Kevin Zhao (zhaokevin79@gmail.com)

Section 3: Exploratory Data Analysis (EDA) and Data quality issues

For every dataset, performing EDA and conducting data quality is vital to ensure the data is up to company and legal standards and is able to be queried over to obtain data analysis results. I chose to evaluate this data using pandas in addition to my SQLite query testing.

Below I will outline a list of steps I took to perform EDA on this dataset, as well as any outstanding issues I have with the data.

1: Data file formatting

The data comes in 3 .json files, each of which contain one JSON object per line. Despite the files not being in standard JSON formatting (typically a .json file will have one JSON object per file), it was simple to parse by calling f.readlines().

2. Data field types

After looking through the 3 .JSON files, I found many fields which had a suboptimal typing:

- Fields in Receipts.json:
    - pointsEarned is a string, but is better represented with a float (string is formatted 'XX.YY' and points could have fractions)
    - purchasedItemCount is an int, but could be better represented with a float (e.g. what if someone buys a fraction of a pound of an ingredient?)
- Fields in the Items table:
    - itemPrice/finalPrice/userFlaggedPrice are both strings, but should be floats
    - quantityPurchased/userFlaggedQuantity are ints, but could be a float (same reason as purchasedItemCount)
    - totalSpent is a string, but should be a float

Other notes about field formats:

- Dates in this dataset were of type UNIX epoch millisecond time (i.e. ints representing the number of milliseconds since Jan 1, 1970)
- Date fields were also nested – each date field (e.g. Users.createdDate) contained another '$date' field within it]

3. Extraneous fields

There are many extra fields for each Item, which I assume come from the various different ways that Items are added to the dataset (e.g. scanning a physical receipt, scanning an eReceipt, manually entering an Item in, etc.) Here is a list of the fields I could find:

- barcode
- brandCode
- competitiveProduct
- description
- deleted
- discountedItemPrice
- finalPrice
- itemNumber
- itemPrice
- needsFetchReview
- needsFetchReviewReason
- originalFinalPrice
- originalMetaBriteBarcode
- originalMetaBriteDescription
- originalMetaBriteItemPrice
- originalMetaBriteQuantityPurchased
- originalMetaBriteReceiptItemText
- partnerItemId
- pointsEarned
- pointsNotAwardedReason
- pointsPayerId
- preventTargetGapPoints
- quantityPurchased
- rewardsGroup
- rewardsProductPartnerId
- targetPrice
- userFlaggedBarcode
- userFlaggedDescription
- userFlaggedNewItem
- userFlaggedPrice
- userFlaggedQuantity

Many extra fields exist in Receipts and Brands as well. While this doesn't cause an issue for my analysis of the dataset as it stands, performing a data audit and verifying the necessity of these fields could help prevent bloat for future data scaling purposes.

4. Data size + shape

Below are the sizes and shapes for each of the 3 .json files:

Receipts.json:
- Size: 2400 KB
- Shape: 1119 Rows X 14 Columns

Users.json:
- Size: 88 KB
- Shape: 495 Rows X 7 Columns

Brands.json:
- Size: 266 KB
- Shape: 1167 Rows X 8 Columns

From the sizes, Receipts.json is much larger than the other two files, meaning it will be the most expensive to re-load into a database should a crash ever happen.

From the shapes, it is clear that there are far more rows than columns – which is supported by the purpose of the tables being a raw storage of data for Receipts, Users, and Brands, rather than an analysis of other tables.

All 3 files having far more rows than columns also means that aggregate functions between the Tables will be quite expensive, so filtering down data before performing aggregations will substantially reduce query times.

5. Missing Data

While performing EDA and writing test queries for part (2), I noticed that many important columns of the data tables had null values, despite being vital pieces of information for the data analysis. For example, 435 out of 1119 rows of Receipts did not have values for `totalSpent`.

Additionally, (`dateScanned, pointsAwardedDate, pointsEarned, purchaseDate, purchasedItemCount, rewardsReceiptStatus, and userId`)

from Receipts, (barcode, brandCode, description, finalPrice, pointsEarned, and quantityPurchased) from Items, and (brandCode, category, and topBrand) from Brands were all important fields that would be commonly used in queries and would generate valuable insights, but have less significance as they stand due to having so many null values. This is also something I would bring up with the stakeholder as a caution for my results from part (2).

6. Duplicate data

Many rows of data from the 3 .json files are duplicates, or have duplicate keys which cause uniqueness constraint failures when inserting the data into a structured table. For example, the Users table has 283 out of 495 rows with a duplicate Users.id, which should be unique to every user.

7. Outliers/Extraneous values

One last check that is important is making sure that no outliers exist in our numeric data, to ensure that the values returned by our queries in part (2) are reasonable and are not unfairly influenced by bad data. Because the main numerical data fields used in my queries in part (2) are Receipts.totalSpent and Items.finalPrice, I will check both of those columns for suspiciously large/small values. I will also make sure that values for numeric fields make sense (e.g. totalSpent counts how much each receipt was paid for, so the value should never be < 0).

Here is a brief summary of Receipts.totalSpent and Items.finalPrice:

Receipts.totalSpent:
Size: 684
Mean: 77.797
Standard Deviation: 347.11
Number of statistical outliers: 55 (out of 684)
Minimum: 0.0
Maximum: 4721.95

Items.finalPrice:
Size: 6767

Mean: 7.872
Standard Deviation: 14.66
Number of statistical outliers: 476 (out of 6767)
Minimum: 0.0
Maximum: 441.58

Based on the data, around 7-8% of values per field are outliers. Because of this, we should use more precaution when performing statistical analyses on this dataset (e.g. use 99th percentile as a soft max than the true max Of the dataset).