In [630]:
```
# reading libraries

import pandas as pd
import json
```

In [635]:
```
# reading the 'brands' data
# Open the file and read lines

with open('/Users/projects/brands.json', 'r') as file:
    data = file.readlines()

# Parse JSON and convert to DataFrame
brands_df = pd.json_normalize([json.loads(line) for line in data])
```

In [ ]:
```
# renaming the columns for ease of use.

brands_df.rename(columns={'_id.$oid': 'brandId'}, inplace=True)
brands_df.rename(columns={'cpg.$id.$oid': 'cpgId'}, inplace=True)
brands_df.rename(columns={'cpg.$ref': 'cpgref'}, inplace=True)
```

In [636]:
```
brands_df.head(4)
```

Out[636]:

| | barcode | category | categoryCode | name | topBrand | _id. |
|---|---|---|---|---|---|---|
| 0 | 511111019862 | Baking | BAKING | test brand @1612366101024 | False | 601ac115be37ce2ead43 |
| 1 | 511111519928 | Beverages | BEVERAGES | Starbucks | False | 601c5460be37ce2ead43 |
| 2 | 511111819905 | Baking | BAKING | test brand @1612366146176 | False | 601ac142be37ce2ead43 |
| 3 | 511111519874 | Baking | BAKING | test brand @1612366146051 | False | 601ac142be37ce2ead43 |

In [549]:
```
# reading users data

with open('/Users/projects/users.json', 'r') as file:
    data = file.readlines()

# Parse JSON and convert to DataFrame
users_df = pd.json_normalize([json.loads(line) for line in data])

# formatting the dates to a standard format

users_df['lastLogin.$date']   = pd.to_datetime(users_df['lastLogin.$da
users_df['createdDate.$date'] = pd.to_datetime(users_df['createdDate.$


# renaming the columns
users_df.rename(columns={'_id.$oid': 'userId'}, inplace=True)
users_df.rename(columns={'createdDate.$date': 'createdDate'}, inplace=
users_df.rename(columns={'lastLogin.$date': 'lastLogin'}, inplace=True
```

In [553]:
```
# got to drop the duplicate values before we load the data into the Us
# we are using 'userId' as primary key and since it contains duplicate

users_df = users_df.drop_duplicates(subset=['user_id'])
```

In [590]:
```
#Analyzing receipts data.
#Upon inspection, we noticed that the 'rewardsReceiptItemList' column
#We'll separate the nested data, normalize it, and then merge it back

import pandas as pd
import json
from pandas import json_normalize

# Initialize empty lists to store the data
main_data = []
rewards_data = []

# Read JSON data from file line by line
with open('/Users/projects/receipts.json', 'r') as file:
    for line in file:
        # Load JSON data from each line
        data = json.loads(line)

        # Check if 'rewardsReceiptItemList' key exists
        if 'rewardsReceiptItemList' in data:
            # If key exists, add to rewards_data list
            rewards_data.extend(data['rewardsReceiptItemList'])
            # Remove 'rewardsReceiptItemList' key from data
            del data['rewardsReceiptItemList']

        # Append the remaining data to main_data list
        main_data.append(data)

# Create DataFrame for main data
df_main = pd.json_normalize(main_data)

# Create DataFrame for rewards data
df_rewards = pd.json_normalize(rewards_data)

# Merge DataFrames
receipts_df = pd.merge(df_main, df_rewards, left_index=True, right_ind
```

In [592]:
```
#formating the dates

receipts_df['dateScanned.$date']         =  pd.to_datetime(receipts_df
receipts_df['createDate.$date']          =  pd.to_datetime(receipts_df
receipts_df['finishedDate.$date']        =  pd.to_datetime(receipts_df
receipts_df['modifyDate.$date']          =  pd.to_datetime(receipts_df
receipts_df['pointsAwardedDate.$date']   =  pd.to_datetime(receipts_df
receipts_df['purchaseDate.$date']        =  pd.to_datetime(receipts_df

# renaming the columns

receipts_df.rename(columns={'_id.$oid':'receipts_id'}, inplace=True)
receipts_df.rename(columns={'dateScanned.$date':'dateScanned'}, inplac
receipts_df.rename(columns={'createDate.$date':'createDate'}, inplace=
receipts_df.rename(columns={'finishedDate.$date':'finishedDate'}, inpl
receipts_df.rename(columns={'modifyDate.$date':'modifyDate'}, inplace=
receipts_df.rename(columns={'pointsAwardedDate.$date':'pointsAwardedDa
receipts_df.rename(columns={'purchaseDate.$date':'purchaseDate'}, inpl
```

