Kitchen Inventory Manager

Amanda Killeen, CSPB 3287 Spring 2021

Project Goal

Taking the concept of reverse grocery list, where you have a list of things you normally buy and figure out what you are out of, I will build a dashboard showing which items I have on hand in my kitchen and where inventory is running low, and map which stores I need to visit, so that I stay within budget, don't over purchase, and can plan meals accordingly.

This inventory will track attributes such as where the food is stored (e.g. Pantry, Fridge, Freezer), category, brand, lastPurchasedDate, quantityOnHand, quantityNeeded and more to be determined that will be stored in relations such as Category, Store, Storage,Item and Recipe. In addition to tracking the items, it will flag when inventory is low and the item needs to be purchased.

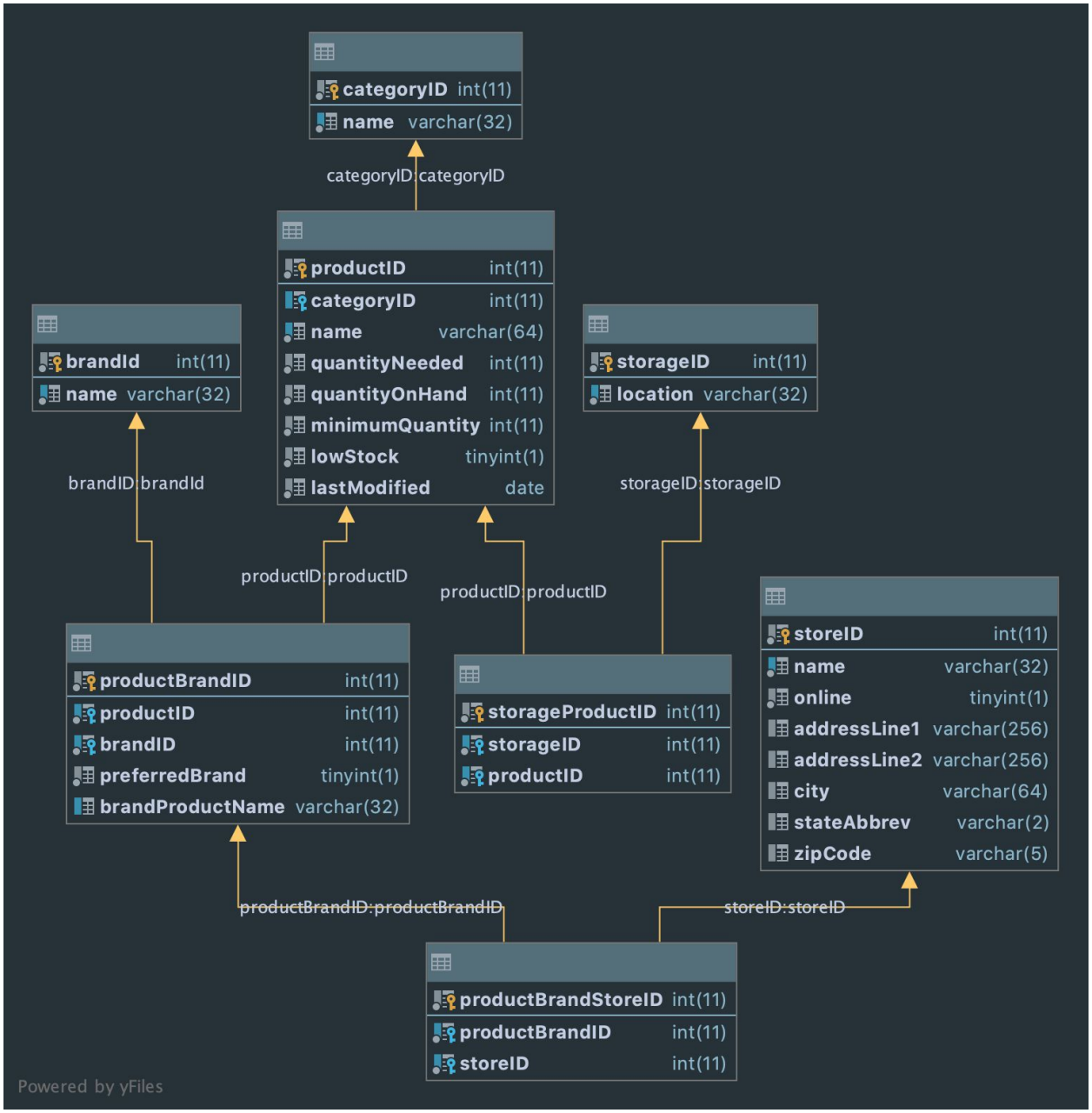
Finally, this project is an opportunity to apply database and SQL skills to parse a dataset into multiple relations efficiently.

