**Table of Contents**

[**Review Existing Unstructured Data and Diagram a New Structured Relational Data Model**](#_heading=h.gjdgxs) **2**

[**Write a query that directly answers a predetermined question from a business stakeholder**](#_heading=h.30j0zll) **3**

[What are the top 5 brands by receipts scanned for most recent month?](#_heading=h.1fob9te) 6

[How does the ranking of the top 5 brands by receipts scanned for the recent month compare to the ranking for the previous month?](#_heading=h.3znysh7) 7

[When considering average spend from receipts with 'rewardsReceiptStatus’ of ‘Accepted’ or ‘Rejected’, which is greater?](#_heading=h.2et92p0) 8

[When considering total number of items purchased from receipts with 'rewardsReceiptStatus’ of ‘Accepted’ or ‘Rejected’, which is greater?](#_heading=h.tyjcwt) 9

[Which brand has the most spend among users who were created within the past 6 months?](#_heading=h.3dy6vkm) 9

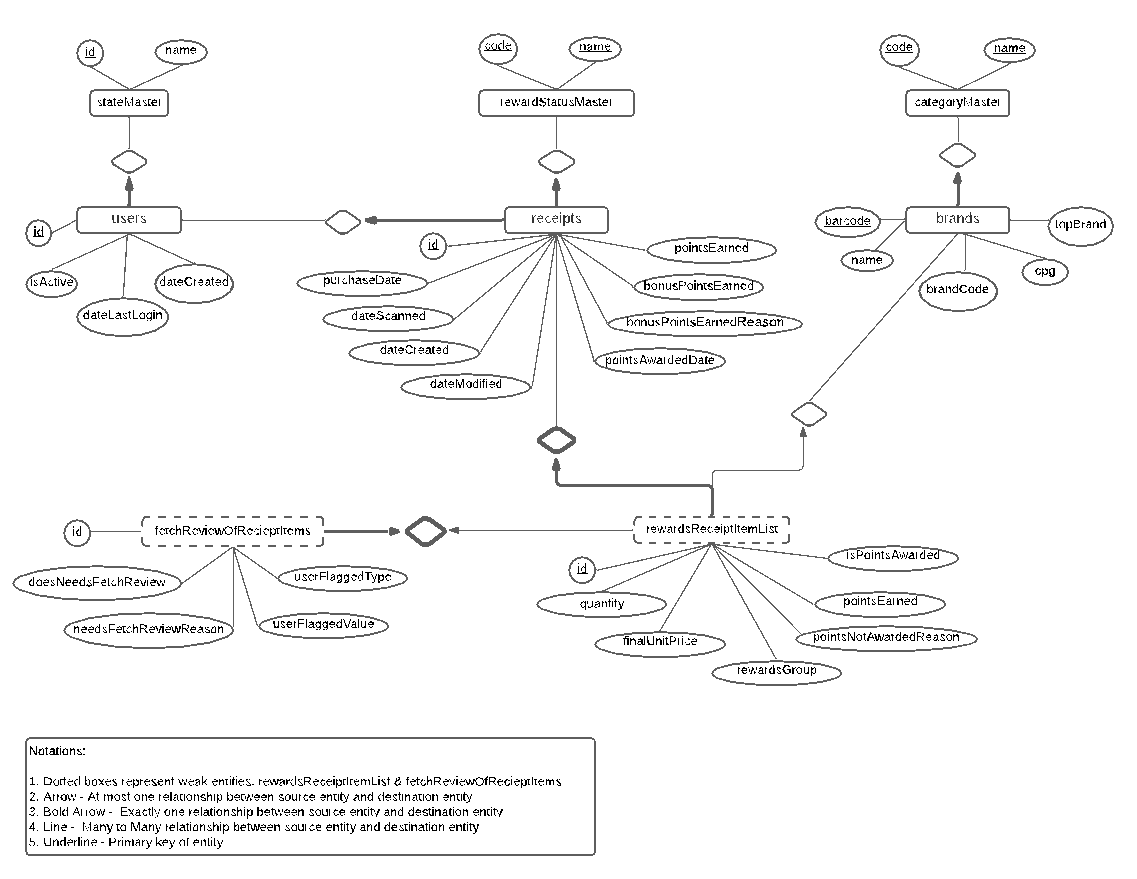
[Which brand has the most transactions among users who were created within the past 6 months?](#_heading=h.1t3h5sf) 10

