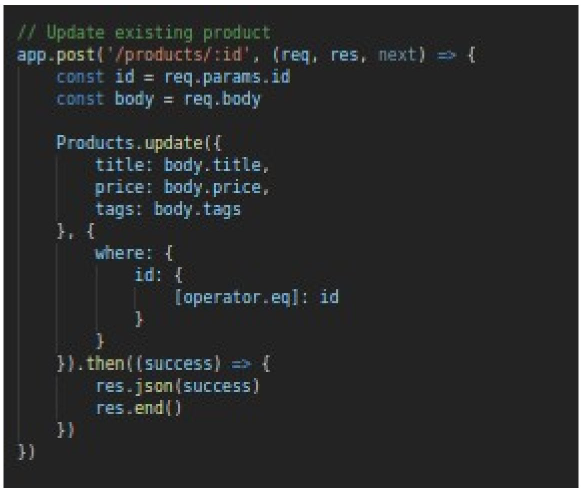
This request does not have a body

Body Cookies Headers (6) Test Results Status: 200 OK Time: 43 ms Size: 345 B

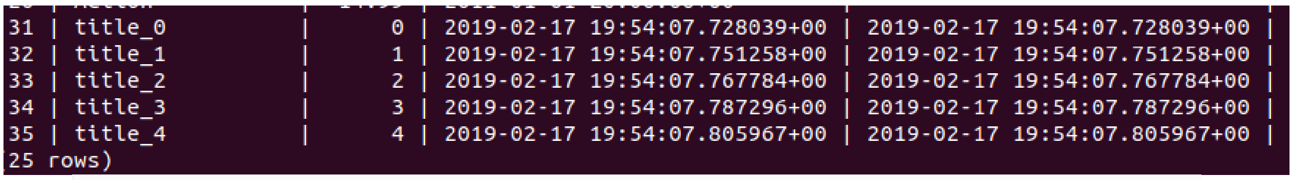
Pretty Raw Preview JSON

```
{
    "id": 5,
    "title": "Coloring Book",
    "price": "5.99",
    "tags": [
        "Book",
        "Children"
    ],
    "created_at": "2011-01-01T20:00:00.000Z",
    "deleted_at": null
}
```

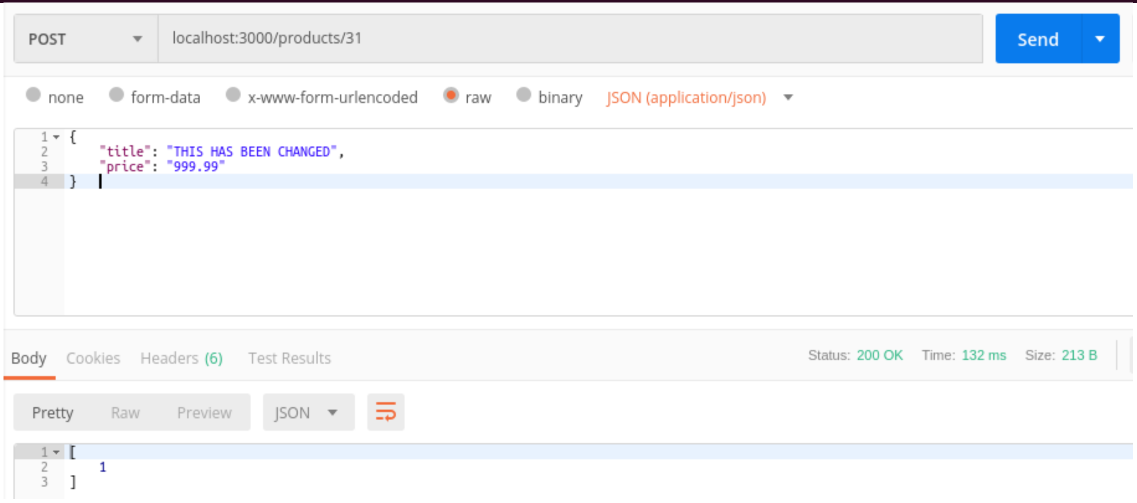
6.c. POST /products



```
// Update existing product
app.post('/products/:id', (req, res, next) => {
    const id = req.params.id
    const body = req.body

    Products.update({
        title: body.title,
        price: body.price,
        tags: body.tags
    }, {
        where: {
            id: {
                [operator.eq]: id
            }
        }
    }).then((success) => {
        res.json(success)
        res.end()
    })
})
```



```
31 | title_0              |      0 | 2019-02-17 19:54:07.728039+00 | 2019-02-17 19:54:07.728039+00 |
32 | title_1              |      1 | 2019-02-17 19:54:07.751258+00 | 2019-02-17 19:54:07.751258+00 |
33 | title_2              |      2 | 2019-02-17 19:54:07.767784+00 | 2019-02-17 19:54:07.767784+00 |
34 | title_3              |      3 | 2019-02-17 19:54:07.787296+00 | 2019-02-17 19:54:07.787296+00 |
35 | title_4              |      4 | 2019-02-17 19:54:07.805967+00 | 2019-02-17 19:54:07.805967+00 |
(25 rows)
```



