Mishra540_Project_Milestone_02

July 2, 2023

[1]:
```
# DSC540,Summer 2023 - T302 Data Preparation(2237-1)
# Assignment: Project Milestone 02
# Author by:  Debabrata Mishra
# Date: 2023-07-02

# Topic - Credit Card Transactional & Demographic Data
```

1 Milestone 2 Tasks

Perform at least 5 data transformation and/or cleansing steps to your flat file data. The below examples are not required - they are just potential transformations you could do. If your data doesn't work for these scenarios, complete different transformations. You can do the same transformation multiple times if needed to clean your data. The goal is a clean dataset at the end of the milestone.

```
Replace Headers
Format data into a more readable format
Identify outliers and bad data
Find duplicates
Fix casing or inconsistent values
Conduct Fuzzy Matching
```

Make sure you clearly label each transformation (Step #1, Step #2, etc.) in your code and describe what it is doing in 1-2 sentences. You can submit a Jupyter Notebook or a PDF of your code. If you submit a .py file you need to also include a PDF or attachment of your results.

2 Cleaning and Formatting of FraudTrain.csv (Flat File Source)

2.1 Flat File

Descrition:

Simulated credit card transaction dataset containing legitimate and fraud transactions from the duration 1st Jan 2019 - 31st Dec 2020. It covers credit cards of 1000+ customers doing transactions across 693 unique merchant ids. It has 23 columns and transactional information along with demographic details of Merchants and Card Holders.

Link: https://www.kaggle.com/code/nathanxiang/credit-card-fraud-analysis-and-modeling/input?select=fraudTrain.csv

I have downloaded that file to local folder and then executed below steps for cleaning & formatting actvities.

Ethical implications : should always be considered when working with sensitive data like credit card fraud data. Proper handling and protection of the data are crucial. Access to the data should be restricted to authorized personnel only, and measures should be in place to prevent unauthorized access or theft. The transformed data should be used solely for legitimate purposes such as fraud detection, prevention, or investigation, and should not be used for any illegal or unethical activities. It is also essential to securely delete the data once it is no longer needed to prevent any further exposure of sensitive information.

I will implement boolean indexing methid to identify the values in the 'amt' column that fall outside the specified range. These outliers should be carefully inspected to ensure their accuracy and determine whether they should be considered valid data points or corrected. It would be done when I will connect this with web/HTML data for range reference.Additionally, metrchant address will be derived when I will connect this data with API

As part of Project Milestone 2: I have considered below transformations.

- Createing and rename header fields
- Createing new variables which required for future calculations like customer age , day, date, month, hour, weekdayfrom transaction date/time field
- Creat Masked the account number and BIN from account number and then drop account number colmun
- Format amount values to 2 decimal points
- Drop columns that I won't be use for any of my planned analysis like dob (custome age will be used), city_pop, customer latitude and lognitude
- Identify outliers using IQR
- Identify outliers using Z-score and drop those records (if any)
- check for duplicate and drop those duplicates (if any)
- Data accuracy check for amount field
- Check for BIN occurance as those help in understanding BOT attacks

[2]:
```
#Load the Necessary Libraries
import requests as r
import pandas as pd
import xlrd
from bs4 import BeautifulSoup
import numpy as np
import matplotlib.pyplot as plt
```

[3]:
```
txn_data = pd.read_csv('fraudTrain.csv', sep=",")
txn_data.head()
```

[3]:
```
   Unnamed: 0 trans_date_trans_time            cc_num  \
0           0   2019-01-01 00:00:18  2703186189652095
1           1   2019-01-01 00:00:44      630423337322
2           2   2019-01-01 00:00:51    38859492057661
3           3   2019-01-01 00:01:16  3534093764340240
4           4   2019-01-01 00:03:06   375534208663984

                             merchant       category     amt      first  \
0          fraud_Rippin, Kub and Mann       misc_net    4.97   Jennifer
1     fraud_Heller, Gutmann and Zieme    grocery_pos  107.23  Stephanie
2                fraud_Lind-Buckridge  entertainment  220.11     Edward
3  fraud_Kutch, Hermiston and Farrell  gas_transport   45.00     Jeremy
4                 fraud_Keeling-Crist       misc_pos   41.96      Tyler

      last gender                        street  …      lat      long  \
0    Banks      F                561 Perry Cove  …  36.0788  -81.1781
1     Gill      F  43039 Riley Greens Suite 393  …  48.8878 -118.2105
2  Sanchez      M      594 White Dale Suite 530  …  42.1808 -112.2620
3    White      M   9443 Cynthia Court Apt. 038  …  46.2306 -112.1138
4   Garcia      M              408 Bradley Rest  …  38.4207  -79.4629

   city_pop                                job         dob  \
0      3495          Psychologist, counselling  1988-03-09
1       149  Special educational needs teacher  1978-06-21
2      4154        Nature conservation officer  1962-01-19
3      1939                    Patent attorney  1967-01-12
4        99     Dance movement psychotherapist  1986-03-28

                          trans_num   unix_time  merch_lat  merch_long  \
0  0b242abb623afc578575680df30655b9  1325376018  36.011293  -82.048315
1  1f76529f8574734946361c461b024d99  1325376044  49.159047 -118.186462
2  a1a22d70485983eac12b5b88dad1cf95  1325376051  43.150704 -112.154481
3  6b849c168bdad6f867558c3793159a81  1325376076  47.034331 -112.561071
4  a41d7549acf90789359a9aa5346dcb46  1325376186  38.674999  -78.632459

   is_fraud
0         0
1         0
2         0
3         0
4         0

[5 rows x 23 columns]
```

