PPP Fraud Detection and Exploratory Analysis

Advanced Data Analytics and Machine Learning in Python

Authors: Aayush Bakre, Ajanya Sharma

Table of Contents

Problem Discription

alt text/cloudfront-us-east-1.images.arcpublishing.com/gray/YT67CCYJY5FDNJDURKRAGZYETY.jpg)

PPP Loan Fraud Detection

The Paycheck Protection Program (PPP) is a loan program by the U.S. government in 2020 to help certain businesses, self-employed workers, sole proprietors, nonprofit organizations, and tribal businesses keep paying their workers during the COVID-19 pandemic. PPP loans are private loans with low interest rates that can be used to cover payroll costs, rent, interest, and utilities. The loan amount is based on the average monthly payroll costs of the applicant,business type and can be forgiven if the business is unable to sustain duing the pandemic.The program is run by the U.S. Small Business Administration and the deadline to apply for a PPP loan was March 31, 2021.

The project is aimed at exploring loan data from the Paycheck Protection Program (PPP), which provided relief to small and medium-sized businesses during the COVID-19 pandemic. The primary objective is to create graphical visualizations of the data and apply anomaly detection methods to identify potentially fraudulent loans. The project outline suggests a few starting points, including reviewing PPP loan program eligibility criteria, downloading the full PPP loan dataset and NAICS codes data dictionary for businesses, summarizing the data through tabular summaries and graphical visualizations, investigating loans that have a high potential for fraud by grouping together loans in common categories and identifying outlier loans, and exploring the use of traditional unsupervised learning techniques such as anomaly detection.

Evaluation

The dataset is a collection of entries on the PPP Loan Fraud application form. There is no training or a test dataset. Initial review doesn't point towards any strong correlations and predictors that would categorize the problem as a predictive or a dependence exerceise. At first glance, the dataset seems to be a good fit for unsupervised outlier detection methods just as the Project Brief suggests.

Approach to Analysis

In order to better understand the PPP Loan dataset, we investigated the data and performed initial exploratory analysis along with data visualizations.

Bringing in the dataset and cleaning the data, which includes handling missing values and fixing inconsistent formatting using tranformations, will be the first steps in getting the data ready for analysis. Following that, we normalized selective features of the dataset to ensure meaningful variability that would better support the analysis.

In order to acquire a deeper understanding of the data, we visualized it using a variety of visualization techniques, including line charts, histograms, correlation matrix, and heatmaps.

Fraud Detection Techniques

We employed two main approaches to Fraud Detection.

1. Calculating risk scores for each loan application by comparing its key attributes (Jobs Reported, Loan Amounts, and Loan Amounts per Employee) to other businesses of the same size in the same industry to rank extremely atypical loan applications. We also checked the dataset against specific conditionalities and logic, which we believed if the loan applications are displaying, would make those applications fraudulent.

2. We applied the unsupervised machine-learning algorithm, Isolation Forest anomaly detection, to forecast anomaly scores for loan applications to find the most anomalous loans, which could be signs of PPP loan fraud.

In the cases above, we treated the high-risk and anomaly scores over a threshold as potentially fraudulent.

Finally, we also conducted exploratory studies on the resulting fraud-identified data to find additional patterns, connections, and trends.

The way this study is designed, we approached it with the intention to aid analysts at SBA and other relevant agencies to help investigate fraud and gain insight into the PPP Loan dataset with the identification of probable PPP loan frauds.

Dataset

The dataset with PPP data is sourced from the official SBA website which can be accessed using the following URL: https://data.sba.gov/dataset/ppp-foia/resource/aab8e9f9-36d1-42e1-b3ba-e59c79f1d7f0?inner_span=True

The dataset with NAICS code is sourced from https://www.sba.gov/document/support-table-size-standards

Setup Imports and Variables

In [ ]:

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
import numpy as np
import pandas as pd

# Set the global default size of matplotlib figures
plt.rc('figure', figsize=(10, 5))

# Size of matplotlib figures that contain subplots
fizsize_with_subplots = (10, 10)

# Size of matplotlib histogram bins
bin_size = 10
```

Explore the Data

Read the Data

Read the first few entries

In [2]:

```
# Read in CSV file
df = pd.read_csv('public_150k_plus_230101.csv')

# View first few rows of data
df.head()
```

Out[2]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	BorrowerAc
0	9547507704	5/1/20	464	PPP	SUMTER COATINGS, INC.	2410 High
1	9777677704	5/1/20	464	PPP	PLEASANT PLACES, INC.	7684 So
2	5791407702	5/1/20	1013	PPP	BOYER CHILDREN'S CLINIC	1850 BOYE
3	6223567700	5/1/20	920	PPP	KIRTLEY CONSTRUCTION INC	1661 M RAN
4	9662437702	5/1/20	101	PPP	AERO BOX LLC	

5 rows × 53 columns

Read the last five entries

In [3]:

```
# View last few rows of data
df.tail()
```

Out[3]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	Bor
968526	4395967002	4/3/20	897	PPP	ROY E PAULSON, JR., P.C.	1
968527	6985647108	4/14/20	897	PPP	SWEETWATER COUNTY CHILD DEVELOPMENTAL CENTER, ...	1
968528	7996438405	2/12/21	897	PPS	ELECTRICAL SYSTEMS OF WYOMING INC	
968529	9054647103	4/15/20	897	PPP	EDEN LIFE CARE	Str
968530	9184687004	4/9/20	897	PPP	S & S JOHNSON ENTERPRISES INC	

5 rows × 53 columns

Observed misread values in columns with dates. So next we call information about data types and missing values

In [4]:

```
# Get information about data types and missing values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 968531 entries, 0 to 968530
Data columns (total 53 columns):
 #   Column                       Non-Null Count   Dtype  
---  ------                       --------------   -----  
 0   LoanNumber                   968531 non-null  int64  
 1   DateApproved                 968531 non-null  object 
 2   SBAOfficeCode                968531 non-null  int64  
 3   ProcessingMethod             968531 non-null  object 
 4   BorrowerName                 968527 non-null  object 
 5   BorrowerAddress              968517 non-null  object 
 6   BorrowerCity                 968519 non-null  object 
 7   BorrowerState                968518 non-null  object 
 8   BorrowerZip                  968518 non-null  object 
 9   LoanStatusDate               942818 non-null  object 
 10  LoanStatus                   968531 non-null  object 
 11  Term                         968531 non-null  int64  
 12  SBAGuarantyPercentage        968531 non-null  int64  
 13  InitialApprovalAmount        968531 non-null  float64
 14  CurrentApprovalAmount        968531 non-null  float64
 15  UndisbursedAmount            968484 non-null  float64
 16  FranchiseName                35405 non-null   object 
 17  ServicingLenderLocationID    968531 non-null  int64  
 18  ServicingLenderName          968531 non-null  object 
 19  ServicingLenderAddress       968531 non-null  object 
 20  ServicingLenderCity          968531 non-null  object 
 21  ServicingLenderState         968531 non-null  object 
 22  ServicingLenderZip           968531 non-null  object 
 23  RuralUrbanIndicator          968531 non-null  object 
 24  HubzoneIndicator             968531 non-null  object 
 25  LMIIndicator                 968531 non-null  object 
 26  BusinessAgeDescription       968530 non-null  object 
 27  ProjectCity                  968518 non-null  object 
 28  ProjectCountyName            968474 non-null  object 
 29  ProjectState                 968522 non-null  object 
 30  ProjectZip                   968517 non-null  object 
 31  CD                           968485 non-null  object 
 32  JobsReported                 968530 non-null  float64
 33  NAICSCode                    961903 non-null  float64
 34  Race                         968531 non-null  object 
 35  Ethnicity                    968531 non-null  object 
 36  UTILITIES_PROCEED            339377 non-null  float64
 37  PAYROLL_PROCEED              966699 non-null  float64
 38  MORTGAGE_INTEREST_PROCEED    46183 non-null   float64
 39  RENT_PROCEED                 99533 non-null   float64
 40  REFINANCE_EIDL_PROCEED       22855 non-null   float64
 41  HEALTH_CARE_PROCEED          57446 non-null   float64
 42  DEBT_INTEREST_PROCEED        31717 non-null   float64
 43  BusinessType                 967809 non-null  object 
 44  OriginatingLenderLocationID  968531 non-null  int64  
 45  OriginatingLender            968531 non-null  object 
 46  OriginatingLenderCity        968531 non-null  object 
 47  OriginatingLenderState       968531 non-null  object 
 48  Gender                       968531 non-null  object 
 49  Veteran                      968531 non-null  object 
 50  NonProfit                    59341 non-null   object 
 51  ForgivenessAmount            938885 non-null  float64
 52  ForgivenessDate              938885 non-null  object 
dtypes: float64(13), int64(6), object(34)
memory usage: 391.6+ MB
```

Now we call for summary statistics for the variables

```
In [5]: # Get summary statistics for numerical columns
        df.describe()
```

Out[5]:

	LoanNumber	SBAOfficeCode	Term	SBAGuarantyPercentage	InitialApprovalAr
count	9.685310e+05	968531.000000	968531.000000	968531.0	9.685310
mean	5.427137e+09	571.519065	36.377761	100.0	5.322537
std	2.551313e+09	263.024816	17.291796	0.0	7.442514
min	1.000007e+09	101.000000	0.000000	100.0	0.000000
25%	3.271108e+09	373.000000	24.000000	100.0	2.002000
50%	5.400677e+09	515.000000	24.000000	100.0	2.951770
75%	7.546303e+09	811.000000	60.000000	100.0	5.402000
max	9.999007e+09	1094.000000	180.000000	100.0	1.000000

We retrieve the exact counts of missing values for each variable.

```
In [6]: # Count the number of rows with missing values in each column
        num_missing = df.isnull().sum(axis=0)

        print(f"Total number of rows with missing values: {num_missing}")
```

```
Total number of rows with missing values: LoanNumber
0
DateApproved                        0
SBAOfficeCode                       0
ProcessingMethod                    0
BorrowerName                        4
BorrowerAddress                    14
BorrowerCity                       12
BorrowerState                      13
BorrowerZip                        13
LoanStatusDate                  25713
LoanStatus                          0
Term                                0
SBAGuarantyPercentage               0
InitialApprovalAmount               0
CurrentApprovalAmount               0
UndisbursedAmount                  47
FranchiseName                  933126
ServicingLenderLocationID           0
ServicingLenderName                 0
ServicingLenderAddress              0
ServicingLenderCity                 0
ServicingLenderState                0
ServicingLenderZip                  0
RuralUrbanIndicator                 0
HubzoneIndicator                    0
LMIIndicator                        0
BusinessAgeDescription              1
ProjectCity                        13
ProjectCountyName                  57
ProjectState                        9
ProjectZip                         14
CD                                 46
JobsReported                        1
NAICSCode                        6628
Race                                0
Ethnicity                           0
UTILITIES_PROCEED              629154
PAYROLL_PROCEED                  1832
MORTGAGE_INTEREST_PROCEED      922348
RENT_PROCEED                   868998
REFINANCE_EIDL_PROCEED         945676
HEALTH_CARE_PROCEED            911085
DEBT_INTEREST_PROCEED          936814
BusinessType                      722
OriginatingLenderLocationID         0
OriginatingLender                   0
OriginatingLenderCity               0
OriginatingLenderState              0
Gender                              0
Veteran                             0
NonProfit                      909190
ForgivenessAmount               29646
ForgivenessDate                 29646
dtype: int64
```

Total number of observations with at least one missing variable

In [7]:

```
# Count the number of rows with missing values
num_rows_missing = (df.isnull().sum(axis=1) > 0).sum()

print(f"Total number of rows with missing values: {num_rows_missing}")
```

```
Total number of rows with missing values: 968530
```

Total number of rows with missing values is the same as the total observations in the dataset. This means that all observations have at least one null value. Upon looking closely at the dataset and the data dictionary, we can say that it may be because some of the variables are mutually exclusive responses to form questions.

Data Transformation and Feature Introduction

Given that we now have an overview of the dataset. We can now carried out initial tranformations which will allow us to break down and visualize individual elements of the dataset such as processing methods, approval dates, forgiveness dates, and loan amounts with respect to business size and employees. We can also use this opportunity to construct feature variables that may help us derive a more meaningful interpretation from the analysis.

First, we transform columns with misinterpreted dates to correctly identify and validaate date values

In [8]:

```
# Transform columns with date values from Object DType to DateTime DType
df[['DateApproved', 'LoanStatusDate', 'ForgivenessDate']] = df[['DateApproved',
df.head()
```

Out[8]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	BorrowerAc
0	9547507704	2020-05-01	464	PPP	SUMTER COATINGS, INC.	2410 High
1	9777677704	2020-05-01	464	PPP	PLEASANT PLACES, INC.	7684 So
2	5791407702	2020-05-01	1013	PPP	BOYER CHILDREN'S CLINIC	1850 BOYE
3	6223567700	2020-05-01	920	PPP	KIRTLEY CONSTRUCTION INC	1661 M RAN
4	9662437702	2020-05-01	101	PPP	AERO BOX LLC	

5 rows × 53 columns

Given the high variability in the loan amounts sanctioned, lets introduce a feature containing normalized loan amounts by adjusting the loan amounts for the number of employees disclosed.

In [9]:
```
df['loan_amount_per_employee'] = df['CurrentApprovalAmount'] / df['JobsReportec
#do box plot, try to find min, max and any outliers
```

In [10]:
```
df
```

Out[10]:
```
        LoanNumber  DateApproved  SBAOfficeCode  ProcessingMethod   BorrowerName  Bor

     0  9547507704    2020-05-01            464               PPP         SUMTER   24
                                                                   COATINGS, INC.
     1  9777677704    2020-05-01            464               PPP       PLEASANT
                                                                    PLACES, INC.
                                                                           BOYER
     2  5791407702    2020-05-01           1013               PPP      CHILDREN'S   18
                                                                          CLINIC
                                                                         KIRTLEY
     3  6223567700    2020-05-01            920               PPP    CONSTRUCTION
                                                                             INC
     4  9662437702    2020-05-01            101               PPP   AERO BOX LLC
   ...         ...           ...            ...               ...            ...
968526  4395967002    2020-04-03            897               PPP ROY E PAULSON,   1(
                                                                       JR., P.C.
                                                                      SWEETWATER
968527  6985647108    2020-04-14            897               PPP   COUNTY CHILD
                                                                   DEVELOPMENTAL
                                                                     CENTER, ...
                                                                      ELECTRICAL
968528  7996438405    2021-02-12            897               PPS     SYSTEMS OF
                                                                    WYOMING INC
968529  9054647103    2020-04-15            897               PPP EDEN LIFE CARE  Str
                                                                    S & S JOHNSON
968530  9184687004    2020-04-09            897               PPP    ENTERPRISES
                                                                             INC

968531 rows × 54 columns
```

Let's import NAICS Industry Descriptions, Industry Size standards in millions of dollars, and Size standards in number of employees based on NAICS Code into the dataframe.

In [11]:
```
naicsdata=pd.read_excel('Table of Size Standards.xlsx',sheet_name='table_of_siz
naicsdata.head()
```

Out[11]:

	NAICS Codes	NAICS Industry Description	Size standards \nin millions of dollars	Size standards in number of employees	Footnotes
0	NaN	Sector 11 – Agriculture, Forestry, Fishing and...	NaN	NaN	NaN
1	Subsector 111 – Crop Production	NaN	NaN	NaN	NaN
2	111110	Soybean Farming	2.25	NaN	NaN
3	111120	Oilseed (except Soybean) Farming	2.25	NaN	NaN
4	111130	Dry Pea and Bean Farming	2.75	NaN	NaN

In [12]:

```
naicsdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1105 entries, 0 to 1104
Data columns (total 5 columns):
 #   Column                                     Non-Null Count  Dtype  
---  ------                                     --------------  -----  
 0   NAICS Codes                                1082 non-null   object 
 1   NAICS Industry Description                 1019 non-null   object 
 2   Size standards 
in millions of dollars  513 non-null    object 
 3   Size standards in number of employees      483 non-null    float64
 4   Footnotes                                  37 non-null     object 
dtypes: float64(1), object(4)
memory usage: 43.3+ KB
```

In [13]:

```
unique_values = naicsdata.nunique()

# Print the result
print(unique_values)
```

```
NAICS Codes                                1082
NAICS Industry Description                 1015
Size standards \nin millions of dollars      74
Size standards in number of employees        27
Footnotes                                    17
dtype: int64
```

In [14]:

