Project: Retailer
Course: Databases

Enterprise description:
The enterprise is a retailer, such as a department store, discount store, supermarket, convenience store, etc.; each of you will choose a specific retailer (either use a real one as your model or a made-up one). To keep the project within bounds, we'll ignore issues of employees, corporate finance, etc., and focus on the retail sales activities. Your retailer sells a large variety of products at multiple stores. Not all products are at all stores. Pricing may be different at different stores. Each store has its own inventory of products and needs to decide when to reorder and in what quantity. Customers may identify themselves by joining your frequent-shopper program. Others may remain anonymous. Your retailer has a Web site that accepts orders. From a database perspective it is just a special store that has no physical location and has no anonymous customers. The database tracks inventory at each store, customer purchases (by market basket and by customer, where possible), sales history by store, etc. Various user interfaces and applications access the database to record sales, initiate reorders, process new orders that arrive, etc.

- **The enterprise:** You may pick the enterprise that you will model. I'd like to see a wide variety chosen, so here is a list to serve as starting point to give you ideas, but other choices are welcome, indeed encouraged: Walmart, Target, J.C. Penny, Sears, Costco, BJ's, Best Buy, American Eagle, Nordstrom, Safeway, Aldi, Albertsons, Acme, HEB, Food Lion, Piggly Wiggly, Wegmans, Walgreens, Rite Aid, CVS, Longs, Superfresh, Carrefour, Tengelmann, Hankyu, Dillards, Wawa, Sheetz, Modells, Petsmart, Radio 2 Shack. I've included some non-US enterprises on this list and encourage you to consider them, or other non-US retailers.

- **products:** Products come in a variety of sizes or means of packaging. Each product has its own UPC code (the bar code that is scanned at the checkout).

- **brands:** A variety of products may be sold under the same brand (e.g. Pepsi and diet Pepsi). For such applications as reorder, specific products and sizes matter. For other applications, data may be aggregated by brand.

- **product types:** A particular type of product may be sold in a variety of sizes and a variety of brands. For example, cola is sold under such brands as Pepsi and Coke. Product types form an specialization/generailzation hierarchy. For example cola is a type of soda, which is a type of beverage, which is a type of food. Some products fit into multiple categories. For example, baking soda is a cleaner, a food (since it is used for baking), and a drug (since it may be used an an antacid), but it is not a type of soda.

- **vendors:** Products are sold to stores by vendors. A vendor may sell many brands (e.g. Pepsico sells Pepsi, Tropicana, Aquafina, Gatorade, Lay's, Doritos, Quaker, and others).

- **stores:** Stores sell certain products, each of which has a certain inventory amount at any point in time. Stores have locations (addresses), hours at which they are open, etc.

- **customers:** Customers who join a frequent-shopper program provide some personal information based on what the enterprise requests. They may refuse to provide some information. Customers come into a store (or go online) to buy a market basket of goods (so you have to store all orders and order items). Not only must this data be stored, but also the system must be able to handle multiple customers buying goods at the same time.

Data Generation:
For simplicity, I will not require perfectly realistic data so that you don't have to ensure that, for example, if you are modeling an HEB Supermarket, it really does sell Shiner Bock beer. However, you should strive for a good degree of realism in your data (and, yes, by the way, except in dry counties, HEB sells Shiner Bock). Where appropriate, randomly generated data are acceptable (and a good way to avoid having lots of data entry to do).

Tasks:

1. E-R Model
   - Construct an E-R diagram representing the conceptual design of the database.
   - Be sure to identify primary keys, relationship cardinalities, etc.

2. Relational Model
   - After creating an initial relational design from your E-R design, refine it based on the principles of relational design (Chapter 8).
   - Create the relations in PostgreSQL database.
   - Create indices and constraints as appropriate.
   - If as you refine your design, you discover flaws in the E-R design, go back and change it (even if the earlier design passed the checkpoint.) Your final E-R design must be consistent with your relational design.
   - Your relational design have to satisfy conditions of all normal forms (Chapter 7)

3. Populate Relations
   - Include enough data to make answers to your queries interesting and nontrivial for test purposes.
   - You may find it helpful to write a program to generate test data.

4. Queries: You should run a number of test queries to see that you have loaded your database in the way you intended. The queries listed below are those that your clients (managers from the retail enterprise) may find of interest. They may provide further hints about database design, so think about them at the outset of your work on this project.
   - What are the 20 top-selling products at each store?
   - What are the 20 top-selling products in each state?
   - What are the 5 stores with the most sales so far this year?
   - In how many stores does Coke outsell Pepsi? (Or, a similar query for enterprises that don't sell soda.)
   - What are the top 3 types of product that customers buy in addition to milk? (Or similar question for nonfood enterprises.)

5. Everything should be in a single folder and uploaded to git repository.

6. Add a README file in the top-level folder that includes short description of the project and explains what is where, etc.