In [594]:
```
# creating the dataframe rewardreceipts. We will be using this a table

df_rewardreceipts = pd.DataFrame(receipts_df, columns = ['userId','rec
```

In [595]:
```
columns_to_delete = [
    'description',
    'finalPrice',
    'itemPrice',
    'needsFetchReview',
    'partnerItemId',
    'preventTargetGapPoints',
    'quantityPurchased',
    'userFlaggedBarcode',
    'userFlaggedNewItem',
    'userFlaggedPrice',
    'userFlaggedQuantity',
    'needsFetchReviewReason',
    'pointsNotAwardedReason',
    'pointsPayerId',
    'rewardsGroup',
    'rewardsProductPartnerId',
    'userFlaggedDescription',
    'originalMetaBriteBarcode',
    'originalMetaBriteDescription',
    'brandCode',
    'competitorRewardsGroup',
    'discountedItemPrice',
    'originalReceiptItemText',
    'itemNumber',
    'originalMetaBriteQuantityPurchased',
    'pointsEarned',
    'targetPrice',
    'competitiveProduct',
    'originalFinalPrice',
    'originalMetaBriteItemPrice',
    'deleted',
    'priceAfterCoupon',
    'metabriteCampaignId']

receipts_df = receipts_df.drop(columns=columns_to_delete)
```

In [596]:
```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

In [637]:
```
receipts_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1119 entries, 0 to 1118
Data columns (total 15 columns):
 #   Column                    Non-Null Count  Dtype         
---  ------                    --------------  -----         
 0   bonusPointsEarned         544 non-null    float64       
 1   bonusPointsEarnedReason   544 non-null    object        
 2   purchasedItemCount        635 non-null    float64       
 3   rewardsReceiptStatus      1119 non-null   object        
 4   totalSpent                684 non-null    object        
 5   userId                    1119 non-null   object        
 6   receipts_id               1119 non-null   object        
 7   createDate                1119 non-null   datetime64[ns]
 8   dateScanned               1119 non-null   datetime64[ns]
 9   finishedDate              568 non-null    datetime64[ns]
 10  modifyDate                1119 non-null   datetime64[ns]
 11  pointsAwardedDate         537 non-null    datetime64[ns]
 12  purchaseDate              671 non-null    datetime64[ns]
 13  barcode                   775 non-null    object        
 14  pointsEarned_receiptItem  286 non-null    object        
dtypes: datetime64[ns](6), float64(2), object(7)
memory usage: 139.9+ KB
```

In [393]:
```
# we shall begin defining the sql scripting part.
# We have 5 tables in total. We begin by defining 'Brands' table
# We are using sqlite3 services for this process
# 'fetch.db' is the database
```

In [603]:
```
# Connect to the SQLite database
conn = sqlite3.connect('fetch.db')
cursor = conn.cursor()

# Execute the SQL statement to drop the table
cursor.execute("DROP TABLE IF EXISTS Receipts")

# Commit the transaction to save the changes
conn.commit()

# Close the database connection
conn.close()
```

In [ ]:
```
# Connect to the SQLite database
conn = sqlite3.connect('fetch.db')
cursor = conn.cursor()

# Execute the SQL statement to drop the table
cursor.execute("DROP TABLE IF EXISTS Users")

# Commit the transaction to save the changes
conn.commit()

# Close the database connection
conn.close()
```

In [564]:
```
# Connect to the SQLite database
conn = sqlite3.connect('fetch.db')
cursor = conn.cursor()

# Execute the SQL statement to drop the table
cursor.execute("DROP TABLE IF EXISTS Brands")

# Commit the transaction to save the changes
conn.commit()

# Close the database connection
conn.close()
```

In [569]:
```
conn = sqlite3.connect('fetch.db')
cursor = conn.cursor()