```
naicsdata.dropna(subset=['NAICS Codes'])
```

Out[14]:

	NAICS Codes	NAICS Industry Description	Size standards \nin millions of dollars	Size standards in number of employees	Footnotes
1	Subsector 111 – Crop Production	NaN	NaN	NaN	NaN
2	111110	Soybean Farming	2.25	NaN	NaN
3	111120	Oilseed (except Soybean) Farming	2.25	NaN	NaN
4	111130	Dry Pea and Bean Farming	2.75	NaN	NaN
5	111140	Wheat Farming	2.25	NaN	NaN
...	...	...	...	...	...
1098	813910	Business Associations	15.5	NaN	NaN
1099	813920	Professional Organizations	23.5	NaN	NaN
1100	813930	Labor Unions and Similar Labor Organizations	16.5	NaN	NaN
1101	813940	Political Organizations	14	NaN	NaN
1102	813990	Other Similar Organizations (except Business, ...	13.5	NaN	NaN

1082 rows × 5 columns

In [15]:

```
df = df.dropna(subset=['NAICSCode'])

missing_count = df['NAICSCode'].isna().sum()
missing_count
```

Out[15]:

```
0
```

Creating columns for NAICS Industry Description, Size standards in millions of dollars, and Size standards in number of employees

In [16]:

```
# Create a dictionary mapping NAICS Codes to NAICS Industry Description in naic
naics_dict = dict(zip(naicsdata['NAICS Codes'], naicsdata['NAICS Industry Descr

# Create a dictionary mapping NAICS Codes to Size standards \nin millions of do
naics_dict2 = dict(zip(naicsdata['NAICS Codes'], naicsdata['Size standards \nin

# Create a dictionary mapping NAICS Codes to Size standards in number of employ
naics_dict3 = dict(zip(naicsdata['NAICS Codes'], naicsdata['Size standards in n

# Create a new column in df (public_150k_plus_230101.csv) with the values from
df['NAICS Industry Description'] = df['NAICSCode'].map(naics_dict)

# Create a new column in df (public_150k_plus_230101.csv) with the values from
df['Size standards \nin millions of dollars'] = df['NAICSCode'].map(naics_dict2
# Create a new column in df (public_150k_plus_230101.csv) with the values from
df['Size standards in number of employees'] = df['NAICSCode'].map(naics_dict3)

df.head()
```

```
/var/folders/f1/8rl13vpj76b49z9s20c53bp80000gn/T/ipykernel_13709/1652607285.p
y:11: 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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['NAICS Industry Description'] = df['NAICSCode'].map(naics_dict)
/var/folders/f1/8rl13vpj76b49z9s20c53bp80000gn/T/ipykernel_13709/1652607285.p
y:14: 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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Size standards \nin millions of dollars'] = df['NAICSCode'].map(naics_di
ct2)
/var/folders/f1/8rl13vpj76b49z9s20c53bp80000gn/T/ipykernel_13709/1652607285.p
y:17: 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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Size standards in number of employees'] = df['NAICSCode'].map(naics_dict
3)
```

Out[16]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	BorrowerAc
0	9547507704	2020-05-01	464	PPP	SUMTER COATINGS, INC.	2410 High
1	9777677704	2020-05-01	464	PPP	PLEASANT PLACES, INC.	7684 So
3	6223567700	2020-05-01	920	PPP	KIRTLEY CONSTRUCTION INC	1661 M RAN
4	9662437702	2020-05-01	101	PPP	AERO BOX LLC	
5	9774337701	2020-05-01	101	PPP	HUDSON EXTRUSIONS INC.	

5 rows × 57 columns

Looking for observed format inconsistencies in NAICSCode columns

In [17]:

```
# create a boolean mask to filter the data
mask = df['Size standards in number of employees'].isnull()

# use boolean indexing to filter the data and group by 'NAICS Codes'
naics_nullcounts = df.loc[mask].groupby('NAICSCode').size()

# print the counts of unique values in 'NAICS Codes' for rows where 'Size stanc
naics_nullcounts
```

Out[17]:

```
NAICSCode
111110.0     203
111120.0      34
111130.0      17
111140.0      54
111150.0     204
            ...
926150.0      40
927110.0      27
928110.0      23
928120.0      30
999990.0    4090
Length: 727, dtype: int64
```

In [18]:

```
missing_count = df['Size standards in number of employees'].isna().sum()
missing_count
```

Out[18]:

```
793572
```

In [19]:

```
# convert the 'NAICSCode' column to integer and then back to string
df['NAICSCode'] = df['NAICSCode'].astype(int).astype(str)

# show the transformed column
print(df['NAICSCode'])
```

```
0         325510
1         561730
3         236115
4         484210
5         326199
           ...
968526    621210
968527    624410
968528    238210
968529    621610
968530    722511
Name: NAICSCode, Length: 961903, dtype: object
```

```
/var/folders/f1/8rl13vpj76b49z9s20c53bp80000gn/T/ipykernel_13709/865044654.py:
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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['NAICSCode'] = df['NAICSCode'].astype(int).astype(str)
```

Imputing Null values with adjusted means for JobsReported by Industry Type as will be required for further analysis

In [20]:

```
# group by NAICSCode and calculate the mean of JobsReported
mean_jobs_reported = df.groupby('NAICSCode')['JobsReported'].mean()

# fill null values in Size standards in number of employees column with mean of
df['Size standards in number of employees'] = df.groupby('NAICSCode')['Size sta
```

```
/var/folders/f1/8rl13vpj76b49z9s20c53bp80000gn/T/ipykernel_13709/2122538760.p
y:5: 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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Size standards in number of employees'] = df.groupby('NAICSCode')['Size
standards in number of employees'].apply(lambda x: x.fillna(mean_jobs_reported
[x.name]))
```

In [21]:

```
missing_count = df['Size standards in number of employees'].isna().sum()
missing_count
```

Out[21]: 0

Imputing Null values with adjusted means for CurrentApprovalAmount by Industry Type as will be required for further analysis

In [22]:

```
# group by NAICSCode and calculate the mean of # group by NAICSCode and calcula
mean_CurrentApprovalAmount = df.groupby('NAICSCode')['CurrentApprovalAmount'].m

# fill null values in CurrentApprovalAmount column with mean of CurrentApproval
df['CurrentApprovalAmount'] = df.groupby('NAICSCode')['CurrentApprovalAmount'].
```

```
/var/folders/f1/8rl13vpj76b49z9s20c53bp80000gn/T/ipykernel_13709/2207975816.p
y:5: 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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['CurrentApprovalAmount'] = df.groupby('NAICSCode')['CurrentApprovalAmoun
t'].apply(lambda x: x.fillna(mean_jobs_reported[x.name]))
```

In [23]:

```
missing_count_CurrentApprovalAmount = df['CurrentApprovalAmount'].isna().sum()
missing_count_CurrentApprovalAmount
```

Out[23]: 0

Imputing Null values with adjusted means for Loan Amopunt Per Employee by Industry Type as will be required for further analysis

In [24]:

```
# group by NAICSCode and calculate the mean of # group by NAICSCode and calcula
mean_loan_amount_per_employee = df.groupby('NAICSCode')['loan_amount_per_employ
```

```
# fill null values in loan_amount_per_employee column with mean of loan_amount_
df['loan_amount_per_employee'] = df.groupby('NAICSCode')['loan_amount_per_emplo
```

```
/var/folders/f1/8rl13vpj76b49z9s20c53bp80000gn/T/ipykernel_13709/125080320.py:
5: 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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['loan_amount_per_employee'] = df.groupby('NAICSCode')['loan_amount_per_em
ployee'].apply(lambda x: x.fillna(mean_loan_amount_per_employee[x.name]))
```

In [25]:

```
missing_count_loan_amount_per_employee = df['loan_amount_per_employee'].isna().
missing_count_loan_amount_per_employee
```

Out[25]:

```
0
```

Now that we have executed the initial transformations, it is time to visualize the data!

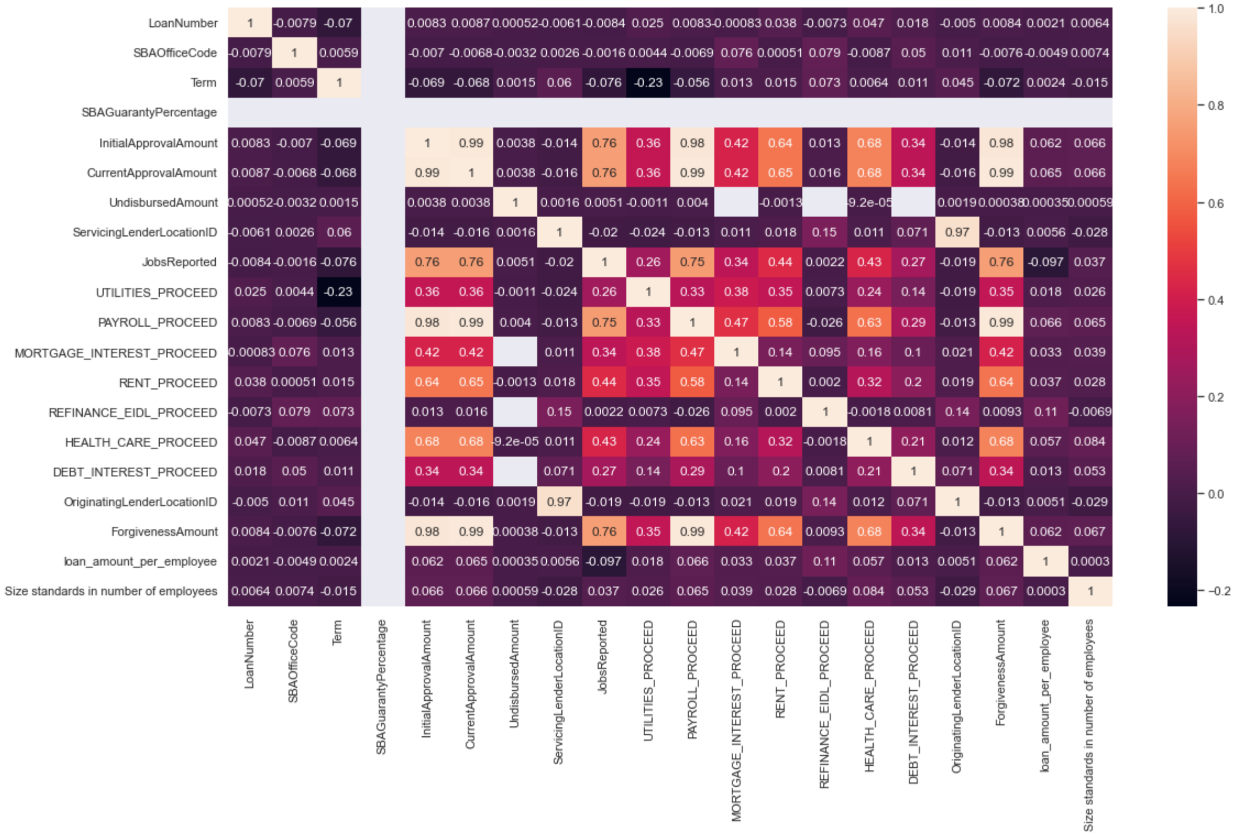
Correlation Matrix to look at variable dependencies

In [26]:

```
#correlation Matrix to study correlation between different variables
matrix=df.corr()
f,ax=plt.subplots(figsize=(18,10))
sns.heatmap(matrix, annot=True)
```

Out[26]:

```
<AxesSubplot:>
```



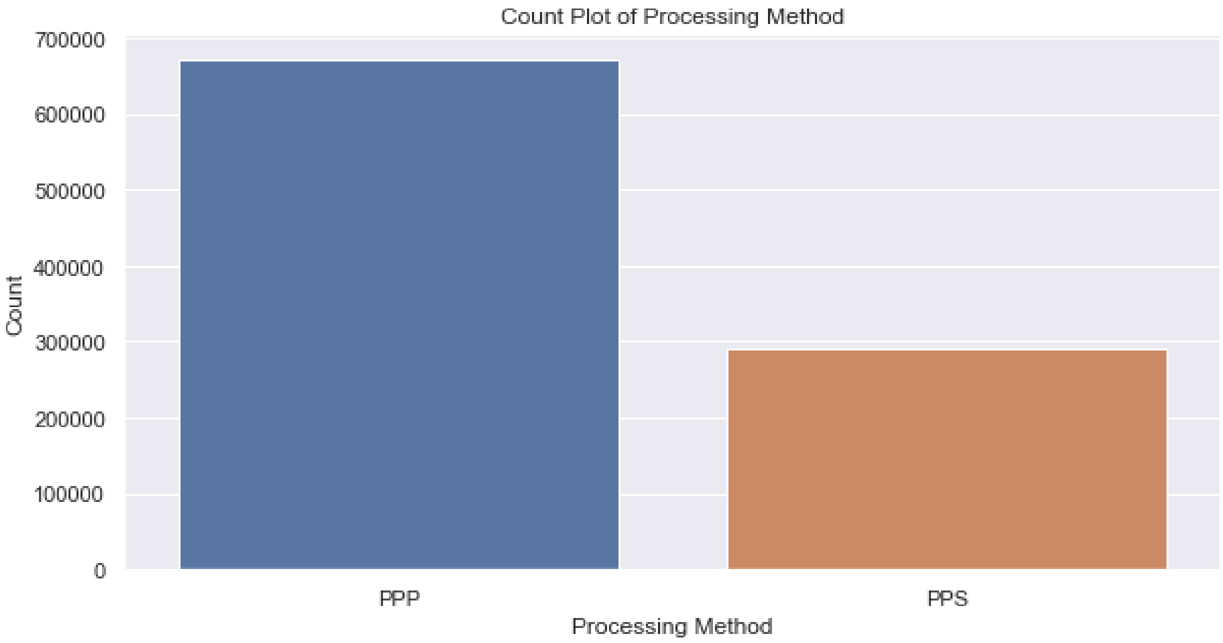
Key observations: High correlation between Payroll proceeds and Initial/Current Loan amounts. This suggests that most applications were applying to use the proceeds to process Payroll.

Other observed correlations between LoanApproval Amounts and Forgiveness amounts are not as telling.

Comparing Processing Methods

In [27]:

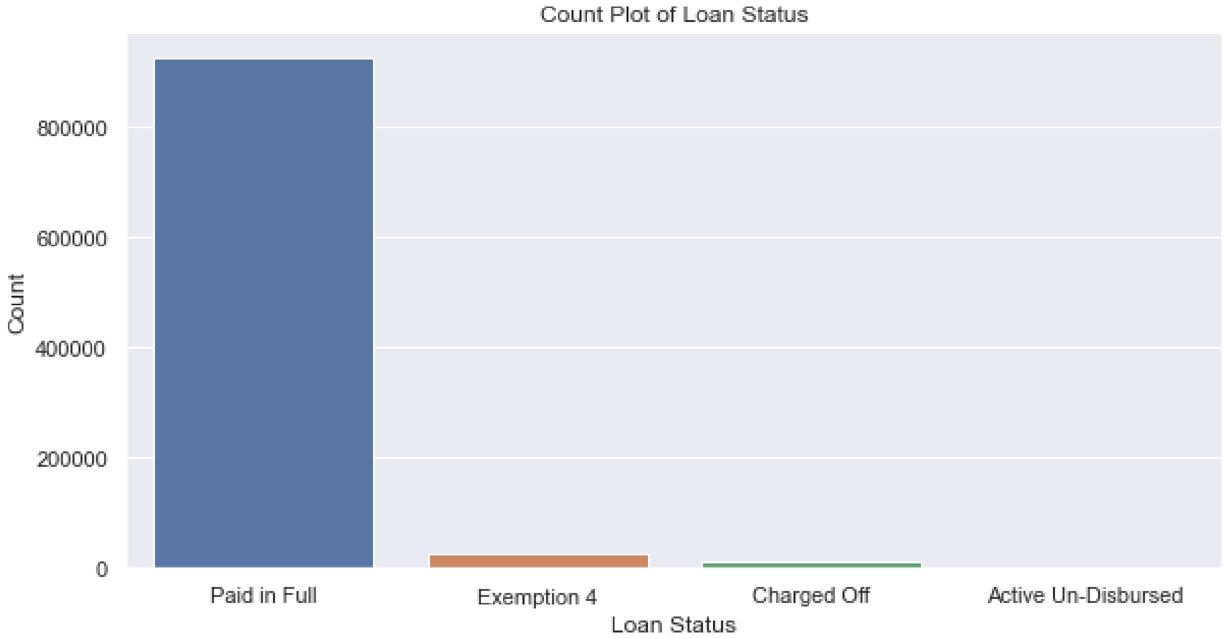
```python
# create a countplot
sns.countplot(x='ProcessingMethod', data=df)

