```python
import pandas as pd

# Load the data
file_path = 'Loan 2.csv'
data = pd.read_csv(file_path)

# Display the first few rows of the data
print(data.head())
```

```
  Customer ID               Name Gender  Age  Income (USD) Income
Stability  \
0     C-36995   Frederica Shealy      F   56       1933.05
Low
1     C-33999  America Calderone      M   32       4952.91
Low
2      C-3770      Rosetta Verne      F   65        988.19
High
3     C-26480         Zoe Chitty      F   65           NaN
High
4     C-23459       Afton Venema      F   31       2614.77
Low

  Profession     Type of Employment    Location  Loan Amount Request
(USD)  \
0    Working            Sales staff  Semi-Urban
72809.58
1    Working                    NaN  Semi-Urban
46837.47
2  Pensioner                    NaN  Semi-Urban
45593.04
3  Pensioner                    NaN       Rural
80057.92
4    Working  High skill tech staff  Semi-Urban
113858.89

   ...  Credit Score No. of Defaults Has Active Credit Card  Property
ID  \
0  ...        809.44               0                    NaN
746
1  ...        780.40               0            Unpossessed
608
2  ...        833.15               0            Unpossessed
546
3  ...        832.70               1            Unpossessed
890
4  ...        745.55               1                 Active
715

   Property Age  Property Type Property Location  Co-Applicant  \
0       1933.05              4             Rural             1
```

```
1       4952.91                2              Rural              1
2        988.19                2              Urban              0
3           NaN                2         Semi-Urban              1
4       2614.77                4         Semi-Urban              1

   Property Price  Loan Sanction Amount (USD)
0       119933.46                    54607.18
1        54791.00                    37469.98
2        72440.58                    36474.43
3       121441.51                    56040.54
4       208567.91                    74008.28

[5 rows x 24 columns]
```

```python
# Get basic information about the data
print(data.info())

# Get statistical summary of the data
print(data.describe())

# Check for null values
print(data.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 24 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   Customer ID                    30000 non-null  object
 1   Name                           30000 non-null  object
 2   Gender                         29947 non-null  object
 3   Age                            30000 non-null  int64
 4   Income (USD)                   25424 non-null  float64
 5   Income Stability               28317 non-null  object
 6   Profession                     30000 non-null  object
 7   Type of Employment             22730 non-null  object
 8   Location                       30000 non-null  object
 9   Loan Amount Request (USD)      30000 non-null  float64
 10  Current Loan Expenses (USD)    29828 non-null  float64
 11  Expense Type 1                 30000 non-null  object
 12  Expense Type 2                 30000 non-null  object
 13  Dependents                     27507 non-null  float64
 14  Credit Score                   28297 non-null  float64
 15  No. of Defaults                30000 non-null  int64
 16  Has Active Credit Card         28434 non-null  object
 17  Property ID                    30000 non-null  int64
 18  Property Age                   25150 non-null  float64
 19  Property Type                  30000 non-null  int64
 20  Property Location              29644 non-null  object
 21  Co-Applicant                   30000 non-null  int64
```

```
 22  Property Price               30000 non-null  float64
 23  Loan Sanction Amount (USD)   29660 non-null  float64
dtypes: float64(8), int64(5), object(11)
memory usage: 5.5+ MB
None
                Age  Income (USD)  Loan Amount Request (USD)  \
count  30000.000000  2.542400e+04                30000.000000
mean      40.092300  2.630574e+03                88826.333855
std       16.045129  1.126272e+04                59536.949605
min       18.000000  3.777000e+02                 6048.240000
25%       25.000000  1.650457e+03                41177.755000
50%       40.000000  2.222435e+03                75128.075000
75%       55.000000  3.090593e+03               119964.605000
max       65.000000  1.777460e+06               621497.820000

       Current Loan Expenses (USD)    Dependents  Credit Score  \
count                  29828.000000  27507.000000  28297.000000
mean                     400.936876      2.253027    739.885381
std                      242.545375      0.951162     72.163846
min                     -999.000000      1.000000    580.000000
25%                      247.667500      2.000000    681.880000
50%                      375.205000      2.000000    739.820000
75%                      521.292500      3.000000    799.120000
max                     3840.880000     14.000000    896.260000

       No. of Defaults   Property ID  Property Age  Property Type  \
count     30000.000000  30000.000000  2.515000e+04   30000.000000
mean          0.193933    501.934700  2.631119e+03       2.460067
std           0.395384    288.158086  1.132268e+04       1.118562
min           0.000000      1.000000  3.777000e+02       1.000000
25%           0.000000    251.000000  1.650450e+03       1.000000
50%           0.000000    504.000000  2.223250e+03       2.000000
75%           0.000000    751.000000  3.091408e+03       3.000000
max           1.000000    999.000000  1.777460e+06       4.000000

       Co-Applicant  Property Price  Loan Sanction Amount (USD)
count  30000.000000    3.000000e+04                29660.000000
mean      -4.743867    1.317597e+05                47649.342208
std       74.614593    9.354955e+04                48221.146686
min     -999.000000   -9.990000e+02                 -999.000000
25%        1.000000    6.057216e+04                    0.000000
50%        1.000000    1.099936e+05                35209.395000
75%        1.000000    1.788807e+05                74261.250000
max        1.000000    1.077967e+06               481907.320000
Customer ID                         0
Name                                0
Gender                             53
Age                                 0
Income (USD)                     4576
Income Stability                 1683
```

```
Profession                          0
Type of Employment               7270
Location                            0
Loan Amount Request (USD)           0
Current Loan Expenses (USD)       172
Expense Type 1                      0
Expense Type 2                      0
Dependents                       2493
Credit Score                     1703
No. of Defaults                     0
Has Active Credit Card           1566
Property ID                         0
Property Age                     4850
Property Type                       0
Property Location                 356
Co-Applicant                        0
Property Price                      0
Loan Sanction Amount (USD)        340
dtype: int64
```

```
# Fill null values for numerical columns with median
for column in data.select_dtypes(include=['float64',
'int64']).columns:
    data[column].fillna(data[column].median(), inplace=True)

# Fill null values for categorical columns with mode
for column in data.select_dtypes(include=['object']).columns:
    data[column].fillna(data[column].mode()[0], inplace=True)

# Verify null values are handled
print(data.isnull().sum())
```

```
Customer ID                     0
Name                            0
Gender                          0
Age                             0
Income (USD)                    0
Income Stability                0
Profession                      0
Type of Employment              0
Location                        0
Loan Amount Request (USD)       0
Current Loan Expenses (USD)     0
Expense Type 1                  0
Expense Type 2                  0
Dependents                      0
Credit Score                    0
No. of Defaults                 0
Has Active Credit Card          0
Property ID                     0
```

```
Property Age                    0
Property Type                   0
Property Location               0
Co-Applicant                    0
Property Price                  0
Loan Sanction Amount (USD)      0
dtype: int64

/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/
ipykernel_72723/4099428775.py:3: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/40994
28775.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/40994
28775.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.
```

```
  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/40994
28775.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/40994
28775.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/40994
28775.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/40994
28775.py:3: FutureWarning: A value is trying to be set on a copy of a
```

```
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/40994
28775.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/40994
28775.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/40994
28775.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.
```

```
For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/40994
28775.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/40994
28775.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/40994
28775.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.
```

```
  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/4099428775.py:7: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  data[column].fillna(data[column].mode()[0], inplace=True)
```

```python
# Function to remove outliers using IQR
def remove_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

# Apply the function to each numerical column
for column in data.select_dtypes(include=['float64', 'int64']).columns:
    data = remove_outliers_iqr(data, column)

# Display the data after removing outliers
print(data.describe())
```

```
                Age  Income (USD)  Loan Amount Request (USD)  \
count  16968.000000  16968.000000                16968.000000
mean      39.127416   2162.684262                76320.368761
std       15.688570    697.519946                44747.602671
min       18.000000    377.700000                 6108.050000
25%       24.000000   1670.475000                39116.837500
50%       39.000000   2222.435000                68607.700000
75%       53.000000   2515.915000               105365.145000
max       65.000000   4378.180000               228220.460000

       Current Loan Expenses (USD)    Dependents  Credit Score  \
count                 16968.000000  16968.000000  16968.000000
mean                    365.290167      2.206742    736.771743
```

```
std                    165.428449      0.853187     67.289045
min                     33.760000      1.000000    580.850000
25%                    236.337500      2.000000    684.887500
50%                    352.540000      2.000000    739.820000
75%                    467.462500      3.000000    788.222500
max                    860.380000      4.000000    890.020000
```

```
       No. of Defaults   Property ID  Property Age  Property Type  \
count          16968.0  16968.000000  16968.000000   16968.000000
mean               0.0    502.008545   2162.960506       2.452735
std                0.0    287.087116    692.549032       1.119334
min                0.0      1.000000    377.700000       1.000000
25%                0.0    253.000000   1676.867500       1.000000
50%                0.0    503.000000   2223.250000       2.000000
75%                0.0    750.000000   2503.145000       3.000000
max                0.0    999.000000   4024.690000       4.000000
```

```
       Co-Applicant  Property Price  Loan Sanction Amount (USD)
count       16968.0    16968.000000                16968.000000
mean            1.0   112231.168012                43893.771452
std             0.0    69120.853542                36528.549203
min             1.0     -999.000000                 -999.000000
25%             1.0    56848.710000                15411.222500
50%             1.0    98803.565000                35941.010000
75%             1.0   155474.310000                67754.017500
max             1.0   317198.700000               148035.580000
```

```
# Fill null values for numerical columns with median
for column in data.select_dtypes(include=['float64',
'int64']).columns:
    data[column].fillna(data[column].median(), inplace=True)

# Fill null values for categorical columns with mode
for column in data.select_dtypes(include=['object']).columns:
    data[column].fillna(data[column].mode()[0], inplace=True)

# Verify null values are handled
null_values_after = data.isnull().sum()
null_values_after
```

```
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/
ipykernel_72723/227065762.py:3: FutureWarning: A value is trying to be
set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
```

```
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/22706
5762.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/22706
5762.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/22706
5762.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
```

```
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/22706
5762.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/22706
5762.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/22706
5762.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/22706
5762.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
```

```
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/22706
5762.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/22706
5762.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/22706
5762.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
```

```
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/22706
5762.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/22706
5762.py:3: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].median(), inplace=True)
/var/folders/zt/3ktd1j_n0xn76lhyf8dqmhdm0000gn/T/ipykernel_72723/22706
5762.py:7: FutureWarning: A value is trying to be set on a copy of a
DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  data[column].fillna(data[column].mode()[0], inplace=True)
```

```
Customer ID                     0
Name                            0
Gender                          0
Age                             0
Income (USD)                    0
Income Stability                0
Profession                      0
Type of Employment              0
Location                        0
Loan Amount Request (USD)       0
Current Loan Expenses (USD)     0
Expense Type 1                  0
Expense Type 2                  0
Dependents                      0
Credit Score                    0
No. of Defaults                 0
Has Active Credit Card          0
Property ID                     0
Property Age                    0
Property Type                   0
Property Location               0
Co-Applicant                    0
Property Price                  0
Loan Sanction Amount (USD)      0
dtype: int64
```

```
# Function to remove outliers using IQR
def remove_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] >= lower_bound) & (df[column] <=
upper_bound)]

# Apply the function to each numerical column
for column in data.select_dtypes(include=['float64',
'int64']).columns:
    data = remove_outliers_iqr(data, column)

# Display the data after removing outliers
data_after_outliers = data.describe()
data_after_outliers
```

```
                Age  Income (USD)  Loan Amount Request (USD)  \
count  15890.000000  15890.000000                15890.000000
mean      39.138011   2098.318277                72465.174687
std       15.690879    638.253844                41416.205829
min       18.000000    472.040000                 6108.050000
```

```
25%       24.000000   1642.635000               38449.522500
50%       39.000000   2220.215000               66282.645000
75%       53.000000   2443.127500               99716.700000
max       65.000000   3626.620000              195086.780000

       Current Loan Expenses (USD)    Dependents  Credit Score  \
count                 15890.000000  15890.000000  15890.000000
mean                    351.725171      2.204468    735.915064
std                     155.085601      0.856376     67.248515
min                      33.760000      1.000000    580.850000
25%                     229.947500      2.000000    683.925000
50%                     342.470000      2.000000    739.820000
75%                     449.947500      3.000000    787.212500
max                     787.630000      4.000000    890.020000

       No. of Defaults   Property ID  Property Age  Property Type  \
count          15890.0  15890.000000  15890.000000   15890.000000
mean               0.0    501.888483   2100.010986       2.453870
std                0.0    287.161871    634.762799       1.119143
min                0.0      1.000000    472.040000       1.000000
25%                0.0    252.000000   1647.622500       1.000000
50%                0.0    502.000000   2223.250000       2.000000
75%                0.0    750.750000   2437.792500       3.000000
max                0.0    999.000000   3626.620000       4.000000

       Co-Applicant  Property Price  Loan Sanction Amount (USD)
count       15890.0    15890.000000                15890.000000
mean            1.0   106426.180276                41604.798665
std             0.0    63767.584119                34232.305441
min             1.0     -999.000000                 -999.000000
25%             1.0    55208.450000                14886.685000
50%             1.0    94256.520000                35209.395000
75%             1.0   146937.782500                63991.097500
max             1.0   287329.980000               137638.120000
```

```python
# 'data' is our processed DataFrame
output_file_path = 'processed_loan_data.csv'

# Save the DataFrame to a CSV file
data.to_csv(output_file_path, index=False)

print(f"Processed data has been saved to {output_file_path}")
```

```
Processed data has been saved to processed_loan_data.csv
```