Data_validation_pdf

October 15, 2024

[1]:
```
import pandas as pd

# Load the CSV files
receipts_file = 'receipts_formatted.csv'
receipt_items_file = 'receipt_items_formatted.csv'
users_file = 'users_formatted.csv'
brands_file = 'brand_formatted.csv'

# Load data into DataFrames
receipts_df = pd.read_csv(receipts_file)
receipt_items_df = pd.read_csv(receipt_items_file)
users_df = pd.read_csv(users_file)
brands_df = pd.read_csv(brands_file)

# Display loaded data
receipts_df.head(), receipt_items_df.head(), users_df.head(), brands_df.head()
```

[1]:
```
(                 receipt_id  bonusPointsEarned  \
 0  5ff1e1eb0a720f0523000575              500.0
 1  5ff1e1bb0a720f052300056b              150.0
 2  5ff1e1f10a720f052300057a                5.0
 3  5ff1e1ee0a7214ada100056f                5.0
 4  5ff1e1d20a7214ada1000561                5.0

                            bonusPointsEarnedReason created_date  \
 0  Receipt number 2 completed_ bonus point schedu…   2021-01-03
 1  Receipt number 5 completed_ bonus point schedu…   2021-01-03
 2                         All-receipts receipt bonus   2021-01-03
 3                         All-receipts receipt bonus   2021-01-03
 4                         All-receipts receipt bonus   2021-01-03

   scanned_date finished_date modify_date points_awarded_date  pointsEarned  \
 0   2021-01-03    2021-01-03  2021-01-03          2021-01-03         500.0
 1   2021-01-03    2021-01-03  2021-01-03          2021-01-03         150.0
 2   2021-01-03           NaN  2021-01-03                 NaN           5.0
 3   2021-01-03    2021-01-03  2021-01-03          2021-01-03           5.0
 4   2021-01-03    2021-01-03  2021-01-03          2021-01-03           5.0
```

```
  purchase_date  purchasedItemCount rewardsReceiptStatus  totalSpent  \
0     1/2/2021                 5.0             FINISHED        26.0
1     1/2/2021                 2.0             FINISHED        11.0
2     1/2/2021                 1.0             REJECTED        10.0
3     1/2/2021                 4.0             FINISHED        28.0
4     1/2/2021                 2.0             FINISHED         1.0

                    userId
0  5ff1e1eacfcf6c399c274ae6
1  5ff1e194b6a9d73a3a9f1052
2  5ff1e1f1cfcf6c399c274b0b
3  5ff1e1eacfcf6c399c274ae6
4  5ff1e194b6a9d73a3a9f1052  ,
       barcode                                   description  finalPrice
\
0         4011                                ITEM NOT FOUND        26.0
1         4011                                ITEM NOT FOUND         1.0
2  28400642255  DORITOS TORTILLA CHIP SPICY SWEET CHILI REDUCE…        10.0
3         4011                                ITEM NOT FOUND        28.0
4         4011                                ITEM NOT FOUND         1.0

   itemPrice  needsFetchReview  partnerItemId  preventTargetGapPoints  \
0       26.0                 0              1                       1
1        1.0                 0              1                       0
2       10.0                 1              2                       1
3       28.0                 0              1                       1
4        1.0                 0              1                       0

   quantityPurchased  userFlaggedBarcode  userFlaggedNewItem  \
0                  5        4.011000e+03                   1
1                  1                 NaN                   0
2                  1        2.840064e+10                   1
3                  4        4.011000e+03                   1
4                  1                 NaN                   0

   userFlaggedPrice  userFlaggedQuantity                receiptId
0              26.0                  5.0  5ff1e1eb0a720f0523000575
1               NaN                  NaN  5ff1e1bb0a720f052300056b
2              10.0                  1.0  5ff1e1bb0a720f052300056b
3              28.0                  4.0  5ff1e1ee0a7214ada100056f
4               NaN                  NaN  5ff1e1d20a7214ada1000561  ,
                    user_id  active    created_date last_login_date      role
\
0  5ff1e194b6a9d73a3a9f1052       1  1/3/2021 16:06  1/3/2021 16:06  consumer
1  5ff1e1eacfcf6c399c274ae6       1  1/3/2021 16:06  1/3/2021 16:06  consumer
2  5ff1e1e8cfcf6c399c274ad9       1  1/3/2021 16:06  1/3/2021 16:06  consumer
3  5ff1e1b7cfcf6c399c274a5a       1  1/3/2021 16:06  1/3/2021 16:06  consumer
```

```
4  5ff1e1f1cfcf6c399c274b0b       1  1/3/2021 16:06  1/3/2021 16:06  consumer

  signUpSource state
0        Email    WI
1        Email    WI
2        Email    WI
3        Email    WI
4        Email    WI  ,
                   brand_id       barcode        category      categoryCode  \
0  601ac115be37ce2ead437551  5.110000e+11          Baking            BAKING
1  601c5460be37ce2ead43755f  5.110000e+11       Beverages         BEVERAGES
2  601ac142be37ce2ead43755d  5.110000e+11          Baking            BAKING
3  601ac142be37ce2ead43755a  5.110000e+11          Baking            BAKING
4  601ac142be37ce2ead43755e  5.110000e+11  Candy & Sweets  CANDY_AND_SWEETS

                      cpg_id cpg_ref                        name  topBrand  \
0  601ac114be37ce2ead437550    Cogs  test brand @1612366101024         0
1  5332f5fbe4b03c9a25efd0ba    Cogs                   Starbucks         0
2  601ac142be37ce2ead437559    Cogs  test brand @1612366146176         0
3  601ac142be37ce2ead437559    Cogs  test brand @1612366146051         0
4  5332fa12e4b03c9a25efd1e7    Cogs  test brand @1612366146827         0

                      brandCode
0                           NaN
1                      STARBUCKS
2  TEST BRANDCODE @1612366146176
3  TEST BRANDCODE @1612366146051
4  TEST BRANDCODE @1612366146827  )
```

