Clothing Store Database - Group 6

Minh An Cao, Alexander Nguyen, Ricky Singh

1. Description of application:

   A database that allows user to customize and buy outfits from different stores and brands by choosing different parts of the outfit like shirts and pants from different brands, and allows users to sort clothing to add to an online cart.

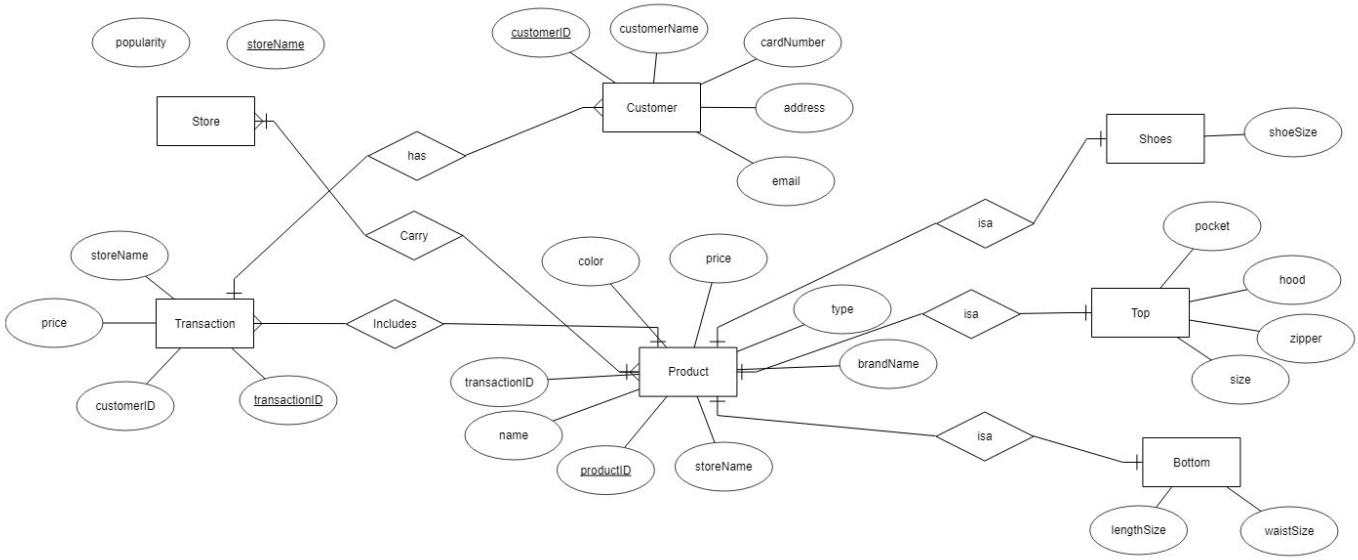
2. Data requirements:
   - **Stores -** Store will have store name and store popularity.
   - **Brands -** Brand will have brand name, brand popularity, types of clothing available, sizes available, material, colors, styles, and price.
   - **Customer Account Info** - Account info for shipping, such as card information, location, name, sizes
   - **Delivery Services -** Delivery services will have tracking information, company, ETA
   - **Transactions -** Transactions will keep track of price, card number, item(s), store, transaction history, ETA
   - **Discounts -** Discounts has data such as coupon codes, mark-offs, applicable brands
   - **Social User Page -** Holds data such as name, outfits created, favorite brands, ratings
   - **Outfit -** Holds outfitID, top, bottom, socks, shoes, accessory, jacket
3. Transactions/Functionalities:
   - List all the clothing a brand sells.
   - List all the brands that sells shirts.
   - Sort discounts (ascending or descending)
   - Sort social users by outfit ratings received
   - Present a report listing all the brands that sells jeans.
   - Display number of items left in stock for a specific clothing item.
   - Display sizes available for a specific clothing item (small, medium, etc.).
   - Search for a specific item
   - Display prices of all clothing sold by a brand.
   - Display number of stores that offer free shipping.
   - List all the brands that a store offers.
   - Sort items by rating
   - Sort items by price (ascending or descending)
   - Update inventory (add, remove, modify item)
   - Recommend size for a product based on transaction history from users of similar height and weight

Project Milestone 1

1) Textual description of the project (Milestone 0)

   Our application will be a database that will allow users to buy clothing from different stores and brands by choosing different parts of the outfit, such as shirts and pants for different brands.

