Data Quality Issues

December 21, 2022

0.1 Evaluating different Data Quality Issues

[111]:
```
pip install pandasql
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting pandasql
  Downloading pandasql-0.7.3.tar.gz (26 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages
(from pandasql) (1.21.6)
Requirement already satisfied: pandas in /usr/local/lib/python3.8/dist-packages
(from pandasql) (1.3.5)
Requirement already satisfied: sqlalchemy in /usr/local/lib/python3.8/dist-
packages (from pandasql) (1.4.45)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.8/dist-
packages (from pandas->pandasql) (2022.6)
Requirement already satisfied: python-dateutil>=2.7.3 in
/usr/local/lib/python3.8/dist-packages (from pandas->pandasql) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-
packages (from python-dateutil>=2.7.3->pandas->pandasql) (1.15.0)
Requirement already satisfied: greenlet!=0.4.17 in
/usr/local/lib/python3.8/dist-packages (from sqlalchemy->pandasql) (2.0.1)
Building wheels for collected packages: pandasql
  Building wheel for pandasql (setup.py) … done
  Created wheel for pandasql: filename=pandasql-0.7.3-py3-none-any.whl
size=26787
sha256=15d1f190248bfb76766b1a21dd1e3e111101e255f5b17967d5d01d1f28870d9b
  Stored in directory: /root/.cache/pip/wheels/ed/8f/46/a383923333728744f01ba24a
dbd8e364f2cb9470a8b8e5b9ff
Successfully built pandasql
Installing collected packages: pandasql
Successfully installed pandasql-0.7.3
```

[112]:
```
import numpy as np
import pandas as pd
import pandasql as ps
```

[113]:
```
receipts = pd.read_json('/content/receipts.json', lines=True)
receipts.head()
```

[113]:

```
                                  _id  bonusPointsEarned  \
0  {'$oid': '5ff1e1eb0a720f0523000575'}              500.0
1  {'$oid': '5ff1e1bb0a720f052300056b'}              150.0
2  {'$oid': '5ff1e1f10a720f052300057a'}                5.0
3  {'$oid': '5ff1e1ee0a7214ada100056f'}                5.0
4  {'$oid': '5ff1e1d20a7214ada1000561'}                5.0

                          bonusPointsEarnedReason  \
0  Receipt number 2 completed, bonus point schedu…
1  Receipt number 5 completed, bonus point schedu…
2                         All-receipts receipt bonus
3                         All-receipts receipt bonus
4                         All-receipts receipt bonus

                createDate               dateScanned  \
0  {'$date': 1609687531000}  {'$date': 1609687531000}
1  {'$date': 1609687483000}  {'$date': 1609687483000}
2  {'$date': 1609687537000}  {'$date': 1609687537000}
3  {'$date': 1609687534000}  {'$date': 1609687534000}
4  {'$date': 1609687506000}  {'$date': 1609687506000}

               finishedDate                modifyDate  \
0  {'$date': 1609687531000}  {'$date': 1609687536000}
1  {'$date': 1609687483000}  {'$date': 1609687488000}
2                       NaN  {'$date': 1609687542000}
3  {'$date': 1609687534000}  {'$date': 1609687539000}
4  {'$date': 1609687511000}  {'$date': 1609687511000}

         pointsAwardedDate  pointsEarned             purchaseDate  \
0  {'$date': 1609687531000}         500.0  {'$date': 1609632000000}
1  {'$date': 1609687483000}         150.0  {'$date': 1609601083000}
2                       NaN           5.0  {'$date': 1609632000000}
3  {'$date': 1609687534000}           5.0  {'$date': 1609632000000}
4  {'$date': 1609687506000}           5.0  {'$date': 1609601106000}

   purchasedItemCount                              rewardsReceiptItemList  \
0                 5.0  [{'barcode': '4011', 'description': 'ITEM NOT …
1                 2.0  [{'barcode': '4011', 'description': 'ITEM NOT …
2                 1.0  [{'needsFetchReview': False, 'partnerItemId': …
3                 4.0  [{'barcode': '4011', 'description': 'ITEM NOT …
4                 2.0  [{'barcode': '4011', 'description': 'ITEM NOT …

  rewardsReceiptStatus  totalSpent                    userId
0             FINISHED         26.0  5ff1e1eacfcf6c399c274ae6
```

```
1             FINISHED        11.0  5ff1e194b6a9d73a3a9f1052
2             REJECTED        10.0  5ff1e1f1cfcf6c399c274b0b
3             FINISHED        28.0  5ff1e1eacfcf6c399c274ae6
4             FINISHED         1.0  5ff1e194b6a9d73a3a9f1052
```