[4]:
```
# Overview of the structure and characteristics
txn_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 1296675 entries, 0 to 1296674
Data columns (total 23 columns):
 #   Column                 Non-Null Count    Dtype
---  ------                 --------------    -----
 0   Unnamed: 0             1296675 non-null  int64
 1   trans_date_trans_time  1296675 non-null  object
 2   cc_num                 1296675 non-null  int64
 3   merchant               1296675 non-null  object
 4   category               1296675 non-null  object
 5   amt                    1296675 non-null  float64
 6   first                  1296675 non-null  object
 7   last                   1296675 non-null  object
 8   gender                 1296675 non-null  object
 9   street                 1296675 non-null  object
 10  city                   1296675 non-null  object
 11  state                  1296675 non-null  object
 12  zip                    1296675 non-null  int64
 13  lat                    1296675 non-null  float64
 14  long                   1296675 non-null  float64
 15  city_pop               1296675 non-null  int64
 16  job                    1296675 non-null  object
 17  dob                    1296675 non-null  object
 18  trans_num              1296675 non-null  object
 19  unix_time              1296675 non-null  int64
 20  merch_lat              1296675 non-null  float64
 21  merch_long             1296675 non-null  float64
 22  is_fraud               1296675 non-null  int64
dtypes: float64(5), int64(6), object(12)
memory usage: 227.5+ MB
```

[5]:
```
# 01- Rename the headers with meaningful names.
txn_data.rename(columns = {'Unnamed: 0':'row_id', 'cc_num':'accountNumber',␣
 ↪'amt':'amount','merchant':'merch_name'}, inplace = True)
txn_data.head()
```

[5]:

	row_id	trans_date_trans_time	accountNumber	merch_name	category	amount	first
0	0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer
1	1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie
2	2	2019-01-01 00:00:51	38859492057661	fraud_Lind-Buckridge	entertainment	220.11	Edward
3	3	2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	Jeremy
4	4	2019-01-01 00:03:06	375534208663984				

```
4                fraud_Keeling-Crist       misc_pos   41.96      Tyler

      last gender                        street  …      lat      long  \
0    Banks      F                561 Perry Cove  …  36.0788  -81.1781
1     Gill      F  43039 Riley Greens Suite 393  …  48.8878 -118.2105
2  Sanchez      M      594 White Dale Suite 530  …  42.1808 -112.2620
3    White      M   9443 Cynthia Court Apt. 038  …  46.2306 -112.1138
4   Garcia      M              408 Bradley Rest  …  38.4207  -79.4629

   city_pop                                job         dob  \
0      3495          Psychologist, counselling  1988-03-09
1       149  Special educational needs teacher  1978-06-21
2      4154        Nature conservation officer  1962-01-19
3      1939                     Patent attorney  1967-01-12
4        99     Dance movement psychotherapist  1986-03-28

                          trans_num   unix_time  merch_lat  merch_long  \
0  0b242abb623afc578575680df30655b9  1325376018  36.011293  -82.048315
1  1f76529f8574734946361c461b024d99  1325376044  49.159047 -118.186462
2  a1a22d70485983eac12b5b88dad1cf95  1325376051  43.150704 -112.154481
3  6b849c168bdad6f867558c3793159a81  1325376076  47.034331 -112.561071
4  a41d7549acf90789359a9aa5346dcb46  1325376186  38.674999  -78.632459

   is_fraud
0         0
1         0
2         0
3         0
4         0

[5 rows x 23 columns]
```

[6]:
```
# 02- Create some new colmuns linked with timestamp and customer date of birth
  ↪which could be used for calculation(s)
#     format the timestamp fields

# Create variables from trasnaction date/time for future calculations
txn_data['trans_date_trans_time'] = pd.
  ↪to_datetime(txn_data['trans_date_trans_time'],errors='coerce')

txn_data['year']=txn_data['trans_date_trans_time'].dt.year
txn_data['month']=txn_data['trans_date_trans_time'].dt.strftime('%b')
txn_data['month']=txn_data['trans_date_trans_time'].dt.month
txn_data['day']=txn_data['trans_date_trans_time'].dt.day
txn_data['hour']=txn_data['trans_date_trans_time'].dt.hour
txn_data['weekday']=txn_data['trans_date_trans_time'].dt.strftime('%a')
txn_data['dayofYear']=txn_data['trans_date_trans_time'].dt.dayofyear
```

```
# Create a variable age of customer on day of transaction

txn_data['txn_date']=txn_data['trans_date_trans_time'].dt.strftime('%Y-%m-%d')
txn_data['txn_date']=pd.to_datetime(txn_data['txn_date'])
txn_data['dob']=pd.to_datetime(txn_data['dob'])
txn_data["customer_age"] =txn_data["txn_date"]-txn_data["dob"]
txn_data["customer_age"] =txn_data["customer_age"].astype('timedelta64[Y]')

# Transform the unix time into timestamp format
txn_data['unix_time'] = pd.to_datetime(txn_data['unix_time'], unit='s')

txn_data.head()
```

