In [ ]:
```
import json
import pandas as pd
import numpy as np
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
```

Data Conversion

Users

In [ ]:
```
# Read the JSON file
with open('users.json', 'r') as file:
    json_data = [json.loads(line) for line in file]

# Convert JSON data to DataFrame
users = pd.json_normalize(json_data)

# Flatten nested JSON fields
users['_id'] = users['_id.$oid']
users['createdDate'] = pd.to_datetime(users['createdDate.$date'], unit='ms')
if 'lastLogin.$date' in users.columns:
    users['lastLogin'] = pd.to_datetime(users['lastLogin.$date'], unit='ms')

# Drop original nested columns
users.drop(columns=['_id.$oid', 'createdDate.$date', 'lastLogin.$date'], inplace=True,

users['lastLogin'].replace('', pd.NA, inplace=True)

# Save the updated DataFrame back to a CSV file
users.to_csv('users.csv', index=False, na_rep='NULL')

print("Users JSON data has been converted to CSV successfully.")
```

```
Users JSON data has been converted to CSV successfully.
```

In [ ]:
```
users.head()
```

Out[ ]:

	active	role	signUpSource	state	_id	createdDate	lastLogin
0	True	consumer	Email	WI	5ff1e194b6a9d73a3a9f1052	2021-01-03 15:24:04.800	2021-01-03 15:25:37.857999872
1	True	consumer	Email	WI	5ff1e194b6a9d73a3a9f1052	2021-01-03 15:24:04.800	2021-01-03 15:25:37.857999872
2	True	consumer	Email	WI	5ff1e194b6a9d73a3a9f1052	2021-01-03 15:24:04.800	2021-01-03 15:25:37.857999872
3	True	consumer	Email	WI	5ff1e1eacfcf6c399c274ae6	2021-01-03 15:25:30.554	2021-01-03 15:25:30.596999936
4	True	consumer	Email	WI	5ff1e194b6a9d73a3a9f1052	2021-01-03 15:24:04.800	2021-01-03 15:25:37.857999872

Brands

In [ ]:

```
# Read the JSON file
with open('brands.json', 'r') as file:
    json_data = [json.loads(line) for line in file]

# Convert JSON data to DataFrame
brands = pd.json_normalize(json_data)

# Flatten nested JSON fields
brands['_id'] = brands['_id.$oid']
brands['cpg_id'] = brands['cpg.$id.$oid']
brands['cpg_ref'] = brands['cpg.$ref']

# Drop original nested columns
brands.drop(columns=['_id.$oid', 'cpg.$id.$oid', 'cpg.$ref'], inplace=True, errors='ig

# Save DataFrame to CSV
brands.to_csv('brands.csv', index=False)

print("Brands JSON data has been converted to CSV successfully.")
```

```
Brands JSON data has been converted to CSV successfully.
```

In [ ]:

```
brands.head()
```

Out[ ]:

	barcode	category	categoryCode	name	topBrand	brandCode	
0	511111019862	Baking	BAKING	test brand @1612366101024	False	NaN	601ac
1	511111519928	Beverages	BEVERAGES	Starbucks	False	STARBUCKS	601c!
2	511111819905	Baking	BAKING	test brand @1612366146176	False	TEST BRANDCODE @1612366146176	601ac
3	511111519874	Baking	BAKING	test brand @1612366146051	False	TEST BRANDCODE @1612366146051	601ac
4	511111319917	Candy & Sweets	CANDY_AND_SWEETS	test brand @1612366146827	False	TEST BRANDCODE @1612366146827	601ac

Receipts

In [ ]:

```
def read_json(filename: str) -> list:
    with open(filename, "r") as f:
        try:
            # Attempt to load the entire file content as a list of JSON objects
            content = f.read()
            data_list = [json.loads(obj) for obj in content.split('\n') if obj.strip()
            return data_list
        except json.JSONDecodeError as e:
            raise Exception(f"Reading {filename} file encountered an error: {e}")

# Read the JSON file
try:
    data_list = read_json("receipts.json")
    print("Data Size:", len(data_list))
except Exception as e:
    print(e)
```

```
Data Size: 1119
```

In [ ]:

```
def normalize_json(data: dict) -> dict:
    new_data = {}
    for key, value in data.items():
        if not isinstance(value, dict):
            new_data[key] = value
        else:
            for k, v in value.items():
                new_data[key + "_" + k] = v
    return new_data

# Normalize each JSON object
normalized_data = [normalize_json(data) for data in data_list]

# Create a pandas dataframe from the list of normalized JSON objects
receipts = pd.DataFrame(normalized_data)
```

```
# Write to a CSV file
receipts.to_csv("receipts.csv", index=False)
```

In [ ]:
```
receipts.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1119 entries, 0 to 1118
Data columns (total 15 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   _id_$oid                 1119 non-null   object 
 1   bonusPointsEarned        544 non-null    float64
 2   bonusPointsEarnedReason  544 non-null    object 
 3   createDate_$date         1119 non-null   int64  
 4   dateScanned_$date        1119 non-null   int64  
 5   finishedDate_$date       568 non-null    float64
 6   modifyDate_$date         1119 non-null   int64  
 7   pointsAwardedDate_$date  537 non-null    float64
 8   pointsEarned             609 non-null    object 
 9   purchaseDate_$date       671 non-null    float64
 10  purchasedItemCount       635 non-null    float64
 11  rewardsReceiptItemList   679 non-null    object 
 12  rewardsReceiptStatus     1119 non-null   object 
 13  totalSpent               684 non-null    object 
 14  userId                   1119 non-null   object 
dtypes: float64(5), int64(3), object(7)
memory usage: 131.3+ KB
```

In [ ]:
```
receipts.describe()
```

Out[ ]:

	bonusPointsEarned	createDate_$date	dateScanned_$date	finishedDate_$date	modifyDate_$dat
count	544.000000	1.119000e+03	1.119000e+03	5.680000e+02	1.119000e+0
mean	238.893382	1.611800e+12	1.611800e+12	1.611058e+12	1.611847e+1
std	299.091731	1.484091e+09	1.484091e+09	9.534641e+08	1.361576e+0
min	5.000000	1.604089e+12	1.604089e+12	1.609687e+12	1.609687e+1
25%	5.000000	1.610652e+12	1.610652e+12	1.610141e+12	1.610660e+1
50%	45.000000	1.611941e+12	1.611941e+12	1.611091e+12	1.611941e+1
75%	500.000000	1.612704e+12	1.612704e+12	1.611769e+12	1.612704e+1
max	750.000000	1.614641e+12	1.614641e+12	1.614379e+12	1.614641e+1

In [ ]:
```
def parse_rewards_receipt_item_list(item):
    if isinstance(item, list):
        return item  # Item is already a list
    if isinstance(item, float):
        return []  # Return an empty list for NaN values
    try:
        return json.loads(item.replace("'", '"'))
    except (json.JSONDecodeError, TypeError):
        return []
```

```
# Create a copy of df with only the necessary columns
df_subset = receipts[['_id_$oid', 'userId', 'rewardsReceiptItemList']].copy()

# Explode the rewardsReceiptItemList to rows
df_subset_exploded = df_subset.explode('rewardsReceiptItemList').dropna()

# Normalize the rewardsReceiptItemList
items = pd.json_normalize(df_subset_exploded['rewardsReceiptItemList'])

# Merge userId and _id_$oid into the normalized items
items = items.join(df_subset_exploded[['_id_$oid', 'userId']].reset_index(drop=True))

# Display the first few rows of the resulting DataFrame
items.head()
```

Out[ ]:

	barcode	description	finalPrice	itemPrice	needsFetchReview	partnerItemId	preventTargetGa
0	4011	ITEM NOT FOUND	26.00	26.00	False	1	
1	4011	ITEM NOT FOUND	1	1	NaN	1	
2	028400642255	DORITOS TORTILLA CHIP SPICY SWEET CHILI REDUCE...	10.00	10.00	True	2	
3	NaN	NaN	NaN	NaN	False	1	
4	4011	ITEM NOT FOUND	28.00	28.00	False	1	

In [ ]:
```
items.to_csv('items.csv', index=False)
```

In [ ]:
```
items.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6941 entries, 0 to 6940
Data columns (total 36 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   barcode                               3090 non-null   object 
 1   description                           6560 non-null   object 
 2   finalPrice                            6767 non-null   object 
 3   itemPrice                             6767 non-null   object 
 4   needsFetchReview                      813 non-null    object 
 5   partnerItemId                         6941 non-null   object 
 6   preventTargetGapPoints                358 non-null    object 
 7   quantityPurchased                     6767 non-null   float64
 8   userFlaggedBarcode                    337 non-null    object 
 9   userFlaggedNewItem                    323 non-null    object 
 10  userFlaggedPrice                      299 non-null    object 
 11  userFlaggedQuantity                   299 non-null    float64
 12  needsFetchReviewReason                219 non-null    object 
 13  pointsNotAwardedReason                340 non-null    object 
 14  pointsPayerId                         1267 non-null   object 
 15  rewardsGroup                          1731 non-null   object 
 16  rewardsProductPartnerId               2269 non-null   object 
 17  userFlaggedDescription                205 non-null    object 
 18  originalMetaBriteBarcode              71 non-null     object 
 19  originalMetaBriteDescription          10 non-null     object 
 20  brandCode                             2600 non-null   object 
 21  competitorRewardsGroup                275 non-null    object 
 22  discountedItemPrice                   5769 non-null   object 
 23  originalReceiptItemText               5760 non-null   object 
 24  itemNumber                            153 non-null    object 
 25  originalMetaBriteQuantityPurchased    15 non-null     float64
 26  pointsEarned                          927 non-null    object 
 27  targetPrice                           378 non-null    object 
 28  competitiveProduct                    645 non-null    object 
 29  originalFinalPrice                    9 non-null      object 
 30  originalMetaBriteItemPrice            9 non-null      object 
 31  deleted                               9 non-null      object 
 32  priceAfterCoupon                      956 non-null    object 
 33  metabriteCampaignId                   863 non-null    object 
 34  _id_$oid                              6941 non-null   object 
 35  userId                                6941 non-null   object 
dtypes: float64(3), object(33)
memory usage: 1.9+ MB
```

In [ ]:
```
items.describe()
```

Out[ ]:

	quantityPurchased	userFlaggedQuantity	originalMetaBriteQuantityPurchased
count	6767.000000	299.000000	15.000000
mean	1.386139	1.872910	1.200000
std	1.204363	1.314823	0.414039
min	1.000000	1.000000	1.000000
25%	1.000000	1.000000	1.000000
50%	1.000000	1.000000	1.000000
75%	1.000000	3.000000	1.000000
max	17.000000	5.000000	2.000000

In [ ]:
```
brands['brandCode'].unique()
```

Out[ ]:
```
array([nan, 'STARBUCKS', 'TEST BRANDCODE @1612366146176',
       'TEST BRANDCODE @1612366146051', 'TEST BRANDCODE @1612366146827',
       'TEST BRANDCODE @1612366146091', 'TEST BRANDCODE @1612366146133',
       'J.L. KRAFT', 'CAMPBELLS HOME STYLE', 'TEST',
       'TEST BRANDCODE @1598813526777', 'CALUMET', '511111205012',
       'AUNT JEMIMA SYRUP', 'MOLSON', 'LOTRIMIN',
       'TEST BRANDCODE @1597342520277', 'ST IVES', 'CHRISIMAGE',
       'ALKA SELTZER', "JACK DANIEL'S BARBECUE", 'MAGNUM Ice Cream',
       '511111105329', 'TEST BRANDCODE @1598635634882', 'TACO BELL',
       'FROSTED CHEERIOS', 'TEST BRANDCODE @1598639199674',
       'GODIVA DRY PACKAGED DESSERTS', 'LARABAR',
       'TEST BRANDCODE @1597350074333', 'TEST BRANDCODE @1607636368717',
       'TEST BRANDCODE @1607707830095', 'COTTONELLE', 'IZZE', 'MIO',
       '511111505365', 'QUILTING SPECIAL EDITION',
       'TEST BRANDCODE @1604437351617', 'HERMAN', 'KEVITA', 'DELIMEX',
       'THE RIGHT TO SHOWER', 'CARESS', 'TEST BRANDCODE @1595340086044',
       'TEST BRANDCODE @1597861700968', 'TEST BRAND CODE',
       'TEST BRANDCODE @1598026275245', 'SWANSON',
       'TEST BRANDCODE @1598026274668', 'DASH-2249 1', '511111805854',
       'KETTLE BRAND', 'CAMPBELLS SOUP AT HAND ', 'RAINBO', 'CRISPIN',
       'TEST BRANDCODE @1607707830137', 'CLARITIN®', 'COOKIE CRISP',
       'TEST BRANDCODE @1605535020442', 'TEST BRANDCODE @1604946244750',
       'SNYDERS OF HANOVER', 'GOOD SEASONS',
       'TEST BRANDCODE @1598633602279', 'REVOLVER', 'KEYSTONE LIGHT',
       'BETTY CROCKER FRUIT ROLL UPS', 'TEST BRANDCODE @1607639232356',
       'MOMOFUKU', 'POPSICLE', 'MUG Root Beer', 'TOMS',
       'TEST BRANDCODE @1606765579244', 'TEST BRANDCODE @1597342520305',
       'PROMISE', 'OFF THE EATEN PATH', 'TEST BRANDCODE @1600291350125',
       'TEST BRANDCODE @1603760319758', "NANCY'S",
       'CAMPBELLS CHUNKY MAXX', 'FOOD NETWORK KITCHEN INSPIRATIONS',
       'TEST BRANDCODE @1597935986390', 'TEST BRANDCODE @1598633693744',
       'SABRITAS', 'GEVALIA KAFFE', 'TEST BRANDCODE @1598290603501',
       "O THAT'S GOOD", "CHESTER'S", "MISS VICKIE'S", "SIR KENSINGTON'S",
       'DORITOS', 'A.1.', 'TEST BRANDCODE @1598290603587', 'LOOZA JUICES',
       'ALEVE', 'SOFT-SHEEN CARSON - HAIR COLOR',
       'TEST BRANDCODE @1604946245499', '',
       'TEST BRANDCODE @1598639199117', 'PULL UPS',
       'TEST BRANDCODE @1605535020629', 'BRISK', 'CHRISXYZ', "BULL'S-EYE",
       'KLONDIKE', 'FRUTTARE', '511111305569', 'KNOX', '511111505716',
       'PLANTERS', 'DEGREE', "CALEB'S KOLA", 'LEMON LEMON', 'VASELINE',
       '511111005216', 'TEST BRANDCODE @1607312532178', 'PINESOL',
       'TEST BRANDCODE @1595340086791', '511111005148', 'ESSIE',
       '0987654321', 'CAMPBELLS WELL YES',
       'TEST BRANDCODE @1607312532304', 'TEST BRANDCODE @1598026274643',
       'TEST BRANDCODE @1601644881995', 'BEEF STEAK', "POND'S",
       'L’OREAL PARIS - COSMETICS', '511111605829',
       'TEST BRANDCODE @1595340086078', 'TEST BRANDCODE @1602694020787',
       'TEST BRANDCODE @1601939013649', 'SNACK FACTORY', 'MILLER64',
       "I CAN'T BELIEVE IT'S NOT BUTTER!", '511111805786',
       'TEST BRANDCODE @1610496647142', 'TEST BRANDCODE @1610496646289',
       'TEST BRANDCODE @1610496646345', 'TEST BRANDCODE @1610496646232',
       'TEST BRANDCODE @1610495466218', 'TEST BRANDCODE @1610495102462',
       'TEST BRANDCODE @1610495466250', 'KRAFT MIRACLE WHIP',
       'TEST BRANDCODE @1598042674173', 'TEST BRANDCODE @1598635634796',
       'SHEAF STOUT', 'CHEETOS', 'NATURES HARVEST', '511111605058',
       'TEST BRANDCODE @1598639198570', 'TEST BRANDCODE @1599097538609',
       'SCOTT', 'TEST BRANDCODE @1607639231674', 'EPIC', 'GROLSCH',
       'TEST BRANDCODE @1603760320407', 'MAGNUM',
       'TEST BRANDCODE @1603925787411', 'TEST BRANDCODE @1610495466861',
       'TEST BRANDCODE @1610495102526', 'TEST BRANDCODE @1610495102491',
```