2) E/R Diagram



3) Description of Entities

   Description of what each entity represents

Entity name	**Description**
Store	The store is the seller of products.
Customer	Customer entity will contain information needed for customer contact.
Transaction	Transaction has all information between the customer, store, and products bought.
Product	The product is any piece of clothing.

Top	The top is one type of clothing for Product.
Bottom	The bottom is one type of clothing for Product.
Shoe	The shoe is one type of clothing for Product.

4) Description of relationships on E/R Diagram

Name	Entity 1	Entity 2	Entity 1 -> Entity 2 Role Cardinality	Entity 2 -> Entity 1 Role Cardinality	Description
Carry	Store	Product	many-to-many	many-to-many	Many stores carry many products
Has	Transaction	Customer	many-to-one	one-to-many	Many transactions correspond to one customer.
Includes	Transaction	Product	one-to-many	many-to-one	One transaction can contain multiple products.
Top is	Product	Top	one-to-one	one-to-one	One product corresponds to one top.
Bottom is	Product	Bottom	one-to-one	one-to-one	One product corresponds to one bottom.
Shoe is	Product	Shoe	one-to-one	one-to-one	One product corresponds to one shoe.

5) Description of Entity attributes

Description of attributes in each entry

Name	Used By	Used By An Identifier	Data Type	Description
storeName	Store	No	VARCHAR	Holds the storeName
popularity	Store	storeName	FLOAT	Holds the popularity.
customerID	Customer	No	INT	Holds the customerID.
customerName	Customer	customerID	VARCHAR	Holds the name of the Customer.
address	Customer	customerID	VARCHAR	Holds the address of the Customer.
email	Customer	customerID	VARCHAR	Holds the email of the Customer.
cardNumber	Customer	customerID	VARCHAR	Holds the card number of the Customer.
price	Transaction	transactionID	FLOAT	Price of the transaction.
customerID	Transaction	transactionID	INT	customerID corresponding with the transaction.
transactionID	Transaction	No	INT	transactionID for the specific transaction.
storeName	Product	productID	VARCHAR	storeName that

				has the Product.
transactionID	Product	productID	INT	transactionID that has the Product.
productID	Product	No	INT	productID that correspond to a specific Product.
name	Product	productID	VARCHAR	Name of the Top.
brandName	Product	productID	VARCHAR	The brand name of the top.
color	Product	productID	VARCHAR	The color of the top.
price	Product	productID	FLOAT	The price of the top.
type	Product	productID	VARCHAR	Describes the type of the product i.e. “t-shirt” is a type of top.
size	Top		VARCHAR	The letter size of the top (small, medium, large, etc.).
zipper	Top		BOOLEAN	Says “true” if there is a zipper on the top, “false” if there is not.
hood	Top		BOOLEAN	Says “true” if there is a hood on the top,

				“false” if there is not.
pocket	Top		BOOLEAN	Says “true” if there is a pocket on the top, “false” if there is not.
lengthSize	Bottom		FLOAT	The length dimension for the size of the pant.
waistSize	Bottom		FLOAT	The width dimension for the size of the pant.
shoeSize	Shoe		FLOAT	The shoe size of the shoe as a number i.e. 10.