POST localhost:3000/products/31 Send

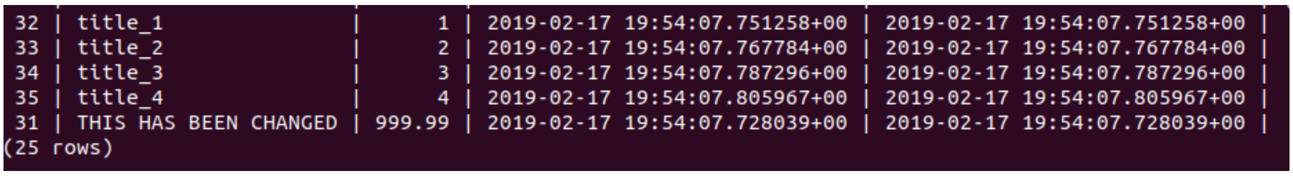
none form-data x-www-form-urlencoded raw binary JSON (application/json)

```
{
    "title": "THIS HAS BEEN CHANGED",
    "price": "999.99"
}
```

Body Cookies Headers (6) Test Results Status: 200 OK Time: 132 ms Size: 213 B

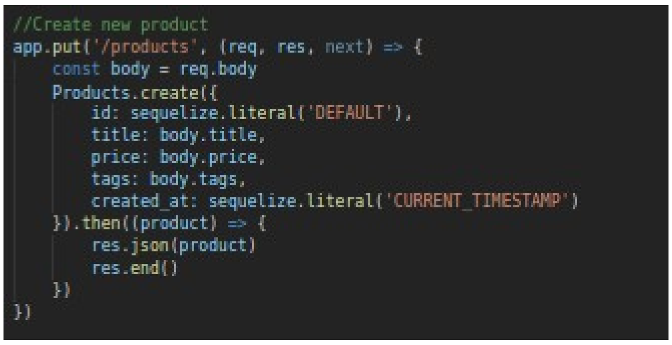
Pretty Raw Preview JSON

```
[
    1
]
```

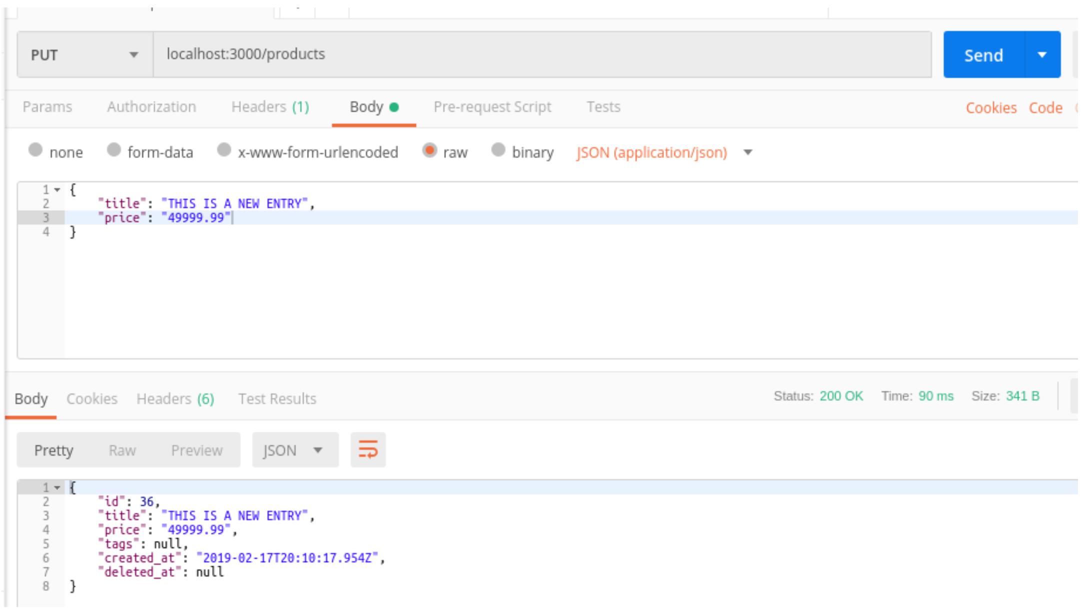


```
32 | title_1              |      1 | 2019-02-17 19:54:07.751258+00 | 2019-02-17 19:54:07.751258+00 |
33 | title_2              |      2 | 2019-02-17 19:54:07.767784+00 | 2019-02-17 19:54:07.767784+00 |
34 | title_3              |      3 | 2019-02-17 19:54:07.787296+00 | 2019-02-17 19:54:07.787296+00 |
35 | title_4              |      4 | 2019-02-17 19:54:07.805967+00 | 2019-02-17 19:54:07.805967+00 |
31 | THIS HAS BEEN CHANGED | 999.99 | 2019-02-17 19:54:07.728039+00 | 2019-02-17 19:54:07.728039+00 |
(25 rows)
```

