We will be performing the required analysis in this Jupyter Notebook using a library called pandasql

pandasql takes a sql query as input and runs it on a pandas dataframe to get the response

```
#installing the required packages
!pip install --upgrade pandas
!pip install pandasql

#Need this version of SQLALchemy
!pip install SQLAlchemy==1.4.46
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pandas in /usr/local/lib/python3.8/dist-packages (1.3.5)
Collecting pandas
  Downloading pandas-1.5.3-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.2/12.2 MB 33.5 MB/s eta 0:00:00
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.8/dist-packages (from pandas) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.8/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: numpy>=1.20.3 in /usr/local/lib/python3.8/dist-packages (from pandas) (1.21.6)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-packages (from python-dateutil>=2.8.1->pandas) (1.1
Installing collected packages: pandas
  Attempting uninstall: pandas
    Found existing installation: pandas 1.3.5
    Uninstalling pandas-1.3.5:
      Successfully uninstalled pandas-1.3.5
Successfully installed pandas-1.5.3
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pandasql
  Downloading pandasql-0.7.3.tar.gz (26 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (from pandasql) (1.21.6)
Requirement already satisfied: pandas in /usr/local/lib/python3.8/dist-packages (from pandasql) (1.5.3)
Requirement already satisfied: sqlalchemy in /usr/local/lib/python3.8/dist-packages (from pandasql) (2.0.0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.8/dist-packages (from pandas->pandasql) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.8/dist-packages (from pandas->pandasql) (2.8.
Requirement already satisfied: typing-extensions>=4.2.0 in /usr/local/lib/python3.8/dist-packages (from sqlalchemy->pandasql)
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.8/dist-packages (from sqlalchemy->pandasql) (2.0.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-packages (from python-dateutil>=2.8.1->pandas->panda
Building wheels for collected packages: pandasql
  Building wheel for pandasql (setup.py) ... done
  Created wheel for pandasql: filename=pandasql-0.7.3-py3-none-any.whl size=26787 sha256=939f5044fb794c09d8f5f722cab121f26da
  Stored in directory: /root/.cache/pip/wheels/ed/8f/46/a383923333728744f01ba24adbd8e364f2cb9470a8b8e5b9ff
Successfully built pandasql
Installing collected packages: pandasql
Successfully installed pandasql-0.7.3
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting SQLAlchemy==1.4.46
  Downloading SQLAlchemy-1.4.46-cp38-cp38-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.w
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 22.2 MB/s eta 0:00:00
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.8/dist-packages (from SQLAlchemy==1.4.46) (2.0.2)
Installing collected packages: SQLAlchemy
  Attempting uninstall: SQLAlchemy
    Found existing installation: SQLAlchemy 2.0.0
    Uninstalling SQLAlchemy-2.0.0:
      Successfully uninstalled SQLAlchemy-2.0.0
Successfully installed SQLAlchemy-1.4.46
```

```
# import the required libraries
import numpy as np
import pandas as pd
from pandasql import sqldf
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
## Uploaded the files to the temporary run time of Google Colab
## Load each of the csvs into a dataframe
brands = pd.read_csv('brands.csv')
receipt_items = pd.read_csv('receipt_items.csv')
receipts = pd.read_csv('receipts.csv')
users = pd.read_csv('users.csv')
```