# set the axis labels and title
plt.xlabel('Processing Method')
plt.ylabel('Count')
plt.title('Count Plot of Processing Method')

# display the plot
plt.show()
```



Visualizing the counts for Loan Status

In [28]:

```python
# create a countplot
sns.countplot(x='LoanStatus', data=df)

# set the axis labels and title
plt.xlabel('Loan Status')
plt.ylabel('Count')
plt.title('Count Plot of Loan Status')

# display the plot
plt.show()
```



Further qualitative analysis revealed that even forgiven loans and grants issued were recorded as "Paid in Full".

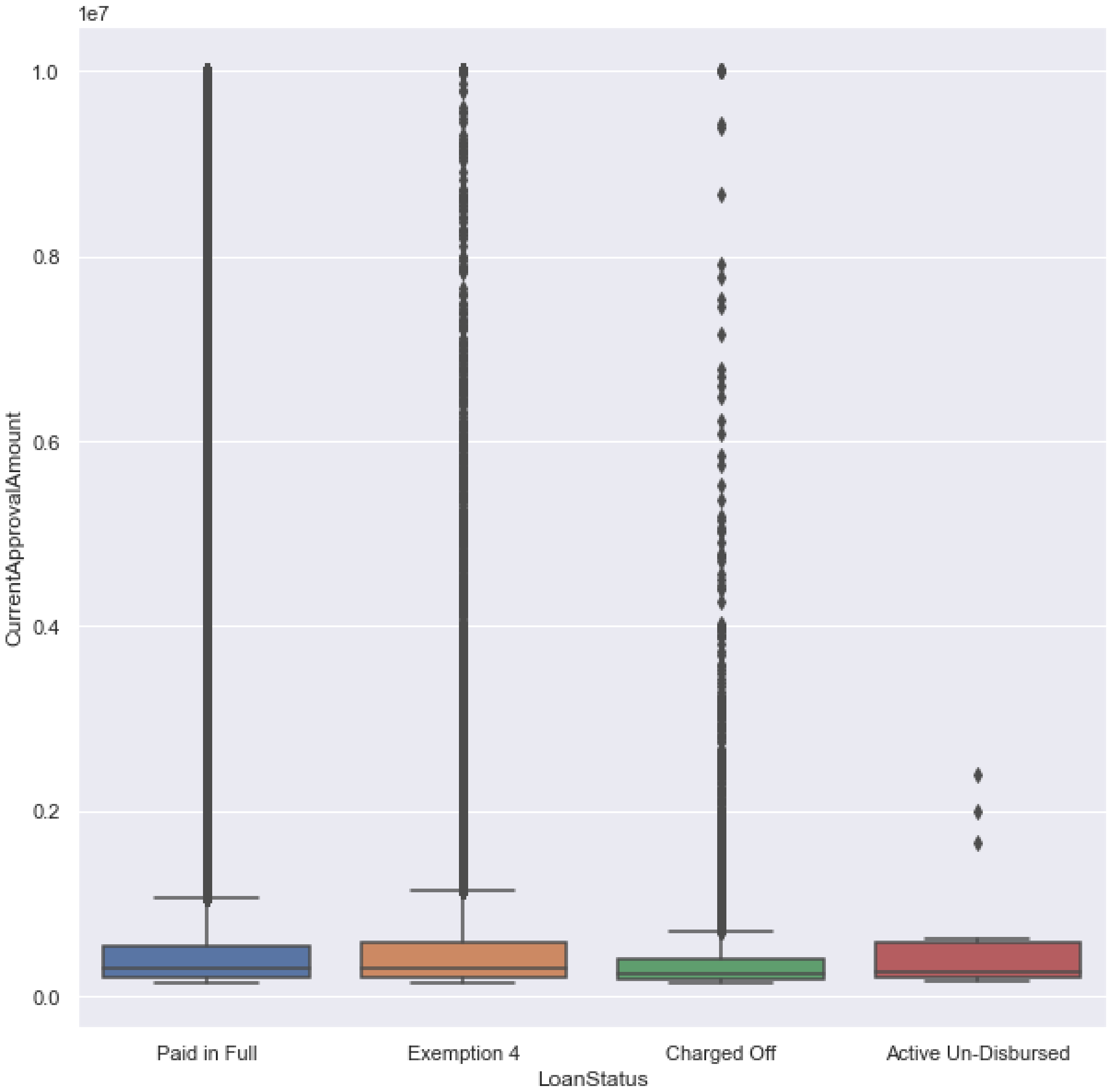
Plotting Box Outliers for Loan Statuses

In [29]:

```
plt.figure(figsize=(10,10))
sns.boxplot(x='LoanStatus', y='CurrentApprovalAmount', data=df)
```

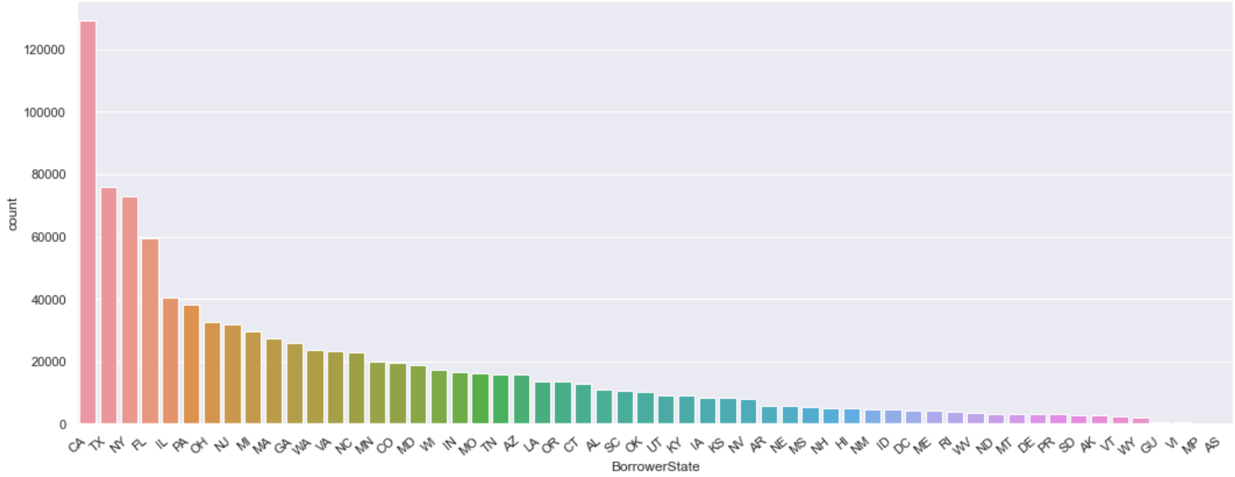
Out[29]:

```
<AxesSubplot:xlabel='LoanStatus', ylabel='CurrentApprovalAmount'>
```



Loan Applications by State

In [30]:
```
# create a countplot

plt.figure(figsize=(15,6))  #this creates an 8 inch wide, 4 inch high
ax=sns.countplot(x="BorrowerState", data=df, order=df['BorrowerState'].value_co

ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")

plt.tight_layout()
plt.show()
```



Now let us try and plot this on the map

In [31]:
```
# create a dictionary to map state abbreviations to full names
state_dict = {'AL': 'Alabama', 'AK': 'Alaska', 'AZ': 'Arizona', 'AR': 'Arkansas


# create a new column with full state names
df['state_name'] = df['BorrowerState'].map(state_dict)

# display the updated DataFrame
df.head()
```

```
/var/folders/f1/8rl13vpj76b49z9s20c53bp80000gn/T/ipykernel_13709/3189627533.p
y:6: 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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['state_name'] = df['BorrowerState'].map(state_dict)
```

Out[31]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	BorrowerAc
0	9547507704	2020-05-01	464	PPP	SUMTER COATINGS, INC.	2410 High
1	9777677704	2020-05-01	464	PPP	PLEASANT PLACES, INC.	7684 So
3	6223567700	2020-05-01	920	PPP	KIRTLEY CONSTRUCTION INC	1661 M RAN
4	9662437702	2020-05-01	101	PPP	AERO BOX LLC	
5	9774337701	2020-05-01	101	PPP	HUDSON EXTRUSIONS INC.	

5 rows × 58 columns

In [32]:

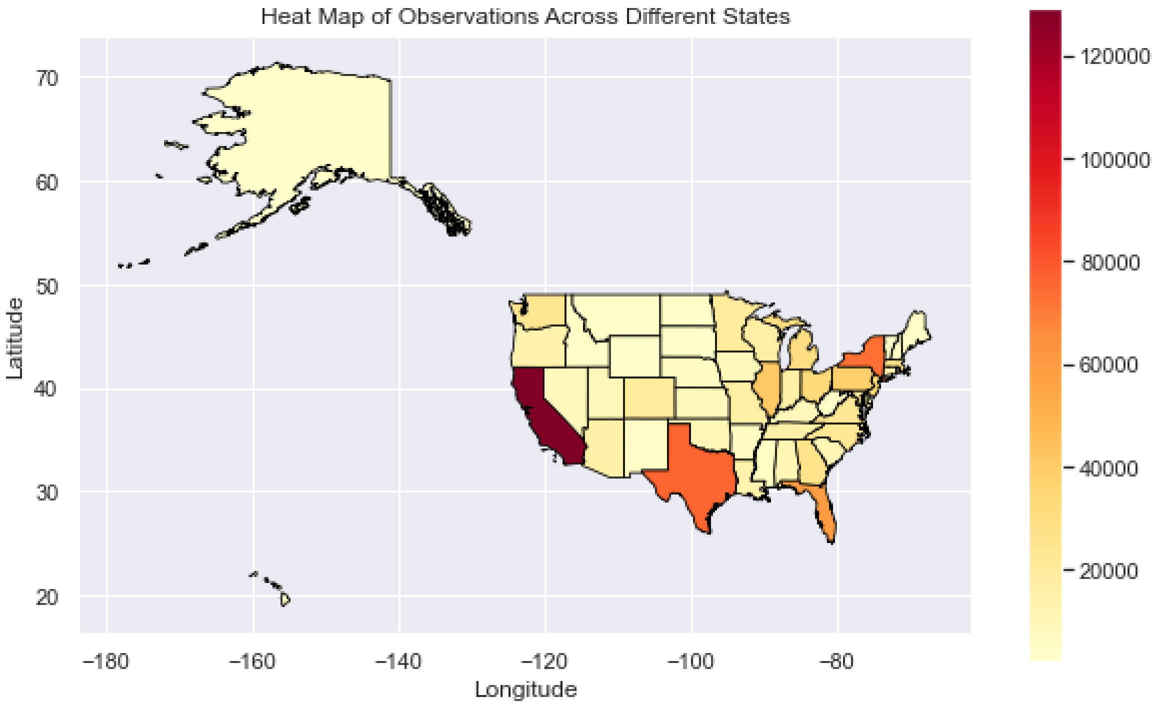
```
# count the unique values in column_1
borrowerstatecounts = df['BorrowerState'].value_counts()

# create a new DataFrame with the counts
borrowerstatecounts_df = pd.DataFrame({'unique_values': borrowerstatecounts.inc

borrowerstatecounts_df['state_name'] = borrowerstatecounts_df['unique_values'].

# display the new DataFrame
print(borrowerstatecounts_df)
```

```
   unique_values   count      state_name
0             CA  129265      California
1             TX   75729           Texas
2             NY   72941        New York
3             FL   59617         Florida
4             IL   40548        Illinois
5             PA   38296    Pennsylvania
6             OH   32434            Ohio
7             NJ   31934      New Jersey
8             MI   29608        Michigan
9             MA   27361   Massachusetts
10            GA   25726         Georgia
11            WA   23631      Washington
12            VA   23208        Virginia
13            NC   22763  North Carolina
14            MN   19696       Minnesota
15            CO   19637        Colorado
16            MD   18899        Maryland
17            WI   17313       Wisconsin
18            IN   16544         Indiana
19            MO   16303        Missouri
20            TN   15842       Tennessee
21            AZ   15668         Arizona
22            LA   13615       Louisiana
23            OR   13386          Oregon
24            CT   12818     Connecticut
25            AL   10899         Alabama
26            SC   10701  South Carolina
27            OK    9988        Oklahoma
28            UT    9233            Utah
29            KY    9144        Kentucky
30            IA    8289            Iowa
31            KS    8195          Kansas
32            NV    8108          Nevada
33            AR    5849        Arkansas
34            NE    5800        Nebraska
35            MS    5498     Mississippi
36            NH    5051   New Hampshire
37            HI    4979          Hawaii
38            NM    4477      New Mexico
39            ID    4400           Idaho
40            DC    4359             NaN
41            ME    4178           Maine
42            RI    3780    Rhode Island
43            WV    3362   West Virginia
44            ND    3219    North Dakota
45            MT    3140         Montana
46            DE    2978        Delaware
47            PR    2897             NaN
48            SD    2695    South Dakota
49            AK    2657          Alaska
50            VT    2263         Vermont
51            WY    2137         Wyoming
52            GU     443             NaN
53            VI     289             NaN
54            MP      81             NaN
55            AS      20             NaN
```

In [33]:

```
#graph showing loan applications per state across the United States
import geopandas as gpd
```

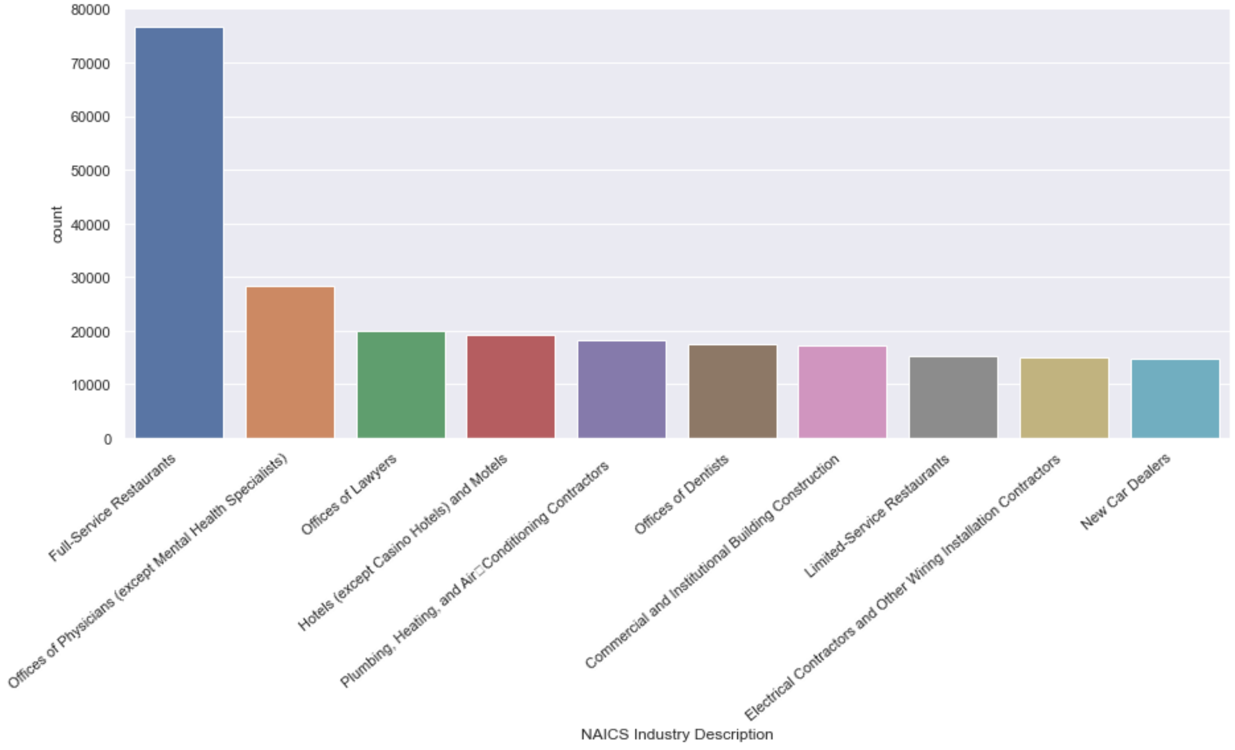
```
# load data from a csv file
#data = pd.read_csv('data.csv')

# load shapefile data of the USA
usa = gpd.read_file('geo_export_f042980c-ca77-4dd6-bff4-38d915a31cce.shp')

# merge the data and shapefile data based on the state column
merged = usa.set_index('state_name').join(borrowerstatecounts_df.set_index('sta