6.d. PUT /products/:id



```
//Create new product
app.put('/products', (req, res, next) => {
    const body = req.body
    Products.create({
        id: sequelize.literal('DEFAULT'),
        title: body.title,
        price: body.price,
        tags: body.tags,
        created_at: sequelize.literal('CURRENT_TIMESTAMP')
    }).then((product) => {
        res.json(product)
        res.end()
    })
})
```



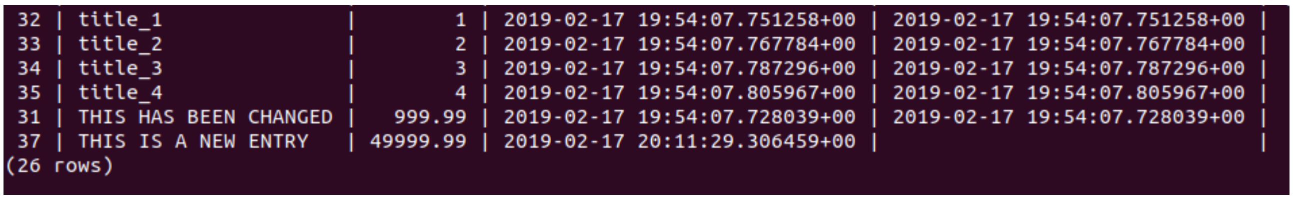
localhost:3000/products

Body: raw, JSON (application/json)

```
{
    "title": "THIS IS A NEW ENTRY",
    "price": "49999.99"
}
```

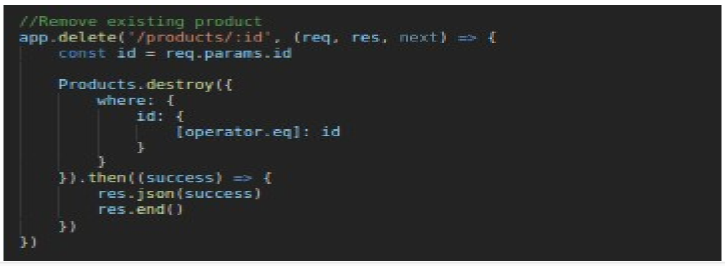
Status: 200 OK Time: 90 ms Size: 341 B

```
{
    "id": 36,
    "title": "THIS IS A NEW ENTRY",
    "price": "49999.99",
    "tags": null,
    "created_at": "2019-02-17T20:10:17.954Z",
    "deleted_at": null
}
```