BRANDS

```
brands.head()
```

	ID	BARCODE	BRAND_CODE	CPG_ID	CATEGORY	CATEGORY_CODE	NAME	ROMANCE_TEX
0	5a8c35dde4b0ccf165fac9e6	511111904175	PEPPERIDGE FARM	5a734034e4b0d58f376be874	Grocery	GROCERY	Pepperidge Farm	Pepperidge Farn has been makin exceptional co..
1	6234af8f4e09b6067c237adb	511111212997	CHEX MIX	6233966e8942a67af4934aa1	Snacks	SNACKS	CHEX MIX	NaN
2	5332f7d3e4b03c9a25efd14e	511111803393	NaN	5332f5f2e4b03c9a25efd0aa	Snacks	SNACKS	Cheez-It	NaN
3	5d6412e9a3a018514994f426	511111304982	BETTER HOMES & GARDENS	53e10d6368abd3c7065097cc	Magazines	MAGAZINES	Better Homes & Gardens Magazine	Better Homes & Gardens offer beautiful photog..
4	621e777eacedc065cefa99a7	511111912859	TRUVIA	621e7754d759b10969cbcc08	Baking	BAKING	Truvia®	NaN

```
## This Gives the count of null values in each column of the Brand Table
##Id is our Primary Key in this casw which is not null
brands.isnull().sum()
```

```
ID                    0
BARCODE               0
BRAND_CODE           25
CPG_ID                0
CATEGORY             27
CATEGORY_CODE        31
NAME                  0
ROMANCE_TEXT        103
RELATED_BRAND_IDS   243
dtype: int64
```

```
len(brands)
```

```
406
```

```
## The table below shows the Number of Unique values in each column. The Brands table has 406 rows as printed in the above cell
brands.nunique()
```

```
ID                  406
BARCODE             406
BRAND_CODE          380
CPG_ID               92
CATEGORY             32
CATEGORY_CODE        32
NAME                404
ROMANCE_TEXT        299
RELATED_BRAND_IDS   156
dtype: int64
```

```
# Shows how many unique categories we have
brands['CATEGORY'].unique()
```

```
array(['Grocery', 'Snacks', 'Magazines', 'Baking', 'Beverages', 'Baby',
       'Frozen', 'Personal Care', 'Dairy & Refrigerated',
       'Cleaning & Home Improvement', 'Beer, Hard Cider & Seltzer',
       'Candy & Sweets', 'Condiments & Sauces', nan, 'Retailers',
       'Health & Wellness', 'Pet', 'Gum & Mints', 'Wine',
       'Breakfast & Cereal', 'Household', 'Beer Wine Spirits', 'Spirits',
       'Beauty', 'Restaurants', 'Beauty & Personal Care',
       'Candy & Chocolate', 'Dairy', 'Canned Goods & Soups',
       'Bread & Bakery', 'Deli', 'Meat & Seafood', 'Oral Care'],
      dtype=object)
```

For Different brand Categories, we check what percent of Each Category is Present in our Brands.csv

Snacks and Beverages take the Top 2 Spots in our Analysis as seen below

```
perc=(brands['CATEGORY'].value_counts()/len(brands))*100
perc.map('{:,.2f}%'.format)
```

```
Snacks                          13.30%
Beverages                       10.84%
Grocery                          7.14%
Frozen                           7.14%
Dairy & Refrigerated             4.93%
Personal Care                    4.43%
Condiments & Sauces              4.43%
Breakfast & Cereal               3.69%
Bread & Bakery                   3.45%
Baking                           2.96%
Beer, Hard Cider & Seltzer       2.96%
Household                        2.96%
Candy & Sweets                   2.71%
Wine                             2.22%
Cleaning & Home Improvement      1.97%
Restaurants                      1.72%
Candy & Chocolate                1.72%
Canned Goods & Soups             1.72%
Health & Wellness                1.48%
Retailers                        1.48%
Pet                              1.23%
Spirits                          1.23%
Baby                             1.23%
Magazines                        1.23%
Meat & Seafood                   0.99%
Deli                             0.74%
Gum & Mints                      0.74%
Dairy                            0.74%
Beauty                           0.74%
Beauty & Personal Care           0.49%
Oral Care                        0.49%
Beer Wine Spirits                0.25%
Name: CATEGORY, dtype: object
```

To Conclude, there are no major Issues with the Brands Table, there are quite a few null values which would need to be handled if we were to go ahead and train a model on this file but otherwise the data seems to be good