[114]:
```
# Extract oid from dictionary, rename it to receipt_id
receipts['receipt_id'] = receipts['_id'].apply(lambda x: x['$oid'])
receipts = receipts.drop('_id', axis=1)
```

[115]:
```
# Extract date from the 6 date columns, and convert to datetime format
def extract_convert_date(col):
    '''Extract $date from dictionary, and convert to datetime format'''
    result = col.apply(lambda x: x['$date'] if type(x) != float else np.nan)
    result = pd.to_datetime(result, unit='ms')
    return result

for col in ['createDate', 'dateScanned', 'finishedDate', 'modifyDate',␣
↪'pointsAwardedDate', 'purchaseDate']:
    receipts[col] = extract_convert_date(receipts[col])

receipts.head()
```

[115]:
```
   bonusPointsEarned                          bonusPointsEarnedReason  \
0              500.0  Receipt number 2 completed, bonus point schedu…
1              150.0  Receipt number 5 completed, bonus point schedu…
2                5.0                          All-receipts receipt bonus
3                5.0                          All-receipts receipt bonus
4                5.0                          All-receipts receipt bonus

           createDate         dateScanned        finishedDate  \
0 2021-01-03 15:25:31 2021-01-03 15:25:31 2021-01-03 15:25:31
1 2021-01-03 15:24:43 2021-01-03 15:24:43 2021-01-03 15:24:43
2 2021-01-03 15:25:37 2021-01-03 15:25:37                 NaT
3 2021-01-03 15:25:34 2021-01-03 15:25:34 2021-01-03 15:25:34
4 2021-01-03 15:25:06 2021-01-03 15:25:06 2021-01-03 15:25:11

           modifyDate   pointsAwardedDate  pointsEarned        purchaseDate  \
0 2021-01-03 15:25:36 2021-01-03 15:25:31         500.0 2021-01-03 00:00:00
1 2021-01-03 15:24:48 2021-01-03 15:24:43         150.0 2021-01-02 15:24:43
2 2021-01-03 15:25:42                 NaT           5.0 2021-01-03 00:00:00
3 2021-01-03 15:25:39 2021-01-03 15:25:34           5.0 2021-01-03 00:00:00
4 2021-01-03 15:25:11 2021-01-03 15:25:06           5.0 2021-01-02 15:25:06

   purchasedItemCount                          rewardsReceiptItemList  \
0                 5.0  [{'barcode': '4011', 'description': 'ITEM NOT …
1                 2.0  [{'barcode': '4011', 'description': 'ITEM NOT …
2                 1.0  [{'needsFetchReview': False, 'partnerItemId': …
```

```
3                 4.0  [{'barcode': '4011', 'description': 'ITEM NOT …
4                 2.0  [{'barcode': '4011', 'description': 'ITEM NOT …

  rewardsReceiptStatus  totalSpent                     userId  \
0             FINISHED        26.0  5ff1e1eacfcf6c399c274ae6
1             FINISHED        11.0  5ff1e194b6a9d73a3a9f1052
2             REJECTED        10.0  5ff1e1f1cfcf6c399c274b0b
3             FINISHED        28.0  5ff1e1eacfcf6c399c274ae6
4             FINISHED         1.0  5ff1e194b6a9d73a3a9f1052

                receipt_id
0  5ff1e1eb0a720f0523000575
1  5ff1e1bb0a720f052300056b
2  5ff1e1f10a720f052300057a
3  5ff1e1ee0a7214ada100056f
4  5ff1e1d20a7214ada1000561
```

[116]:
```
items = []
for index, row in receipts.iterrows():
    if type(row['rewardsReceiptItemList']) != float:
        for item in row['rewardsReceiptItemList']:
            item['receipt_id'] = row['receipt_id']
            item['receipt_rewardsReceiptStatus'] = row['rewardsReceiptStatus']
            items.append(item)
items = pd.DataFrame(items)
items.head()
```