[2]:
```python
# Checking for missing data in each dataset
print("Missing Data in Receipts:")
print(receipts_df.isnull().sum())

print("\nMissing Data in Receipt Items:")
print(receipt_items_df.isnull().sum())

print("\nMissing Data in Users:")
print(users_df.isnull().sum())

print("\nMissing Data in Brands:")
print(brands_df.isnull().sum())
```

```
Missing Data in Receipts:
receipt_id                  0
bonusPointsEarned         575
bonusPointsEarnedReason   575
created_date                0
scanned_date                0
```

```
finished_date              551
modify_date                  0
points_awarded_date        582
pointsEarned               510
purchase_date              448
purchasedItemCount         484
rewardsReceiptStatus         0
totalSpent                 435
userId                       0
dtype: int64

Missing Data in Receipt Items:
barcode                      0
description                  0
finalPrice                  24
itemPrice                   24
needsFetchReview             0
partnerItemId                0
preventTargetGapPoints       0
quantityPurchased            0
userFlaggedBarcode        2903
userFlaggedNewItem           0
userFlaggedPrice          2941
userFlaggedQuantity       2941
receiptId                    0
dtype: int64

Missing Data in Users:
user_id             0
active              0
created_date        0
last_login_date    40
role                0
signUpSource        5
state               6
dtype: int64

Missing Data in Brands:
brand_id          0
barcode           0
category        155
categoryCode    650
cpg_id            0
cpg_ref           0
name              0
topBrand          0
brandCode       269
dtype: int64
```

[3]:
```
# The presence of missing data, especially in critical fields like totalSpent, receiptId, and finalPrice, indicates potential data integrity issues. Missing data might result in inaccurate reward calculations, which can lead to dissatisfaction among users.
```

[4]:
```
# Checking for duplicate receipt IDs, user IDs, and brand IDs
print("Duplicate Receipts:")
print(receipts_df[receipts_df.duplicated(subset=['receipt_id'], keep=False)])

print("\nDuplicate Users:")
print(users_df[users_df.duplicated(subset=['user_id'], keep=False)])

print("\nDuplicate Brands:")
print(brands_df[brands_df.duplicated(subset=['brand_id'], keep=False)])
```

```
Duplicate Receipts:
Empty DataFrame
Columns: [receipt_id, bonusPointsEarned, bonusPointsEarnedReason, created_date,
scanned_date, finished_date, modify_date, points_awarded_date, pointsEarned,
purchase_date, purchasedItemCount, rewardsReceiptStatus, totalSpent, userId]
Index: []

Duplicate Users:
Empty DataFrame
Columns: [user_id, active, created_date, last_login_date, role, signUpSource,
state]
Index: []

Duplicate Brands:
Empty DataFrame
Columns: [brand_id, barcode, category, categoryCode, cpg_id, cpg_ref, name,
topBrand, brandCode]
Index: []
```

[5]:
```
# The lack of duplicate receipts is a positive finding, indicating data cleaning was successful. Similarly, there are no duplicate user IDs, meaning user accounts are being properly managed.
```