[6]:

```
   row_id trans_date_trans_time     accountNumber  \
0       0   2019-01-01 00:00:18  2703186189652095
1       1   2019-01-01 00:00:44      630423337322
2       2   2019-01-01 00:00:51    38859492057661
3       3   2019-01-01 00:01:16  3534093764340240
4       4   2019-01-01 00:03:06   375534208663984

                          merch_name       category  amount      first  \
0          fraud_Rippin, Kub and Mann       misc_net    4.97   Jennifer
1     fraud_Heller, Gutmann and Zieme    grocery_pos  107.23  Stephanie
2                fraud_Lind-Buckridge  entertainment  220.11     Edward
3  fraud_Kutch, Hermiston and Farrell  gas_transport   45.00     Jeremy
4                 fraud_Keeling-Crist       misc_pos   41.96      Tyler

      last gender                        street  …  merch_long is_fraud  \
0    Banks      F                561 Perry Cove  …  -82.048315        0
1     Gill      F  43039 Riley Greens Suite 393  … -118.186462        0
2  Sanchez      M      594 White Dale Suite 530  … -112.154481        0
3    White      M   9443 Cynthia Court Apt. 038  … -112.561071        0
4   Garcia      M              408 Bradley Rest  …  -78.632459        0

   year  month  day  hour weekday dayofYear   txn_date customer_age
0  2019      1    1     0     Tue         1 2019-01-01         30.0
1  2019      1    1     0     Tue         1 2019-01-01         40.0
2  2019      1    1     0     Tue         1 2019-01-01         56.0
3  2019      1    1     0     Tue         1 2019-01-01         51.0
4  2019      1    1     0     Tue         1 2019-01-01         32.0

[5 rows x 31 columns]
```

[7]:

```
# 03 - Mask the account number , Create the BIN (Bank Identification Number)␣
 ↪and then drop the clear account number
```

```
# Define a function to mask the account number
def mask_account_number(accountNumber):
    return accountNumber[:4] + '*' * (len(accountNumber) - 8) +␣
 ↪accountNumber[-4:]

# Apply the masking function to the 'accountNumber' column

# Convert AccountNumber column to string
txn_data['accountNumber'] = txn_data['accountNumber'].astype(str)
txn_data['masked_accountNumber'] = txn_data['accountNumber'].
 ↪apply(mask_account_number)

# Extract the BIN from the masked account number
txn_data['BIN'] = txn_data['accountNumber'].str[:6]

# Drop the 'accountNumber' column
txn_data = txn_data.drop('accountNumber', axis=1)

txn_data.head()
```

[7]:

```
   row_id trans_date_trans_time                          merch_name  \
0       0   2019-01-01 00:00:18          fraud_Rippin, Kub and Mann
1       1   2019-01-01 00:00:44     fraud_Heller, Gutmann and Zieme
2       2   2019-01-01 00:00:51                fraud_Lind-Buckridge
3       3   2019-01-01 00:01:16  fraud_Kutch, Hermiston and Farrell
4       4   2019-01-01 00:03:06                 fraud_Keeling-Crist

        category  amount      first     last gender  \
0       misc_net    4.97   Jennifer    Banks      F
1   grocery_pos  107.23  Stephanie     Gill      F
2  entertainment  220.11     Edward  Sanchez      M
3  gas_transport   45.00     Jeremy    White      M
4       misc_pos   41.96      Tyler   Garcia      M

                        street            city  …  year  month  day  hour  \
0                561 Perry Cove  Moravian Falls  …  2019      1    1     0
1  43039 Riley Greens Suite 393          Orient  …  2019      1    1     0
2      594 White Dale Suite 530     Malad City  …  2019      1    1     0
3   9443 Cynthia Court Apt. 038         Boulder  …  2019      1    1     0
4              408 Bradley Rest        Doe Hill  …  2019      1    1     0

   weekday dayofYear   txn_date customer_age masked_accountNumber     BIN
0      Tue         1 2019-01-01         30.0     2703********2095  270318
1      Tue         1 2019-01-01         40.0         6304****7322  630423
2      Tue         1 2019-01-01         56.0       3885******7661  388594
3      Tue         1 2019-01-01         51.0     3534********0240  353409
4      Tue         1 2019-01-01         32.0      3755*******3984  375534
```

```
[5 rows x 32 columns]
```