# create a choropleth map
fig, ax = plt.subplots(figsize=(10, 6))
merged.plot(column='count', cmap='YlOrRd', linewidth=0.8, edgecolor='black', ax

# set the axis labels and title
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.set_title('Heat Map of Observations Across Different States')

# display the plot
plt.show()
```



Loan Applications by Type of Business

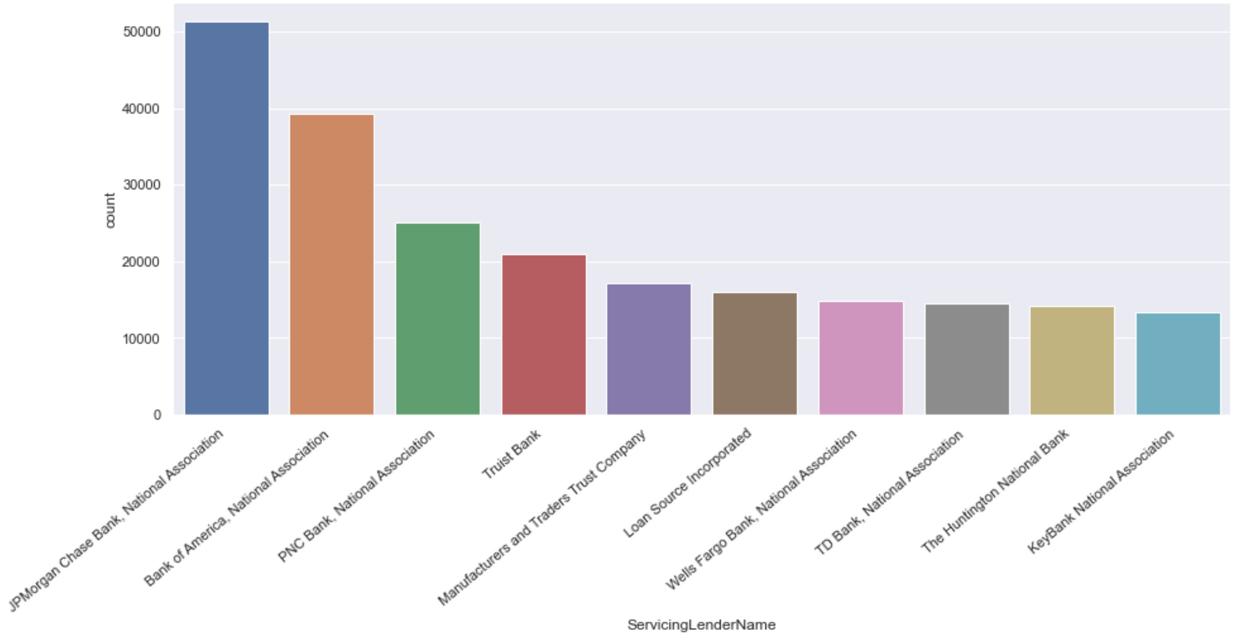
In [34]:
```
# loan application counts by industry
plt.figure(figsize=(15,6))  #this creates an 8 inch wide, 4 inch high

ax=sns.countplot(x="NAICS Industry Description", data=df,
                 order=df['NAICS Industry Description'].value_counts().iloc[:1(
ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")

plt.show()
```

```
/opt/anaconda3/lib/python3.9/site-packages/IPython/core/pylabtools.py:151: Use
rWarning: Glyph 8209 (\N{NON-BREAKING HYPHEN}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



Loan Applications by Lender

In [35]:
```
plt.figure(figsize=(15,6))

ax=sns.countplot(x="ServicingLenderName", data=df,
                 order=df['ServicingLenderName'].value_counts().iloc[:10].index
ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")

plt.show()
```



Calculating Application Risk Scores

Calculating Application Risk Scores for each loan application by comparing its key attributes (Jobs Reported, Loan Amounts, and Loan Amounts per Employee) to other businesses of the same size in the same industry to rank extremely atypical loan applications.

Calculating Deviant Jobs Reported to see which applications are reporting jobs atypical of their industry type and firm size

In [36]:

```
df['Deviant Jobs Reported'] = np.abs(df['JobsReported'] - df['Size standards ir

# calculate the percentile rank of each observation in the 'Deviant Jobs Report
df['Deviant JR Risk Score'] = df['Deviant Jobs Reported'].rank(pct=True, methoc

df.head()
```

```
/var/folders/f1/8rl13vpj76b49z9s20c53bp80000gn/T/ipykernel_13709/3475568661.p
y: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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Deviant Jobs Reported'] = np.abs(df['JobsReported'] - df['Size standards
in number of employees']) / df['Size standards in number of employees']
/var/folders/f1/8rl13vpj76b49z9s20c53bp80000gn/T/ipykernel_13709/3475568661.p
y:4: 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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Deviant JR Risk Score'] = df['Deviant Jobs Reported'].rank(pct=True, met
hod='min')
```

Out[36]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	BorrowerAc
0	9547507704	2020-05-01	464	PPP	SUMTER COATINGS, INC.	2410 High
1	9777677704	2020-05-01	464	PPP	PLEASANT PLACES, INC.	7684 So
3	6223567700	2020-05-01	920	PPP	KIRTLEY CONSTRUCTION INC	1661 M RAN
4	9662437702	2020-05-01	101	PPP	AERO BOX LLC	
5	9774337701	2020-05-01	101	PPP	HUDSON EXTRUSIONS INC.	

5 rows × 60 columns

Calculating Deviant Loan Amounts to see which applications are requesting loans atypical of their industry type and firm size

In [37]:

```
# group by NAICSCode and calculate mean of CurrentApprovalAmount
grouped_mean = df.groupby('NAICSCode')['CurrentApprovalAmount'].transform('mear

df['Deviant_CurrentApprovalAmount'] = np.abs(df['CurrentApprovalAmount'] - grou

# calculate the percentile rank of each observation in the 'Deviant_CurrentAppr
df['Deviant CAA Risk Score'] = df['Deviant_CurrentApprovalAmount'].rank(pct=Tru

df.head()
```

```
/var/folders/f1/8rl13vpj76b49z9s20c53bp80000gn/T/ipykernel_13709/1525821375.p
y:4: 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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Deviant_CurrentApprovalAmount'] = np.abs(df['CurrentApprovalAmount'] - g
rouped_mean) / grouped_mean
/var/folders/f1/8rl13vpj76b49z9s20c53bp80000gn/T/ipykernel_13709/1525821375.p
y:7: 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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Deviant CAA Risk Score'] = df['Deviant_CurrentApprovalAmount'].rank(pct=
True, method='min')
```

Out[37]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	BorrowerAc
0	9547507704	2020-05-01	464	PPP	SUMTER COATINGS, INC.	2410 High
1	9777677704	2020-05-01	464	PPP	PLEASANT PLACES, INC.	7684 So
3	6223567700	2020-05-01	920	PPP	KIRTLEY CONSTRUCTION INC	1661 M RAN
4	9662437702	2020-05-01	101	PPP	AERO BOX LLC	
5	9774337701	2020-05-01	101	PPP	HUDSON EXTRUSIONS INC.	

5 rows × 62 columns

Calculating Deviant Loan Amount Per Employee to see which applications are showing Per-employee Loan Amounts atypical of their industry type and firm size

In [38]:

```
# group by NAICSCode and calculate mean of loan_amount_per_employee
grouped_mean_loan_amount_per_employee = df.groupby('NAICSCode')['loan_amount_pe

df['Deviant_loan_amount_per_employee'] = np.abs(df['loan_amount_per_employee']

# calculate the percentile rank of each observation in the 'Deviant_loan_amount
```

```
df['Deviant LAPE Risk Score'] = df['Deviant_loan_amount_per_employee'].rank(pct

df.head()
```

```
/var/folders/f1/8rl13vpj76b49z9s20c53bp80000gn/T/ipykernel_13709/507173473.py:
4: 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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Deviant_loan_amount_per_employee'] = np.abs(df['loan_amount_per_employ
e'] - grouped_mean_loan_amount_per_employee) / grouped_mean_loan_amount_per_em
ployee
/var/folders/f1/8rl13vpj76b49z9s20c53bp80000gn/T/ipykernel_13709/507173473.py:
7: 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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Deviant LAPE Risk Score'] = df['Deviant_loan_amount_per_employee'].rank
(pct=True, method='min')
```

Out[38]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	BorrowerAc
0	9547507704	2020-05-01	464	PPP	SUMTER COATINGS, INC.	2410 High
1	9777677704	2020-05-01	464	PPP	PLEASANT PLACES, INC.	7684 So
3	6223567700	2020-05-01	920	PPP	KIRTLEY CONSTRUCTION INC	1661 M RAN
4	9662437702	2020-05-01	101	PPP	AERO BOX LLC	
5	9774337701	2020-05-01	101	PPP	HUDSON EXTRUSIONS INC.	

5 rows × 64 columns

Calculating Total Average Risk Scores as a combination of all Risk Scores to retrieve overall loan applications atypical of their industry type and firm size

In [39]:

```
#Average Risk Score
```

```
df['Total Average Risk Score'] = df[['Deviant JR Risk Score', 'Deviant CAA Risk

df.head()
```

```
/var/folders/f1/8rl13vpj76b49z9s20c53bp80000gn/T/ipykernel_13709/2194753620.p
y:3: 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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Total Average Risk Score'] = df[['Deviant JR Risk Score', 'Deviant CAA R
isk Score', 'Deviant LAPE Risk Score']].mean(axis=1)
```

Out[39]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	BorrowerAc
0	9547507704	2020-05-01	464	PPP	SUMTER COATINGS, INC.	2410 High
1	9777677704	2020-05-01	464	PPP	PLEASANT PLACES, INC.	7684 So
3	6223567700	2020-05-01	920	PPP	KIRTLEY CONSTRUCTION INC	1661 M RAN
4	9662437702	2020-05-01	101	PPP	AERO BOX LLC	
5	9774337701	2020-05-01	101	PPP	HUDSON EXTRUSIONS INC.	

5 rows × 65 columns

Possible Frauds based on Exporatory Analysis

In [40]:

```
# check null values for each column
null_counts = df.isnull().sum()

# Print the results
print('Null value counts by column:')
print(null_counts)
```

```
Null value counts by column:
LoanNumber                          0
DateApproved                        0
SBAOfficeCode                       0
ProcessingMethod                    0
BorrowerName                        4
                                 ...
Deviant_CurrentApprovalAmount       0
Deviant CAA Risk Score              0
Deviant_loan_amount_per_employee    17589
Deviant LAPE Risk Score             17589
Total Average Risk Score            0
Length: 65, dtype: int64
```

In [41]:

```
# Drop rows with null values in columns
df = df.dropna(subset=['BorrowerName','BorrowerAddress','BorrowerState','Borrow

# Print the result
df
```

Out[41]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	Bor
13	5502308207	2020-08-08	1084	PPP	KAKIVIK ASSET MANAGEMENT, LLC	5
14	6110847106	2020-04-14	1084	PPP	ARCTIC SLOPE NATIVE ASSOCIATION, LTD.	
15	4539098204	2020-08-06	1084	PPP	CORVUS AIRLINES INC	
16	5120868804	2021-04-17	1084	PPP	HOPE COMMUNITY RESOURCES INC.	540
17	6650277102	2020-04-14	1084	PPP	SOUTH PENINSULA HOSPITAL INC	43
...	...	...	...	...	...	
968526	4395967002	2020-04-03	897	PPP	ROY E PAULSON, JR., P.C.	10
968527	6985647108	2020-04-14	897	PPP	SWEETWATER COUNTY CHILD DEVELOPMENTAL CENTER, ...	1
968528	7996438405	2021-02-12	897	PPS	ELECTRICAL SYSTEMS OF WYOMING INC	
968529	9054647103	2020-04-15	897	PPP	EDEN LIFE CARE	Str
968530	9184687004	2020-04-09	897	PPP	S & S JOHNSON ENTERPRISES INC	

961882 rows × 65 columns

In [42]:

```
# check null values for each column
null_counts = df.isnull().sum()

# Print the results
print('Null value counts by column:')
print(null_counts)
```

```
Null value counts by column:
LoanNumber                           0
DateApproved                         0
SBAOfficeCode                        0
ProcessingMethod                     0
BorrowerName                         0
                                  ...
Deviant_CurrentApprovalAmount        0
Deviant CAA Risk Score               0
Deviant_loan_amount_per_employee    17589
Deviant LAPE Risk Score             17589
Total Average Risk Score             0
Length: 65, dtype: int64
```

PPP loan eligibility criteria

```
First draw PPP loans

Your business was operational before February 15, 2020(startup
done)
You have no more than 500 employees(done)
took loan from different lenders
Took loan within between those loans is every near

Second draw PPP loans
You have used up your first PPP loan
Your business was operational before February 15, 2020
You have no more than 300 employees (done)
If your business has multiple locations, you have no more than
300 employees per location
Check for spike in number of employees

Not eligible for PPP loans due to size, type of business, or
other criteria, applied for and received loans
```

In [43]:

```python
df['BusinessAgeDescription'].unique()
```

Out[43]:

```
array(['Existing or more than 2 years old', 'Unanswered',
       'New Business or 2 years or less', 'Change of Ownership',
       'Startup, Loan Funds will Open Business', nan], dtype=object)
```

In [44]:

```python
# business is fraud id loan used to open new business
def Is_Fraud_business_start_date(BusinessAgeDescription):
    if BusinessAgeDescription == 'Startup, Loan Funds will Open Business':
        return 1
    else:
        return 0
df['Is_Fraud_business_start_date'] = df['BusinessAgeDescription'].apply(Is_Frau

# Print the updated DataFrame
df
```

```
/var/folders/f1/8rl13vpj76b49z9s20c53bp80000gn/T/ipykernel_13709/3465551521.p
y:7: 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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Is_Fraud_business_start_date'] = df['BusinessAgeDescription'].apply(Is_F
raud_business_start_date)
```

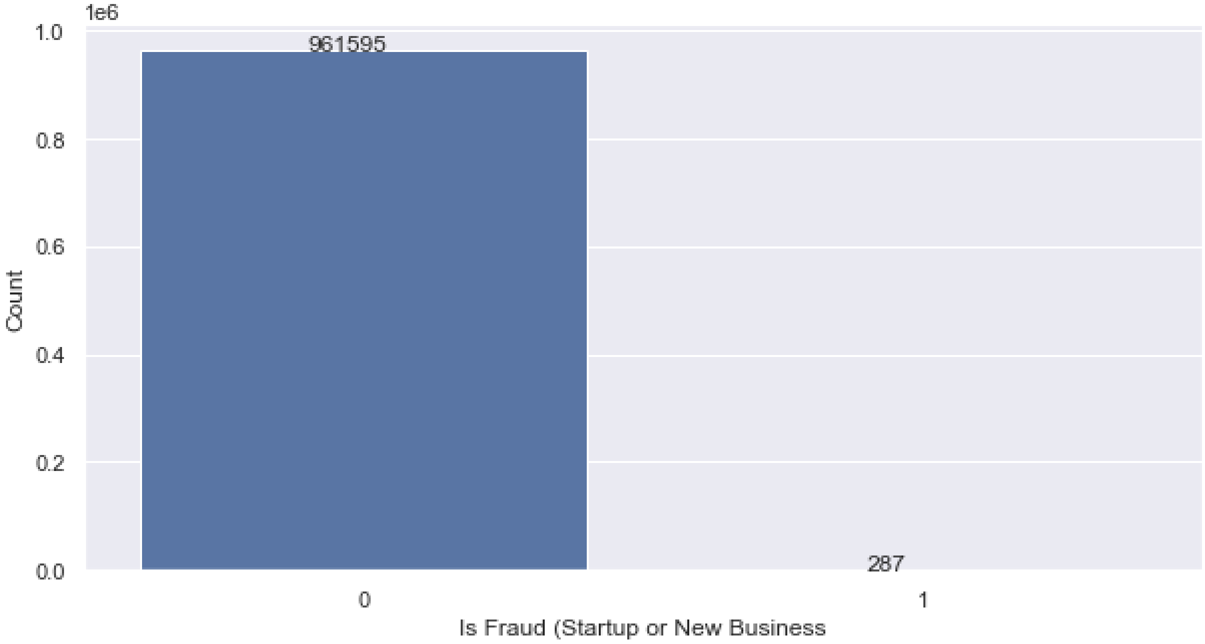
Out[44]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	Bor
13	5502308207	2020-08-08	1084	PPP	KAKIVIK ASSET MANAGEMENT, LLC	5
14	6110847106	2020-04-14	1084	PPP	ARCTIC SLOPE NATIVE ASSOCIATION, LTD.	
15	4539098204	2020-08-06	1084	PPP	CORVUS AIRLINES INC	
16	5120868804	2021-04-17	1084	PPP	HOPE COMMUNITY RESOURCES INC.	54(
17	6650277102	2020-04-14	1084	PPP	SOUTH PENINSULA HOSPITAL INC	4:
...	...	...	...	...	...	
968526	4395967002	2020-04-03	897	PPP	ROY E PAULSON, JR., P.C.	1(
968527	6985647108	2020-04-14	897	PPP	SWEETWATER COUNTY CHILD DEVELOPMENTAL CENTER, ...	'
968528	7996438405	2021-02-12	897	PPS	ELECTRICAL SYSTEMS OF WYOMING INC	
968529	9054647103	2020-04-15	897	PPP	EDEN LIFE CARE	Str
968530	9184687004	2020-04-09	897	PPP	S & S JOHNSON ENTERPRISES INC	