[*Assumptions:*](#_heading=h.4d34og8) *10*

[**Evaluate Data Quality Issues in the Data Provided**](#_heading=h.2s8eyo1) **10**

[**Communicate with Stakeholders**](#_heading=h.17dp8vu) **11**

# Review Existing Unstructured Data and Diagram a New Structured Relational Data Model



Every entity can be joined with other tables based on their Primary Keys.

* *rewardsRecieptItemList* PRIMARY KEY (id, receiptId, barcode),
* *fetchReviewOfRecieptItems* PRIMARY KEY (id, receiptId, receiptItemId, barcode)
* *Users* JOIN *State* on state.id
* *Reciepts* JOIN *User* on Userid
* *rewardsRecieptItemList* JOIN *Reciepts* on Reciepts.id
* *rewardsRecieptItemList* JOIN *rewardStatusMaster* on rewardStatusMaster.code
* *Brands* JOIN *rewardsRecieptItemList* on brands.barcode
* *fetchReviewOfRecieptItems* JOIN *rewardsRecieptItemList* on RecieptItemList.id, Reciepts.id

# Write a query that directly answers a predetermined question from a business stakeholder

Refer : 2\_fetchRewards.sql

DROP TABLE IF EXISTS [dbo].[fetchReviewOfRecieptItems];

DROP TABLE IF EXISTS [dbo].[rewardsReceiptItemList];

DROP TABLE IF EXISTS [dbo].[receipts];

DROP TABLE IF EXISTS [dbo].[users];

DROP TABLE IF EXISTS [dbo].[brands];

DROP TABLE IF EXISTS [dbo].[stateMaster];

DROP TABLE IF EXISTS [dbo].[categoryMaster];

DROP TABLE IF EXISTS [dbo].[rewardStatusMaster];

DROP TABLE IF EXISTS [dbo].[topBrandLatestMonth];

DROP TABLE IF EXISTS [dbo].[topBrandPreviousMonth];

CREATE TABLE stateMaster(

id varchar(2) PRIMARY KEY,

name varchar(50)

);

CREATE TABLE categoryMaster(

code varchar(20) PRIMARY KEY,

name varchar(50)

);

CREATE TABLE rewardStatusMaster(

code integer PRIMARY KEY,

name varchar(50)

);

CREATE TABLE users(

id varchar(24) PRIMARY KEY,

stateId varchar(2),

isActive bit,

dateLastLogin date,

dateCreated date,

FOREIGN KEY (stateId) REFERENCES stateMaster(id)

);

CREATE TABLE brands(

barcode numeric(12,0) PRIMARY KEY,

name varchar(200),

brandCode varchar(50),

categoryCode varchar(20),

cpg varchar(20),

topBrand bit,

FOREIGN KEY (categoryCode) REFERENCES categoryMaster(code)

);

CREATE TABLE receipts(

id varchar(24) PRIMARY KEY,

userId varchar(24) NOT NULL,

purchaseDate date NOT NULL,

pointsEarned integer,

bonusPointsEarned integer,

bonusPointsEarnedReason varchar(100),

pointsAwardedDate date,

rewardsReceiptStatusId integer,

dateScanned date,

dateFinished date,

dateCreated date NOT NULL,

dateModified date,

FOREIGN KEY (userId) REFERENCES users(id),

FOREIGN KEY (rewardsReceiptStatusId) REFERENCES rewardStatusMaster(code)

);

