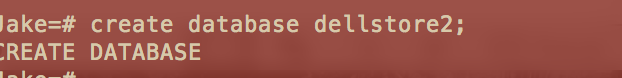
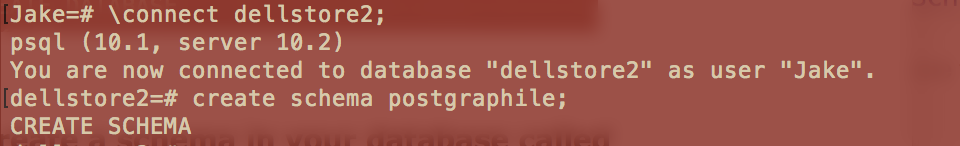
**Set Up:**

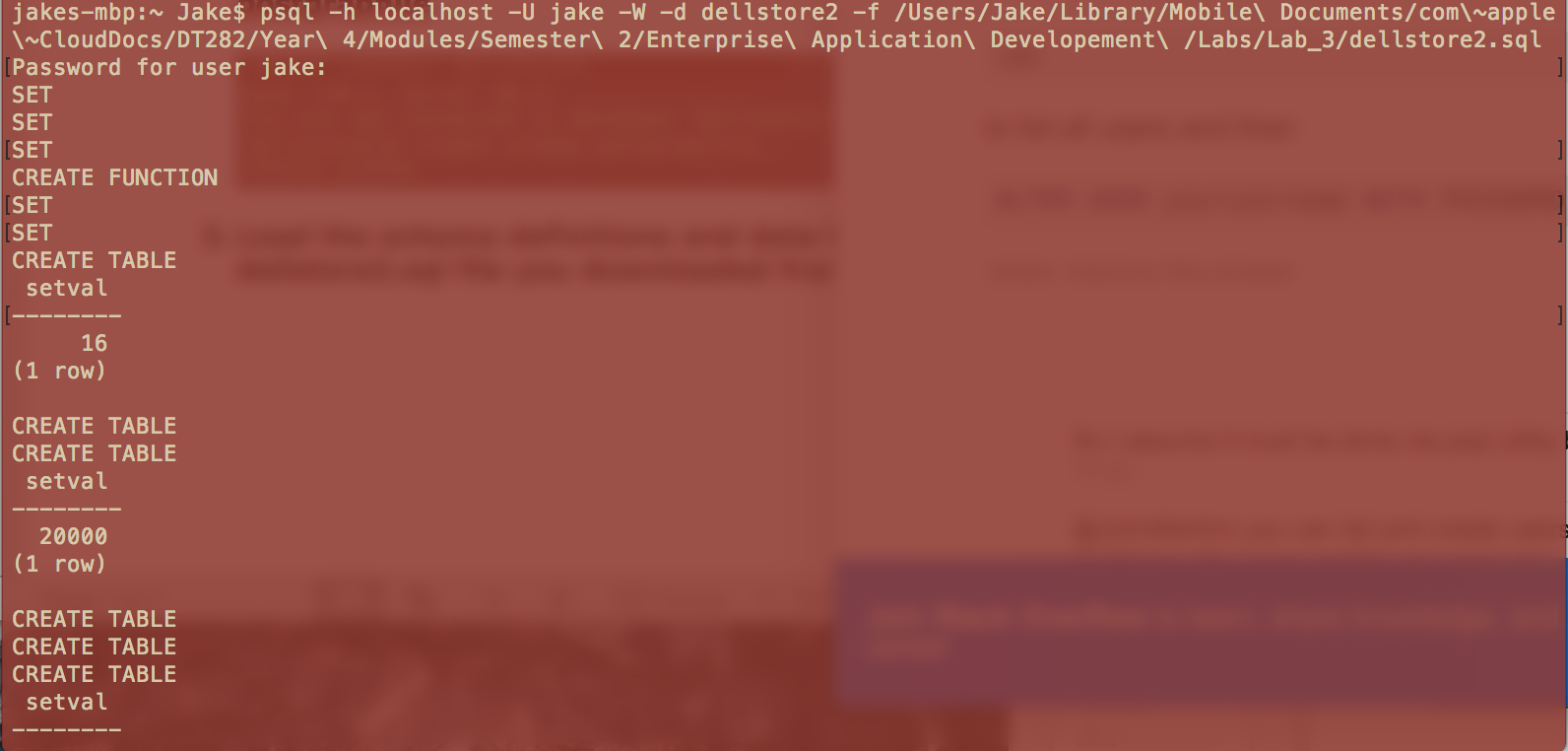
1. **Create a database called dellstore2**