[116]:
```
        barcode                                        description finalPrice  \
0          4011                                 ITEM NOT FOUND      26.00
1          4011                                 ITEM NOT FOUND          1
2  028400642255  DORITOS TORTILLA CHIP SPICY SWEET CHILI REDUCE…      10.00
3           NaN                                            NaN        NaN
4          4011                                 ITEM NOT FOUND      28.00

  itemPrice needsFetchReview partnerItemId preventTargetGapPoints  \
0     26.00            False             1                   True
1         1              NaN             1                    NaN
2     10.00             True             2                   True
3       NaN            False             1                   True
4     28.00            False             1                   True

   quantityPurchased userFlaggedBarcode userFlaggedNewItem  … itemNumber  \
0                5.0               4011                True  …        NaN
1                1.0                NaN                 NaN  …        NaN
2                1.0       028400642255                True  …        NaN
3                NaN               4011                True  …        NaN
4                4.0               4011                True  …        NaN
```

```
   originalMetaBriteQuantityPurchased pointsEarned targetPrice  \
0                                 NaN          NaN         NaN
1                                 NaN          NaN         NaN
2                                 NaN          NaN         NaN
3                                 NaN          NaN         NaN
4                                 NaN          NaN         NaN

  competitiveProduct originalFinalPrice originalMetaBriteItemPrice deleted  \
0                NaN                NaN                        NaN     NaN
1                NaN                NaN                        NaN     NaN
2                NaN                NaN                        NaN     NaN
3                NaN                NaN                        NaN     NaN
4                NaN                NaN                        NaN     NaN

  priceAfterCoupon metabriteCampaignId
0              NaN                NaN
1              NaN                NaN
2              NaN                NaN
3              NaN                NaN
4              NaN                NaN

[5 rows x 36 columns]
```

[117]:
```
brands = pd.read_json('/content/brands.json', lines=True)
brands.head()
```

[117]:
```
                                  _id       barcode        category  \
0  {'$oid': '601ac115be37ce2ead437551'}  511111019862          Baking
1  {'$oid': '601c5460be37ce2ead43755f'}  511111519928       Beverages
2  {'$oid': '601ac142be37ce2ead43755d'}  511111819905          Baking
3  {'$oid': '601ac142be37ce2ead43755a'}  511111519874          Baking
4  {'$oid': '601ac142be37ce2ead43755e'}  511111319917  Candy & Sweets

       categoryCode                                                cpg  \
0            BAKING  {'$id': {'$oid': '601ac114be37ce2ead437550'}, …
1         BEVERAGES  {'$id': {'$oid': '5332f5fbe4b03c9a25efd0ba'}, …
2            BAKING  {'$id': {'$oid': '601ac142be37ce2ead437559'}, …
3            BAKING  {'$id': {'$oid': '601ac142be37ce2ead437559'}, …
4  CANDY_AND_SWEETS  {'$id': {'$oid': '5332fa12e4b03c9a25efd1e7'}, …

                        name  topBrand                      brandCode
0  test brand @1612366101024       0.0                            NaN
1                  Starbucks       0.0                      STARBUCKS
2  test brand @1612366146176       0.0  TEST BRANDCODE @1612366146176
3  test brand @1612366146051       0.0  TEST BRANDCODE @1612366146051
4  test brand @1612366146827       0.0  TEST BRANDCODE @1612366146827
```

[118]:
```
brands['brand_id'] = brands['_id'].apply(lambda x: x['$oid'])
brands = brands.drop('_id', axis=1)
```

[119]:
```
brands['cpg_id'] = brands['cpg'].apply(lambda x: x['$id']['$oid'])
brands['cpg_ref'] = brands['cpg'].apply(lambda x: x['$ref'])
brands = brands.drop('cpg', axis=1)
brands.head()
```

[119]:

	barcode	category	categoryCode	name	topBrand	brandCode	brand_id	cpg_id	cpg_ref
0	511111019862	Baking	BAKING	test brand @1612366101024	0.0	NaN	601ac115be37ce2ead437551	601ac114be37ce2ead437550	Cogs
1	511111519928	Beverages	BEVERAGES	Starbucks	0.0	STARBUCKS	601c5460be37ce2ead43755f	5332f5fbe4b03c9a25efd0ba	Cogs
2	511111819905	Baking	BAKING	test brand @1612366146176	0.0	TEST BRANDCODE @1612366146176	601ac142be37ce2ead43755d	601ac142be37ce2ead437559	Cogs
3	511111519874	Baking	BAKING	test brand @1612366146051	0.0	TEST BRANDCODE @1612366146051	601ac142be37ce2ead43755a	601ac142be37ce2ead437559	Cogs
4	511111319917	Candy & Sweets	CANDY_AND_SWEETS	test brand @1612366146827	0.0	TEST BRANDCODE @1612366146827	601ac142be37ce2ead43755e	5332fa12e4b03c9a25efd1e7	Cogs