[8]:
```
# 04 - Format the amount to float with 2 decimal points
txn_data['amount'] = txn_data['amount'].round(2)
txn_data.head()
```

[8]:
```
   row_id trans_date_trans_time                         merch_name  \
0       0   2019-01-01 00:00:18          fraud_Rippin, Kub and Mann
1       1   2019-01-01 00:00:44     fraud_Heller, Gutmann and Zieme
2       2   2019-01-01 00:00:51                fraud_Lind-Buckridge
3       3   2019-01-01 00:01:16  fraud_Kutch, Hermiston and Farrell
4       4   2019-01-01 00:03:06                 fraud_Keeling-Crist

        category  amount      first     last gender  \
0       misc_net    4.97   Jennifer    Banks      F
1    grocery_pos  107.23  Stephanie     Gill      F
2  entertainment  220.11     Edward  Sanchez      M
3  gas_transport   45.00     Jeremy    White      M
4       misc_pos   41.96      Tyler   Garcia      M

                        street            city  …  year  month  day  hour  \
0                561 Perry Cove  Moravian Falls  …  2019      1    1     0
1  43039 Riley Greens Suite 393          Orient  …  2019      1    1     0
2      594 White Dale Suite 530     Malad City  …  2019      1    1     0
3   9443 Cynthia Court Apt. 038         Boulder  …  2019      1    1     0
4              408 Bradley Rest        Doe Hill  …  2019      1    1     0

   weekday dayofYear   txn_date customer_age masked_accountNumber     BIN
0      Tue         1 2019-01-01         30.0     2703********2095  270318
1      Tue         1 2019-01-01         40.0         6304****7322  630423
2      Tue         1 2019-01-01         56.0       3885******7661  388594
3      Tue         1 2019-01-01         51.0     3534********0240  353409
4      Tue         1 2019-01-01         32.0      3755*******3984  375534

[5 rows x 32 columns]
```

[9]:
```
# 05- Check for the missing values for all columns

for col in txn_data.columns:
    count = txn_data[col].isnull().sum()
    if count>0:
        print("{} has {} missing value(s).".format(col,miss))
    else:
        print("{} has no missing values.".format(col))
```

```
row_id has no missing values.
trans_date_trans_time has no missing values.
```

```
merch_name has no missing values.
category has no missing values.
amount has no missing values.
first has no missing values.
last has no missing values.
gender has no missing values.
street has no missing values.
city has no missing values.
state has no missing values.
zip has no missing values.
lat has no missing values.
long has no missing values.
city_pop has no missing values.
job has no missing values.
dob has no missing values.
trans_num has no missing values.
unix_time has no missing values.
merch_lat has no missing values.
merch_long has no missing values.
is_fraud has no missing values.
year has no missing values.
month has no missing values.
day has no missing values.
hour has no missing values.
weekday has no missing values.
dayofYear has no missing values.
txn_date has no missing values.
customer_age has no missing values.
masked_accountNumber has no missing values.
BIN has no missing values.
```

[10]:
```
# 06 - Drop/ Delete the 'city_pop' and 'dob' column.
columns_to_drop = ['dob', 'city_pop', 'lat','long']
txn_data = txn_data.drop(columns_to_drop, axis=1)
txn_data.head()
```

[10]:

	row_id	trans_date_trans_time	merch_name	category	amount	first	last	gender
0	0	2019-01-01 00:00:18	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Banks	F
1	1	2019-01-01 00:00:44	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	Gill	F
2	2	2019-01-01 00:00:51	fraud_Lind-Buckridge	entertainment	220.11	Edward	Sanchez	M

```
3  gas_transport   45.00     Jeremy    White      M
4       misc_pos   41.96      Tyler   Garcia      M

                         street            city  …  year  month day hour  \
0                561 Perry Cove  Moravian Falls  …  2019      1   1    0
1  43039 Riley Greens Suite 393          Orient  …  2019      1   1    0
2      594 White Dale Suite 530     Malad City  …  2019      1   1    0
3   9443 Cynthia Court Apt. 038         Boulder  …  2019      1   1    0
4              408 Bradley Rest        Doe Hill  …  2019      1   1    0

  weekday  dayofYear   txn_date  customer_age  masked_accountNumber     BIN
0     Tue          1 2019-01-01          30.0      2703********2095  270318
1     Tue          1 2019-01-01          40.0          6304****7322  630423
2     Tue          1 2019-01-01          56.0        3885******7661  388594
3     Tue          1 2019-01-01          51.0      3534********0240  353409
4     Tue          1 2019-01-01          32.0       3755*******3984  375534

[5 rows x 28 columns]
```

