Kevin Zhao (zhaokevin79@gmail.com)

Section 1: Simplifying data + Loading into SQLite

For the data given to us to be structured and more easy to query over, I will rearrange all tables into 3rd Normal Form (3NF). In 3NF, all tables will have a primary key, not have duplicate columns or arrays, all non-key columns will be dependent on the primary key, and all non-key-columns will not depend on any other non-key columns. Doing so will remove redundant columns from tables and ensure data integrity as we normalize the raw data given to us.

Restructured Table Format Given Perfect Data* **:

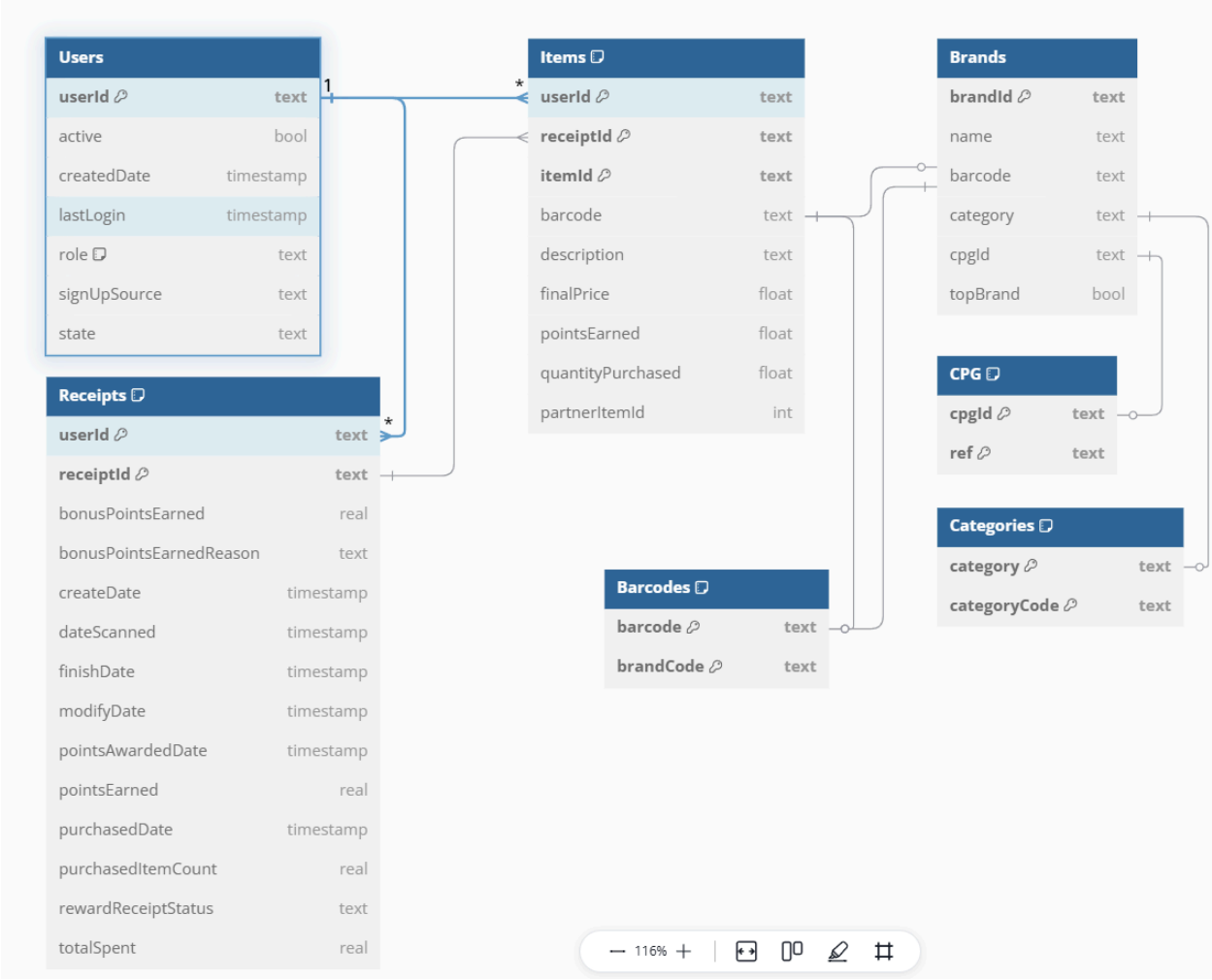

Table relationships:

- Users 1:M Items (i.e. each user can be associated with more than 1 item)
- Users 1:M Receipts
- Receipts 1:M Items
- Items 1:1 Brands (i.e. each item can only be associated with one brand)
- Items 1:1 Barcodes
- Barcodes 1:1 Brands
- Brands 1:1 Categories
- Brands 1:1 CPG

* Receipts were given a prefix key of 'userId', despite receiptId being sufficient enough to uniquely identify each row. This is because in SQLite, data rows with primary keys having the same prefix key will be stored in the same place in memory, leading to faster performances for expensive queries.

** After designing a table schema that I believed would maximize performance efficiency, I noticed many data quality errors that would prevent this table schema from being the one I used for my queries.

I rearranged the tables in order to fit the data differently due to null values, despite the new schema being suboptimal. Below are notes that I took on the tables themselves.

Notes on the Receipts table:

The field 'rewardsReceiptItemList' contains an array of Items that correspond to the receipt itself. Moving these Items to its own table and creating a foreign key that links back to its Receipt will reduce query memory size and execution time for queries involving receipts and maintain 3NF for the Receipts table.

