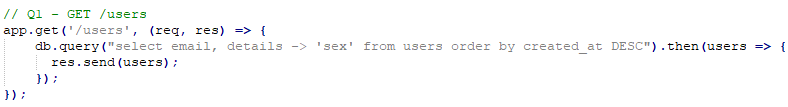
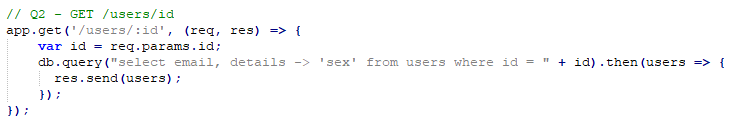
# Problem Set One

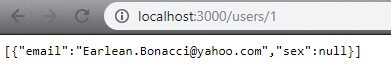
Part One – GET /users



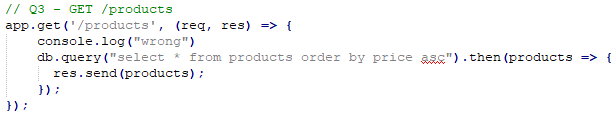


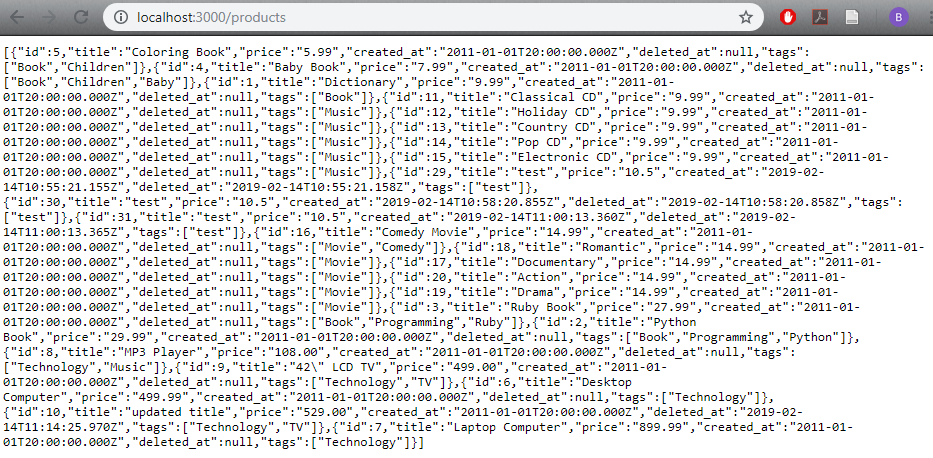
Part Two – GET /users/:id



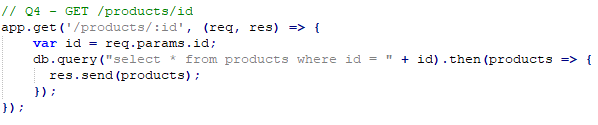


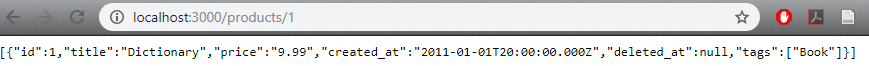
Part Three – GET /products



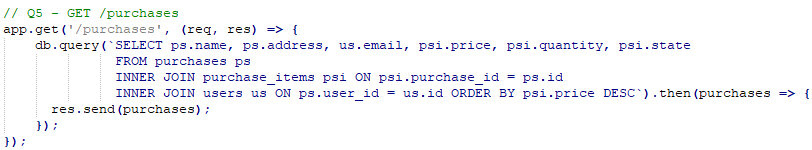


Part Four – GET /products/:id





Part Five – GET /purchases

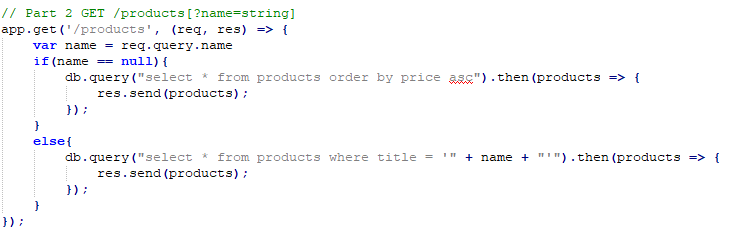


# 

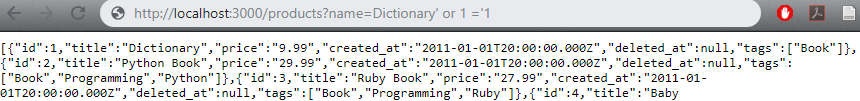
# Problem Set Two

Part One – GET /products[?name=*string*]

If/else statement is used to differentiate between when a user enters string or not.



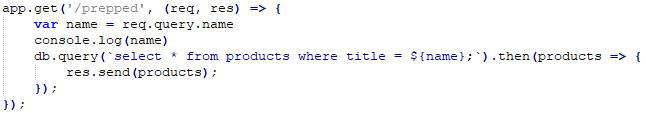
First implementation selects all products



Following query deletes all from the user table

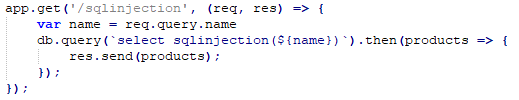
# Problem Set Three

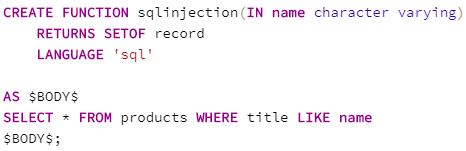
Part One – Parameterised Query

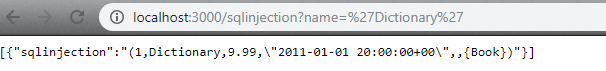


Searches for a single title work, sql injection attempts throw promise errors.

