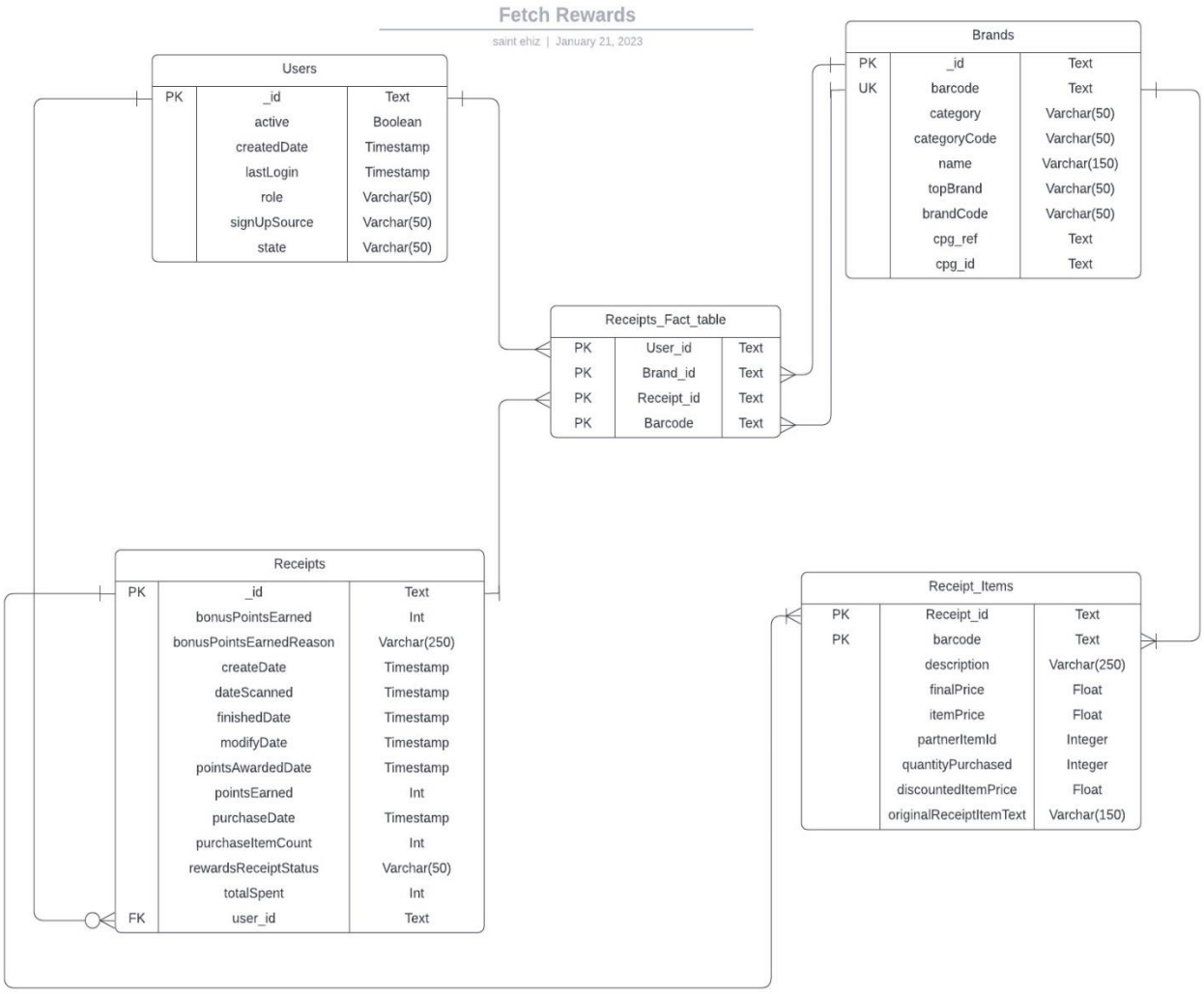

Fig 1: ERD to model data in a data warehouse.

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
```