Receipts

```
receipts.head()
```

	ID	STORE_NAME	PURCHASE_DATE	PURCHASE_TIME	DATE_SCANNED	TOTAL_SPENT	REWARDS_RECEIPT_STATUS	
0	62868f660a72546bef0b2dd0	TOWN OF ROCKY MOUNT	2022-05-19T00:00:00Z	2:05 PM	2022-05-19T18:41:42.53Z	859.87	FINISHED	61375
1	6096b7370a7216d316001149	NaN	NaN	NaN	2021-05-08T16:07:19.03Z	NaN	SUBMITTED	60047
2	6269a4ea0a7241077408b6e1	FAMILY DOLLAR	2022-04-27T00:00:00Z	4:15 PM	2022-04-27T20:17:46.09Z	11.00	FINISHED	6157
3	625b25e70a723eb9730d2c9c	PUBLIX	2022-04-15T00:00:00Z	4:45 PM	2022-04-16T20:24:07.259Z	10.67	FINISHED	6048
4	60e3bd7e0a7215bd550fb8cc	COSTCO	2021-06-30T00:00:00Z	1:16 PM	2021-07-06T02:18:38.495Z	61.90	FINISHED	6048

5 rows × 21 columns

```
receipts.isnull().sum()
```

```
ID                         0
STORE_NAME              1836
PURCHASE_DATE           2066
PURCHASE_TIME           4947
DATE_SCANNED               0
TOTAL_SPENT             1492
REWARDS_RECEIPT_STATUS     0
USER_ID                    0
```

```
USER_VIEWED                  6465
PURCHASED_ITEM_COUNT         1452
CREATE_DATE                     0
PENDING_DATE                 1453
MODIFY_DATE                     2
FLAGGED_DATE                66576
PROCESSED_DATE              70601
FINISHED_DATE                6252
REJECTED_DATE               66217
NEEDS_FETCH_REVIEW          70276
DIGITAL_RECEIPT                 0
DELETED                     69733
NON_POINT_EARNING_RECEIPT    8986
dtype: int64
```

```
receipts.isnull().sum() * 100 / len(receipts)
```

```
ID                             0.000000
STORE_NAME                     2.600530
PURCHASE_DATE                  2.926304
PURCHASE_TIME                  7.006983
DATE_SCANNED                   0.000000
TOTAL_SPENT                    2.113285
REWARDS_RECEIPT_STATUS         0.000000
USER_ID                        0.000000
USER_VIEWED                    9.157094
PURCHASED_ITEM_COUNT           2.056628
CREATE_DATE                    0.000000
PENDING_DATE                   2.058045
MODIFY_DATE                    0.002833
FLAGGED_DATE                  94.298948
PROCESSED_DATE               100.000000
FINISHED_DATE                  8.855399
REJECTED_DATE                 93.790456
NEEDS_FETCH_REVIEW            99.539667
DIGITAL_RECEIPT                0.000000
DELETED                       98.770556
NON_POINT_EARNING_RECEIPT     12.727865
dtype: float64
```

This seems to be one of the major issues with the receipts data. There are lots of nulls present which would need to be handled if we were going to train a model or perform any sort of analysis on the data.

There are columns like Processed Date, Flagged Date, Rejected Date etc which have more than 90% values missing. Depending on their importance, it might be better to completely do away with these columns

```
## Converting from type object to datetime
receipts[["PURCHASE_DATE", "PURCHASE_TIME", "DATE_SCANNED","CREATE_DATE","PENDING_DATE","MODIFY_DATE","FLAGGED_DATE","PROCESSED_DA
```

Which user spent the most money in the month of August?

```
# Pandasql accepts sql lite version of the query. I have mentioned both of the Syntaxes
#         SELECT USER_ID,SUM(TOTAL_SPENT) as Total \
#         from receipts\
#         WHERE extract(year from PURCHASE_DATE) =2022 and\
#         extract(month from PURCHASE_DATE = 8\
#         GROUP BY USER_ID\
#         ORDER BY TOTAL DESC\
#         LIMIT 1"

#Most Money Spent in August 2022
query = "SELECT USER_ID,SUM(TOTAL_SPENT) as Total \
        from receipts\
        WHERE strftime('%Y', PURCHASE_DATE)= '2022' \
        AND strftime('%m', PURCHASE_DATE)= '08'\
        GROUP BY USER_ID\
        ORDER BY TOTAL DESC\
        LIMIT 1"

df = sqldf(query)
df.head()
```