961882 rows × 66 columns

In [45]:

```python
# Make a count plot to show the number of True and False values in the 'is_frau
ax = sns.countplot(data=df, x='Is_Fraud_business_start_date')

# Label the axes
ax.set_xlabel('Is Fraud (Startup or New Business')
ax.set_ylabel('Count')

# Add count labels
for p in ax.patches:
    ax.annotate(p.get_height(), (p.get_x() + 0.3, p.get_height() + 0.5))

# Show the plot
plt.show()
```



In [46]:

```
# Use boolean indexing to filter the rows where 'is_fraud_business_description
filtered_df_startup = df[df['Is_Fraud_business_start_date'] == 1]

# Print the filtered DataFrame
filtered_df_startup
```

Out[46]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	Borro
13977	1580718209	2020-07-30	669	PPP	CLAY COUNTY ELECTRIC COOPERATIVE COOPERATION	3111
22677	3614747209	2020-04-27	988	PPP	SUNLAND SPRINGS MEMORY CARE LLC	24
28070	8295587704	2020-05-01	988	PPP	SVAZ LLC	
30361	2634907702	2020-05-01	988	PPP	FIREPIT HOLDINGS CORP DBA SERVPRO OF GI LBERT ...	45 N
37722	4841347103	2020-04-13	912	PPP	ROOSTIFY INC.	18 Suite
...	...	...	...	...	...	
957934	8663017005	2020-04-08	563	PPP	MAD CITY POWER SPORTS, INC.	4246
958581	9290727007	2020-04-09	563	PPP	PORTAGE COLD STORAGE, INC.	110
958931	1040697110	2020-04-09	563	PPP	CRR FRANCHISING INC	17
958932	1281487107	2020-04-10	563	PPP	UNIFIED COLD STORAGE LLC	4211
964477	6916557003	2020-04-07	390	PPP	ILA PROPERTIES INC	41

287 rows × 66 columns

In [47]:

```python
# Check for number of jobs reported based on 500 employees for First Round PPP
max_income = df['JobsReported'].max()
min_income = df['JobsReported'].min()

# Print the results
print(f"The maximum JobsReported is {max_income}.")
print(f"The minimum JobsReported is {min_income}.")

zero_JobsReported_df = df[df['JobsReported'] == 0]

# Print the filtered DataFrame
zero_JobsReported_df
```

```
The maximum JobsReported is 500.0.
The minimum JobsReported is 0.0.
```

Out[47]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	Bo
123809	6557167306	2020-04-30	914	PPP	ST. MARGARET MARY SCHOOL	25
550930	5780187005	2020-04-06	766	PPP	RELIANT TRANSPORTATION, INC.	
763779	4864227203	2020-04-27	303	PPP	VINCERA REHAB LLC	12
967205	4563247009	2020-04-03	897	PPP	WEEDEN CONSTRUCTION LLC	

4 rows × 66 columns

In [48]:

```python
# business is fraud id if jobs reoprted as 0, hence updated fraud columns for t
df.loc[df['JobsReported'] == 0, 'Is_Fraud_JobsReported'] = 1
df
```

Out[48]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	Bor
13	5502308207	2020-08-08	1084	PPP	KAKIVIK ASSET MANAGEMENT, LLC	5
14	6110847106	2020-04-14	1084	PPP	ARCTIC SLOPE NATIVE ASSOCIATION, LTD.	
15	4539098204	2020-08-06	1084	PPP	CORVUS AIRLINES INC	
16	5120868804	2021-04-17	1084	PPP	HOPE COMMUNITY RESOURCES INC.	54(
17	6650277102	2020-04-14	1084	PPP	SOUTH PENINSULA HOSPITAL INC	43
...	...	...	...	...	...	
968526	4395967002	2020-04-03	897	PPP	ROY E PAULSON, JR., P.C.	1(
968527	6985647108	2020-04-14	897	PPP	SWEETWATER COUNTY CHILD DEVELOPMENTAL CENTER, ...	1
968528	7996438405	2021-02-12	897	PPS	ELECTRICAL SYSTEMS OF WYOMING INC	
968529	9054647103	2020-04-15	897	PPP	EDEN LIFE CARE	Str
968530	9184687004	2020-04-09	897	PPP	S & S JOHNSON ENTERPRISES INC	

961882 rows × 67 columns

In [49]:

```
#mark rows as fraud based on jobs reported, if jobs> 300 in PPS, its a fraud

# Select the rows where loan processing method is 'PPS' and jobs reported is gı
df_filtered2=df.loc[(df['ProcessingMethod'] == 'PPS') & (df['JobsReported'] > 3
df_filtered2
```

Out[49]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	Borro
19943	3363518505	2021-02-23	988	PPS	KIND HOSPITALITY INC.	1225
19955	4715638505	2021-02-26	988	PPS	SOUTHWEST PIZZA INC.	5925
20874	2677328908	2021-04-27	988	PPS	YAWBUS INC	
24010	1344808907	2021-04-24	988	PPS	ARIZONA SUBWAY DEVELOPMENT CORP	
24118	4024268709	2021-03-31	988	PPS	FIRST CUP PARTNERS LAS VEGAS LLC	106
...	...	...	...	...	...	
922837	9323328608	2021-03-25	1013	PPS	GRAND CENTRAL BAKERY INC	21
946294	1252898610	2021-03-13	563	PPS	DECADE PROPERTIES INC	1355
946301	1865908505	2021-02-19	563	PPS	THE LOWLANDS GROUP LLC	142
946386	9534238605	2021-03-26	563	PPS	NORTH CENTRAL STAFFING INC.	1600
966431	5785798507	2021-03-01	897	PPS	NORTHERN ARAPAHO ENTERPRISE 2	180

284 rows × 67 columns

In [50]:

```
# business is fraud id if jobs reoprted are greater than 300 in PPS round, henc

df.loc[(df['ProcessingMethod'] == 'PPS') & (df['JobsReported'] > 300), 'Is_Frau
df.loc[(df['ProcessingMethod'] == 'PPS') & (df['JobsReported'] <= 300), 'Is_Fra
df.loc[df['LoanNumber'] == 9323328608]
df['Is_Fraud_JobsReported'] = df['Is_Fraud_JobsReported'].fillna(0).replace([np
```

In [51]:

```
df
```

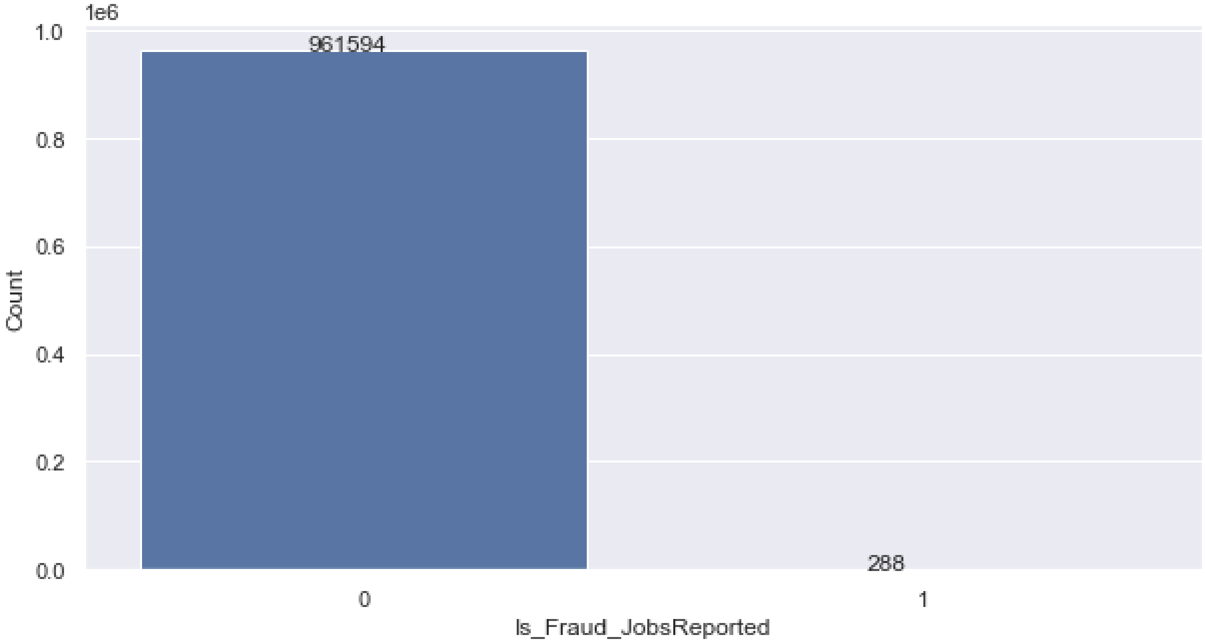
Out[51]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	Bor
13	5502308207	2020-08-08	1084	PPP	KAKIVIK ASSET MANAGEMENT, LLC	5
14	6110847106	2020-04-14	1084	PPP	ARCTIC SLOPE NATIVE ASSOCIATION, LTD.	
15	4539098204	2020-08-06	1084	PPP	CORVUS AIRLINES INC	
16	5120868804	2021-04-17	1084	PPP	HOPE COMMUNITY RESOURCES INC.	54(
17	6650277102	2020-04-14	1084	PPP	SOUTH PENINSULA HOSPITAL INC	43
...	...	...	...	...	...	
968526	4395967002	2020-04-03	897	PPP	ROY E PAULSON, JR., P.C.	1(
968527	6985647108	2020-04-14	897	PPP	SWEETWATER COUNTY CHILD DEVELOPMENTAL CENTER, ...	1
968528	7996438405	2021-02-12	897	PPS	ELECTRICAL SYSTEMS OF WYOMING INC	
968529	9054647103	2020-04-15	897	PPP	EDEN LIFE CARE	Str
968530	9184687004	2020-04-09	897	PPP	S & S JOHNSON ENTERPRISES INC	

961882 rows × 67 columns

In [52]:

```python
# Make a count plot to show the number of True and False values in the 'Is_Frau
ax = sns.countplot(data=df, x='Is_Fraud_JobsReported')

# Label the axes
ax.set_xlabel('Is_Fraud_JobsReported')
ax.set_ylabel('Count')

# Add count labels
for p in ax.patches:
    ax.annotate(p.get_height(), (p.get_x() + 0.3, p.get_height() + 0.5))

# Show the plot
plt.show()
```



In [53]:

```
# Find duplicate values based on business name ,processing method and locations
duplicates = df[df.duplicated(['BorrowerName','BorrowerAddress','BorrowerState'
if not duplicates.empty:
    print('Duplicate values found:')
    #print(duplicates)
else:
    print('No duplicate values found.')
```

```
Duplicate values found:
```

In [54]:

```
duplicates
```

Out[54]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	Borro
4667	7959257001	2020-04-08	459	PPP	CREEK INDIAN ENTERPRISES DEVELOPMENT AUTHORITY	100 B
8600	5710377106	2020-04-13	459	PPP	CREEK INDIAN ENTERPRISES DEVELOPMENT AUTHORITY	100 B
9721	4470468405	2021-02-06	459	PPS	KEEP INFORMATION TECHNOLOGY SIMPLE LLC	190
10382	6359988910	2021-05-01	459	PPS	KEEP INFORMATION TECHNOLOGY SIMPLE LLC	190
19635	9773187201	2020-04-28	988	PPP	FITNESS ALLIANCE, LLC	1 E
...	...	...	...	...	...	
937766	1002737206	2020-04-15	1013	PPP	WILDFIN NORTHWEST, LLC	835
951900	5267107209	2020-04-27	563	PPP	EXQUISITE THREADING, LLC	2727
952372	2214287710	2020-05-01	563	PPP	EXQUISITE THREADING, LLC	2727
955569	5028227006	2020-04-04	563	PPP	LARSON OAKWOOD BUSINESS PARK LLC	35
962384	4985867007	2020-04-04	563	PPP	LARSON OAKWOOD BUSINESS PARK LLC	35

164 rows × 67 columns

In [55]:

```
#Fraud loan example with loans taken for same business twice in First round PPP

df[df['BorrowerName'] == 'LARSON OAKWOOD BUSINESS PARK LLC']['JobsReported']
```

Out[55]:

```
955569    23.0
962384    14.0
Name: JobsReported, dtype: float64
```

In [56]:

```
#Fraud loan examples with loans taken by a business within same week from diffe
```

```
duplicates.loc[df['BorrowerName'] == 'EXQUISITE THREADING, LLC']
```

Out[56]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	Borro
951900	5267107209	2020-04-27	563	PPP	EXQUISITE THREADING, LLC	2727 I
952372	2214287710	2020-05-01	563	PPP	EXQUISITE THREADING, LLC	2727 I

2 rows × 67 columns

In [57]:
```
df[df['BorrowerName'] == 'EXQUISITE THREADING, LLC']['OriginatingLender']
```

Out[57]:
```
951900                          The Bippus State Bank
952372    JPMorgan Chase Bank, National Association
Name: OriginatingLender, dtype: object
```

In [58]:
```
df[df['BorrowerName'] == 'EXQUISITE THREADING, LLC']['NAICSCode']
```

Out[58]:
```
951900    812112
952372    812113
Name: NAICSCode, dtype: object
```

In [59]:
```
#identifying duplicated businesses and flagging them as Is_Fraud_duplicated
df['is_Fraud_duplicate'] = ((df.duplicated(subset=['BorrowerName', 'BorrowerAdd
                                           keep=False))).astype(int)
```

In [60]:
```
df
```

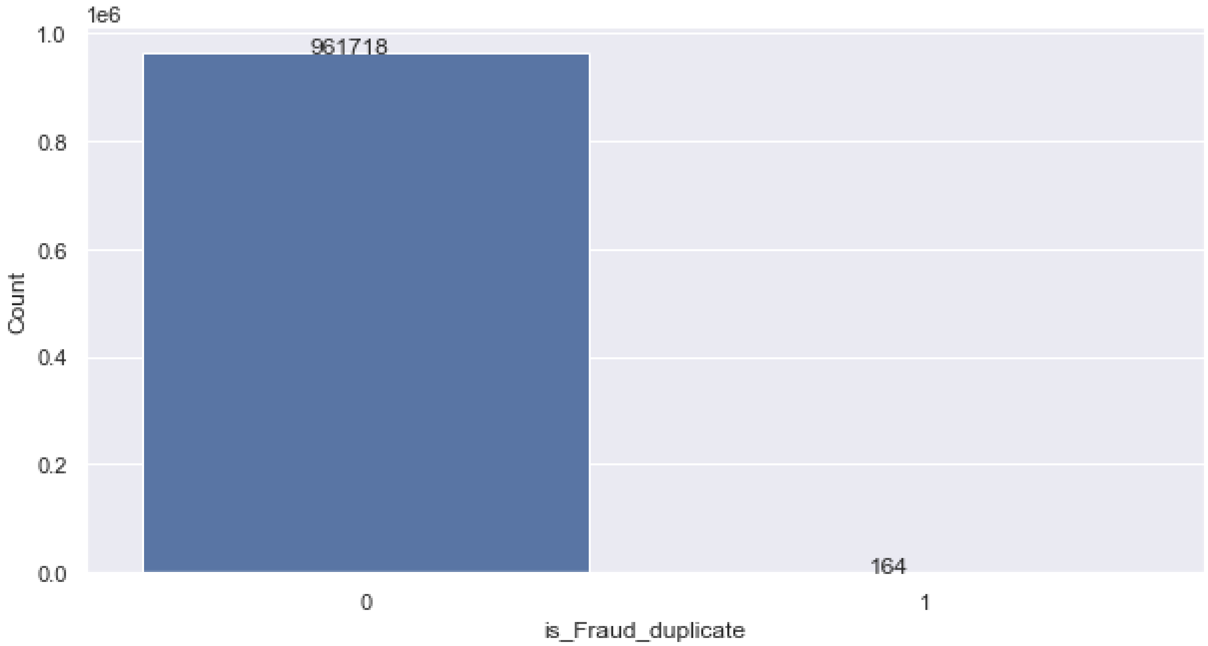
Out[60]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	Bor
13	5502308207	2020-08-08	1084	PPP	KAKIVIK ASSET MANAGEMENT, LLC	5
14	6110847106	2020-04-14	1084	PPP	ARCTIC SLOPE NATIVE ASSOCIATION, LTD.	
15	4539098204	2020-08-06	1084	PPP	CORVUS AIRLINES INC	
16	5120868804	2021-04-17	1084	PPP	HOPE COMMUNITY RESOURCES INC.	54
17	6650277102	2020-04-14	1084	PPP	SOUTH PENINSULA HOSPITAL INC	4
...	...	...	...	...	...	
968526	4395967002	2020-04-03	897	PPP	ROY E PAULSON, JR., P.C.	1
968527	6985647108	2020-04-14	897	PPP	SWEETWATER COUNTY CHILD DEVELOPMENTAL CENTER, ...	1
968528	7996438405	2021-02-12	897	PPS	ELECTRICAL SYSTEMS OF WYOMING INC	
968529	9054647103	2020-04-15	897	PPP	EDEN LIFE CARE	Str
968530	9184687004	2020-04-09	897	PPP	S & S JOHNSON ENTERPRISES INC	