`_id, createDate, dateScanned, modifyDate, rewardsReceiptStatus, userId` are the only columns that are not null in Receipts. However there are many other values that would make sense to be not null, or have a default value (e.g. `pointsEarned, totalSpent, purchasedItemCount, receiptItems`). This could raise issues with data quality/integrity (in section 3)

It is possible for totalSpent to be calculated by summing the individual finalPrice fields for each Item related to the receipt – however, the small storage reduction gained by removing this field does not justify the increased execution time in needing to join the two tables together to calculate the value (especially if we need the totalSpent field for multiple different Receipts). Thus I will keep totalSpent as a field of Receipts.

Notes on the Items table:

The barcode field seems to map 1:1 with brandCode, as each barcode will scan to one item, which has one brand. However, many barcode fields are empty in the Items table, so I will not move barcode and brandCode to a separate table. Additionally, the normalization created by moving these two fields to a separate table does not justify the increased execution time needed to join the Items table to a Barcode table if needed; keeping them together is the right choice.

However, the barcode field is also null for a large portion of the data – to combat this, I will create a new field Items.id, which will serve as the new Primary Key.

Notes on the Users table:

The original structure of this data is already in 3NF as long as we keep the primary key as Users.id, so that all other fields are dependent on the primary key.

Notes on the Brands table:

The cpg column contains two fields: id and ref. Ref is dependent on id, so moving these two fields to a separate Table and keeping a cpgId field in Brands will help normalize the table.

Table Format Given Data Inconsistencies:

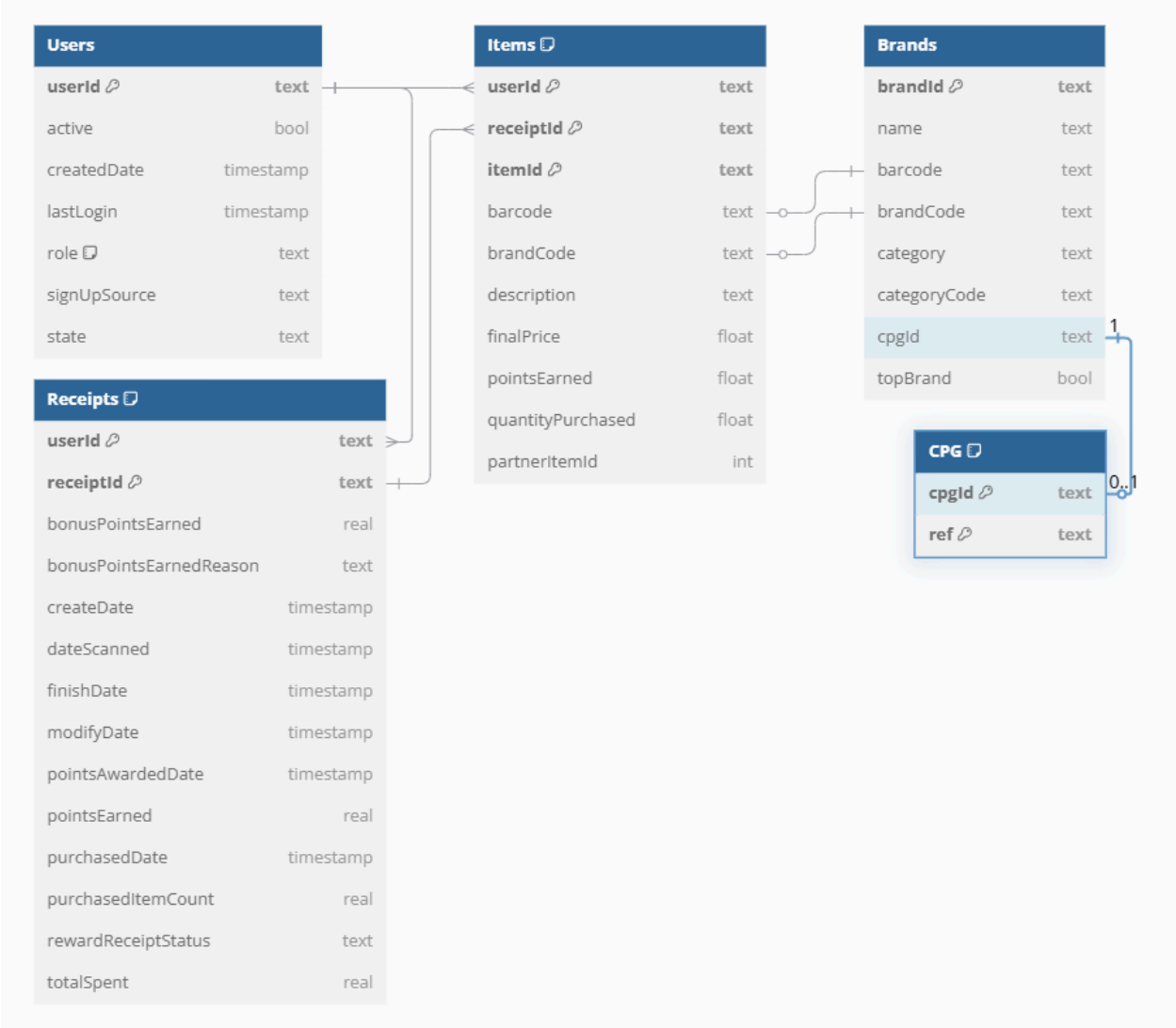

Table relationships:

- Receipts 1:M Items
- Receipts M:1 Users
- Users 1:M Items
- Items 1:1 Brands
- Brands 1:1 CPG