Junhui Huang and Yenny Huang
CSc 33600 Database
Final Project Report

Bodega Database

The Bodega database contains three relations consisting of products, prices, and purchases. The motivation behind this database is that a Bodega contains a variety of products and prices that can be evaluated and queried which piqued our interest. We were able to choose what types of products and the prices to set for each, which made it more personal to our choice and enjoyment.

Since there were many parts to this project, we were able to split the parts of writing the Python script, populating CSV files, making the ER diagram, and writing the report. Junhui was responsible for creating the Bodega database, populating the CSV files for the products and prices, writing the Python script, making the ER diagram, and writing the report. Yenny was responsible for populating the CSV file for purchases, writing the Python script, and writing the report. We were both able to share and participate in each other's responsible parts.

The Logic of the Database:

```
CREATE TABLE Products (
      product_id INT NOT NULL auto_increment,
    product_name VARCHAR(50) NOT NULL,
    PRIMARY KEY (product_id),
    UNIQUE(product_name)
);
```

We first created a table of products that consists of the unique ID and unique names for the products, and listed the Product Names associated with that ID. For instance, in our database, when the Product ID is 1, the Product Name will display Coca-Cola. The ID is a primary key each ID is uniquely identified only to a single product.

```
CREATE TABLE Prices (
        product_id INT,
    price DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (product_id) REFERENCES
Products(product_id) ON DELETE CASCADE
ON UPDATE CASCADE
);
```

The price table lists the following attributes of Product ID, Price, and a foreign key that references the Product ID to uniquely identify the specific product's prices. If the ID is 1, then its price would be $1.99. ON DELETE ON UPDATE CASCADE is used when we remove or update a record. The changes in all tables will be based on the product ID.

```
CREATE TABLE Purchases (
        purchase_id INT NOT NULL auto_increment
PRIMARY KEY,
    product_id INT NOT NULL,
    purchase_date DATE NOT NULL,
    quantity INT NOT NULL,
    FOREIGN KEY (product_id) REFERENCES
Products(product_id) ON DELETE CASCADE ON UPDATE
CASCADE
);
```

The purchase table has a unique Purchase ID that indicates each specific purchase for each product. The Product ID will reference the products table. The Purchase Date refers to when the product was recently purchased. The Quantity will display the number of products purchased. The ON DELETE ON UPDATE

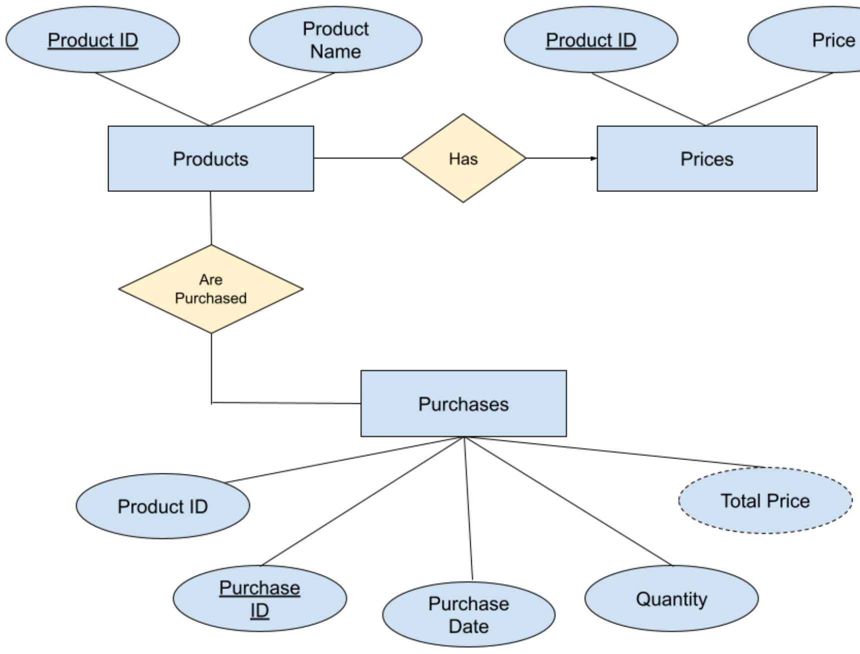
CASCADE is used when we have to remove or update a record. The changes will reflect based on the product ID. The Total (not shown above) column will be created in Python script, and will give the total price for that purchase (total = price * quantity).

Program Overview

The initial process of creating the script (.ipynb) was to create the authentication for a user to have access and query the database. Then, we used CSV files for products, prices, and purchases to populate and insert it into the databases using a Python script. From there we would be able to query and update the database as needed.

For the Products, Prices, Purchases tables, we populated them by inserting the values from our CSV files. Then, we were able to select and view all products. We queried and removed specific products based on their Product Name. Then, we were able to select and view all prices. We started by querying a product for its specific price, which will return our Product IDs that are associated with the price. For the Purchases table, we also inserted a total price column that can calculate the total price from the number of products multiplied by the price of the product. All tables were queried and manipulated to retrieve and apply specific data. For example, we queried the Products table to look for Red Bull and removed an inventory of Kit-Kat. By deleting the record based on the Product Name or Product ID, both Prices and Purchases will cascade the changes. In addition, most of our query statements use parameterized queries, which makes our code more efficient to use. We also inserted products such as Popcorn and Pepsi with their respective new inserts on prices and updates on purchases.

ER Diagram



We were able to successfully create a database in BCNF and write a Python script that can populate the database using CSV files, query, insert, and remove records. Some difficulties that we had were mainly syntax errors and data consistency. Initially, our data were not consistent with each other, which caused issues in writing the scripts, and syntax errors were very common. We had to constantly check and compare our scripts to ensure that our project was working on both ends. To improve our code, we would try to optimize the performance by using more batches of SQL statements. We could have grouped statements into one that had similar functions to ensure that our code or queries are more efficient. For future steps, we would like to add more relations with the products such as including specific suppliers for each product and we would also like to include customer logs of purchases.