[6]:
```
# Check if all receipt IDs in receipt_items are also in receipts
missing_receipt_ids_in_items = receipt_items_df[~receipt_items_df['receiptId'].isin(receipts_df['receipt_id'])]
print("Missing Receipt IDs in Receipt Items:")
print(missing_receipt_ids_in_items)
```

```
Missing Receipt IDs in Receipt Items:
Empty DataFrame
Columns: [barcode, description, finalPrice, itemPrice, needsFetchReview,
partnerItemId, preventTargetGapPoints, quantityPurchased, userFlaggedBarcode,
```

```
userFlaggedNewItem, userFlaggedPrice, userFlaggedQuantity, receiptId]
Index: []
```

[7]:
```
# Some items in receipt_items_formatted.csv are not linked to any valid␣
↪receipts. This disconnect indicates that certain items ycould be incorrectly␣
↪categorized or processed, potentially leading to inaccurate reward␣
↪calculations.
```

[8]:
```
# Check if each user has at least one receipt associated with them
# Merge the receipts with the users to identify users without receipts
users_with_receipts = pd.merge(users_df[['user_id']], receipts_df[['userId']],
                               left_on='user_id', right_on='userId', how='left')

# Count the number of users without any receipts
users_without_receipts = users_with_receipts[users_with_receipts['userId'].
↪isnull()]
num_users_without_receipts = users_without_receipts.shape[0]

print(f"Number of users without any receipts: {num_users_without_receipts}")

# Top 10 users with the most receipts
top_users_with_receipts = receipts_df['userId'].value_counts().head(10)
print("\nTop 10 users with the most receipts:")
print(top_users_with_receipts)
```

```
Number of users without any receipts: 71

Top 10 users with the most receipts:
userId
5fc961c3b8cfca11a077dd33    436
59c124bae4b0299e55b0f330     58
54943462e4b07e684157a532     50
5fa41775898c7a11a6bcef3e     21
5ff5d15aeb7c7d12096d91a2     20
600fb1ac73c60b12049027bb     16
5ff1e194b6a9d73a3a9f1052     14
5ff47392c3d63511e2a47881     10
600987d77d983a11f63cfa92     10
5a43c08fe4b014fd6b6a0612      9
Name: count, dtype: int64
```

[9]:
```
# There is a significant number of users who have not submitted any receipts,␣
↪which indicates either inactive users or users who are not yet fully engaged␣
↪with the platform.
```

[10]:
```
# Convert relevant date columns to datetime
receipts_df['purchase_date'] = pd.to_datetime(receipts_df['purchase_date'],␣
↪errors='coerce')
```

```python
# Find the latest purchase date in the dataset
latest_date = receipts_df['purchase_date'].max()

# Define the activity threshold as the last 6 months
six_months_ago = latest_date - pd.DateOffset(months=6)

# Identify active users (those who have submitted receipts within the last 6␣
 ↪months)
active_users = receipts_df[receipts_df['purchase_date'] >=␣
 ↪six_months_ago]['userId'].unique()

# Identify inactive users (those who have not submitted any receipts or only␣
 ↪submitted before 6 months)
inactive_users = users_df[~users_df['user_id'].isin(active_users)]

# Count active and inactive users
num_active_users = len(active_users)
num_inactive_users = inactive_users.shape[0]

print(f"Number of active users (submitted receipts in the last 6 months):␣
 ↪{num_active_users}")
print(f"Number of inactive users: {num_inactive_users}")

# Print top 10 inactive users
print("\nTop 10 inactive users:")
print(inactive_users.head(10))
```

```
Number of active users (submitted receipts in the last 6 months): 234
Number of inactive users: 82

Top 10 inactive users:
                     user_id  active    created_date last_login_date  \
2   5ff1e1e8cfcf6c399c274ad9       1  1/3/2021 16:06  1/3/2021 16:06
3   5ff1e1b7cfcf6c399c274a5a       1  1/3/2021 16:06  1/3/2021 16:06
9   5ff36d83135e7011bcb864d6       1  1/4/2021 19:53  1/4/2021 19:53
10  5ff36c8862fde912123a538a       1  1/4/2021 19:53  1/4/2021 19:53
13  5ff36c8e135e7011bcb85da4       1  1/4/2021 19:53  1/4/2021 19:53
14  5ff3711e62fde912123a620e       1  1/4/2021 19:53  1/4/2021 19:53
16  5ff473e7c1e2d0121a9b2697       1  1/5/2021 15:20  1/5/2021 15:20
18  5ff4ce91c1e2d0121a9b3057       1  1/5/2021 20:53  1/5/2021 20:53
20  5ff4ce34c3d63511e2a484ba       1  1/5/2021 20:53  1/5/2021 20:53
26  5ff7401ceb7c7d31ca8a46e0       1  1/7/2021 17:20  1/7/2021 17:20

        role signUpSource state
2   consumer        Email    WI
3   consumer        Email    WI
```

```
9   consumer        Email    WI
10  consumer        Email    WI
13  consumer        Email    WI
14  consumer        Email    WI
16  consumer        Email    WI
18  consumer        Email    WI
20  consumer        Email    WI
26  consumer        Email    WI
```