1. **Create a schema in your database called postgraphile**



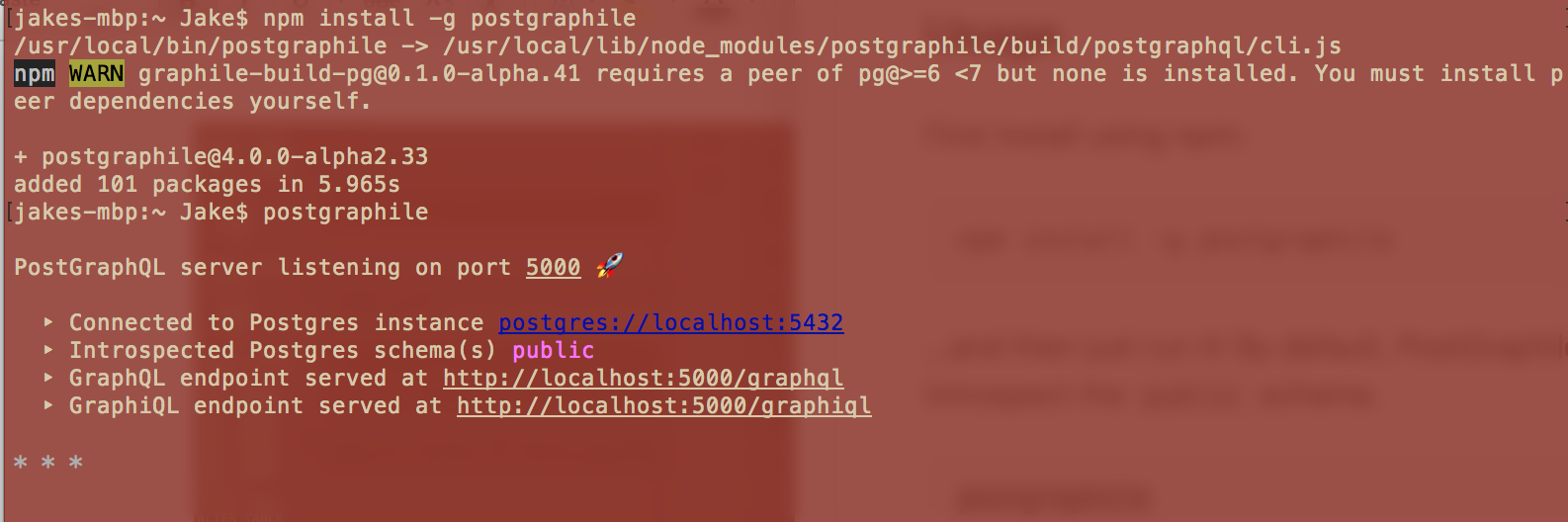
1. **Load the schema definitions and data from the dellstore2.sql file you downloaded from Webcourses**