CREATE TABLE rewardsReceiptItemList(

id integer NOT NULL IDENTITY(1,1),

receiptId varchar(24),

barcode numeric(12,0),

quantity integer NOT NULL,

finalUnitPrice numeric(6,2) NOT NULL,

isPointsAwarded bit NOT NULL,

pointsEarned numeric(10,2),

pointsNotAwardedReason varchar(200),

rewardsGroup varchar(200),

PRIMARY KEY (id, receiptId, barcode),

FOREIGN KEY (receiptId) REFERENCES receipts(id) ON DELETE CASCADE,

FOREIGN KEY (barcode) REFERENCES brands(barcode)

);

CREATE TABLE fetchReviewOfRecieptItems(

id integer IDENTITY(1,1),

receiptId varchar(24),

receiptItemId integer,

barcode numeric(12,0),

doesNeedsFetchReview bit,

needsFetchReviewReason varchar(50),

userFlaggedType varchar(10),

userFlaggedValue varchar(50),

PRIMARY KEY (id, receiptId, receiptItemId, barcode),

FOREIGN KEY (receiptItemId,receiptId,barcode) REFERENCES rewardsReceiptItemList(id,receiptId, barcode)

ON DELETE CASCADE);

/\*\*\*\*\*\*\*\*\*\*Add dummy data for testing query correctness\*\*\*\*\*\*\*\*/

INSERT INTO StateMaster Values ('NY', 'New York');

INSERT INTO StateMaster Values('NJ', 'New Jersey');

INSERT INTO StateMaster Values('NM', 'New Mexico');

INSERT INTO categoryMaster Values ( 'DAIRY', 'Dairy');

INSERT INTO categoryMaster Values ( 'BEVERAGES', 'Beverages');

INSERT INTO categoryMaster Values ( 'Health & Wellness', 'Health & Wellness');

INSERT INTO categoryMaster Values ('Condiments & Sauces', 'Condiments & Sauces')

INSERT INTO categoryMaster Values ('Breakfast & Cereal', 'Breakfast & Cereal')

INSERT INTO rewardStatusMaster Values (1, 'REJECTED');

INSERT INTO rewardStatusMaster Values (2, 'FLAGGED');

INSERT INTO rewardStatusMaster Values (3, 'FINISHED');

INSERT INTO users Values ('5a43c08fe4b014fd6b6a0612', 'NY', 1, '2021-10-10', '2021-01-01');

INSERT INTO users Values ('5e27526d0bdb6a138c32b556', 'NJ', 1, '2021-10-10', '2021-10-06');

INSERT INTO users Values ('5f2068904928021530f8fc34', 'NJ', 1, '2021-10-10', '2021-11-07');

INSERT INTO users Values ('5fa32b4d898c7a11a6bcebce', 'NY', 1, '2021-10-10', '2021-08-08');

INSERT INTO users Values ('5fa41775898c7a11a6bcef3e', 'NM', 1, '2021-10-10', '2021-07-09');

INSERT INTO brands Values (511111704140, 'DIETCHRIS2', 'Diet Chris Cola', 'BEVERAGES', 'Cogs1' ,0);

INSERT INTO brands Values (511111919803, 'PREGO', 'Prego', 'Condiments & Sauces',' Cogs2',1);

INSERT INTO brands Values (511111906124, 'BASIC 4', 'Basic 4™','Breakfast & Cereal', 'Cpgs2', 0);

INSERT INTO brands Values (511111905967, 'Mountain High', 'Mountain High™', 'DAIRY', 'Cpgs1', 1);

INSERT INTO brands Values (511111906070, 'WHEATIES', ' Wheaties™','Breakfast & Cereal','Cogs3',0);

INSERT INTO brands Values (028400642255, 'FLINTSTONES MULTIVITAMIN GUMMY', ' Flintstones™ MULTIVITAMIN GUMMY', 'Health & Wellness', 'Cogs4', 0);

INSERT INTO receipts Values ('5ff1e1eb0a720f0523000575', '5a43c08fe4b014fd6b6a0612',