Part Two – Stored Procedure

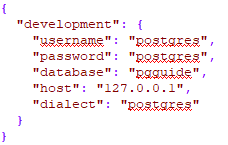






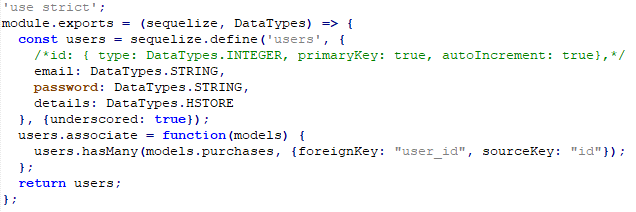
# Problem Set Four

config

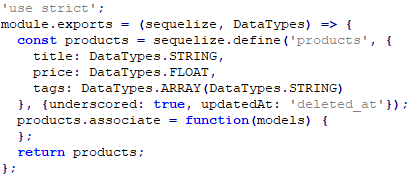


## Models

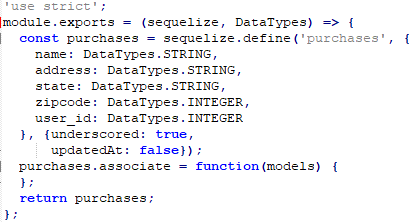
users.js



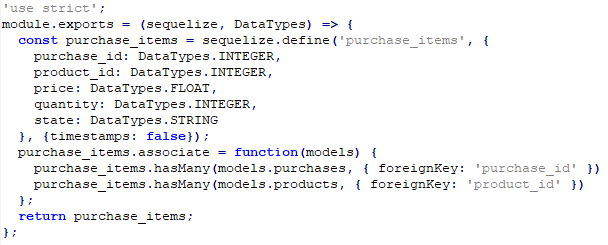
products.js



purchases.js

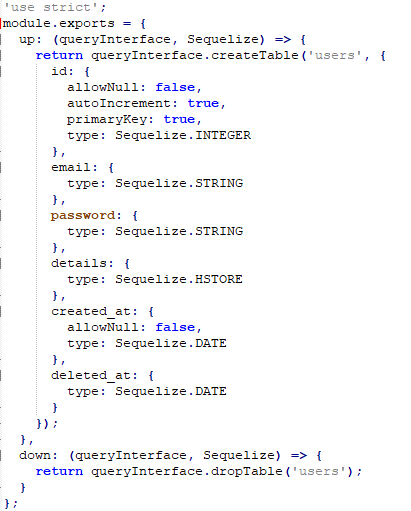


purchase\_items.js

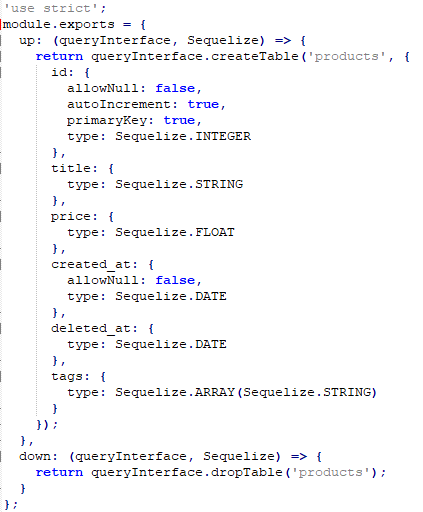