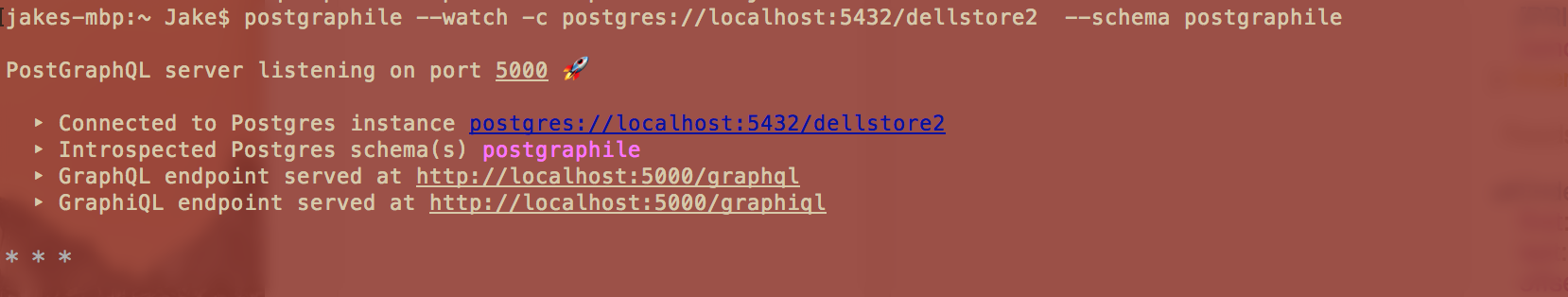
1. **Install postgraphile as per the instructions here:**

