In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats
```

Loading the dataset

In [147…
```
df=pd.read_csv('loan_default.csv')
```

Identification of variables and datatypes

In [3]:
```
df.head(2)
```

Out[3]:

	ID	year	loan_limit	Gender	loan_type	loan_purpose	business_or_commercial	loa
0	24890	2019	cf	Sex Not Available	type1	p1	nob/c	
1	24891	2019	cf	Male	type2	p1	b/c	

In [4]:
```
df.tail(2)
```

Out[4]:

	ID	year	loan_limit	Gender	loan_type	loan_purpose	business_or_commercia
148668	173558	2019	cf	Female	type1	p4	nob/
148669	173559	2019	cf	Female	type1	p3	nob/

In [5]:
```
df.shape
```

Out[5]:
```
(148670, 20)
```

In [6]:
```
df.ndim
```

Out[6]:
```
2
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148670 entries, 0 to 148669
Data columns (total 20 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   ID                        148670 non-null  int64  
 1   year                      148670 non-null  int64  
 2   loan_limit                145326 non-null  object 
 3   Gender                    148670 non-null  object 
 4   loan_type                 148670 non-null  object 
 5   loan_purpose              148536 non-null  object 
 6   business_or_commercial    148670 non-null  object 
 7   loan_amount               148670 non-null  int64  
 8   rate_of_interest          112231 non-null  float64
 9   Upfront_charges           109028 non-null  float64
 10  property_value            133572 non-null  float64
 11  occupancy_type            148670 non-null  object 
 12  income                    139520 non-null  float64
 13  credit_type               148670 non-null  object 
 14  Credit_Score              148670 non-null  int64  
 15  co-applicant_credit_type  148670 non-null  object 
 16  age                       148470 non-null  object 
 17  LTV                       133572 non-null  float64
 18  Region                    148670 non-null  object 
 19  Status                    148670 non-null  int64  
dtypes: float64(5), int64(5), object(10)
memory usage: 22.7+ MB
```

In [8]:
```
df.columns
```

Out[8]:
```
Index(['ID', 'year', 'loan_limit', 'Gender', 'loan_type', 'loan_purpose',
       'business_or_commercial', 'loan_amount', 'rate_of_interest',
       'Upfront_charges', 'property_value', 'occupancy_type', 'income',
       'credit_type', 'Credit_Score', 'co-applicant_credit_type', 'age', 'LTV',
       'Region', 'Status'],
      dtype='object')
```

In [9]:
```
df.dtypes
```

Out[9]:
```
ID                          int64
year                        int64
loan_limit                 object
Gender                     object
loan_type                  object
loan_purpose               object
business_or_commercial     object
loan_amount                 int64
rate_of_interest           float64
Upfront_charges            float64
property_value             float64
occupancy_type              object
income                     float64
credit_type                 object
Credit_Score                int64
co-applicant_credit_type    object
age                         object
LTV                        float64
Region                      object
Status                      int64
dtype: object
```

Analyzing basic metrics

In [10]:
```
df.describe(include='all')
```

Out[10]:

	ID	year	loan_limit	Gender	loan_type	loan_purpose	business_or_
count	148670.000000	148670.0	145326	148670	148670	148536	
unique	NaN	NaN	2	4	3	4	
top	NaN	NaN	cf	Male	type1	p3	
freq	NaN	NaN	135348	42346	113173	55934	
mean	99224.500000	2019.0	NaN	NaN	NaN	NaN	
std	42917.476598	0.0	NaN	NaN	NaN	NaN	
min	24890.000000	2019.0	NaN	NaN	NaN	NaN	
25%	62057.250000	2019.0	NaN	NaN	NaN	NaN	
50%	99224.500000	2019.0	NaN	NaN	NaN	NaN	
75%	136391.750000	2019.0	NaN	NaN	NaN	NaN	
max	173559.000000	2019.0	NaN	NaN	NaN	NaN	

In [11]:
```
df.describe(include=[np.number]).T
```

Out[11]:

	count	mean	std	min	25%	
ID	148670.0	99224.500000	42917.476598	24890.000000	62057.25000	992
year	148670.0	2019.000000	0.000000	2019.000000	2019.00000	20
loan_amount	148670.0	331117.743997	183909.310127	16500.000000	196500.00000	2965
rate_of_interest	112231.0	4.045476	0.561391	0.000000	3.62500	
Upfront_charges	109028.0	3224.996127	3251.121510	0.000000	581.49000	25
property_value	133572.0	497893.465696	359935.315562	8000.000000	268000.00000	4180
income	139520.0	6957.338876	6496.586382	0.000000	3720.00000	57
Credit_Score	148670.0	699.789103	115.875857	500.000000	599.00000	6
LTV	133572.0	72.746457	39.967603	0.967478	60.47486	
Status	148670.0	0.246445	0.430942	0.000000	0.00000	

In [12]:
```
df.describe(include=['object']).T
```

Out[12]:

	count	unique	top	freq
loan_limit	145326	2	cf	135348
Gender	148670	4	Male	42346
loan_type	148670	3	type1	113173
loan_purpose	148536	4	p3	55934
business_or_commercial	148670	2	nob/c	127908
occupancy_type	148670	3	pr	138201
credit_type	148670	4	CIB	48152
co-applicant_credit_type	148670	2	CIB	74392
age	148470	7	45-54	34720
Region	148670	4	North	74722

Identification of null values and duplicates

In [13]:
```
round((df.isna().sum()/len(df))*100,2)
```

Out[13]:
```
ID                          0.00
year                        0.00
loan_limit                  2.25
Gender                      0.00
loan_type                   0.00
loan_purpose                0.09
business_or_commercial      0.00
loan_amount                 0.00
rate_of_interest           24.51
Upfront_charges            26.66
property_value             10.16
occupancy_type              0.00
income                      6.15
credit_type                 0.00
Credit_Score                0.00
co-applicant_credit_type    0.00
age                         0.13
LTV                        10.16
Region                      0.00
Status                      0.00
dtype: float64
```

In [14]:
```
np.any(df.duplicated())
```

Out[14]:
```
False
```

Datatype Conversion, Memory optimization and Non-Graphical Analysis

In [146…
```
def column_details(df,column):
    print("Details of",column,"column")
    print("\nDataType: ",df[column].dtype)
    print("\nNumber of Unique Values: ",df[column].nunique())
    print("\nDistribution of column:\n")
    print(df[column].value_counts())

    count_null = df[column].isnull().sum()
    if count_null==0:
        print("\nThere are no null values")
    elif count_null>0:
        print("\nThere are ",count_null," null values")

def convert_dtype(df,column,dtype):
    print('Memory usage of',column, 'column : ', df[column].memory_usage())
    df[column] = df[column].astype(dtype)
    print('Updated Memory usage of ',column,'column : ', df[column].memory_usage())


def fill_with_mode(df,column):
    mode_value = df[column].mode()[0]  # Calculate the mode
    df[column] = df[column].fillna(mode_value)
    print("\nNo. of missing values after filling with column mode:",df[column].isnu

def outliers_treatment(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
```

```
    IQR = Q3 - Q1
    median_value = df[column].median()
    df[column] = df[column].apply(lambda x: median_value if (x < Q1 - 1.5 * IQR or

def count_plot(df,column):
    plt.style.use('dark_background')
    sns.countplot(x=df[column],data=df,color="orange")
    plt.xlabel(column, color='white')
    plt.ylabel("Frequency", color='white')
    plt.title(f'Distribution of {column}', color='white')
    plt.xticks(color='white')
    plt.yticks(color='white')
    plt.show()

def donut_plot(df,column):
    plt.style.use('dark_background')
    wedges, texts, autotexts = plt.pie(df[column].value_counts(), autopct='%1.1f%%'
    plt.setp(autotexts, size=8)
    plt.title(f'Distribution of {column}', color='white')
    plt.axis('equal')
    labels = df[column].value_counts().index
    plt.legend(labels, title=column, loc='upper right')
    plt.show()

def displot(df,column):
    plt.style.use('dark_background')
    sns.displot(df[column], kde=True, color="orange", rug=True, bins=6)
    plt.xlabel(column, color='white')
    plt.ylabel("Frequency", color='white')
    plt.title(f'Distribution of {column}', color='white')
    plt.xticks(color='white')
    plt.yticks(color='white')
    plt.show()

def violin_plot(df,column):
    plt.style.use('dark_background')
    sns.violinplot(df[column], color="orange",inner='quartile' )
    plt.xlabel(column, fontsize=12, color='white')
    plt.ylabel("Density", fontsize=12, color='white')
    plt.title(f'Distribution of {column}', fontsize=14, color='white')
    plt.xticks(fontsize=10, color='white')
    plt.yticks(fontsize=10, color='white')
    plt.show()
```

In [145…

```
def box_plot(df,column):
    plt.style.use('dark_background')
    sns.boxplot(df[column], color="orange" )
    plt.xlabel(column, fontsize=12, color='white')
    plt.ylabel("Density", fontsize=12, color='white')
    plt.title(f'Distribution of {column}', fontsize=14, color='white')
    plt.xticks(fontsize=10, color='white')
    plt.yticks(fontsize=10, color='white')
    plt.show()
```

In [17]:
```
column_details(df,'ID')
convert_dtype(df,'ID','int32')
```

```
Details of ID column

DataType:  int64

Number of Unique Values:  148670

Distribution of column:

ID
24890     1
123979    1
123999    1
124000    1
124001    1
         ..
74447     1
74448     1
74449     1
74450     1
173559    1
Name: count, Length: 148670, dtype: int64