'TEST BRANDCODE @1610495466175', 'TEST BRANDCODE @1610495103255',
'TEST BRANDCODE @1610495466285', 'TEST BRANDCODE @1610049748118',
'TEST BRANDCODE @1610049748181', 'TEST BRANDCODE @1610049748154',
'TEST BRANDCODE @1610049748724', 'TEST BRANDCODE @1610053719707',
'TEST BRANDCODE @1607987891932', 'TEST BRANDCODE @1607987891893',
'TEST BRANDCODE @1607987891852', 'TEST BRANDCODE @1607987892541',
'TEST BRANDCODE @1607707829962', "KILLIAN'S",
'TEST BRANDCODE @1597861700880', "KRAFT SHAKE 'N BAKE", 'HP',
'DRUMSTICK CEREAL', 'ALEVE D SINUS & COLD', 'CHEEZ WHIZ',
'TEST BRANDCODE @1607463685522', 'COOL WHIP', 'TRIX',
'FLINTSTONES', 'ABSOLUT® GRAPEFRUIT', 'ALTOS', 'DEL MAGUEY ',
'JAMESON® BLACK BARREL ', 'JAMESON', 'MONKEY 47', 'MALIBU',
'LILLET', '511111705161', 'AMP2', 'TEST BRANDCODE @1597861700940',
'TEST BRANDCODE @1604437351567', 'KELSEN', '511111405818',
'ALEVE DIRECT THERAPY PAIN RELIEF DEVICES', 'COLORADO NATIVE',
'SOBE', 'KNUDSEN', '511111105763', '511111005421', "CAP'N CRUNCH",
"MRS BAIRD'S", 'TRESEMME', 'PURE LEAF Iced Tea Beverages',
'ONE A DAY KIDS', 'TEST BRANDCODE @1599159968998', 'LIFEWTR',
'511111804994', 'SIMILAC', 'CORICIDIN HBP',
'TEST BRANDCODE @1601939016290', 'BADEDAS',
'TEST BRANDCODE @1604946244876', 'BENIHANA', 'MOUNTAIN HIGH',
'TOSTITOS', 'TEST BRANDCODE @1603760319788', 'GERBER BABY FOOD',
'PACE', 'CAMPBELLS SLOW KETTLE ', 'ARCHWAY', 'BAYER® ASPIRIN',
'TEST BRANDCODE @1610055793449', 'TEST BRANDCODE @1610055793411',
'TEST BRANDCODE @1610055793500', 'TEST BRANDCODE @1610055794196',
'TEST BRANDCODE @1610055793537', 'TEST BRANDCODE @1610058181549',
'TEST BRANDCODE @1610053719944', 'TEST BRANDCODE @1610053719998',
'TEST BRANDCODE @1610053719869', 'TEST BRANDCODE @1610053721033',
'DEPEND', 'TEST BRANDCODE @1608313322283',
'TEST BRANDCODE @1608313321374', 'TEST BRANDCODE @1608313321191',
'TEST BRANDCODE @1608313321301', 'TEST BRANDCODE @1608313321222',
'NATURE VALLEY', 'TEST BRANDCODE @1598633693767',
'TEST BRANDCODE @1598633693791', 'ANZIO', '511111705000',
"HERSHEY'S WHIPPED TOPPING", 'TEST BRANDCODE @1598633693720',
'TEST BRANDCODE @1605535020671', 'IMPERIAL', 'ALEVE PM NIGHT TIME',
'TEST BRANDCODE @1603925787294', 'APOTHECARE ESSENTIALS',
"HENRY WEINHARD'S", 'CAMPBELLS CHUNKY ', 'ONE A DAY® WOMENS',
'TEST BRANDCODE @1595340085980', 'KINGSFORD', 'YOPLAIT GO-GURT',
'FRITO-LAY', 'GREY POUPON', 'TEST BRANDCODE @1598296704763',
'BERROCA®', 'PROPEL', "MILWAUKEE'S BEST", 'SPITZ', 'IMAGINE',
'TEST BRANDCODE @1597342520969', 'TUSK & GRAIN', "REDD'S ALE",
'LANCE', 'PACIFIC FOODS', 'YOPLAIT DUNKERS',
'TEST BRANDCODE @1598554813705', 'LIPTON Soup', 'MINIONS CEREAL',
'LOVE HOME AND PLANET', 'TEST BRANDCODE @1595968943012',
'ONETOUCH', 'TEST BRANDCODE @1597935986248', 'KRAFT', 'BASIC 4',
'WOLFGANG PUCK', 'TEST BRANDCODE @1601659120828',
'TEST BRANDCODE @1603925787441', 'THE SOULFUL PROJECT',
'TEST BRANDCODE @1598633602924', 'DREAM WHIP',
'TEST BRANDCODE @1597527951461', 'TEST BRANDCODE @1607639231623',
'KJELDSEN', 'TEST BRANDCODE @1598635435896',
'TEST BRANDCODE @1599849713773', 'TEST BRANDCODE @1607636369405',
'TEST BRANDCODE @1601644363827', 'PEPPERIDGE FARM',
'TEST BRANDCODE @1611088503689', 'TEST BRANDCODE @1605535020580',
'MILLER HIGH LIFE', 'QUAKER', '511111705727', 'ATHENOS',
'TEST BRANDCODE @1599159969028', 'TEST BRANDCODE @1597350074404',
'SEDAL', 'RED ROCK DELI', 'ALKA SELTZER PLUS SINUS CAP/ GEL/ TAB',
'TEST BRANDCODE @1598633602253', 'SMARTFOOD',
'TEST BRANDCODE @1598042673532', 'TEST BRANDCODE @1601939762881',
'OSCAR MAYER', 'LECH PREMIUM', '511111805137', 'HEINERS',
'PARENTS', 'PIONEER WOMAN', 'TEST BRANDCODE @1611088504474',

'511111605102', 'HOP VALLEY', 'TEST BRANDCODE @1597861700914',
'YUBAN', '511111105114', 'TEST BRANDCODE @1603760319682',
'TEST BRANDCODE @1598716509394', 'TIGI', "AUTUMN'S GOLD",
'BROOKE BOND', '511111105046', 'TEST BRANDCODE @1595968943049',
'AMPTEST', 'COORS', 'BOLT 24', 'MIST TWST',
'TEST BRANDCODE @1608136484526', 'TEST BRANDCODE @1608136485398',
'TEST BRANDCODE @1608136484574', 'SARGENTO',
'TEST BRANDCODE @1608136484633', 'TEST BRANDCODE @1608136484689',
'TEST BRANDCODE @1610038521565', 'TEST BRANDCODE @1610038521677',
'TEST BRANDCODE @1610038521624', 'TEST BRANDCODE @1610038522559',
'TEST BRANDCODE @1610040576117', 'TEST BRANDCODE @1610040576672',
'TEST BRANDCODE @1610040576088', 'TEST BRANDCODE @1610040576148',
'TEST BRANDCODE @1610040576053', 'ALKA SELTZER PLUS',
'TEST BRANDCODE @1610653897113', 'TEST BRANDCODE @1610653894519',
'TEST BRANDCODE @1610660035659', 'TEST BRANDCODE @1610653894662',
'TEST BRANDCODE @1610653894371', 'TEST BRANDCODE @1601659122798',
'COCOA PUFFS', 'TEST BRANDCODE @1601659120793', 'JAYS', 'KNORR',
'CRISTAL', 'TEST BRANDCODE @1607312532874', 'BOCA', 'ALEXA', 'KIX',
'TEST BRANDCODE @1595531348246', "GRANDMA'S",
'TEST BRANDCODE @1602694015178', "SHARP'S NEAR BEER", 'BAILEYS',
'FINISH', 'TEST BRANDCODE @1598639197926',
'TEST BRANDCODE @1602694015743', 'JELL-O',
'TEST BRANDCODE @1601644364536', 'TEST BRANDCODE @1603925787997',
'GOLDFISH', 'TEST BRANDCODE @1598633602304',
'TEST BRANDCODE @1601659120694', 'CAPE LINE',
'TEST BRANDCODE @1597935987233', 'NEAR EAST', '511111505150',
'CLOROX', 'MAY-BUD', 'TEST BRANDCODE @1599849713798', 'GATORADE',
'TEST BRANDCODE @1597342520132', 'GARDEN FRESH GOURMET',
'AUNT JEMIMA DRY BREAKFAST MIXES', 'OLDE ENGLISH',
'TEST BRANDCODE @1600291349255', 'MATADOR', '511111305286',
"REESE'S PUFFS", 'GOLDEN GRAHAMS', 'TEST BRANDCODE @1598026274692',
'511111805342', 'TEST BRANDCODE @1598554813744', 'RICE-A-RONI',
'STELLA DORO', 'TEST BRANDCODE @1598813526686',
'TEST BRANDCODE @1604437351678', 'CAMPBELLS READY MEALS',
'RESOLVE', 'MAUI STYLE', 'CULTURE REPUBLICK ', 'WHEATIES',
'TEST BRANDCODE @1601644882062', 'TEST BRANDCODE @1601939013444',
'511111805571', 'TEST BRANDCODE @1598635634821',
'TEST BRANDCODE @1607707830844', 'ZIMA',
'TEST BRANDCODE @1598026274609', 'CHESTERS',
'TEST BRANDCODE @1601644363784', 'CAPE COD',
'COUNTRY CROCK PLANT BUTTER', '511111205388',
'CHOCOLATE LUCKY CHARMS', 'TEST BRANDCODE @1598554813535',
'TEST BRANDCODE @1610493496975', 'TEST BRANDCODE @1610495102411',
'TEST BRANDCODE @1610494831056', 'TEST BRANDCODE @1610494831082',
'TEST BRANDCODE @1610494831023', 'TEST BRANDCODE @1610493497005',
'TEST BRANDCODE @1610493497517', 'TEST BRANDCODE @1610493497034',
'TEST BRANDCODE @1610494830985', 'TEST BRANDCODE @1610494831583',
'TEST BRANDCODE @1608313051339', 'TEST BRANDCODE @1608313051291',
'TEST BRANDCODE @1608313052049', 'TEST BRANDCODE @1608313051133',
'TEST BRANDCODE @1608313051244', 'BALL PARK',
'TEST BRANDCODE @1598290603618', 'HIDDEL BARREL COLLECTION',
'LAUNCH BOX', 'SIMPLE', 'SALON SELECTIVES', 'CRACKER BARREL',
'HUGGIES LITTLE SWIMMERS', 'TEST BRANDCODE @1598042673502',
'TEST BRANDCODE @1607707830173', 'SHREDS', 'BAKEN ETS',
'HONEY NUT CHEERIOS', 'DASH-2249 1 BRAND 1', 'CRACKER JACK',
'GERBER GOOD START', 'TEST BRANDCODE @1598554813654',
'TEST BRANDCODE @1598635436505', 'TEST BRANDCODE @1597527951436',
'BETTER HOMES & GARDENS', 'BIMBO', "LEINENKUGEL'S", 'NESTLE NAN',
'TEST BRANDCODE @1598042673466', 'JET-PUFFED',
'TEST BRANDCODE @1601644882548', 'HEALTH',

'TEST BRANDCODE @1601644882019', 'TEST BRANDCODE @1601644363876',
'511111705444', 'MUNCHIES', '511111605775',
'TEST BRANDCODE @1607463684816', 'TEST BRANDCODE @1598633694347',
'TEST BRANDCODE @1598296704639', "ANNIE'S",
'TEST BRANDCODE @1598881562991', 'TEST BRANDCODE @1598813526656',
'MIDOL', 'TWISTED RANCH', 'EMERALD',
'TEST BRANDCODE @1598881562500', 'TEST BRANDCODE @1598711015538',
'HERSHEYS HOT COCOA', 'PG TIPS', 'MUNCHOS', 'LUNCHABLES',
'WOOLITE', '511111104971', 'HUGGIES',
'TEST BRANDCODE @1598711015353', 'TEST BRANDCODE @1605535021531',
'BISCA', 'TEST BRANDCODE @1598881563900',
'TEST BRANDCODE @1601939762909', 'PERONI',
'TEST BRANDCODE @1598554814581', 'TEST BRANDCODE @1607463684722',
'BETTY CROCKER GUSHERS', 'TEST BRANDCODE @1597861701615',
'LUCKY CHARMS', 'BACK TO NATURE DINNERS',
'TEST BRANDCODE @1597350074366', 'TEST BRANDCODE @1606765578747',
'511111305125', 'GRANDMA SYCAMORE', 'MERMAID CEREAL',
'TEST BRANDCODE @1597350074237', 'AQUAFINA', ' SOBE',
'511111905240', 'TEST BRANDCODE @1599849713825',
'CAMPBELLS DINNER SAUCES ', 'WILD STYLE', 'DOVE BODY',
'TEST BRANDCODE @1601939012950', 'TEST BRANDCODE @1597342520238',
'WEIGHT WATCHERS SMART ONES', 'ALEVE LIQUID GELS', 'SANTITAS',
'FRUIT LOVE', 'TROPICANA', 'RAISIN NUT BRAN', '511111405696',
'VELVEETA', 'GOOD HUMOR', 'JUST CRACK AN EGG',
'TEST BRANDCODE @1604946244789', '511111305071',
'TEST BRANDCODE @1598813526618', 'PEPSI', "MOTT'S", 'RIBERHUS',
"HERSHEY'S PUDDING", 'TEST BRANDCODE @1599097538395',
'TEST BRANDCODE @1607636368260', 'GO & GROW',
'TEST BRANDCODE @1595968943087', 'COUNTRY CROCK', "STACY'S",
"MR. YOSHIDA'S", 'TEST BRANDCODE @1604437352151',
'TEST BRANDCODE @1595531348410', 'TEST BRANDCODE @1598296705444',
'LEA & PERRINS', 'HEINZ', 'SARA LEE ',
'TEST BRANDCODE @1602694015481', 'OVEN FRY',
'TEST BRANDCODE @1598716510157', 'TWO HATS',
'STARBUCKS Bottled Drinks', 'FRITOS', 'PASTA RONI', 'BRAND CODE',
'ALLRECIPES', 'V8 V FUSION ', 'TEST BRANDCODE @1599097538537',
'H20H', 'MILLER GENUINE DRAFT', 'TEST BRANDCODE @1603760319814',
'AMAZING GRAINS', 'TEST BRANDCODE @1595531348374',
'CHOCOLATE CHEERIOS', 'ONE', 'KEYSTONE ICE', 'DIGIORNO CHEESE',
'CALEBS KOLA', 'TEST BRANDCODE @1611088503776', 'PILSNER URQUELL',
'TEST BRANDCODE @1597935986474', ' MILANO', 'GENERAL MILLS CEREAL',
'TEST BRANDCODE @1598635635448', 'CLEAR',
'ARNOLD PALMER SPIKED HALF & HALF ORIGINAL', 'ABSOLUT® LIME',
'ABSOLUT® JUICE STRAWBERRY', 'ABSOLUT® ORIGINAL',
'ABSOLUT® MANDRIN', 'ABSOLUT ELYX', 'ABSOLUT® JUICE APPLE',
'ABERLOUR', 'JAMESON COLD BREW ', 'JEFFERSON’S RESERVE',
'CAMPO VIEJO', 'KAHLUA', 'MUMM NAPA ', 'MALFY', 'MALIBU® LIME',
'MALIBU® STRAWBERRY', 'REDBREAST 12YO ',
'TEST BRANDCODE @1597527951412', 'SOL',
'TEST BRANDCODE @1599159969725', "WYLER'S",
'TEST BRANDCODE @1601939762943', 'FAT RABBIT', 'TOTAL',
'TEST BRANDCODE @1598635435829', 'TEST BRANDCODE @1607708191093',
'MALIBU® PINEAPPLE', 'PURE BLISS', 'ROYAL DANSK',
'TEST BRANDCODE @1598711015578', 'DEVOUR', 'CAMPBELLS',
'TEST BRANDCODE @1595340086011', 'STUBBORN SODA', 'AVION',
'TEST BRANDCODE @1610039590413', 'TEST BRANDCODE @1610039590349',
'TEST BRANDCODE @1610039590443', 'TEST BRANDCODE @1610039590383',
'TEST BRANDCODE @1610039590990', 'TEST BRANDCODE @1610495622485',
'TEST BRANDCODE @1610496646104', 'TEST BRANDCODE @1610495622510',
'TEST BRANDCODE @1610495623019', 'TEST BRANDCODE @1610495622536',

'TEST BRANDCODE @1610495622560', 'ONE A DAY', '511111705239',
'COUNTRY CROCK ORIGINAL', "HELLMANN'S/BEST FOODS",
'TEST BRANDCODE @1597527951382', 'TEST BRANDCODE @1601644882040',
'TEST BRANDCODE @1597527952048', 'TEST BRANDCODE @1595968943787',
'TALENTI', 'TEST BRANDCODE @1598635634767', '511111305842',
'RED DOG', 'SMITH & FORGE', 'LIPTON Iced Tea Beverages', 'RUFFLES',
'V8 PLUS ENERGY ', 'POP SECRET', 'TEST BRANDCODE @1601939763539',
'CAPRI SUN', 'NUT HARVEST', 'TEST BRANDCODE @1598881563437',
'TEST BRANDCODE @1600291349208', 'HENRY WEINHARDS',
'TEST BRANDCODE @1599097539367', '511111905035', 'FRUITY CHEERIOS',
'FLINTSTONES MULTIVITAMIN GUMMY', "FOSTER'S",
'F WHITLOCK AND SONS', 'TEST BRANDCODE @1610046687007',
'TEST BRANDCODE @1610046687035', 'TEST BRANDCODE @1610046686980',
'TEST BRANDCODE @1599097538572', "I CAN'T BELIEVE IT'S NOT BUTTER",
'TEST BRANDCODE @1607463684777', 'BAKEN-ETS', 'LIBERTE',
'APPLE CINNAMON CHEERIOS', 'ST STEFANUS',
'DOLE CHILLED FRUIT JUICES', 'SUAVE', 'CLASSICO', 'STROEHMANN',
'BUBLY', '1915 BOLTHOUSE FARMS', 'TEST BRANDCODE @1610046687624',
'TEST BRANDCODE @1610046686954', 'TEST BRANDCODE @1610049748079',
'MIRALAX LAXATIVES', 'TEST BRANDCODE @1607708191128',
'TEST BRANDCODE @1607708191160', 'TEST BRANDCODE @1607708191197',
'TEST BRANDCODE @1607708191721', 'TEST BRANDCODE @1610562208718',
'TEST BRANDCODE @1610653894219', 'TEST BRANDCODE @1610562208600',
'TEST BRANDCODE @1610562208645', 'TEST BRANDCODE @1610562209325',
'TEST BRANDCODE @1610562208680', 'TEST BRANDCODE @1610660035827',
'TEST BRANDCODE @1610660035788', 'TEST BRANDCODE @1610660036398',
'TEST BRANDCODE @1610718082494', "T.G.I. FRIDAY'S", 'CUSQUENA',
'ARNOLD', 'MC CAFE', 'O-KE-DOKE', 'TEST BRANDCODE @1607312532260',
'ALKA SELTZER ADULT HEARTBURN CHEWS/ GUMMY/ TAB',
'TEST BRANDCODE @1595531348450', 'TEST BRANDCODE @1598635435926',
'TEST BRANDCODE @1607636368136', 'OATMEAL CRISP',
'TEST BRANDCODE @1609794603283', "PHILLIPS'®DIGESTIVE",
'TEST BRANDCODE @1610660035741', 'TEST BRANDCODE @1609794603237',
'TEST BRANDCODE @1609794603367', 'TEST BRANDCODE @1609794603327',
'TEST BRANDCODE @1609794603942', 'TEST BRANDCODE @1610038521409',
'TEST BRANDCODE @1610718082601', 'TEST BRANDCODE @1610718083279',
'TEST BRANDCODE @1610718082685', 'TEST BRANDCODE @1610718082644',
'TEST BRANDCODE @1610135036684', 'TEST BRANDCODE @1610493496943',
'TEST BRANDCODE @1610493089082', 'TEST BRANDCODE @1610135035614',
'TEST BRANDCODE @1610135035679', 'TEST BRANDCODE @1610135035546',
'TEST BRANDCODE @1610493088287', 'TEST BRANDCODE @1610493088459',
'TEST BRANDCODE @1610493088421', 'TEST BRANDCODE @1610493088494',
'TEST BRANDCODE @1610058181760', 'TEST BRANDCODE @1610135035332',
'TEST BRANDCODE @1610058182453', 'TEST BRANDCODE @1610058181682',
'TEST BRANDCODE @1610058181719', 'HERSHEYS CEREAL',
'TEST BRANDCODE @1598296704728', 'AMP', '511111905318', 'AXE',
'TEST BRANDCODE @1598635435855', 'TEST BRANDCODE @1606765582946',
"ALEXA-O'S", '511111005704', '511111805069', 'STEEL RESERVE',
'POP WORKS & COMPANY', 'SABRITONES', '511111205456', 'SURE-JELL',
'TERRAPIN', 'SMART MADE', '511111405023', 'LOVE BEAUTY AND PLANET',
'PROTEIN ONE', '511111605492', 'TEST BRANDCODE @1603925787381',
'KOOL-AID', 'TEST BRANDCODE @1601939013241', 'ABSOLUT® CITRON',
'AVION R44', 'TEST BRANDCODE @1607639231710', 'MILLER FORTUNE',
'LEVER 2000', 'A+D', 'PULLUPS', 'TEST BRANDCODE @1606765578435',
'MOUNTAIN DEW', 'SUN CHIPS', 'TEST BRANDCODE @1598813527796',
'PURE LEAF Tea Leaves', 'CORN NUTS', "BREAKSTONE'S", 'SPARKS',
'TEST BRANDCODE @1598716509358', 'MAXWELL HOUSE',
'TEST BRANDCODE @1599849714378', 'TEST BRANDCODE @1598711015496',
'CONTINENTAL SOUP', 'GARNIER - HAIR COLOR', 'AFRIN® NASAL SPRAY',
'GOODBELLY', 'NEXXUS', 'V8', "COLMAN'S", 'PREGO',

```
'TEST BRANDCODE @1606765578985', 'BRUMMEL & BROWN', 'YQ YOPLAIT',
'511111805298', 'TEST BRANDCODE @1604946244833',
'ALKA SELTZER COLD AND FLU TAB/ CAPS/ GEL',
'TEST BRANDCODE @1595531349206', '511111905806', '09090909090',
'511111105831', 'DIETCHRIS2', 'BOLD BUTCHER', "LAY'S",
"BEN & JERRY'S", 'TEST BRANDCODE @1598716509304', 'FRS920',
'TEST BRANDCODE @1598042673378', 'V8 HYDRATE', 'COUNTRY TIME',
'TEST BRANDCODE @1611088503732', 'NAKED JUICE', 'BREYERS',
'TEST BRANDCODE @1604437351646', '511111105275',
'GIRL SCOUT CEREAL', 'MILLER LITE', 'THIRD SHIFT', 'GOODNITES',
'TEST BRANDCODE @1599159969061', 'TEST BRANDCODE @1604434433706',
'MAYBELLINE', 'YOPLAIT', 'TEST BRANDCODE @1598290604214',
'TEST BRANDCODE @1607463684591', 'L'OREAL PARIS - HAIR COLOR',
'ALEVE CAPLETS, TABLETS, AND GEL CAPS', '511111605263',
'TEST BRANDCODE @1598633602226', 'TEST BRANDCODE @1607987891733',
'SAINT ARCHER', 'PHILADELPHIA', 'LATE JULY', 'FREIHOFERS',
'TYSKIE GRONIE', 'KLEENEX', 'TEST BRANDCODE @1601659120870',
'EL ISLENO', "MAISER'S ", 'TEST BRANDCODE @1601939762976', 'VIVA',
'POISE', 'TEST BRANDCODE @1607639231747', 'ORE-IDA',
'511111605331', 'BIAGLUT', 'BITTEN', 'TANG', 'PLUM ORGANICS',
'THE GLENLIVET CARIBBEAN RESERVE',
"THE GLENLIVET® FOUNDER'S RESERVE", 'THE GLENLIVET® 12 YEAR ',
'TASSIMO', 'DIPPIN DOTS', 'STOVE TOP', 'BROWNBERRY',
'TEST BRANDCODE @1601644363664', 'OUI BY YOPLAIT',
'CERVEZA AGUILA', 'TEST BRANDCODE @1598716507602',
'TEST BRANDCODE @1600291349297', 'ICEHOUSE', 'FDS',
'BOLTHOUSE FARMS', 'Q-TIPS', "HOFFMAN'S",
'BETTY CROCKER FRUIT BY THE FOOT', 'ZUMBIDA',
'TEST BRANDCODE @1598296704794', 'TEST BRANDCODE @1607636368759',
'KRUNCHERS', 'TEST BRANDCODE @1607312532218', 'FUNYUNS',
'CAMPBELLS SPAGHETTIOS', 'BRUMMEL AND BROWN',
'TEST BRANDCODE @1597350074997', 'TEST BRANDCODE @1598711019491',
'SIERRA MIST', 'TEST BRANDCODE @1598881572443',
'TEST BRANDCODE @1598290603646', 'TEST BRANDCODE @1599159969108',
'OROWEAT', 'ROLD GOLD', 'TEST BRANDCODE @1611088503815',
'TEST BRANDCODE @1602694015992', 'TEST BRANDCODE @1595968942896',
'V8 SPLASH', 'KOTEX', '511111505082', 'OCEAN SPRAY',
'511111305798', 'BAGEL BITES', 'POLLY-O', 'SUNSILK', "BAKER'S",
'ECCE PANIS', '511111405306', 'AIR WICK', "HAMM'S",
'TEST BRANDCODE @1599849713740', 'MONSTERS CEREAL',
' TAZO BOTTLED TEAS', 'TONI&GUY', '511111005377', '511111905479',
'TEST BRANDCODE @1597935986434', "MICKEY'S", 'CRYSTAL LIGHT',
'COORS EXTRA GOLD', 'D ITALIANO', 'TEST BRANDCODE @1600291349043',
'CLAUSSEN', 'BLUE MOON', 'DIPPIN DOTS CEREAL',
'TEST BRANDCODE @1598639215217', 'LIPTON TEA Leaves',
'TEST BRANDCODE @1613158231644'], dtype=object)
```