'2021-01-10', 500,500,' Receipt number 2 completed, bonus point schedule DEFAULT (5cefdcacf3693e0b50e83a36)','2021-01-10',3,'2021-01-10','2021-01-10', '2021-01-10','2021-01-10');

INSERT INTO receipts Values ('5ff1e1bb0a720f052300056b', '5e27526d0bdb6a138c32b556',

'2021-10-10', 150,150,' Receipt number 5 completed, bonus point schedule DEFAULT (5cefdcacf3693e0b50e83a36)','2021-10-10',3,'2021-10-10','2021-10-10','2021-10-10','2021-10-11');

INSERT INTO receipts Values ('5ff1e1cd0a720f052300056f', '5f2068904928021530f8fc34','2021-11-01', 5,5,'All-receipts receipt bonus','2021-11-01', 3,'2021-11-01','2021-11-01','2021-11-01','2021-11-02');

INSERT INTO receipts Values ('5f9c74f70a7214ad07000037', '5fa32b4d898c7a11a6bcebce',

'2021-10-23', 5,5,' All-receipts receipt bonus','2021-10-23',1,'2021-10-23', '2021-10-23','2021-10-23','2021-10-25');

INSERT INTO receipts Values ('5ff1e1d20a7214ada1000561', '5fa41775898c7a11a6bcef3e',

'2021-10-05', 750,7500,'All-receipts receipt bonus','2021-10-05',3,'2021-10-05',

'2021-10-05','2021-10-05','2021-10-07');

INSERT INTO rewardsReceiptItemList values ('5f9c74f70a7214ad07000037',511111704140,

5, 26, 0, 0, 'User Flagged', NULL);

INSERT INTO rewardsReceiptItemList values ('5ff1e1bb0a720f052300056b', 511111704140,1,1.00,0,0, NULL, NULL);

INSERT INTO rewardsReceiptItemList values ('5ff1e1bb0a720f052300056b',

028400642255, 1, 10, 0, 0, 'Action not allowed for user and CPG', ' DORITOS SPICY SWEET CHILI SINGLE SERVE');

INSERT INTO rewardsReceiptItemList values ('5ff1e1cd0a720f052300056f ', 511111919803,

1, 2.23, 0, 0, NULL, NULL);

INSERT INTO rewardsReceiptItemList values ('5ff1e1d20a7214ada1000561',

511111919803, 3, 2.25, 0, 0, 'Action not allowed for user and CPG', 'OLD EL PASO BEANS & PEPPERS');

INSERT INTO rewardsReceiptItemList values ('5ff1e1d20a7214ada1000561',

511111704140, 1, 1, 0, 0, 'Action not allowed for user and CPG', 'OLD EL PASO BEANS & PEPPERS');

INSERT INTO fetchReviewOfRecieptItems values(

'5f9c74f70a7214ad07000037', 1, 511111704140, 0, NULL, NULL, 26);

INSERT INTO fetchReviewOfRecieptItems values(

'5ff1e1bb0a720f052300056b', 2, 511111704140, NULL, NULL, NULL, NULL);

INSERT INTO fetchReviewOfRecieptItems values(

'5ff1e1bb0a720f052300056b', 3, 028400642255, 1, 'USER\_FLAGGED', 'Price', 10);

INSERT INTO fetchReviewOfRecieptItems values(

'5ff1e1cd0a720f052300056f', 4, 511111919803, 0, NULL, NULL, 28);

INSERT INTO fetchReviewOfRecieptItems values(

'5ff1e1d20a7214ada1000561', 5, 511111919803, 1, 'USER\_FLAGGED','Price', 2.56);

INSERT INTO fetchReviewOfRecieptItems values(

'5ff1e1d20a7214ada1000561', 6,

511111704140, NULL, NULL, NULL, NULL);

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

/\*\*\* Assuming TOP 5 Brands means Brands with Most Sales \*\*\*/

SELECT TOP 5 brnd\_itms\_sold.name brand, COUNT(\*) tot\_itm\_sold FROM

(SELECT brnd.barcode, brnd.name, rct\_itm\_lst.receiptId, rct\_itm\_lst.id

FROM receipts rct

JOIN rewardsReceiptItemList rct\_itm\_lst

ON rct.id = rct\_itm\_lst.receiptId

JOIN brands brnd ON brnd.barcode = rct\_itm\_lst.barcode

WHERE MONTH(dateScanned)