**https://github.com/graphile/postgraphile**

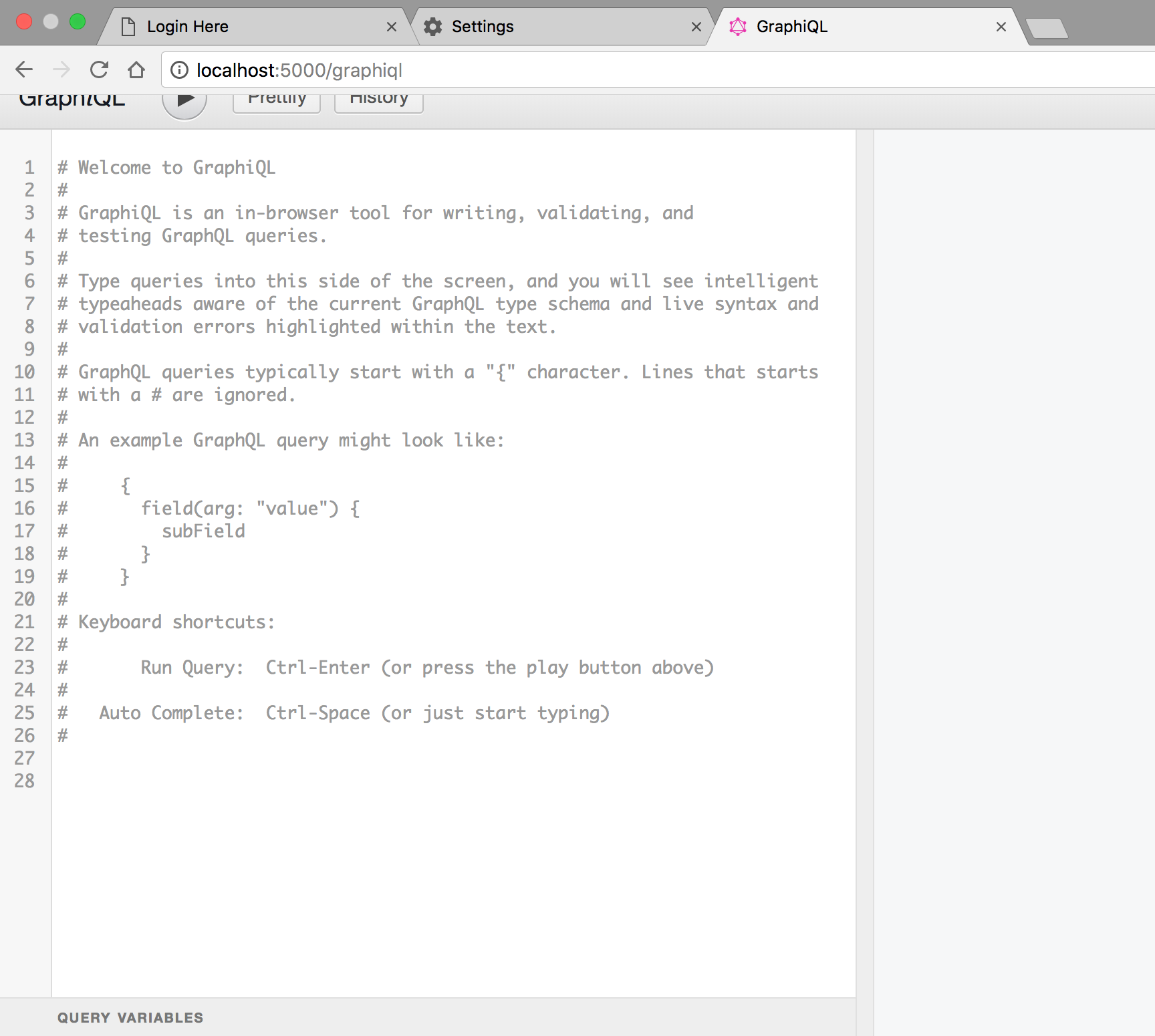


1. **Start the postgraphile server in watch-mode on your newly-created database/schema from the earlier steps**

postgraphile --watch -c postgres://localhost:5432/dellstore2 --schema postgraphile

****

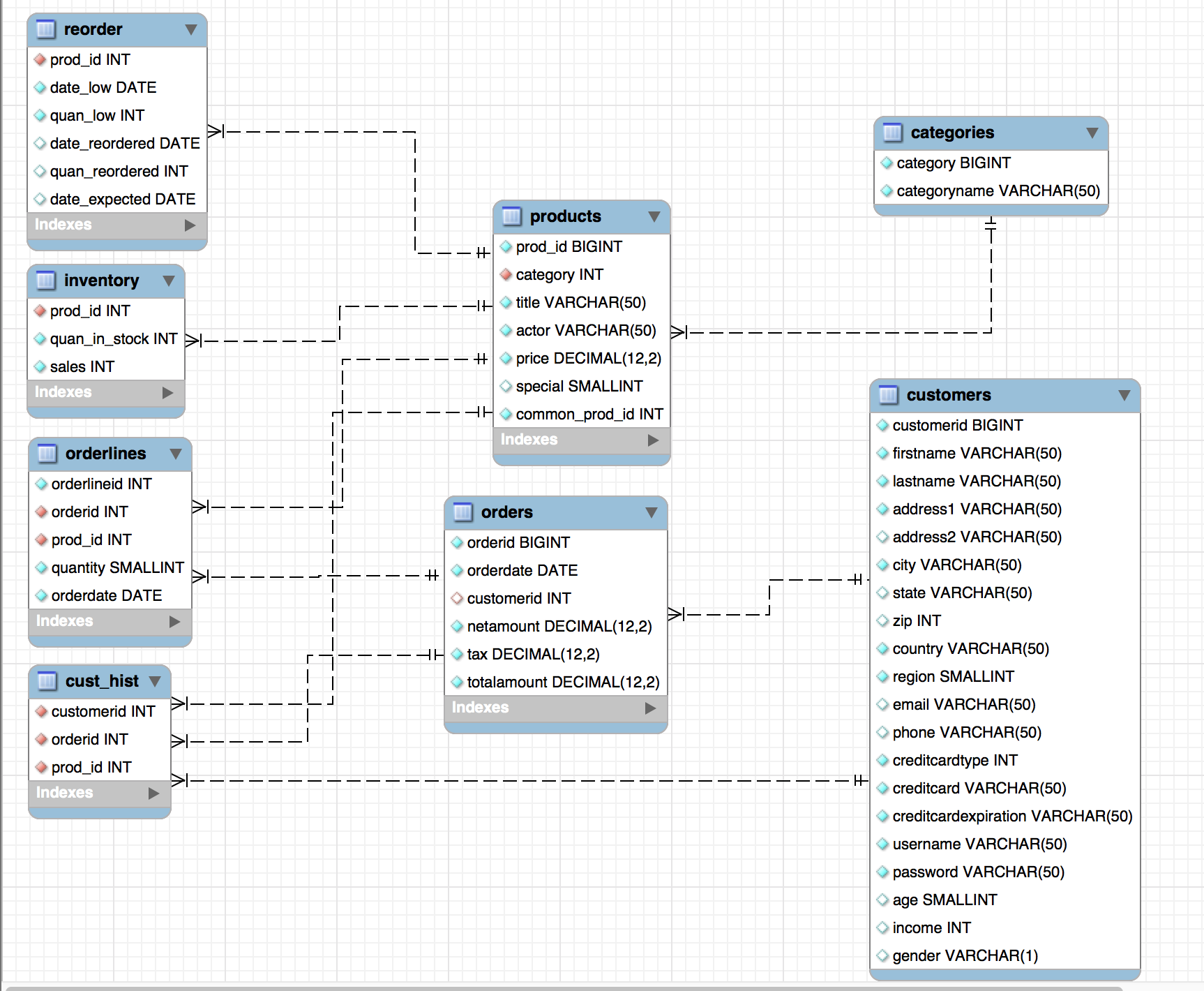
1. **Connect over a browser to** [**http://localhost:5000/graphiql**](http://localhost:5000/graphiql)