[120]:
```
users = pd.read_json('/content/users.json', lines=True)
users.head()
```

[120]:

	_id	active	createdDate	lastLogin	role	signUpSource	state
0	{'$oid': '5ff1e194b6a9d73a3a9f1052'}	True	{'$date': 1609687444800}	{'$date': 1609687537858}	consumer	Email	WI
1	{'$oid': '5ff1e194b6a9d73a3a9f1052'}	True	{'$date': 1609687444800}	{'$date': 1609687537858}	consumer	Email	WI
2	{'$oid': '5ff1e194b6a9d73a3a9f1052'}	True	{'$date': 1609687444800}	{'$date': 1609687537858}	consumer	Email	WI
3	{'$oid': '5ff1e1eacfcf6c399c274ae6'}	True	{'$date': 1609687530554}	{'$date': 1609687530597}	consumer	Email	WI
4	{'$oid': '5ff1e194b6a9d73a3a9f1052'}	True	{'$date': 1609687444800}	{'$date': 1609687537858}	consumer	Email	WI

[121]:
```
users['user_id'] = users['_id'].apply(lambda x: x['$oid'])
users = users.drop('_id', axis=1)
```

[122]:
```
for col in ['createdDate', 'lastLogin']:
    users[col] = extract_convert_date(users[col])
```

[123]:
```
users.head()
```

[123]:

	active	createdDate	lastLogin	role	\
0	True	2021-01-03 15:24:04.800	2021-01-03 15:25:37.857999872	consumer	
1	True	2021-01-03 15:24:04.800	2021-01-03 15:25:37.857999872	consumer	
2	True	2021-01-03 15:24:04.800	2021-01-03 15:25:37.857999872	consumer	
3	True	2021-01-03 15:25:30.554	2021-01-03 15:25:30.596999936	consumer	
4	True	2021-01-03 15:24:04.800	2021-01-03 15:25:37.857999872	consumer	

	signUpSource	state	user_id
0	Email	WI	5ff1e194b6a9d73a3a9f1052
1	Email	WI	5ff1e194b6a9d73a3a9f1052
2	Email	WI	5ff1e194b6a9d73a3a9f1052
3	Email	WI	5ff1e1eacfcf6c399c274ae6
4	Email	WI	5ff1e194b6a9d73a3a9f1052

0.1.1 1. Missing Data Check

Having missing data is the most common kind of data quality issue and it is imperative to learn if the data considered for analysis has any missing values or not, else it can render wrong results. Thus, I ran an analysis to check whether or not NaN/NULL values are present in the given dataframes and I found the first data quality issue, that is, significant amount of data is missing in the given dataframes. For this purpose, I decided to find what percent of data is missing in individual dataframes that were provided. Below is the code that I wrote to find the same.

[82]:
```
missing_data = pd.DataFrame([], columns = ['column_name', 'missing_percent',␣
↪'table'])
```

[83]:
```
print('Average Missing Data in Receipts', receipts.isna().mean().mean() * 100)
print('Average Missing Data in Brands', brands.isna().mean().mean() * 100)
print('Average Missing Data in Users', users.isna().mean().mean() * 100)
```

```
Average Missing Data in Receipts 27.405421507298183
Average Missing Data in Brands 15.719318290012374
Average Missing Data in Users 4.790764790764791
```

It is found that 27.4%, 15.71%, and 4.7% of data is missing on an average in the Receipts, Brands, and Users dataframes respectively. Further, I decided to study that what columns from individual tables contribute to this missing proportion. Below is the code for the same.