There are no null values
Memory usage of ID column :  1189488
Updated Memory usage of  ID column :  594808
```

In [18]:
```
column_details(df,'year')
convert_dtype(df,'year','int16')
```

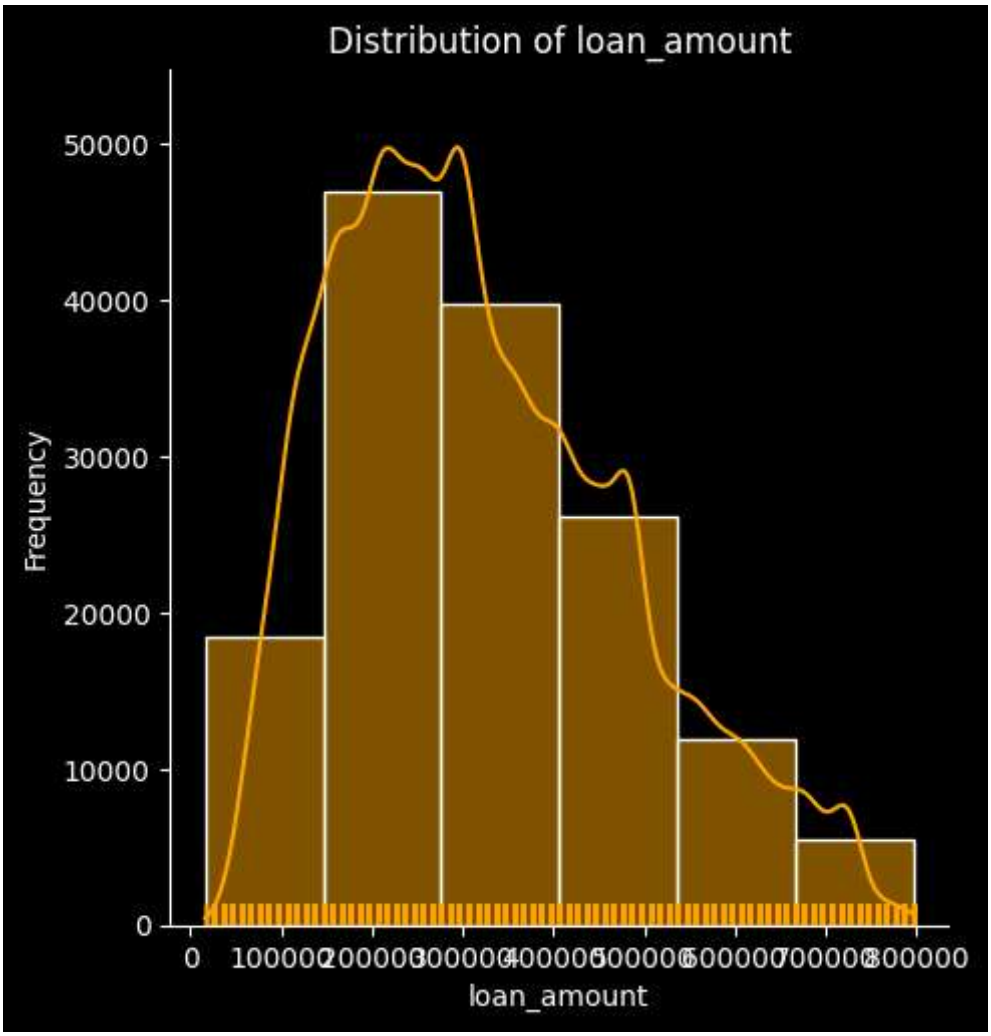
```
Details of year column

DataType:  int64

Number of Unique Values:  1

Distribution of column:

year
2019    148670
Name: count, dtype: int64

There are no null values
Memory usage of year column :  1189488
Updated Memory usage of  year column :  297468
```

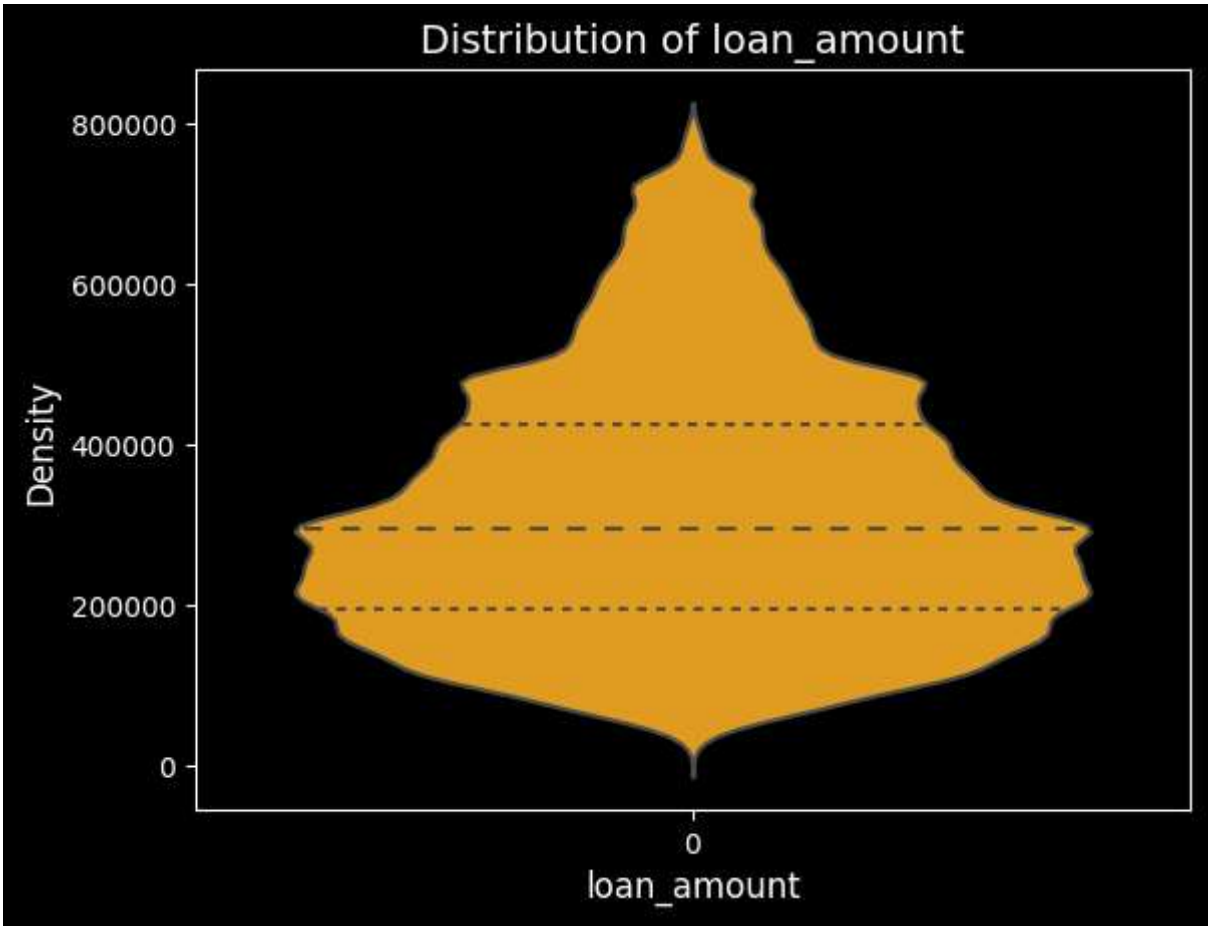
In [148…
```
column_details(df,'loan_amount')
outliers_treatment(df,'loan_amount')
fill_with_mode(df,'loan_amount')
convert_dtype(df,'loan_amount','int32')
```

```
Details of loan_amount column

DataType:  int64

Number of Unique Values:  211

Distribution of column:

loan_amount
206500     4610
256500     4079
156500     3967
226500     3944
486500     3819
           ...
2206500       1
1746500       1
2396500       1
3576500       1
1956500       1
Name: count, Length: 211, dtype: int64

There are no null values

No. of missing values after filling with column mode: 0
Memory usage of loan_amount column :  1189488
Updated Memory usage of  loan_amount column :  594808
```

In [149…