**Problem Set**

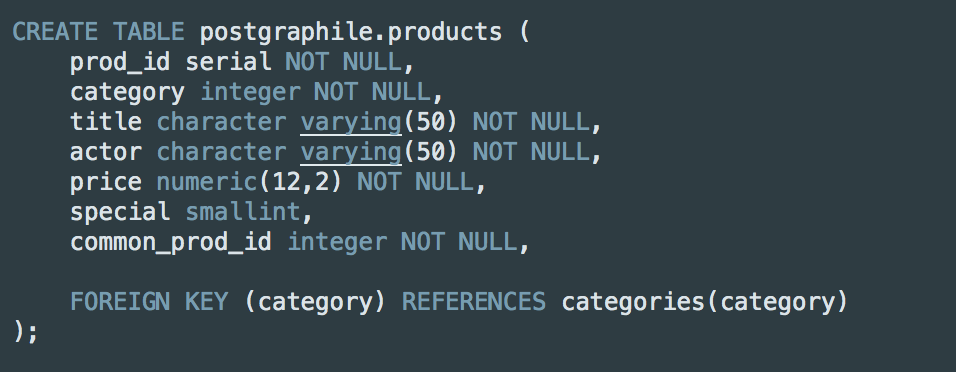
1. **Make your loaded schema exactly match the ERD below**