[84]:
```
missing_r = dict( (receipts.isna().sum() / receipts.shape[0]) * 100)
missing_b = dict( (brands.isna().sum() / brands.shape[0]) * 100)
missing_u = dict( (users.isna().sum() / users.shape[0]) * 100)
```

[85]:
```
for key, value in missing_r.items():
  row = [key, value, 'receipts']
  missing_data.loc[len(missing_data), :] = row

for key, value in missing_b.items():
  row = [key, value, 'brands']
  missing_data.loc[len(missing_data), :] = row

for key, value in missing_u.items():
  row = [key, value, 'users']
  missing_data.loc[len(missing_data), :] = row
```

[212]:
```
missing_data.sort_values('missing_percent', ascending = False)
```

[212]:

	column_name	missing_percent	table
17	categoryCode	55.698372	brands
19	topBrand	52.442159	brands
6	pointsAwardedDate	52.010724	receipts
0	bonusPointsEarned	51.385165	receipts
1	bonusPointsEarnedReason	51.385165	receipts
4	finishedDate	49.240393	receipts
7	pointsEarned	45.576408	receipts
9	purchasedItemCount	43.252904	receipts
8	purchaseDate	40.035746	receipts
10	rewardsReceiptItemList	39.320822	receipts
12	totalSpent	38.873995	receipts
20	brandCode	20.051414	brands
16	category	13.281919	brands
26	lastLogin	12.525253	users
29	state	11.313131	users
28	signUpSource	9.69697	users
22	cpg_id	0.0	brands
25	createdDate	0.0	users
24	active	0.0	users
27	role	0.0	users
23	cpg_ref	0.0	brands
15	barcode	0.0	brands
21	brand_id	0.0	brands
18	name	0.0	brands
14	receipt_id	0.0	receipts
13	userId	0.0	receipts
11	rewardsReceiptStatus	0.0	receipts
5	modifyDate	0.0	receipts

```
3              dateScanned            0.0  receipts
2               createDate            0.0  receipts
30                 user_id            0.0     users
```

The above table represents the percentage of missing data for each tables that is contributed by individual columns. It is important to handle these missing values by either dropping the rows or replacing them according to the use-case as the presence of missing rows can either throw errors in computation or provides wrong outputs.

0.1.2 2. Duplicate Data Check

Next, it is possible in many cases that dataframes contains repeated entries for similar records thus creating duplicate data. In the process of testing this, I found that there are 70 different user_ids in user table that has duplicate entries. For demonstration purpose, I have displayed the duplicate rows in the users table for the user_id 54943462e4b07e684157a532. The rows are duplicate and provides redundant information. This issue can create bias in the data and can not only increase the computation but can give misleading results as well. Following is the python code I wrote to identify the same.

[96]:
```
users['user_id'].value_counts().loc[lambda x: x > 1]
```

[96]:
```
54943462e4b07e684157a532    20
5fc961c3b8cfca11a077dd33    20
5ff5d15aeb7c7d12096d91a2    18
5fa41775898c7a11a6bcef3e    18
59c124bae4b0299e55b0f330    18
                            ..
60229990b57b8a12187fe9e0     2
601c2c05969c0b11f7d0b097     2
5ff73b90eb7c7d31ca8a452b     2
5ffcb47d04929111f6e9256c     2
5ff7268eeb7c7d12096da2a9     2
Name: user_id, Length: 70, dtype: int64
```

[91]:
```
users[users['user_id'] == '54943462e4b07e684157a532']
```

[91]:
```
     active             createdDate               lastLogin         role  \
475    True 2014-12-19 14:21:22.381 2021-03-05 16:52:23.204  fetch-staff
476    True 2014-12-19 14:21:22.381 2021-03-05 16:52:23.204  fetch-staff
477    True 2014-12-19 14:21:22.381 2021-03-05 16:52:23.204  fetch-staff
478    True 2014-12-19 14:21:22.381 2021-03-05 16:52:23.204  fetch-staff
479    True 2014-12-19 14:21:22.381 2021-03-05 16:52:23.204  fetch-staff
480    True 2014-12-19 14:21:22.381 2021-03-05 16:52:23.204  fetch-staff
481    True 2014-12-19 14:21:22.381 2021-03-05 16:52:23.204  fetch-staff
482    True 2014-12-19 14:21:22.381 2021-03-05 16:52:23.204  fetch-staff
483    True 2014-12-19 14:21:22.381 2021-03-05 16:52:23.204  fetch-staff
484    True 2014-12-19 14:21:22.381 2021-03-05 16:52:23.204  fetch-staff
```

```
485    True 2014-12-19 14:21:22.381 2021-03-05 16:52:23.204  fetch-staff
486    True 2014-12-19 14:21:22.381 2021-03-05 16:52:23.204  fetch-staff
487    True 2014-12-19 14:21:22.381 2021-03-05 16:52:23.204  fetch-staff
488    True 2014-12-19 14:21:22.381 2021-03-05 16:52:23.204  fetch-staff
489    True 2014-12-19 14:21:22.381 2021-03-05 16:52:23.204  fetch-staff
490    True 2014-12-19 14:21:22.381 2021-03-05 16:52:23.204  fetch-staff
491    True 2014-12-19 14:21:22.381 2021-03-05 16:52:23.204  fetch-staff
492    True 2014-12-19 14:21:22.381 2021-03-05 16:52:23.204  fetch-staff
493    True 2014-12-19 14:21:22.381 2021-03-05 16:52:23.204  fetch-staff
494    True 2014-12-19 14:21:22.381 2021-03-05 16:52:23.204  fetch-staff
```