```
displot(df,'loan_amount')
violin_plot(df,'loan_amount')
```

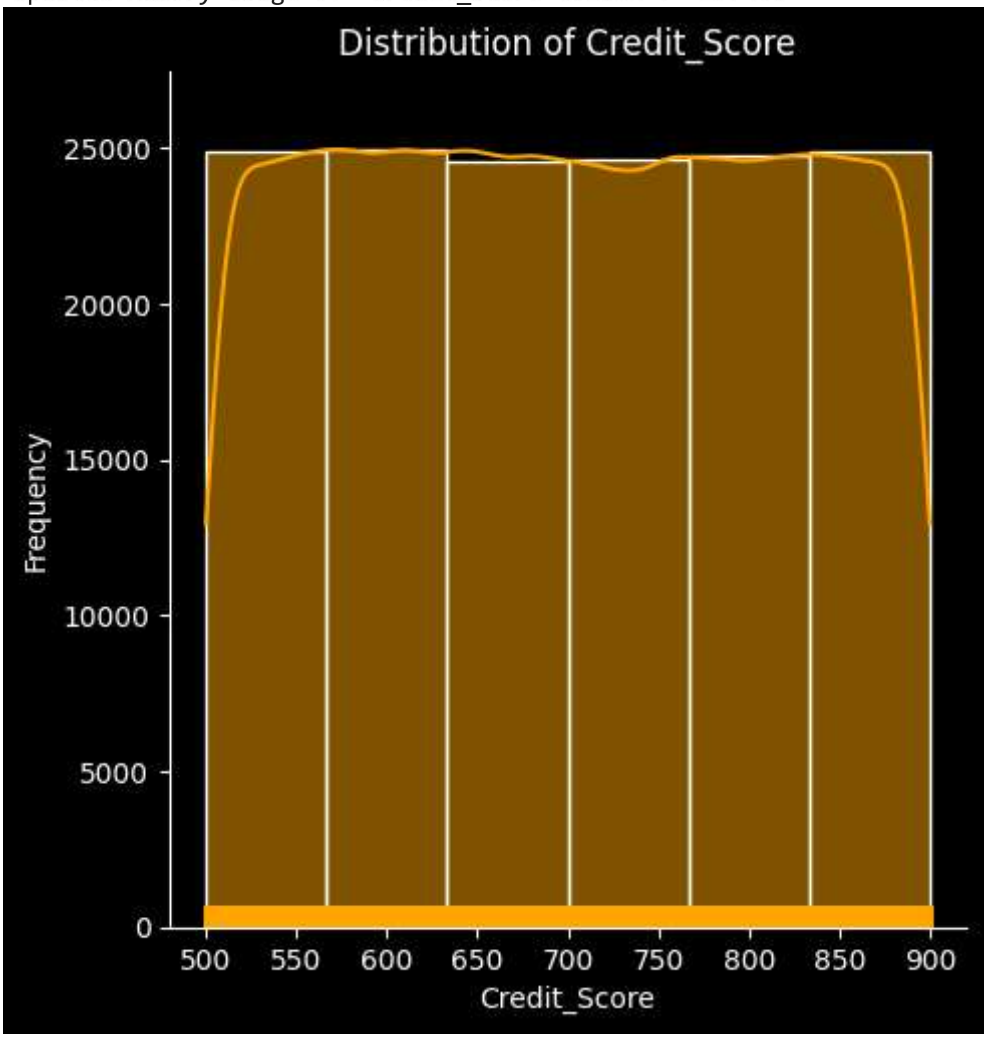

Distribution of loan_amount
Frequency
50000
40000
30000
20000
10000
0
0
100000
200000
300000
400000
500000
600000
700000
800000
loan_amount

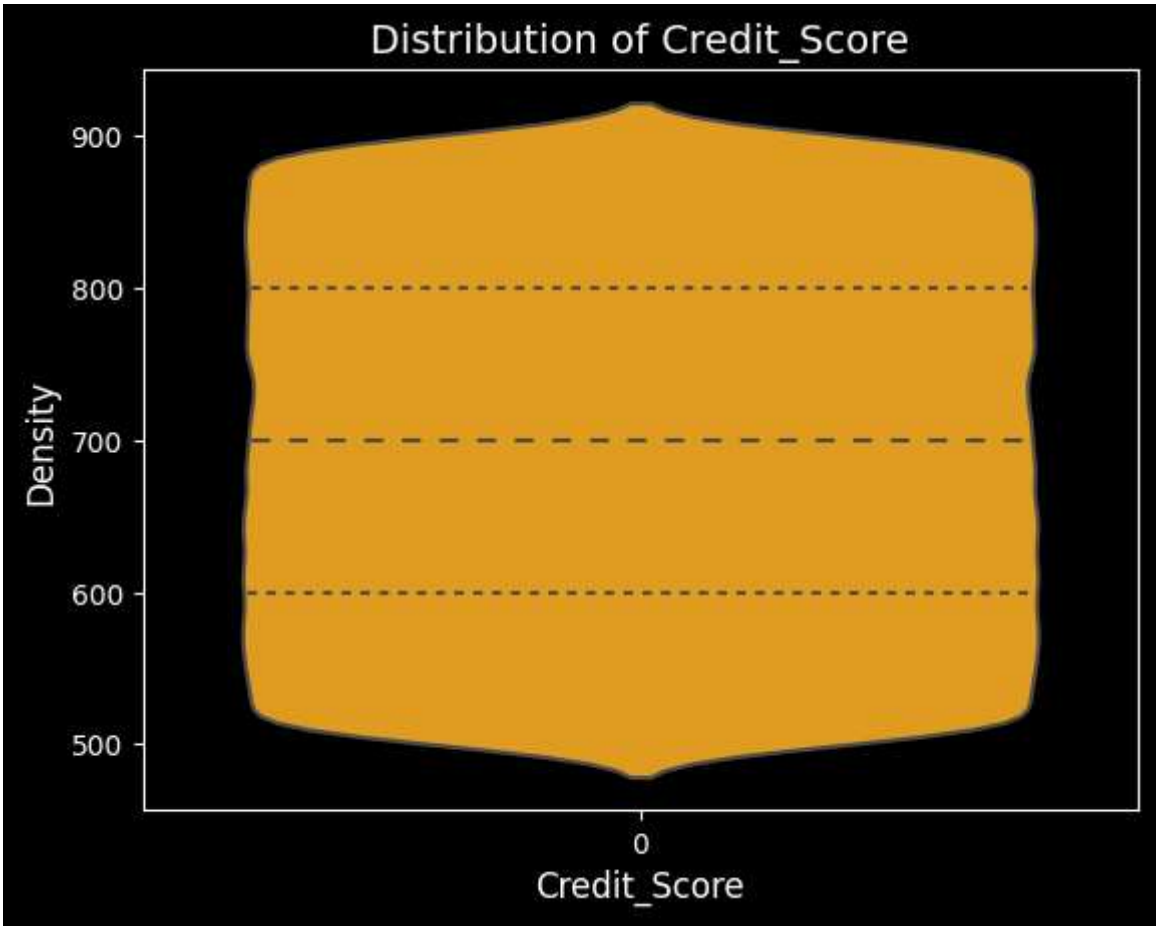


In [150…

```
column_details(df,'Credit_Score')
convert_dtype(df,'Credit_Score','int16')
displot(df,'Credit_Score')
violin_plot(df,'Credit_Score')
```

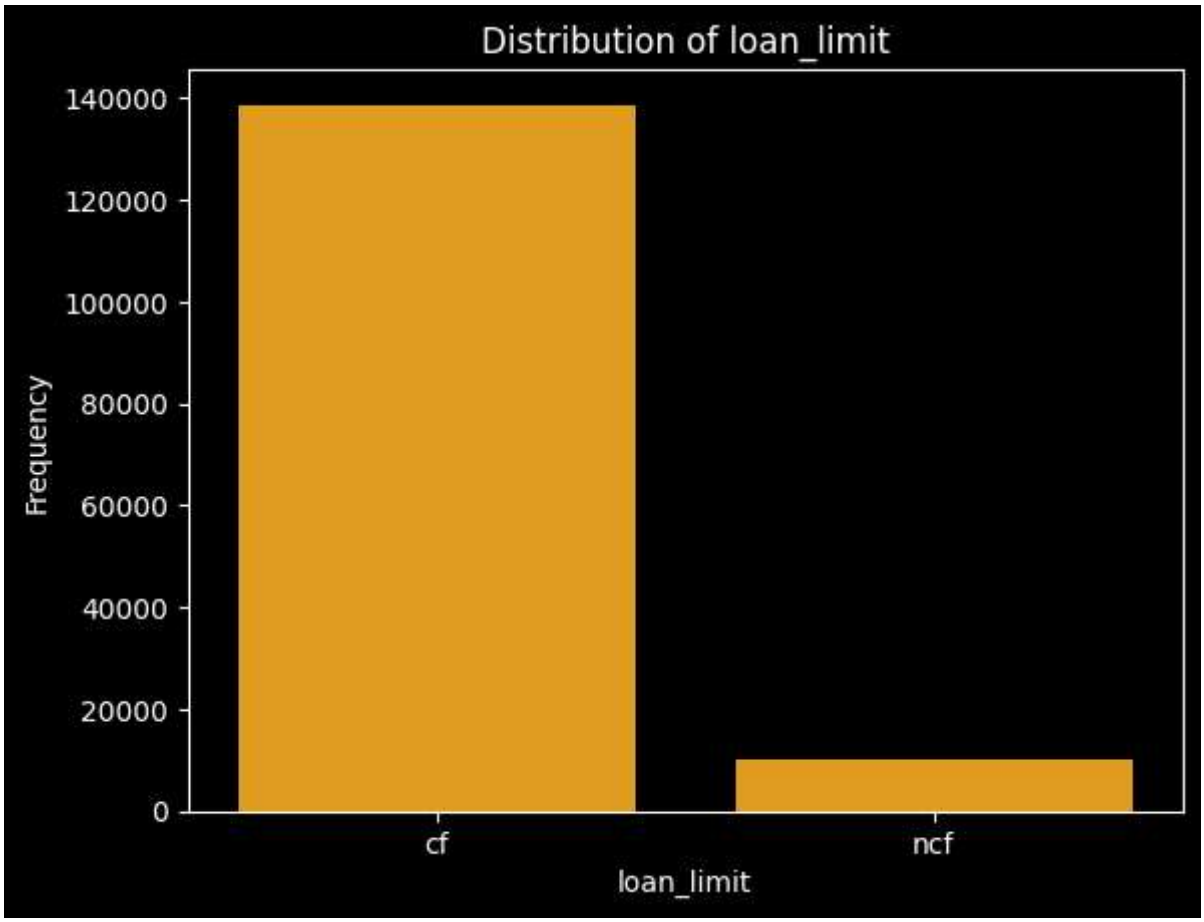
```
Details of Credit_Score column

DataType:  int64

Number of Unique Values:  401

Distribution of column:

Credit_Score
763    415
867    413
639    411
581    408
554    407
      ...
745    330
573    330
743    327
748    324
559    321
Name: count, Length: 401, dtype: int64