In [ ]:
```
b = items['brandCode'].dropna()
b.unique()
```

Out[ ]:
```
array(['MISSION', 'BRAND', 'KRAFT EASY CHEESE', 'PEPSI', 'DORITOS',
       'KLEENEX', 'WINGSTOP', 'GERM-X', 'BEN AND JERRYS', 'BORDEN',
       'KNORR', 'KLARBRUNN', 'HY-VEE', 'LIGHT & FIT GREEK',
       "CONNIE'S PIZZA", "VAN DE KAMP'S", 'HATCH FARMS', "KELLOGG'S",
       'TEMPTATIONS', "NATURE'S PATH ORGANIC", 'DOLE', 'EL MONTEREY',
       'BIGELOW', 'HY-VEE SELECT', 'KIKKOMAN', 'SPECIAL K', 'SWANSON',
       'YUBAN', 'HILLSHIRE FARM', 'JUST BARE', "LAURA'S LEAN BEEF",
       'CAL-ORGANIC FARMS', 'DOLE CHILLED FRUIT JUICES', "BUSH'S BEST",
       'FOLGERS', 'KASHI', 'LIPTON', 'KRAFT', 'GREEN GIANT',
       'HARVEST SNAPS', "THAT'S SMART!", 'TOSTITOS', 'ADVIL',
       'CHICKEN OF THE SEA', 'RICE-A-RONI', 'STARKIST', 'TIC TAC',
       'SO DELICIOUS', 'WONDERFUL', 'LIGHT & FIT', 'HANOVER',
       'HIDDEN VALLEY', 'DANNON', 'KETTLE BRAND', 'FAGE', 'ORAL-B GLIDE',
       "CAMPBELL'S", "FRENCH'S", 'CRISPIX', 'KING ARTHUR FLOUR',
       'KITCHEN BASICS', 'MCCORMICK', 'OLD EL PASO', 'PEPPERIDGE FARM',
       'STOVE TOP', 'ZESTA', 'AZTECA', 'BUNNY', 'NATURE VALLEY',
       'HONEY BUNCHES OF OATS', 'SIMPLE TRUTH ORGANIC', 'BOTA BOX',
       'DARE', 'LINDT', 'ARNOLD', 'ORGANIC ROOT STIMULATOR',
       'GREY POUPON', "MERKT'S", 'MORTON', 'FRONTERA', 'KARO', 'KLONDIKE',
       'CHEESE', 'CRACKER BARREL', "FLORIDA'S NATURAL", 'BLUE DIAMOND',
       'LUNDBERG FAMILY FARMS', 'NUTRI-GRAIN', 'QUAKER', 'DIGIORNO',
       'KEMPS', 'THAI KITCHEN', 'PHILADELPHIA', "THOMAS'", 'POWER CRUNCH',
       'GERBER', 'TACO BELL', 'ORGANICVILLE', 'PURINA ONE', 'DR PEPPER',
       'COTTONELLE', 'C&H', 'CHEERIOS', 'SIMPLE TRUTH', "STOUFFER'S",
       'PRINGLES', "HELLMANN'S/BEST FOODS", 'MCCORMICK GRILL MATES',
       'THOMAS ENGLISH MUFFINS', "MARTINELLI'S", 'COOL WHIP',
       "MARIE CALLENDER'S", "HEMPLER'S", 'JIMMY DEAN', 'SIGNATURE',
       'FRESH EXPRESS', 'MOUNTAIN DEW', 'JELL-O', 'CLASSICO', 'NABISCO',
       'RITZ TOASTED CHIPS', 'LUNCHABLES', 'RAGU', 'KRUSTEAZ',
       'OSCAR MAYER', 'PILLSBURY', 'TILLAMOOK', 'TYSON',
       'BEAR CREEK COUNTRY KITCHENS', 'BIC', 'GENERAL MILLS',
       "JACK LINK'S", 'PLANTERS', 'PROGRESSO', 'RENUZIT', "BRASWELL'S",
       'JOHNSONVILLE', 'OREO', 'THOMAS', 'FRANZ', 'BETTY CROCKER', 'CHEX',
       'CINNAMON TOAST CRUNCH', 'TROLLI', 'VELVEETA', 'ORBIT', 'WISHBONE',
       "WELCH'S", 'PEARLS', 'NUTELLA', 'KRAZY GLUE', 'PREGO',
       'VITAL FARMS ALFRESCO EGGS', 'OVALTINE', 'LA BANDERITA',
       'GRIMMWAY FARMS', 'EGGO', 'PACIFIC FOODS', 'SABRA',
       "MRS. RENFRO'S", "REESE'S", 'KIT KAT', 'ORE-IDA',
       'FORTUNE YAKISOBA', 'FINISH', 'POMPEIAN', 'JUST CRACK AN EGG',
       'SIMPLY POTATOES', 'CHIQUITA', 'CHEEZ-IT', 'FRESH STEP',
       'SIGNATURE KITCHEN', 'ENERGIZER MAX', "HERSHEY'S KISSES", 'KERR',
       'PRIVATE SELECTION', 'SKITTLES', 'KROGER', 'JOHN SOULES FOODS',
       'COLEMAN NATURAL', 'SPARKLING ICE', 'SARGENTO',
       "STOUFFER'S CLASSICS", "BOAR'S HEAD", "JACK DANIEL'S READY TO EAT",
       'ESSENTIAL EVERYDAY', "EGGLAND'S BEST", "MRS. CUBBISON'S",
       'GALLO FAMILY VINEYARDS', 'LITEHOUSE', 'CADBURY', 'TOP FLIGHT',
       'CHEETOS', 'LINDSAY', 'HEWLETT PACKARD', 'PLAYTEX',
       'CREST 3D WHITE', 'HORMEL', 'SMITHFIELD', 'JENNIE-O', 'COKE',
       'DIET COKE', 'V8', 'R.W. KNUDSEN', '7UP', 'HAWAIIAN',
       'TAYLOR FARMS', 'RITZ', 'ARROWHEAD', 'EDWARDS', 'TOLL HOUSE',
       'NESTLE', "SCHWEBEL'S", "MARIE'S", 'JELLY BELLY', 'KERRYGOLD',
       'CARAMELLO', 'CALIFIA FARMS', "FAMOUS DAVE'S", 'BANZA',
       'AMERICAN BEAUTY', 'DELI', 'HERITAGE FARM', 'ROSARITA', 'SHOPRITE',
       'HUGGIES', 'VIVA'], dtype=object)
```

In [ ]:
```
intersect = [item for item in brands['brandCode'].unique() if item in b.unique()]
intersect
```

Out[ ]:
```
['TACO BELL',
 'COTTONELLE',
 'SWANSON',
 'KETTLE BRAND',
 'DORITOS',
 'KLONDIKE',
 'PLANTERS',
 'CHEETOS',
 'COOL WHIP',
 'TOSTITOS',
 'NATURE VALLEY',
 'GREY POUPON',
 'PACIFIC FOODS',
 'KRAFT',
 'PEPPERIDGE FARM',
 'QUAKER',
 'OSCAR MAYER',
 'YUBAN',
 'SARGENTO',
 'KNORR',
 'FINISH',
 'JELL-O',
 'RICE-A-RONI',
 'CRACKER BARREL',
 'LUNCHABLES',
 'HUGGIES',
 'VELVEETA',
 'JUST CRACK AN EGG',
 'PEPSI',
 "HELLMANN'S/BEST FOODS",
 'DOLE CHILLED FRUIT JUICES',
 'CLASSICO',
 'ARNOLD',
 'MOUNTAIN DEW',
 'V8',
 'PREGO',
 'PHILADELPHIA',
 'KLEENEX',
 'VIVA',
 'ORE-IDA',
 'STOVE TOP']
```

Data Cleaning

In [ ]:
```
users = pd.read_csv('users.csv')
brands = pd.read_csv('brands.csv')
receipts = pd.read_csv('receipts.csv')
items_df = pd.read_csv('items.csv')
```

Users

In [ ]:
```
users.head()
```

Out[ ]:

	active	role	signUpSource	state	_id	createdDate	lastLogin
0	True	consumer	Email	WI	5ff1e194b6a9d73a3a9f1052	2021-01-03 15:24:04.800	2021-01-03 15:25:37.857999872
1	True	consumer	Email	WI	5ff1e1eacfcf6c399c274ae6	2021-01-03 15:25:30.554	2021-01-03 15:25:30.596999936
2	True	consumer	Email	WI	5ff1e1e8cfcf6c399c274ad9	2021-01-03 15:25:28.354	2021-01-03 15:25:28.392000000
3	True	consumer	Email	WI	5ff1e1b7cfcf6c399c274a5a	2021-01-03 15:24:39.626	2021-01-03 15:24:39.664999936
4	True	consumer	Email	WI	5ff1e1f1cfcf6c399c274b0b	2021-01-03 15:25:37.564	2021-01-03 15:25:37.599000064

Users Data Schema

- _id: user Id
- state: state abbreviation
- createdDate: when the user created their account
- lastLogin: last time the user was recorded logging in to the app
- role: constant value set to 'CONSUMER'
- active: indicates if the user is active; only Fetch will de-activate an account with this flag

In [ ]:
```
users.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 495 entries, 0 to 494
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   active        495 non-null    bool  
 1   role          495 non-null    object
 2   signUpSource  447 non-null    object
 3   state         439 non-null    object
 4   _id           495 non-null    object
 5   createdDate   495 non-null    object
 6   lastLogin     433 non-null    object
dtypes: bool(1), object(6)
memory usage: 23.8+ KB
```

In [ ]:
```
users.isna().sum()
```

Out[ ]:
```
active           0
role             0
signUpSource    48
state           56
_id              0
createdDate      0
lastLogin       62
dtype: int64
```

In [ ]:
```
users.describe(include='all')
```

Out[ ]:

	active	role	signUpSource	state	_id	createdDate	lastl
count	495	495	447	439	495	495	
unique	2	2	2	8	212	212	
top	True	consumer	Email	WI	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-(16:52:23.2040(
freq	494	413	443	396	20	20	

- Checking Value Counts for some columns

In [ ]:
```
for col in ['active', 'role', 'signUpSource']:
    print(f"{col}: {users[col].value_counts()}")
```

```
active: active
True     494
False      1
Name: count, dtype: int64
role: role
consumer       413
fetch-staff     82
Name: count, dtype: int64
signUpSource: signUpSource
Email     443
Google      4
Name: count, dtype: int64
```

- Just One Inactive user, let's see that user

In [ ]:
```
users[users['active'] == False]
```

Out[ ]:

	active	role	signUpSource	state	_id	createdDate	lastLogi
240	False	consumer	Email	WI	6008622ebe5fc9247bab4eb9	2021-01-20 17:02:38.662	2021-01-2 17:02:38.93100006

- Since there are duplicate user IDs, let's see if the same user appears again in the table or not.

In [ ]:
```
users[users['_id'] == '6008622ebe5fc9247bab4eb9']
```

Out[ ]:

	active	role	signUpSource	state	_id	createdDate	lastLogi
240	False	consumer	Email	WI	6008622ebe5fc9247bab4eb9	2021-01-20 17:02:38.662	2021-01-2 17:02:38.93100006

- What is fetch-staff? Since the data description says that it is consumer by default

In [ ]:
```
users[users['role'] == 'fetch-staff']
```

Out[ ]:

	active	role	signUpSource	state	_id	createdDate	lastLogin
395	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-08 16:42:58.116999936
396	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-08 16:42:58.116999936
397	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-08 16:42:58.116999936
398	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-08 16:42:58.116999936
399	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-08 16:42:58.116999936
400	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-08 16:42:58.116999936
401	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-08 16:42:58.116999936
402	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-08 16:42:58.116999936
403	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-08 16:42:58.116999936
405	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-08 16:42:58.116999936
406	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-08 16:42:58.116999936
407	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-08 16:42:58.116999936
408	True	fetch-staff	Email	WI	5f2068904928021530f8fc34	2020-07-28 18:04:00.905	2021-02-04 15:30:05.375000064
409	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-08 16:42:58.116999936
410	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-08 16:42:58.116999936
411	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-08 16:42:58.116999936
412	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-08 16:42:58.116999936
413	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-08 16:42:58.116999936
414	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-08 16:42:58.116999936
432	True	fetch-staff	Email	NaN	5fbc35711d967d1222cbfefc	2020-11-23 22:19:29.509	2021-02-26 04:25:51.056999936
433	True	fetch-staff	Email	NaN	5fbc35711d967d1222cbfefc	2020-11-23 22:19:29.509	2021-02-26 04:25:51.056999936

