**Shipt Coding Exercise:**

Create a very basic API application, where a customer can have an order that is made up of products.

Rest API’s:

app.get('/', logIn.goToLogInPage);  
app.get('/logIn', logIn.goToLogInPage);  
app.get('/signUp', signUp.goToSignUpPage);  
app.post('/signUp', signUp.afterSignUpPage);  
app.post('/logIn', logIn.afterLogInPage);  
app.get('/logOut', home.goToLogoutPage);  
app.get('/home', home.goToHomePage);  
app.post('/placeOrder', home.processOrder);  
app.get('/users', user.list);  
app.get('/products', product.showAllProducts);  
app.get('/categories/:id/products', category.getCategoryWiseProducts);  
app.get('/categories', category.showAllCategories);  
app.get('/orders', order.showAllOrders);  
app.get('/query', queryResult.showQueryResult);

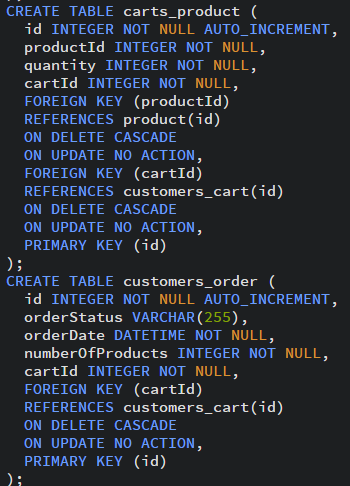
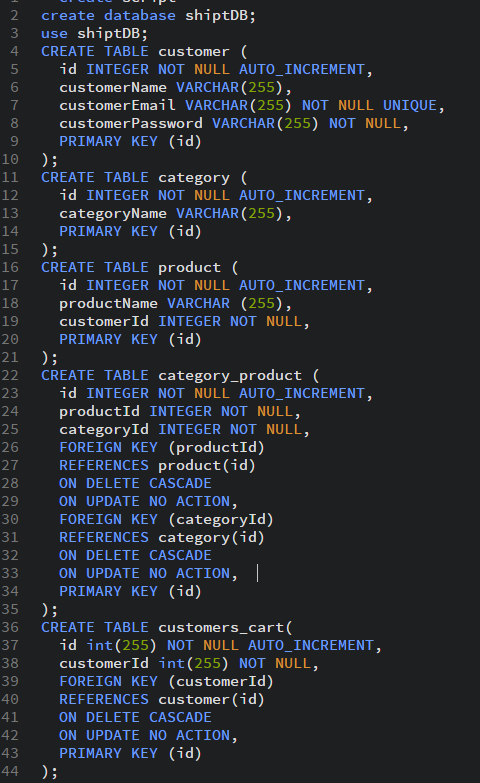
I have implemented the whole application using NodeJS, AngularJS, ExpressJS, EJS, HTML, CSS, JavaScript, JQuery, Ajax and MySQL database is used. Additional functionalities user can login and signup and the credentials are encrypted first then stored in the database and I have implemented local storage whenever the user logout and login again he/she can see the items in the cart as the user’s data are stored locally within user’s browser.

1. A product belongs to many categories. A category has many products.

2. A customer can have many orders. An order is comprised of many products. An order has a status stating if the

order is waiting for delivery, on its way, or delivered.

Database:



To increase the performance and interact less with database, I am not storing the cart details to the database until and unless the order has been placed.

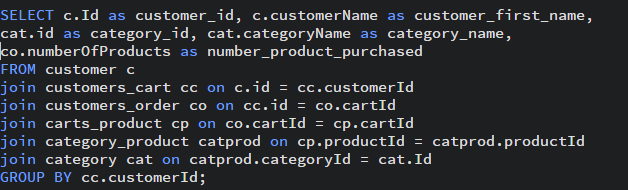
3. Write a SQL query to return the results as display below:

\*\*\*Example\*\*\*

customer\_id | customer\_first\_name | category\_id | category\_name | number\_purchased

--- | --- | --- | --- | --- | ---

1 |John | 1 | Bouquets | 15



4. Include the previous result as part of a function in the application. If you are using an ORM, please write the query in your ORM's DSL. Leave the original SQL in a separate file.

URL- /queryList ---see showQueryResult function in queryList.js

5. An API end point that returns the orders for a customer.

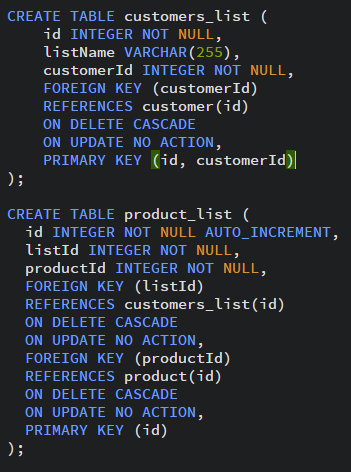
URL- /orders ----see showAllOrders function in order.js

**# Additional questions**

\*No coding necessary, explain the concept or sketch your thoughts. \*

- We want to give customers the ability to create lists of products for one-click ordering of bulk items. How would you design the tables, what are the pros and cons of your approach?

I would create two more tables to achieve this requirement one for associating list with customers and other for items in that list.



**Prons:** Easy to implement.

**Cons:** Extra storage.

- If Shipt knew exact inventory of stores, and when facing a high traffic and limited supply of particular item, how do you distribute the inventory among customers checking out?

Obvious way is to increase the price of that item in that case, the user who really wants to buy and need it will keep it in their cart. This will in turn help company to be profitable.

Other strategies are

While adding an item to cart a simple check shall be performed to see if the items are available or not. However, the count of items shall not be updated unless the customer eventually places the order. That way, we avoid the contention of resources from customers who simply book items in their cart and leave it idle.

Update the count when a customer adds an item to the cart but define a grace period within which, the item, if not ordered shall automatically be added back to the inventory (just updating the available count) while still being available in the customer’s cart. Finally, on placing the order later a check shall be made if the item was available or not and the count shall be updated in the inventory accordingly.