IN(SELECT MONTH(MAX(dateScanned)) FROM receipts)) brnd\_itms\_sold

GROUP BY brnd\_itms\_sold.barcode, brnd\_itms\_sold.name

ORDER BY tot\_itm\_sold DESC

/\*\*If item not found to be considered, query will change with LEFT JOIN rewardsReceiptItemList \*\*/

## How does the ranking of the top 5 brands by receipts scanned for the recent month compare to the ranking for the previous month?

/\*\*\*Pseudo code in absence of SQL Server\*\*\*/

CREATE FUNCTION ufn\_topBrandsByMonth (@month int)

RETURNS TABLE

AS

RETURN

(

SELECT TOP 5 brnd\_itms\_sold.name, COUNT(\*) tot\_itm\_sold\_rank FROM

(SELECT brnd.barcode, brnd.name,

rct\_itm\_lst.receiptId, rct\_itm\_lst.id

FROM receipts rct

JOIN rewardsReceiptItemList rct\_itm\_lst

ON rct.id = rct\_itm\_lst.receiptId

JOIN brands brnd ON brnd.barcode = rct\_itm\_lst.barcode

WHERE MONTH(dateScanned) = @month) brnd\_itms\_sold

GROUP BY brnd\_itms\_sold.barcode, brnd\_itms\_sold.name

ORDER BY tot\_itm\_sold\_rank DESC

);

DECLARE @mon INT;

SELECT @mon = MONTH(MAX(dateScanned)) FROM receipts;

SELECT \* INTO topBrandLatestMonth FROM ufn\_topBrandsByMonth(@mon);

SELECT \* INTO topBrandPreviousMonth FROM ufn\_topBrandsByMonth(@mon-1);

SELECT topBrandLatestMonth.brand brandLatestMonth

, topBrandLatestMonth.tot\_itm\_sold\_rank topBrandRankLatestMonth

, topBrandPreviousMonth.brand brandPreviousMonth

, topBrandPreviousMonth.tot\_itm\_sold\_rank topBrandRankPrevMonth

FROM topBrandLatestMonth FULL OUTER JOIN topBrandPreviousMonth

ON topBrandLatestMonth.brand = topBrandPreviousMonth.brand

/\*\*\*Tested code\*\*\*/

DECLARE @month INT;

SELECT @month = MONTH(MAX(dateScanned)) FROM receipts;