	active	role	signUpSource	state	_id	createdDate	lastLogin
434	True	fetch-staff	Email	NaN	5fbc35711d967d1222cbfefc	2020-11-23 22:19:29.509	2021-02-26 04:25:51.056999936
435	True	fetch-staff	Email	NH	5fc961c3b8cfca11a077dd33	2020-12-03 22:08:03.936	2021-02-26 22:39:16.799000064
436	True	fetch-staff	Email	NH	5fc961c3b8cfca11a077dd33	2020-12-03 22:08:03.936	2021-02-26 22:39:16.799000064
437	True	fetch-staff	Email	NH	5fc961c3b8cfca11a077dd33	2020-12-03 22:08:03.936	2021-02-26 22:39:16.799000064
438	True	fetch-staff	Email	NH	5fc961c3b8cfca11a077dd33	2020-12-03 22:08:03.936	2021-02-26 22:39:16.799000064
439	True	fetch-staff	Email	NH	5fc961c3b8cfca11a077dd33	2020-12-03 22:08:03.936	2021-02-26 22:39:16.799000064
440	True	fetch-staff	Email	NH	5fc961c3b8cfca11a077dd33	2020-12-03 22:08:03.936	2021-02-26 22:39:16.799000064
441	True	fetch-staff	Email	NH	5fc961c3b8cfca11a077dd33	2020-12-03 22:08:03.936	2021-02-26 22:39:16.799000064
442	True	fetch-staff	Email	NH	5fc961c3b8cfca11a077dd33	2020-12-03 22:08:03.936	2021-02-26 22:39:16.799000064
443	True	fetch-staff	Email	NH	5fc961c3b8cfca11a077dd33	2020-12-03 22:08:03.936	2021-02-26 22:39:16.799000064
444	True	fetch-staff	Email	NH	5fc961c3b8cfca11a077dd33	2020-12-03 22:08:03.936	2021-02-26 22:39:16.799000064
445	True	fetch-staff	Email	NH	5fc961c3b8cfca11a077dd33	2020-12-03 22:08:03.936	2021-02-26 22:39:16.799000064
446	True	fetch-staff	Email	NH	5fc961c3b8cfca11a077dd33	2020-12-03 22:08:03.936	2021-02-26 22:39:16.799000064
447	True	fetch-staff	Email	NH	5fc961c3b8cfca11a077dd33	2020-12-03 22:08:03.936	2021-02-26 22:39:16.799000064
448	True	fetch-staff	Email	NH	5fc961c3b8cfca11a077dd33	2020-12-03 22:08:03.936	2021-02-26 22:39:16.799000064
449	True	fetch-staff	Email	NH	5fc961c3b8cfca11a077dd33	2020-12-03 22:08:03.936	2021-02-26 22:39:16.799000064
450	True	fetch-staff	Email	NH	5fc961c3b8cfca11a077dd33	2020-12-03 22:08:03.936	2021-02-26 22:39:16.799000064
451	True	fetch-staff	Email	NH	5fc961c3b8cfca11a077dd33	2020-12-03 22:08:03.936	2021-02-26 22:39:16.799000064
452	True	fetch-staff	Email	NH	5fc961c3b8cfca11a077dd33	2020-12-03 22:08:03.936	2021-02-26 22:39:16.799000064
453	True	fetch-staff	Email	NH	5fc961c3b8cfca11a077dd33	2020-12-03 22:08:03.936	2021-02-26 22:39:16.799000064
454	True	fetch-staff	Email	NH	5fc961c3b8cfca11a077dd33	2020-12-03 22:08:03.936	2021-02-26 22:39:16.799000064

	active	role	signUpSource	state	_id	createdDate	lastLogin
455	True	fetch-staff	Email	NaN	5fa41775898c7a11a6bcef3e	2020-11-05 15:17:09.396	2021-03-04 16:02:02.025999872
456	True	fetch-staff	Google	AL	5fa32b4d898c7a11a6bcebce	2020-11-04 22:29:33.309	2021-03-04 07:21:58.047000064
457	True	fetch-staff	Email	NaN	5fa41775898c7a11a6bcef3e	2020-11-05 15:17:09.396	2021-03-04 16:02:02.025999872
458	True	fetch-staff	Email	NaN	5fa41775898c7a11a6bcef3e	2020-11-05 15:17:09.396	2021-03-04 16:02:02.025999872
459	True	fetch-staff	Email	NaN	5fa41775898c7a11a6bcef3e	2020-11-05 15:17:09.396	2021-03-04 16:02:02.025999872
460	True	fetch-staff	Email	NaN	5fa41775898c7a11a6bcef3e	2020-11-05 15:17:09.396	2021-03-04 16:02:02.025999872
461	True	fetch-staff	Email	NaN	5fa41775898c7a11a6bcef3e	2020-11-05 15:17:09.396	2021-03-04 16:02:02.025999872
462	True	fetch-staff	NaN	IL	5964eb07e4b03efd0c0f267b	2017-07-11 15:13:11.771	2021-03-04 19:07:49.769999872
463	True	fetch-staff	Email	NaN	5fa41775898c7a11a6bcef3e	2020-11-05 15:17:09.396	2021-03-04 16:02:02.025999872
464	True	fetch-staff	Email	NaN	5fa41775898c7a11a6bcef3e	2020-11-05 15:17:09.396	2021-03-04 16:02:02.025999872
465	True	fetch-staff	Email	NaN	5fa41775898c7a11a6bcef3e	2020-11-05 15:17:09.396	2021-03-04 16:02:02.025999872
466	True	fetch-staff	Email	NaN	5fa41775898c7a11a6bcef3e	2020-11-05 15:17:09.396	2021-03-04 16:02:02.025999872
467	True	fetch-staff	Email	NaN	5fa41775898c7a11a6bcef3e	2020-11-05 15:17:09.396	2021-03-04 16:02:02.025999872
468	True	fetch-staff	Email	NaN	5fa41775898c7a11a6bcef3e	2020-11-05 15:17:09.396	2021-03-04 16:02:02.025999872
469	True	fetch-staff	Email	NaN	5fa41775898c7a11a6bcef3e	2020-11-05 15:17:09.396	2021-03-04 16:02:02.025999872
470	True	fetch-staff	Email	NaN	5fa41775898c7a11a6bcef3e	2020-11-05 15:17:09.396	2021-03-04 16:02:02.025999872
471	True	fetch-staff	Email	NaN	5fa41775898c7a11a6bcef3e	2020-11-05 15:17:09.396	2021-03-04 16:02:02.025999872
472	True	fetch-staff	Email	NaN	5fa41775898c7a11a6bcef3e	2020-11-05 15:17:09.396	2021-03-04 16:02:02.025999872
473	True	fetch-staff	Email	NaN	5fa41775898c7a11a6bcef3e	2020-11-05 15:17:09.396	2021-03-04 16:02:02.025999872
474	True	fetch-staff	Email	NaN	5fa41775898c7a11a6bcef3e	2020-11-05 15:17:09.396	2021-03-04 16:02:02.025999872
475	True	fetch-staff	NaN	NaN	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-03-05 16:52:23.204000000

	active	role	signUpSource	state	_id	createdDate	lastLogin
476	True	fetch-staff	NaN	NaN	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-03-05 16:52:23.204000000
477	True	fetch-staff	NaN	NaN	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-03-05 16:52:23.204000000
478	True	fetch-staff	NaN	NaN	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-03-05 16:52:23.204000000
479	True	fetch-staff	NaN	NaN	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-03-05 16:52:23.204000000
480	True	fetch-staff	NaN	NaN	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-03-05 16:52:23.204000000
481	True	fetch-staff	NaN	NaN	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-03-05 16:52:23.204000000
482	True	fetch-staff	NaN	NaN	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-03-05 16:52:23.204000000
483	True	fetch-staff	NaN	NaN	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-03-05 16:52:23.204000000
484	True	fetch-staff	NaN	NaN	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-03-05 16:52:23.204000000
485	True	fetch-staff	NaN	NaN	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-03-05 16:52:23.204000000
486	True	fetch-staff	NaN	NaN	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-03-05 16:52:23.204000000
487	True	fetch-staff	NaN	NaN	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-03-05 16:52:23.204000000
488	True	fetch-staff	NaN	NaN	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-03-05 16:52:23.204000000
489	True	fetch-staff	NaN	NaN	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-03-05 16:52:23.204000000
490	True	fetch-staff	NaN	NaN	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-03-05 16:52:23.204000000
491	True	fetch-staff	NaN	NaN	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-03-05 16:52:23.204000000
492	True	fetch-staff	NaN	NaN	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-03-05 16:52:23.204000000
493	True	fetch-staff	NaN	NaN	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-03-05 16:52:23.204000000
494	True	fetch-staff	NaN	NaN	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-03-05 16:52:23.204000000

- Missing Signup sources assumption: They are primarily fetch-staff.
- Conclusion: False. there are some consumers as well.

In [ ]:
```
missing_signupsource = users.groupby('role')['signUpSource'].apply(lambda x: x.isna().
print(missing_signupsource)
```

```
role
consumer        9
fetch-staff    39
Name: signUpSource, dtype: int64
```

- How many signup sources from Google?

In [ ]:
```
users[users['signUpSource'] == 'Google']
```

Out[ ]:

	active	role	signUpSource	state	_id	createdDate	lastLog
170	True	consumer	Google	WI	5e27526d0bdb6a138c32b556	2020-01-21 19:35:09.795	Na
420	True	consumer	Google	AL	5fb0a078be5fc9775c1f3945	2020-11-15 03:28:56.818	Na
429	True	consumer	Google	AL	5fb0a078be5fc9775c1f3945	2020-11-15 03:28:56.818	Na
456	True	fetch-staff	Google	AL	5fa32b4d898c7a11a6bcebce	2020-11-04 22:29:33.309	2021-03- 07:21:58.0470000

- Correcting the Date columns into proper format: '%Y-%m-%d %H:%M:%S.%f' and filling the lastlogin column using 2 strategies:

1. If a user ID is repeated, check if other rows of the same user ID have a lastlogin value, if yes, use that value.
2. If a user IDs all instances have lastlogin as NULL, fill the value with the created date as the user must be logged in for the first time when the account was created.

In [ ]:
```
# Convert the 'createdDate' and 'lastLogin' columns to datetime format
users['createdDate'] = pd.to_datetime(users['createdDate'], format='%Y-%m-%d %H:%M:%S.
users['lastLogin'] = pd.to_datetime(users['lastLogin'], format='%Y-%m-%d %H:%M:%S.%f')

# Fill missing 'lastLogin' values with corresponding 'createdDate' for each '_id'
users['lastLogin'] = users.groupby('_id')['lastLogin'].transform(lambda x: x.fillna(me
users['lastLogin'] = users['lastLogin'].fillna(users['createdDate'])

users.head()
```

Out[ ]:

	active	role	signUpSource	state	_id	createdDate	lastLogin
0	True	consumer	Email	WI	5ff1e194b6a9d73a3a9f1052	2021-01-03 15:24:04.800	2021-01-03 15:25:37.857999872
1	True	consumer	Email	WI	5ff1e194b6a9d73a3a9f1052	2021-01-03 15:24:04.800	2021-01-03 15:25:37.857999872
2	True	consumer	Email	WI	5ff1e194b6a9d73a3a9f1052	2021-01-03 15:24:04.800	2021-01-03 15:25:37.857999872
3	True	consumer	Email	WI	5ff1e1eacfcf6c399c274ae6	2021-01-03 15:25:30.554	2021-01-03 15:25:30.596999936
4	True	consumer	Email	WI	5ff1e194b6a9d73a3a9f1052	2021-01-03 15:24:04.800	2021-01-03 15:25:37.857999872

In [ ]:
```
users.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 495 entries, 0 to 494
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype         
---  ------        --------------  -----         
 0   active        495 non-null    bool          
 1   role          495 non-null    object        
 2   signUpSource  447 non-null    object        
 3   state         439 non-null    object        
 4   _id           495 non-null    object        
 5   createdDate   495 non-null    datetime64[ns]
 6   lastLogin     495 non-null    datetime64[ns]
dtypes: bool(1), datetime64[ns](2), object(4)
memory usage: 23.8+ KB
```

- Filling the missing values in singUpSource and state using the same strategy 1 as lastLogin. But none of the cells got filled using this strategy
- I won't drop any other missing values since we can work around those.

In [ ]:
```
users['signUpSource'] = users.groupby('_id')['signUpSource'].transform(lambda x: x.fil
users['state'] = users.groupby('_id')['state'].transform(lambda x: x.fillna(method='ff
```

In [ ]:
```
users.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 495 entries, 0 to 494
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype         
---  ------        --------------  -----         
 0   active        495 non-null    bool          
 1   role          495 non-null    object        
 2   signUpSource  447 non-null    object        
 3   state         439 non-null    object        
 4   _id           495 non-null    object        
 5   createdDate   495 non-null    datetime64[ns]
 6   lastLogin     495 non-null    datetime64[ns]
dtypes: bool(1), datetime64[ns](2), object(4)
memory usage: 23.8+ KB
```

- Final Missing Value Rows in Users:

In [ ]:
```
missing = users[users.isna().any(axis=1)]
missing[:20]
```

Out[ ]:

	active	role	signUpSource	state	_id	createdDate	lastLogi
344	True	consumer	Email	NaN	60145ff384231211ce796d51	2021-01-29 19:20:19.722	2021-01-2 19:20:19.7220000(
350	True	consumer	Email	NaN	60145ff384231211ce796d51	2021-01-29 19:20:19.722	2021-01-2 19:20:19.7220000(
375	True	consumer	Email	NaN	60186237c8b50e11d8454d5f	2021-02-01 20:19:03.551	2021-02-(20:19:03.5510000(
376	True	consumer	Email	NaN	60186237c8b50e11d8454d5f	2021-02-01 20:19:03.551	2021-02-(20:19:03.5510000(
378	True	consumer	Email	NaN	60186237c8b50e11d8454d5f	2021-02-01 20:19:03.551	2021-02-(20:19:03.5510000(
381	True	consumer	Email	NaN	60186237c8b50e11d8454d5f	2021-02-01 20:19:03.551	2021-02-(20:19:03.5510000(
382	True	consumer	Email	NaN	60186237c8b50e11d8454d5f	2021-02-01 20:19:03.551	2021-02-(20:19:03.5510000(
388	True	consumer	NaN	WI	55308179e4b0eabd8f99caa2	2015-04-17 03:43:53.186	2018-05-(17:23:40.0030000(
395	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-(16:42:58.1169999(
396	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-(16:42:58.1169999(
397	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-(16:42:58.1169999(
398	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-(16:42:58.1169999(
399	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-(16:42:58.1169999(
400	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-(16:42:58.1169999(
401	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-(16:42:58.1169999(
402	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-(16:42:58.1169999(
403	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-(16:42:58.1169999(
405	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-(16:42:58.1169999(
406	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-(16:42:58.1169999(
407	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-(16:42:58.1169999(

- Checking for duplicated rows

In [ ]:
```
users.duplicated().sum()
```

Out[ ]:
```
283
```

- Dropping duplicate Rows

In [ ]:
```
cleaned_users = users.drop_duplicates().reset_index(drop=True)
cleaned_users.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 212 entries, 0 to 211
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype         
---  ------        --------------  -----         
 0   active        212 non-null    bool          
 1   role          212 non-null    object        
 2   signUpSource  207 non-null    object        
 3   state         206 non-null    object        
 4   _id           212 non-null    object        
 5   createdDate   212 non-null    datetime64[ns]
 6   lastLogin     212 non-null    datetime64[ns]
dtypes: bool(1), datetime64[ns](2), object(4)
memory usage: 10.3+ KB
```

In [ ]:
```
cleaned_users.head(10)
```

Out[ ]:

	active	role	signUpSource	state	_id	createdDate	lastLogin
0	True	consumer	Email	WI	5ff1e194b6a9d73a3a9f1052	2021-01-03 15:24:04.800	2021-01-03 15:25:37.857999872
1	True	consumer	Email	WI	5ff1e1eacfcf6c399c274ae6	2021-01-03 15:25:30.554	2021-01-03 15:25:30.596999936
2	True	consumer	Email	WI	5ff1e1e8cfcf6c399c274ad9	2021-01-03 15:25:28.354	2021-01-03 15:25:28.392000000
3	True	consumer	Email	WI	5ff1e1b7cfcf6c399c274a5a	2021-01-03 15:24:39.626	2021-01-03 15:24:39.664999936
4	True	consumer	Email	WI	5ff1e1f1cfcf6c399c274b0b	2021-01-03 15:25:37.564	2021-01-03 15:25:37.599000064
5	True	consumer	Email	WI	5ff1e1e4cfcf6c399c274ac3	2021-01-03 15:25:24.656	2021-01-03 15:25:24.694000128
6	True	consumer	Email	WI	5ff1e1b4cfcf6c399c274a54	2021-01-03 15:24:36.410	2021-01-03 15:24:36.452000000
7	True	consumer	Email	WI	5ff370c562fde912123a5e0e	2021-01-04 19:47:17.776	2021-01-04 19:50:50.563000064
8	True	consumer	Email	WI	5ff36d0362fde912123a5535	2021-01-04 19:31:15.973	2021-01-04 19:34:42.944000000
9	True	consumer	Email	WI	5ff36d83135e7011bcb864d6	2021-01-04 19:33:23.244	2021-01-04 19:33:23.424999936

In [ ]:

```
missing = cleaned_users [cleaned_users.isna().any(axis=1)]
missing[:20]
```

Out[ ]:

	active	role	signUpSource	state	_id	createdDate	lastLog
165	True	consumer	Email	NaN	60145ff384231211ce796d51	2021-01-29 19:20:19.722	2021-01-2 19:20:19.7220000
180	True	consumer	Email	NaN	60186237c8b50e11d8454d5f	2021-02-01 20:19:03.551	2021-02-(20:19:03.5510000
188	True	consumer	NaN	WI	55308179e4b0eabd8f99caa2	2015-04-17 03:43:53.186	2018-05-(17:23:40.0030000
194	True	fetch-staff	NaN	WI	59c124bae4b0299e55b0f330	2017-09-19 14:07:54.302	2021-02-(16:42:58.1169999
204	True	consumer	NaN	NaN	5a43c08fe4b014fd6b6a0612	2017-12-27 15:47:27.059	2021-02-1 16:22:37.1550000
206	True	fetch-staff	Email	NaN	5fbc35711d967d1222cbfefc	2020-11-23 22:19:29.509	2021-02-2 04:25:51.0569999
208	True	fetch-staff	Email	NaN	5fa41775898c7a11a6bcef3e	2020-11-05 15:17:09.396	2021-03-(16:02:02.0259998
210	True	fetch-staff	NaN	IL	5964eb07e4b03efd0c0f267b	2017-07-11 15:13:11.771	2021-03-(19:07:49.7699998
211	True	fetch-staff	NaN	NaN	54943462e4b07e684157a532	2014-12-19 14:21:22.381	2021-03-(16:52:23.2040000

In [ ]:
```
cleaned_users.to_csv('cleaned_users.csv', index=False)
```

brands

Brand Data Schema

- _id: brand uuid
- barcode: the barcode on the item
- brandCode: String that corresponds with the brand column in a partner product file
- category: The category name for which the brand sells products in
- categoryCode: The category code that references a BrandCategory
- cpg: reference to CPG collection
- topBrand: Boolean indicator for whether the brand should be featured as a 'top brand'
- name: Brand name

In [ ]:
```
cleaned_brands = brands.copy()
```

In [ ]:
```
cleaned_brands.head()
```

Out[ ]:

	barcode	category	categoryCode	name	topBrand	brandCode	
0	511111019862	Baking	BAKING	test brand @1612366101024	False	NaN	601ac
1	511111519928	Beverages	BEVERAGES	Starbucks	False	STARBUCKS	601c!
2	511111819905	Baking	BAKING	test brand @1612366146176	False	TEST BRANDCODE @1612366146176	601ac
3	511111519874	Baking	BAKING	test brand @1612366146051	False	TEST BRANDCODE @1612366146051	601ac
4	511111319917	Candy & Sweets	CANDY_AND_SWEETS	test brand @1612366146827	False	TEST BRANDCODE @1612366146827	601ac

In [ ]:
```
cleaned_brands.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1167 entries, 0 to 1166
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   barcode       1167 non-null   int64 
 1   category      1012 non-null   object
 2   categoryCode  517 non-null    object
 3   name          1167 non-null   object
 4   topBrand      555 non-null    object
 5   brandCode     898 non-null    object
 6   _id           1167 non-null   object
 7   cpg_id        1167 non-null   object
 8   cpg_ref       1167 non-null   object
dtypes: int64(1), object(8)
memory usage: 82.2+ KB
```

