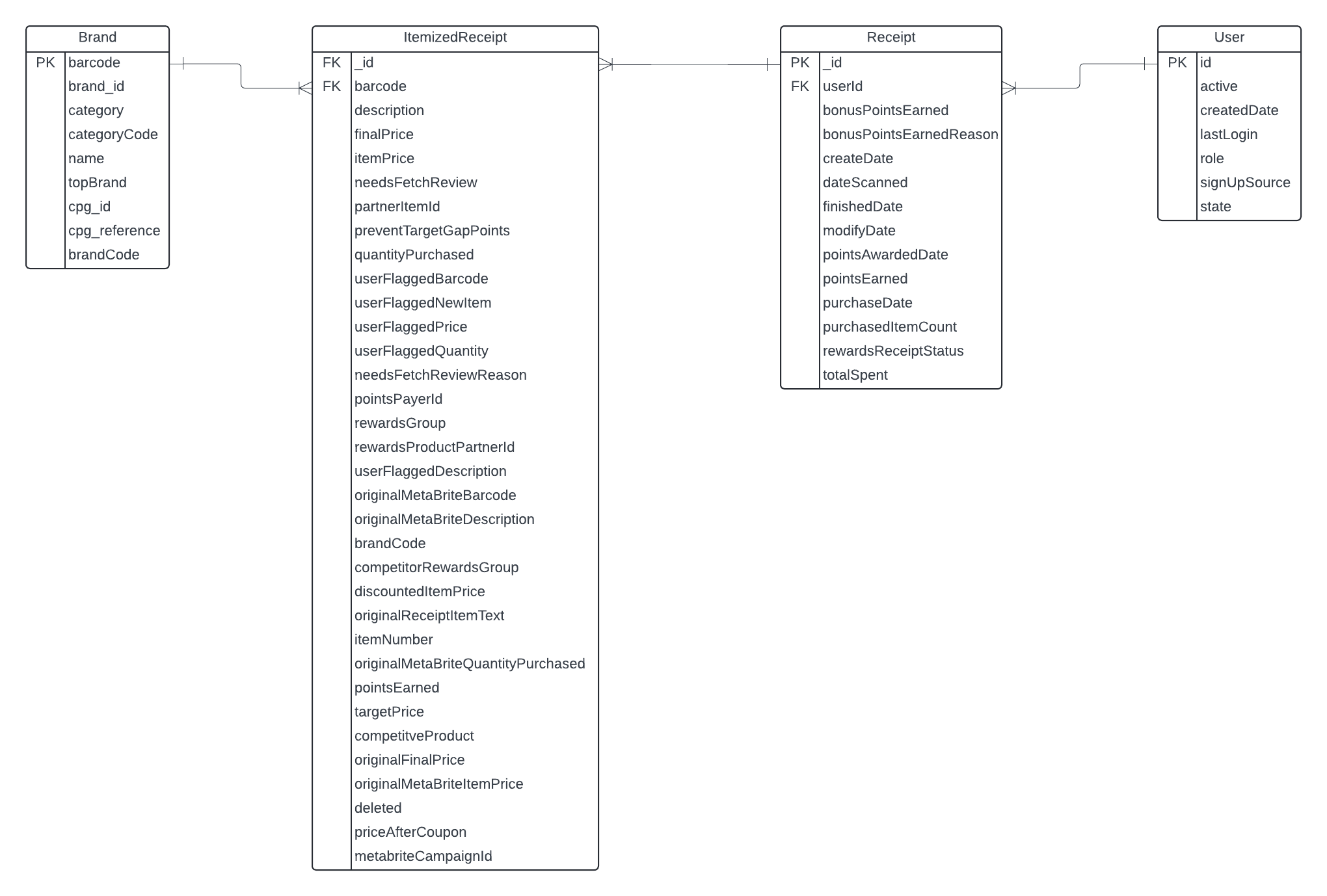
# Fetch Rewards Exercise

## New Structured Relational Data Model

I have decided to move the "itemizedReceipt" table, which corresponds to the "rewardsReceiptItemList" attribute from our receipts.json file. The primary keys and foreign keys are denoted in the first column of each table. While I could have flattened the entire receipts.json into a single table, I opted to maintain a cleaner structure by using a junction table to separate the items. This approach allows for better organization of data and relationships between the tables, enhancing flexibility and normalization in our database schema.



Notes:

1. ItemizedReceipt table is a Junction Table – Brand and Receipt Table had a many to many relationship.
2. Since “barcode” attribute is unique for each brand item, I have used it to establish a relation between the Brand and ItemReceipt tables in the queries.
3. “\_id” and “barcode” attributes together are a composite primary key for the ItemizedReceipt table.
4. “cpg” attribute was a dictionary array, which is broken down into “cpg\_id” and “cpg\_reference” for granularity.
5. Could have made a table ‘Category” but did not have the data for it – need a primary key.