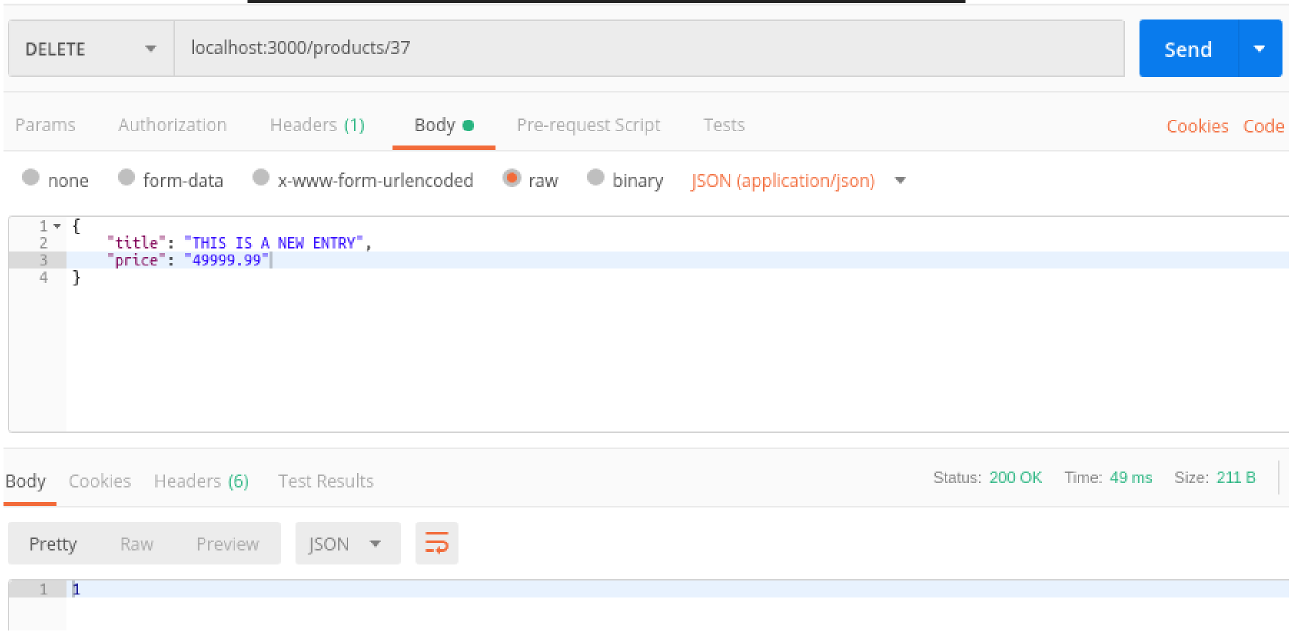


```
 32 | title_1               |        1 | 2019-02-17 19:54:07.751258+00 | 2019-02-17 19:54:07.751258+00 |
 33 | title_2               |        2 | 2019-02-17 19:54:07.767784+00 | 2019-02-17 19:54:07.767784+00 |
 34 | title_3               |        3 | 2019-02-17 19:54:07.787296+00 | 2019-02-17 19:54:07.787296+00 |
 35 | title_4               |        4 | 2019-02-17 19:54:07.805967+00 | 2019-02-17 19:54:07.805967+00 |
 31 | THIS HAS BEEN CHANGED |   999.99 | 2019-02-17 19:54:07.728039+00 | 2019-02-17 19:54:07.728039+00 |
 37 | THIS IS A NEW ENTRY   | 49999.99 | 2019-02-17 20:11:29.306459+00 |                               |
(26 rows)
```

6.e. DELETE /products/:id



```
//Remove existing product
app.delete('/products/:id', (req, res, next) => {
    const id = req.params.id

    Products.destroy({
        where: {
            id: {
                [operator.eq]: id
            }
        }
    }).then((success) => {
        res.json(success)
        res.end()
    })
})
```

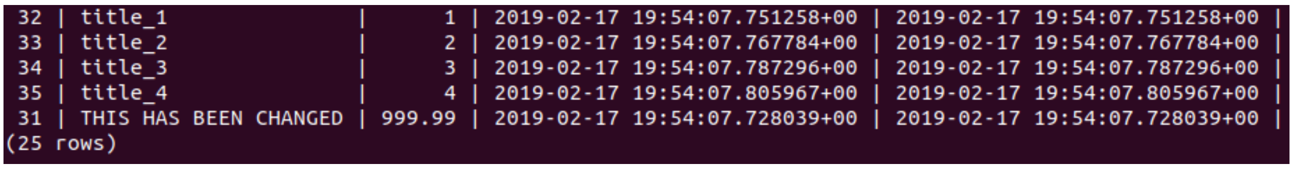


DELETE localhost:3000/products/37

```
{
    "title": "THIS IS A NEW ENTRY",
    "price": "49999.99"
}
```

Status: 200 OK Time: 49 ms Size: 211 B

```
1
```



```
 32 | title_1                 |       1 | 2019-02-17 19:54:07.751258+00 | 2019-02-17 19:54:07.751258+00 |
 33 | title_2                 |       2 | 2019-02-17 19:54:07.767784+00 | 2019-02-17 19:54:07.767784+00 |
 34 | title_3                 |       3 | 2019-02-17 19:54:07.787296+00 | 2019-02-17 19:54:07.787296+00 |
 35 | title_4                 |       4 | 2019-02-17 19:54:07.805967+00 | 2019-02-17 19:54:07.805967+00 |
 31 | THIS HAS BEEN CHANGED | 999.99 | 2019-02-17 19:54:07.728039+00 | 2019-02-17 19:54:07.728039+00 |
(25 rows)
```