[11]:
```
# 07- Identify outliers and bad data
Q1 = txn_data['amount'].quantile(0.25)
Q3 = txn_data['amount'].quantile(0.75)
IQR = Q3 - Q1
outliers = txn_data[(txn_data['amount'] < (Q1 - 1.5 * IQR))␣
 ↪|(txn_data['amount'] > (Q3 + 1.5 * IQR))]
print("Outliers:\n", outliers)
```

```
Outliers:
          row_id trans_date_trans_time  \
2              2   2019-01-01 00:00:51
9              9   2019-01-01 00:06:01
16            16   2019-01-01 00:10:49
17            17   2019-01-01 00:10:58
36            36   2019-01-01 00:26:22
…            …                   …
1296547  1296547   2020-06-21 11:07:51
1296557  1296557   2020-06-21 11:14:04
1296590  1296590   2020-06-21 11:32:55
1296603  1296603   2020-06-21 11:36:41
1296657  1296657   2020-06-21 12:07:20

                                    merch_name       category  amount  \
2                        fraud_Lind-Buckridge  entertainment  220.11
9             fraud_Schoen, Kuphal and Nitzsche    grocery_pos  198.39
16                       fraud_Lebsack and Sons       misc_net  327.00
17                           fraud_Mayert Group   shopping_pos  341.67
36                        fraud_Heidenreich PLC    grocery_pos  207.36
…                                           …            …     …
```

```
1296547                       fraud_Bednar Group       misc_net  754.81
1296557                           fraud_Kuhn LLC       misc_net  374.71
1296590        fraud_Osinski, Ledner and Leuschke    grocery_pos  194.07
1296603                         fraud_Klein Group  entertainment  255.52
1296657  fraud_Greenfelder, Bartoletti and Davis       misc_net  264.22

           first       last gender                          street  \
2         Edward    Sanchez      M         594 White Dale Suite 530
9        Melissa    Aguilar      F   21326 Taylor Squares Suite 708
16          Lisa     Mendez      F     44259 Beth Station Suite 215
17        Nathan     Thomas      M    4923 Campbell Pines Suite 717
36        Ashley      Lopez      F             9333 Valentine Point
…           …         …    …                                  …
1296547   Robert      Evans      M    01892 Patricia Vista Apt. 828
1296557  Jessica     Garcia      F           13108 Jennifer Passage
1296590    Tammy      Davis      F            77663 Colleen Freeway
1296603    Aaron       Pena      M      793 Hooper Tunnel Suite 154
1296657  Breanna  Rodriguez      F     118 Cabrera Springs Apt. 105

                   city  …  year  month day hour weekday  dayofYear  \
2             Malad City  …  2019      1   1    0     Tue          1
9            Clarksville  …  2019      1   1    0     Tue          1
16               Lahoma  …  2019      1   1    0     Tue          1
17             Carlisle  …  2019      1   1    0     Tue          1
36             Bellmore  …  2019      1   1    0     Tue          1
…                 …  …   …    …  ..  …     …        …
1296547          Sachse  …  2020      6  21   11     Sun        173
1296557       Mc Cracken  …  2020      6  21   11     Sun        173
1296590      Moundsville  …  2020      6  21   11     Sun        173
1296603            Burke  …  2020      6  21   11     Sun        173
1296657  Lanark Village  …  2020      6  21   12     Sun        173

          txn_date  customer_age  masked_accountNumber     BIN
2       2019-01-01          56.0        3885******7661  388594
9       2019-01-01          44.0      2720********1674  272083
16      2019-01-01          66.0      6011********7910  601186
17      2019-01-01          80.0      3565********6143  356542
36      2019-01-01          48.0      3598********4754  359821
…          …            …                   …     …
1296547 2020-06-21          35.0         4755*****1492  475569
1296557 2020-06-21          59.0         4783*****9001  478322
1296590 2020-06-21          42.0      4681********8160  468160
1296603 2020-06-21          69.0   4958**********6883  495858
1296657 2020-06-21          30.0         4464*****2619  446445

[67290 rows x 28 columns]
```

[12]:
```
# 08- Identify outliers and remove those records

size_prev = txn_data.shape

# Calculate z-scores for 'amount' column
z_scores = np.abs((txn_data['amount'] - txn_data['amount'].mean()) /␣
 ↪txn_data['amount'].std())

# Set a threshold for outlier detection (e.g., z-score > 3)
lower_threshold = -3
upper_threshold = 3

# Remove rows with outliers in 'amount' column
txn_data = txn_data[(z_scores >= lower_threshold) & (z_scores <=␣
 ↪upper_threshold)]

# Display the updated dataset
size_after = txn_data.shape
print(f"The size of previous data was - {size_prev} rows and the size of the␣
 ↪new one is - {size_after} rows")
```

```
The size of previous data was - (1296675, 28) rows and the size of the new one
is - (1283937, 28) rows
```

[13]:
```
# 09- Identify any duplicate rows
duplicates = txn_data[txn_data.duplicated()]
print("Duplicates:\n", duplicates)
```

```
Duplicates:
 Empty DataFrame
Columns: [row_id, trans_date_trans_time, merch_name, category, amount, first,
last, gender, street, city, state, zip, job, trans_num, unix_time, merch_lat,
merch_long, is_fraud, year, month, day, hour, weekday, dayofYear, txn_date,
customer_age, masked_accountNumber, BIN]
Index: []

[0 rows x 28 columns]
```