	USER_ID	Total

Most Money Spent in the month of August irrespective of the Year

```
query = "SELECT USER_ID,SUM(TOTAL_SPENT) as Total \
        from receipts\
        WHERE strftime('%m', PURCHASE_DATE)= '08'\
        GROUP BY USER_ID\
        ORDER BY TOTAL DESC\
        LIMIT 1"

df = sqldf(query)
df.head()
```

	USER_ID	Total
0	609ab37f7a2e8f2f95ae968f	157739.14

How many users scanned in each month?

```
#mysql_query = "SELECT extract(month from DATE_SCANNED) as MONTH_SCANNED,count(USER_ID) as\
#         from receipts\
#         GROUP BY MONTH_SCANNED\
#         ORDER BY TOTAL DESC"


query = "SELECT strftime('%m', DATE_SCANNED) as MONTH_SCANNED,count(USER_ID) as TOTAL_SCANNED \
        from receipts\
        GROUP BY MONTH_SCANNED\
        ORDER BY TOTAL_SCANNED DESC"



df = sqldf(query)
df
```

	MONTH_SCANNED	TOTAL_SCANNED
0	12	8447
1	11	7512
2	10	7305
3	09	6355
4	08	6191
5	07	6058
6	05	5627
7	06	5405
8	04	4882
9	03	4767
10	01	4222
11	02	3830

Number of Users Scanned Month Wise

```
receipt_items.isnull().sum() * 100 / len(receipts)
```

RECEIPT_ITEMS

```
receipt_items.head()
```

	REWARDS_RECEIPT_ID	ITEM_INDEX	REWARDS_RECEIPT_ITEM_ID	DESCRIPTION	BARCODE	BRAND_CODE	QUANTITY_PURCHASED
0	60bb28c10a720d557b128262	0	1efd6d7c75ecbae32214acb6cda41d12	RLGULAR SALE	NaN	NaN	1.0
1	60bb28c10a720d557b128262	1	79482a8fa3bd0eef3d626f1c862042e8	82 GOURMET HOUSEW	000240292012	NaN	1.0
2	627151230a724d730825106a	0	b26669cf4ce90cc9d7d3b0ab588cb04b	GOLDILOCKS NOPIA R BLAGK	NaN	NaN	1.0
3	627151230a724d730825106a	1	b4fafd04d8274a1e95b97155edaade2f	KURI-IRI DORAYAKI CAKE	NaN	NaN	1.0
4	627151230a724d730825106a	2	39694b0880b511e8a12bfb76cf2c20f3	YIZMANG FISH BALL	NaN	NaN	1.0

Check for percent of Null Values Column Wise

```
(receipt_items.isnull().sum() * 100 / len(receipt_items)).sort_values(ascending=False)
```

```
POINTS_EARNED                 94.741063
REWARDS_GROUP                 82.813276
BRAND_CODE                    57.020842
BARCODE                       37.563163
QUANTITY_PURCHASED             2.152191
ORIGINAL_RECEIPT_ITEM_TEXT     0.466456
DESCRIPTION                    0.302739
TOTAL_FINAL_PRICE              0.192021
REWARDS_RECEIPT_ID             0.000000
ITEM_INDEX                     0.000000
REWARDS_RECEIPT_ITEM_ID        0.000000
MODIFY_DATE                    0.000000
dtype: float64
```