There are no null values
Memory usage of Credit_Score column :  1189488
Updated Memory usage of  Credit_Score column :  297468
```

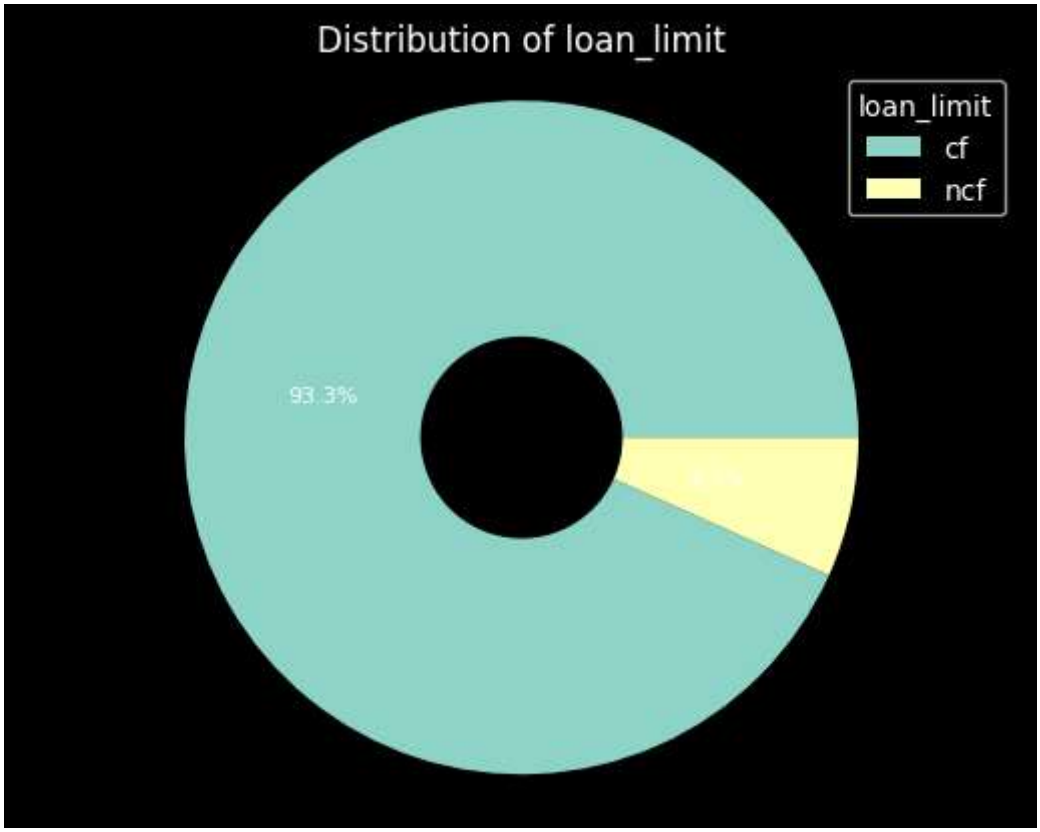




In [151…
```
column_details(df,'loan_limit')
convert_dtype(df,'loan_limit','category')
fill_with_mode(df,'loan_limit')
count_plot(df,'loan_limit')
donut_plot(df,'loan_limit')
```

```
Details of loan_limit column

DataType:  object

Number of Unique Values:  2

Distribution of column:

loan_limit
cf     135348
ncf      9978
Name: count, dtype: int64

There are  3344  null values
Memory usage of loan_limit column :  1189488
Updated Memory usage of  loan_limit column :  148922

No. of missing values after filling with column mode: 0
```



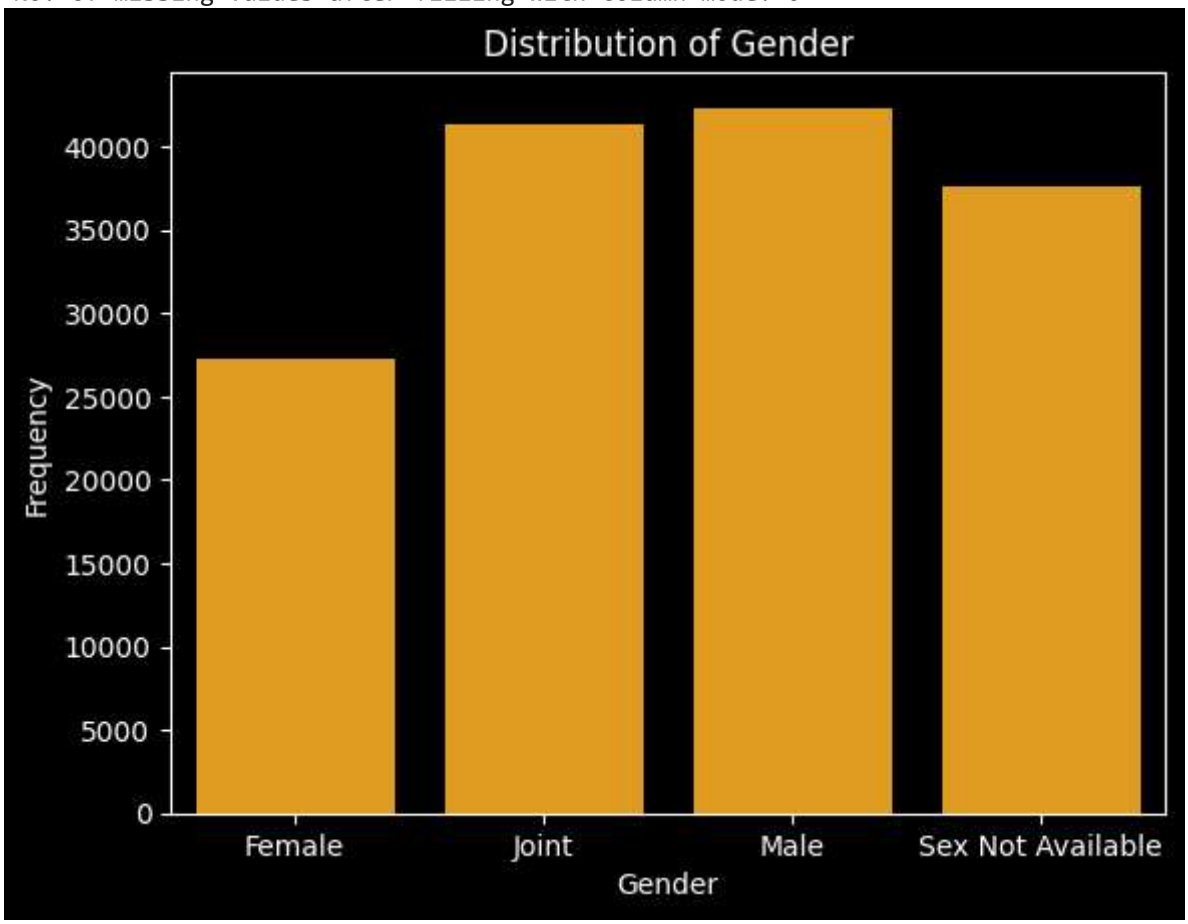


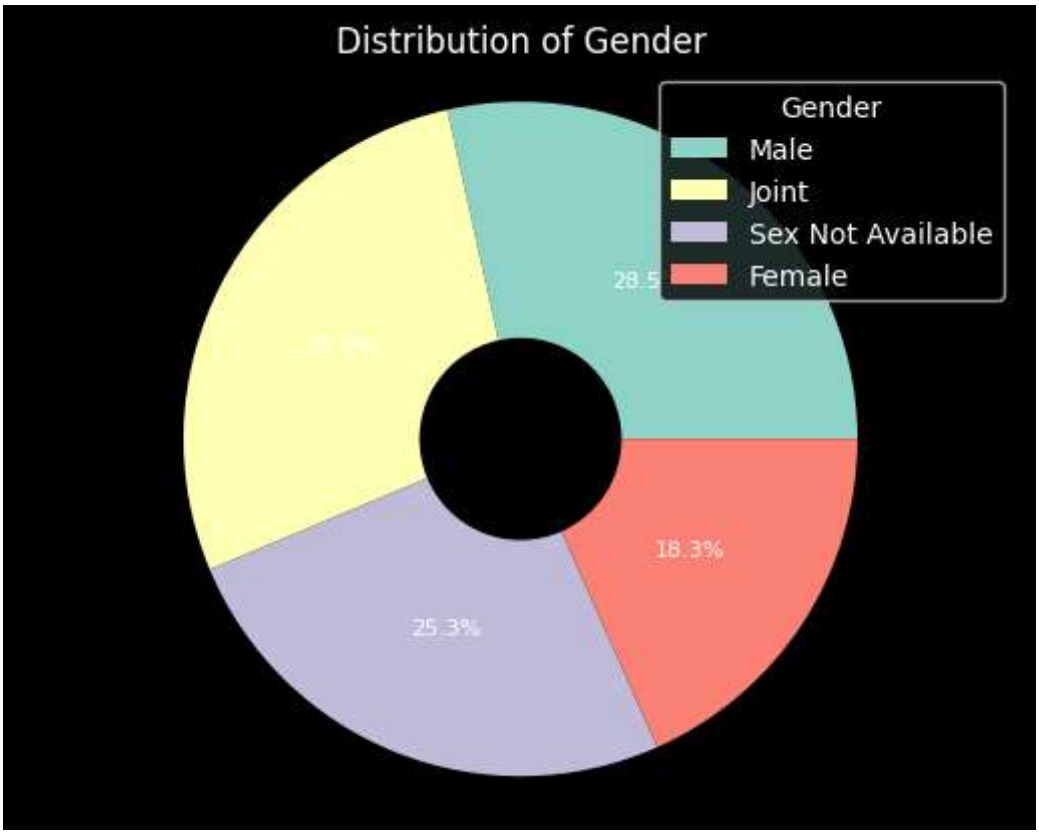
In [152…

```
column_details(df,'Gender')
convert_dtype(df,'Gender','category')
fill_with_mode(df,'Gender')
```

```
count_plot(df,'Gender')
donut_plot(df,'Gender')
```

```
Details of Gender column

DataType:  object

Number of Unique Values:  4

Distribution of column:

Gender
Male                42346
Joint               41399
Sex Not Available   37659
Female              27266
Name: count, dtype: int64

There are no null values
Memory usage of Gender column :  1189488
Updated Memory usage of  Gender column :  149002

