Diagram, schematic

Description automatically generated

**Fig 1:** ERD to model data in a data warehouse.

**SQL CODE TO CREATE TABLES (POSTGRESQL)**

CREATE TABLE users (

\_id Text,

active Boolean,

createdDate Timestamp,

lastLogin Timestamp,

role Varchar(50),

signUpSource Varchar(50),

state Varchar(50),

PRIMARY KEY (\_id)

);

CREATE TABLE receipts (

\_id Text,

bonusPointsEarned float(2),

bonusPointsEarnedReason Varchar(250),

createDate Timestamp,

dateScanned Timestamp,

finishedDate Timestamp,

modifyDate Timestamp,

pointsAwardedDate Timestamp,

pointsEarned float(2),

purchaseDate Timestamp,

purchaseItemCount float(2),

rewardsReceiptStatus Varchar(50),

totalSpent float(2),

user\_id Text,

PRIMARY KEY (\_id),

CONSTRAINT FK\_Receipts\_user\_id

FOREIGN KEY (user\_id)

REFERENCES users(\_id)

);

CREATE TABLE brands (

\_id Text,

barcode Text,

category Varchar(50),

categoryCode Varchar(50),

name Varchar(150),

topBrand Varchar(50),

brandCode Varchar(50),

cpg\_ref Text,

cpg\_id Text,

PRIMARY KEY (\_id),

UNIQUE (barcode)

);

CREATE TABLE receipts\_fact\_table (

user\_id Text,

brand\_id Text,

receipt\_id Text,

barcode Text,

PRIMARY KEY (user\_id, brand\_id, receipt\_id, barcode),

CONSTRAINT FK\_Receipts\_Fact\_table\_Brand\_id

FOREIGN KEY (brand\_id)

REFERENCES brands(\_id),

CONSTRAINT FK\_Receipts\_Fact\_table\_Barcode

FOREIGN KEY (barcode)

REFERENCES brands(barcode),

CONSTRAINT FK\_Receipts\_Fact\_table\_Receipt\_id

FOREIGN KEY (receipt\_id)

REFERENCES receipts(\_id),

CONSTRAINT FK\_Receipts\_Fact\_table\_User\_id

FOREIGN KEY (user\_id)

REFERENCES users(\_id)

);

CREATE TABLE receipt\_Items (

receipt\_id Text,

barcode Text,

description Varchar(250),

finalPrice Float,

itemPrice Float,

partnerItemId float,

quantityPurchased float(2),

discountedItemPrice Float,

originalReceiptItemText Varchar(250),

PRIMARY KEY (receipt\_id, barcode),

CONSTRAINT FK\_Receipt\_Items\_Receipt\_id

FOREIGN KEY (receipt\_id)

REFERENCES receipts(\_id),

CONSTRAINT FK\_Receipt\_Items\_barcode

FOREIGN KEY (barcode)

REFERENCES brands(barcode)

);

**PYTHON SCRIPT TO CLEAN & TRANSFORM THE JSON FILES**

#importing neccessary libraries

import json

import pandas as pd

import numpy as np

from datetime import datetime

#reading data with pandas

receipts = pd.read\_json('receipts.json', lines = True)

brands = pd.read\_json('brands.json', lines = True)

users = pd.read\_json('users.json', lines = True)

# Python script for cleaning and transforming users data

def clean\_id(x):

try:

return x['$oid']

except:

return None

def clean\_createdDate(x):

try:

return x['$date']

except:

return None

def clean\_lastLogin(x):

try:

if isinstance(x, dict):

return x['$date']

else:

return x

except:

return None

def transform\_date(timestamp):

try:

return datetime.fromtimestamp(timestamp/1000)

except:

return None

users['\_id'] = users['\_id'].apply(clean\_id)

users['createdDate'] = users['createdDate'].apply(clean\_createdDate)

users['lastLogin'] = users['lastLogin'].apply(clean\_lastLogin)

users['createdDate'] = users['createdDate'].apply(transform\_date)

users['lastLogin'] = users['lastLogin'].apply(transform\_date)