Tools

- DataGrip - Database & SQL IDE. This will be the main tool I use to create tables and write queries.
- Tableau - Data visualization tool for creating visualizations related to the database, including the mapping of stores and item inventory status.
- CSV file Inventory - A single table will all inventory data data to be parsed into multiple relations using SQL.
- MySQL - SQL dialect to be used in creation and management of database
- Google Cloud Platform - Database hosting platform
- JupyterLab via Anaconda - Final report write-up that will include table creation statements, aggregations, queries and tests

Schema Design

The final database will be made-up of multiple relations that can be joined together to re-create the original inventory CSV.



Storage Table

Dimension	Datatype	Description
storageID	Integer	Primary Key
location	VarChar(32)	Where items are stored; example values pantry, refrigerator, freezer etc.

Category Table

Dimension	Datatype	Description
categoryID	Integer	Primary Key
name	VarChar(32)	Name of category that a product will fall into; example values baking, produce, grains.

Product Table

Dimension	Datatype	Description
productID	Integer	Primary Key
name	VarChar(64)	Name of product; example values flour, sugar, apples, ground beef.
quantityNeeded	Integer	The preferred amount of the product to have in stock
quantityOnHand	Integer	The actual amount of product in stock
minimumQuantity	Integer	Threshold to flag low inventory
lowStock	Bool	Flag to indicate if stock is low; updated via trigger
categoryID	Integer	Foreign Key (Category)
lastModified	Date	Date the product was last modified, applies to quantityOnHand; updated via trigger

StorageProduct Table

Dimension	Datatype	Description
storageProductID	Integer	Primary Key
storageID	Integer	Foreign Key (Storage)
productID	Integer	Foreign Key (Product)

Brand Table

Dimension	Datatype	Description
brandID	Integer	Primary Key

name	VarChar(32)	Name of product brand; example values Simple Truth, Fred Meyer, Heinz, Kraft.

Store Table

Dimension	**Datatype**	**Description**
storeID	Integer	Primary Key
name	VarChar(32)	Name of store; example values Trader Joe's, Thrive Market, Fred Meyer
online	Boolean	Indicates if an online store vs brick & mortar
addressLine1	VarChar(256)	Address line 1 of store if a brick & mortar location
addressLine2	VarChar(256)	Address line 1 of store if a brick & mortar location
city	VarChar(64)	City associated with brick & mortar location
stateAbbrev	VarChar(2)	Two character state abbreviation
zipCode	VarChar(5)	Five character zip code

ProductBrand Table

Dimension	**Datatype**	**Description**
productBrandID	Integer	Primary Key
productID	Integer	Foreign Key (Product)
brandID	Integer	Foreign Key (Brand)
preferredBrand	Boolean	For items available from more than one brand, boolean indicates preference.
brandProductName	VarChar(32)	Trademarked name of product; example values Rice Krispies, Cheez-It

ProductBrandStore

Dimension	**Datatype**	**Description**

productBrandStoreID	Integer	Primary Key
productBrandID	Integer	Foreign Key (ProductBrand)
storeID	Integer	Foreign Key (Store)

Queries & Aggregations

- For products that are available from multiple brands (ex. Mustard, brands available: French's or generic, where generic is preferred brand), implement GROUP_CONCAT to combine available brands into a single column, and show preferred brand and store for the item.
- Identify Stores associated with the most Products flagged as lowStock to indicate which shopping trips are necessary.
- Identify how stocked the kitchen and storage locations are, using percentages. Show which locations are well stocked vs low stocked and how stocked overall the kitchen is.

Testing

CTEs & Joins

Testing will be based on a series of CTEs (Common Table Expressions) and joins. CTEs are similar to nested queries, but in my opinion, easier to read/follow if you are not the code author and enable joining groups of tables together into a temporary table that can then be joined together in the main query, supporting testing of units of code to confirm joins works as designed before joining to another set of tables. Since the goal of the project is to parse a single relation into multiple relations, joins will be used to recreate the single relation and compare to ensure it is lossless. This will be the primary test.

Triggers

At this time there are currently two triggers on the database, to change the lowStock flag in the Product table based on quantityOnHand compared to the minimumQuantity value and update the modifiedDate when the quantityOnHand is updated. This will be tested to show what happens when quantityOnHand fluctuates, through updates.

Constraints

Constraints (primary, foreign and unique keys) are used throughout the relations and will be tested to ensure cascading works correctly.

Visualization

Tableau will be utilized to create a dashboard to visualize the database with components including mapping of store locations, number of items to purchase at the store-location, and a visual of products that are indicated as lowStock.