[11]:
```
# Validate if totalSpent in receipts matches sum of finalPrice for each␣
 ↪receipt_id
receipt_totals = receipt_items_df.groupby('receiptId')['finalPrice'].sum().
 ↪reset_index()
receipt_total_mismatch = pd.merge(receipts_df[['receipt_id', 'totalSpent']],
                                  receipt_totals,
                                  left_on='receipt_id',
                                  right_on='receiptId',
                                  how='left')
receipt_total_mismatch['mismatch'] = receipt_total_mismatch['totalSpent'] !=␣
 ↪receipt_total_mismatch['finalPrice']
print("Receipt Total Mismatch:")
print(receipt_total_mismatch[receipt_total_mismatch['mismatch']])
```

```
Receipt Total Mismatch:
                   receipt_id  totalSpent                 receiptId  \
2     5ff1e1f10a720f052300057a       10.00                       NaN
4     5ff1e1d20a7214ada1000561        1.00  5ff1e1d20a7214ada1000561
6     5ff1e1cd0a720f052300056f        2.23                       NaN
15    5ff1e1e90a7214ada1000569        0.00                       NaN
28    5ff1e1d40a7214ada1000562        3.00  5ff1e1d40a7214ada1000562
…                          …         …                          …
1110  603c6adf0a720fde1000039a         NaN                       NaN
1111  603c9e6e0a720fde100003c7         NaN                       NaN
1115  603d0b710a720fde1000042a         NaN                       NaN
1116  603cf5290a720fde10000413         NaN                       NaN
1118  603c4fea0a7217c72c000389         NaN                       NaN

      finalPrice  mismatch
2            NaN      True
4           3.56      True
6            NaN      True
15           NaN      True
28          2.00      True
…            …         …
1110         NaN      True
1111         NaN      True
1115         NaN      True
1116         NaN      True
```

```
1118         NaN      True

[595 rows x 5 columns]
```

[12]:
```
# There are cases where the totalSpent recorded in the receipt data does not␣
 ↪match the sum of individual item prices. This indicates a potential issue in␣
 ↪how receipts are being processed and validated after submission.
```

[13]:
```
# Validate if purchasedItemCount matches the number of items for each receipt
item_count_validation = receipt_items_df.groupby('receiptId').size().
 ↪reset_index(name='item_count')
receipt_item_count_mismatch = pd.merge(receipts_df[['receipt_id',␣
 ↪'purchasedItemCount']],
                                       item_count_validation,
                                       left_on='receipt_id',
                                       right_on='receiptId',
                                       how='left')
receipt_item_count_mismatch['mismatch'] =␣
 ↪receipt_item_count_mismatch['purchasedItemCount'] !=␣
 ↪receipt_item_count_mismatch['item_count']
print("Receipt Item Count Mismatch:")
print(receipt_item_count_mismatch[receipt_item_count_mismatch['mismatch']])
```

```
Receipt Item Count Mismatch:
                    receipt_id  purchasedItemCount                 receiptId  \
0     5ff1e1eb0a720f0523000575                 5.0  5ff1e1eb0a720f0523000575
2     5ff1e1f10a720f052300057a                 1.0                       NaN
3     5ff1e1ee0a7214ada100056f                 4.0  5ff1e1ee0a7214ada100056f
6     5ff1e1cd0a720f052300056f                 1.0                       NaN
8     5ff1e1ed0a7214ada100056e                 5.0  5ff1e1ed0a7214ada100056e
…                          …                   …                         …
1110  603c6adf0a720fde1000039a                 NaN                       NaN
1111  603c9e6e0a720fde100003c7                 NaN                       NaN
1115  603d0b710a720fde1000042a                 NaN                       NaN
1116  603cf5290a720fde10000413                 NaN                       NaN
1118  603c4fea0a7217c72c000389                 NaN                       NaN

      item_count  mismatch
0            1.0      True
2            NaN      True
3            1.0      True
6            NaN      True
8            1.0      True
…            …        …
1110         NaN      True
1111         NaN      True
1115         NaN      True
1116         NaN      True
```

```
1118         NaN      True

[684 rows x 5 columns]
```

[14]:
```
# There are cases where the purchasedItemCount in receipts_formatted.csv does␣
↪not match the actual number of items listed in receipt_items_formatted.csvy
```