SELECT TOP 5 brnd\_itms\_sold.name brand, COUNT(\*) tot\_itm\_sold\_rank

INTO topBrandLatestMonth FROM

(SELECT brnd.barcode, brnd.name,

rct\_itm\_lst.receiptId, rct\_itm\_lst.id

FROM receipts rct

JOIN rewardsReceiptItemList rct\_itm\_lst ON rct.id = rct\_itm\_lst.receiptId

JOIN brands brnd ON brnd.barcode = rct\_itm\_lst.barcode

WHERE MONTH(dateScanned) = @month) brnd\_itms\_sold

GROUP BY brnd\_itms\_sold.barcode, brnd\_itms\_sold.name

ORDER BY tot\_itm\_sold\_rank DESC

SELECT TOP 5 brnd\_itms\_sold.name brand, COUNT(\*) tot\_itm\_sold\_rank

INTO topBrandPreviousMonth FROM

(SELECT brnd.barcode, brnd.name,

rct\_itm\_lst.receiptId, rct\_itm\_lst.id

FROM receipts rct

JOIN rewardsReceiptItemList rct\_itm\_lst ON rct.id = rct\_itm\_lst.receiptId

JOIN brands brnd ON brnd.barcode = rct\_itm\_lst.barcode

WHERE MONTH(dateScanned) = @month-1) brnd\_itms\_sold

GROUP BY brnd\_itms\_sold.barcode, brnd\_itms\_sold.name

ORDER BY tot\_itm\_sold\_rank DESC

SELECT topBrandLatestMonth.brand brandLatestMonth

, topBrandLatestMonth.tot\_itm\_sold\_rank topBrandRankLatestMonth

, topBrandPreviousMonth.brand brandPreviousMonth

, topBrandPreviousMonth.tot\_itm\_sold\_rank topBrandRankPrevMonth

FROM topBrandLatestMonth FULL OUTER JOIN topBrandPreviousMonth

ON topBrandLatestMonth.brand = topBrandPreviousMonth.brand

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

/\*\* There is no status Accepted in Dataset, using Finished instead\*\*\*/

SELECT CASE WHEN rewardsReceiptStatusId=1 THEN 'Rejected'

ELSE 'Finished' END as rewardsReceiptStatus

, AVG(totSpentPerRct.tot\_spent) avgSpentPerStatus

, RANK() OVER(ORDER BY AVG(totSpentPerRct.tot\_spent) DESC) Rank

FROM receipts rct

JOIN

(SELECT receiptId

, SUM(finalUnitPrice) tot\_spent

FROM rewardsReceiptItemList

GROUP BY receiptId) totSpentPerRct

ON rct.id = totSpentPerRct.receiptId

WHERE rct.rewardsReceiptStatusId IN

(SELECT code

FROM rewardStatusMaster

WHERE name IN('Finished','Rejected'))

GROUP BY rct.rewardsReceiptStatusId

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

/\*\* There is no status Accepted in Dataset, using Finished instead\*\*\*/

SELECT CASE WHEN rewardsReceiptStatusId=1 THEN 'Rejected'

ELSE 'Finished' END as rewardsReceiptStatus

,SUM(itm\_count.tot\_itms) tot\_itm\_purchsd

,RANK() OVER(ORDER BY SUM(itm\_count.tot\_itms) DESC) Rank

FROM receipts rct

JOIN

(SELECT receiptId

, SUM(quantity) tot\_itms

FROM rewardsReceiptItemList

GROUP BY receiptId) itm\_count

ON rct.id = itm\_count.receiptId

WHERE rct.rewardsReceiptStatusId IN

(SELECT code

FROM rewardStatusMaster

WHERE name IN('Finished','Rejected'))

GROUP BY rct.rewardsReceiptStatusId

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

SELECT brnd.name brand

, sale\_lst.brand\_spending

, RANK() OVER(ORDER BY sale\_lst.brand\_spending DESC) Rank

FROM Brands brnd

JOIN

(SElECT barcode

, SUM(finalUnitPrice\*quantity) brand\_spending

FROM rewardsReceiptItemList

WHERE receiptId IN

(SELECT id FROM receipts

WHERE userId IN

(SELECT id FROM Users

WHERE dateCreated >= DATEADD(month, -6, getdate())))

GROUP BY barcode) sale\_lst

ON brnd.barcode = sale\_lst.barcode

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

SELECT brnd.name

, sale\_lst.brand\_trnsctn

, DENSE\_RANK() OVER(ORDER BY sale\_lst.brand\_trnsctn DESC) Rank

FROM Brands brnd

JOIN

(SELECT barcode

, COUNT(\*) brand\_trnsctn

FROM rewardsReceiptItemList

WHERE receiptId IN

(SELECT id FROM receipts

WHERE userId IN

(SELECT id FROM Users

WHERE dateCreated >= DATEADD(month, -6, getdate())))

GROUP BY barcode) sale\_lst

ON brnd.barcode = sale\_lst.barcode

### Assumptions:

1. Item Price in receiptItemList is unit Price
2. If quantity in receiptItemList > 1, it is assumed that item is charged as a group
3. Sum of all receiptItemList.quantity should be the purchaseItemCount of receipt
4. Sum of all receiptItemList.itemPrice should be the finalPrice of receipt
5. Reciept status 'Accepted' meant 'Finished'
6. There are a lot of values in receiptItemList, only some of which have been used in our analysis based on the predetermined questions.

