In [2]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import glob
import statsmodels.api as sm
```

In [3]:

```
file_list = glob.glob("C:\\Users\\Sri Manaswini\\Desktop\\DS_Capstone\\CS
data = []

column_mapping = {
 'Year' : 'Year',
 'Loan Number' : 'LoanNumber',
 'Amount' : 'NoteAmount',
 'PropertyType' : 'PropType',
 'FHLBankID' : 'Bank',
 'FHLBank' : 'Bank',
 'Co-Borrower Credit Score' : 'CoBoCreditScor',
 'CoBorrower Credit Score' : 'CoBoCreditScor',
 'CoCreditScor'  : 'CoBoCreditScor',
 'CoCreditScore' : 'CoBoCreditScor',
 'CoBoCreditScore' : 'CoBoCreditScor',
 'Co Borrower Credit Score' : 'CoBoCreditScor',
 'Borrower2CreditScoreValue' : 'CoBoCreditScor',
 'Borrower Credit Score' : 'BoCreditScore',
 'Borrower1CreditScoreValue' : 'BoCreditScore',
 'BoCreditScor' : 'BoCreditScore',
 'Assigned ID' : 'AssignedID',
 'LoanCharacteristicsID' : 'AssignedID',
 'AcquDate' : 'AcqDate',
 'LoanAcquistionDate' : 'AcqDate',
 'LoanAcquisitionDate' : 'AcqDate',
 'Borrower1EthnicityType' : 'BoEth',
 'Borrower1Race2Type' : 'Race2',
 'Borrower1Race3Type' : 'Race3',
 'Borrower1Race4Type' : 'Race4',
 'Borrower1Race5Type' : 'Race5',
 'Borrower2EthnicityType' : 'CoEth',
 'Borrower2Race2Type' : 'Corace2',
 'Borrower2Race3Type' : 'Corace3',
 'Borrower2Race4Type' : 'Corace4',
 'Borrower2Race5Type' : 'Corace5',
 'CoRace2' : 'Corace2',
 'CoRace3' : 'Corace3',
 'CoRace4' : 'Corace4',
 'CoRace5' : 'Corace5',
 'HOEPALoanStatusType' : 'HOEPA',
 'LienPriorityType' : 'LienStatus',
 'PrepaymentPenaltyExpirationDate' : 'PrepayP',
 'PMICoveragePercent' : 'PMI',
 'EmploymentBorrowerSelfEmployed' :'Self',
 'IndexSourceType' : 'ArmIndex',
 'ArmMarg' : 'MarginRatePercent',
 'FIPSStateNumericCode' : 'FIPSStateCode',
 'CoreBasedStatisticalAreaCode' : 'MSA',
 'CensusTractIdentifier' : 'Tract',
 'CensusTractMinorityRatioPercent' : 'MinPer',
 'CensusTractMedFamIncomeAmount' : 'TraMedY',
 'LocalAreaMedianIncomeAmount' : 'LocMedY',
 'TotalMonthlyIncomeAmount' : 'Income',
 'HUDMedianIncomeAmount' : 'CurAreY',
 'LoanAcquisitionActualUPBAmt' : 'UPB',
 'LTVRatioPercent' : 'LTV',
 'NoteDate' : 'MortDate',
 'LoanPurposeType' : 'Purpose',
 'ProductCategoryName' : 'Product',
 'MortgageType' : 'FedGuar',
 'ScheduledTotalPaymentCount' : 'Term',
 'LoanAmortizationMaxTermMonths' : 'AmorTerm',
 'MortgageLoanSellerInstType' : 'SellType',
 'BorrowerCount' : 'NumBor',
 'BorrowerFirstTimeHomebuyer' :'First',
 'Borrower1Race1Type' : 'BoRace',
 'Borrower2Race1Type' : 'CoRace',
 'Borrower1GenderType' : 'BoGender',
 'Borrower2GenderType' : 'CoGender',
 'Borrower1AgeAtApplicationYears' : 'BoAge',
 'Borrower2AgeAtApplicationYears' : 'CoAge',
 'PropertyUsageType' : 'Occup',
 'PropertyUnitCount' : 'NumUnits',
 'NoteRatePercent' : 'Rate',
 'HousingExpenseRatioPercent' : 'Front',
 'TotalDebtExpenseRatioPercent' : 'Back'
}


for file in file_list:
    df = pd.read_csv(file)
    df = df.rename(columns=column_mapping)
    data.append(df)

merged_data = pd.concat(data, ignore_index=True)
merged_data.to_csv("C:\\Users\\Sri Manaswini\\Desktop\\DS_Capstone\\FHLB
```

In [4]:

```
#Read in data
merged_data_1 = pd.read_csv('C:\\Users\\Sri Manaswini\\Desktop\\DS_Capsto
```

```
C:\Users\Sri Manaswini\AppData\Local\Temp\ipykernel_14916\3339429981.py:
2: DtypeWarning: Columns (2,90,91,92) have mixed types. Specify dtype op
tion on import or set low_memory=False.
  merged_data_1 = pd.read_csv('C:\\Users\\Sri Manaswini\\Desktop\\DS_Cap
stone\\FHLB Data\\merged_data1.csv')
```

In [4]:

```
merged_data_1.tail()
```

Out[4]:

```
        Year  AssignedID    Bank  FIPSStateCode  FIPSCountyCode    MSA     FeatureID
743828  2014         NaN  Topeka             46             135  49460  1.259091e+06
743829  2014         NaN  Topeka             48              77  48660  1.000000e+10
743830  2014         NaN  Topeka             56               9  99999  1.587750e+06
743831  2014         NaN  Topeka             56              21  16940  1.609077e+06
743832  2014         NaN  Topeka             56              25  16220  1.586424e+06

5 rows × 93 columns
```

In [5]:

```
merged_data_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 743833 entries, 0 to 743832
Data columns (total 93 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   Year             743833 non-null  int64  
 1   AssignedID       466925 non-null  float64
 2   Bank             693331 non-null  object 
 3   FIPSStateCode    743833 non-null  int64  
 4   FIPSCountyCode   743833 non-null  int64  
 5   MSA              743833 non-null  int64  
 6   FeatureID        457135 non-null  float64
 7   Tract            743833 non-null  float64
 8   MinPer           743833 non-null  float64
 9   TraMedY          743833 non-null  int64  
 10  LocMedY          743833 non-null  int64  
 11  Tractrat         507070 non-null  float64
 12  Income           743833 non-null  int64  
 13  CurAreY          743833 non-null  int64  
 14  IncRat           507070 non-null  float64
 15  UPB              743833 non-null  int64  
 16  LTV              743833 non-null  float64
 17  MortDate         743833 non-null  int64  
 18  AcqDate          743833 non-null  int64  
 19  Purpose          743833 non-null  int64  
 20  Coop             507070 non-null  float64
 21  Product          743833 non-null  int64  
 22  FedGuar          743833 non-null  int64  
 23  Term             743833 non-null  int64  
 24  AmorTerm         743833 non-null  int64  
 25  SellType         743833 non-null  int64  
 26  NumBor           743833 non-null  int64  
 27  First            743833 non-null  int64  
 28  CICA             507070 non-null  float64
 29  BoRace           743833 non-null  int64  
 30  CoRace           743833 non-null  int64  
 31  BoGender         743833 non-null  int64  
 32  CoGender         743833 non-null  int64  
 33  BoAge            743833 non-null  int64  
 34  CoAge            743833 non-null  int64  
 35  Occup            743833 non-null  int64  
 36  NumUnits         743833 non-null  int64  
 37  Bed1             507070 non-null  float64
 38  Bed2             507070 non-null  float64
 39  Bed3             507070 non-null  float64
 40  Bed4             507070 non-null  float64
 41  Aff1             507070 non-null  float64
 42  Aff2             507070 non-null  float64
 43  Aff3             507070 non-null  float64
 44  Aff4             507070 non-null  float64
 45  Rent1            507070 non-null  float64
 46  Rent2            507070 non-null  float64
 47  Rent3            507070 non-null  float64
 48  Rent4            507070 non-null  float64
 49  RentUt1          507070 non-null  float64
 50  RentUt2          507070 non-null  float64
 51  RentUt3          507070 non-null  float64
 52  RentUt4            507070 non-null  float64
 53  Geog               507070 non-null  float64
 54  Rate               743833 non-null  float64
 55  NoteAmount         743833 non-null  int64
 56  Front              743833 non-null  float64
 57  Back               743833 non-null  float64
 58  BoCreditScore      743833 non-null  int64
 59  CoBoCreditScor     743833 non-null  int64
 60  PMI                743833 non-null  float64
 61  Self               743833 non-null  int64
 62  PropType           743833 non-null  object
 63  ArmIndex           743833 non-null  int64
 64  MarginRatePercent  743833 non-null  int64
 65  PrepayP            743833 non-null  object
 66  BoEth              743833 non-null  int64
 67  Race2              743833 non-null  int64
 68  Race3              743833 non-null  int64
 69  Race4              743833 non-null  int64
 70  Race5              743833 non-null  int64
 71  CoEth              743833 non-null  int64
 72  Corace2            743833 non-null  int64
 73  Corace3            743833 non-null  int64
 74  Corace4            743833 non-null  int64
 75  Corace5            743833 non-null  int64
 76  HOEPA              743833 non-null  int64
 77  LienStatus         743833 non-null  int64
 78  SpcHsgGoals        507070 non-null  float64
 79  FedFinStbltyPlan   507070 non-null  float64
 80  AcqTyp             507070 non-null  float64
 81  GSEREO             507070 non-null  float64
 82  Unnamed: 82        0 non-null       float64
 83  Unnamed: 83        0 non-null       float64
 84  Unnamed: 84        0 non-null       float64
 85  Unnamed: 85        0 non-null       float64
 86  Unnamed: 86        0 non-null       float64
 87  LoanNumber         276908 non-null  float64
 88  Program            276908 non-null  float64
 89  FHFBID             276908 non-null  float64
 90  Seller             276908 non-null  object
 91  SellCity           276908 non-null  object
 92  SellSt             276908 non-null  object
dtypes: float64(42), int64(45), object(6)
memory usage: 527.8+ MB
```

In [6]:

```
merged_data_1 = merged_data_1.drop(['Unnamed: 82','Unnamed: 83','Unnamed:
```

In [7]:

```
merged_data_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 743833 entries, 0 to 743832
Data columns (total 88 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   Year             743833 non-null  int64  
 1   AssignedID       466925 non-null  float64
 2   Bank             693331 non-null  object 
 3   FIPSStateCode    743833 non-null  int64  
 4   FIPSCountyCode   743833 non-null  int64  
 5   MSA              743833 non-null  int64  
 6   FeatureID        457135 non-null  float64
 7   Tract            743833 non-null  float64
 8   MinPer           743833 non-null  float64
 9   TraMedY          743833 non-null  int64  
 10  LocMedY          743833 non-null  int64  
 11  Tractrat         507070 non-null  float64
 12  Income           743833 non-null  int64  
 13  CurAreY          743833 non-null  int64  
 14  IncRat           507070 non-null  float64
 15  UPB              743833 non-null  int64  
 16  LTV              743833 non-null  float64
 17  MortDate         743833 non-null  int64  
 18  AcqDate          743833 non-null  int64  
 19  Purpose          743833 non-null  int64  
 20  Coop             507070 non-null  float64
 21  Product          743833 non-null  int64  
 22  FedGuar          743833 non-null  int64  
 23  Term             743833 non-null  int64  
 24  AmorTerm         743833 non-null  int64  
 25  SellType         743833 non-null  int64  
 26  NumBor           743833 non-null  int64  
 27  First            743833 non-null  int64  
 28  CICA             507070 non-null  float64
 29  BoRace           743833 non-null  int64  
 30  CoRace           743833 non-null  int64  
 31  BoGender         743833 non-null  int64  
 32  CoGender         743833 non-null  int64  
 33  BoAge            743833 non-null  int64  
 34  CoAge            743833 non-null  int64  
 35  Occup            743833 non-null  int64  
 36  NumUnits         743833 non-null  int64  
 37  Bed1             507070 non-null  float64
 38  Bed2             507070 non-null  float64
 39  Bed3             507070 non-null  float64
 40  Bed4             507070 non-null  float64
 41  Aff1             507070 non-null  float64
 42  Aff2             507070 non-null  float64
 43  Aff3             507070 non-null  float64
 44  Aff4             507070 non-null  float64
 45  Rent1            507070 non-null  float64
 46  Rent2            507070 non-null  float64
 47  Rent3            507070 non-null  float64
 48  Rent4            507070 non-null  float64
 49  RentUt1          507070 non-null  float64
 50  RentUt2          507070 non-null  float64
 51  RentUt3          507070 non-null  float64
 52  RentUt4            507070 non-null  float64
 53  Geog               507070 non-null  float64
 54  Rate               743833 non-null  float64
 55  NoteAmount         743833 non-null  int64
 56  Front              743833 non-null  float64
 57  Back               743833 non-null  float64
 58  BoCreditScore      743833 non-null  int64
 59  CoBoCreditScor     743833 non-null  int64
 60  PMI                743833 non-null  float64
 61  Self               743833 non-null  int64
 62  PropType           743833 non-null  object
 63  ArmIndex           743833 non-null  int64
 64  MarginRatePercent  743833 non-null  int64
 65  PrepayP            743833 non-null  object
 66  BoEth              743833 non-null  int64
 67  Race2              743833 non-null  int64
 68  Race3              743833 non-null  int64
 69  Race4              743833 non-null  int64
 70  Race5              743833 non-null  int64
 71  CoEth              743833 non-null  int64
 72  Corace2            743833 non-null  int64
 73  Corace3            743833 non-null  int64
 74  Corace4            743833 non-null  int64
 75  Corace5            743833 non-null  int64
 76  HOEPA              743833 non-null  int64
 77  LienStatus         743833 non-null  int64
 78  SpcHsgGoals        507070 non-null  float64
 79  FedFinStbltyPlan   507070 non-null  float64
 80  AcqTyp             507070 non-null  float64
 81  GSEREO             507070 non-null  float64
 82  LoanNumber         276908 non-null  float64
 83  Program            276908 non-null  float64
 84  FHFBID             276908 non-null  float64
 85  Seller             276908 non-null  object
 86  SellCity           276908 non-null  object
 87  SellSt             276908 non-null  object
dtypes: float64(37), int64(45), object(6)
memory usage: 499.4+ MB
```

In [8]:

```
merged_data_1[merged_data_1['AssignedID'] == 1997542]
```

Out[8]:

	Year	AssignedID	Bank	FIPSStateCode	FIPSCountyCode	MSA	FeatureID	Tract
0	2015	1997542.0	Atlanta	51	69	49020	1740338.0	511.01

1 rows × 88 columns

In [9]:

```
# counting and printing number of rows
print("Number of Rows = ", len(merged_data_1.axes[0]))

# counting and printing number of columns
print("Number of Columns = ", len(merged_data_1.axes[1]))
```

```
Number of Rows =  743833
Number of Columns =  88
```

In [10]:

```
merged_data_1.shape
```

Out[10]:

```
(743833, 88)
```

In [11]:

```
#Counting no of records present with respect to year
count_year = merged_data_1['Year'].value_counts()
print(count_year)
```

```
2019    89767
2020    83106
2012    66411
2018    65703
2021    63890
2016    60989
2017    55990
2009    50502
2015    47480
2011    43914
2010    41220
2013    40547
2014    34314
Name: Year, dtype: int64
```

In [12]:

```
merged_data
```

Out[12]:

	Year	AssignedID	Bank	FIPSStateCode	FIPSCountyCode	MSA	FeatureID
0	2015	1997542.0	Atlanta	51	69	49020	1.740338e+06
1	2015	1997543.0	Atlanta	18	39	21140	4.352270e+05
2	2015	1997544.0	Atlanta	13	245	12260	3.562620e+05
3	2015	1997545.0	Atlanta	12	9	37340	2.945890e+05
4	2015	1997546.0	Atlanta	32	3	29820	8.473880e+05
...	...	...	...	...	...	...	...
743828	2014	NaN	Topeka	46	135	49460	1.259091e+06
743829	2014	NaN	Topeka	48	77	48660	1.000000e+10
743830	2014	NaN	Topeka	56	9	99999	1.587750e+06
743831	2014	NaN	Topeka	56	21	16940	1.609077e+06
743832	2014	NaN	Topeka	56	25	16220	1.586424e+06

743833 rows × 93 columns

In [13]:

```
merged_data_1.describe()
```

Out[13]:

	Year	AssignedID	FIPSStateCode	FIPSCountyCode	MSA	
count	743833.000000	4.669250e+05	743833.000000	743833.000000	743833.000000	4.57
mean	2015.710653	2.305588e+06	28.918924	84.843860	40097.304129	2.47
std	3.785652	2.388545e+05	12.994376	68.565875	28534.893908	4.31
min	2009.000000	1.949641e+06	1.000000	1.000000	29.000000	5.16
25%	2012.000000	2.080544e+06	19.000000	37.000000	19380.000000	7.35
50%	2016.000000	2.409419e+06	27.000000	73.000000	30700.000000	1.62
75%	2019.000000	2.528635e+06	39.000000	119.000000	44100.000000	2.41
max	2021.000000	2.690373e+06	78.000000	840.000000	99999.000000	1.00

8 rows × 82 columns

In [14]:

```
merged_data_1.dtypes
```

Out[14]:

```
Year                int64
AssignedID        float64
Bank               object
FIPSStateCode       int64
FIPSCountyCode      int64
                   ...
Program           float64
FHFBID            float64
Seller             object
SellCity           object
SellSt             object
Length: 88, dtype: object
```

In [15]:

```
count = merged_data_1.isnull().sum()
print(f'Count of NULL values with respect to each column are given below
print(f'\nTotal number of rows that contain Null values:{sum(count)}')
```

```
Count of NULL values with respect to each column are given below :
Year                     0
AssignedID          276908
Bank                 50502
FIPSStateCode            0
FIPSCountyCode           0
                     ...
Program             466925
FHFBID              466925
Seller              466925
SellCity            466925
SellSt              466925
Length: 88, dtype: int64

Total number of rows that contain Null values:9334733
```

In [16]:

```
print(merged_data_1.shape)
```

```
(743833, 88)
```

In [17]:

```
merged_data_1['HousePrice'] = merged_data_1['NoteAmount'] / merged_data_1
```

In [18]:

```
merged_data_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 743833 entries, 0 to 743832
Data columns (total 89 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   Year               743833 non-null  int64
 1   AssignedID         466925 non-null  float64
 2   Bank               693331 non-null  object
 3   FIPSStateCode      743833 non-null  int64
 4   FIPSCountyCode     743833 non-null  int64
 5   MSA                743833 non-null  int64
 6   FeatureID          457135 non-null  float64
 7   Tract              743833 non-null  float64
 8   MinPer             743833 non-null  float64
 9   TraMedY            743833 non-null  int64
 10  LocMedY            743833 non-null  int64
 11  Tractrat           507070 non-null  float64
 12  Income             743833 non-null  int64
 13  CurAreY            743833 non-null  int64
 14  IncRat             507070 non-null  float64
 15  UPB                743833 non-null  int64
 16  LTV                743833 non-null  float64
 17  MortDate           743833 non-null  int64
 18  AcqDate            743833 non-null  int64
 19  Purpose            743833 non-null  int64
 20  Coop               507070 non-null  float64
 21  Product            743833 non-null  int64
 22  FedGuar            743833 non-null  int64
 23  Term               743833 non-null  int64
 24  AmorTerm           743833 non-null  int64
 25  SellType           743833 non-null  int64
 26  NumBor             743833 non-null  int64
 27  First              743833 non-null  int64
 28  CICA               507070 non-null  float64
 29  BoRace             743833 non-null  int64
 30  CoRace             743833 non-null  int64
 31  BoGender           743833 non-null  int64
 32  CoGender           743833 non-null  int64
 33  BoAge              743833 non-null  int64
 34  CoAge              743833 non-null  int64
 35  Occup              743833 non-null  int64
 36  NumUnits           743833 non-null  int64
 37  Bed1               507070 non-null  float64
 38  Bed2               507070 non-null  float64
 39  Bed3               507070 non-null  float64
 40  Bed4               507070 non-null  float64
 41  Aff1               507070 non-null  float64
 42  Aff2               507070 non-null  float64
 43  Aff3               507070 non-null  float64
 44  Aff4               507070 non-null  float64
 45  Rent1              507070 non-null  float64
 46  Rent2              507070 non-null  float64
 47  Rent3              507070 non-null  float64
 48  Rent4              507070 non-null  float64
 49  RentUt1            507070 non-null  float64
 50  RentUt2            507070 non-null  float64
 51  RentUt3            507070 non-null  float64
 52  RentUt4            507070 non-null  float64
 53  Geog               507070 non-null  float64
 54  Rate               743833 non-null  float64
 55  NoteAmount         743833 non-null  int64
 56  Front              743833 non-null  float64
 57  Back               743833 non-null  float64
 58  BoCreditScore      743833 non-null  int64
 59  CoBoCreditScor     743833 non-null  int64
 60  PMI                743833 non-null  float64
 61  Self               743833 non-null  int64
 62  PropType           743833 non-null  object
 63  ArmIndex           743833 non-null  int64
 64  MarginRatePercent  743833 non-null  int64
 65  PrepayP            743833 non-null  object
 66  BoEth              743833 non-null  int64
 67  Race2              743833 non-null  int64
 68  Race3              743833 non-null  int64
 69  Race4              743833 non-null  int64
 70  Race5              743833 non-null  int64
 71  CoEth              743833 non-null  int64
 72  Corace2            743833 non-null  int64
 73  Corace3            743833 non-null  int64
 74  Corace4            743833 non-null  int64
 75  Corace5            743833 non-null  int64
 76  HOEPA              743833 non-null  int64
 77  LienStatus         743833 non-null  int64
 78  SpcHsgGoals        507070 non-null  float64
 79  FedFinStbltyPlan   507070 non-null  float64
 80  AcqTyp             507070 non-null  float64
 81  GSEREO             507070 non-null  float64
 82  LoanNumber         276908 non-null  float64
 83  Program            276908 non-null  float64
 84  FHFBID             276908 non-null  float64
 85  Seller             276908 non-null  object
 86  SellCity           276908 non-null  object
 87  SellSt             276908 non-null  object
 88  HousePrice         743833 non-null  float64
dtypes: float64(38), int64(45), object(6)
memory usage: 505.1+ MB
```

In [19]:

```
merged_data_1 = merged_data_1.drop(
    ['AcqTyp','Aff1','Aff2','Aff3','Aff4','Bed1','Bed2','Bed3','Bed4','CI
     'RentUt1','RentUt2','RentUt3','RentUt4','GSEREO','FeatureID','FedFin
     'Tractrat', 'LoanNumber','Corace2','Corace3','Corace4','Corace5','Se
     'Race4','Race5','AssignedID','MSA','FeatureID','Program','FHFBID' ],
```

In [20]:

```
merged_data_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 743833 entries, 0 to 743832
Data columns (total 47 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   Year               743833 non-null  int64  
 1   Bank               693331 non-null  object 
 2   FIPSStateCode      743833 non-null  int64  
 3   FIPSCountyCode     743833 non-null  int64  
 4   Tract              743833 non-null  float64
 5   MinPer             743833 non-null  float64
 6   TraMedY            743833 non-null  int64  
 7   LocMedY            743833 non-null  int64  
 8   Income             743833 non-null  int64  
 9   CurAreY            743833 non-null  int64  
 10  UPB                743833 non-null  int64  
 11  LTV                743833 non-null  float64
 12  MortDate           743833 non-null  int64  
 13  AcqDate            743833 non-null  int64  
 14  Purpose            743833 non-null  int64  
 15  Product            743833 non-null  int64  
 16  FedGuar            743833 non-null  int64  
 17  Term               743833 non-null  int64  
 18  AmorTerm           743833 non-null  int64  
 19  SellType           743833 non-null  int64  
 20  NumBor             743833 non-null  int64  
 21  First              743833 non-null  int64  
 22  BoRace             743833 non-null  int64  
 23  CoRace             743833 non-null  int64  
 24  BoGender           743833 non-null  int64  
 25  CoGender           743833 non-null  int64  
 26  BoAge              743833 non-null  int64  
 27  CoAge              743833 non-null  int64  
 28  Occup              743833 non-null  int64  
 29  NumUnits           743833 non-null  int64  
 30  Rate               743833 non-null  float64
 31  NoteAmount         743833 non-null  int64  
 32  Front              743833 non-null  float64
 33  Back               743833 non-null  float64
 34  BoCreditScore      743833 non-null  int64  
 35  CoBoCreditScor     743833 non-null  int64  
 36  PMI                743833 non-null  float64
 37  Self               743833 non-null  int64  
 38  PropType           743833 non-null  object 
 39  ArmIndex           743833 non-null  int64  
 40  MarginRatePercent  743833 non-null  int64  
 41  PrepayP            743833 non-null  object 
 42  BoEth              743833 non-null  int64  
 43  CoEth              743833 non-null  int64  
 44  HOEPA              743833 non-null  int64  
 45  LienStatus         743833 non-null  int64  
 46  HousePrice         743833 non-null  float64
dtypes: float64(8), int64(36), object(3)
memory usage: 266.7+ MB
```

In [21]:

```
merged_data_1.describe()
```

Out[21]:

	Year	FIPSStateCode	FIPSCountyCode	Tract	MinPer	
count	743833.000000	743833.000000	743833.000000	743833.000000	743833.000000	74
mean	2015.710653	28.918924	84.843860	3110.106976	13.884637	7
std	3.785652	12.994376	68.565875	3891.420896	15.829420	2
min	2009.000000	1.000000	1.000000	1.000000	0.000000	
25%	2012.000000	19.000000	37.000000	106.000000	4.050000	5
50%	2016.000000	27.000000	73.000000	529.040000	8.097000	7
75%	2019.000000	39.000000	119.000000	7109.000000	16.969000	9
max	2021.000000	78.000000	840.000000	47700.000000	100.000000	25

8 rows × 44 columns

In [22]:

```
merged_data_1.shape
```

Out[22]:

```
(743833, 47)
```

In [23]:

```
duplicate_values = merged_data_1.duplicated().sum()
print(f'This dataset contains {duplicate_values} duplicate rows')
```

```
This dataset contains 17 duplicate rows
```

In [24]:

```
merged_data_1.drop_duplicates(inplace=True)

duplicate_values = merged_data_1.duplicated().sum()
print(f'This dataset contains {duplicate_values} duplicate rows')
```