In [ ]:
```
cleaned_brands.describe(include='all')
```

Out[ ]:

	barcode	category	categoryCode	name	topBrand	brandCode	
count	1.167000e+03	1012	517	1167	555	898	
unique	NaN	23	14	1156	2	896	
top	NaN	Baking	BAKING	Huggies	False	GOODNITES	601ac115be37ce2ead43
freq	NaN	369	359	2	524	2	
mean	5.111115e+11	NaN	NaN	NaN	NaN	NaN	
std	2.874497e+05	NaN	NaN	NaN	NaN	NaN	
min	5.111110e+11	NaN	NaN	NaN	NaN	NaN	
25%	5.111112e+11	NaN	NaN	NaN	NaN	NaN	
50%	5.111114e+11	NaN	NaN	NaN	NaN	NaN	
75%	5.111117e+11	NaN	NaN	NaN	NaN	NaN	
max	5.111119e+11	NaN	NaN	NaN	NaN	NaN	

In [ ]:
```
cleaned_brands.isna().sum()
```

Out[ ]:
```
barcode           0
category        155
categoryCode    650
name              0
topBrand        612
brandCode       269
_id               0
cpg_id            0
cpg_ref           0
dtype: int64
```

- Checking for duplicate rows

In [ ]:
```
cleaned_brands.duplicated().sum()
```

Out[ ]:
```
0
```

- Getting an overview of value distribution of some columns

In [ ]:
```
for col in ['brandCode', 'name', 'category', 'categoryCode']:
    print(f"{col}: {cleaned_brands[col].nunique()} values: {cleaned_brands[col].value_
    print()
```

```
brandCode: 896 values: brandCode
GOODNITES                          2
HUGGIES                            2
TEST BRANDCODE @1598711015578      1
SOL                                1
TEST BRANDCODE @1599159969725      1
                                  ..
TEST BRANDCODE @1599159969028      1
TEST BRANDCODE @1597350074404      1
SEDAL                              1
RED ROCK DELI                      1
TEST BRANDCODE @1613158231644      1
Name: count, Length: 896, dtype: int64

name: 1156 values: name
Huggies                        2
V8 Hydrate                     2
Pull-Ups                       2
Dippin Dots® Cereal            2
Diabetic Living Magazine       2
                             ..
Claritin® KIDS                 1
Athenos                        1
test brand @1599159969028      1
test brand @1597350074404      1
test brand @1613158231643      1
Name: count, Length: 1156, dtype: int64

category: 23 values: category
Baking                           369
Beer Wine Spirits                 90
Snacks                            75
Candy & Sweets                    71
Beverages                         63
Magazines                         44
Health & Wellness                 44
Breakfast & Cereal                40
Grocery                           39
Dairy                             33
Condiments & Sauces               27
Frozen                            24
Personal Care                     20
Baby                              18
Canned Goods & Soups              12
Beauty                             9
Cleaning & Home Improvement        6
Deli                               6
Beauty & Personal Care             6
Household                          5
Bread & Bakery                     5
Dairy & Refrigerated               5
Outdoor                            1
Name: count, dtype: int64

categoryCode: 14 values: categoryCode
BAKING                           359
CANDY_AND_SWEETS                  71
BEER_WINE_SPIRITS                 31
HEALTHY_AND_WELLNESS              14
GROCERY                           11
```

```
BABY                                7
CLEANING_AND_HOME_IMPROVEMENT       6
BREAD_AND_BAKERY                    5
DAIRY_AND_REFRIGERATED              5
PERSONAL_CARE                       4
BEVERAGES                           1
OUTDOOR                             1
MAGAZINES                           1
FROZEN                              1
Name: count, dtype: int64
```

In [ ]:
```
cleaned_brands.columns
```

Out[ ]:
```
Index(['barcode', 'category', 'categoryCode', 'name', 'topBrand', 'brandCode', '_id',
'cpg_id', 'cpg_ref'], dtype='object')
```

- Making sure values in these columns are consistent and not have extra leading or trailing spaces:

In [ ]:
```
cleaned_brands['name'] = brands['name'].str.strip()
cleaned_brands['category'] = brands['category'].str.strip()
cleaned_brands['brandCode'] = brands['brandCode'].str.strip().str.upper()
```

- Checking first 20 rows with missing values

In [ ]:
```
missing = cleaned_brands[cleaned_brands.isna().any(axis=1)]
missing[:20]
```

Out[ ]:

	barcode	category	categoryCode	name	topBrand	brandCode
0	511111019862	Baking	BAKING	test brand @1612366101024	False	NaN
7	511111104810	Condiments & Sauces	NaN	J.L. Kraft	NaN	J.L. KRAFT
8	511111504412	Canned Goods & Soups	NaN	Campbell's Home Style	False	CAMPBELLS HOME STYLE
9	511111504788	Baking	NaN	test	NaN	TEST
10	511111516354	Baking	BAKING	test brand @1598813526777	NaN	TEST BRANDCODE @1598813526777
11	511111102540	NaN	NaN	MorningStar	NaN	NaN
12	511111201076	Baking	NaN	Calumet	False	CALUMET
13	511111205012	Magazines	NaN	Entertainment Weekly	NaN	511111205012
14	511111801801	Breakfast & Cereal	NaN	AUNT JEMIMA Syrup	False	AUNT JEMIMA SYRUP
15	511111202233	Beer Wine Spirits	NaN	Molson Canadian	False	MOLSON
17	511111515319	Baking	BAKING	test brand @1597342520277	NaN	TEST BRANDCODE @1597342520277
18	511111317364	Baking	BAKING	test brand @1605535049181	False	NaN
19	511111300700	Beauty	NaN	ST. IVES	False	ST IVES
20	511111305125	Baby	NaN	Chris Image Test	NaN	CHRISIMAGE
21	511111005650	Health & Wellness	HEALTHY_AND_WELLNESS	Alka-Seltzer®	NaN	ALKA SELTZER
22	511111802129	Condiments & Sauces	NaN	Jack Daniel's	False	JACK DANIEL'S BARBECUE
23	511111303947	NaN	NaN	Bottled Starbucks	NaN	NaN
24	511111802914	NaN	NaN	Full Throttle	NaN	NaN
25	511111914549	Baking	BAKING	PopUp Brand A	NaN	NaN
26	511111400769	Frozen	NaN	MAGNUM Ice Cream	False	MAGNUM ICE CREAM

- Checking the topBrand column

In [ ]:
```
cleaned_brands['topBrand'].value_counts()
```

Out[ ]:
```
topBrand
False    524
True      31
Name: count, dtype: int64
```

- Hmm, total values don't match the total number of rows. Let's see how many missing rows are there:

In [ ]:
```
cleaned_brands['topBrand'].isna().sum()
```

Out[ ]:
```
612
```

- **Assumption**: I am listing NULL topBrands as False. Because if they were important to be noted as Top Brand, the value should have been present.

In [ ]:
```
cleaned_brands['topBrand'] = brands['topBrand'].fillna(False)
```

- Type casting to INT for easier import in MySQL. MySQL Requires imported boolean fields to be noted as integers. The field topBrand in the MySQL schema will be denoted as Boolean only

In [ ]:
```
cleaned_brands['topBrand'] = cleaned_brands['topBrand'].astype(int)
cleaned_brands['topBrand'].value_counts()
```

Out[ ]:
```
topBrand
0    1136
1      31
Name: count, dtype: int64
```

- So, we have 31 brands as topBrand. Let's check these brands

In [ ]:
```
top_brands = cleaned_brands[cleaned_brands['topBrand'] == 1]
top_brands
```

Out[ ]:

	barcode	category	categoryCode	name	topBrand	brandCode	
58	511111106876	Grocery	NaN	DASH-2249 Brand1	1	TEST BRAND CODE	5c76d3cd95144c5
109	511111801757	Snacks	NaN	Chester's	1	CHESTER'S	585a9645e4b03e6
115	511111001119	Snacks	NaN	Doritos	1	DORITOS	5887a372e4b0218
116	511111101895	Condiments & Sauces	NaN	A.1.	1	A.1.	57ed0697e4b072a
152	511111204923	Grocery	NaN	Brand1	1	0987654321	5c45f91b87ff355
192	511111001768	Snacks	NaN	Cheetos	1	CHEETOS	585a963ce4b03e6
245	511111812449	Magazines	NaN	Test brand1	1	NaN	5d66961cee7f2d2
258	511111902461	Baby	NaN	Antarctica	1	AMP2	57c0827de4b071
271	511111501770	Breakfast & Cereal	NaN	Cap'n Crunch	1	CAP'N CRUNCH	585a9637e4b03e6
278	511111801689	Snacks	NaN	Lay's Kettle Cooked	1	NaN	585a967fe4b03e6
344	511111700531	Snacks	NaN	FRITO-LAY	1	FRITO-LAY	5a4d1f06e4b0d5c
428	511111208532	Grocery	NaN	Test FRS-920 again1	1	AMPTEST	5c9a975495144c1
467	511111004790	Baking	NaN	alexa	1	ALEXA	5c409ab4cd244a3
471	511111901709	Snacks	NaN	Grandma's	1	GRANDMA'S	585a966be4b03e6
580	511111806899	Baby	NaN	DASH-2249 1 brand 1	1	DASH-2249 1 BRAND 1	5c76dd4d95144c5
581	511111301745	Snacks	NaN	Cracker Jack	1	CRACKER JACK	585a964ce4b03e6
721	511111601722	Snacks	NaN	Fritos	1	FRITOS	585a965be4b03e6
723	511111702665	Beverages	NaN	7 up	1	BRAND CODE	55b6309ce4b0d8e
773	511111201793	Snacks	NaN	Baked!	1	NaN	585a9618e4b03e6
848	511111701781	Snacks	NaN	Baken-Ets	1	BAKEN-ETS	585a961fe4b03e6
943	511111806509	Breakfast & Cereal	NaN	alexa-o's	1	ALEXA-O'S	5c6d51d695144c6
978	511111312949	Baby	NaN	Pull-Ups	1	PULLUPS	5db3288aee7f2d6
1012	511111605058	Dairy	NaN	Brand2	1	09090909090	5c4637ba87ff356
1018	511111601678	Snacks	NaN	Lay's Original	1	LAY'S	585a9686e4b03e6
1020	511111300939	Frozen	FROZEN	BEN &	1	BEN &	592486bde410d61

	barcode	category	categoryCode	name	topBrand	brandCode	
				JERRY'S		JERRY'S	
1023	511111106432	Beverages	NaN	new-brand, of course	1	FRS920	5c6c2f9295144c6
1030	511111800927	Frozen	NaN	BREYERS	1	BREYERS	592486bde410d61
1036	511111112938	Baby	BABY	GoodNites	1	GOODNITES	5db32879ee7f2d6
1061	511111900542	Snacks	NaN	EL ISLENO	1	EL ISLENO	5a4d1ec9e4b0bcb
1074	511111707202	Baby	BABY	Huggies	1	HUGGIES	5c7d9cb395144c3
1109	511111401711	Snacks	NaN	Funyuns	1	FUNYUNS	585a9661e4b03e6

In [ ]:
```
cleaned_brands.nunique()
```

Out[ ]:
```
barcode         1160
category          23
categoryCode      14
name            1156
topBrand           2
brandCode        895
_id             1167
cpg_id           196
cpg_ref            2
dtype: int64
```

In [ ]:
```
cleaned_brands.head()
```

Out[ ]:

	barcode	category	categoryCode	name	topBrand	brandCode	
0	511111019862	Baking	BAKING	test brand @1612366101024	0	NaN	601ac
1	511111519928	Beverages	BEVERAGES	Starbucks	0	STARBUCKS	601c
2	511111819905	Baking	BAKING	test brand @1612366146176	0	TEST BRANDCODE @1612366146176	601ac
3	511111519874	Baking	BAKING	test brand @1612366146051	0	TEST BRANDCODE @1612366146051	601ac
4	511111319917	Candy & Sweets	CANDY_AND_SWEETS	test brand @1612366146827	0	TEST BRANDCODE @1612366146827	601ac

In [ ]:
```
cleaned_brands.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1167 entries, 0 to 1166
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   barcode       1167 non-null   int64 
 1   category      1012 non-null   object
 2   categoryCode  517 non-null    object
 3   name          1167 non-null   object
 4   topBrand      1167 non-null   int64 
 5   brandCode     898 non-null    object
 6   _id           1167 non-null   object
 7   cpg_id        1167 non-null   object
 8   cpg_ref       1167 non-null   object
dtypes: int64(2), object(7)
memory usage: 82.2+ KB
```

In [ ]:
```
cleaned_brands.isna().sum()
```

Out[ ]:
```
barcode           0
category        155
categoryCode    650
name              0
topBrand          0
brandCode       269
_id               0
cpg_id            0
cpg_ref           0
dtype: int64
```

- **Assumption**: Categories have corresponding category code. Since a lot of category codes are missing, we can fill those codes with corresponding category.

In [ ]:
```
missing
```

Out[ ]:

	barcode	category	categoryCode	name	topBrand	brandCode
0	511111019862	Baking	BAKING	test brand @1612366101024	False	NaN
7	511111104810	Condiments & Sauces	NaN	J.L. Kraft	NaN	J.L. KRAFT
8	511111504412	Canned Goods & Soups	NaN	Campbell's Home Style	False	CAMPBELLS HOME STYLE
9	511111504788	Baking	NaN	test	NaN	TEST
10	511111516354	Baking	BAKING	test brand @1598813526777	NaN	TEST BRANDCODE @1598813526777
...	...	...	...	...	...	...
1161	511111403845	Beer Wine Spirits	NaN	Blue Moon	False	BLUE MOON
1162	511111116752	Baking	BAKING	test brand @1601644365844	NaN	NaN
1163	511111706328	Breakfast & Cereal	NaN	Dippin Dots® Cereal	NaN	DIPPIN DOTS CEREAL
1164	511111416173	Candy & Sweets	CANDY_AND_SWEETS	test brand @1598639215217	NaN	TEST BRANDCODE @1598639215217
1165	511111400608	Grocery	NaN	LIPTON TEA Leaves	False	LIPTON TEA LEAVES

912 rows × 9 columns

- We can see that category code for row 9 and 10 is null but the category is baking. We see row 0 has baking as category and as the category code.
- We can use this information to fill the missing category code in row 9 and 10

In [ ]:
```
def fill_with_mode(series):
    if series.mode().empty:
        return series
    else:
        return series.fillna(series.mode()[0])

cleaned_brands['categoryCode'] = cleaned_brands.groupby('category')['categoryCode'].tr
```

In [ ]:
```
cleaned_brands.isna().sum()
```

Out[ ]:
```
barcode           0
category        155
categoryCode    368
name              0
topBrand          0
brandCode       269
_id               0
cpg_id            0
cpg_ref           0
dtype: int64
```

- Some rows did get filled. Let's check the first 15 rows and focus on row 9 and 10

In [ ]:
```
cleaned_brands.head(10)
```

Out[ ]:

	barcode	category	categoryCode	name	topBrand	brandCode	
0	511111019862	Baking	BAKING	test brand @1612366101024	0	NaN	60
1	511111519928	Beverages	BEVERAGES	Starbucks	0	STARBUCKS	60
2	511111819905	Baking	BAKING	test brand @1612366146176	0	TEST BRANDCODE @1612366146176	601
3	511111519874	Baking	BAKING	test brand @1612366146051	0	TEST BRANDCODE @1612366146051	60
4	511111319917	Candy & Sweets	CANDY_AND_SWEETS	test brand @1612366146827	0	TEST BRANDCODE @1612366146827	60
5	511111719885	Baking	BAKING	test brand @1612366146091	0	TEST BRANDCODE @1612366146091	601
6	511111219897	Baking	BAKING	test brand @1612366146133	0	TEST BRANDCODE @1612366146133	60
7	511111104810	Condiments & Sauces	NaN	J.L. Kraft	0	J.L. KRAFT	5c
8	511111504412	Canned Goods & Soups	NaN	Campbell's Home Style	0	CAMPBELLS HOME STYLE	5ab
9	511111504788	Baking	BAKING	test	0	TEST	5c

- We can see the cells are filled as we planned!

