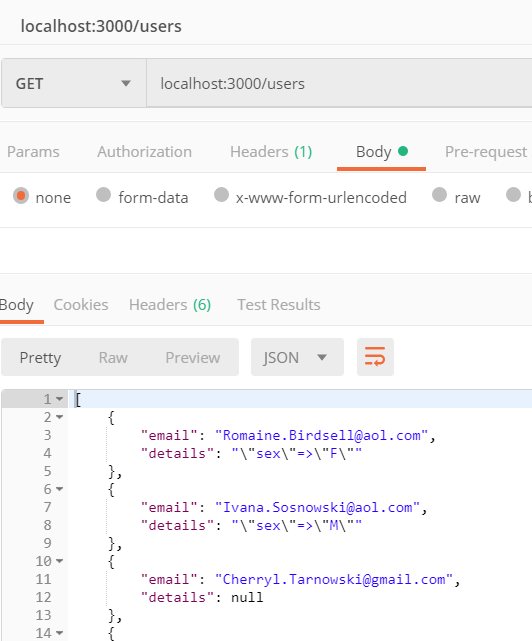
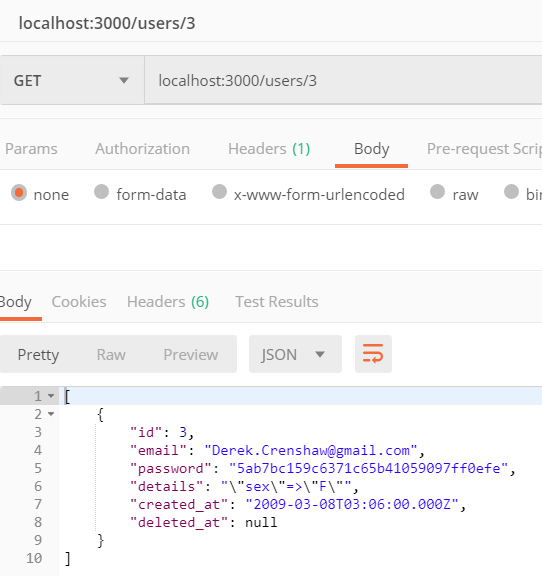
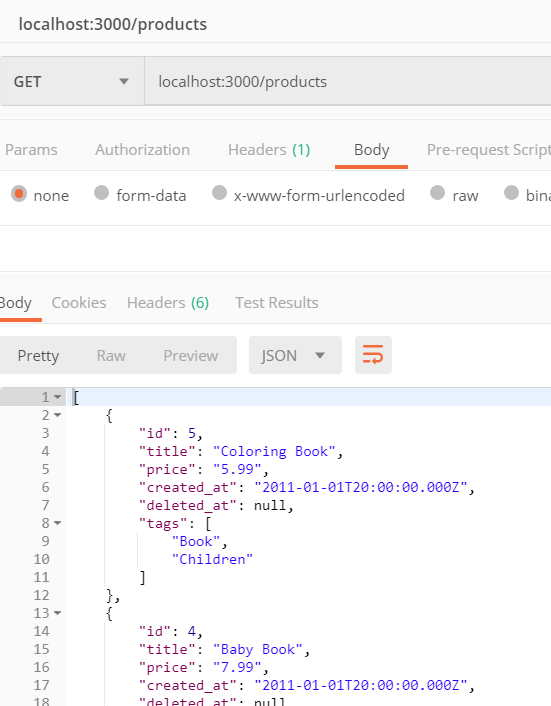
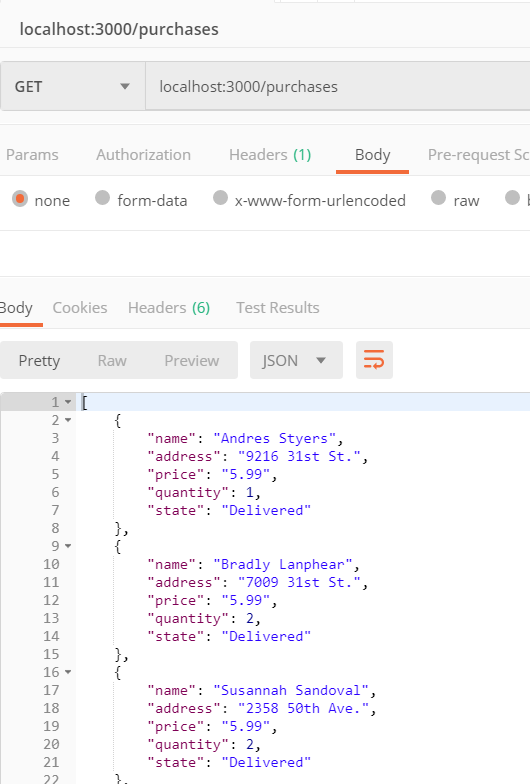
D16123420-wks-1