```
This dataset contains 0 duplicate rows
```

In [25]:

```python
count = merged_data_1.isnull().sum()
print(f'Count of NULL values with respect to each column are given below
print(f'\nTotal number of rows that contain Null values:{sum(count)}')
```

```
Count of NULL values with respect to each column are given below :
Year                     0
Bank                 50497
FIPSStateCode            0
FIPSCountyCode           0
Tract                    0
MinPer                   0
TraMedY                  0
LocMedY                  0
Income                   0
CurAreY                  0
UPB                      0
LTV                      0
MortDate                 0
AcqDate                  0
Purpose                  0
Product                  0
FedGuar                  0
Term                     0
AmorTerm                 0
SellType                 0
NumBor                   0
First                    0
BoRace                   0
CoRace                   0
BoGender                 0
CoGender                 0
BoAge                    0
CoAge                    0
Occup                    0
NumUnits                 0
Rate                     0
NoteAmount               0
Front                    0
Back                     0
BoCreditScore            0
CoBoCreditScor           0
PMI                      0
Self                     0
PropType                 0
ArmIndex                 0
MarginRatePercent        0
PrepayP                  0
BoEth                    0
CoEth                    0
HOEPA                    0
LienStatus               0
HousePrice               0
dtype: int64

Total number of rows that contain Null values:50497
```

In [26]:

```
count = merged_data_1.isnull().sum()
print(f'\nTotal number of rows that contain Null values:{sum(count)}')
```

```

Total number of rows that contain Null values:50497
```

In [27]:

```
#Dataset balanced/not
# Compute the class distribution of the target variable
class_distribution = merged_data_1['HousePrice'].value_counts()

# Print the class distribution
print(class_distribution)
df = merged_data_1.to_csv("C:\\Users\\Sri Manaswini\\Desktop\\DS_Capstone
```

```
200000.000000    3308
250000.000000    3223
150000.000000    2653
300000.000000    2284
125000.000000    2230
                 ... 
3125.531915         1
3522.820896         1
4892.621622         1
9539.285714         1
259816.494845       1
Name: HousePrice, Length: 196719, dtype: int64
```

In [5]:

```
#Read in data
final_df =pd.read_csv('C:\\Users\\Sri Manaswini\\Desktop\\DS_Capstone\\FH
final_df.head()
```

Out[5]:

	Year	Bank	FIPSStateCode	FIPSCountyCode	Tract	MinPer	TraMedY	LocMedY	In
0	2015	Atlanta	51	69	511.01	16.96	88049	61537	1
1	2015	Atlanta	18	39	5.02	30.64	47088	53742	12
2	2015	Atlanta	13	245	109.03	35.36	66219	54953	7
3	2015	Atlanta	12	9	644.00	13.56	51191	60842	8
4	2015	Atlanta	32	3	32.32	22.33	101161	63888	10

5 rows × 47 columns

In [29]:

```
final_df.shape
```

Out[29]:

```
(743816, 47)
```

In [30]:

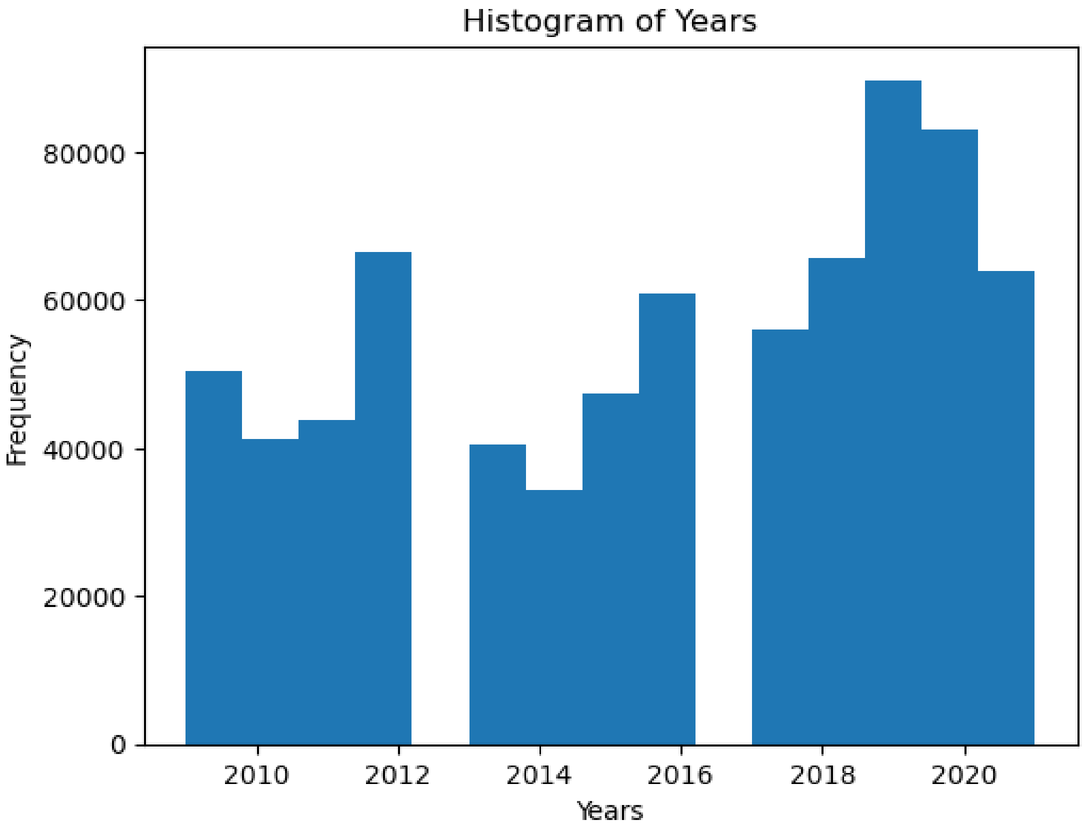
```
final_df.dtypes
```

Out[30]:

```
Year                   int64
Bank                  object
FIPSStateCode          int64
FIPSCountyCode         int64
Tract                float64
MinPer               float64
TraMedY                int64
LocMedY                int64
Income                 int64
CurAreY                int64
UPB                    int64
LTV                  float64
MortDate               int64
AcqDate                int64
Purpose                int64
Product                int64
FedGuar                int64
Term                   int64
AmorTerm               int64
SellType               int64
NumBor                 int64
First                  int64
BoRace                 int64
CoRace                 int64
BoGender               int64
CoGender               int64
BoAge                  int64
CoAge                  int64
Occup                  int64
NumUnits               int64
Rate                 float64
NoteAmount             int64
Front                float64
Back                 float64
BoCreditScore          int64
CoBoCreditScor         int64
PMI                  float64
Self                   int64
PropType              object
ArmIndex               int64
MarginRatePercent      int64
PrepayP               object
BoEth                  int64
CoEth                  int64
HOEPA                  int64
LienStatus             int64
HousePrice           float64
dtype: object
```

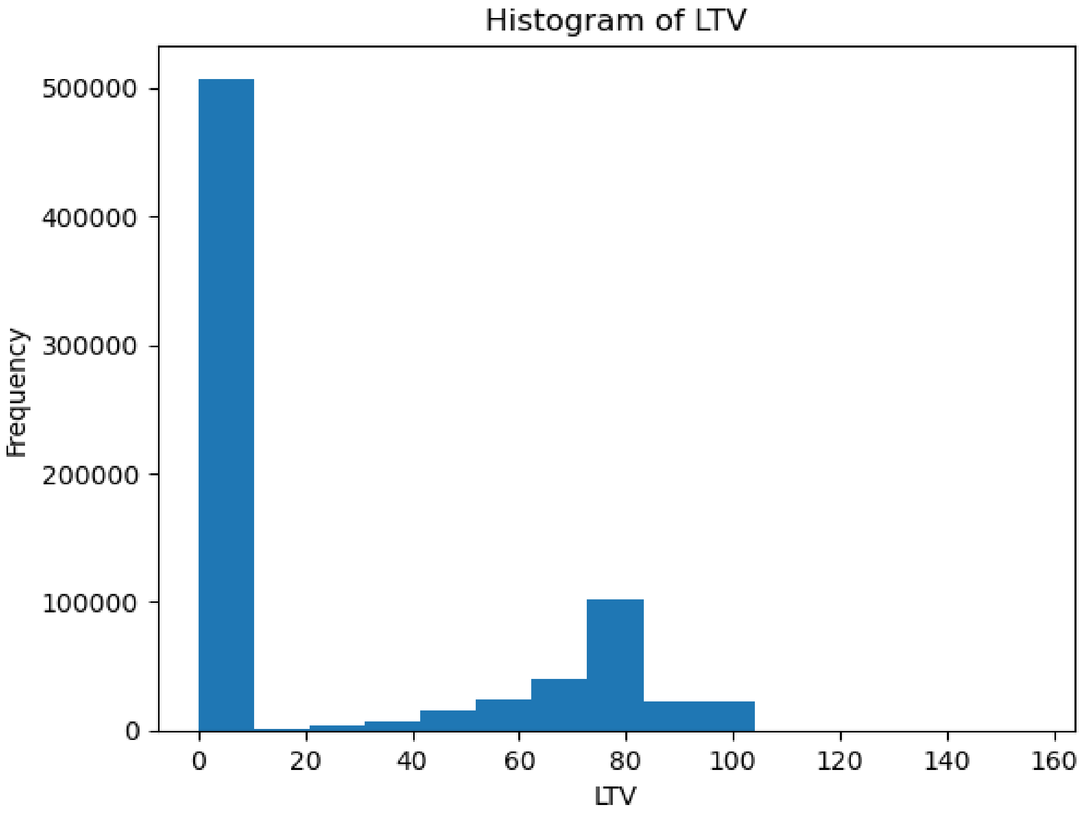
In [31]:

```
import matplotlib.pyplot as plt

# Assuming your DataFrame is called 'df' and contains the columns 'HouseP

# Plot histogram for HousePrice
plt.hist(final_df['Year'], bins=15)  # Adjust the number of bins as neede
plt.xlabel('Years')
plt.ylabel('Frequency')
plt.title('Histogram of Years')
plt.show()
```



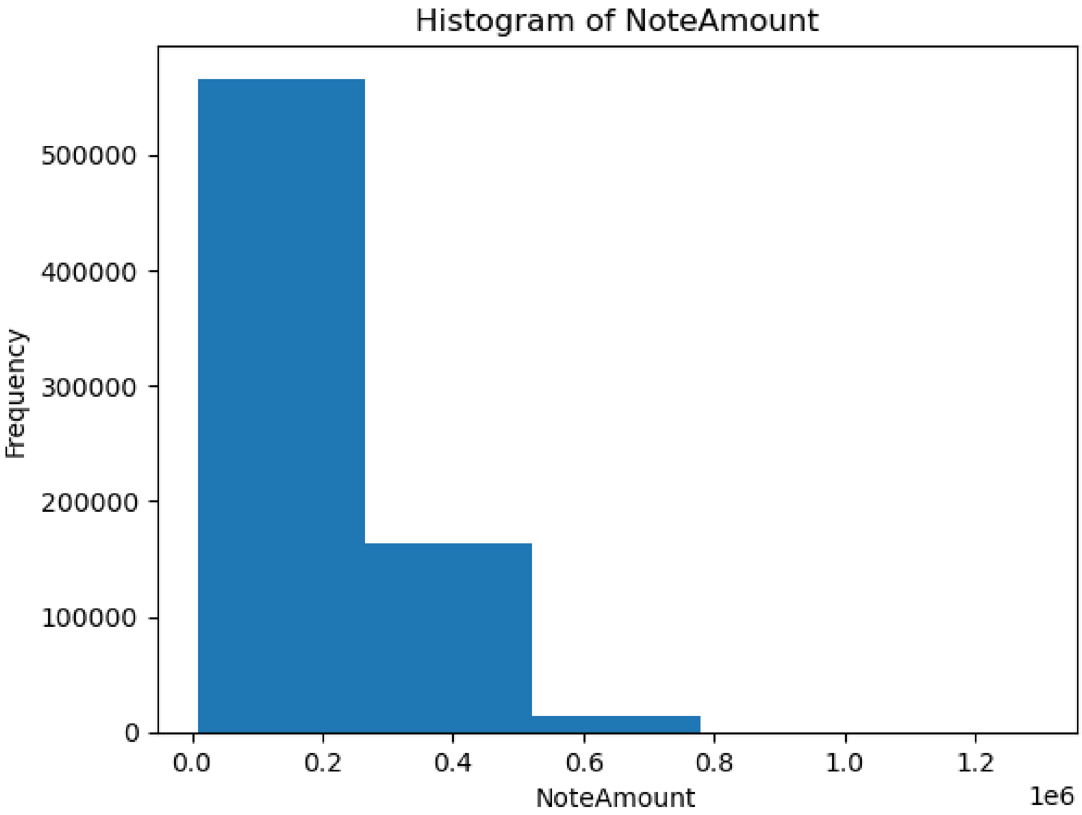
In [32]:

```python
import matplotlib.pyplot as plt

# Assuming your DataFrame is called 'df' and contains the columns 'HouseP

# Plot histogram for HousePrice
plt.hist(final_df['LTV'], bins=15)  # Adjust the number of bins as needed
plt.xlabel('LTV')
plt.ylabel('Frequency')
plt.title('Histogram of LTV')
plt.show()
```



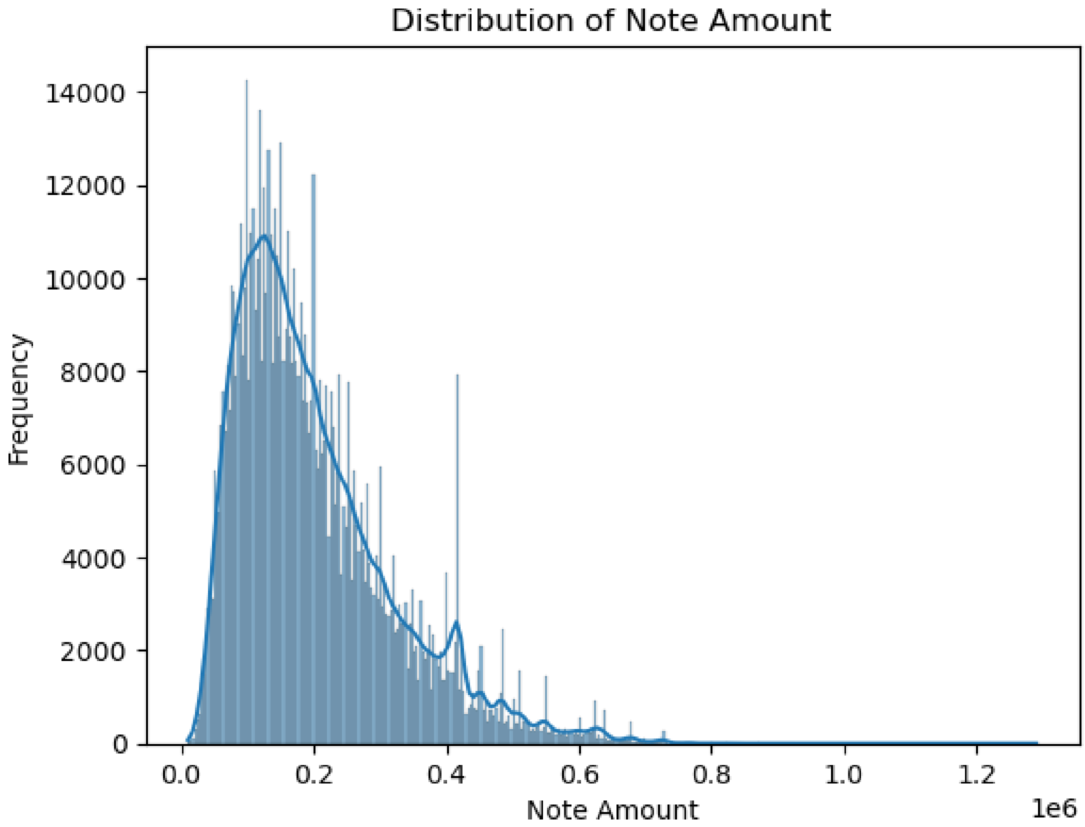
In [59]:

```python
import matplotlib.pyplot as plt

# Assuming your DataFrame is called 'df' and contains the columns 'HouseP

# Plot histogram for HousePrice
plt.hist(final_df['NoteAmount'], bins=5)  # Adjust the number of bins as
plt.xlabel('NoteAmount')
plt.ylabel('Frequency')
plt.title('Histogram of NoteAmount')
plt.show()
```



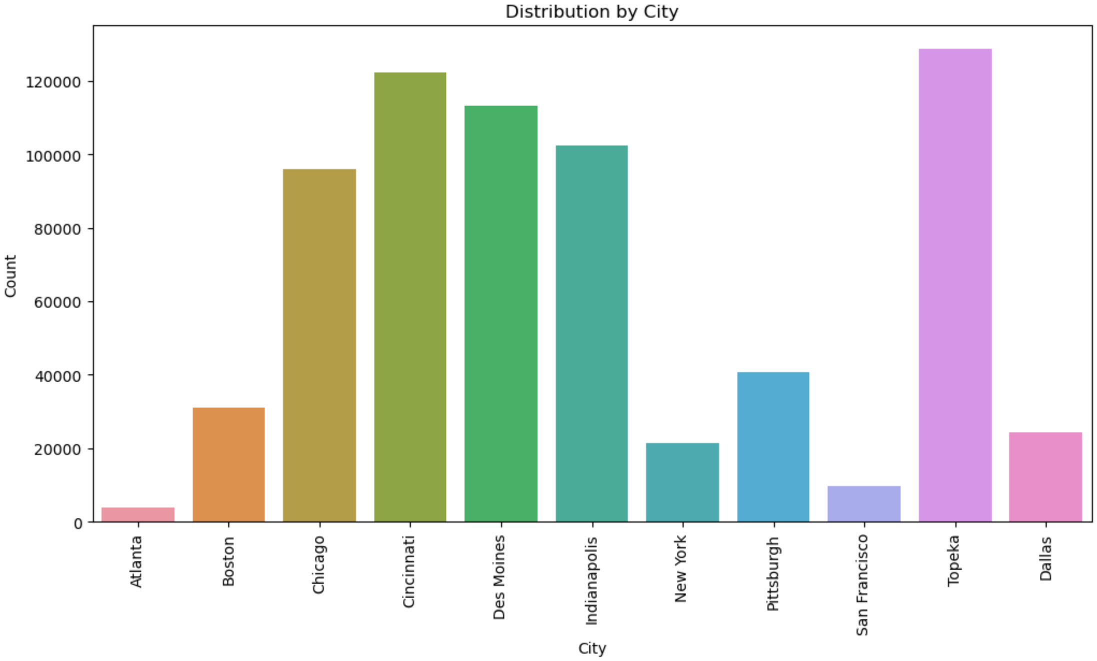
In [61]:

```python
note_amount = final_df['NoteAmount']

# Plotting a histogram with continuous bins
sns.histplot(note_amount, bins='auto', kde=True)
plt.xlabel('Note Amount')
plt.ylabel('Frequency')
plt.title('Distribution of Note Amount')
plt.show()
```



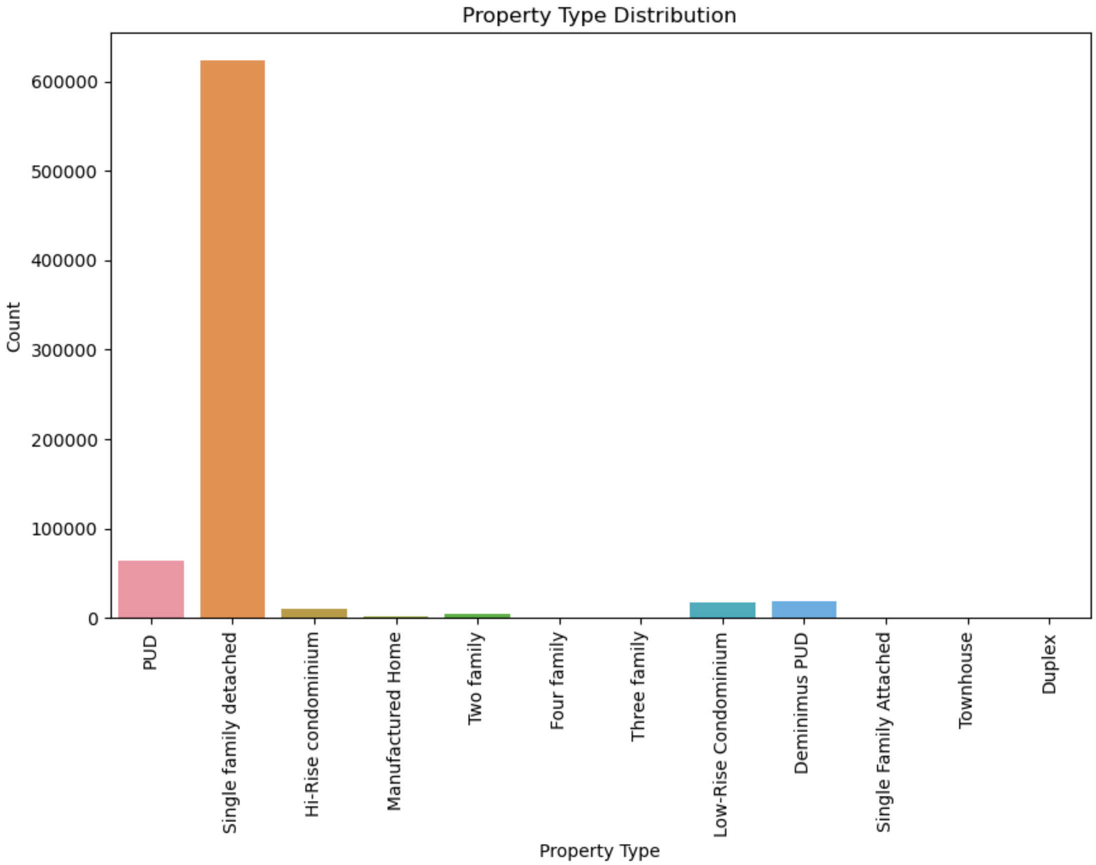
In [34]:

```
plt.figure(figsize=(12, 6))  # Adjust the figure size as needed
sns.countplot(data=merged_data, x='Bank')
plt.xlabel('City')
plt.ylabel('Count')
plt.title( 'Distribution by City')
plt.xticks(rotation=90)  # Rotate x-axis labels for better readability
plt.show()
```



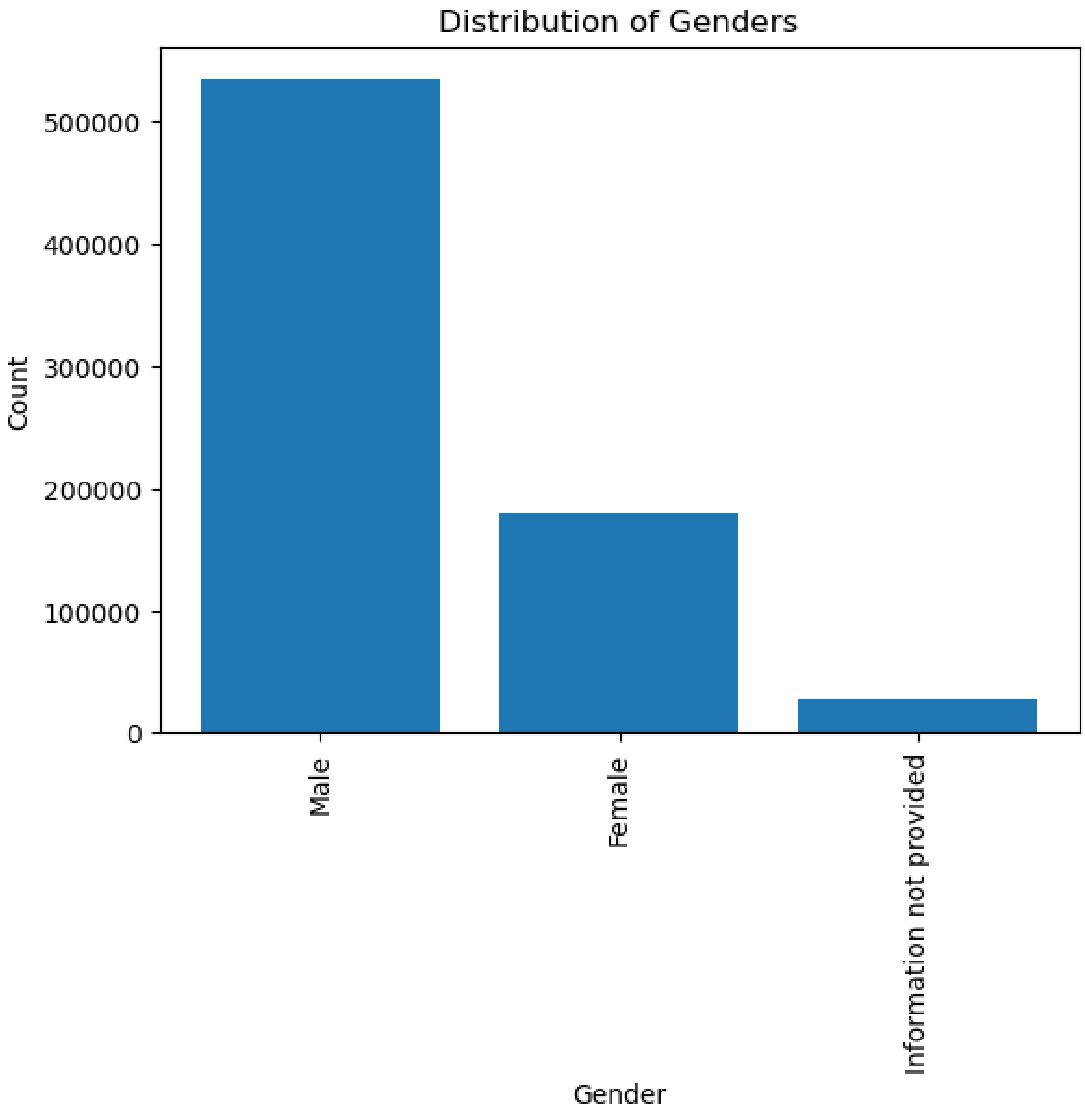
In [35]:

```
prop_type_mapping = {
    'PT01': 'Single family detached',
    'PT02': 'Deminimus PUD',
    'PT03': 'Single Family Attached',
    'PT04': 'Two family',
    'PT05': 'Townhouse',
    'PT06': 'Low-Rise Condominium',
    'PT07': 'PUD',
    'PT08': 'Duplex',
    'PT09': 'Three family',
    'PT10': 'Four family',
    'PT11': 'Hi-Rise condominium',
    'PT12': 'Manufactured Home'
}