No. of missing values after filling with column mode: 0
```

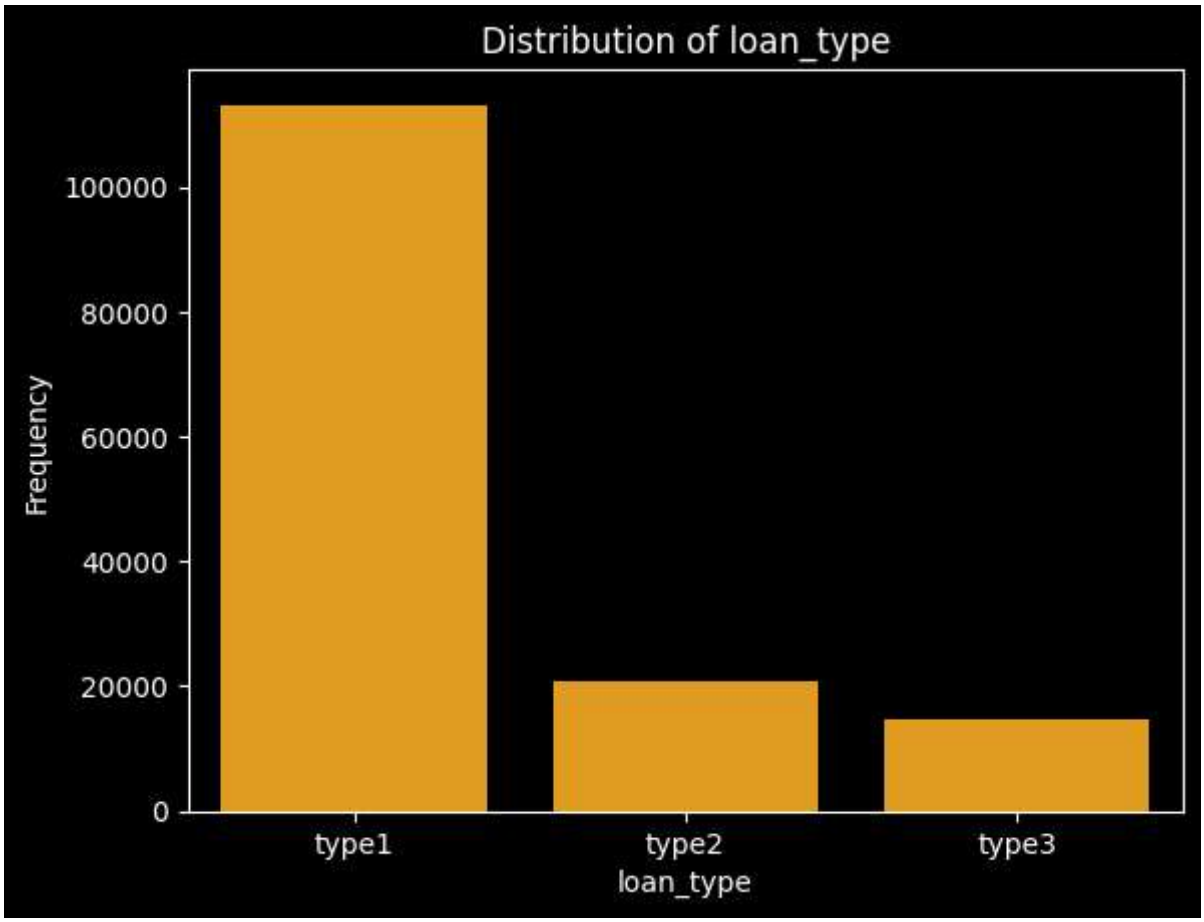


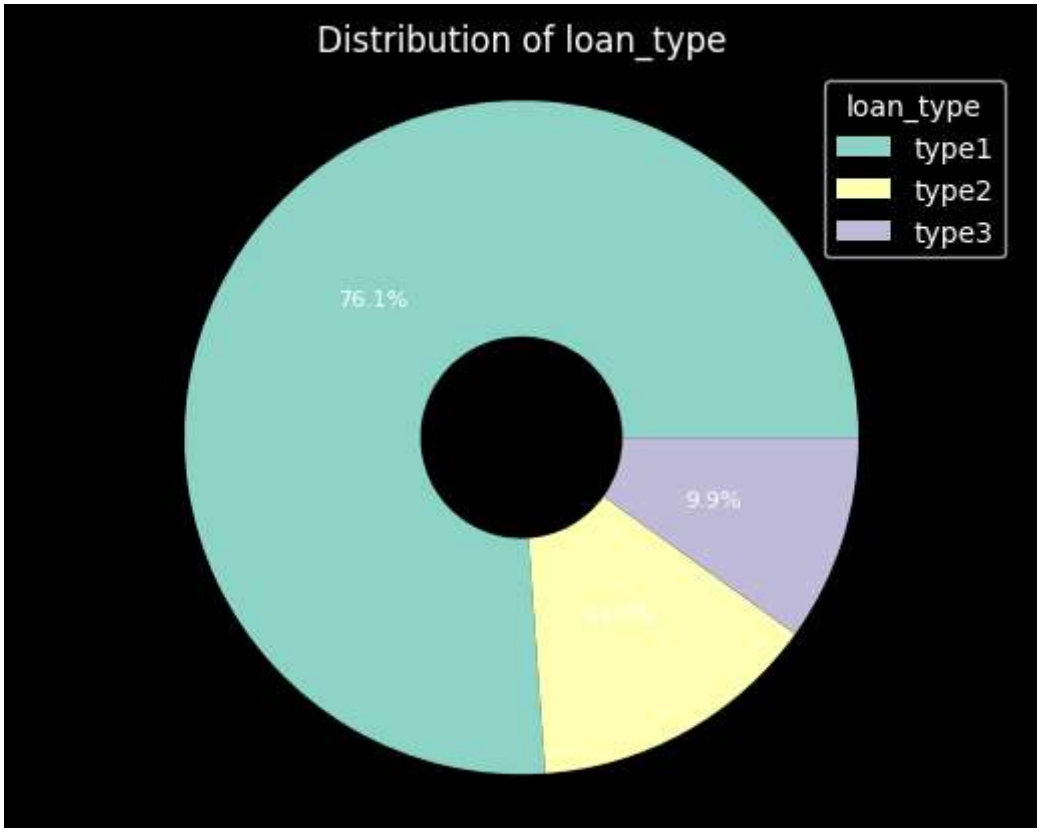


In [153…

```
column_details(df,'loan_type')
convert_dtype(df,'loan_type','category')
count_plot(df,'loan_type')
donut_plot(df,'loan_type')
```

```
Details of loan_type column

DataType:  object

Number of Unique Values:  3

Distribution of column:

loan_type
type1    113173
type2     20762
type3     14735
Name: count, dtype: int64

There are no null values
Memory usage of loan_type column :  1189488
Updated Memory usage of  loan_type column :  148930
```



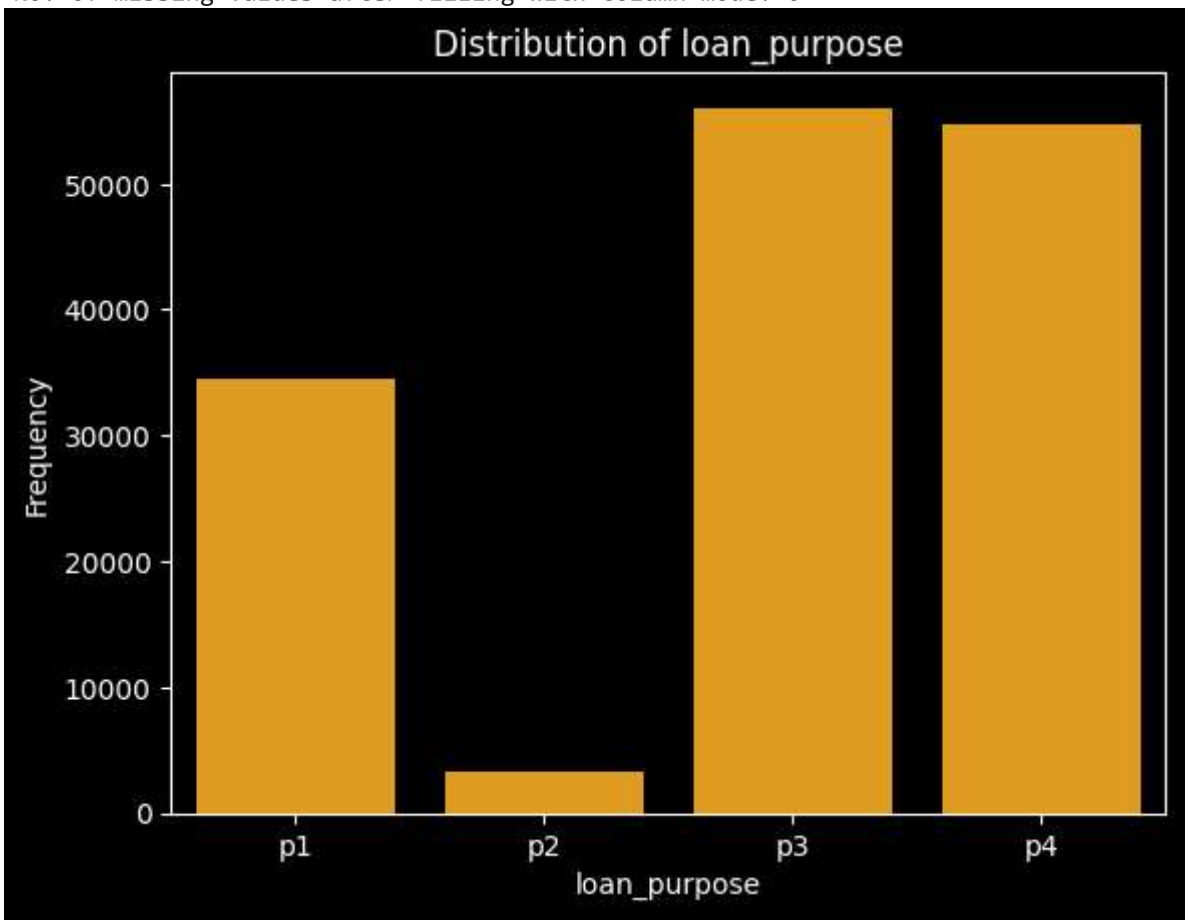


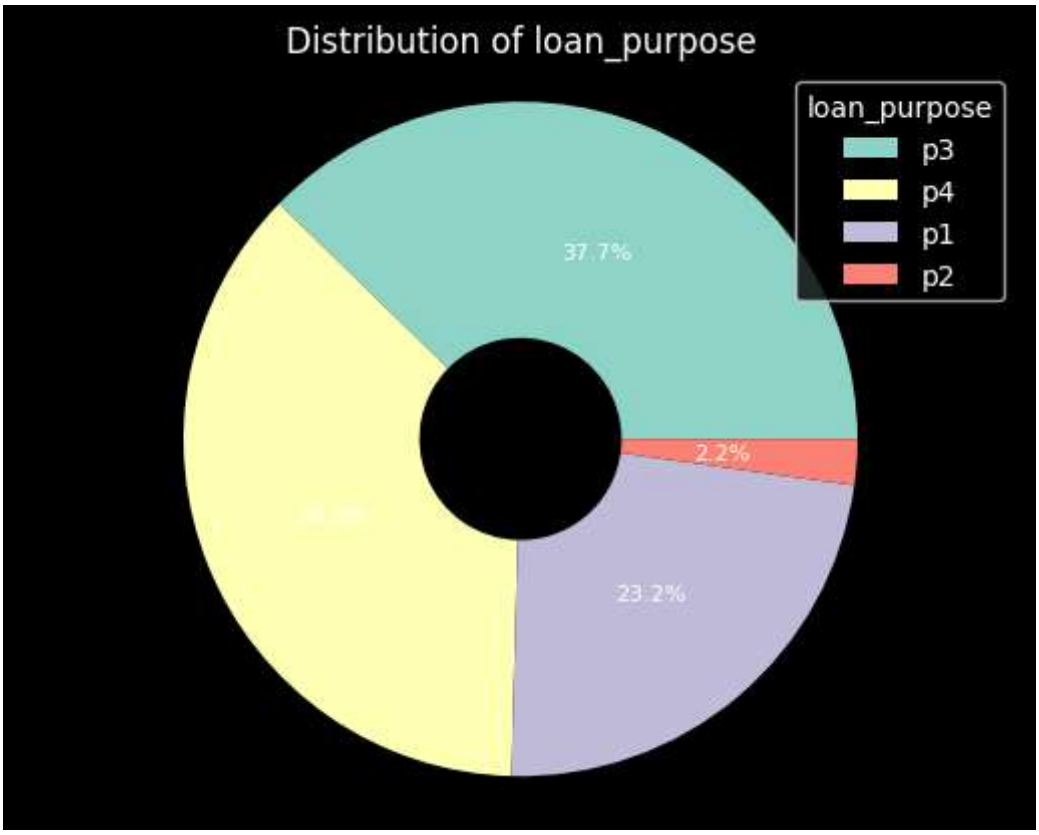
In [154…