[14]:
```
# 10 - Data accuracy and Check for negative 'amount' values

negative_amounts = txn_data[txn_data['amount'] < 0]
if not negative_amounts.empty:
    # Replace negative values with NaN
    txn_data.loc[txn_data['amount'] < 0, 'amount'] = np.nan
    print("Negative amounts found and replaced with NaN.")
else:
    print("No negative amounts found.")
```

```
No negative amounts found.
```

[15]:
```
# 11 - Get the unique values of the 'BIN' field
unique_bins = txn_data['BIN'].unique()

# Display the unique BIN values
print(unique_bins)
```

```
['270318' '630423' '388594' '353409' '375534' '476726' '300746' '601136'
 '492271' '272083' '464289' '377234' '180042' '555985' '351486' '601199'
 '601186' '356542' '234824' '495682' '446977' '230533' '180048' '630441'
 '442878' '343464' '374930' '433423' '422599' '426005' '180094' '459973'
 '630412' '271220' '374125' '349612' '359821' '213141' '305612' '478991'
 '676372' '300114' '464225' '180067' '213126' '226673' '355362' '351105'
 '411026' '426577' '358379' '213193' '604955' '472583' '304042' '213178'
 '235627' '356873' '450914' '354756' '443309' '272043' '471656' '353691'
 '352566' '470786' '466631' '224873' '451282' '350150' '302730' '355888'
 '304876' '571465' '378006' '341283' '448113' '468726' '545677' '498113'
 '601189' '653462' '376028' '571365' '351875' '357757' '213112' '359232'
 '468352' '350208' '651134' '419847' '442348' '351823' '659325' '601149'
 '414923' '639046' '223561' '510480' '180040' '412976' '401810' '447432'
 '303762' '348789' '180014' '226080' '497942' '375767' '601138' '460707'
 '422973' '502012' '501882' '302704' '468363' '356787' '601110' '350659'
 '351960' '676245' '363604' '354329' '601150' '495858' '356594' '418653'
 '651506' '354168' '213175' '367214' '351137' '381990' '213113' '465387'
 '358363' '354116' '348379' '374821' '304424' '213124' '464684' '223388'
 '462356' '374656' '499773' '368902' '453856' '351489' '408909' '441058'
 '357766' '355865' '231806' '499810' '407977' '474071' '601155' '411655'
 '357578' '456000' '439352' '448825' '300929' '430248' '353374' '496985'
 '303446' '354155' '476140' '423955' '651721' '356033' '436401' '482665'
 '359511' '474599' '357643' '356609' '359889' '499234' '657356' '357338'
 '226493' '353433' '444953' '360781' '422034' '471082' '411976' '417139'
 '377026' '353317' '223533' '419574' '456189' '601123' '468169' '340953'
 '373905' '492637' '213154' '450253' '372246' '487378' '341546' '471079'
 '302429' '581686' '458626' '302352' '458765' '364858' '355916' '659207'
 '465849' '501802' '456039' '498830' '545671' '676195' '483961' '213180'
 '485585' '343819' '377550' '356121' '374497' '213195' '476942' '351923'
 '272001' '659788' '352181' '435579' '359621' '389476' '650698' '404850'
 '409970' '445236' '350604' '404503' '440001' '486229' '352959' '228405'
 '303283' '425539' '213186' '431958' '374238' '450538' '271391' '213161'
 '601193' '472724' '377895' '300820' '345060' '445773' '467172' '180068'
 '224254' '414760' '228506' '377654' '601143' '342360' '479729' '430247'
 '180072' '503874' '359311' '354359' '458681' '385885' '416938' '353112'
 '375248' '369135' '213156' '356599' '247508' '180065' '180069' '347073'
 '493997' '354689' '228623' '301435' '358704' '303710' '429274' '655424'
 '501831' '354667' '447156' '228874' '225491' '659393' '446445' '355258'
 '495258' '550108' '483699' '415502' '357554' '304076' '300303' '180036'
 '466199' '213102' '432892' '479893' '493346' '400377' '601114' '300423'
 '213157' '473684' '474288' '412873' '226976' '449791' '415894' '370348'
 '427723' '378278' '538865' '495164' '301534' '445143' '385304' '676248'
```