# Replace PropType values with their descriptions
final_df['PropType_Description'] = final_df['PropType'].map(prop_type_map



plt.figure(figsize=(10, 6))  # Adjust the figure size as needed
sns.countplot(data=final_df, x='PropType_Description')
plt.xlabel('Property Type')
plt.ylabel('Count')
plt.title('Property Type Distribution')
plt.xticks(rotation=90)  # Rotate x-axis labels for better readability
plt.show()
```



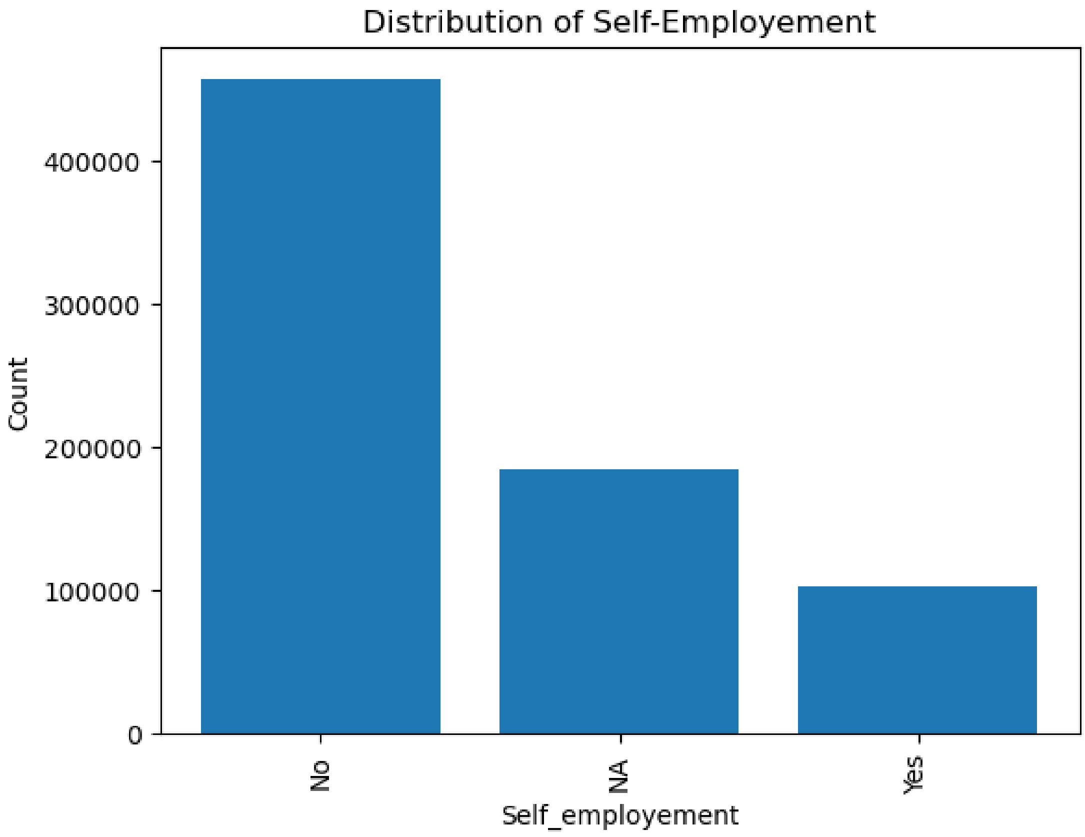
In [36]:

```
gender_mapping = {1: 'Male', 2: 'Female', 3: 'Information not provided'}

# Replace the numerical values with their descriptions
final_df['BoGender'] = final_df['BoGender'].map(gender_mapping)

# Count the occurrences of each gender category
gender_counts = final_df['BoGender'].value_counts()

# Create a bar plot
plt.bar(gender_counts.index, gender_counts.values)

# Add labels and title
plt.xlabel('Gender')
plt.ylabel('Count')
plt.title('Distribution of Genders')

# Rotate x-axis labels if needed
plt.xticks(rotation='vertical')

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



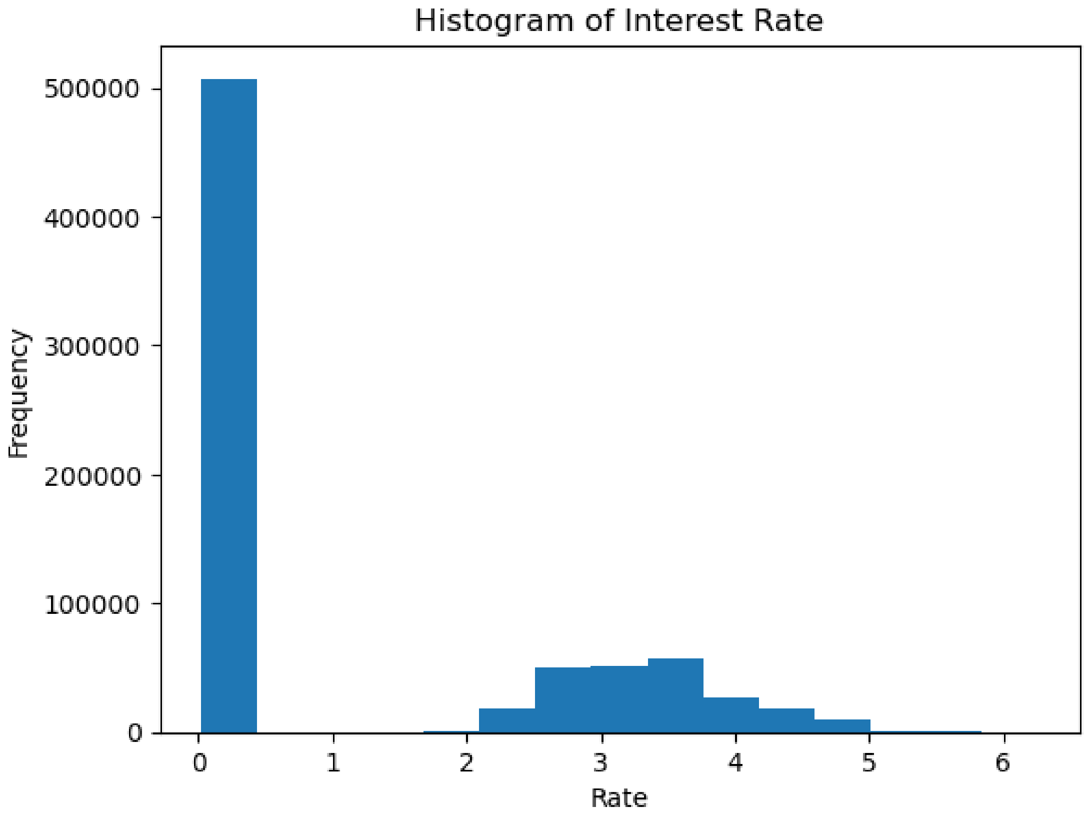
In [37]:

```python
prop_type_mapping = {
    1: 'Yes',
    2: 'No',
    0: 'NA'
}

# Replace PropType values with their descriptions
final_df['Self_employement_description'] = final_df['Self'].map(prop_type
Self_counts = final_df['Self_employement_description'].value_counts()

plt.bar(Self_counts.index, Self_counts.values)

# Add labels and title
plt.xlabel('Self_employement')
plt.ylabel('Count')
plt.title('Distribution of Self-Employement')

# Rotate x-axis labels if needed
plt.xticks(rotation='vertical')

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



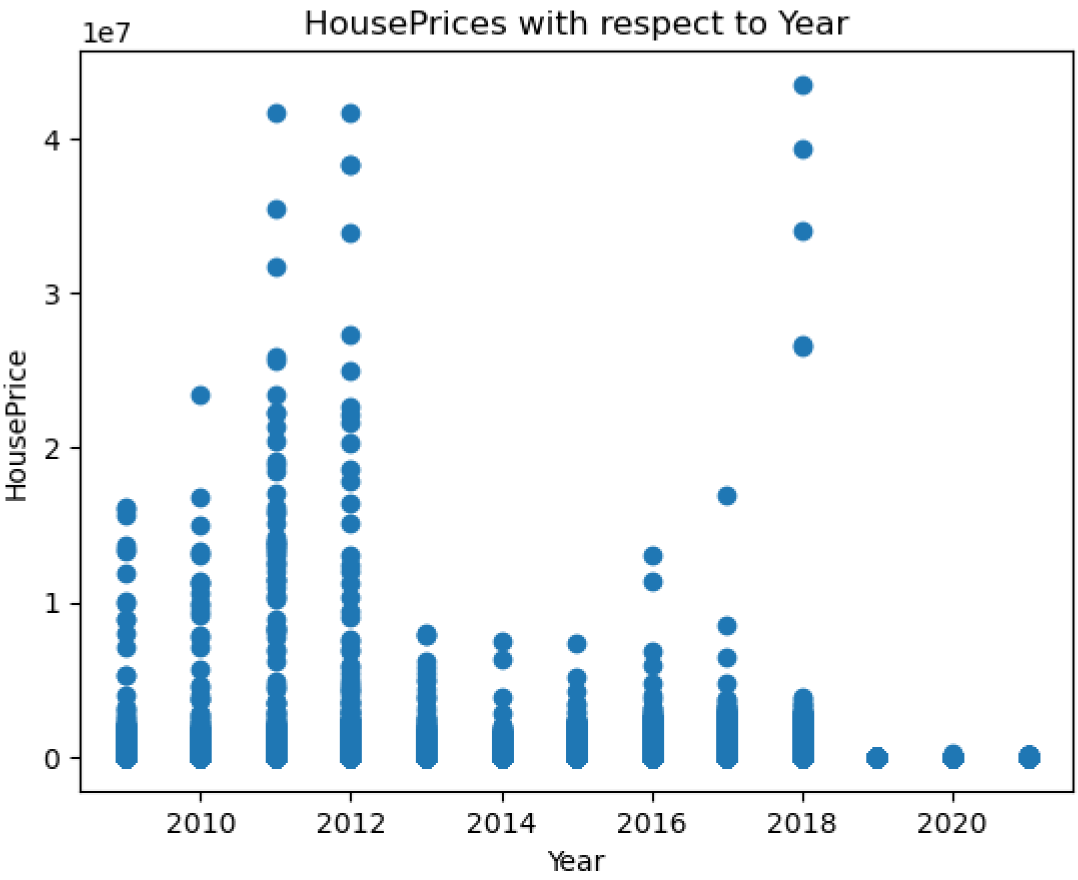
In [38]:

```
import matplotlib.pyplot as plt

# Assuming your DataFrame is called 'df' and contains the columns 'HouseP

# Plot histogram for HousePrice
plt.hist(final_df['Rate'], bins=15)  # Adjust the number of bins as neede
plt.xlabel('Rate')
plt.ylabel('Frequency')
plt.title('Histogram of Interest Rate')
plt.show()
```



In [39]:

```
house_price = final_df['HousePrice']
year = final_df['Year']

# Create a scatter plot
plt.scatter(year, house_price)

# Add labels and title
plt.xlabel('Year')
plt.ylabel('HousePrice')
plt.title('HousePrices with respect to Year')

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



In [40]:

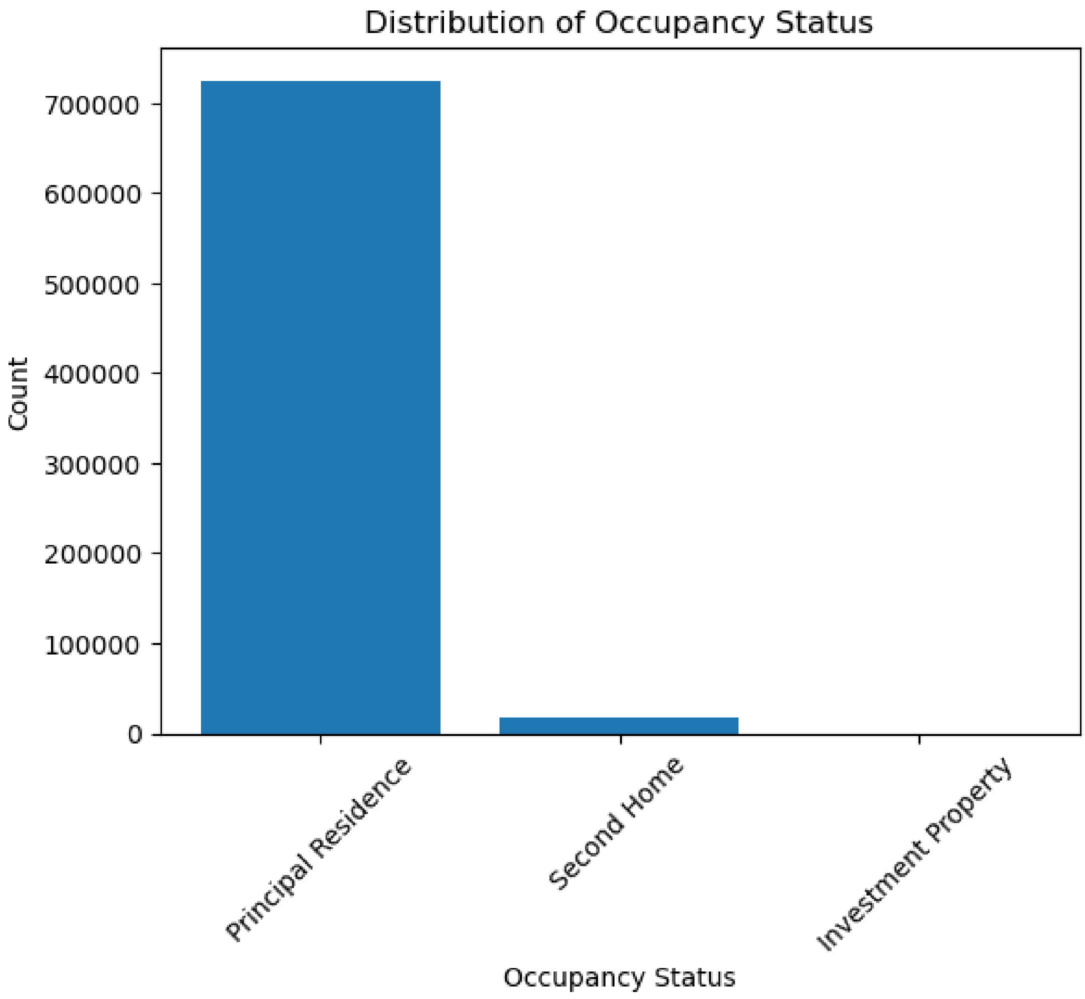
```
final_df.shape
```

Out[40]:

```
(743816, 49)
```

In [41]:

```
occup_mapping = {
    1: 'Principal Residence',
    2: 'Second Home',
    3: 'Investment Property'
}

# Map the values in the 'Occup' column to their descriptions
final_df['Occup_Description'] = final_df['Occup'].map(occup_mapping)

# Count the occurrences of each occupancy status
occup_counts = final_df['Occup_Description'].value_counts()

# Create a bar plot
plt.bar(occup_counts.index, occup_counts.values)

# Add labels and title
plt.xlabel('Occupancy Status')
plt.ylabel('Count')
plt.title('Distribution of Occupancy Status')

# Rotate the x-axis labels if needed
plt.xticks(rotation=45)

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



In [55]:

```
# Specify the columns for which you want the description
columns = ['HousePrice']

# Get the description of the specified columns
description = final_df[columns].info()

# Print the description
print(description)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 743816 entries, 0 to 743815
Data columns (total 1 columns):
 #   Column      Non-Null Count   Dtype  
---  ------      --------------   -----  
 0   HousePrice  743816 non-null  float64
dtypes: float64(1)
memory usage: 5.7 MB
None
```

In [ ]:

```

```

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima.model import ARIMA
df = pd.read_csv('/content/merged_data2.csv')
```

In [ ]:

In [ ]:

```
print(df.columns)
```

```
Index(['Year', 'Bank', 'FIPSStateCode', 'FIPSCountyCode', 'Tract', 'MinPer',
       'TraMedY', 'LocMedY', 'Income', 'CurAreY', 'UPB', 'LTV', 'MortDate',
       'AcqDate', 'Purpose', 'Product', 'FedGuar', 'Term', 'AmorTerm',
       'SellType', 'NumBor', 'First', 'BoRace', 'CoRace', 'BoGender',
       'CoGender', 'BoAge', 'CoAge', 'Occup', 'NumUnits', 'Rate', 'NoteAmount',
       'Front', 'Back', 'BoCreditScore', 'CoBoCreditScor', 'PMI', 'Self',
       'PropType', 'ArmIndex', 'MarginRatePercent', 'PrepayP', 'BoEth',
       'CoEth', 'HOEPA', 'LienStatus', 'HousePrice'],
      dtype='object')
```

In [ ]:

```
column_dtypes = df.dtypes
print(column_dtypes)
```

```
Year                   int64
Bank                  object
FIPSStateCode          int64
FIPSCountyCode         int64
Tract                float64
MinPer               float64
TraMedY                int64
LocMedY                int64
Income                 int64
CurAreY                int64
UPB                    int64
LTV                  float64
MortDate               int64
AcqDate                int64
Purpose                int64
Product                int64
FedGuar                int64
Term                   int64
AmorTerm               int64
SellType               int64
NumBor                 int64
First                  int64
BoRace                 int64
CoRace                 int64
BoGender               int64
CoGender               int64
BoAge                  int64
CoAge                  int64
Occup                  int64
NumUnits               int64
Rate                 float64
NoteAmount             int64
Front                float64
Back                 float64
BoCreditScore          int64
CoBoCreditScor         int64
PMI                  float64
Self                   int64
PropType              object
ArmIndex               int64
MarginRatePercent      int64
PrepayP               object
BoEth                  int64
CoEth                  int64
HOEPA                  int64
LienStatus             int64
HousePrice           float64
dtype: object
```

In [ ]:
```
print(df['Year'].unique())
```

```
[2015 2016 2017 2018 2019 2020 2021 2009 2010 2011 2012 2013 2014]
```

In [ ]:
```
df['Year'] = pd.to_datetime(df['Year'], format='%Y')
```

In [ ]:
```
df.dtypes
```

Out[ ]:

```
Year                 datetime64[ns]
Bank                         object
FIPSStateCode                 int64
FIPSCountyCode                int64
Tract                       float64
MinPer                      float64
TraMedY                       int64
LocMedY                       int64
Income                        int64
CurAreY                       int64
UPB                           int64
LTV                         float64
MortDate                      int64
AcqDate                       int64
Purpose                       int64
Product                       int64
FedGuar                       int64
Term                          int64
AmorTerm                      int64
SellType                      int64
NumBor                        int64
First                         int64
BoRace                        int64
CoRace                        int64
BoGender                      int64
CoGender                      int64
BoAge                         int64
CoAge                         int64
Occup                         int64
NumUnits                      int64
Rate                        float64
NoteAmount                    int64
Front                       float64
Back                        float64
BoCreditScore                 int64
CoBoCreditScor                int64
PMI                         float64
Self                          int64
PropType                     object
ArmIndex                      int64
MarginRatePercent             int64
PrepayP                      object
BoEth                         int64
CoEth                         int64
HOEPA                         int64
LienStatus                    int64
HousePrice                  float64
dtype: object
```

In [ ]:

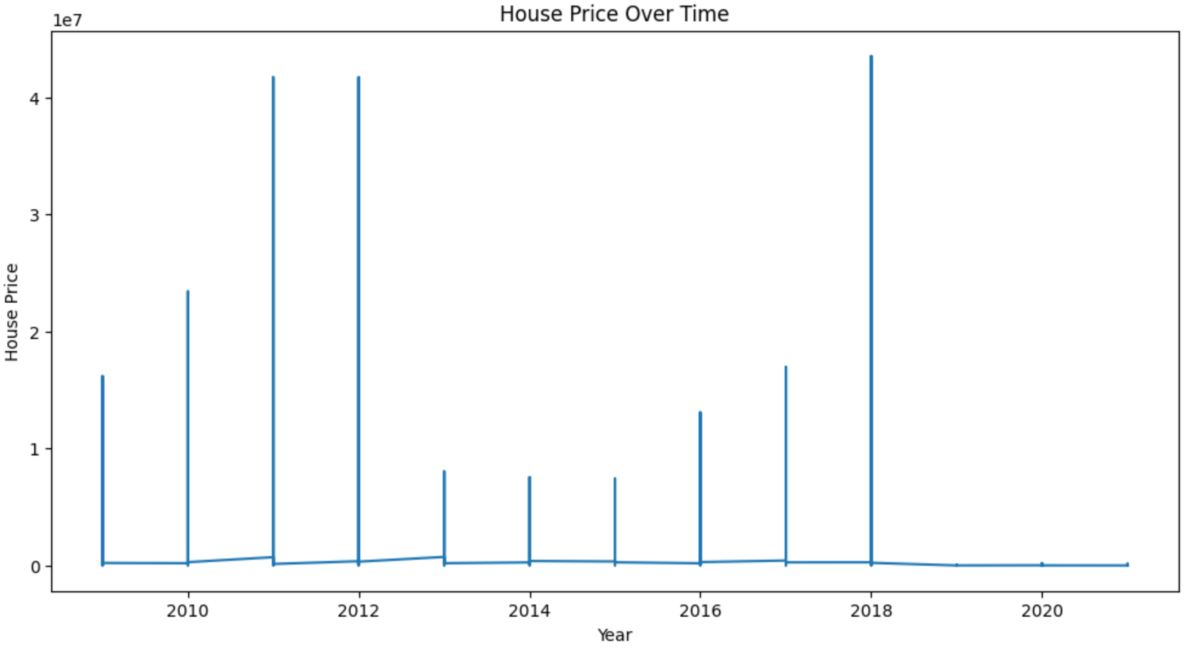
```
df.set_index('Year', inplace=True)
df.sort_index(inplace=True)
```

In [ ]:

```
data = df[['TraMedY','FIPSStateCode','FIPSCountyCode','Tract','Income','SellType','
```

In [ ]:

```
plt.figure(figsize=(12, 6))
plt.plot(df.index, df['HousePrice'])
plt.xlabel('Year')
plt.ylabel('House Price')
plt.title('House Price Over Time')
plt.show()
```



In [ ]:

In [ ]:

```
missing_values = df['HousePrice'].isna().sum()
print("Number of missing values in 'HousePrice' column:", missing_values)
```

```
Number of missing values in 'HousePrice' column: 0
```

In [ ]:

```
unique_values = df['HousePrice'].unique()
print(unique_values)
```

```
[132500.        282894.7368    420000.        ...    2858.426966    3452.307692
   3548.571429]
```

In [ ]:

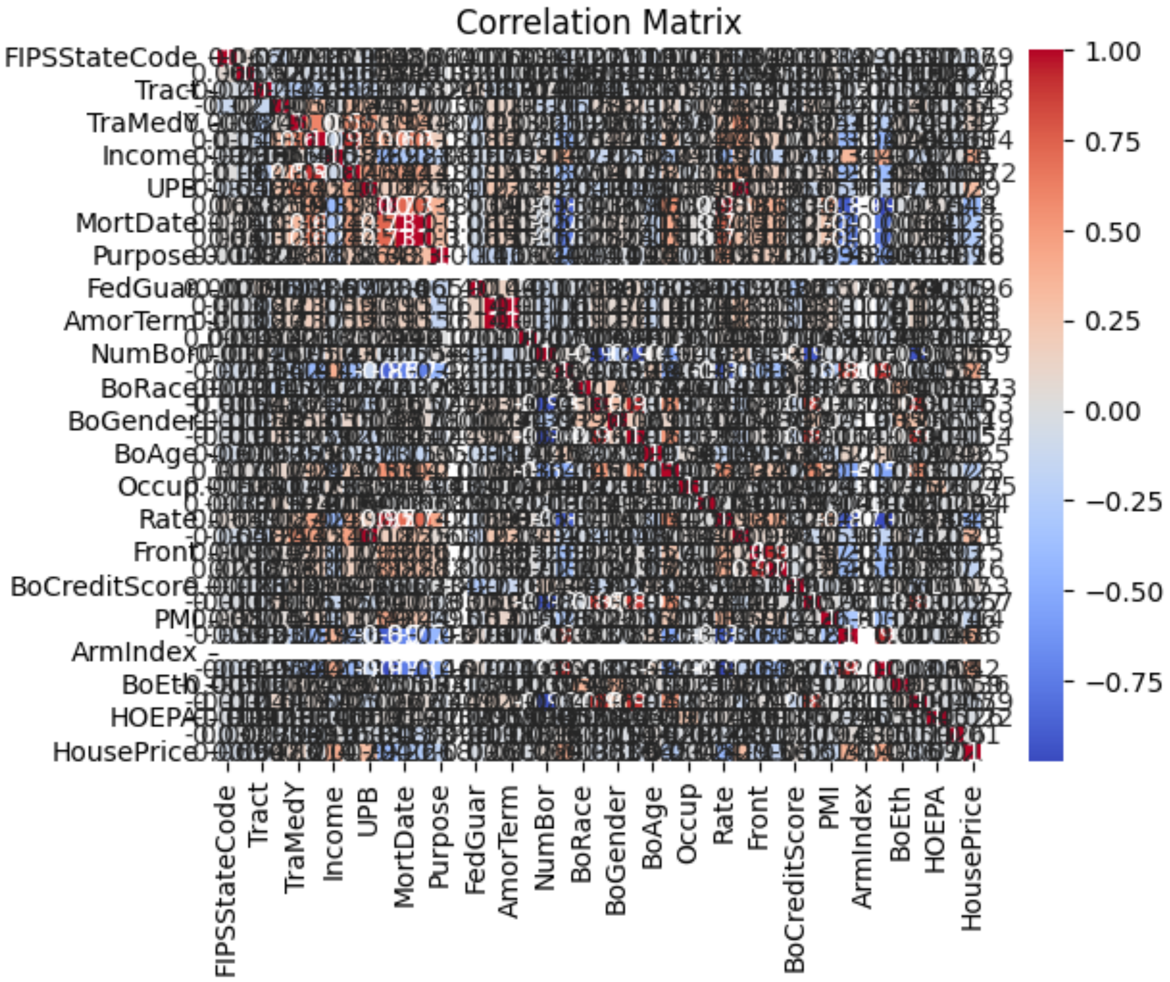
```
missing_values = df.isnull().sum()
print(missing_values)
```

```
Bank                 0
FIPSStateCode        0
FIPSCountyCode       0
Tract                0
MinPer               0
TraMedY              0
LocMedY              0
Income               0
CurAreY              0
UPB                  0
LTV                  0
MortDate             0
AcqDate              0
Purpose              0
Product              0
FedGuar              0
Term                 0
AmorTerm             0
SellType             0
NumBor               0
First                0
BoRace               0
CoRace               0
BoGender             0
CoGender             0
BoAge                0
CoAge                0
Occup                0
NumUnits             0
Rate                 0
NoteAmount           0
Front                0
Back                 0
BoCreditScore        0
CoBoCreditScor       0
PMI                  0
Self                 0
PropType             0
ArmIndex             0
MarginRatePercent    0
PrepayP              0
BoEth                0
CoEth                0
HOEPA                0
LienStatus           0
HousePrice           0
dtype: int64
```

In [ ]:

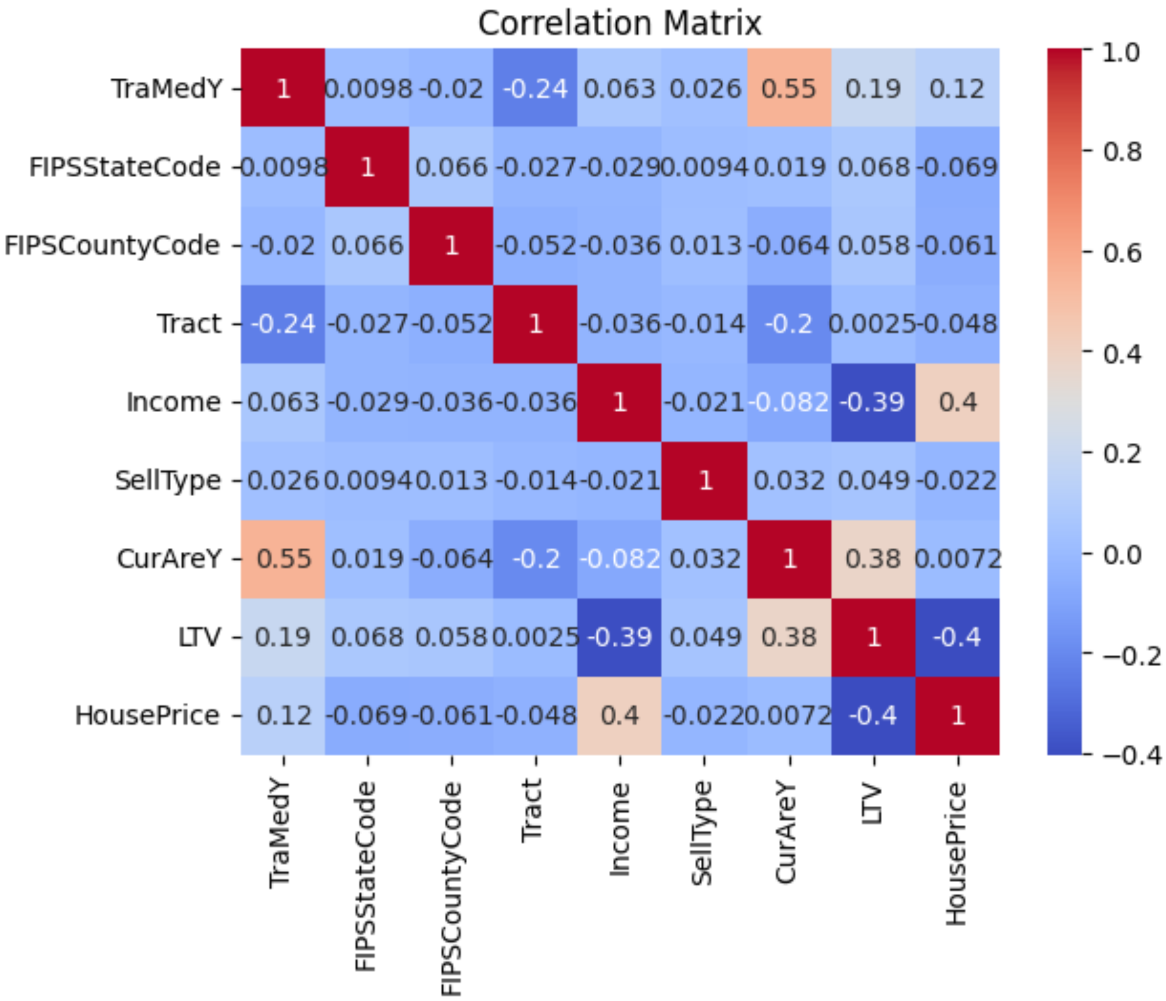
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

```
<ipython-input-13-5a57b9f524a2>:7: FutureWarning: The default value of numeric_only
in DataFrame.corr is deprecated. In a future version, it will default to False. Sel
ect only valid columns or specify the value of numeric_only to silence this warnin
g.
  correlation_matrix = df.corr()
```



```
In [ ]: correlation = df[['TraMedY','FIPSStateCode','FIPSCountyCode','Tract','Income','Sell
```

```
In [ ]: import seaborn as sns
        import matplotlib.pyplot as plt

        sns.heatmap(correlation, annot=True, cmap='coolwarm')
        plt.title('Correlation Matrix')
        plt.show()
```



	TraMedY	FIPSStateCode	FIPSCountyCode	Tract	Income	SellType	CurAreY	LTV	HousePrice
TraMedY	1	0.0098	-0.02	-0.24	0.063	0.026	0.55	0.19	0.12
FIPSStateCode	0.0098	1	0.066	-0.027	-0.029	0.0094	0.019	0.068	-0.069
FIPSCountyCode	-0.02	0.066	1	-0.052	-0.036	0.013	-0.064	0.058	-0.061
Tract	-0.24	-0.027	-0.052	1	-0.036	-0.014	-0.2	0.0025	-0.048
Income	0.063	-0.029	-0.036	-0.036	1	-0.021	-0.082	-0.39	0.4
SellType	0.026	0.0094	0.013	-0.014	-0.021	1	0.032	0.049	-0.022
CurAreY	0.55	0.019	-0.064	-0.2	-0.082	0.032	1	0.38	0.0072
LTV	0.19	0.068	0.058	0.0025	-0.39	0.049	0.38	1	-0.4
HousePrice	0.12	-0.069	-0.061	-0.048	0.4	-0.022	0.0072	-0.4	1

```
In [ ]: df
```

Out[ ]:

	Bank	FIPSStateCode	FIPSCountyCode	Tract	MinPer	TraMedY	LocMedY	Inco
Year								
2009-01-01	New Madison	29	127	9605.00	12.33	23547	30047	90
2009-01-01	Cincinnati	31	155	9881.00	2.07	45926	44649	91
2009-01-01	Geneva	20	45	7.97	10.95	55234	37701	108
2009-01-01	Grand Island	20	13	9807.00	10.20	35092	33784	73
2009-01-01	Fremont	31	141	9853.00	7.25	40897	34122	111
...	...	...	...	...	...	...	...	
2021-01-01	Cincinnati	39	135	4001.00	3.00	57200	67512	27
2021-01-01	Cincinnati	39	15	9515.00	2.00	67460	79402	26
2021-01-01	Cincinnati	39	45	307.00	5.00	99167	78289	7
2021-01-01	Cincinnati	39	61	260.01	2.00	106765	79402	12
2021-01-01	Chicago	55	125	9505.00	4.21	57500	70111	8

743816 rows × 46 columns

In [ ]:
```
print(df.columns)
```

```
Index(['Bank', 'FIPSStateCode', 'FIPSCountyCode', 'Tract', 'MinPer', 'TraMedY',
       'LocMedY', 'Income', 'CurAreY', 'UPB', 'LTV', 'MortDate', 'AcqDate',
       'Purpose', 'Product', 'FedGuar', 'Term', 'AmorTerm', 'SellType',
       'NumBor', 'First', 'BoRace', 'CoRace', 'BoGender', 'CoGender', 'BoAge',
       'CoAge', 'Occup', 'NumUnits', 'Rate', 'NoteAmount', 'Front', 'Back',
       'BoCreditScore', 'CoBoCreditScor', 'PMI', 'Self', 'PropType',
       'ArmIndex', 'MarginRatePercent', 'PrepayP', 'BoEth', 'CoEth', 'HOEPA',
       'LienStatus', 'HousePrice'],
      dtype='object')
```

In [ ]:
```
df
```

Out[ ]:

	Bank	FIPSStateCode	FIPSCountyCode	Tract	MinPer	TraMedY	LocMedY	Inco
Year								
2009-01-01	New Madison	29	127	9605.00	12.33	23547	30047	90
2009-01-01	Cincinnati	31	155	9881.00	2.07	45926	44649	91
2009-01-01	Geneva	20	45	7.97	10.95	55234	37701	108
2009-01-01	Grand Island	20	13	9807.00	10.20	35092	33784	73
2009-01-01	Fremont	31	141	9853.00	7.25	40897	34122	111
...	...	...	...	...	...	...	...	
2021-01-01	Cincinnati	39	135	4001.00	3.00	57200	67512	27
2021-01-01	Cincinnati	39	15	9515.00	2.00	67460	79402	26
2021-01-01	Cincinnati	39	45	307.00	5.00	99167	78289	7
2021-01-01	Cincinnati	39	61	260.01	2.00	106765	79402	12
2021-01-01	Chicago	55	125	9505.00	4.21	57500	70111	8

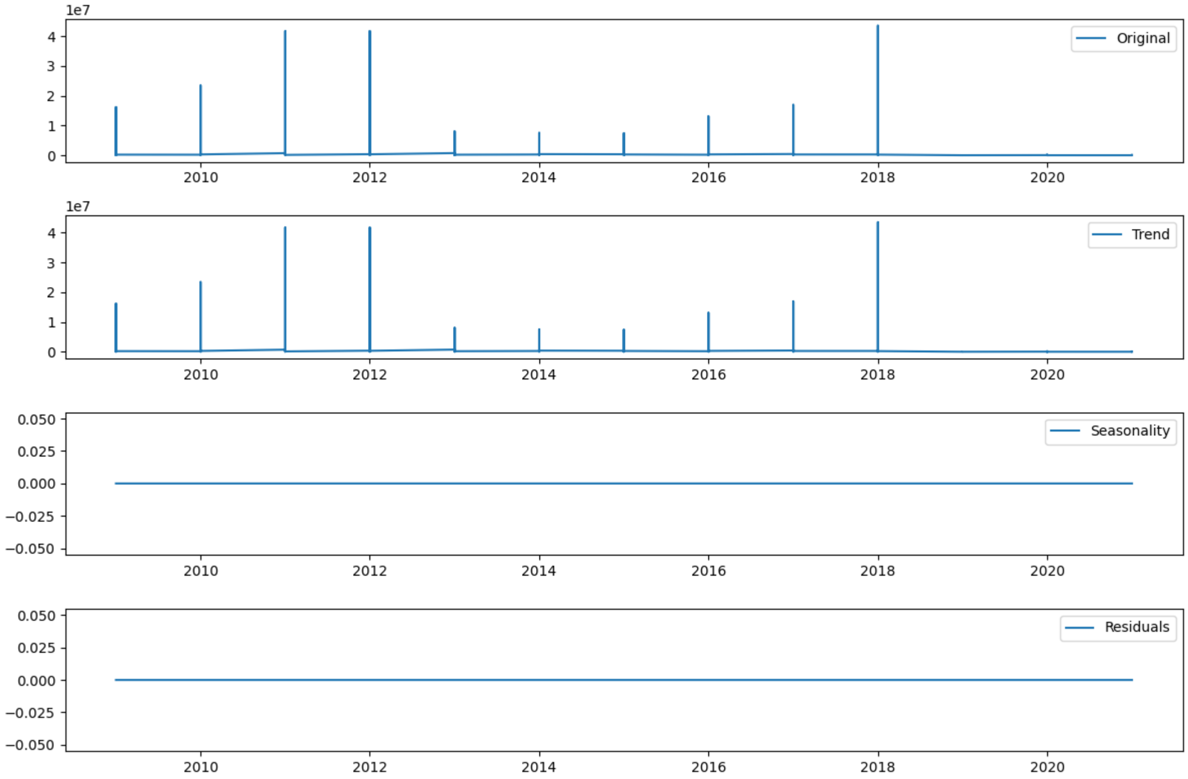
743816 rows × 46 columns

In [ ]:

```
from statsmodels.tsa.seasonal import seasonal_decompose
import matplotlib.pyplot as plt

decomposition = seasonal_decompose(df['HousePrice'], model='additive', period=1)
```

In [ ]:

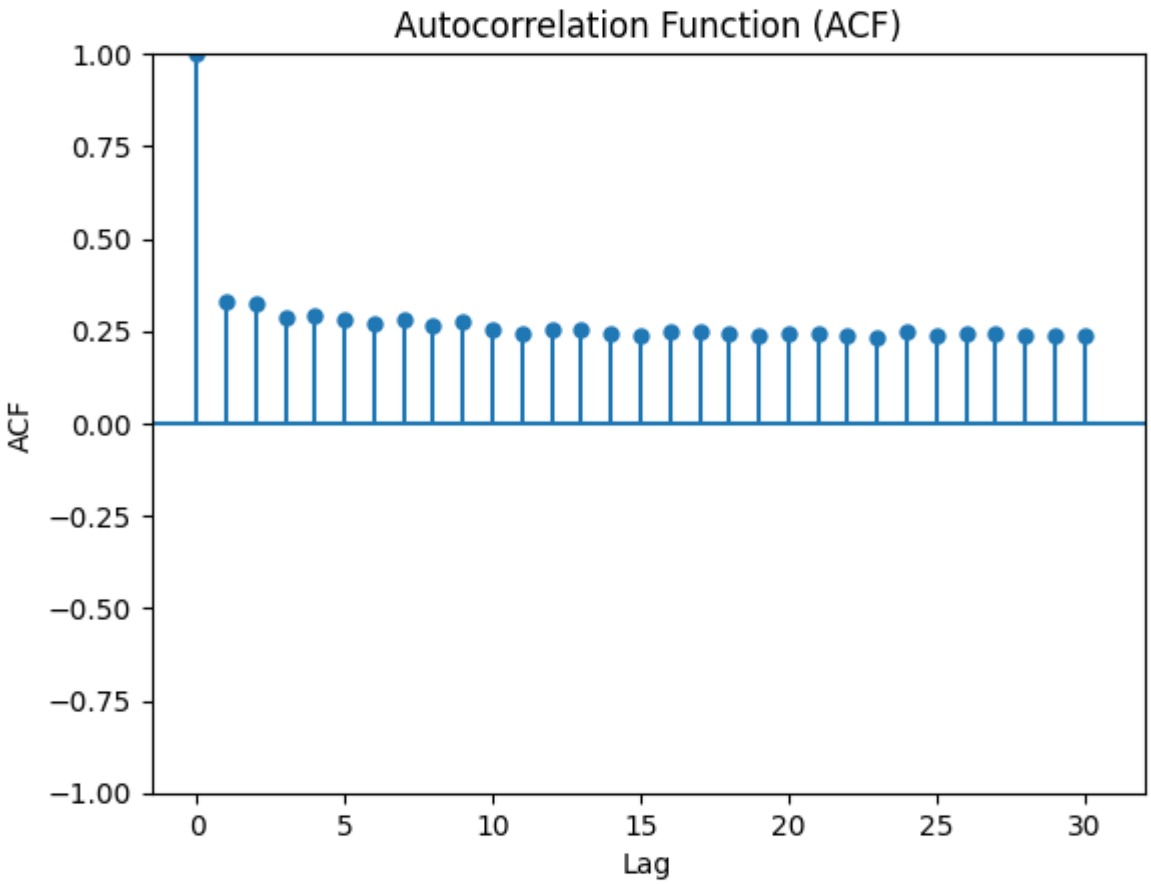
```
trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.resid

plt.figure(figsize=(12, 8))
plt.subplot(411)
plt.plot(df.index, df['HousePrice'], label='Original')
plt.legend(loc='best')
plt.subplot(412)
plt.plot(df.index, trend, label='Trend')
plt.legend(loc='best')
plt.subplot(413)
plt.plot(df.index, seasonal, label='Seasonality')
plt.legend(loc='best')
plt.subplot(414)
plt.plot(df.index, residual, label='Residuals')
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```



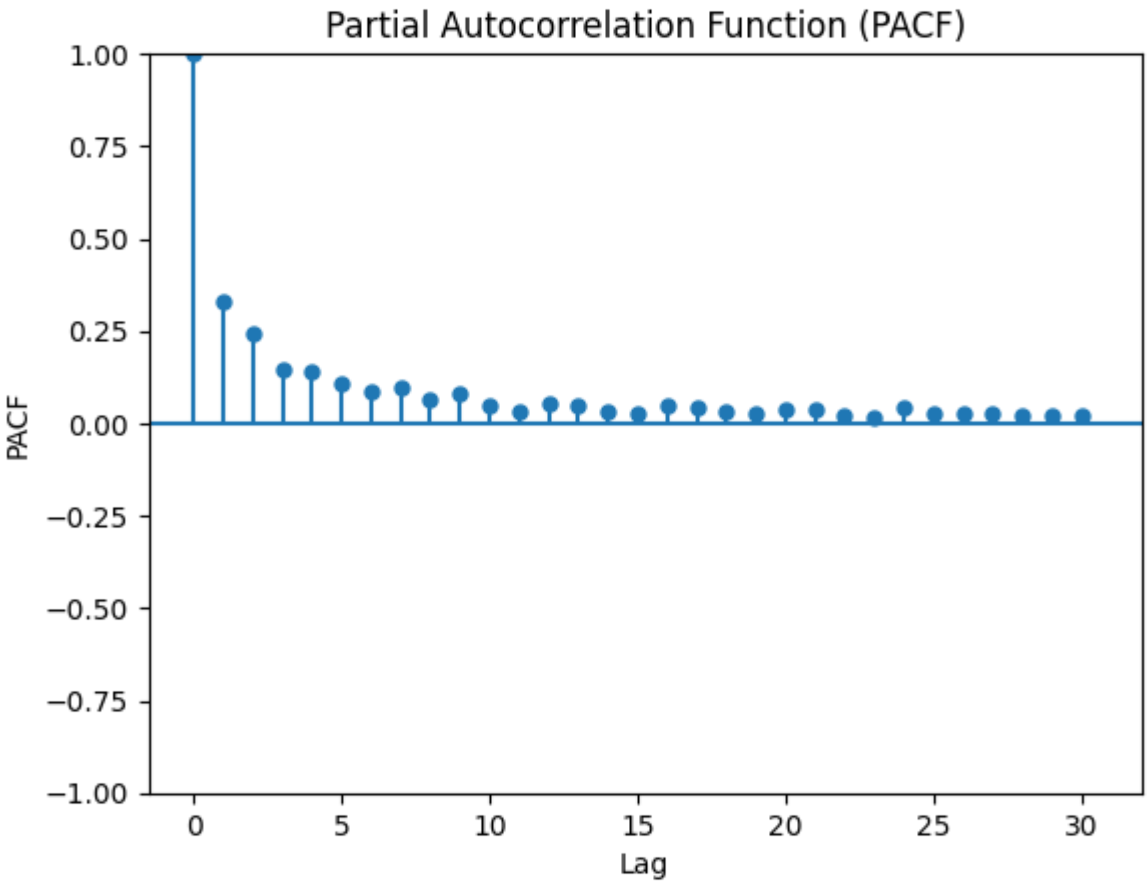
In [ ]:

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf


plt.figure(figsize=(12, 6))
plot_acf(df['HousePrice'], lags=30, alpha=0.05)
plt.title('Autocorrelation Function (ACF)')
plt.xlabel('Lag')
plt.ylabel('ACF')
plt.show()

plt.figure(figsize=(12, 6))
plot_pacf(df['HousePrice'], lags=30, alpha=0.05)
plt.title('Partial Autocorrelation Function (PACF)')
plt.xlabel('Lag')
plt.ylabel('PACF')
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```



```
/usr/local/lib/python3.10/dist-packages/statsmodels/graphics/tsaplots.py:348: Futur
eWarning: The default method 'yw' can produce PACF values outside of the [-1,1] int
erval. After 0.13, the default will change tounadjusted Yule-Walker ('ywm'). You ca
n use this method now by setting method='ywm'.
  warnings.warn(
<Figure size 1200x600 with 0 Axes>
```



In [ ]:

```
print(df.dtypes)
```

```
Bank                  object
FIPSStateCode          int64
FIPSCountyCode         int64
Tract                float64
MinPer               float64
TraMedY                int64
LocMedY                int64
Income                 int64
CurAreY                int64
UPB                    int64
LTV                  float64
MortDate               int64
AcqDate                int64
Purpose                int64
Product                int64
FedGuar                int64
Term                   int64
AmorTerm               int64
SellType               int64
NumBor                 int64
First                  int64
BoRace                 int64
CoRace                 int64
BoGender               int64
CoGender               int64
BoAge                  int64
CoAge                  int64
Occup                  int64
NumUnits               int64
Rate                 float64
NoteAmount             int64
Front                float64
Back                 float64
BoCreditScore          int64
CoBoCreditScor         int64
PMI                  float64
Self                   int64
PropType              object
ArmIndex               int64
MarginRatePercent      int64
PrepayP               object
BoEth                  int64
CoEth                  int64
HOEPA                  int64
LienStatus             int64
HousePrice           float64
dtype: object
```

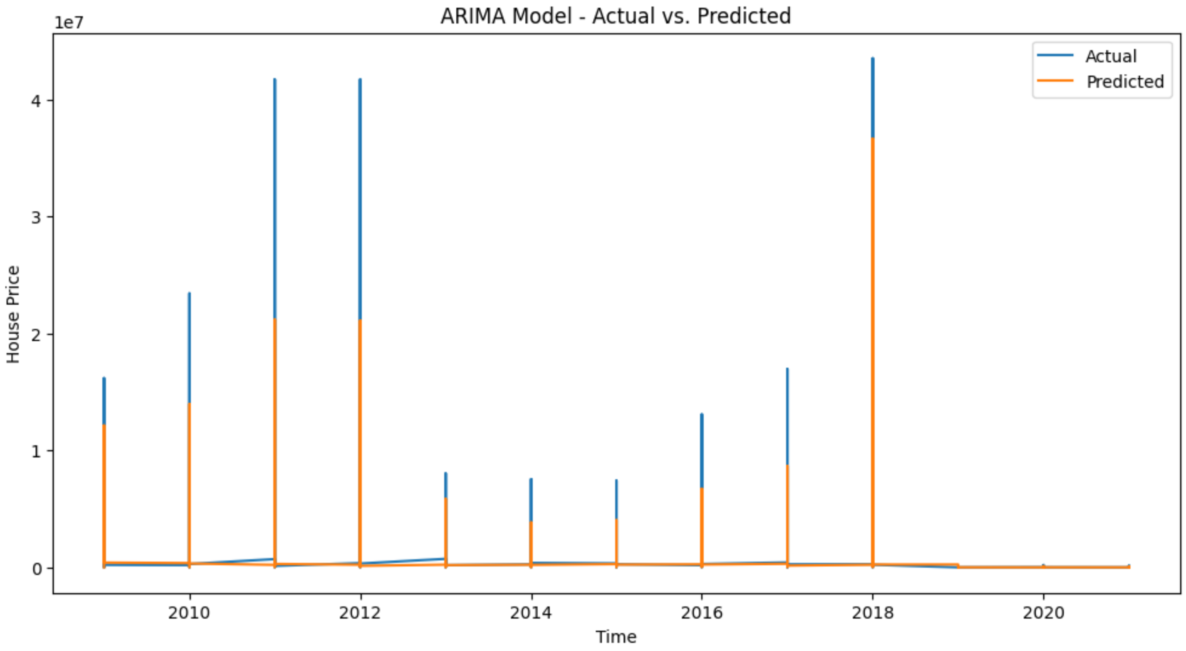
In [ ]:

```
df['PrepayP'] = pd.to_numeric(df['PrepayP'], errors='coerce')
df['PropType'] = pd.to_numeric(df['PropType'], errors='coerce')
df['Bank'] = pd.to_numeric(df['Bank'], errors='coerce')
```

In [ ]:

```
from statsmodels.tsa.arima.model import ARIMA
p=1
d=1
q=0
order = (p, d, q)

model = ARIMA(df['HousePrice'], order=order)
model_fit = model.fit()
predictions = model_fit.predict()
plt.figure(figsize=(12, 6))
plt.plot(df.index, df['HousePrice'], label='Actual')
plt.plot(df.index, predictions, label='Predicted')
plt.title('ARIMA Model - Actual vs. Predicted')
plt.xclabel('Time')
plt.ylabel('House Price')
plt.legend()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: Valu
eWarning: A date index has been provided, but it has no associated frequency inform
ation and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: Valu
eWarning: A date index has been provided, but it has no associated frequency inform
ation and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:471: Valu
eWarning: A date index has been provided, but it has no associated frequency inform
ation and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
```



In [ ]:

```
print(model_fit.summary())
```

```
                               SARIMAX Results                                
==============================================================================
Dep. Variable:             HousePrice   No. Observations:               743816
Model:                 ARIMA(1, 1, 0)   Log Likelihood           -10438874.739
Date:                Sat, 24 Jun 2023   AIC                       20877753.478
Time:                        23:59:21   BIC                       20877776.517
Sample:                             0   HQIC                      20877759.895
                             - 743816                                         
Covariance Type:                  opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.4947   2.59e-05  -1.91e+04      0.000      -0.495      -0.495
sigma2      9.068e+10   4.31e-18    2.1e+28      0.000    9.07e+10    9.07e+10
===================================================================================
Ljung-Box (L1) (Q):               15492.74   Jarque-Bera (JB):     731409110798.95
Prob(Q):                              0.00   Prob(JB):                        0.00
Heteroskedasticity (H):               0.01   Skew:                           26.68
Prob(H) (two-sided):                  0.00   Kurtosis:                     4860.66
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-ste
p).
[2] Covariance matrix is singular or near-singular, with condition number 2.52e+41.
Standard errors may be unstable.
```

In [ ]:

```
residuals = model_fit.resid
aic = model_fit.aic
bic = model_fit.bic
params = model_fit.params

print("AIC:", aic)
print("BIC:", bic)
print("Parameters:", params)
```

```
AIC: 20877753.47807726
BIC: 20877776.51717251
Parameters: ar.L1    -4.947436e-01
sigma2    9.067923e+10
dtype: float64
```