In [ ]:
```
cleaned_brands.to_csv('cleaned_brands.csv', index=False)
```

Receipts

Receipts Data Schema

- _id: uuid for this receipt
- bonusPointsEarned: Number of bonus points that were awarded upon receipt completion
- bonusPointsEarnedReason: event that triggered bonus points
- createDate: The date that the event was created
- dateScanned: Date that the user scanned their receipt
- finishedDate: Date that the receipt finished processing
- modifyDate: The date the event was modified
- pointsAwardedDate: The date we awarded points for the transaction
- pointsEarned: The number of points earned for the receipt
- purchaseDate: the date of the purchase
- purchasedItemCount: Count of number of items on the receipt
- rewardsReceiptItemList: The items that were purchased on the receipt
- rewardsReceiptStatus: status of the receipt through receipt validation and processing
- totalSpent: The total amount on the receipt
- userId: string id back to the User collection for the user who scanned the receipt

In [ ]:
```
cleaned_receipts = receipts.copy()
cleaned_receipts.head()
```

Out[ ]:

	_id_$oid	bonusPointsEarned	bonusPointsEarnedReason	createDate_$date	dateSca
0	5ff1e1eb0a720f0523000575	500.0	Receipt number 2 completed, bonus point schedu...	1609687531000	160
1	5ff1e1bb0a720f052300056b	150.0	Receipt number 5 completed, bonus point schedu...	1609687483000	160
2	5ff1e1f10a720f052300057a	5.0	All-receipts receipt bonus	1609687537000	160
3	5ff1e1ee0a7214ada100056f	5.0	All-receipts receipt bonus	1609687534000	160
4	5ff1e1d20a7214ada1000561	5.0	All-receipts receipt bonus	1609687506000	160

In [ ]:
```
cleaned_receipts.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1119 entries, 0 to 1118
Data columns (total 15 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   _id_$oid                 1119 non-null   object
 1   bonusPointsEarned        544 non-null    float64
 2   bonusPointsEarnedReason  544 non-null    object
 3   createDate_$date         1119 non-null   int64
 4   dateScanned_$date        1119 non-null   int64
 5   finishedDate_$date       568 non-null    float64
 6   modifyDate_$date         1119 non-null   int64
 7   pointsAwardedDate_$date  537 non-null    float64
 8   pointsEarned             609 non-null    float64
 9   purchaseDate_$date       671 non-null    float64
 10  purchasedItemCount       635 non-null    float64
 11  rewardsReceiptItemList   679 non-null    object
 12  rewardsReceiptStatus     1119 non-null   object
 13  totalSpent               684 non-null    float64
 14  userId                   1119 non-null   object
dtypes: float64(7), int64(3), object(5)
memory usage: 131.3+ KB
```

In [ ]:
```
cleaned_receipts.describe(include='all')
```

Out[ ]:

	_id_$oid	bonusPointsEarned	bonusPointsEarnedReason	createDate_$date	da
count	1119	544.000000	544	1.119000e+03	
unique	1119	NaN	9	NaN	
top	5ff1e1eb0a720f0523000575	NaN	All-receipts receipt bonus	NaN	
freq	1	NaN	183	NaN	
mean	NaN	238.893382	NaN	1.611800e+12	
std	NaN	299.091731	NaN	1.484091e+09	
min	NaN	5.000000	NaN	1.604089e+12	
25%	NaN	5.000000	NaN	1.610652e+12	
50%	NaN	45.000000	NaN	1.611941e+12	
75%	NaN	500.000000	NaN	1.612704e+12	
max	NaN	750.000000	NaN	1.614641e+12	

In [ ]:
```
def remove_date_suffix(df):
  """
  Removes the `$date` suffix from column names that end with it.

  Args:
      df: The pandas DataFrame to modify.

  Returns:
      A new DataFrame with the modified column names.
  """
```

```
  new_columns = []
  for column in df.columns:
    if column.endswith('_$date'):
      new_columns.append(column[:-6])
    else:
      new_columns.append(column)

  df.columns = new_columns
  return df

def clean_receipts(df):
  # Remove the $oid from column name
  df.rename(columns={"_id_$oid": "_id"}, inplace=True)
  df = remove_date_suffix(df)

  # Cast the date fields into datetime data type - similar to how it was done for user
  date_columns = ['createDate', 'dateScanned', 'finishedDate', 'modifyDate', 'pointsAw
  for col in date_columns:
      df[col] = pd.to_datetime(df[col], unit='ms', errors='coerce')
      df[col] = df[col].dt.strftime('%Y-%m-%d %H:%M:%S.%f')
      df[col] = pd.to_datetime(df[col], format='%Y-%m-%d %H:%M:%S.%f')

  # Due to some import problems in MySQL, this was a safeguard step to avoid fatal imp
  # Filled the null values in bonusPointsEarned and TotalSpent columns to 0
  # Casted the data type as float - the data was already float but there were some fat
  for col in ['bonusPointsEarned', 'totalSpent']:
      df[col] = df[col].fillna(0.0)
      df[col] = df[col].astype(float)

clean_receipts(cleaned_receipts)
cleaned_receipts.head()
```

Out[ ]:

	_id	bonusPointsEarned	bonusPointsEarnedReason	createDate	dateScanned
0	5ff1e1eb0a720f0523000575	500.0	Receipt number 2 completed, bonus point schedu...	2021-01-03 15:25:31	2021-01-03 15:25:31
1	5ff1e1bb0a720f052300056b	150.0	Receipt number 5 completed, bonus point schedu...	2021-01-03 15:24:43	2021-01-03 15:24:43
2	5ff1e1f10a720f052300057a	5.0	All-receipts receipt bonus	2021-01-03 15:25:37	2021-01-03 15:25:37
3	5ff1e1ee0a7214ada100056f	5.0	All-receipts receipt bonus	2021-01-03 15:25:34	2021-01-03 15:25:34
4	5ff1e1d20a7214ada1000561	5.0	All-receipts receipt bonus	2021-01-03 15:25:06	2021-01-03 15:25:06

In [ ]:

```
cleaned_receipts.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1119 entries, 0 to 1118
Data columns (total 15 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   _id                      1119 non-null   object
 1   bonusPointsEarned        1119 non-null   float64
 2   bonusPointsEarnedReason  544 non-null    object
 3   createDate               1119 non-null   datetime64[ns]
 4   dateScanned              1119 non-null   datetime64[ns]
 5   finishedDate             568 non-null    datetime64[ns]
 6   modifyDate               1119 non-null   datetime64[ns]
 7   pointsAwardedDate        537 non-null    datetime64[ns]
 8   pointsEarned             609 non-null    float64
 9   purchaseDate             671 non-null    datetime64[ns]
 10  purchasedItemCount       635 non-null    float64
 11  rewardsReceiptItemList   679 non-null    object
 12  rewardsReceiptStatus     1119 non-null   object
 13  totalSpent               1119 non-null   float64
 14  userId                   1119 non-null   object
dtypes: datetime64[ns](6), float64(4), object(5)
memory usage: 131.3+ KB
```

In [ ]:
```
cleaned_receipts.columns
```

Out[ ]:
```
Index(['_id', 'bonusPointsEarned', 'bonusPointsEarnedReason', 'createDate', 'dateScanned', 'finishedDate', 'modifyDate', 'pointsAwardedDate', 'pointsEarned', 'purchaseDate', 'purchasedItemCount', 'rewardsReceiptItemList', 'rewardsReceiptStatus', 'totalSpent', 'userId'], dtype='object')
```

In [ ]:
```
cleaned_receipts.head(20)
```

Out[ ]:

	_id	bonusPointsEarned	bonusPointsEarnedReason	createDate	dateScanned
0	5ff1e1eb0a720f0523000575	500.0	Receipt number 2 completed, bonus point schedu...	2021-01-03 15:25:31	2021-01-03 15:25:31
1	5ff1e1bb0a720f052300056b	150.0	Receipt number 5 completed, bonus point schedu...	2021-01-03 15:24:43	2021-01-03 15:24:43
2	5ff1e1f10a720f052300057a	5.0	All-receipts receipt bonus	2021-01-03 15:25:37	2021-01-03 15:25:37
3	5ff1e1ee0a7214ada100056f	5.0	All-receipts receipt bonus	2021-01-03 15:25:34	2021-01-03 15:25:34
4	5ff1e1d20a7214ada1000561	5.0	All-receipts receipt bonus	2021-01-03 15:25:06	2021-01-03 15:25:06
5	5ff1e1e40a7214ada1000566	750.0	Receipt number 1 completed, bonus point schedu...	2021-01-03 15:25:24	2021-01-03 15:25:24
6	5ff1e1cd0a720f052300056f	5.0	All-receipts receipt bonus	2021-01-03 15:25:01	2021-01-03 15:25:01
7	5ff1e1a40a720f0523000569	500.0	Receipt number 2 completed, bonus point schedu...	2021-01-03 15:24:20	2021-01-03 15:24:20
8	5ff1e1ed0a7214ada100056e	5.0	All-receipts receipt bonus	2021-01-03 15:25:33	2021-01-03 15:25:33
9	5ff1e1eb0a7214ada100056b	250.0	Receipt number 3 completed, bonus point schedu...	2021-01-03 15:25:31	2021-01-03 15:25:31
10	5ff1e1c50a720f052300056c	100.0	Receipt number 6 completed, bonus point schedu...	2021-01-03 15:24:53	2021-01-03 15:24:53
11	5ff1e1a10a720f0523000568	750.0	Receipt number 1 completed, bonus point schedu...	2021-01-03 15:24:17	2021-01-03 15:24:17
12	5ff1e1b60a7214ada100055c	150.0	Receipt number 5 completed, bonus point schedu...	2021-01-03 15:24:38	2021-01-03 15:24:38
13	5f9c74f70a7214ad07000037	750.0	Receipt number 1 completed, bonus point schedu...	2020-10-30 20:17:59	2020-10-30 20:17:59
14	5ff1e1b20a7214ada100055a	300.0	Receipt number 4 completed, bonus point schedu...	2021-01-03 15:24:34	2021-01-03 15:24:34
15	5ff1e1e90a7214ada1000569	0.0	NaN	2021-01-03 15:25:29	2021-01-03 15:25:29

	_id	bonusPointsEarned	bonusPointsEarnedReason	createDate	dateScanned
16	5ff1e1df0a7214ada1000564	750.0	Receipt number 1 completed, bonus point schedu...	2021-01-03 15:25:19	2021-01-03 15:25:19
17	5ff1e1b40a7214ada100055b	750.0	Receipt number 1 completed, bonus point schedu...	2021-01-03 15:24:36	2021-01-03 15:24:36
18	5ff1e1eb0a720f0523000576	300.0	Receipt number 4 completed, bonus point schedu...	2021-01-03 15:25:31	2021-01-03 15:25:31
19	5ff1e1c80a720f052300056d	5.0	All-receipts receipt bonus	2021-01-03 15:24:56	2021-01-03 15:24:56

- dropping the rewardsReceiptItemList column as we have a dedicated table (items) for this field.

In [ ]:
```
cleaned_receipts.drop(columns=['rewardsReceiptItemList'], inplace=True)
```

- Checking duplicate rows

In [ ]:
```
cleaned_receipts.duplicated().sum()
```

Out[ ]:
```
0
```

- There are many other data cleaning steps that can be performed on this table, but given the business requirements, this much data cleaning is sufficient. Other cleaning steps can be performed as per the client's/stakeholder's requirements.

In [ ]:
```
cleaned_receipts.to_csv('cleaned_receipts.csv', index=False)
```

Items

- Correcting the name of the ID field similar to receipts

In [ ]:
```
items.rename(columns={"_id_$oid": "_id"}, inplace=True)
```

In [ ]:
```
items.head()
```

Out[ ]:

	barcode	description	finalPrice	itemPrice	needsFetchReview	partnerItemId	preventTargetGa
0	4011	ITEM NOT FOUND	26.00	26.00	False	1	
1	4011	ITEM NOT FOUND	1	1	NaN	1	
2	028400642255	DORITOS TORTILLA CHIP SPICY SWEET CHILI REDUCE...	10.00	10.00	True	2	
3	NaN	NaN	NaN	NaN	False	1	
4	4011	ITEM NOT FOUND	28.00	28.00	False	1	

In [ ]:
```
items.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6941 entries, 0 to 6940
Data columns (total 36 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   barcode                               3090 non-null   object 
 1   description                           6560 non-null   object 
 2   finalPrice                            6767 non-null   object 
 3   itemPrice                             6767 non-null   object 
 4   needsFetchReview                      813 non-null    object 
 5   partnerItemId                         6941 non-null   object 
 6   preventTargetGapPoints                358 non-null    object 
 7   quantityPurchased                     6767 non-null   float64
 8   userFlaggedBarcode                    337 non-null    object 
 9   userFlaggedNewItem                    323 non-null    object 
 10  userFlaggedPrice                      299 non-null    object 
 11  userFlaggedQuantity                   299 non-null    float64
 12  needsFetchReviewReason                219 non-null    object 
 13  pointsNotAwardedReason                340 non-null    object 
 14  pointsPayerId                         1267 non-null   object 
 15  rewardsGroup                          1731 non-null   object 
 16  rewardsProductPartnerId               2269 non-null   object 
 17  userFlaggedDescription                205 non-null    object 
 18  originalMetaBriteBarcode              71 non-null     object 
 19  originalMetaBriteDescription          10 non-null     object 
 20  brandCode                             2600 non-null   object 
 21  competitorRewardsGroup                275 non-null    object 
 22  discountedItemPrice                   5769 non-null   object 
 23  originalReceiptItemText               5760 non-null   object 
 24  itemNumber                            153 non-null    object 
 25  originalMetaBriteQuantityPurchased    15 non-null     float64
 26  pointsEarned                          927 non-null    object 
 27  targetPrice                           378 non-null    object 
 28  competitiveProduct                    645 non-null    object 
 29  originalFinalPrice                    9 non-null      object 
 30  originalMetaBriteItemPrice            9 non-null      object 
 31  deleted                               9 non-null      object 
 32  priceAfterCoupon                      956 non-null    object 
 33  metabriteCampaignId                   863 non-null    object 
 34  _id                                   6941 non-null   object 
 35  userId                                6941 non-null   object 
dtypes: float64(3), object(33)
memory usage: 1.9+ MB
```

In [ ]:
```
items.nunique()
```

Out[ ]:
```
barcode                                568
description                           1889
finalPrice                             828
itemPrice                              828
needsFetchReview                         2
partnerItemId                          916
preventTargetGapPoints                   1
quantityPurchased                       13
userFlaggedBarcode                       6
userFlaggedNewItem                       1
userFlaggedPrice                        13
userFlaggedQuantity                      5
needsFetchReviewReason                   2
pointsNotAwardedReason                   1
pointsPayerId                           15
rewardsGroup                           182
rewardsProductPartnerId                 16
userFlaggedDescription                   3
originalMetaBriteBarcode                 6
originalMetaBriteDescription             2
brandCode                              227
competitorRewardsGroup                  30
discountedItemPrice                    817
originalReceiptItemText               1738
itemNumber                              47
originalMetaBriteQuantityPurchased       2
pointsEarned                           277
targetPrice                              2
competitiveProduct                       2
originalFinalPrice                       2
originalMetaBriteItemPrice               2
deleted                                  1
priceAfterCoupon                       334
metabriteCampaignId                     75
_id                                    679
userId                                 246
dtype: int64
```

In [ ]:
```
items.duplicated().sum()
```

Out[ ]:
```
0
```

- There won't be any cleaning with this table as the fields are not necessarily present for each receipt and user.
- The rows that had all the field values as NULL except the user and receipt IDs were already dropped during the table creation stage.
- For example: Not every item has a coupon applied to it, which results in a preiceAfterCoupon field for that particular item in the receipt.
- This table is created for easier access of fields required for the queries.

In [ ]:
```
items.to_csv('items.csv', index=False)
```

Exploring the Data before Queries

In [ ]:
```
brand_names = cleaned_brands['brandCode'].unique()
brand_names
```

Out[ ]:
```
array([nan, 'STARBUCKS', 'TEST BRANDCODE @1612366146176',
       'TEST BRANDCODE @1612366146051', 'TEST BRANDCODE @1612366146827',
       'TEST BRANDCODE @1612366146091', 'TEST BRANDCODE @1612366146133',
       'J.L. KRAFT', 'CAMPBELLS HOME STYLE', 'TEST',
       'TEST BRANDCODE @1598813526777', 'CALUMET', '511111205012',
       'AUNT JEMIMA SYRUP', 'MOLSON', 'LOTRIMIN',
       'TEST BRANDCODE @1597342520277', 'ST IVES', 'CHRISIMAGE',
       'ALKA SELTZER', "JACK DANIEL'S BARBECUE", 'MAGNUM ICE CREAM',
       '511111105329', 'TEST BRANDCODE @1598635634882', 'TACO BELL',
       'FROSTED CHEERIOS', 'TEST BRANDCODE @1598639199674',
       'GODIVA DRY PACKAGED DESSERTS', 'LARABAR',
       'TEST BRANDCODE @1597350074333', 'TEST BRANDCODE @1607636368717',
       'TEST BRANDCODE @1607707830095', 'COTTONELLE', 'IZZE', 'MIO',
       '511111505365', 'QUILTING SPECIAL EDITION',
       'TEST BRANDCODE @1604437351617', 'HERMAN', 'KEVITA', 'DELIMEX',
       'THE RIGHT TO SHOWER', 'CARESS', 'TEST BRANDCODE @1595340086044',
       'TEST BRANDCODE @1597861700968', 'TEST BRAND CODE',
       'TEST BRANDCODE @1598026275245', 'SWANSON',
       'TEST BRANDCODE @1598026274668', 'DASH-2249 1', '511111805854',
       'KETTLE BRAND', 'CAMPBELLS SOUP AT HAND', 'RAINBO', 'CRISPIN',
       'TEST BRANDCODE @1607707830137', 'CLARITIN®', 'COOKIE CRISP',
       'TEST BRANDCODE @1605535020442', 'TEST BRANDCODE @1604946244750',
       'SNYDERS OF HANOVER', 'GOOD SEASONS',
       'TEST BRANDCODE @1598633602279', 'REVOLVER', 'KEYSTONE LIGHT',
       'BETTY CROCKER FRUIT ROLL UPS', 'TEST BRANDCODE @1607639232356',
       'MOMOFUKU', 'POPSICLE', 'MUG ROOT BEER', 'TOMS',
       'TEST BRANDCODE @1606765579244', 'TEST BRANDCODE @1597342520305',
       'PROMISE', 'OFF THE EATEN PATH', 'TEST BRANDCODE @1600291350125',
       'TEST BRANDCODE @1603760319758', "NANCY'S",
       'CAMPBELLS CHUNKY MAXX', 'FOOD NETWORK KITCHEN INSPIRATIONS',
       'TEST BRANDCODE @1597935986390', 'TEST BRANDCODE @1598633693744',
       'SABRITAS', 'GEVALIA KAFFE', 'TEST BRANDCODE @1598290603501',
       "O THAT'S GOOD", "CHESTER'S", "MISS VICKIE'S", "SIR KENSINGTON'S",
       'DORITOS', 'A.1.', 'TEST BRANDCODE @1598290603587', 'LOOZA JUICES',
       'ALEVE', 'SOFT-SHEEN CARSON - HAIR COLOR',
       'TEST BRANDCODE @1604946245499', 'TEST BRANDCODE @1598639199117',
       'PULL UPS', 'TEST BRANDCODE @1605535020629', 'BRISK', 'CHRISXYZ',
       "BULL'S-EYE", 'KLONDIKE', 'FRUTTARE', '511111305569', 'KNOX',
       '511111505716', 'PLANTERS', 'DEGREE', "CALEB'S KOLA",
       'LEMON LEMON', 'VASELINE', '511111005216',
       'TEST BRANDCODE @1607312532178', 'PINESOL',
       'TEST BRANDCODE @1595340086791', '511111005148', 'ESSIE',
       '0987654321', 'CAMPBELLS WELL YES',
       'TEST BRANDCODE @1607312532304', 'TEST BRANDCODE @1598026274643',
       'TEST BRANDCODE @1601644881995', 'BEEF STEAK', "POND'S",
       'L’OREAL PARIS - COSMETICS', '511111605829',
       'TEST BRANDCODE @1595340086078', 'TEST BRANDCODE @1602694020787',
       'TEST BRANDCODE @1601939013649', 'SNACK FACTORY', 'MILLER64',
       "I CAN'T BELIEVE IT'S NOT BUTTER!", '511111805786',
       'TEST BRANDCODE @1610496647142', 'TEST BRANDCODE @1610496646289',
       'TEST BRANDCODE @1610496646345', 'TEST BRANDCODE @1610496646232',
       'TEST BRANDCODE @1610495466218', 'TEST BRANDCODE @1610495102462',
       'TEST BRANDCODE @1610495466250', 'KRAFT MIRACLE WHIP',
       'TEST BRANDCODE @1598042674173', 'TEST BRANDCODE @1598635634796',
       'SHEAF STOUT', 'CHEETOS', 'NATURES HARVEST', '511111605058',
       'TEST BRANDCODE @1598639198570', 'TEST BRANDCODE @1599097538609',
       'SCOTT', 'TEST BRANDCODE @1607639231674', 'EPIC', 'GROLSCH',
       'TEST BRANDCODE @1603760320407', 'MAGNUM',
       'TEST BRANDCODE @1603925787411', 'TEST BRANDCODE @1610495466861',
       'TEST BRANDCODE @1610495102526', 'TEST BRANDCODE @1610495102491',
```