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



**Foreign Keys:**

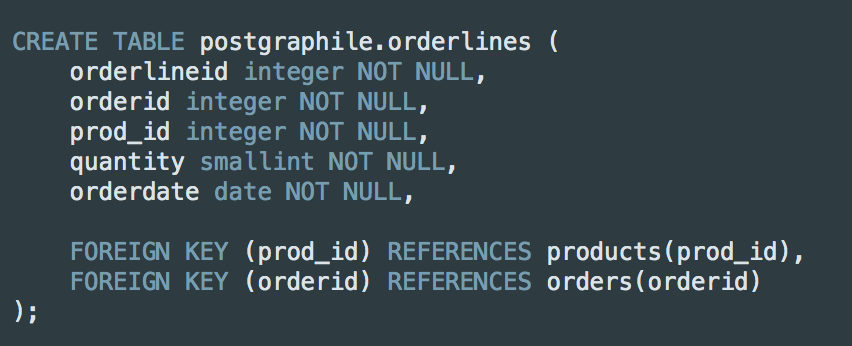
* **Products**



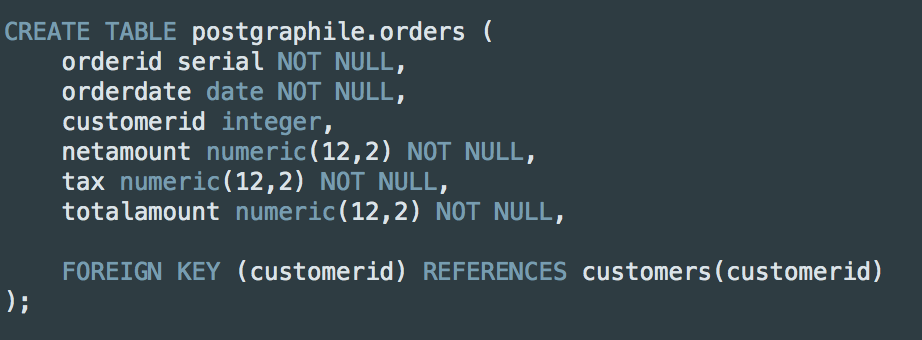
* **Inventory**



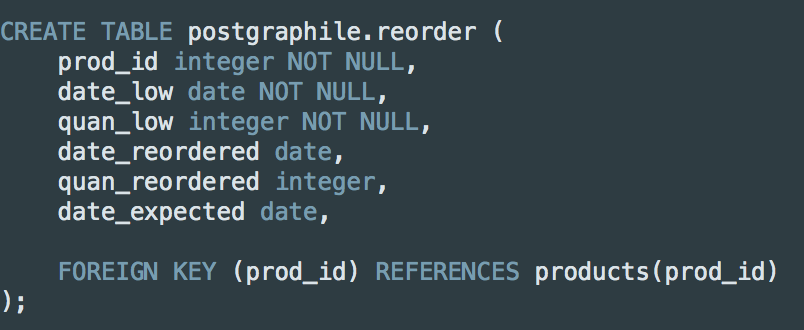
* **Orderlines**



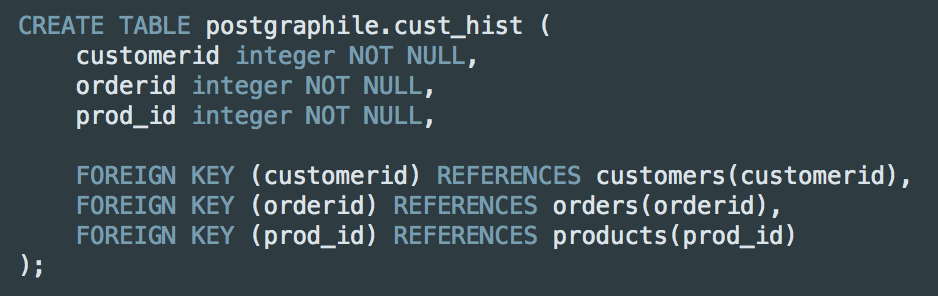
* **Orders**



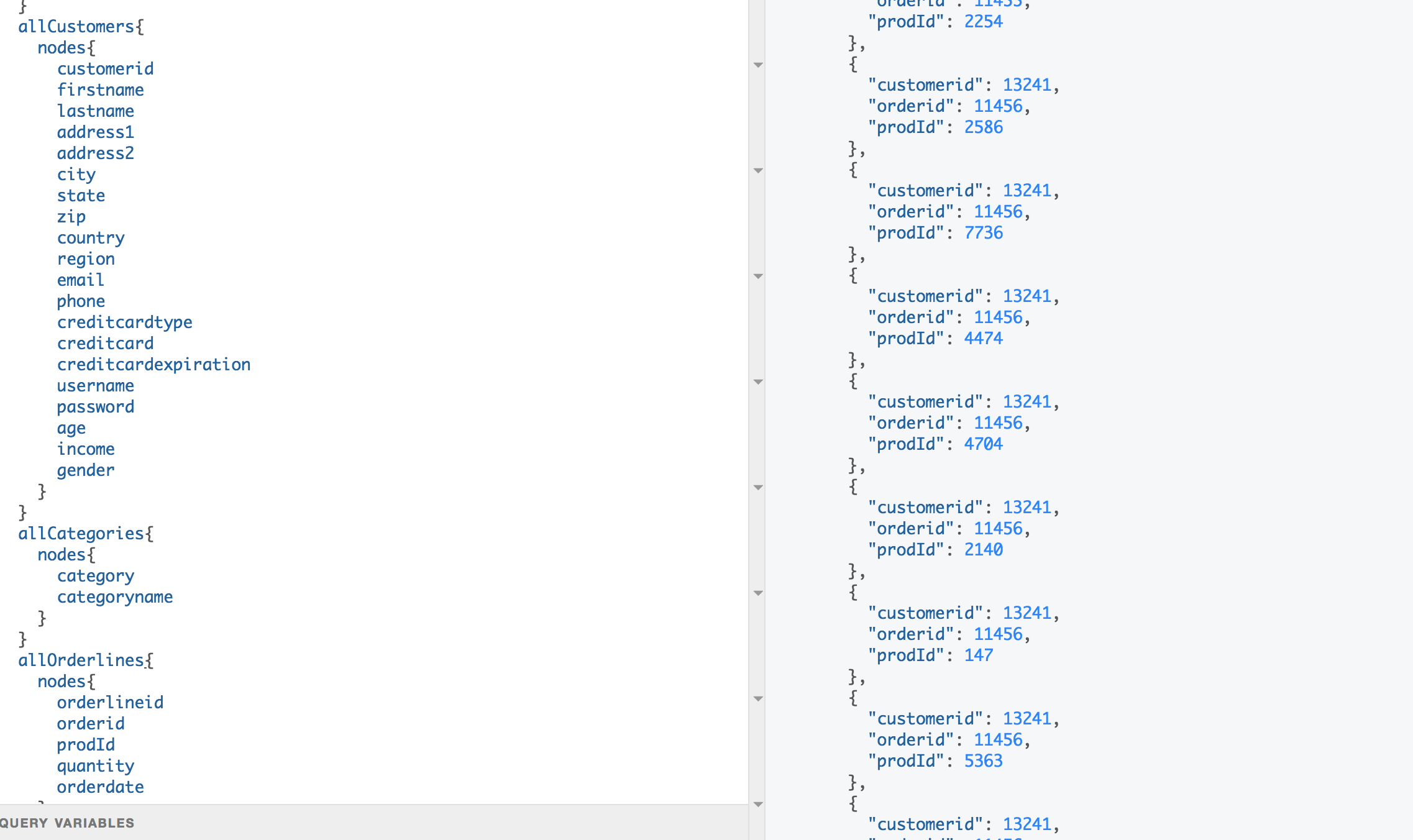
* **Reorder**



* **Cust\_Hist**

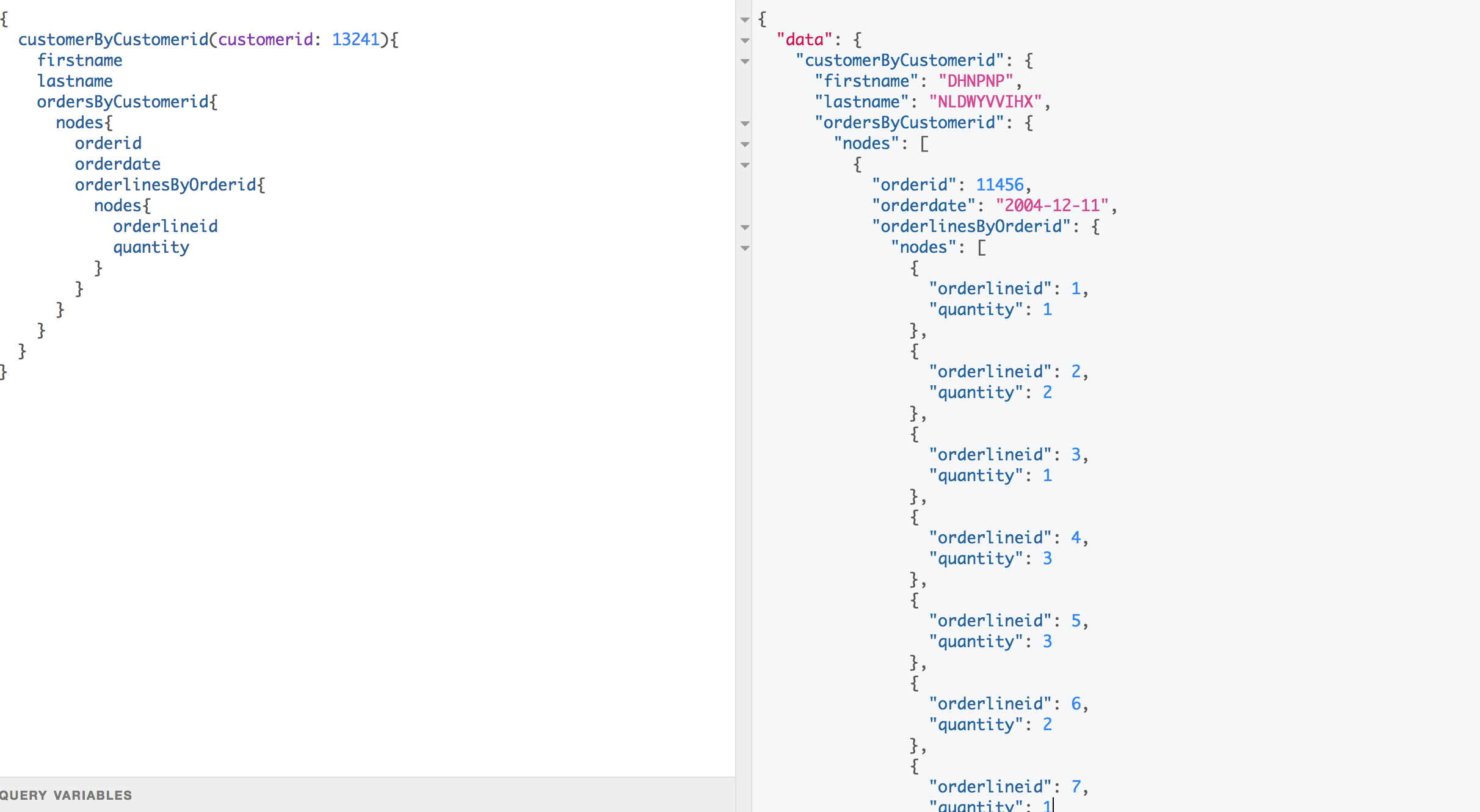


1. **Build a GraphQL query which returns the attributes from a single database relation. Have your query include one computed-field (which you can implement and a user-defined function in Postgres)**



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

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



This query is a nested query which will display data surrounding the customer who has an ID of 13241, it will display the first and last name of the customer with the specified ID, then the order table is joined using the customer id, this table will display the orderID and orderdate. Finally, the third joined table is the orderline table which is joined using the orderID and displays the orderlineID and quantity.

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

**` **

1. **In the previous problems you used postgraphql which conveniently introspects the database schema and dynamically builds the query and mutation objects and exports these over the graphiql API**

**In this problem, you should manually implement the query from problem 2 (above) directly using GraphQL and Express (or another server of your choice), i.e. not using postgraphql. For this you will probably need MassiveJS, Sequelize or similar to query Postgres as part of your resolver function**

