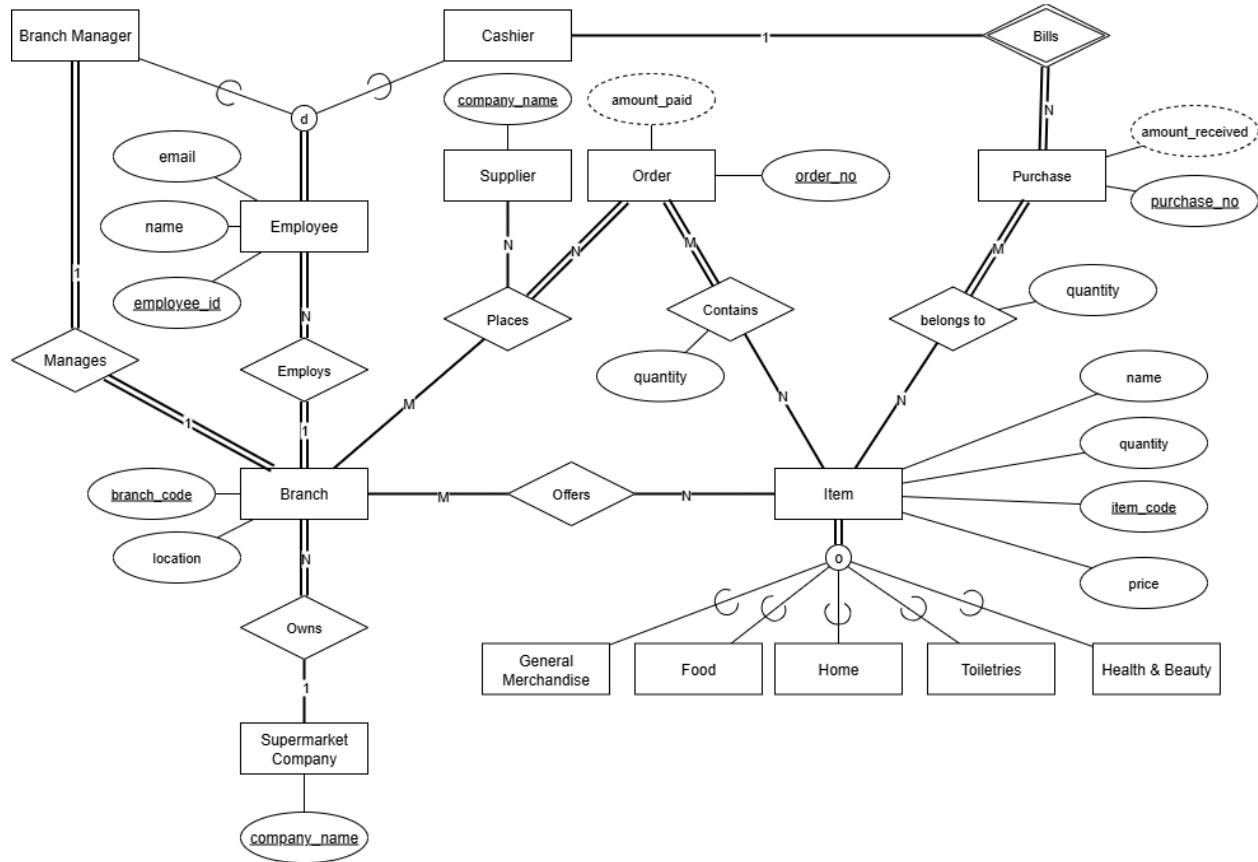


Supermarket Inventory Management System (SIMS) Proposal CPSC 471 - Team T02-2

Group Members: Adhira John (10193007) | Leonie Mertens (30298707) | Taha Ghumman (30176505)

Users: *Branch Manager* (access to all), *Cashier* (access to Items)

ER Diagram:



[Link to ER Diagram](#) (Higher Quality, easier to see)

Executive Summary

Introducing **SIMS** (Supermarket Inventory Management Systems), a one-stop inventory management system. From a way to organize by item type, but also to order new items off of when supply gets low. With the use of a web-application, supermarket chains can manage and organize their inventory effectively. Get alerts when stock is low, and find out who the designated supplier is for that item. Users are employees of the supermarket chain and can be listed with either Branch Manager (BM) permissions or Cashier permissions. BM's can order new stock once inventory runs out through the system directly with the

supplier. Cashier's when scanning an item, automatically have the stock of an item updated. If the quantity dips below a certain threshold, SIMS alerts the BM of low stockage so that they may request new stock immediately.

Diagram Details

We've designed an ER diagram that encompasses a generic supermarket store branch detailing the interactions that affect inventory of a branch on a day-to-day basis. Monetary totals, like the amount received in a transaction, are treated as derived attributes as the focus for our application is specifically for inventory and not accounting. The customer is represented by the Purchase entity, which is identified by its purchase number. The customer does not mean much to the system except for showcasing how many items are sold. This means that the system can sort by transaction number to find how many items were bought alongside the quantity of each item. We then can see what item sells the most so that we can potentially double the stock going into the next shipping haul.

System Specifications

Items can fit into overlapping specializations of five categories, General Merchandise, Home, Food, Toiletries and Health & Beauty. Employees relevant to SIMS fit disjointly into either being a Branch Manager or a Cashier. Only Items can be purchased, and only Cashiers can bill them. This makes it so the system only cares about the items being sold, not necessarily about staff, staff amount, or anything relating to the other aspects of the supermarket. Of course, the system can be broadened depending on the supermarket, so that the Items category has other sub-categories underneath it, but for the purpose of the proposal, our diagram features what is considered the most generic categories found in most supermarkets. Each item is identified by its code and quantity, this allows the system to track when a supplier reload may be required depending on the unique id.

Employee information is also kept for verification purposes. If an item sells, the employee's id is saved alongside the item to know who executed the sale. If you wanted to track who was the hardest working employee, or see what employee executed the most sales, you can filter by the employee's and then sort and order by transactions made.

Reference Sources

Flask API: <https://flask.palletsprojects.com/en/stable/api/>

This API is great for seeing all methods within Flask alongside their signatures for their arguments. It's useful to see a brief description on usage as well as the example usages for some methods to effectively use Flask for certain use-cases.

Flask Web App Tutorial:

<https://www.geeksforgeeks.org/python/flask-creating-first-simple-application/>

This tutorial for building an application by geeksforgeeks is great to get an initial set up for a first application. It is more specific than general documentation or tutorials and provides a good framework for our web application.

Flask Tutorials: <https://realpython.com/tutorials/flask/>

This includes many different tutorials for building Flask applications from scratch, including instructions for using SQLAlchemy and integrating a database into Flask. It is particularly useful for our project because it explains step-by-step how to connect Flask with a relational database, which is essential for building our inventory management system.