# Python script for cleaning and transforming brands data

def clean\_id(x):

try:

if isinstance(x, dict):

return x['$oid']

else:

return x

except:

return None

def clean\_cpg(x):

try:

if isinstance(x, dict):

return x['$ref']

else:

return x

except:

return None

def clean\_cpg\_id(x):

try:

if isinstance(x, dict):

y = x['$id']

if isinstance(y, dict):

return y['$oid']

else:

return y

else:

return x

except:

return None

brands['\_id'] = brands['\_id'].apply(clean\_id)

brands['cpg\_ref'] = brands['cpg'].apply(clean\_cpg)

brands['cpg\_id'] = brands['cpg'].apply(clean\_cpg\_id)

brands = brands.drop(['cpg'], axis = 1)

# Python script for cleaning and transforming receipts data

def clean\_id(x):

try:

if isinstance(x, dict):

return x['$oid']

else:

return x

except:

return None

def clean\_date(x):

try:

if isinstance(x, dict):

return x['$date']

else:

return x

except:

return None

def transform\_date(timestamp):

try:

return datetime.fromtimestamp(timestamp/1000)

except:

return None

receipts['\_id'] = receipts['\_id'].apply(clean\_id)

receipts['createDate'] = receipts['createDate'].apply(clean\_date)

receipts['dateScanned'] = receipts['dateScanned'].apply(clean\_date)

receipts['finishedDate'] = receipts['finishedDate'].apply(clean\_date)

receipts['modifyDate'] = receipts['modifyDate'].apply(clean\_date)

receipts['pointsAwardedDate'] = receipts['pointsAwardedDate'].apply(clean\_date)

receipts['pointsAwardedDate'] = receipts['pointsAwardedDate'].apply(clean\_date)

receipts['purchaseDate'] = receipts['purchaseDate'].apply(clean\_date)

receipts.createDate = receipts.createDate.apply(transform\_date)

receipts.dateScanned = receipts.dateScanned.apply(transform\_date)

receipts.finishedDate = receipts.finishedDate.apply(transform\_date)

receipts.modifyDate = receipts.modifyDate.apply(transform\_date)

receipts.pointsAwardedDate = receipts.pointsAwardedDate.apply(transform\_date)

receipts.purchaseDate = receipts.purchaseDate.apply(transform\_date)

# Python script for breaking receipts data into receipts and receipts\_items

df\_receipts = receipts[['\_id', 'rewardsReceiptItemList']]

df\_receipts = df\_receipts.dropna()

df\_receipts = df\_receipts.reset\_index()

df\_receipts = df\_receipts.drop(['index'], axis = 1)

df\_receipts.tail()

x = pd.DataFrame()

for i in range(len(df\_receipts)):

temp = pd.DataFrame(df\_receipts.rewardsReceiptItemList.loc[i])

temp['\_id'] = df\_receipts.\_id.loc[i]

x = pd.concat([x, temp])

receipt\_items = x

receipt\_items = receipt\_items[['\_id', 'barcode', 'description', 'finalPrice', 'itemPrice', 'partnerItemId', 'quantityPurchased', 'discountedItemPrice', 'originalReceiptItemText']]

receipts = receipts.drop(['rewardsReceiptItemList'], axis = 1)

#python to covert from python dataframe to CSV files

receipt\_items.to\_csv(r'C:\Users\saint\Downloads\receipt\_items.csv', index = False, header = False)

receipts.to\_csv(r'C:\Users\saint\Downloads\receipts.csv', index = False, header = False)

users.to\_csv(r'C:\Users\saint\Downloads\users.csv', index = False, header = False)

brands.to\_csv(r'C:\Users\saint\Downloads\brands.csv', index = False, header = False)

**POSTGRESQL CODE TO LOAD CSV INTO DATA WAREHOUSE**

\COPY Users from C:\Users\saint\Downloads\users.csv CSV

\COPY Brands from C:\Users\saint\Downloads\brands.csv CSV

\COPY Receipts from C:\Users\saint\Downloads\receipts.csv CSV

\COPY Receipt\_Items from C:\Users\saint\Downloads\receipt\_items.csv CSV