'345832' '354021' '351328' '446745' '475603' '342952' '304913' '400604'
'639030' '406309' '359519' '561942' '356072' '650131' '352753' '535954'
'559634' '355484' '302905' '213163' '601197' '301813' '480755' '350417'
'302387' '351781' '458513' '676326' '639023' '222937' '213173' '464039'
'359073' '229744' '365815' '630425' '676173' '604230' '451424' '356769'
'676298' '437899' '501828' '357297' '301759' '378904' '356196' '355121'
'213136' '251231' '652891' '350194' '422049' '361538' '223118' '180046'
'354086' '304341' '478322' '601160' '385804' '300290' '581508' '425999'
'601162' '604278' '359733' '601154' '355967' '487926' '459928' '371284'
'301352' '301973' '359830' '501899' '401000' '425407' '581293' '353852'
'450310' '344709' '659724' '675990' '213120' '477706' '378858' '471574'
'302349' '304282' '341542' '424792' '406713' '257670' '303433' '300443'
'502038' '352389' '302305' '453699' '353380' '356069' '225205' '356825'
'371226' '630424' '410431' '630484' '654098' '468160' '352682' '270697'
'421007' '448242' '402561' '573860' '300267' '222982' '414963' '676281'
'442433' '301318' '654473' '630471' '423943' '372382' '356637' '272089'
'370877' '359635' '470899' '245082' '343472' '302635' '359293' '342319'
'372509' '302669' '406057' '601191' '302488' '355661' '340951' '653889'
'353580' '437926' '580954' '676234' '601165' '601134' '584673' '407051'
'456154' '676369' '465317' '601168' '382956' '370612' '448894' '463495'
'486131' '445083' '503848' '371034' '568279' '375904' '450000' '487401'
'227491' '265785' '451835' '442815' '342351' '406659' '675945' '458493'
'676275' '353681' '354007' '484131' '358329' '350016' '356079' '345933'
'489042' '502049' '380144' '376656' '490468' '430102' '659224' '305516'
'444058' '347612' '652677' '238346' '213198' '601167' '443088' '358796'
'447684' '305606' '659286' '352559' '463533' '357602' '604905' '359801'
'499626' '354510' '515490' '367226' '512741' '406997' '604162' '481078'
'422370' '358925' '359923' '461331' '305181' '418183' '656445' '417395'
'400398' '441272' '357779' '445195' '352055' '228374' '429493' '354920'
'343668' '371009' '571844' '475569' '305964' '422841' '479262' '434495'
'358275' '528928' '376445' '490884' '342035' '433553' '351361' '357303'
'359191' '601194' '630469' '460163' '460590' '303640' '180084' '630451'
'410200' '419110' '305016' '676309' '375848' '654767' '653844' '356031'
'358505' '513048' '224834' '377113' '228881' '418981' '480970' '345389'
'659597' '358600' '261052' '652644' '346273' '429290' '271949' '352457'
'639095' '650611' '478810' '463306' '430663' '180081' '180058' '416975'
'491718' '213191' '429404' '350810' '431136' '380520' '180049' '420811'
'444636' '450144' '499049' '431213' '213174' '481083' '353050' '354460'
'601172' '601170' '601130' '225479' '565399' '417068' '415875' '583699'
'344342' '351866' '359798' '356383' '375974' '229116' '461005' '305464'
'358574' '353301' '434253' '485952' '447715' '456136' '361926' '351752'
'676102' '213148' '356024' '449745' '404443' '438491' '359792' '222200'
'305108' '601139' '515205' '482236' '490662' '601169' '370818' '462945'
'346208' '567868' '490062' '180018' '213114' '350237' '675961' '354757'
'455059' '653965' '302354' '355151' '435813' '436538' '465811' '303602'
'442516' '676148' '301996' '303738' '358928' '426620' '415800' '417971'
'355972' '652318' '422621' '343746' '180047' '577588' '601158' '350096'
'420969' '446203' '301031' '354216' '356418' '601132' '601147' '375082'

```
 '222767' '472567' '180017' '352141' '650219' '359863' '229600' '424495'
 '351236' '676314' '371683' '676308' '400567' '180031' '427916' '213199'
 '301437' '371985' '372520' '554063' '450992' '376944' '491722' '213107'
 '639077' '376012' '356226' '437733' '356519' '458757' '356687' '476420'
 '300331' '304460' '355744' '468914' '485548' '352384' '573283' '560881'
 '180064' '301184' '225623' '357759' '213153' '180056' '497222' '379897'
 '409245' '340187' '305182' '468274' '442780' '304270' '387974' '496100'
 '487836' '434878' '482402' '570273' '404009' '412453' '245407' '417809'
 '480039' '373043' '652437' '426012' '498032' '412802' '356541' '465726'
 '439096' '465100' '357614' '355055' '352840' '357651' '498984' '353471'
 '301507' '303573' '558056' '358695' '305411' '351350' '271401' '410315'
 '213155' '356483' '377993' '402622' '385443' '305701' '483904' '357920'
 '213125' '467061' '356752' '356267' '376262' '222215' '495792' '340103'
 '601119' '358113' '353779' '180011' '348253' '379141' '438352' '471346'
 '659673' '601125' '455510' '375237' '675909' '353160' '497353' '355481'
 '571314' '676292' '304083' '442607' '457063' '493585' '604229' '430815'
 '359339' '228087' '656909' '419583' '352823' '676118' '487400' '357628'
 '449249' '655039' '652695' '460015' '469816' '491470' '413445' '498692'
 '380575' '439668' '489706' '235812' '430658' '350136' '653532' '461006'
 '354001' '456282' '480644' '476012' '604870' '352706' '222410' '346243'
 '377264' '437337' '358309' '358240' '373213' '226822' '377834' '541005'
 '425711' '354557' '388175' '473431' '378262' '577891' '466955' '420423'
 '471401' '415721' '449267' '459356' '354388' '354733' '491181' '465962'
 '601173' '418019' '407617' '501851' '530164' '514404' '354871' '601182'
 '345225' '424352' '484424' '448117' '501894' '228727' '416869' '416287'
 '304997' '229596' '497545' '352448' '676179' '676327' '352126' '385342'
 '349813' '459802' '445748' '180038' '355442' '657777' '354643' '651777'
 '462912' '352978' '354282' '650024' '375623' '352804' '353521' '601151'
 '271601' '422562' '445713' '359094' '180097' '501818' '601109' '340214'
 '431841' '357620' '489309' '350128' '457525' '474886' '356279' '503886']
```