	signUpSource	state	user_id
475	NaN	NaN	54943462e4b07e684157a532
476	NaN	NaN	54943462e4b07e684157a532
477	NaN	NaN	54943462e4b07e684157a532
478	NaN	NaN	54943462e4b07e684157a532
479	NaN	NaN	54943462e4b07e684157a532
480	NaN	NaN	54943462e4b07e684157a532
481	NaN	NaN	54943462e4b07e684157a532
482	NaN	NaN	54943462e4b07e684157a532
483	NaN	NaN	54943462e4b07e684157a532
484	NaN	NaN	54943462e4b07e684157a532
485	NaN	NaN	54943462e4b07e684157a532
486	NaN	NaN	54943462e4b07e684157a532
487	NaN	NaN	54943462e4b07e684157a532
488	NaN	NaN	54943462e4b07e684157a532
489	NaN	NaN	54943462e4b07e684157a532
490	NaN	NaN	54943462e4b07e684157a532
491	NaN	NaN	54943462e4b07e684157a532
492	NaN	NaN	54943462e4b07e684157a532
493	NaN	NaN	54943462e4b07e684157a532
494	NaN	NaN	54943462e4b07e684157a532

0.1.3 3. Inconsistent Data

Data Inconsistency 1: The following data inconsistency is to check whether the date when the receipts were scanned is before the purchase date mentioned on the receipts or not. For the same I have written the following python code.

[144]:
```
temp = receipts[['dateScanned', 'purchaseDate']]
```

[150]:
```
temp['after_scan_purchase_check'] = temp['dateScanned'] < temp['purchaseDate']
temp['after_scan_purchase_check'] = temp['after_scan_purchase_check'].
 ↪astype(int)
```

```
<ipython-input-150-15f20f3f61c1>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  temp['after_scan_purchase_check'] = temp['dateScanned'] < temp['purchaseDate']
<ipython-input-150-15f20f3f61c1>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  temp['after_scan_purchase_check'] =
temp['after_scan_purchase_check'].astype(int)
```

[153]:
```
temp[temp['after_scan_purchase_check'] == 1]
```

[153]:

	dateScanned	purchaseDate	after_scan_purchase_check
12	2021-01-03 15:24:38	2021-02-03 15:24:38	1
14	2021-01-03 15:24:34	2021-02-03 15:24:35	1
85	2021-01-05 20:39:00	2021-02-05 20:39:00	1
139	2021-01-07 16:50:41	2021-02-07 16:50:41	1
158	2021-01-08 15:02:09	2021-02-08 15:02:10	1
190	2021-01-11 20:26:56	2021-02-11 20:26:56	1
244	2021-01-13 16:59:26	2021-02-13 16:59:26	1
265	2021-01-13 16:59:29	2021-02-13 16:59:29	1
294	2021-01-14 23:33:16	2021-02-14 23:33:17	1
362	2021-01-20 19:41:10	2021-02-20 19:41:10	1
553	2021-01-29 18:55:57	2021-02-28 18:55:58	1
644	2021-02-01 16:41:13	2021-03-01 16:41:13	1
871	2021-02-08 17:37:13	2021-03-08 17:37:13	1

From the above table we can see that there are 13 records for which the purchaseDate of the receipt items is after the dateScanned of the receipt which is technically not possible. This inconsistency can create a major issue as this might be because of the following reasons: - The receipt that was scanned was a fake receipt and thus the date entered could be a random date. - The OCR algorithm used in the backend to scan the receipts might not be that accurate which might have created the issue.