```
column_details(df,'loan_purpose')
convert_dtype(df,'loan_purpose','category')
fill_with_mode(df,'loan_purpose')
```

```
count_plot(df,'loan_purpose')
donut_plot(df,'loan_purpose')
```

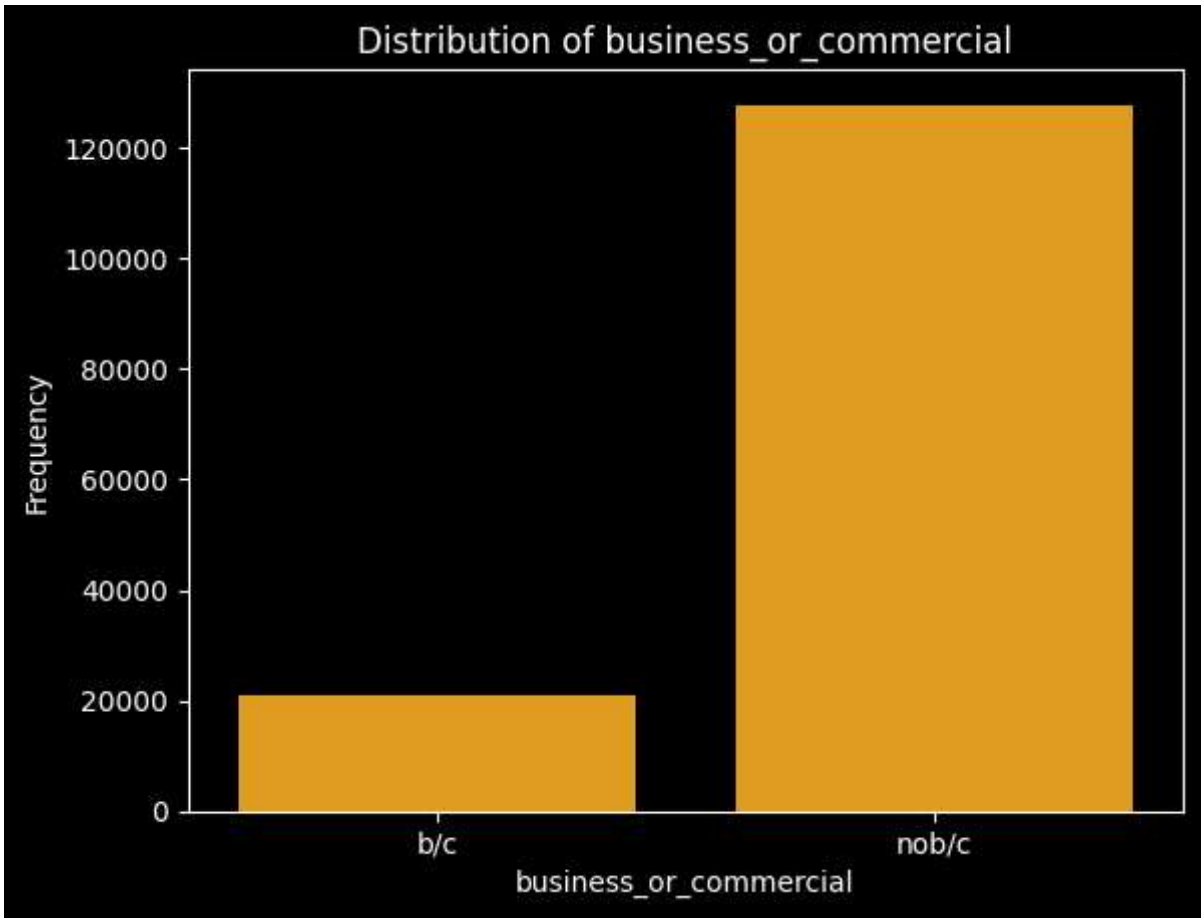
```
Details of loan_purpose column

DataType:  object

Number of Unique Values:  4

Distribution of column:

loan_purpose
p3    55934
p4    54799
p1    34529
p2     3274
Name: count, dtype: int64

There are  134  null values
Memory usage of loan_purpose column :  1189488
Updated Memory usage of  loan_purpose column :  149002

No. of missing values after filling with column mode: 0
```

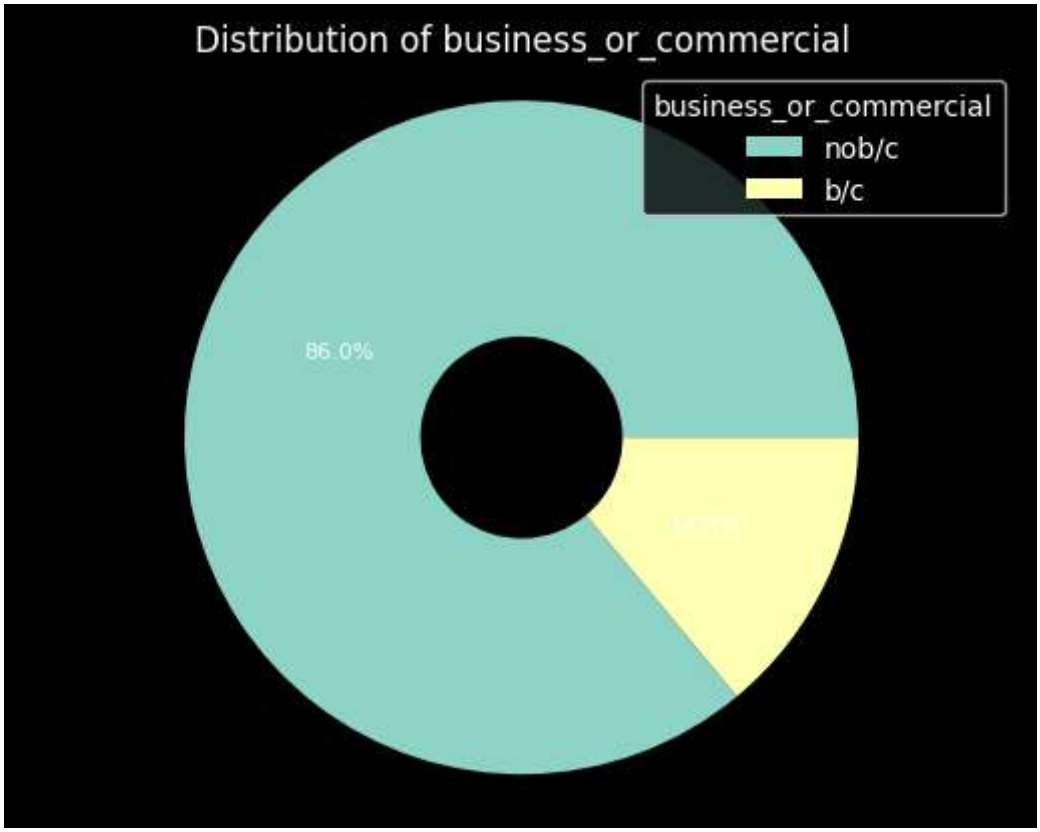




In [155…
```
column_details(df,'business_or_commercial')
convert_dtype(df,'business_or_commercial','category')
count_plot(df,'business_or_commercial')
donut_plot(df,'business_or_commercial')
```

```
Details of business_or_commercial column

DataType:  object

Number of Unique Values:  2

Distribution of column:

business_or_commercial
nob/c    127908
b/c       20762
Name: count, dtype: int64

There are no null values
Memory usage of business_or_commercial column :  1189488
Updated Memory usage of  business_or_commercial column :  148922
```

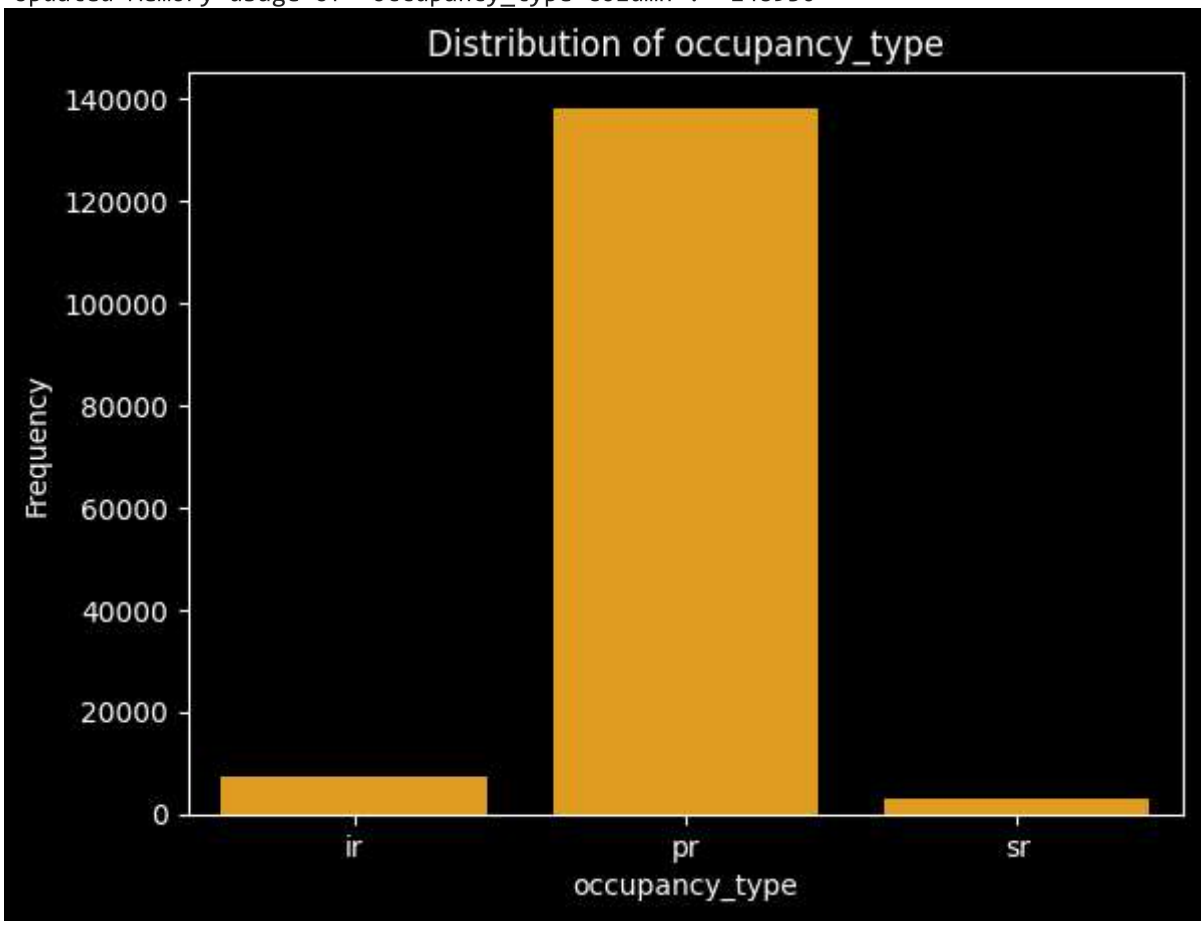


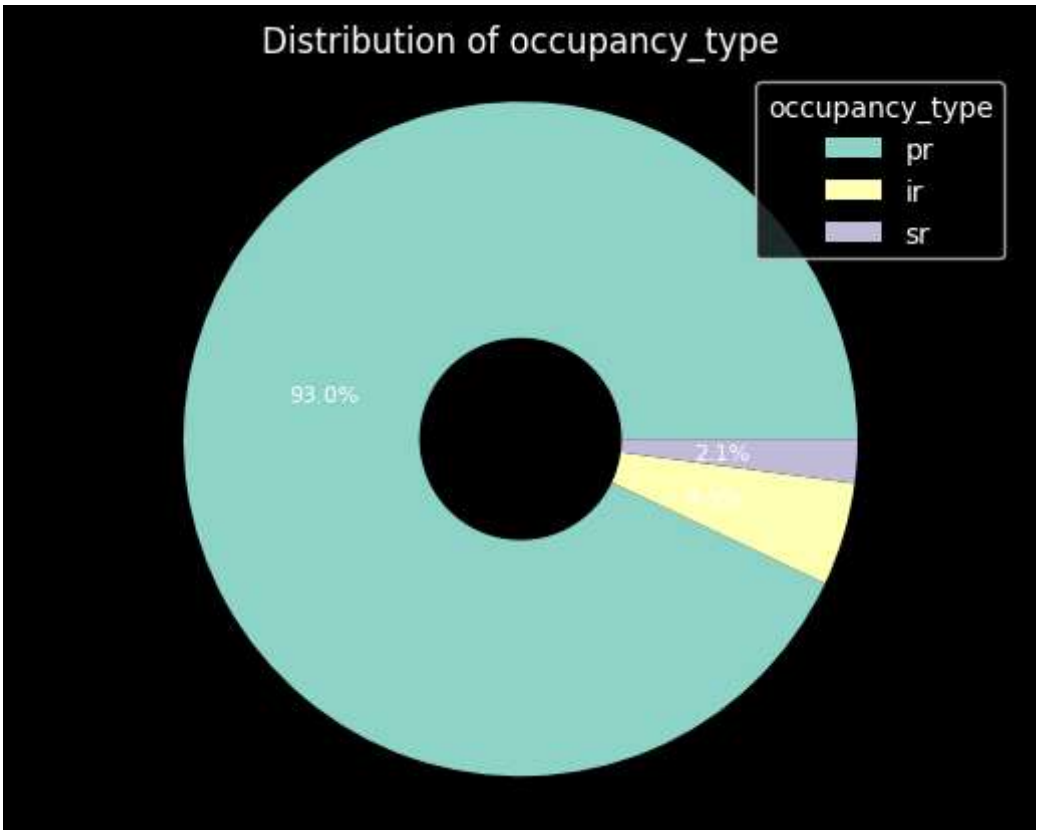


In [156…
```
column_details(df,'occupancy_type')
convert_dtype(df,'occupancy_type','category')
```

```
count_plot(df,'occupancy_type')
donut_plot(df,'occupancy_type')
```

```
Details of occupancy_type column

DataType:  object

Number of Unique Values:  3

Distribution of column:

occupancy_type
pr    138201
ir      7340
sr      3129
Name: count, dtype: int64

There are no null values
Memory usage of occupancy_type column :  1189488
Updated Memory usage of  occupancy_type column :  148930
```

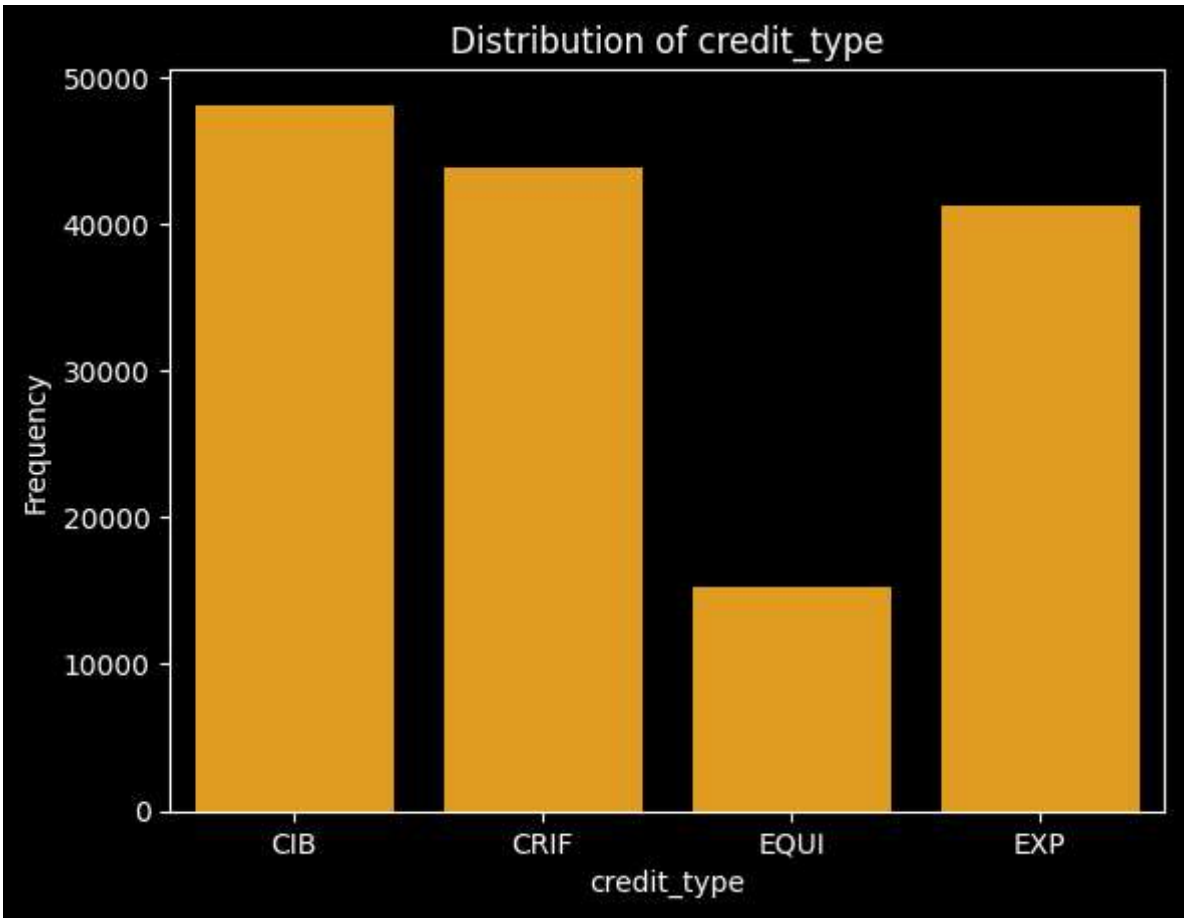


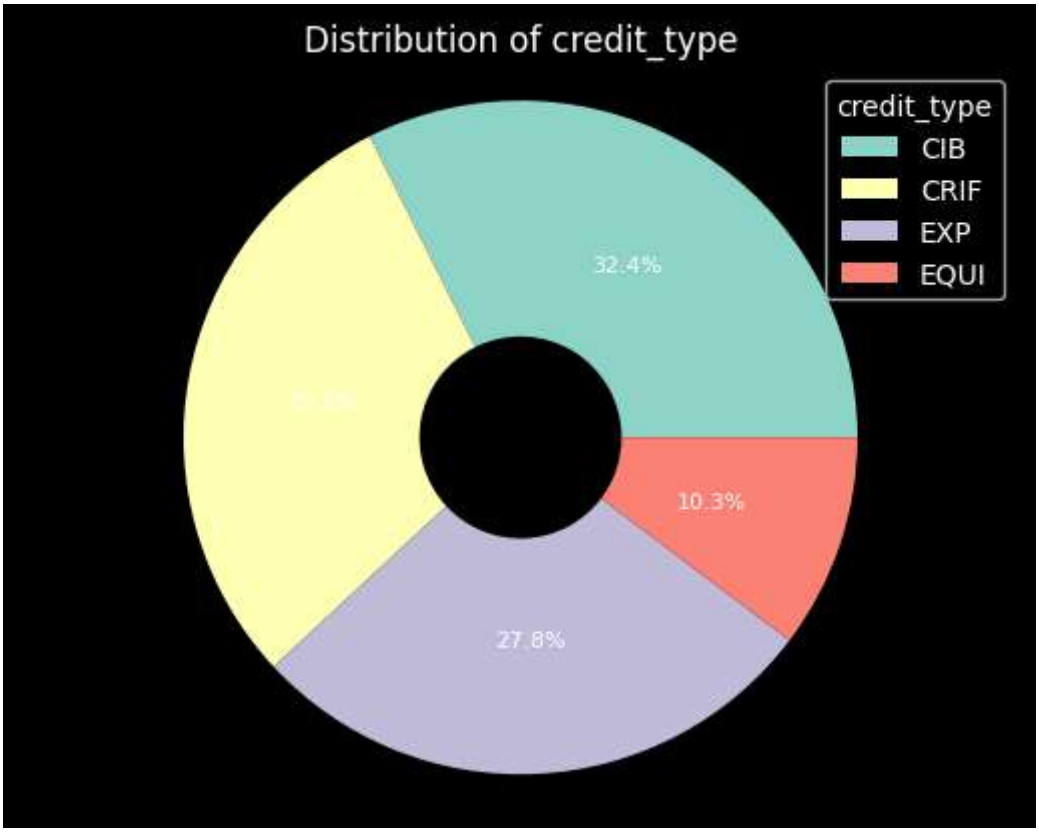


In [157…