961882 rows × 68 columns

In [61]:

```python
# Make a count plot to show the number of True and False values in the 'is_Frau
ax = sns.countplot(data=df, x='is_Fraud_duplicate')

# Label the axes
ax.set_xlabel('is_Fraud_duplicate')
ax.set_ylabel('Count')

# Add count labels
for p in ax.patches:
    ax.annotate(p.get_height(), (p.get_x() + 0.3, p.get_height() + 0.5))

# Show the plot
plt.show()
```



In [62]:

```
#introducing threshold for risk score in top 10 percentifile as probable fraud
threshold_risk = df['Total Average Risk Score'].quantile(0.90)

# create a new column and set the value to "fraud" if the condition is true
df['Is_Fraud_risk_avg'] = [1 if x > threshold_risk else 0 for x in df['Total Av
df
```

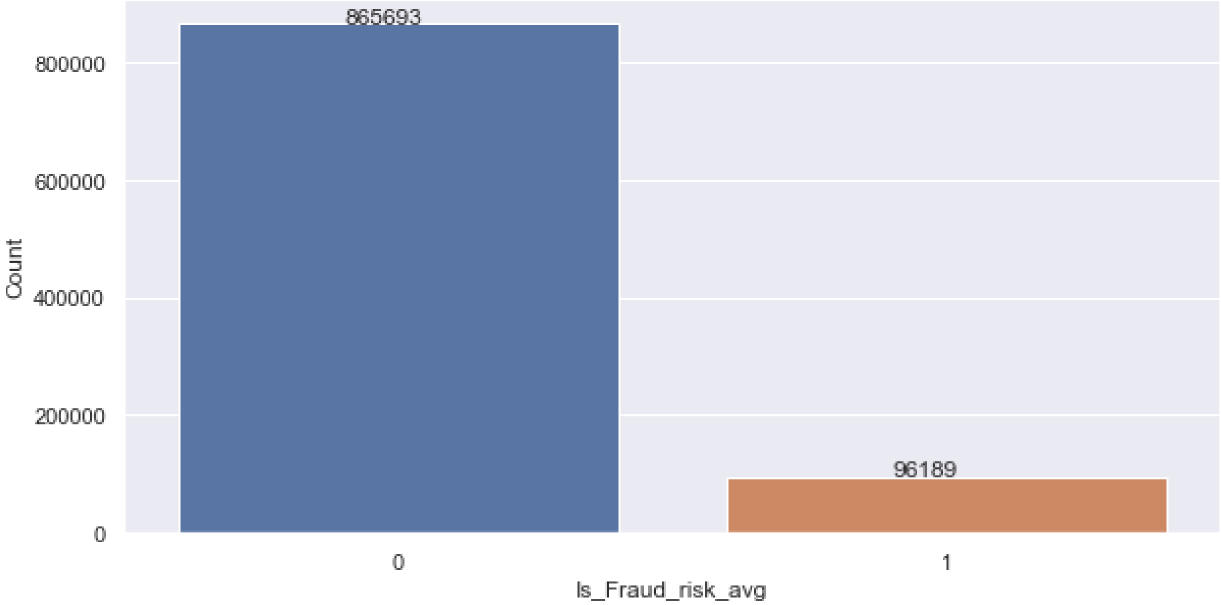
Out[62]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	Bor
13	5502308207	2020-08-08	1084	PPP	KAKIVIK ASSET MANAGEMENT, LLC	5
14	6110847106	2020-04-14	1084	PPP	ARCTIC SLOPE NATIVE ASSOCIATION, LTD.	
15	4539098204	2020-08-06	1084	PPP	CORVUS AIRLINES INC	
16	5120868804	2021-04-17	1084	PPP	HOPE COMMUNITY RESOURCES INC.	54
17	6650277102	2020-04-14	1084	PPP	SOUTH PENINSULA HOSPITAL INC	43
...	...	...	...	...	...	
968526	4395967002	2020-04-03	897	PPP	ROY E PAULSON, JR., P.C.	1
968527	6985647108	2020-04-14	897	PPP	SWEETWATER COUNTY CHILD DEVELOPMENTAL CENTER, ...	
968528	7996438405	2021-02-12	897	PPS	ELECTRICAL SYSTEMS OF WYOMING INC	
968529	9054647103	2020-04-15	897	PPP	EDEN LIFE CARE	Str
968530	9184687004	2020-04-09	897	PPP	S & S JOHNSON ENTERPRISES INC	

961882 rows × 69 columns

In [63]:

```
# Make a count plot to show the frauds based on average risk score
ax = sns.countplot(data=df, x='Is_Fraud_risk_avg')

# Label the axes
ax.set_xlabel('Is_Fraud_risk_avg')
ax.set_ylabel('Count')

# Add count labels
for p in ax.patches:
    ax.annotate(p.get_height(), (p.get_x() + 0.3, p.get_height() + 0.5))

# Show the plot
plt.show()
```



In [64]:
```
#Accumulate all fraud cases together
Is_Fraud = df[['Is_Fraud_risk_avg', 'Is_Fraud_business_start_date', 'Is_Fraud_J

# introduce new column 'Is_Fraud' with value 1 if any of the four columns have
df['Is_Fraud'] = Is_Fraud.astype(int)
df
```

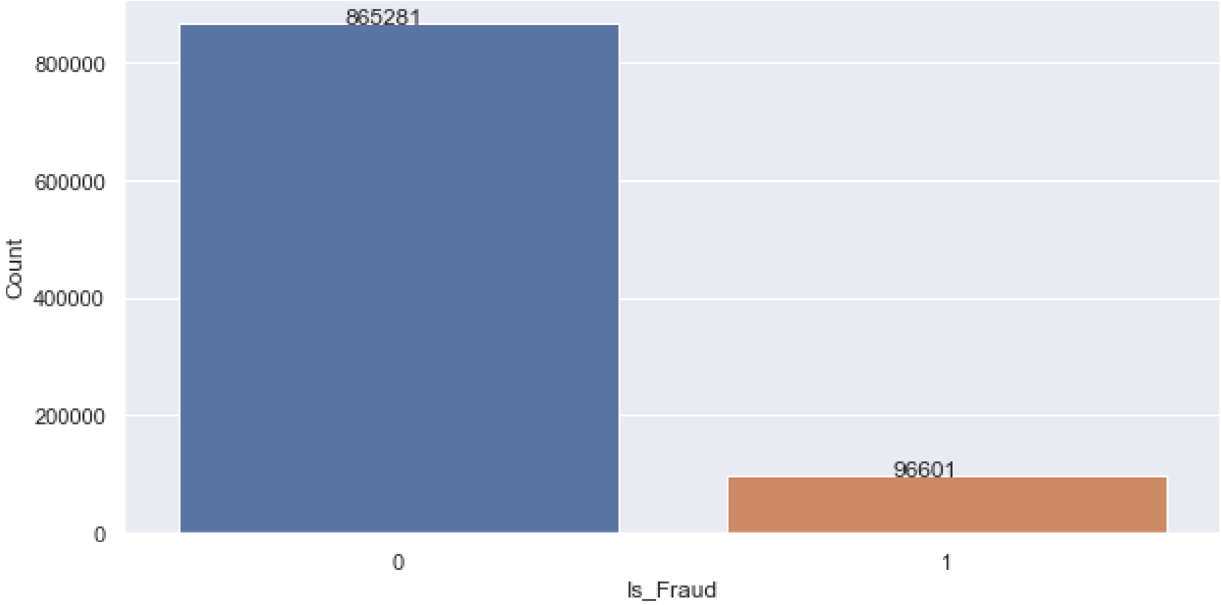
Out[64]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	Bor
13	5502308207	2020-08-08	1084	PPP	KAKIVIK ASSET MANAGEMENT, LLC	5
14	6110847106	2020-04-14	1084	PPP	ARCTIC SLOPE NATIVE ASSOCIATION, LTD.	
15	4539098204	2020-08-06	1084	PPP	CORVUS AIRLINES INC	
16	5120868804	2021-04-17	1084	PPP	HOPE COMMUNITY RESOURCES INC.	54
17	6650277102	2020-04-14	1084	PPP	SOUTH PENINSULA HOSPITAL INC	43
...	...	...	...	...	...	
968526	4395967002	2020-04-03	897	PPP	ROY E PAULSON, JR., P.C.	1
968527	6985647108	2020-04-14	897	PPP	SWEETWATER COUNTY CHILD DEVELOPMENTAL CENTER, ...	1
968528	7996438405	2021-02-12	897	PPS	ELECTRICAL SYSTEMS OF WYOMING INC	
968529	9054647103	2020-04-15	897	PPP	EDEN LIFE CARE	Str
968530	9184687004	2020-04-09	897	PPP	S & S JOHNSON ENTERPRISES INC	

961882 rows × 70 columns

In [65]:

```python
# Make a count plot to show the frauds identified in the entire dataset
ax = sns.countplot(data=df, x='Is_Fraud')

# Label the axes
ax.set_xlabel('Is_Fraud')
ax.set_ylabel('Count')

# Add count labels
for p in ax.patches:
    ax.annotate(p.get_height(), (p.get_x() + 0.3, p.get_height() + 0.5))

# Show the plot
plt.show()
```



In [66]:

```
# create a dictionary to map state abbreviations to full names
state_dict = {'AL': 'Alabama', 'AK': 'Alaska', 'AZ': 'Arizona', 'AR': 'Arkansas


# create a new column with full state names
df['state_name'] = df['BorrowerState'].map(state_dict)
```

In [67]:

```
#frauds density state wise

fraud_count = df.groupby('BorrowerState')['Is_Fraud'].sum().reset_index()
fraud_count['BorrowerState'] = fraud_count['BorrowerState'].astype(str)
import plotly.express as px

fig = px.choropleth(fraud_count, locations='BorrowerState',
                    locationmode="USA-states", color='Is_Fraud', scope="usa")
fig.update_layout(title='State-wise Fraud Counts in the USA')
fig.show()
```

Anomaly Detection using Isolation Forest

In [68]:

```
num_missing = df[['PAYROLL_PROCEED', 'JobsReported','CurrentApprovalAmount', 'B
                     'BorrowerAddress', 'BorrowerCity', 'BorrowerState', 'Origina

print(f"Total number of rows with missing values: {num_missing}")

#df[['PAYROLL_PROCEED', 'JobsReported','InitialApprovalAmount']]
```

```
Total number of rows with missing values: PAYROLL_PROCEED          1820
JobsReported                1
CurrentApprovalAmount       0
BorrowerName                0
BorrowerAddress             0
BorrowerCity                0
BorrowerState               0
OriginatingLender           0
NAICSCode                   0
dtype: int64
```

In [69]:

```
df = df.dropna(subset=['PAYROLL_PROCEED', 'JobsReported'])

num_missing = df[['PAYROLL_PROCEED', 'JobsReported','CurrentApprovalAmount', 'B
                     'BorrowerAddress', 'BorrowerCity', 'BorrowerState', 'Origina
```

```
print(f"Total number of rows with missing values: {num_missing}")
```

```
Total number of rows with missing values: PAYROLL_PROCEED          0
JobsReported            0
CurrentApprovalAmount   0
BorrowerName            0
BorrowerAddress         0
BorrowerCity            0
BorrowerState           0
OriginatingLender       0
NAICSCode               0
dtype: int64
```

In [70]:

```
LAPE_nan_count = df['loan_amount_per_employee'].isna().sum()
print(LAPE_nan_count)
```

```
0
```

In [71]:

```
df['loan_amount_per_employee'] = df['loan_amount_per_employee'].replace([np.inf
```

```
/var/folders/f1/8rl13vpj76b49z9s20c53bp80000gn/T/ipykernel_13709/2005413637.p
y: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/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [72]:

```
LAPE_nan_count = df['loan_amount_per_employee'].isna().sum()
print(LAPE_nan_count)
```

```
4
```

In [73]:

```
df = df.dropna(subset=['loan_amount_per_employee'])
LAPE_nan_count = df['loan_amount_per_employee'].isna().sum()
print(LAPE_nan_count)
```

```
0
```

In [74]:

```
TARS_nan_count = df['Total Average Risk Score'].isna().sum()
print(TARS_nan_count)
```

```
0
```

In [75]:

```
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import StandardScaler

# Scale the features to have zero mean and unit variance
X = df[['PAYROLL_PROCEED', 'JobsReported', 'InitialApprovalAmount', 'loan_amour
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Initialize the Isolation Forest model
clf = IsolationForest(n_estimators=100, max_samples='auto', contamination=0.01,

# Fit the model to the data
```

```
clf.fit(X_scaled)

# Predict the anomaly scores for each data point
scores = clf.decision_function(X_scaled)

# Add the anomaly scores as a new column in your dataframe
df['anomaly_score'] = scores

# Sort the dataframe by anomaly score in descending order
df = df.sort_values(by='anomaly_score', ascending=False)

# Print the rows with the highest anomaly scores, which could potentially indic
df.head(10)
```

Out[75]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	Borro
858306	7496047207	2020-04-28	610	PPP	GENESIS RESOURCES, LLC	
581363	9655287108	2020-04-15	299	PPP	KAM MAN SUPERMARKET LLC	
779878	1577258408	2021-02-02	165	PPS	CHURCHILL & BANKS COMPANIES LLC	'
858108	8753077408	2020-05-19	671	PPP	COASTWIDE MARINE SERVICES, LLC	163
779879	5855487007	2020-04-06	165	PPP	CHURCHILL & BANKS COMPANIES LLC	10
804514	6250327203	2020-04-27	474	PPP	FIRST CHOICE SERVICES INC.	4135
177855	9934747107	2020-04-15	811	PPP	JACK'S BEAN COMPANY, LLC	INTEF
700781	8234128306	2021-01-29	549	PPS	MCMAHON MASONRY RESTORATION LTD MCMAHON MASONR...	944
508836	5878107001	2020-04-06	768	PPP	I & I, INC.	5105
804445	1554467108	2020-04-10	474	PPP	EMB QUALITY MASONRY	2 Ro

10 rows × 71 columns

In [76]:

```
# Get split values of the individual trees
tree_split_values = np.zeros((X.shape[1], clf.n_estimators))
for i, tree in enumerate(clf.estimators_):
    for j in range(X.shape[1]):
        tree_split_values[j, i] = tree.tree_.threshold[j]

# Calculate feature importance based on split values
feat_importance = np.mean(tree_split_values, axis=1)

# Print feature importances in descending order
for i in np.argsort(feat_importance)[::-1]:
    print(f"{X.columns[i]}: {feat_importance[i]}")
```

```
PAYROLL_PROCEED: 2.651886161705919
JobsReported: 1.0553154151301845
InitialApprovalAmount: 0.301919596242056
loan_amount_per_employee: -0.0261636964201355
Total Average Risk Score: -0.13883244503281827
```

In [77]:

```
# calculate the 95th percentile value
threshold = df['anomaly_score'].quantile(0.90)

# create a new column and set the value to "fraud" if the condition is true
df['Is_Fraud_Anomaly'] = [1 if x > threshold else 0 for x in df['anomaly_score'
df
```

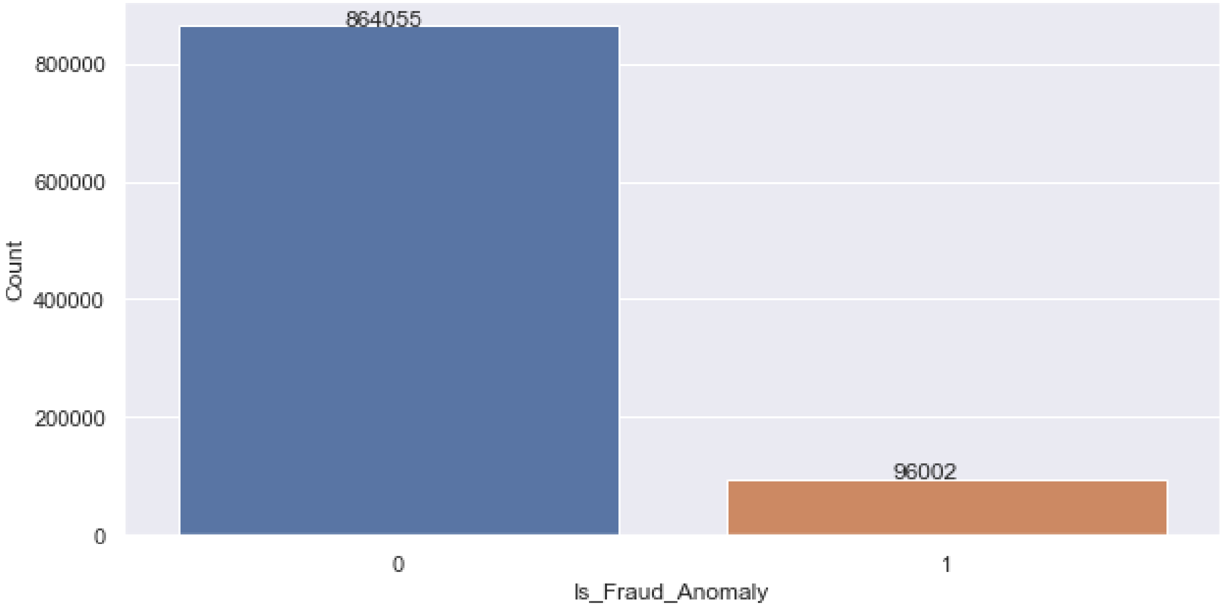
Out[77]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	Borro
858306	7496047207	2020-04-28	610	PPP	GENESIS RESOURCES, LLC	
581363	9655287108	2020-04-15	299	PPP	KAM MAN SUPERMARKET LLC	
779878	1577258408	2021-02-02	165	PPS	CHURCHILL & BANKS COMPANIES LLC	'
858108	8753077408	2020-05-19	671	PPP	COASTWIDE MARINE SERVICES, LLC	163
779879	5855487007	2020-04-06	165	PPP	CHURCHILL & BANKS COMPANIES LLC	10
...	...	...	...	...	...	
810634	4688397001	2020-04-04	678	PPP	VENABLE'S WELDING & ROUSTABOUT	CC
560561	8015747003	2020-04-08	299	PPP	APPLE FOOD SERVICE OF NEW JERSEY LLC	
2675	1517597200	2020-04-15	459	PPP	ACTION ENTERPRISE HOLDINGS LLC	20
605898	7569987307	2020-04-30	202	PPP	COUNTY AGENCY INC.	
605942	3319697106	2020-04-11	202	PPP	SH GROUP, INC.	118 1

960057 rows × 72 columns

In [78]:

```
#Countplot of Frauds vs Non-Frauds based on isolation forest
ax = sns.countplot(data=df, x='Is_Fraud_Anomaly')

# Label the axes
ax.set_xlabel('Is_Fraud_Anomaly')
ax.set_ylabel('Count')

# Add count labels
for p in ax.patches:
    ax.annotate(p.get_height(), (p.get_x() + 0.3, p.get_height() + 0.5))

# Show the plot
plt.show()
```



```
In [79]: fraud_count = df.groupby('BorrowerState').agg({'Is_Fraud_Anomaly': 'sum', 'Loar
         fraud_count['state_name'] = fraud_count['BorrowerState'].map(state_dict)
         fraud_count
```

Out[79]:

	BorrowerState	Is_Fraud_Anomaly	LoanNumber	state_name
0	AK	208	2650	Alaska
1	AL	1331	10860	Alabama
2	AR	650	5842	Arkansas
3	AS	0	20	NaN
4	AZ	1627	15638	Arizona
5	CA	12036	129043	California
6	CO	2003	19592	Colorado
7	CT	1218	12785	Connecticut
8	DC	290	4350	NaN
9	DE	293	2975	Delaware
10	FL	6487	59502	Florida
11	GA	2702	25684	Georgia
12	GU	22	443	NaN
13	HI	552	4977	Hawaii
14	IA	935	8265	Iowa
15	ID	503	4393	Idaho
16	IL	3424	40487	Illinois
17	IN	1872	16501	Indiana
18	KS	827	8171	Kansas
19	KY	1074	9130	Kentucky
20	LA	1550	13595	Louisiana
21	MA	2368	27304	Massachusetts
22	MD	1800	18869	Maryland
23	ME	535	4167	Maine
24	MI	3053	29509	Michigan
25	MN	1747	19663	Minnesota
26	MO	1595	16263	Missouri
27	MP	2	81	NaN
28	MS	604	5481	Mississippi
29	MT	382	3127	Montana
30	NC	2623	22723	North Carolina
31	ND	330	3213	North Dakota
32	NE	687	5775	Nebraska
33	NH	520	5039	New Hampshire
34	NJ	2885	31865	New Jersey

	BorrowerState	Is_Fraud_Anomaly	LoanNumber	state_name
35	NM	501	4470	New Mexico
36	NV	853	8098	Nevada
37	NY	6399	72843	New York
38	OH	3365	32358	Ohio
39	OK	1135	9949	Oklahoma
40	OR	1372	13367	Oregon
41	PA	3842	38214	Pennsylvania
42	PR	156	2896	NaN
43	RI	352	3774	Rhode Island
44	SC	1259	10689	South Carolina
45	SD	327	2684	South Dakota
46	TN	1776	15820	Tennessee
47	TX	7632	75583	Texas
48	UT	1030	9222	Utah
49	VA	2330	23166	Virginia
50	VI	44	289	NaN
51	VT	285	2255	Vermont
52	WA	2260	23610	Washington
53	WI	1716	17295	Wisconsin
54	WV	377	3360	West Virginia
55	WY	256	2133	Wyoming

In [80]:

```
#plot fraud count per state based on anamoly detection

fraud_count_states = df.groupby('BorrowerState')['Is_Fraud_Anomaly'].sum().rese
fraud_count_states['BorrowerState'] = fraud_count_states['BorrowerState'].astyp
import plotly.express as px

fig = px.choropleth(fraud_count_states, locations='BorrowerState',
                    locationmode="USA-states", color='Is_Fraud_Anomaly', scope=
fig.update_layout(title='State-wise Fraud Counts in the USA')
fig.show()
```

In [81]:

```
# fraud_count = df.groupby('BorrowerState')['Is_Fraud_Anomaly'].sum().reset_inc
# fraud_count['BorrowerState'].dtypes
```

In [82]:

```
# sort the DataFrame in descending order by the 'Fraud' column based on isolati
fraud_count_sorted = fraud_count.sort_values('Is_Fraud_Anomaly', ascending=Fals

# create a multiple line plot using plotly express
fig = px.line(fraud_count_sorted, x='BorrowerState', y=['Is_Fraud_Anomaly', 'Lo
              title='Fraud vs. LoanNumber by State (sorted by Fraud)')

# show the plot
fig.show()
```

In [ ]:

```
```

In [83]:

```
# create a new column for the fraud-to-loan ratio
fraud_count['FraudRatio'] = fraud_count['Is_Fraud_Anomaly'] / fraud_count['Loar
fraud_count['FraudRatio'] = fraud_count['FraudRatio'].round(2)

fraud_count_sorted = fraud_count.sort_values('FraudRatio', ascending=False)

# create a bar plot using plotly express
fig = px.bar(fraud_count_sorted, x='state_name', y='FraudRatio', title='Fraud-t


# show the plot
fig.show()
```

Few other observations and conclusions based on our own Fraud analysis

In [84]:

```
#total initial approval loan amount
column_sums = df['InitialApprovalAmount'].sum()

print("InitialApprovalAmount:",column_sums)
column_sums_b=column_sums/1000000000
print("InitialApprovalAmount in billions:",column_sums_b)
```

```
InitialApprovalAmount: 511963558645.63965
InitialApprovalAmount in billions: 511.96355864563964
```

In [85]:

```
#total current approval loan amount

column_sums = df['CurrentApprovalAmount'].sum()

print("CurrentApprovalAmount:",column_sums)
column_sums_b=column_sums/1000000000
print("CurrentApprovalAmount in billions:",column_sums_b)
```

```
CurrentApprovalAmount: 510388406009.3
CurrentApprovalAmount in billions: 510.3884060093
```

In [86]:

```
#total forgiveness amount
column_sums = df['ForgivenessAmount'].sum()

print("ForgivenessAmount:",column_sums)
column_sums_b=column_sums/1000000000
print("ForgivenessAmount in billions:",column_sums_b)
```

```
ForgivenessAmount: 492922605322.24994
ForgivenessAmount in billions: 492.9226053222499
```

In [87]:

```
# fraud forgiveness amount
fraud_df = df[df['Is_Fraud'] == 1] # select only the rows where is_fraud is 1
forgiveness_sum = fraud_df['ForgivenessAmount'].sum()

print("ForgivenessAmount:",forgiveness_sum)
forgiveness_sum_b=forgiveness_sum/1000000000
print("ForgivenessAmount in Fraud data in billions:",forgiveness_sum_b)
```

```
ForgivenessAmount: 115567368358.58
ForgivenessAmount in Fraud data in billions: 115.56736835858
```

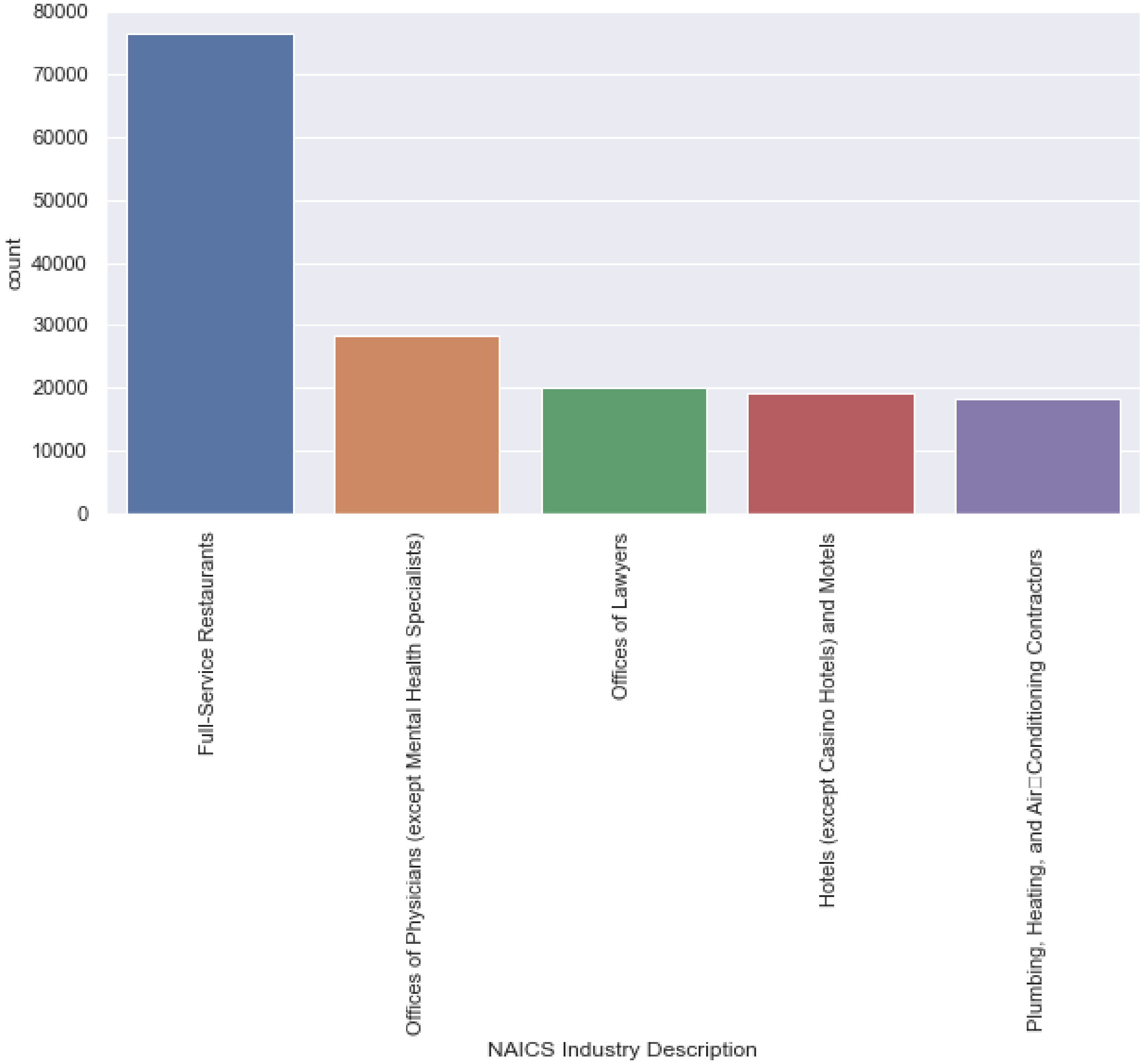
In [88]:

```
# identify top 5 lenders
counts = df['NAICS Industry Description'].value_counts()

# Select the top 5 categories
top_categories = counts[:5].index.tolist()

# Create a countplot for the top 5 categories with vertical x-axis labels
sns.countplot(x='NAICS Industry Description', data=df[df['NAICS Industry Descri
plt.xticks(rotation=90)
```

Out[88]:

```
(array([0, 1, 2, 3, 4]),
 [Text(0, 0, 'Full-Service Restaurants'),
  Text(1, 0, 'Offices of Physicians (except Mental Health Specialists)'),
  Text(2, 0, 'Offices of Lawyers'),
  Text(3, 0, 'Hotels (except Casino Hotels) and Motels'),
  Text(4, 0, 'Plumbing, Heating, and Air-Conditioning Contractors   ')])
```

```
/opt/anaconda3/lib/python3.9/site-packages/IPython/core/events.py:89: UserWarn
ing:

Glyph 8209 (\N{NON-BREAKING HYPHEN}) missing from current font.