As a solution, these receipts should be flagged and the review team should investigate in these scenarios.

Data Inconsistency 2: The following data inconsistency is to check whether the final price in the receipts data is equal to all the price * quantity of different Items for the same receipt_id or not. There can be many scenarios where this inconsistency exist, for the analysis I have cross verified only for 1 receipt_id.

[155]:
```
receipts[receipts['receipt_id'] == '5ff1e1d20a7214ada1000561']
```

[155]:
```
   bonusPointsEarned     bonusPointsEarnedReason          createDate  \
4                5.0  All-receipts receipt bonus 2021-01-03 15:25:06

          dateScanned        finishedDate          modifyDate  \
4 2021-01-03 15:25:06 2021-01-03 15:25:11 2021-01-03 15:25:11

    pointsAwardedDate  pointsEarned        purchaseDate  purchasedItemCount  \
4 2021-01-03 15:25:06           5.0 2021-01-02 15:25:06                 2.0

                              rewardsReceiptItemList rewardsReceiptStatus  \
4  [{'barcode': '4011', 'description': 'ITEM NOT …             FINISHED

   totalSpent                     userId                receipt_id
4         1.0  5ff1e194b6a9d73a3a9f1052  5ff1e1d20a7214ada1000561
```

[202]:
```
temp2 = list(receipts[receipts['receipt_id'] ==␣
↪'5ff1e1d20a7214ada1000561']['rewardsReceiptItemList'].values)[0]
```

[203]:
```
t1 = pd.DataFrame([list(temp2[1].values())], columns = list(temp2[1].keys()))
```

[204]:
```
t2 = pd.DataFrame([list(temp2[0].values())], columns = list(temp2[0].keys()))
```

[205]:
```
items = t1.append(t2)
```

[206]:
```
items['total_price'] = items['finalPrice'].astype(float) *␣
↪items['quantityPurchased'].astype(float)
```

[210]:
```
items
```

[210]:
```
  barcode finalPrice itemPrice needsFetchReview needsFetchReviewReason  \
0    1234       2.56      2.56             True           USER_FLAGGED
0    4011          1         1              NaN                    NaN

  partnerItemId preventTargetGapPoints  quantityPurchased userFlaggedBarcode  \
0             2                   True                  3               1234
0             1                    NaN                  1                NaN

  userFlaggedDescription userFlaggedNewItem userFlaggedPrice  \
0                                     True             2.56
0                    NaN               NaN              NaN

   userFlaggedQuantity               receipt_id receipt_rewardsReceiptStatus  \
0                  3.0  5ff1e1d20a7214ada1000561                    FINISHED
0                  NaN  5ff1e1d20a7214ada1000561                    FINISHED

      description  total_price
0             NaN         7.68
```

```
0  ITEM NOT FOUND         1.00
```

[207]:
```
items['total_price'].sum()
```

[207]:
```
8.68
```

[209]:
```
receipts[receipts['receipt_id'] == '5ff1e1d20a7214ada1000561']
```

[209]:
```
   bonusPointsEarned     bonusPointsEarnedReason          createDate  \
4                5.0  All-receipts receipt bonus 2021-01-03 15:25:06

          dateScanned        finishedDate          modifyDate  \
4 2021-01-03 15:25:06 2021-01-03 15:25:11 2021-01-03 15:25:11

    pointsAwardedDate  pointsEarned        purchaseDate  purchasedItemCount  \
4 2021-01-03 15:25:06           5.0 2021-01-02 15:25:06                 2.0

                            rewardsReceiptItemList rewardsReceiptStatus  \
4  [{'barcode': '4011', 'description': 'ITEM NOT …             FINISHED

   totalSpent                     userId               receipt_id
4         1.0  5ff1e194b6a9d73a3a9f1052  5ff1e1d20a7214ada1000561
```

From the above analysis, we can see that the total amount spent according to the receipts was 8.68 however the totalSpent mentioned in the database is 1.

This is a major inconsistency in the data as the whole rewards program is based on the amount an individual user spends at a time.

As a solution, on backend we should write a query/code to verify if the items listed in the rewardsReceiptItemList column sums up to the totalSpent column in the receipt data, if not these receipts should be flagged and then the number should be updated in the database so that the data is consistent.

[ ]:
```
!jupyter nbconvert --to pdf '/content/drive/MyDrive/Colab Notebooks/Data␣
↪Quality Issues.ipynb'
```