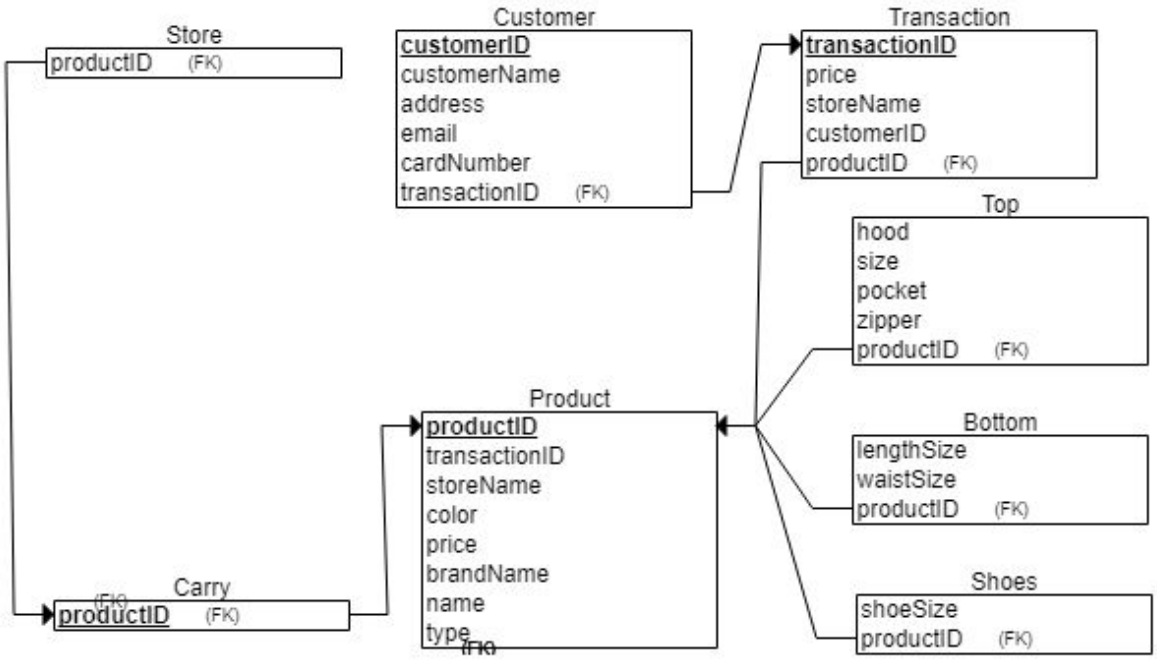
6) Analysis of functional and non-functional requirements

Some assumptions made are each product has a unique ID that tells the brand name, price, color, name and size. Each product will contain a specific top, bottom, and shoe. The user will be able to get the product(s) information and customer information when looking at a transaction.

For functional requirements, the application should allow customers to create an account with a name, address, email, and card number.
A user can purchase products, and create a transaction with a store. Products include tops, bottoms, and shoes, and stores carry these products.

For non-functional requirements, there are relationships created between each table to achieve scalability, flexibility, extensibility, efficiency of storage, and efficiency of processing. Scalability is achieved by just adding new stores and products to the database. Flexibility is achieved by the different attributes of each product. The application is extensible because new products can be added easily. It is efficient because just a product ID is needed for the qualities. Processing is easy because each transaction will be identified by a transaction ID.

7) Relational model (translating E/R into table model)



8) Normalization - 3NF

Check that each table in your design is normalized. If not perform normalization.

They are normalized.

9) DDL Script that creates a database

```
CREATE DATABASE ClothingDatabase;
USE ClothingDatabase;
```

```
CREATE TABLE Store (
        storeName VARCHAR(255),
        popularity FLOAT,
        PRIMARY KEY(storeName)
);

CREATE TABLE Customer (
        customerID INT,
        customerName VARCHAR(255),
        address VARCHAR(255),
        email VARCHAR(255),
        cardNumber VARCHAR(16),
        transactionID INT,
        PRIMARY KEY(customerID)
);

CREATE TABLE Transaction (
        transactionID INT,
        price FLOAT,
        storeName VARCHAR(255),
        customerID INT,
        productID INT,
        PRIMARY KEY(transactionID)
);

CREATE TABLE Product (
        productID INT,
        transactionID INT,
        storeName VARCHAR(255),
        color VARCHAR(255),
        price FLOAT,
        brandName VARCHAR(255),
        name VARCHAR(255),
        type VARCHAR(255),
        PRIMARY KEY(productID)
);
CREATE TABLE Top (
        productID INT PRIMARY KEY,
        hood BIT,
```

```
        size VARCHAR(5),
        pocket BIT,
        zipper BIT,
);

CREATE TABLE Bottom (
        productID INT PRIMARY KEY,
        lengthSize FLOAT,
        waistSize FLOAT
);
CREATE TABLE Shoe (
        productID INT PRIMARY KEY,
        shoeSize FLOAT
);
```

10) Populating tables with sample data

See text files in zip.

DDL data loading script included in the DDL script that creates the database.