/opt/anaconda3/lib/python3.9/site-packages/IPython/core/pylabtools.py:151: Use
rWarning:

Glyph 8209 (\N{NON-BREAKING HYPHEN}) missing from current font.
```

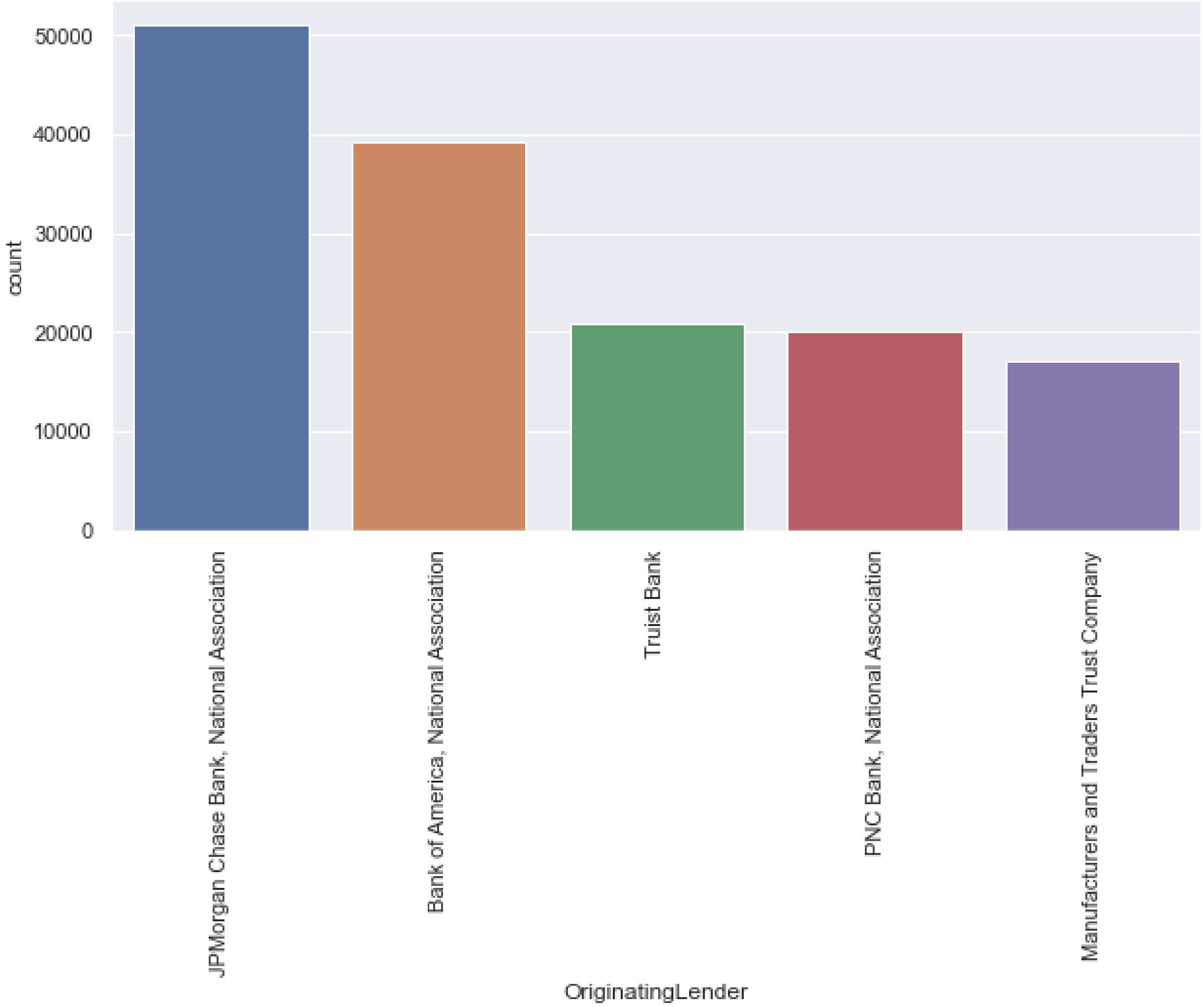


In [89]:

```
# identify top 5 lenders
counts = df['OriginatingLender'].value_counts()

# Select the top 5 categories
top_categories = counts[:5].index.tolist()

# Create a countplot for the top 5 categories with vertical x-axis labels
sns.countplot(x='OriginatingLender', data=df[df['OriginatingLender'].isin(top_c
plt.xticks(rotation=90)
```

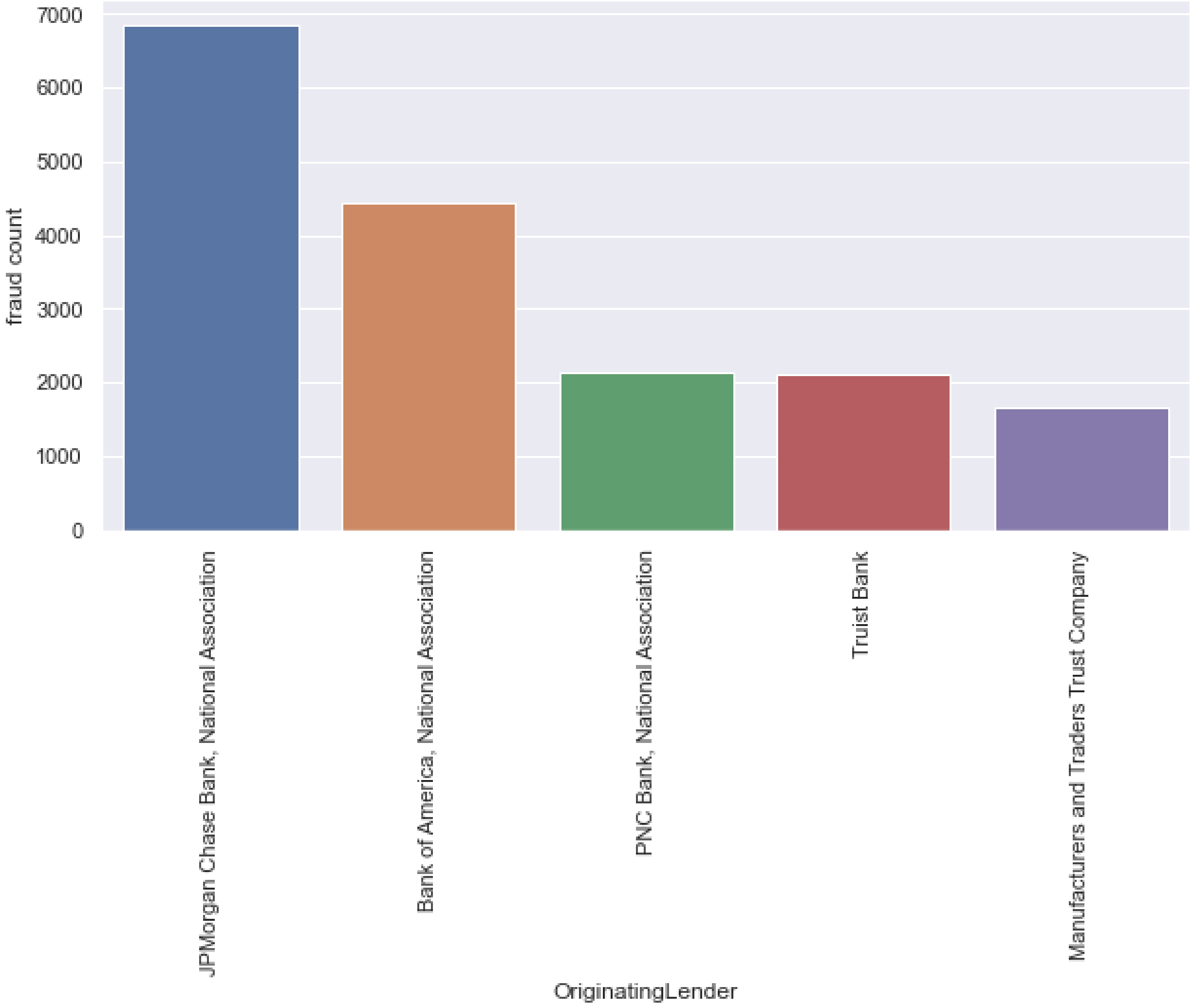
Out[89]:

```
(array([0, 1, 2, 3, 4]),
 [Text(0, 0, 'JPMorgan Chase Bank, National Association'),
  Text(1, 0, 'Bank of America, National Association'),
  Text(2, 0, 'Truist Bank'),
  Text(3, 0, 'PNC Bank, National Association'),
  Text(4, 0, 'Manufacturers and Traders Trust Company')])
```



In [90]:
```
#Identify top five lenders who did fraud

frauds_by_lender = df[df['Is_Fraud'] == 1].groupby('OriginatingLender').size().

# Sort the results by count and select the top 5 lenders
top_lenders = frauds_by_lender.sort_values(by='fraud count', ascending=False).h

# Create a barplot of the top 5 lenders with vertical x-axis labels
sns.barplot(x='OriginatingLender', y='fraud count', data=top_lenders)
plt.xticks(rotation=90)
```

Out[90]:
```
(array([0, 1, 2, 3, 4]),
 [Text(0, 0, 'JPMorgan Chase Bank, National Association'),
  Text(1, 0, 'Bank of America, National Association'),
  Text(2, 0, 'PNC Bank, National Association'),
  Text(3, 0, 'Truist Bank'),
  Text(4, 0, 'Manufacturers and Traders Trust Company')])
```

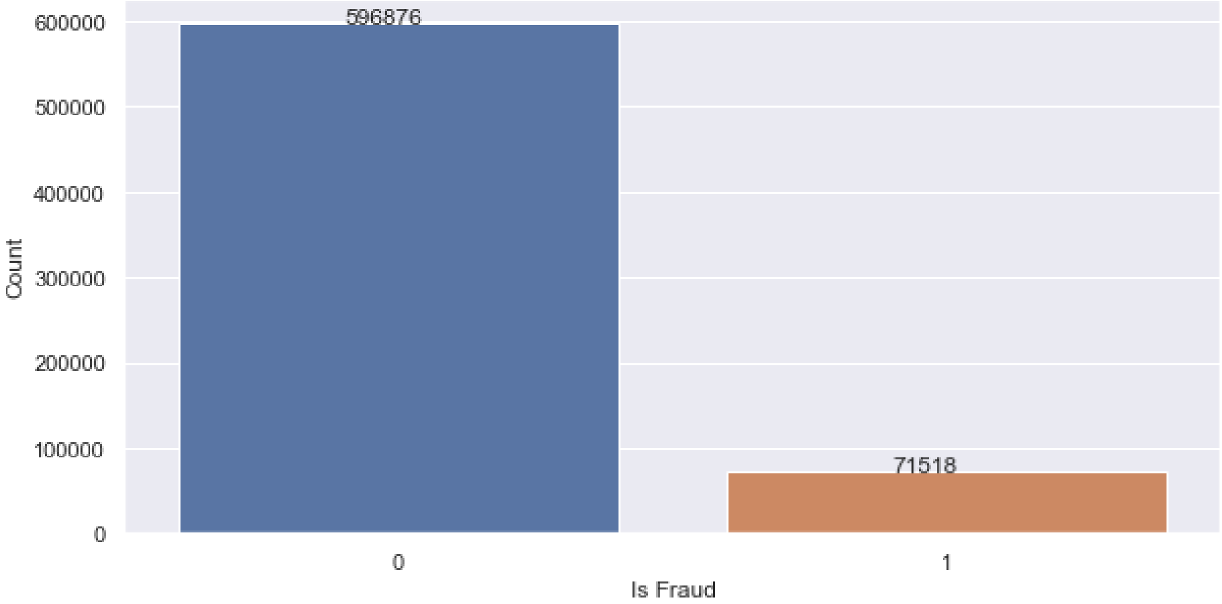


In [91]:

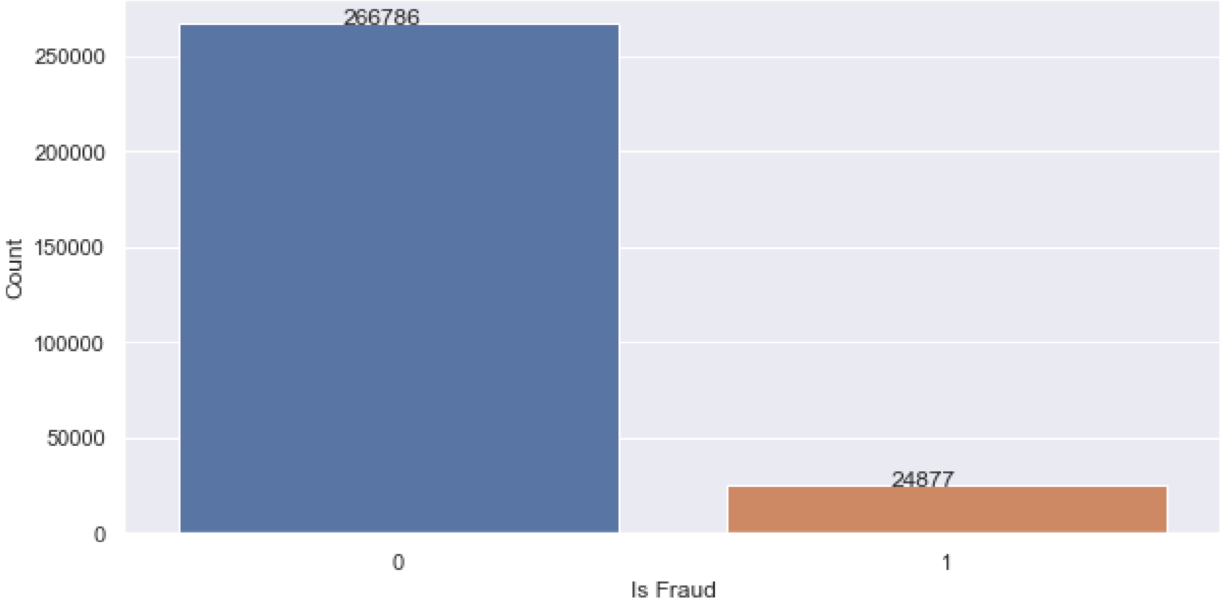
```
FirstRound=df.loc[df['ProcessingMethod']=='PPP']
SecondRound=df.loc[df['ProcessingMethod']=='PPS']
```

In [92]:

```
#frauds in First Round
ax = sns.countplot(data=FirstRound, x='Is_Fraud')

# Label the axes
ax.set_xlabel('Is Fraud')
ax.set_ylabel('Count')

# Add count labels
for p in ax.patches:
    ax.annotate(p.get_height(), (p.get_x() + 0.3, p.get_height() + 0.5))

# Show the plot
plt.show()
```



In [93]:
```
#frauds in First Round
ax = sns.countplot(data=SecondRound, x='Is_Fraud')

# Label the axes
ax.set_xlabel('Is Fraud')
ax.set_ylabel('Count')

# Add count labels
for p in ax.patches:
    ax.annotate(p.get_height(), (p.get_x() + 0.3, p.get_height() + 0.5))

# Show the plot
plt.show()
```



% of fraud loans issued both by amount

In [94]:
```
column_sums = df['CurrentApprovalAmount'].sum()
```

```
print("CurrentApprovalAmount:",column_sums)
column_sums_b=column_sums/1000000000
print("CurrentApprovalAmount in billions:",column_sums_b)

fraud_df = df[df['Is_Fraud'] == 1] # select only the rows where is_fraud is 1
fraud_CAA_sum = fraud_df['CurrentApprovalAmount'].sum()

print("Fraud loan approved amount:",fraud_CAA_sum)
fraud_CAA_b=forgiveness_sum/1000000000
print("Fraud loan approved amount: in billions:",fraud_CAA_b)

fraud_amount_percent=fraud_CAA_b/column_sums_b
print("% of fraud loans issued both by amount",fraud_amount_percent*100)
```

```
CurrentApprovalAmount: 510388406009.3
CurrentApprovalAmount in billions: 510.3884060093
Fraud loan approved amount: 121633664550.96
Fraud loan approved amount: in billions: 115.56736835858
% of fraud loans issued both by amount 22.64302382222887
```

Minimum loans having atleast one fraud flag

```
In [95]: df
```

Out[95]:

	LoanNumber	DateApproved	SBAOfficeCode	ProcessingMethod	BorrowerName	Borro
858306	7496047207	2020-04-28	610	PPP	GENESIS RESOURCES, LLC	
581363	9655287108	2020-04-15	299	PPP	KAM MAN SUPERMARKET LLC	
779878	1577258408	2021-02-02	165	PPS	CHURCHILL & BANKS COMPANIES LLC	
858108	8753077408	2020-05-19	671	PPP	COASTWIDE MARINE SERVICES, LLC	163
779879	5855487007	2020-04-06	165	PPP	CHURCHILL & BANKS COMPANIES LLC	10
...	...	...	...	...	...	
810634	4688397001	2020-04-04	678	PPP	VENABLE'S WELDING & ROUSTABOUT	CO
560561	8015747003	2020-04-08	299	PPP	APPLE FOOD SERVICE OF NEW JERSEY LLC	
2675	1517597200	2020-04-15	459	PPP	ACTION ENTERPRISE HOLDINGS LLC	20
605898	7569987307	2020-04-30	202	PPP	COUNTY AGENCY INC.	
605942	3319697106	2020-04-11	202	PPP	SH GROUP, INC.	118 1

960057 rows × 72 columns

In [96]:

```
# Assuming the dataset is stored in a DataFrame named df
mask_1 = df['Is_Fraud'] == 1
mask_2 = df['Is_Fraud_risk_avg'] == 1


count_1 = mask_1.sum()
count_2 = mask_2.sum()

min_count = min(count_1, count_2)

percentage = min_count / len(df)

print("Minimum count:", min_count)
print("Percentage of minimum fraud:", percentage)
```

```
Minimum count: 95985
Percentage of minimum fraud: 0.09997843878019742
```

More than 9% of PPP loans had at least one indication of potential fraud

```
In [ ]:
```