## Queries to answer predetermined business questions

I have built a database and ran all the queries using Google BigQuery (GoogleSQL).

### Query 1 : What are the top 5 brands by receipts scanned for most recent month?

SELECT

    IFNULL(b.name, "Brand Unknown") AS brand\_name, i.barcode,

    SUM (i.quantityPurchased) AS timesBought

FROM

    `stately-lodge-438101-s3.Fetch.itemizedReceipt` AS i

LEFT JOIN

    `stately-lodge-438101-s3.Fetch.brand` AS b ON i.barcode = CAST(b.barcode AS STRING)

LEFT JOIN

    `stately-lodge-438101-s3.Fetch.receipt` AS r ON i.\_id = r.\_id

WHERE

    EXTRACT(MONTH FROM r.dateScanned\_converted) =

    (SELECT EXTRACT(MONTH FROM DATE\_SUB(DATE(MAX(dateScanned\_converted)), INTERVAL 1 MONTH)) FROM `stately-lodge-438101-s3.Fetch.receipt`)

    AND i.barcode IS NOT NULL

GROUP BY

    brand\_name, i.barcode

ORDER BY

    timesBought DESC

LIMIT 5;  -- Get the Top 5 brands with the highest number of transactions

A screenshot of a data

Description automatically generated

Notes:

1. “dateScanned” attributed converted from unix time to datetime format, hence the name “dateScanned\_converted”.
2. No receipts scanned in the most recent month (October 2024), so I have calculated the most recent month in the column and used that.
3. Noticing that the most recent month did not have many receipts scanned and didn’t get the top 5 results, hence went back one more month.
4. These barcodes do not have a matching brand name to it – important for business to identify these.
5. Left Join was used because there were no matching barcodes for the receipts scanned even after the assumption in point number 3 leaving me with no results.
6. Added all the quantity purchased for each item since each receipt can have more than one similar item purchased.
7. Instead of keeping the brand name column null, I decided to replace it with “Brand Unknown”.
8. I included the barcode column since I think it is important for the business to know and acknowledge them even though there is no brand name for them since they were the top 5 barcodes with highest number of transactions.

### Query 3 : When considering *average spend* from receipts with 'rewardsReceiptStatus’ of ‘Accepted’ or ‘Rejected’, which is greater?

SELECT

    rewardsReceiptStatus, round(AVG(totalSpent), 2) AS avgSpending

FROM

    `stately-lodge-438101-s3.Fetch.receipt`

GROUP BY

    rewardsReceiptStatus

ORDER BY

    avgSpending DESC;

A screenshot of a computer screen

Description automatically generated

Notes:

1. Assumed “Accepted” is the same as “Finished”.
2. **“Finished” status is greater when considering average spend from receipts.**

### Query 4 : When considering *total number of items purchased* from receipts with 'rewardsReceiptStatus’ of ‘Accepted’ or ‘Rejected’, which is greater?

SELECT

    rewardsReceiptStatus, SUM(purchasedItemCount) AS itemPurchasedCount

FROM

    `stately-lodge-438101-s3.Fetch.receipt`

GROUP BY

    rewardsReceiptStatus

ORDER BY

    itemPurchasedCount DESC;

A screenshot of a report

Description automatically generated

Notes:

1. Assumed “Accepted” is the same as “Finished”.
2. **“Finished” status is greater when considering total number of items purchased from receipts.**

### Query 5 : Which brand has the most *spend* among users who were created within the past 6 months?

SELECT

    b.name AS brand\_name, i.barcode,

    ROUND(SUM (i.finalPrice),2) as Amount -- sum of amount spent

FROM

    `stately-lodge-438101-s3.Fetch.user` AS u

INNER JOIN

    `stately-lodge-438101-s3.Fetch.receipt` AS r ON u.id = r.userId

INNER JOIN

    `stately-lodge-438101-s3.Fetch.itemizedReceipt` AS i ON r.\_id = i.\_id

INNER JOIN

    `stately-lodge-438101-s3.Fetch.brand` AS b ON i.barcode = CAST(b.barcode AS STRING)

WHERE

    u.createdDate\_converted BETWEEN (

        SELECT TIMESTAMP(DATE\_SUB(DATE(MAX(u.createdDate\_converted)), INTERVAL 6 MONTH))

        FROM `stately-lodge-438101-s3.Fetch.user` AS u

    ) AND (

        SELECT TIMESTAMP(MAX(u.createdDate\_converted))

        FROM `stately-lodge-438101-s3.Fetch.user` AS u

    )

GROUP BY

    brand\_name, i.barcode

ORDER BY

    Amount DESC

LIMIT 5;  -- Get the brand with the highest spend

A screenshot of a data

Description automatically generated

Notes:

1. Since the data is from 2021, past 6 month is calculated from the latest date in the “createdDate\_converted”. Assumption was made to have results.
2. “createdDate” attributed converted from unix time to datetime format, hence the name “createdDate\_converted”.
3. Used Inner join to show results which have a brand name and a barcode. There are a lot of brands with a higher spend than $196.98 but do not have a brand name, only a barcode. It would be useful for the business to find them.
4. **From this result we can see that “Cracker Barrel Cheese” has the most spend, $196.88.**

### Query 6 : Which brand has the most *transactions* among users who were created within the past 6 months?

SELECT

    b.name AS brand\_name, i.barcode,

    COUNT (r.dateScanned) as transactionCount -- Count of transactions

FROM

    `stately-lodge-438101-s3.Fetch.user` AS u

INNER JOIN

    `stately-lodge-438101-s3.Fetch.receipt` AS r ON u.id = r.userId

INNER JOIN

    `stately-lodge-438101-s3.Fetch.itemizedReceipt` AS i ON r.\_id = i.\_id

INNER JOIN

    `stately-lodge-438101-s3.Fetch.brand` AS b ON i.barcode = CAST(b.barcode AS STRING)

WHERE

    u.createdDate\_converted BETWEEN (

        SELECT TIMESTAMP(DATE\_SUB(DATE(MAX(u.createdDate\_converted)), INTERVAL 6 MONTH))

        FROM `stately-lodge-438101-s3.Fetch.user` AS u

    ) AND (

        SELECT TIMESTAMP(MAX(u.createdDate\_converted))

        FROM `stately-lodge-438101-s3.Fetch.user` AS u

    )

GROUP BY

    brand\_name, i.barcode

ORDER BY

    transactionCount DESC

LIMIT 1;  -- Get the brand with the highest number of transactions

A screenshot of a barcode

Description automatically generated

Notes:

1. Since the data is from 2021, past 6 month is calculated from the latest date in the “createdDate\_converted”. Assumption was made to have results.
2. “createdDate” attributed converted from unix time to datetime format, hence the name “createdDate\_converted”.
3. Used Inner join to show results which have a brand name and a barcode. There are a lot of brands with many more transactions than 23 but do not have a brand name, only a barcode. It would be useful for the business to find them.
4. Counted the number of times a brand appeared on a receipt and not the quantity of the item sold since the word transaction was mentioned in the question and I assume one transaction is an entire receipt irrespective of the times an item was bought.
5. **From this result we can see that “Tostitos” has the most transaction count, 23.**

## Data quality issues encountered in the new structured relational data model

* Missing Data:
  + There are many barcodes without any brand name, assuming the barcode in the “rewardsReceiptItemList” column in the receipt table is the same barcode in the brand table. The predetermined business questions had me calculate top brands; lot of the top brands from the query results did not have a name but only a barcode. There are 553 barcodes without a match for brand name. I have attached a snippet for some of the barcodes.

A screenshot of a computer

Description automatically generated

* + Many receipts were empty, they only had IDs and date with no other useful information.

A screenshot of a computer

Description automatically generated

* Redundancy: There were many duplicate User Ids in the Users data, making it hard to join tables.

A screenshot of a table with numbers

Description automatically generated

* While answering the business question : When considering *average spend* from receipts with 'rewardsReceiptStatus’ of ‘Accepted’ or ‘Rejected’, which is greater? and When considering *total number of items purchased* from receipts with 'rewardsReceiptStatus’ of ‘Accepted’ or ‘Rejected’, which is greater? There is no ‘Accepted’ status in the ‘rewardsReceiptStatus’ column. Only ‘Finished’, ‘Rejected’, ‘Pending’, ‘Flagged’.

A screenshot of a computer screen

Description automatically generated

## Communicate with the Stakeholders

Hello Stakeholders,

Hi, my name is Siddharth Gada, from the Analytics Department. We have received the data, and before proceeding with the implementation, I would like to discuss a few aspects to ensure we are aligned.

1. I came across many empty receipts, can you give a scenario in which that is possible and if it contains any meaningful information, or can we simply disregard those receipts?
2. Many of the top performing brands only have barcodes and do not have a brand name, making it hard to identify those brands. Is there any way in which we can identify those brands to help the business?
3. We would like to have a deeper understanding of the dates, there are many gaps in the date columns, for example in the receipt table “createDate” and “dateScanned” attributes have the earliest date in 2020 but “purchaseDate” attribute has the earliest date in 2017.
4. After reviewing the data, I’m confident that key concerns in production will include managing growing data volumes, maintaining query performance, and handling real-time data processing. To address these challenges, I would implement strategies like data partitioning, indexing, and query optimization to enhance performance. Cloud platforms such as AWS and GCP offer powerful tools for improving performance and analyzing data. I’d like to discuss this further in more detail.

Thank you for your time, and I look forward to hearing from you.

Best regards,

Siddharth Gada