1.









2.

//SQL inject

app.get('/products', (req, res) => {

var titlename = req.query.titlename

if (titlename == null){

req.app.get('db').query('select \* from products').then(items => {

res.json(items);

});

}

else{

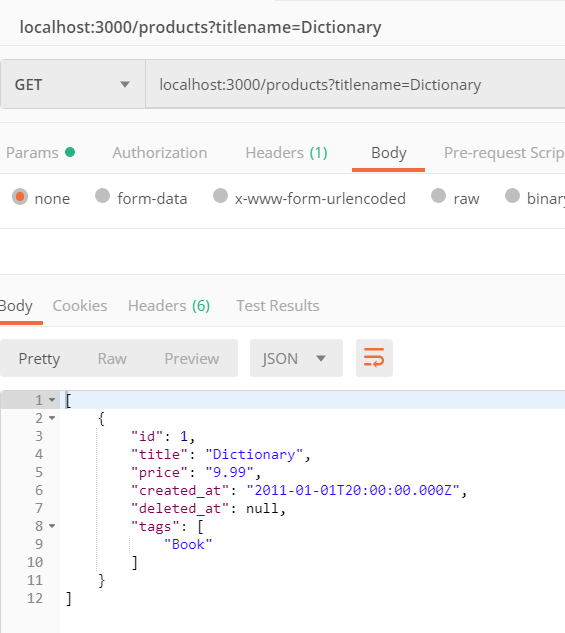
req.app.get('db').query('select \* from products WHERE title = \'' + titlename + '\'').then(items => {

res.json(items);

});

}

});



3.

Using a parameterised query

// use parameterised

app.get('/api/products', (req, res) => {

var titlename = req.query.titlename

if (titlename == null){

req.app.get('db').query('select \* from products').then(items => {

res.json(items);

});

}

else{

req.app.get('db').query('select \* from products WHERE title = ${title}',{

title: titlename

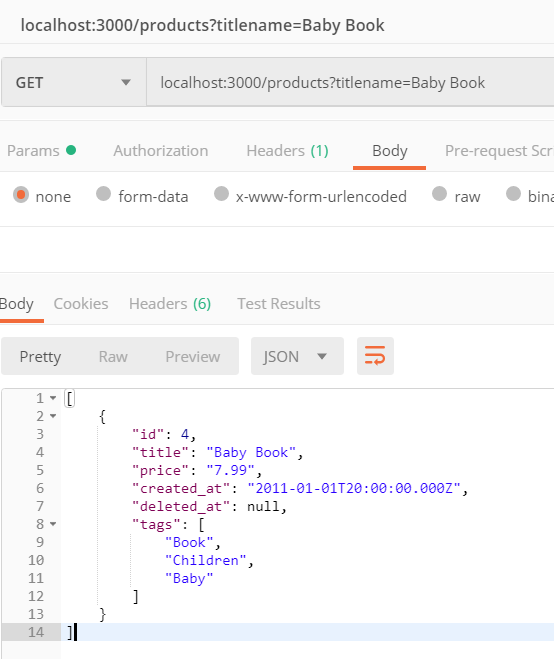
}).then(items => {

res.json(items);

});

}

});



Using a stored procedure using SQL or PLPGSQL whichever you prefer

//use StoredProsedure

/\* create function procedure\_title(titlename varchar(20))

return setof products

language sql

as $$

SELECT \* FROM products where title = titlename

$$;

\*/

app.get('/api/StoredProcedure', (req, res) => {

var titlename = req.query.titlename

if (titlename == null){

req.app.get('db').query('select \* from products').then(items => {

res.json(items);

});

}

else{

req.app.get('db').query("select \* from procedure\_title(\'' + titlename + '\')",).then(items => {

res.json(items);

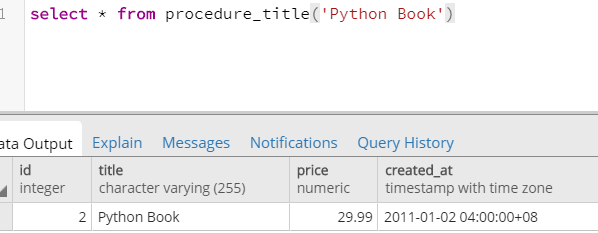
});

}

});

http.createServer(app).listen(3000);

});



4.

Create a folder "product", and create bin and server folders in "product"

npm init – y

npm install --save express body-parser Morgan

Create the app.js file in the current directory

Go into the bin directory and create a WWW file without an extension

npm i -D nodemon

Open package.json and modify line 6

Run file : npm run start:dev (use this for all future runs) test type: localhost:8000 in the browser

npm install - g sequelize – cli

Create the .Sequelizerc file in the current folder

npm install --save sequelize pg pg-hstore

sequelize init

Modify config.json under server\config

sequelize model:create --name Product --attributes title:string,,price:numeric,tags:string

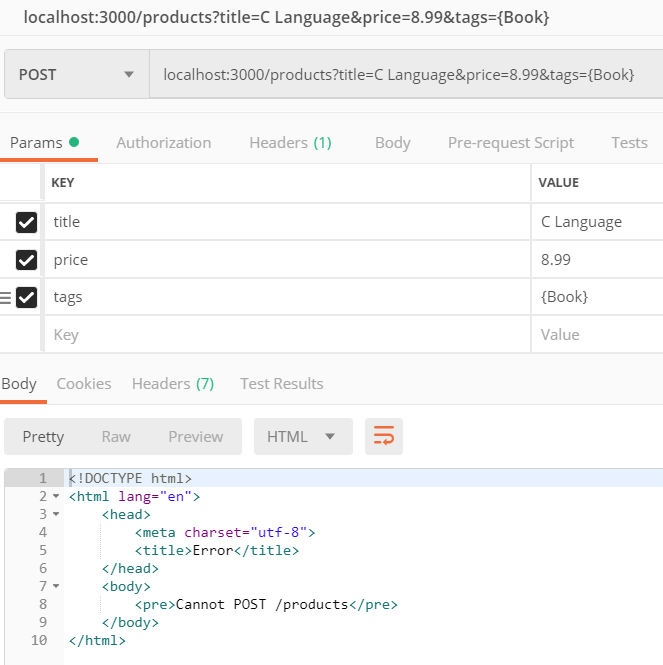
sequelize db:migrate

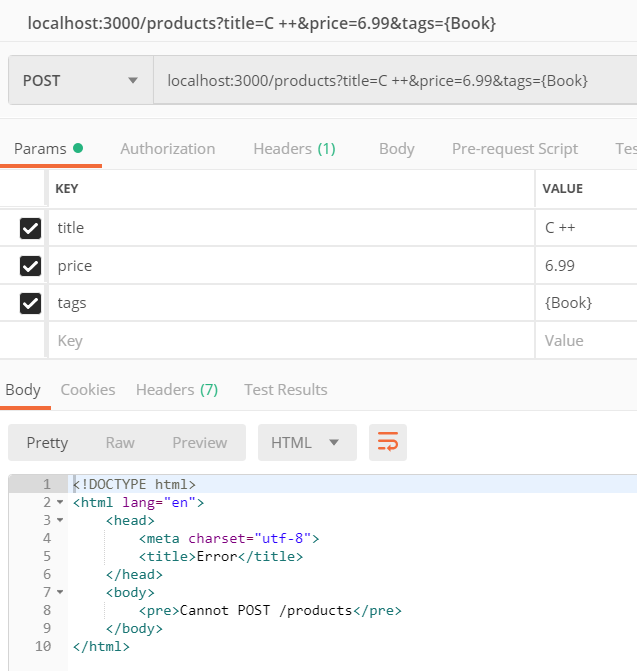
Complete the data migration above and prepare for question 5 and question 6.

5.

POST

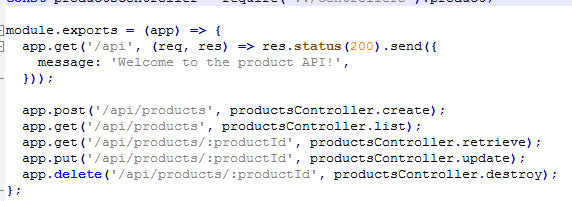
Post two books

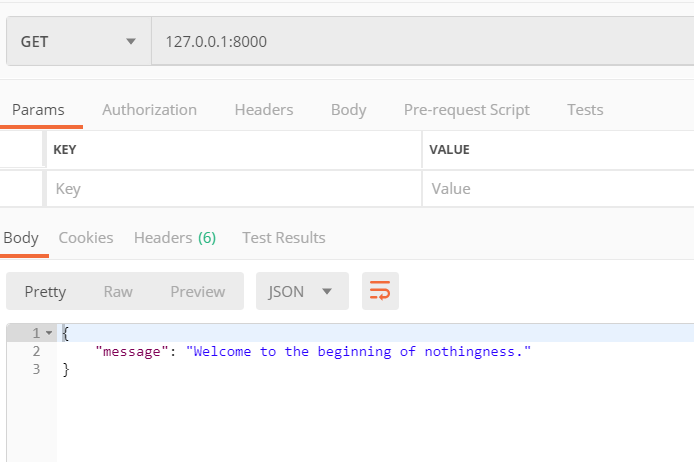


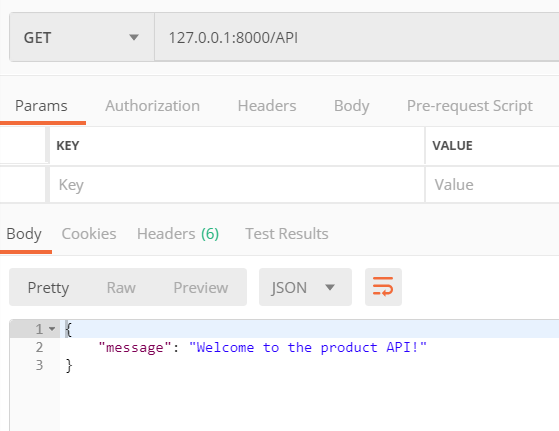


6.

Routing request







GET product ID

retrieve(req, res) {

return Product

.findById(req.params.productId, {

})

.then(product => {

if (!product) {

return res.status(404).send({

message: 'Product Not Found',

});

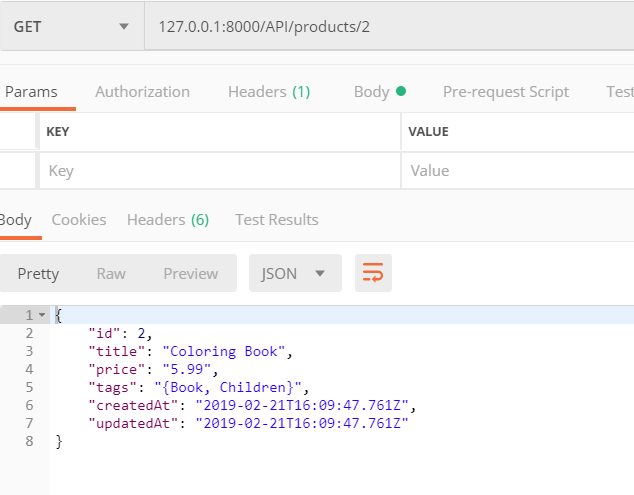
}

return res.status(200).send(product);

})

.catch(error => res.status(400).send(error));

},



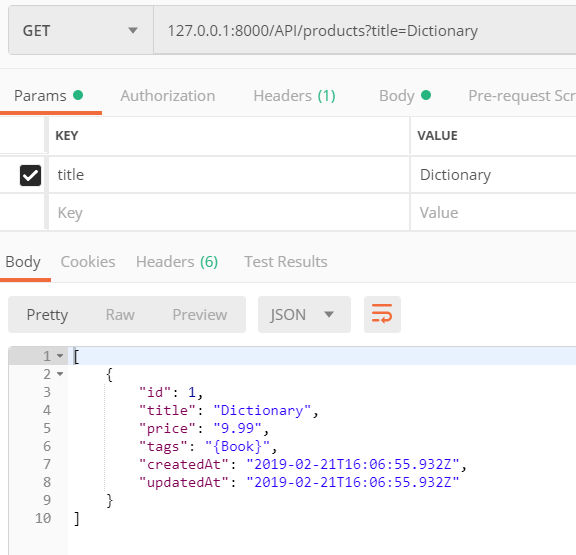
list(req, res) {

return Product

.findAll({where: req.query})

.then(products => res.status(200).send(products))

.catch(error => res.status(400).send(error));



PUT

update(req, res) {

return Product

.findById(req.params.productId, {

})

.then(product => {

if (!product) {

return res.status(404).send({

message: 'Product Not Found',

});

}

return product

.update({

title: req.body.title || product.title,

})

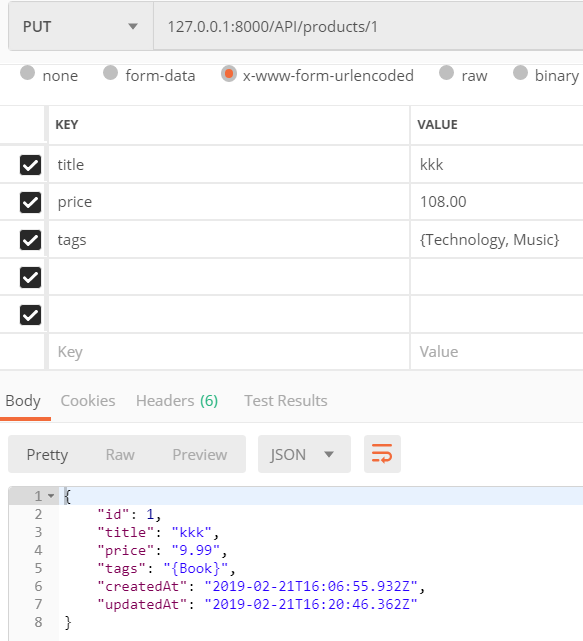
.then(() => res.status(200).send(product))

.catch((error) => res.status(400).send(error));

})

.catch((error) => res.status(400).send(error));

},



Delete

destroy(req, res) {

return Product

.findById(req.params.productId)

.then(product => {

if (!product) {

return res.status(400).send({

message: 'Product Not Found',

});

}

return product

.destroy()

.then(() => res.status(204).send())

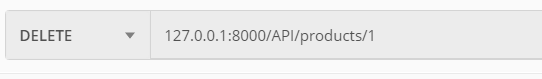
.catch(error => res.status(400).send(error));

})

.catch(error => res.status(400).send(error));

},

};



POST