# Evaluate Data Quality Issues in the Data Provided

Refer: 3\_fetch\_data\_analyst

# Communicate with Stakeholders

Hello All,

I have completed an in-depth analysis of the datasets shared. Please find below a summary of the Fetch Rewards Data Analysis.

**Overview of data:**

The data is available in 3 json files (unstructured format)

* Users – Contains information about users/customers.
* Brands – Contains information about various brands sold.
* Receipts – Contains detailed information of each sale as captured in its receipts.

The requirement is to convert data from unstructured format to a structured format to make it more readable and for facilitating running queries on the data for further analysis.

**Data Quality Issues Observed:**

* User dataset should ideally serve as the master table for user information, with no redundant user information. But, *UserID* in the Users dataset does not have unique values.
* Brand dataset should ideally serve as the master table for brand information, with no redundant brand information. But, *barcode or brandCode* in the Brand Dataset does not have unique values.
* Receipts dataset stores data at individual receipt level. The itemized list of the purchase is stored in attribute *rewardsReceiptItemList.* The *rewardsReceiptItemList* attribute has a nested structure to accommodate the list of details of each item purchased. The problem with this design is it cannot be converted into a structured relational model without making it into a separate entity. Moreover, the structure of rewardsReceiptItemList is highly inconsistent. It has many columns which appear to be redundant e.g. *targetPrice, discountedPrice, finalPrice*.
* Dates are stored as numbers in all the datasets.

**Possible resolution of data issues:**

* Dropping duplicates from Users and Brands dataset will help remove redundancy from the datasets.
* In my view a better design would be to split the Receipts into 3 parts –
  + Receipts summary information – Receipts
  + Receipts itemized bill information – rewardsReceiptItemList
  + Receipts points information for each item in the itemized bill - fetchReviewOfReceiptItems
* The nested json structure is difficult to traverse and compare across rows. Data transformation is necessary to make the datasets query-able.
* If Dates were fudged, no resolution required; else knowing the original format can help understand how it got transformed into this numerical form and if there is any way to revert it by datatype conversions.

**Open Questions about data:**

Due to the nested nature of json I used a combination of Tableau and Jupyter Notebook to query and analyze the data. During my analysis I could not arrive at conclusive answers for relation and significance of certain attributes.

* No observable logic deciphered to draw a relation between points and price. Having a dataset which defines the rules of awarding points will be very helpful.
* The significance of CPG for brands and the purpose of storing it in rewardsReceiptItemList is not well understood.
* What does the various types of price stand for and under what business rule do they vary?
* What does the various types of points stand for and under what business rule do they vary?
* Does status UserFlagged indicate an issue with the transaction/purchase/points? Should this be of special interest?
* A handful of data has needsFetchReview as True. Does this have an impact on the overall receipt/pointsEarned/BonusPoints?

I have made the following assumptions regarding the Receipts Dataset and rewardsReceiptItemList attribute. If these holds then the suggested 3-part decomposition of the Receipt dataset should work seamlessly.

1. Item Price in receiptItemList is unit Price
2. If quantity in receiptItemList > 1, it is assumed that item is charged as a group
3. Sum of all receiptItemList.quantity should be the purchaseItemCount of receipt
4. Sum of all receiptItemList.itemPrice should be the finalPrice of receipt
5. User\_FlaggedType and User\_FlaggedValue should be enough to capture the various kinds of reasons a user can flag an item. This structure will remove multiple NULL valued columns

Performance issues should be first tried to be handled using indexing and query rewrite. If the data store grows exponentially, using a Platform as a Service (PaaS) architecture can help to dynamically add more resources in minimum downtime when the allocated resource limit is reached. Distributed parallel processing of transactions using Spark on datafarm can also be a viable option to handle performance issues that cannot be handled by traditional database systems.

Please let me know your thoughts on this.

Best,

Bornita Das