Points Earned and Rewards Group have more than 80% values as Null

Checking for Duplicate rows

```
receipt_items[receipt_items.duplicated()]
```

REWARDS_RECEIPT_ID	ITEM_INDEX	REWARDS_RECEIPT_ITEM_ID	DESCRIPTION	BARCODE	BRAND_CODE	QUANTITY_PURCHASED	TOTAL_FINAL_PI

No duplicated Rows in The Data

What brand saw the most money spent in the month of June

```
# Money Spent in June Throughout
# sql_query = "SELECT BRAND_CODE,SUM(TOTAL_FINAL_PRICE) as Total \
        # from receipt_items\
        # WHERE extract(month from MODIFY_DATE)= '06'\
        # GROUP BY BRAND_CODE\
        # ORDER BY TOTAL DESC\
        # LIMIT 5"

query = "SELECT BRAND_CODE,SUM(TOTAL_FINAL_PRICE) as Total \
        from receipt_items\
        WHERE strftime('%m', MODIFY_DATE)= '06'\
        GROUP BY BRAND_CODE\
        ORDER BY TOTAL DESC\
```

```
        LIMIT 5"

df = sqldf(query)
df.head()
```

	BRAND_CODE	Total
0	None	179922.28
1	KIRKLAND SIGNATURE	2610.67
2	GREAT VALUE	1543.84
3	MEMBER'S MARK	819.93
4	KROGER	785.29

```
#Money spent in June 2022
query = "SELECT BRAND_CODE,SUM(TOTAL_FINAL_PRICE) as Total \
        from receipt_items\
        WHERE strftime('%Y', MODIFY_DATE)= '2022'\
        AND strftime('%m', MODIFY_DATE)= '06'\
        GROUP BY BRAND_CODE\
        ORDER BY TOTAL DESC\
        LIMIT 5"

df = sqldf(query)
df.head()
```

	BRAND_CODE	Total
0	None	112145.74
1	KIRKLAND SIGNATURE	1822.17
2	GREAT VALUE	1185.49
3	ANDERSEN	706.00
4	CARDELL	556.98

USERS

```
users.head()
```

	CREATED_DATE	BIRTH_DATE	GENDER	LAST_REWARDS_LOGIN	STATE	SIGN_UP_PLATFORM	SIGN_UP_SOURCE	
0	2021-12-20T00:29:17.118Z	1984-03-20T00:00:00Z	transgender	2023-01-04T16:32:15Z	FL	NaN	Apple	61bfce
1	2021-10-21T17:15:25.825Z	1987-08-08T05:00:00Z	prefer_not_to_say	2023-01-04T16:04:33Z	PA	unknown	Google	6171a(
2	2021-10-23T19:19:18.305Z	1995-06-18T05:00:00Z	male	2023-01-04T16:13:13Z	FL	NaN	Apple	617460
3	2021-03-30T02:35:41.249Z	1999-08-23T07:00:00Z	transgender	2023-01-04T16:09:51Z	MI	ios	Google	60628
4	2021-04-26T23:15:54.375Z	1992-10-28T16:16:23Z	male	2023-01-04T16:24:18Z	CA	andrioid	Email	60874

```
#Checking for % of Nulls in Users

(users.isnull().sum() * 100 / len(users)).sort_values(ascending=False)
```

```
SIGN_UP_PLATFORM       27.439024
CREATED_DATE            0.000000
BIRTH_DATE              0.000000
GENDER                  0.000000
LAST_REWARDS_LOGIN      0.000000
STATE                   0.000000
SIGN_UP_SOURCE          0.000000
ID                      0.000000
dtype: float64
```

Only the SIGN_UP_PLATFORM is Missing 28% values. Rest of the Data is in a very good shape

Lets find the top 5 States Percentage Wise where Most of the Users are from

```
perc=(users['STATE'].value_counts()/len(users))*100
perc.map('{:,.2f}%'.format)[:5]
```

```
FL    9.76%
NY    9.15%
PA    7.32%
TX    6.71%
CA    6.71%
Name: STATE, dtype: object
```

What platform are most of the users signing up from

```
perc=(users['SIGN_UP_SOURCE'].value_counts()/len(users))*100
perc.map('{:,.2f}%'.format)
```

```
Apple       31.10%
Google      23.17%
Facebook    23.17%
Email       22.56%
Name: SIGN_UP_SOURCE, dtype: object
```

0s completed at 12:46 PM