# Create Users table with brandId as primary key
cursor.execute('''
    CREATE TABLE Brands (
    barcode TEXT ,
    category TEXT,
    categoryCode TEXT,
    name TEXT NOT NULL,
    topBrand TEXT,
    brandId TEXT NOT NULL PRIMARY KEY,
    cpgId TEXT,
    cpgref TEXT,
    brandCode TEXT,
    extracted_id TEXT,
    FOREIGN KEY (cpgId) REFERENCES CPGs(cpgId)
    )
''')

# Commit changes and close connection
conn.commit()
conn.close()
```

In [ ]:
```
brands_df_two  = brands_df.drop_duplicates(subset=['brand_id'])
```

In [570]:
```
conn = sqlite3.connect('fetch.db')

# Assuming users_df is your DataFrame containing the Users table data
# Replace 'your_table_name' with the actual table name
brands_df.to_sql('Brands', conn, if_exists='append', index=False)

# Commit the transaction to save the changes
conn.commit()

# Close the database connection
conn.close()
```

In [615]:
```
conn = sqlite3.connect('fetch.db')
cursor = conn.cursor()

# Create Users table with user_id as primary key
cursor.execute('''
    CREATE TABLE IF NOT EXISTS Users (
    userId TEXT PRIMARY KEY,
    state TEXT,
    createdDate DATE,
    lastLogin DATE,
    role TEXT,
    active BOOLEAN,
    signupSource TEXT
  )
''')

# Commit changes and close connection
conn.commit()
conn.close()
```

In [621]:
```
conn = sqlite3.connect('fetch.db')
cursor = conn.cursor()

# Create Users table with user_id as primary key
cursor.execute('''
    CREATE TABLE Receipts (
    userId TEXT,
    receipts_id TEXT PRIMARY KEY,
    bonusPointsEarned FLOAT,
    bonusPointsEarnedReason TEXT,
    purchasedItemCount FLOAT,
    rewardsReceiptStatus TEXT,
    barcode TEXT,
    totalSpent FLOAT,
    createDate TIMESTAMP,
    dateScanned TIMESTAMP,
    finishedDate TIMESTAMP,
    modifyDate TIMESTAMP,
    pointsAwardedDate TIMESTAMP,
    purchaseDate TIMESTAMP,
    pointsEarned_receiptItem TEXT,
    FOREIGN KEY (userId) REFERENCES Users(userId)
)
''')

# Commit changes and close connection
conn.commit()
conn.close()
```

In [622]:
```
receipts_df = receipts_df.drop_duplicates(subset=['receipts_id'])
```

In [623]:
```
conn = sqlite3.connect('fetch.db')

# Assuming users_df is your DataFrame containing the Users table data
# Replace 'your_table_name' with the actual table name
receipts_df.to_sql('Receipts', conn, if_exists='append', index=False)

# Commit the transaction to save the changes
conn.commit()

# Close the database connection
conn.close()
```

In [ ]:
```
users_df = users_df.drop_duplicates(subset=['userId'])
```

In [616]:
```
conn = sqlite3.connect('fetch.db')

# Assuming users_df is your DataFrame containing the Users table data
# Replace 'your_table_name' with the actual table name
users_df.to_sql('Users', conn, if_exists='append', index=False)

# Commit the transaction to save the changes
conn.commit()

# Close the database connection
conn.close()
```

In [586]:
```
conn = sqlite3.connect('fetch.db')

# Define the SQL query to select all data from the users table
sql_query = "SELECT * FROM Receipts"

# Read data from the SQLite database into a DataFrame
brands_ = pd.read_sql_query(sql_query, conn)

# Close the database connection
conn.close()

# Display the DataFrame
```

Out[586]:

| | userId | receipts_id | bonusPointsEarned | bonusPointsEarn |
|---|---|---|---|---|
| **0** | 5ff1e1eacfcf6c399c274ae6 | 5ff1e1eb0a720f0523000575 | 500.0 | Receipt completed, b |

| | | | | |
|---|---|---|---|---|
| **1** | 5ff1e194b6a9d73a3a9f1052 | 5ff1e1bb0a720f052300056b | 150.0 | Receipt completed, b |
| **2** | 5ff1e1f1cfcf6c399c274b0b | 5ff1e1f10a720f052300057a | 5.0 | All-receipts rec |
| **3** | 5ff1e1eacfcf6c399c274ae6 | 5ff1e1ee0a7214ada100056f | 5.0 | All-receipts rec |
| **4** | 5ff1e194b6a9d73a3a9f1052 | 5ff1e1d20a7214ada1000561 | 5.0 | All-receipts rec |
| **5** | 5ff1e1e4cfcf6c399c274ac3 | 5ff1e1e40a7214ada1000566 | 750.0 | Receipt completed, b |
| **6** | 5ff1e194b6a9d73a3a9f1052 | 5ff1e1cd0a720f052300056f | 5.0 | All-receipts rec |
| **7** | 5ff1e194b6a9d73a3a9f1052 | 5ff1e1a40a720f0523000569 | 500.0 | Receipt completed, b |
| **8** | 5ff1e1eacfcf6c399c274ae6 | 5ff1e1ed0a7214ada100056e | 5.0 | All-receipts rec |
| **9** | 5ff1e1eacfcf6c399c274ae6 | 5ff1e1eb0a7214ada100056b | 250.0 | Receipt completed, b |
| **10** | 5ff1e194b6a9d73a3a9f1052 | 5ff1e1c50a720f052300056c | 100.0 | Receipt completed, b |
| **11** | 5ff1e194b6a9d73a3a9f1052 | 5ff1e1a10a720f0523000568 | 750.0 | Receipt completed, b |
| **12** | 5ff1e194b6a9d73a3a9f1052 | 5ff1e1b60a7214ada100055c | 150.0 | Receipt completed, b |
| **13** | 5f9c74f7c88c1415cbddb839 | 5f9c74f70a7214ad07000037 | 750.0 | Receipt completed, b |
| **14** | 5ff1e194b6a9d73a3a9f1052 | 5ff1e1b20a7214ada100055a | 300.0 | Receipt completed, b |
| **15** | 5ff1e1e9b6a9d73a3a9f10f6 | 5ff1e1e90a7214ada1000569 | NaN | |
| **16** | 5ff1e1dfcfcf6c399c274ab3 | 5ff1e1df0a7214ada1000564 | 750.0 | Receipt completed, b |
| **17** | 5ff1e1b4cfcf6c399c274a54 | 5ff1e1b40a7214ada100055b | 750.0 | Receipt completed, b |
| | | | | Receipt |

| | | | | |
|---|---|---|---|---|
| **18** | 5ff1e1eacfcf6c399c274ae6 | 5ff1e1eb0a720f0523000576 | 300.0 | completed, b |
| **19** | 5ff1e194b6a9d73a3a9f1052 | 5ff1e1c80a720f052300056d | 5.0 | All-receipts rec |

In [632]:
```
conn = sqlite3.connect('fetch.db')
cursor = conn.cursor()

# Drop a table (example: Users)
cursor.execute('DROP TABLE IF EXISTS receipt_items')

# Commit changes and close connection
conn.commit()
conn.close()
```

In [633]:
```
conn = sqlite3.connect('fetch.db')
cursor = conn.cursor()

# Define the SQL statement to create the table
create_table_query = '''
CREATE TABLE receipt_items (
    receipts_id TEXT PRIMARY KEY,
    userId TEXT,
    barcode TEXT,
    description TEXT,
    finalPrice NUMERIC,
    itemPrice NUMERIC,
    needsFetchReview TEXT,
    partnerItemId TEXT,
    preventTargetGapPoints TEXT,
    quantityPurchased NUMERIC,
    userFlaggedBarcode TEXT,
    userFlaggedNewItem TEXT,
    userFlaggedPrice TEXT,
    userFlaggedQuantity NUMERIC,
    needsFetchReviewReason TEXT,
    pointsNotAwardedReason TEXT,
    pointsPayerId TEXT,
    rewardsGroup TEXT,
    rewardsProductPartnerId TEXT,
    userFlaggedDescription TEXT,
    originalMetaBriteBarcode TEXT,
    originalMetaBriteDescription TEXT,
    brandCode TEXT,
    competitorRewardsGroup TEXT,
    discountedItemPrice NUMERIC,
    originalReceiptItemText TEXT,
    itemNumber TEXT,
```

```
    originalMetaBriteQuantityPurchased NUMERIC,
    pointsEarned NUMERIC,
    targetPrice NUMERIC,
    competitiveProduct TEXT,
    originalFinalPrice NUMERIC,
    originalMetaBriteItemPrice NUMERIC,
    deleted TEXT,
    priceAfterCoupon NUMERIC,
    metabriteCampaignId TEXT,
    FOREIGN KEY (userId) REFERENCES Users(userId)
);
'''

# Execute the SQL statement to create the table
cursor.execute(create_table_query)

conn.commit()
conn.close()
```

In [634]:
```
conn = sqlite3.connect('fetch.db')

# Assuming users_df is your DataFrame containing the Users table data
# Replace 'your_table_name' with the actual table name
df_rewardreceipts.to_sql('receipt_items', conn, if_exists='append', in

# Commit the transaction to save the changes
conn.commit()

# Close the database connection
conn.close()
```

In [628]:
```
conn = sqlite3.connect('fetch.db')

# Define the SQL query to select all data from the users table
sql_query = "SELECT * FROM receipt_items"

# Read data from the SQLite database into a DataFrame
users_display = pd.read_sql_query(sql_query, conn)

# Close the database connection
conn.close()

# Display the DataFrame
users_display.head(1)
```

Out[628]:

| | receipts_id | userId | barcode | description | finalPrice | itemPri |
|---|---|---|---|---|---|---|
| **0** | 5ff1e1eb0a720f0523000575 | 5ff1e1eacfcf6c399c274ae6 | 4011 | ITEM NOT FOUND | 26.0 | 26 |

In [ ]:
```
#. Answering a couple of questions.
```

In [ ]:
```
# When considering average spend from receipts with 'rewardsReceiptSta
```

In [638]:
```
conn = sqlite3.connect('fetch.db')

# Query to calculate average spend for 'Accepted' receipts
query_accepted = "SELECT AVG(totalSpent) AS avg_accepted_spend FROM Re

# Query to calculate average spend for 'Rejected' receipts
query_rejected = "SELECT AVG(totalSpent) AS avg_rejected_spend FROM Re

# Execute queries and fetch results using Pandas
df_accepted = pd.read_sql_query(query_accepted, conn)
df_rejected = pd.read_sql_query(query_rejected, conn)

# Close database connection
conn.close()

# Display results
print("Average spend from receipts with 'Accepted' status:", df_accept
print("Average spend from receipts with 'Rejected' status:", df_reject
```

```
Average spend from receipts with 'Accepted' status: 80.85430501930502
Average spend from receipts with 'Rejected' status: 23.32605633802818
4
```

In [ ]:
```
#When considering total number of items purchased from receipts with '
```

In [642]:
```
conn = sqlite3.connect('fetch.db')


query_accepted = "SELECT SUM(purchasedItemCount) AS total_items_accept

# Query to calculate total number of items purchased for 'Rejected' re
query_rejected = "SELECT SUM(purchasedItemCount) AS total_items_reject

# Execute queries and fetch results
cursor = conn.cursor()
cursor.execute(query_accepted)
total_items_accepted = cursor.fetchone()[0]

cursor.execute(query_rejected)
total_items_rejected = cursor.fetchone()[0]

print("Average spend from receipts with 'Accepted' status:", total_ite
print("Average spend from receipts with 'Accepted' status:", total_ite
```

```
Average spend from receipts with 'Accepted' status: 8184.0
Average spend from receipts with 'Accepted' status: 173.0
```

In [ ]:
```
# Which brand has the most spend among users who were created within t
```

In [ ]:
```
conn = sqlite3.connect('fetch.db')

query_accepted = '''
                SELECT b.name AS brand_name, SUM(r.totalSpent) AS tota
                FROM Users u
                JOIN Receipts r ON u.userId = r.userId
                JOIN receipt_items ri ON r.receiptId = ri.receiptId
                JOIN Brands b ON ri.brandCode = b.brandCode
                WHERE u.createdDate >= DATE_SUB(CURRENT_DATE(), INTERV
                GROUP BY b.name
                ORDER BY total_spend DESC
                LIMIT 1;'''
```

In [ ]:
```
# unfortunately I could not get the desired output as the relation bet
```