'TEST BRANDCODE @1610495466175', 'TEST BRANDCODE @1610495103255',
'TEST BRANDCODE @1610495466285', 'TEST BRANDCODE @1610049748118',
'TEST BRANDCODE @1610049748181', 'TEST BRANDCODE @1610049748154',
'TEST BRANDCODE @1610049748724', 'TEST BRANDCODE @1610053719707',
'TEST BRANDCODE @1607987891932', 'TEST BRANDCODE @1607987891893',
'TEST BRANDCODE @1607987891852', 'TEST BRANDCODE @1607987892541',
'TEST BRANDCODE @1607707829962', "KILLIAN'S",
'TEST BRANDCODE @1597861700880', "KRAFT SHAKE 'N BAKE", 'HP',
'DRUMSTICK CEREAL', 'ALEVE D SINUS & COLD', 'CHEEZ WHIZ',
'TEST BRANDCODE @1607463685522', 'COOL WHIP', 'TRIX',
'FLINTSTONES', 'ABSOLUT® GRAPEFRUIT', 'ALTOS', 'DEL MAGUEY',
'JAMESON® BLACK BARREL', 'JAMESON', 'MONKEY 47', 'MALIBU',
'LILLET', '511111705161', 'AMP2', 'TEST BRANDCODE @1597861700940',
'TEST BRANDCODE @1604437351567', 'KELSEN', '511111405818',
'ALEVE DIRECT THERAPY PAIN RELIEF DEVICES', 'COLORADO NATIVE',
'SOBE', 'KNUDSEN', '511111105763', '511111005421', "CAP'N CRUNCH",
"MRS BAIRD'S", 'TRESEMME', 'PURE LEAF ICED TEA BEVERAGES',
'ONE A DAY KIDS', 'TEST BRANDCODE @1599159968998', 'LIFEWTR',
'511111804994', 'SIMILAC', 'CORICIDIN HBP',
'TEST BRANDCODE @1601939016290', 'BADEDAS',
'TEST BRANDCODE @1604946244876', 'BENIHANA', 'MOUNTAIN HIGH',
'TOSTITOS', 'TEST BRANDCODE @1603760319788', 'GERBER BABY FOOD',
'PACE', 'CAMPBELLS SLOW KETTLE', 'ARCHWAY', 'BAYER® ASPIRIN',
'TEST BRANDCODE @1610055793449', 'TEST BRANDCODE @1610055793411',
'TEST BRANDCODE @1610055793500', 'TEST BRANDCODE @1610055794196',
'TEST BRANDCODE @1610055793537', 'TEST BRANDCODE @1610058181549',
'TEST BRANDCODE @1610053719944', 'TEST BRANDCODE @1610053719998',
'TEST BRANDCODE @1610053719869', 'TEST BRANDCODE @1610053721033',
'DEPEND', 'TEST BRANDCODE @1608313322283',
'TEST BRANDCODE @1608313321374', 'TEST BRANDCODE @1608313321191',
'TEST BRANDCODE @1608313321301', 'TEST BRANDCODE @1608313321222',
'NATURE VALLEY', 'TEST BRANDCODE @1598633693767',
'TEST BRANDCODE @1598633693791', 'ANZIO', '511111705000',
"HERSHEY'S WHIPPED TOPPING", 'TEST BRANDCODE @1598633693720',
'TEST BRANDCODE @1605535020671', 'IMPERIAL', 'ALEVE PM NIGHT TIME',
'TEST BRANDCODE @1603925787294', 'APOTHECARE ESSENTIALS',
"HENRY WEINHARD'S", 'CAMPBELLS CHUNKY', 'ONE A DAY® WOMENS',
'TEST BRANDCODE @1595340085980', 'KINGSFORD', 'YOPLAIT GO-GURT',
'FRITO-LAY', 'GREY POUPON', 'TEST BRANDCODE @1598296704763',
'BERROCA®', 'PROPEL', "MILWAUKEE'S BEST", 'SPITZ', 'IMAGINE',
'TEST BRANDCODE @1597342520969', 'TUSK & GRAIN', "REDD'S ALE",
'LANCE', 'PACIFIC FOODS', 'YOPLAIT DUNKERS',
'TEST BRANDCODE @1598554813705', 'LIPTON SOUP', 'MINIONS CEREAL',
'LOVE HOME AND PLANET', 'TEST BRANDCODE @1595968943012',
'ONETOUCH', 'TEST BRANDCODE @1597935986248', 'KRAFT', 'BASIC 4',
'WOLFGANG PUCK', 'TEST BRANDCODE @1601659120828',
'TEST BRANDCODE @1603925787441', 'THE SOULFUL PROJECT',
'TEST BRANDCODE @1598633602924', 'DREAM WHIP',
'TEST BRANDCODE @1597527951461', 'TEST BRANDCODE @1607639231623',
'KJELDSEN', 'TEST BRANDCODE @1598635435896',
'TEST BRANDCODE @1599849713773', 'TEST BRANDCODE @1607636369405',
'TEST BRANDCODE @1601644363827', 'PEPPERIDGE FARM',
'TEST BRANDCODE @1611088503689', 'TEST BRANDCODE @1605535020580',
'MILLER HIGH LIFE', 'QUAKER', '511111705727', 'ATHENOS',
'TEST BRANDCODE @1599159969028', 'TEST BRANDCODE @1597350074404',
'SEDAL', 'RED ROCK DELI', 'ALKA SELTZER PLUS SINUS CAP/ GEL/ TAB',
'TEST BRANDCODE @1598633602253', 'SMARTFOOD',
'TEST BRANDCODE @1598042673532', 'TEST BRANDCODE @1601939762881',
'OSCAR MAYER', 'LECH PREMIUM', '511111805137', 'HEINERS',
'PARENTS', 'PIONEER WOMAN', 'TEST BRANDCODE @1611088504474',

'511111605102', 'HOP VALLEY', 'TEST BRANDCODE @1597861700914',
'YUBAN', '511111105114', 'TEST BRANDCODE @1603760319682',
'TEST BRANDCODE @1598716509394', 'TIGI', "AUTUMN'S GOLD",
'BROOKE BOND', '511111105046', 'TEST BRANDCODE @1595968943049',
'AMPTEST', 'COORS', 'BOLT 24', 'MIST TWST',
'TEST BRANDCODE @1608136484526', 'TEST BRANDCODE @1608136485398',
'TEST BRANDCODE @1608136484574', 'SARGENTO',
'TEST BRANDCODE @1608136484633', 'TEST BRANDCODE @1608136484689',
'TEST BRANDCODE @1610038521565', 'TEST BRANDCODE @1610038521677',
'TEST BRANDCODE @1610038521624', 'TEST BRANDCODE @1610038522559',
'TEST BRANDCODE @1610040576117', 'TEST BRANDCODE @1610040576672',
'TEST BRANDCODE @1610040576088', 'TEST BRANDCODE @1610040576148',
'TEST BRANDCODE @1610040576053', 'ALKA SELTZER PLUS',
'TEST BRANDCODE @1610653897113', 'TEST BRANDCODE @1610653894519',
'TEST BRANDCODE @1610660035659', 'TEST BRANDCODE @1610653894662',
'TEST BRANDCODE @1610653894371', 'TEST BRANDCODE @1601659122798',
'COCOA PUFFS', 'TEST BRANDCODE @1601659120793', 'JAYS', 'KNORR',
'CRISTAL', 'TEST BRANDCODE @1607312532874', 'BOCA', 'ALEXA', 'KIX',
'TEST BRANDCODE @1595531348246', "GRANDMA'S",
'TEST BRANDCODE @1602694015178', "SHARP'S NEAR BEER", 'BAILEYS',
'FINISH', 'TEST BRANDCODE @1598639197926',
'TEST BRANDCODE @1602694015743', 'JELL-O',
'TEST BRANDCODE @1601644364536', 'TEST BRANDCODE @1603925787997',
'GOLDFISH', 'TEST BRANDCODE @1598633602304',
'TEST BRANDCODE @1601659120694', 'CAPE LINE',
'TEST BRANDCODE @1597935987233', 'NEAR EAST', '511111505150',
'CLOROX', 'MAY-BUD', 'TEST BRANDCODE @1599849713798', 'GATORADE',
'TEST BRANDCODE @1597342520132', 'GARDEN FRESH GOURMET',
'AUNT JEMIMA DRY BREAKFAST MIXES', 'OLDE ENGLISH',
'TEST BRANDCODE @1600291349255', 'MATADOR', '511111305286',
"REESE'S PUFFS", 'GOLDEN GRAHAMS', 'TEST BRANDCODE @1598026274692',
'511111805342', 'TEST BRANDCODE @1598554813744', 'RICE-A-RONI',
'STELLA DORO', 'TEST BRANDCODE @1598813526686',
'TEST BRANDCODE @1604437351678', 'CAMPBELLS READY MEALS',
'RESOLVE', 'MAUI STYLE', 'CULTURE REPUBLICK', 'WHEATIES',
'TEST BRANDCODE @1601644882062', 'TEST BRANDCODE @1601939013444',
'511111805571', 'TEST BRANDCODE @1598635634821',
'TEST BRANDCODE @1607707830844', 'ZIMA',
'TEST BRANDCODE @1598026274609', 'CHESTERS',
'TEST BRANDCODE @1601644363784', 'CAPE COD',
'COUNTRY CROCK PLANT BUTTER', '511111205388',
'CHOCOLATE LUCKY CHARMS', 'TEST BRANDCODE @1598554813535',
'TEST BRANDCODE @1610493496975', 'TEST BRANDCODE @1610495102411',
'TEST BRANDCODE @1610494831056', 'TEST BRANDCODE @1610494831082',
'TEST BRANDCODE @1610494831023', 'TEST BRANDCODE @1610493497005',
'TEST BRANDCODE @1610493497517', 'TEST BRANDCODE @1610493497034',
'TEST BRANDCODE @1610494830985', 'TEST BRANDCODE @1610494831583',
'TEST BRANDCODE @1608313051339', 'TEST BRANDCODE @1608313051291',
'TEST BRANDCODE @1608313052049', 'TEST BRANDCODE @1608313051133',
'TEST BRANDCODE @1608313051244', 'BALL PARK',
'TEST BRANDCODE @1598290603618', 'HIDDEL BARREL COLLECTION',
'LAUNCH BOX', 'SIMPLE', 'SALON SELECTIVES', 'CRACKER BARREL',
'HUGGIES LITTLE SWIMMERS', 'TEST BRANDCODE @1598042673502',
'TEST BRANDCODE @1607707830173', 'SHREDS', 'BAKEN ETS',
'HONEY NUT CHEERIOS', 'DASH-2249 1 BRAND 1', 'CRACKER JACK',
'GERBER GOOD START', 'TEST BRANDCODE @1598554813654',
'TEST BRANDCODE @1598635436505', 'TEST BRANDCODE @1597527951436',
'BETTER HOMES & GARDENS', 'BIMBO', "LEINENKUGEL'S", 'NESTLE NAN',
'TEST BRANDCODE @1598042673466', 'JET-PUFFED',
'TEST BRANDCODE @1601644882548', 'HEALTH',

'TEST BRANDCODE @1601644882019', 'TEST BRANDCODE @1601644363876',
'511111705444', 'MUNCHIES', '511111605775',
'TEST BRANDCODE @1607463684816', 'TEST BRANDCODE @1598633694347',
'TEST BRANDCODE @1598296704639', "ANNIE'S",
'TEST BRANDCODE @1598881562991', 'TEST BRANDCODE @1598813526656',
'MIDOL', 'TWISTED RANCH', 'EMERALD',
'TEST BRANDCODE @1598881562500', 'TEST BRANDCODE @1598711015538',
'HERSHEYS HOT COCOA', 'PG TIPS', 'MUNCHOS', 'LUNCHABLES',
'WOOLITE', '511111104971', 'HUGGIES',
'TEST BRANDCODE @1598711015353', 'TEST BRANDCODE @1605535021531',
'BISCA', 'TEST BRANDCODE @1598881563900',
'TEST BRANDCODE @1601939762909', 'PERONI',
'TEST BRANDCODE @1598554814581', 'TEST BRANDCODE @1607463684722',
'BETTY CROCKER GUSHERS', 'TEST BRANDCODE @1597861701615',
'LUCKY CHARMS', 'BACK TO NATURE DINNERS',
'TEST BRANDCODE @1597350074366', 'TEST BRANDCODE @1606765578747',
'511111305125', 'GRANDMA SYCAMORE', 'MERMAID CEREAL',
'TEST BRANDCODE @1597350074237', 'AQUAFINA', '511111905240',
'TEST BRANDCODE @1599849713825', 'CAMPBELLS DINNER SAUCES',
'WILD STYLE', 'DOVE BODY', 'TEST BRANDCODE @1601939012950',
'TEST BRANDCODE @1597342520238', 'WEIGHT WATCHERS SMART ONES',
'ALEVE LIQUID GELS', 'SANTITAS', 'FRUIT LOVE', 'TROPICANA',
'RAISIN NUT BRAN', '511111405696', 'VELVEETA', 'GOOD HUMOR',
'JUST CRACK AN EGG', 'TEST BRANDCODE @1604946244789',
'511111305071', 'TEST BRANDCODE @1598813526618', 'PEPSI', "MOTT'S",
'RIBERHUS', "HERSHEY'S PUDDING", 'TEST BRANDCODE @1599097538395',
'TEST BRANDCODE @1607636368260', 'GO & GROW',
'TEST BRANDCODE @1595968943087', 'COUNTRY CROCK', "STACY'S",
"MR. YOSHIDA'S", 'TEST BRANDCODE @1604437352151',
'TEST BRANDCODE @1595531348410', 'TEST BRANDCODE @1598296705444',
'LEA & PERRINS', 'HEINZ', 'SARA LEE',
'TEST BRANDCODE @1602694015481', 'OVEN FRY',
'TEST BRANDCODE @1598716510157', 'TWO HATS',
'STARBUCKS BOTTLED DRINKS', 'FRITOS', 'PASTA RONI', 'BRAND CODE',
'ALLRECIPES', 'V8 V FUSION', 'TEST BRANDCODE @1599097538537',
'H20H', 'MILLER GENUINE DRAFT', 'TEST BRANDCODE @1603760319814',
'AMAZING GRAINS', 'TEST BRANDCODE @1595531348374',
'CHOCOLATE CHEERIOS', 'ONE', 'KEYSTONE ICE', 'DIGIORNO CHEESE',
'CALEBS KOLA', 'TEST BRANDCODE @1611088503776', 'PILSNER URQUELL',
'TEST BRANDCODE @1597935986474', 'MILANO', 'GENERAL MILLS CEREAL',
'TEST BRANDCODE @1598635635448', 'CLEAR',
'ARNOLD PALMER SPIKED HALF & HALF ORIGINAL', 'ABSOLUT® LIME',
'ABSOLUT® JUICE STRAWBERRY', 'ABSOLUT® ORIGINAL',
'ABSOLUT® MANDRIN', 'ABSOLUT ELYX', 'ABSOLUT® JUICE APPLE',
'ABERLOUR', 'JAMESON COLD BREW', 'JEFFERSON’S RESERVE',
'CAMPO VIEJO', 'KAHLUA', 'MUMM NAPA', 'MALFY', 'MALIBU® LIME',
'MALIBU® STRAWBERRY', 'REDBREAST 12YO',
'TEST BRANDCODE @1597527951412', 'SOL',
'TEST BRANDCODE @1599159969725', "WYLER'S",
'TEST BRANDCODE @1601939762943', 'FAT RABBIT', 'TOTAL',
'TEST BRANDCODE @1598635435829', 'TEST BRANDCODE @1607708191093',
'MALIBU® PINEAPPLE', 'PURE BLISS', 'ROYAL DANSK',
'TEST BRANDCODE @1598711015578', 'DEVOUR', 'CAMPBELLS',
'TEST BRANDCODE @1595340086011', 'STUBBORN SODA', 'AVION',
'TEST BRANDCODE @1610039590413', 'TEST BRANDCODE @1610039590349',
'TEST BRANDCODE @1610039590443', 'TEST BRANDCODE @1610039590383',
'TEST BRANDCODE @1610039590990', 'TEST BRANDCODE @1610495622485',
'TEST BRANDCODE @1610496646104', 'TEST BRANDCODE @1610495622510',
'TEST BRANDCODE @1610495623019', 'TEST BRANDCODE @1610495622536',
'TEST BRANDCODE @1610495622560', 'ONE A DAY', '511111705239',