## Migrations

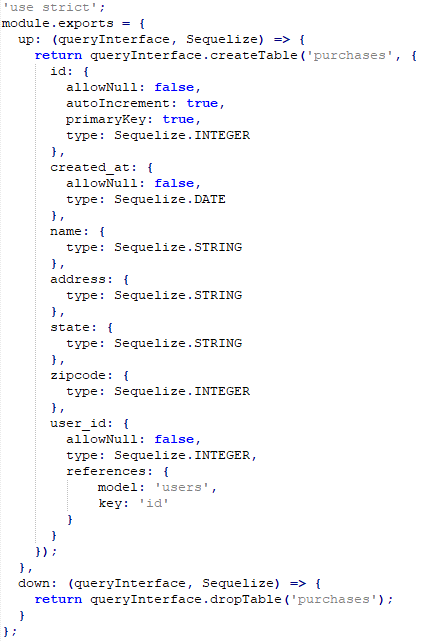
1MigrateUsers.js



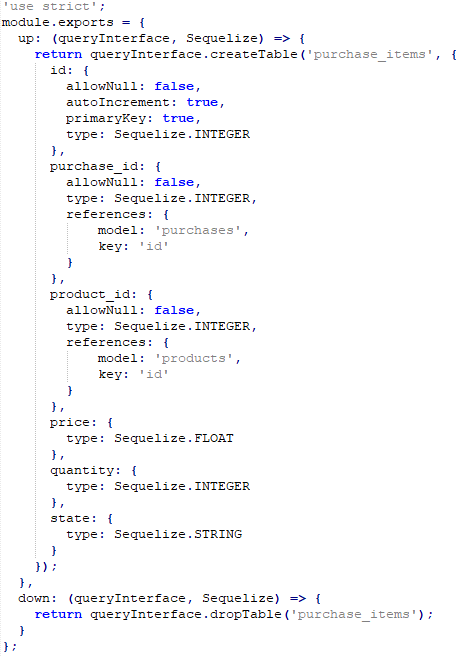
2MigrateProducts.js



3MigratePurchases.js



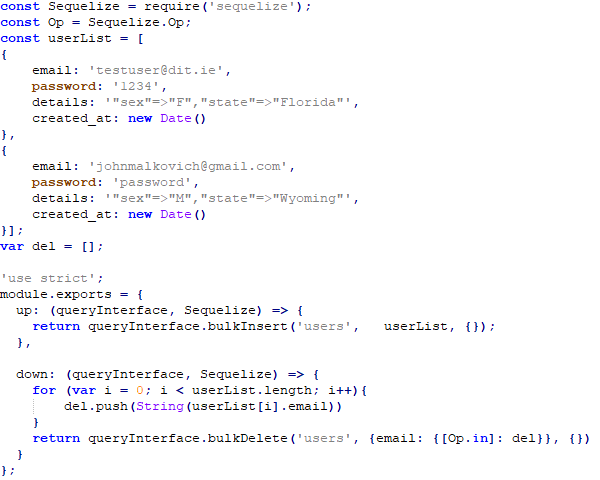
4MigratePurchaseItems.js



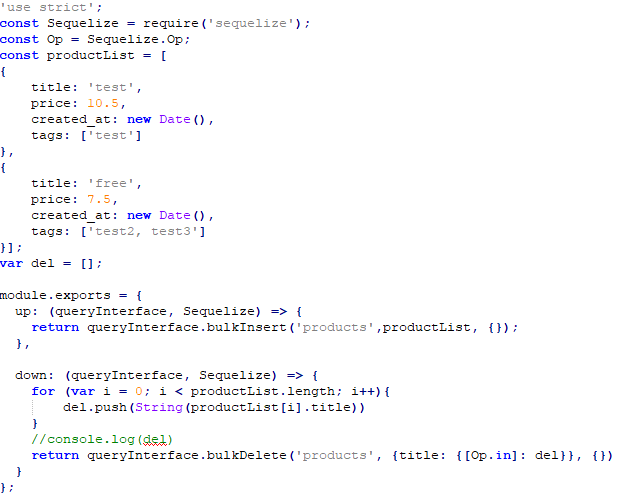
# Problem Set Five

The database was populated with test data using sequelize seeds. Hardcoded records were inserted using one seed file per table.