[16]:
```
# 12- Get the count of occurrences for each unique value in the 'BIN' field
bin_counts = txn_data['BIN'].value_counts()

# Display the count of occurrences
print(bin_counts)
```

```
601136    6133
601110    5601
213161    4538
601138    4023
180067    3553
          ...
271601       2
353521       1
349813       1
657777       1
601173       1
```

```
Name: BIN, Length: 960, dtype: int64
```

[17]:
```
# After all Transformation and the final transactional data
txn_data
```

[17]:

	row_id	trans_date_trans_time	merch_name	\
0	0	2019-01-01 00:00:18	fraud_Rippin, Kub and Mann	
1	1	2019-01-01 00:00:44	fraud_Heller, Gutmann and Zieme	
2	2	2019-01-01 00:00:51	fraud_Lind-Buckridge	
3	3	2019-01-01 00:01:16	fraud_Kutch, Hermiston and Farrell	
4	4	2019-01-01 00:03:06	fraud_Keeling-Crist	
…	…	…	…	
1296670	1296670	2020-06-21 12:12:08	fraud_Reichel Inc	
1296671	1296671	2020-06-21 12:12:19	fraud_Abernathy and Sons	
1296672	1296672	2020-06-21 12:12:32	fraud_Stiedemann Ltd	
1296673	1296673	2020-06-21 12:13:36	fraud_Reinger, Weissnat and Strosin	
1296674	1296674	2020-06-21 12:13:37	fraud_Langosh, Wintheiser and Hyatt	

	category	amount	first	last	gender	\
0	misc_net	4.97	Jennifer	Banks	F	
1	grocery_pos	107.23	Stephanie	Gill	F	
2	entertainment	220.11	Edward	Sanchez	M	
3	gas_transport	45.00	Jeremy	White	M	
4	misc_pos	41.96	Tyler	Garcia	M	
…	…	…	…	…	…	
1296670	entertainment	15.56	Erik	Patterson	M	
1296671	food_dining	51.70	Jeffrey	White	M	
1296672	food_dining	105.93	Christopher	Castaneda	M	
1296673	food_dining	74.90	Joseph	Murray	M	
1296674	food_dining	4.30	Jeffrey	Smith	M	

	street	city	…	year	\
0	561 Perry Cove	Moravian Falls	…	2019	
1	43039 Riley Greens Suite 393	Orient	…	2019	
2	594 White Dale Suite 530	Malad City	…	2019	
3	9443 Cynthia Court Apt. 038	Boulder	…	2019	
4	408 Bradley Rest	Doe Hill	…	2019	
…	…	…	…	…	
1296670	162 Jessica Row Apt. 072	Hatch	…	2020	
1296671	8617 Holmes Terrace Suite 651	Tuscarora	…	2020	
1296672	1632 Cohen Drive Suite 639	High Rolls Mountain Park	…	2020	
1296673	42933 Ryan Underpass	Manderson	…	2020	
1296674	135 Joseph Mountains	Sula	…	2020	

	month	day	hour	weekday	dayofYear	txn_date	customer_age	\
0	1	1	0	Tue	1	2019-01-01	30.0	
1	1	1	0	Tue	1	2019-01-01	40.0	

```
2            1   1    0     Tue          1 2019-01-01          56.0
3            1   1    0     Tue          1 2019-01-01          51.0
4            1   1    0     Tue          1 2019-01-01          32.0
…        …   ..  …    …        …        …           …
1296670      6  21   12     Sun        173 2020-06-21          58.0
1296671      6  21   12     Sun        173 2020-06-21          40.0
1296672      6  21   12     Sun        173 2020-06-21          52.0
1296673      6  21   12     Sun        173 2020-06-21          39.0
1296674      6  21   12     Sun        173 2020-06-21          24.0

         masked_accountNumber     BIN
0            2703********2095  270318
1                6304****7322  630423
2              3885******7661  388594
3            3534********0240  353409
4             3755*******3984  375534
…                        …     …
1296670        3026******4123  302635
1296671      6011********6997  601114
1296672      3514********4695  351486
1296673      2720********6919  272001
1296674   4292**********3207  429290

[1283937 rows x 28 columns]
```

[ ]: