1. **Make your loaded schema exactly match the ERD below**

**Hint: You will need to add several more foreign-key constraints which are missing from the dellstore2.sql file**

***All forging keys are created to match keys in the ERD in the lab sheet:***

ALTER TABLE ONLY postgraphile.reorder ADD CONSTRAINT fk\_reordet\_prodid FOREIGN KEY (prod\_id) REFERENCES postgraphile.products(prod\_id) ON DELETE CASCADE;

ALTER TABLE ONLY postgraphile.inventory ADD CONSTRAINT fk\_inventory\_prodid FOREIGN KEY (prod\_id) REFERENCES postgraphile.products(prod\_id) ON DELETE CASCADE;

ALTER TABLE ONLY postgraphile.orderlines ADD CONSTRAINT fk\_orderline\_orderlineid FOREIGN KEY (prod\_id) REFERENCES postgraphile.products(prod\_id) ON DELETE CASCADE;

ALTER TABLE ONLY postgraphile.orderlines ADD CONSTRAINT fk\_orderline\_orderid FOREIGN KEY (orderid) REFERENCES postgraphile.orders(orderid) ON DELETE CASCADE;

ALTER TABLE ONLY postgraphile.cust\_hist ADD CONSTRAINT fk\_custhist\_prodid FOREIGN KEY (prod\_id) REFERENCES postgraphile.products(prod\_id) ON DELETE CASCADE;

ALTER TABLE ONLY postgraphile.cust\_hist ADD CONSTRAINT fk\_custhist\_orderid FOREIGN KEY (orderid) REFERENCES postgraphile.orders(orderid) ON DELETE CASCADE;

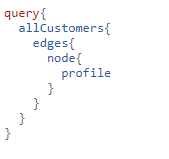
ALTER TABLE ONLY postgraphile.products ADD CONSTRAINT fk\_products\_category FOREIGN KEY (category) REFERENCES postgraphile.categories (category) ON DELETE CASCADE;

1. **Build a GraphQL query which returns the attributes from a single database relation. Have your query include one computed-field (which you can implement and a user-defined function in Postgres)**

**--Postgres function:**

create function customers\_profile(customers customers) returns text as $$ select customers.firstname || ',' || customers.lastname || ',' || customers.age || ',' || customers.gender $$ language sql stable;

**Running the query:**

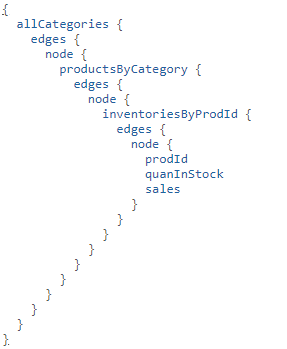


**Results:**



1. **Build a GraphQL query which returns the attributes from 3 joined database relations having 2 levels of nesting in the resultant output.**

**Running the query:**



**Results of Query:**



**Describe an application of the query you have chosen to write:**

This query could be used to get all the inventory, products and category’s very quickly. This database tables are all connected and related to each other thus we can see the transition through each table in an order. The customer would be looking in a particular category then this will bring them to the products of that category so the search will be refined. We will then be displayed with the product id, quantity in stock and sales of this product. This is how most clothes, Amazon and other website of that nature work.

1. **Create a mutation to add a new order to the database. Your mutation should update the orders, orderlines and cust\_hist relations**

**Running the query:**  



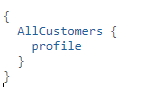

**Results of the query:**



1. **Implement the query from problem 2 (above) directly using GraphQL and Express**

**Express query:**  
const query = "SELECT firstname || ',' || lastname || ',' || age || ',' || gender AS profile from postgraphile.customers;";

**Graphql query:**



**Results of quey:**