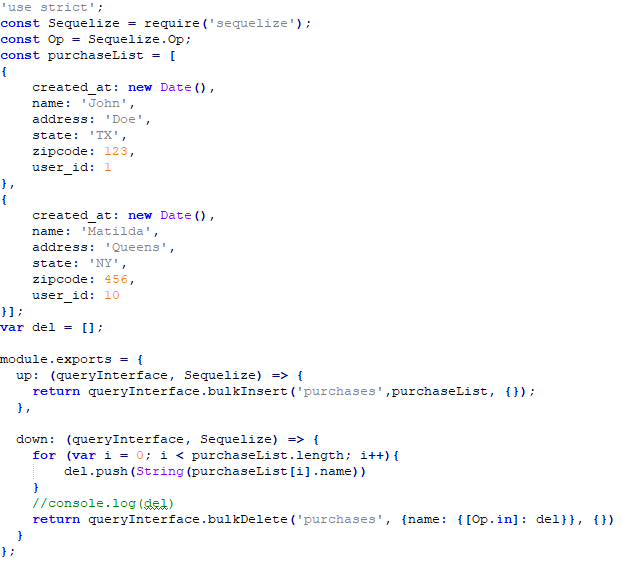
1updateusers.js



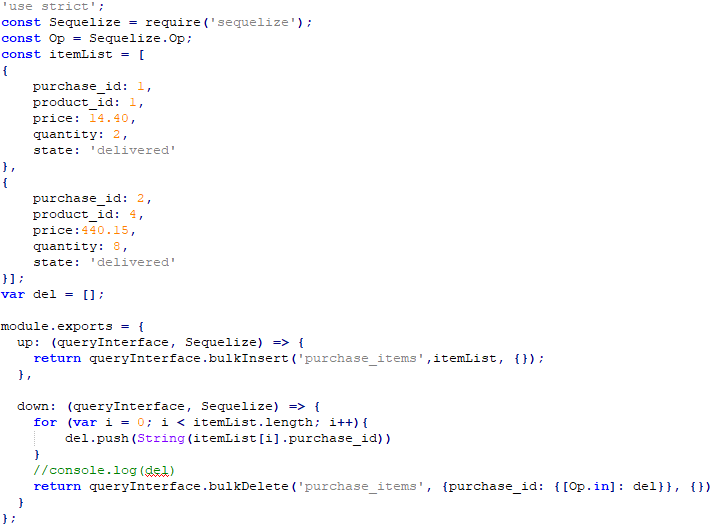
2updateproducts.js

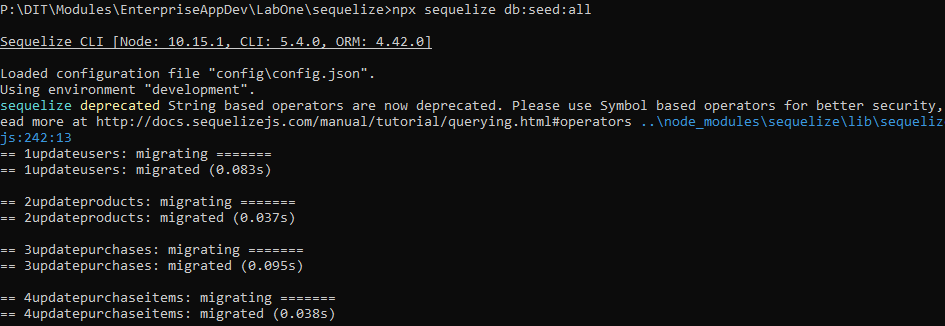


3updatepurchases.js



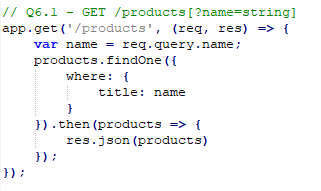
4updatepurchaseitems.js

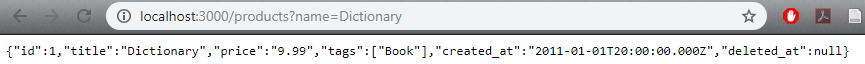




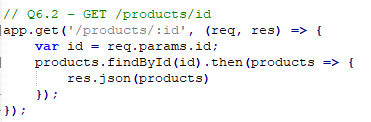
# Problem Set Six

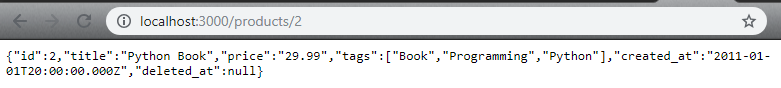