'COUNTRY CROCK ORIGINAL', "HELLMANN'S/BEST FOODS",
'TEST BRANDCODE @1597527951382', 'TEST BRANDCODE @1601644882040',
'TEST BRANDCODE @1597527952048', 'TEST BRANDCODE @1595968943787',
'TALENTI', 'TEST BRANDCODE @1598635634767', '511111305842',
'RED DOG', 'SMITH & FORGE', 'LIPTON ICED TEA BEVERAGES', 'RUFFLES',
'V8 PLUS ENERGY', 'POP SECRET', 'TEST BRANDCODE @1601939763539',
'CAPRI SUN', 'NUT HARVEST', 'TEST BRANDCODE @1598881563437',
'TEST BRANDCODE @1600291349208', 'HENRY WEINHARDS',
'TEST BRANDCODE @1599097539367', '511111905035', 'FRUITY CHEERIOS',
'FLINTSTONES MULTIVITAMIN GUMMY', "FOSTER'S",
'F WHITLOCK AND SONS', 'TEST BRANDCODE @1610046687007',
'TEST BRANDCODE @1610046687035', 'TEST BRANDCODE @1610046686980',
'TEST BRANDCODE @1599097538572', "I CAN'T BELIEVE IT'S NOT BUTTER",
'TEST BRANDCODE @1607463684777', 'BAKEN-ETS', 'LIBERTE',
'APPLE CINNAMON CHEERIOS', 'ST STEFANUS',
'DOLE CHILLED FRUIT JUICES', 'SUAVE', 'CLASSICO', 'STROEHMANN',
'BUBLY', '1915 BOLTHOUSE FARMS', 'TEST BRANDCODE @1610046687624',
'TEST BRANDCODE @1610046686954', 'TEST BRANDCODE @1610049748079',
'MIRALAX LAXATIVES', 'TEST BRANDCODE @1607708191128',
'TEST BRANDCODE @1607708191160', 'TEST BRANDCODE @1607708191197',
'TEST BRANDCODE @1607708191721', 'TEST BRANDCODE @1610562208718',
'TEST BRANDCODE @1610653894219', 'TEST BRANDCODE @1610562208600',
'TEST BRANDCODE @1610562208645', 'TEST BRANDCODE @1610562209325',
'TEST BRANDCODE @1610562208680', 'TEST BRANDCODE @1610660035827',
'TEST BRANDCODE @1610660035788', 'TEST BRANDCODE @1610660036398',
'TEST BRANDCODE @1610718082494', "T.G.I. FRIDAY'S", 'CUSQUENA',
'ARNOLD', 'MC CAFE', 'O-KE-DOKE', 'TEST BRANDCODE @1607312532260',
'ALKA SELTZER ADULT HEARTBURN CHEWS/ GUMMY/ TAB',
'TEST BRANDCODE @1595531348450', 'TEST BRANDCODE @1598635435926',
'TEST BRANDCODE @1607636368136', 'OATMEAL CRISP',
'TEST BRANDCODE @1609794603283', "PHILLIPS'®DIGESTIVE",
'TEST BRANDCODE @1610660035741', 'TEST BRANDCODE @1609794603237',
'TEST BRANDCODE @1609794603367', 'TEST BRANDCODE @1609794603327',
'TEST BRANDCODE @1609794603942', 'TEST BRANDCODE @1610038521409',
'TEST BRANDCODE @1610718082601', 'TEST BRANDCODE @1610718083279',
'TEST BRANDCODE @1610718082685', 'TEST BRANDCODE @1610718082644',
'TEST BRANDCODE @1610135036684', 'TEST BRANDCODE @1610493496943',
'TEST BRANDCODE @1610493089082', 'TEST BRANDCODE @1610135035614',
'TEST BRANDCODE @1610135035679', 'TEST BRANDCODE @1610135035546',
'TEST BRANDCODE @1610493088287', 'TEST BRANDCODE @1610493088459',
'TEST BRANDCODE @1610493088421', 'TEST BRANDCODE @1610493088494',
'TEST BRANDCODE @1610058181760', 'TEST BRANDCODE @1610135035332',
'TEST BRANDCODE @1610058182453', 'TEST BRANDCODE @1610058181682',
'TEST BRANDCODE @1610058181719', 'HERSHEYS CEREAL',
'TEST BRANDCODE @1598296704728', 'AMP', '511111905318', 'AXE',
'TEST BRANDCODE @1598635435855', 'TEST BRANDCODE @1606765582946',
"ALEXA-O'S", '511111005704', '511111805069', 'STEEL RESERVE',
'POP WORKS & COMPANY', 'SABRITONES', '511111205456', 'SURE-JELL',
'TERRAPIN', 'SMART MADE', '511111405023', 'LOVE BEAUTY AND PLANET',
'PROTEIN ONE', '511111605492', 'TEST BRANDCODE @1603925787381',
'KOOL-AID', 'TEST BRANDCODE @1601939013241', 'ABSOLUT® CITRON',
'AVION R44', 'TEST BRANDCODE @1607639231710', 'MILLER FORTUNE',
'LEVER 2000', 'A+D', 'PULLUPS', 'TEST BRANDCODE @1606765578435',
'MOUNTAIN DEW', 'SUN CHIPS', 'TEST BRANDCODE @1598813527796',
'PURE LEAF TEA LEAVES', 'CORN NUTS', "BREAKSTONE'S", 'SPARKS',
'TEST BRANDCODE @1598716509358', 'MAXWELL HOUSE',
'TEST BRANDCODE @1599849714378', 'TEST BRANDCODE @1598711015496',
'CONTINENTAL SOUP', 'GARNIER - HAIR COLOR', 'AFRIN® NASAL SPRAY',
'GOODBELLY', 'NEXXUS', 'V8', "COLMAN'S", 'PREGO',
'TEST BRANDCODE @1606765578985', 'BRUMMEL & BROWN', 'YQ YOPLAIT',

```
'511111805298', 'TEST BRANDCODE @1604946244833',
'ALKA SELTZER COLD AND FLU TAB/ CAPS/ GEL',
'TEST BRANDCODE @1595531349206', '511111905806', '09090909090',
'511111105831', 'DIETCHRIS2', 'BOLD BUTCHER', "LAY'S",
"BEN & JERRY'S", 'TEST BRANDCODE @1598716509304', 'FRS920',
'TEST BRANDCODE @1598042673378', 'V8 HYDRATE', 'COUNTRY TIME',
'TEST BRANDCODE @1611088503732', 'NAKED JUICE', 'BREYERS',
'TEST BRANDCODE @1604437351646', '511111105275',
'GIRL SCOUT CEREAL', 'MILLER LITE', 'THIRD SHIFT', 'GOODNITES',
'TEST BRANDCODE @1599159969061', 'TEST BRANDCODE @1604434433706',
'MAYBELLINE', 'YOPLAIT', 'TEST BRANDCODE @1598290604214',
'TEST BRANDCODE @1607463684591', 'L'OREAL PARIS - HAIR COLOR',
'ALEVE CAPLETS, TABLETS, AND GEL CAPS', '511111605263',
'TEST BRANDCODE @1598633602226', 'TEST BRANDCODE @1607987891733',
'SAINT ARCHER', 'PHILADELPHIA', 'LATE JULY', 'FREIHOFERS',
'TYSKIE GRONIE', 'KLEENEX', 'TEST BRANDCODE @1601659120870',
'EL ISLENO', "MAISER'S", 'TEST BRANDCODE @1601939762976', 'VIVA',
'POISE', 'TEST BRANDCODE @1607639231747', 'ORE-IDA',
'511111605331', 'BIAGLUT', 'BITTEN', 'TANG', 'PLUM ORGANICS',
'THE GLENLIVET CARIBBEAN RESERVE',
"THE GLENLIVET® FOUNDER'S RESERVE", 'THE GLENLIVET® 12 YEAR',
'TASSIMO', 'DIPPIN DOTS', 'STOVE TOP', 'BROWNBERRY',
'TEST BRANDCODE @1601644363664', 'OUI BY YOPLAIT',
'CERVEZA AGUILA', 'TEST BRANDCODE @1598716507602',
'TEST BRANDCODE @1600291349297', 'ICEHOUSE', 'FDS',
'BOLTHOUSE FARMS', 'Q-TIPS', "HOFFMAN'S",
'BETTY CROCKER FRUIT BY THE FOOT', 'ZUMBIDA',
'TEST BRANDCODE @1598296704794', 'TEST BRANDCODE @1607636368759',
'KRUNCHERS', 'TEST BRANDCODE @1607312532218', 'FUNYUNS',
'CAMPBELLS SPAGHETTIOS', 'BRUMMEL AND BROWN',
'TEST BRANDCODE @1597350074997', 'TEST BRANDCODE @1598711019491',
'SIERRA MIST', 'TEST BRANDCODE @1598881572443',
'TEST BRANDCODE @1598290603646', 'TEST BRANDCODE @1599159969108',
'OROWEAT', 'ROLD GOLD', 'TEST BRANDCODE @1611088503815',
'TEST BRANDCODE @1602694015992', 'TEST BRANDCODE @1595968942896',
'V8 SPLASH', 'KOTEX', '511111505082', 'OCEAN SPRAY',
'511111305798', 'BAGEL BITES', 'POLLY-O', 'SUNSILK', "BAKER'S",
'ECCE PANIS', '511111405306', 'AIR WICK', "HAMM'S",
'TEST BRANDCODE @1599849713740', 'MONSTERS CEREAL',
'TAZO BOTTLED TEAS', 'TONI&GUY', '511111005377', '511111905479',
'TEST BRANDCODE @1597935986434', "MICKEY'S", 'CRYSTAL LIGHT',
'COORS EXTRA GOLD', 'D ITALIANO', 'TEST BRANDCODE @1600291349043',
'CLAUSSEN', 'BLUE MOON', 'DIPPIN DOTS CEREAL',
'TEST BRANDCODE @1598639215217', 'LIPTON TEA LEAVES',
'TEST BRANDCODE @1613158231644'], dtype=object)
```

In [ ]:
```
item_brands_names = items['brandCode'].unique()
item_brands_names
```

Out[ ]:
```
array([nan, 'MISSION', 'BRAND', 'KRAFT EASY CHEESE', 'PEPSI', 'DORITOS',
       'KLEENEX', 'WINGSTOP', 'GERM-X', 'BEN AND JERRYS', 'BORDEN',
       'KNORR', 'KLARBRUNN', 'HY-VEE', 'LIGHT & FIT GREEK',
       "CONNIE'S PIZZA", "VAN DE KAMP'S", 'HATCH FARMS', "KELLOGG'S",
       'TEMPTATIONS', "NATURE'S PATH ORGANIC", 'DOLE', 'EL MONTEREY',
       'BIGELOW', 'HY-VEE SELECT', 'KIKKOMAN', 'SPECIAL K', 'SWANSON',
       'YUBAN', 'HILLSHIRE FARM', 'JUST BARE', "LAURA'S LEAN BEEF",
       'CAL-ORGANIC FARMS', 'DOLE CHILLED FRUIT JUICES', "BUSH'S BEST",
       'FOLGERS', 'KASHI', 'LIPTON', 'KRAFT', 'GREEN GIANT',
       'HARVEST SNAPS', "THAT'S SMART!", 'TOSTITOS', 'ADVIL',
       'CHICKEN OF THE SEA', 'RICE-A-RONI', 'STARKIST', 'TIC TAC',
       'SO DELICIOUS', 'WONDERFUL', 'LIGHT & FIT', 'HANOVER',
       'HIDDEN VALLEY', 'DANNON', 'KETTLE BRAND', 'FAGE', 'ORAL-B GLIDE',
       "CAMPBELL'S", "FRENCH'S", 'CRISPIX', 'KING ARTHUR FLOUR',
       'KITCHEN BASICS', 'MCCORMICK', 'OLD EL PASO', 'PEPPERIDGE FARM',
       'STOVE TOP', 'ZESTA', 'AZTECA', 'BUNNY', 'NATURE VALLEY',
       'HONEY BUNCHES OF OATS', 'SIMPLE TRUTH ORGANIC', 'BOTA BOX',
       'DARE', 'LINDT', 'ARNOLD', 'ORGANIC ROOT STIMULATOR',
       'GREY POUPON', "MERKT'S", 'MORTON', 'FRONTERA', 'KARO', 'KLONDIKE',
       'CHEESE', 'CRACKER BARREL', "FLORIDA'S NATURAL", 'BLUE DIAMOND',
       'LUNDBERG FAMILY FARMS', 'NUTRI-GRAIN', 'QUAKER', 'DIGIORNO',
       'KEMPS', 'THAI KITCHEN', 'PHILADELPHIA', "THOMAS'", 'POWER CRUNCH',
       'GERBER', 'TACO BELL', 'ORGANICVILLE', 'PURINA ONE', 'DR PEPPER',
       'COTTONELLE', 'C&H', 'CHEERIOS', 'SIMPLE TRUTH', "STOUFFER'S",
       'PRINGLES', "HELLMANN'S/BEST FOODS", 'MCCORMICK GRILL MATES',
       'THOMAS ENGLISH MUFFINS', "MARTINELLI'S", 'COOL WHIP',
       "MARIE CALLENDER'S", "HEMPLER'S", 'JIMMY DEAN', 'SIGNATURE',
       'FRESH EXPRESS', 'MOUNTAIN DEW', 'JELL-O', 'CLASSICO', 'NABISCO',
       'RITZ TOASTED CHIPS', 'LUNCHABLES', 'RAGU', 'KRUSTEAZ',
       'OSCAR MAYER', 'PILLSBURY', 'TILLAMOOK', 'TYSON',
       'BEAR CREEK COUNTRY KITCHENS', 'BIC', 'GENERAL MILLS',
       "JACK LINK'S", 'PLANTERS', 'PROGRESSO', 'RENUZIT', "BRASWELL'S",
       'JOHNSONVILLE', 'OREO', 'THOMAS', 'FRANZ', 'BETTY CROCKER', 'CHEX',
       'CINNAMON TOAST CRUNCH', 'TROLLI', 'VELVEETA', 'ORBIT', 'WISHBONE',
       "WELCH'S", 'PEARLS', 'NUTELLA', 'KRAZY GLUE', 'PREGO',
       'VITAL FARMS ALFRESCO EGGS', 'OVALTINE', 'LA BANDERITA',
       'GRIMMWAY FARMS', 'EGGO', 'PACIFIC FOODS', 'SABRA',
       "MRS. RENFRO'S", "REESE'S", 'KIT KAT', 'ORE-IDA',
       'FORTUNE YAKISOBA', 'FINISH', 'POMPEIAN', 'JUST CRACK AN EGG',
       'SIMPLY POTATOES', 'CHIQUITA', 'CHEEZ-IT', 'FRESH STEP',
       'SIGNATURE KITCHEN', 'ENERGIZER MAX', "HERSHEY'S KISSES", 'KERR',
       'PRIVATE SELECTION', 'SKITTLES', 'KROGER', 'JOHN SOULES FOODS',
       'COLEMAN NATURAL', 'SPARKLING ICE', 'SARGENTO',
       "STOUFFER'S CLASSICS", "BOAR'S HEAD", "JACK DANIEL'S READY TO EAT",
       'ESSENTIAL EVERYDAY', "EGGLAND'S BEST", "MRS. CUBBISON'S",
       'GALLO FAMILY VINEYARDS', 'LITEHOUSE', 'CADBURY', 'TOP FLIGHT',
       'CHEETOS', 'LINDSAY', 'HEWLETT PACKARD', 'PLAYTEX',
       'CREST 3D WHITE', 'HORMEL', 'SMITHFIELD', 'JENNIE-O', 'COKE',
       'DIET COKE', 'V8', 'R.W. KNUDSEN', '7UP', 'HAWAIIAN',
       'TAYLOR FARMS', 'RITZ', 'ARROWHEAD', 'EDWARDS', 'TOLL HOUSE',
       'NESTLE', "SCHWEBEL'S", "MARIE'S", 'JELLY BELLY', 'KERRYGOLD',
       'CARAMELLO', 'CALIFIA FARMS', "FAMOUS DAVE'S", 'BANZA',
       'AMERICAN BEAUTY', 'DELI', 'HERITAGE FARM', 'ROSARITA', 'SHOPRITE',
       'HUGGIES', 'VIVA'], dtype=object)
```

In [ ]:
```
same_values = set(item_brands_names) & set(brand_names)
same_values
```

Out[ ]:
```
{'ARNOLD',
 'CHEETOS',
 'CLASSICO',
 'COOL WHIP',
 'COTTONELLE',
 'CRACKER BARREL',
 'DOLE CHILLED FRUIT JUICES',
 'DORITOS',
 'FINISH',
 'GREY POUPON',
 "HELLMANN'S/BEST FOODS",
 'HUGGIES',
 'JELL-O',
 'JUST CRACK AN EGG',
 'KETTLE BRAND',
 'KLEENEX',
 'KLONDIKE',
 'KNORR',
 'KRAFT',
 'LUNCHABLES',
 'MOUNTAIN DEW',
 'NATURE VALLEY',
 'ORE-IDA',
 'OSCAR MAYER',
 'PACIFIC FOODS',
 'PEPPERIDGE FARM',
 'PEPSI',
 'PHILADELPHIA',
 'PLANTERS',
 'PREGO',
 'QUAKER',
 'RICE-A-RONI',
 'SARGENTO',
 'STOVE TOP',
 'SWANSON',
 'TACO BELL',
 'TOSTITOS',
 'V8',
 'VELVEETA',
 'VIVA',
 'YUBAN',
 nan}
```

In [ ]:
```
cleaned_receipts.iloc[11]['totalSpent'].dtype
```

Out[ ]:
```
dtype('float64')
```

In [ ]:
```
cleaned_brands.head(20)
```

Out[ ]:

	barcode	category	categoryCode	name	topBrand	brandCode
0	511111019862	Baking	BAKING	test brand @1612366101024	0	NaN
1	511111519928	Beverages	BEVERAGES	Starbucks	0	STARBUCKS
2	511111819905	Baking	BAKING	test brand @1612366146176	0	TEST BRANDCODE @1612366146176
3	511111519874	Baking	BAKING	test brand @1612366146051	0	TEST BRANDCODE @1612366146051
4	511111319917	Candy & Sweets	CANDY_AND_SWEETS	test brand @1612366146827	0	TEST BRANDCODE @1612366146827
5	511111719885	Baking	BAKING	test brand @1612366146091	0	TEST BRANDCODE @1612366146091
6	511111219897	Baking	BAKING	test brand @1612366146133	0	TEST BRANDCODE @1612366146133
7	511111104810	Condiments & Sauces	NaN	J.L. Kraft	0	J.L. KRAFT
8	511111504412	Canned Goods & Soups	NaN	Campbell's Home Style	0	CAMPBELLS HOME STYLE
9	511111504788	Baking	BAKING	test	0	TEST
10	511111516354	Baking	BAKING	test brand @1598813526777	0	TEST BRANDCODE @1598813526777
11	511111102540	NaN	NaN	MorningStar	0	NaN
12	511111201076	Baking	BAKING	Calumet	0	CALUMET
13	511111205012	Magazines	MAGAZINES	Entertainment Weekly	0	511111205012
14	511111801801	Breakfast & Cereal	NaN	AUNT JEMIMA Syrup	0	AUNT JEMIMA SYRUP
15	511111202233	Beer Wine Spirits	BEER_WINE_SPIRITS	Molson Canadian	0	MOLSON
16	511111817376	Health & Wellness	HEALTHY_AND_WELLNESS	Lotrimin®	0	LOTRIMIN
17	511111515319	Baking	BAKING	test brand @1597342520277	0	TEST BRANDCODE @1597342520277
18	511111317364	Baking	BAKING	test brand @1605535049181	0	NaN
19	511111300700	Beauty	NaN	ST. IVES	0	ST IVES

In [ ]:
```
cleaned_receipts.to_csv('cleaned_receipts.csv', index=False)
```

In [ ]:
```
cleaned_receipts.describe()
```

Out[ ]:

	bonusPointsEarned	createDate	dateScanned	finishedDate	modifyDat
count	1119.000000	1119	1119	568	111
mean	116.137623	2021-01-28 02:09:41.600271616	2021-01-28 02:09:41.600272384	2021-01-19 12:10:05.020589568	2021-01-2 15:14:28.70304384
min	0.000000	2020-10-30 20:17:59	2020-10-30 20:17:59	2021-01-03 15:24:10	2021-01-0 15:24:1
25%	0.000000	2021-01-14 19:13:03.690499840	2021-01-14 19:13:03.690499840	2021-01-08 21:22:42.500000	2021-01-1 21:32:25.50000
50%	0.000000	2021-01-29 17:18:22	2021-01-29 17:18:22	2021-01-19 21:13:57.500000	2021-01-2 17:18:4
75%	45.000000	2021-02-07 13:20:13.736999936	2021-02-07 13:20:13.736999936	2021-01-27 17:42:13.500000	2021-02-0 13:20:13.73699993
max	750.000000	2021-03-01 23:17:34.772000	2021-03-01 23:17:34.772000	2021-02-26 22:36:25	2021-03-0 23:17:34.77200
std	240.243665	NaN	NaN	NaN	Nal

In [ ]:
```
```