Part One – GET /products[?name=*string*]



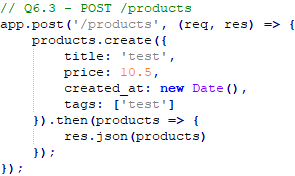


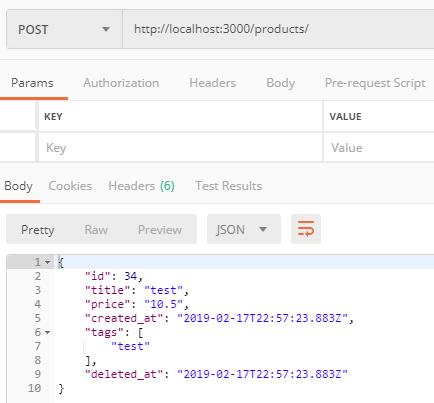
Part Two – GET /products/:id





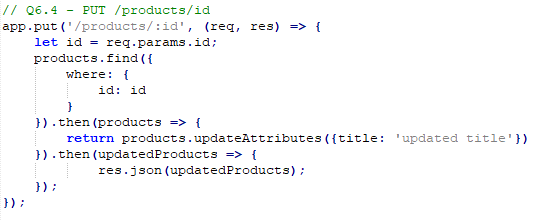
Part Three – POST /products

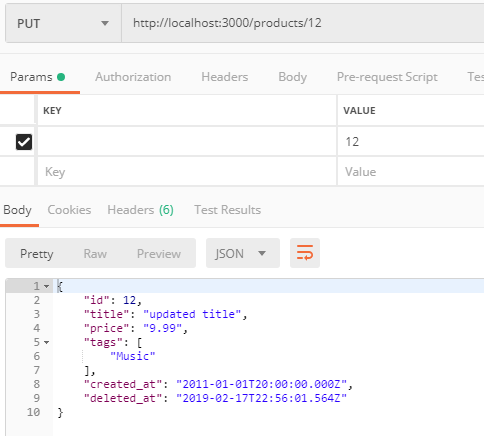






Part Four – PUT /products/:id







Part Five – DELETE /products/:id