```
column_details(df,'credit_type')
convert_dtype(df,'credit_type','category')
count_plot(df,'credit_type')
donut_plot(df,'credit_type')
```

```
Details of credit_type column

DataType:  object

Number of Unique Values:  4

Distribution of column:

credit_type
CIB     48152
CRIF    43901
EXP     41319
EQUI    15298
Name: count, dtype: int64

There are no null values
Memory usage of credit_type column :  1189488
Updated Memory usage of  credit_type column :  149002
```



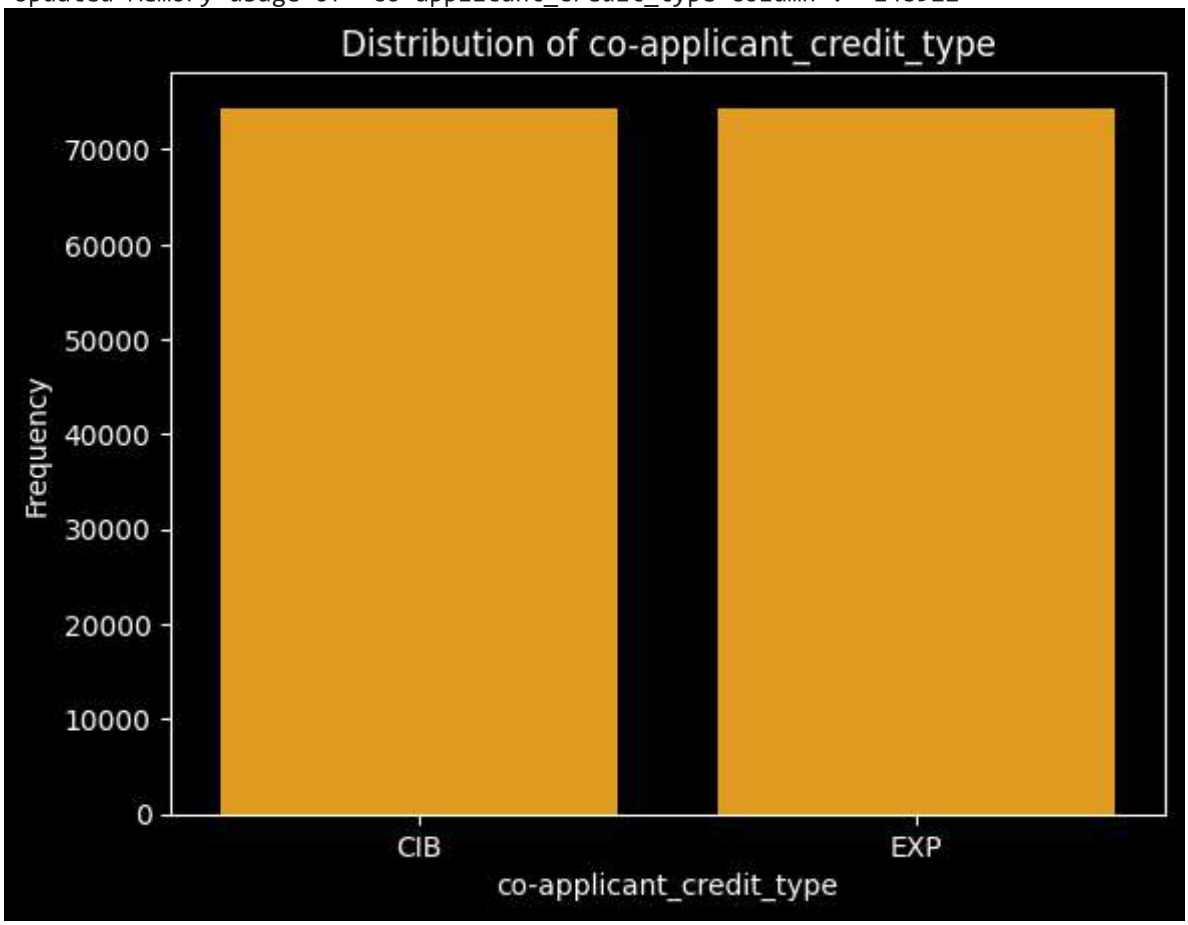


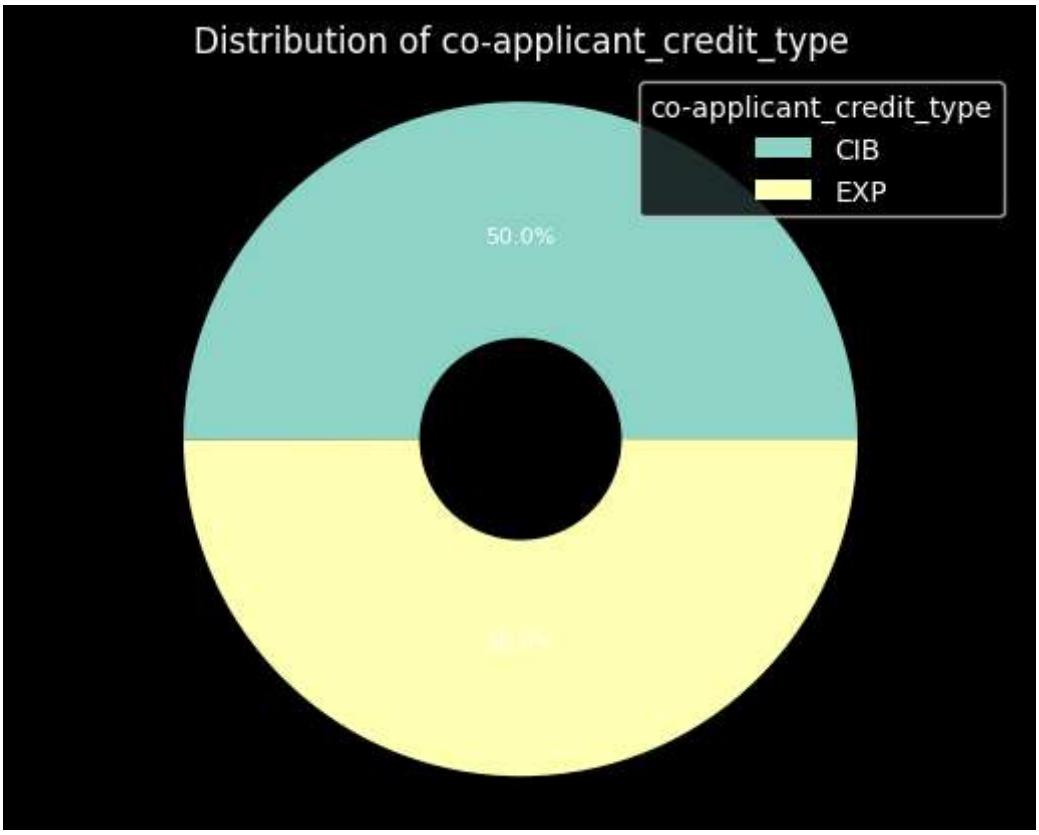
In [158…

```
column_details(df,'co-applicant_credit_type')
convert_dtype(df,'co-applicant_credit_type','category')
```

```
count_plot(df,'co-applicant_credit_type')
donut_plot(df,'co-applicant_credit_type')
```

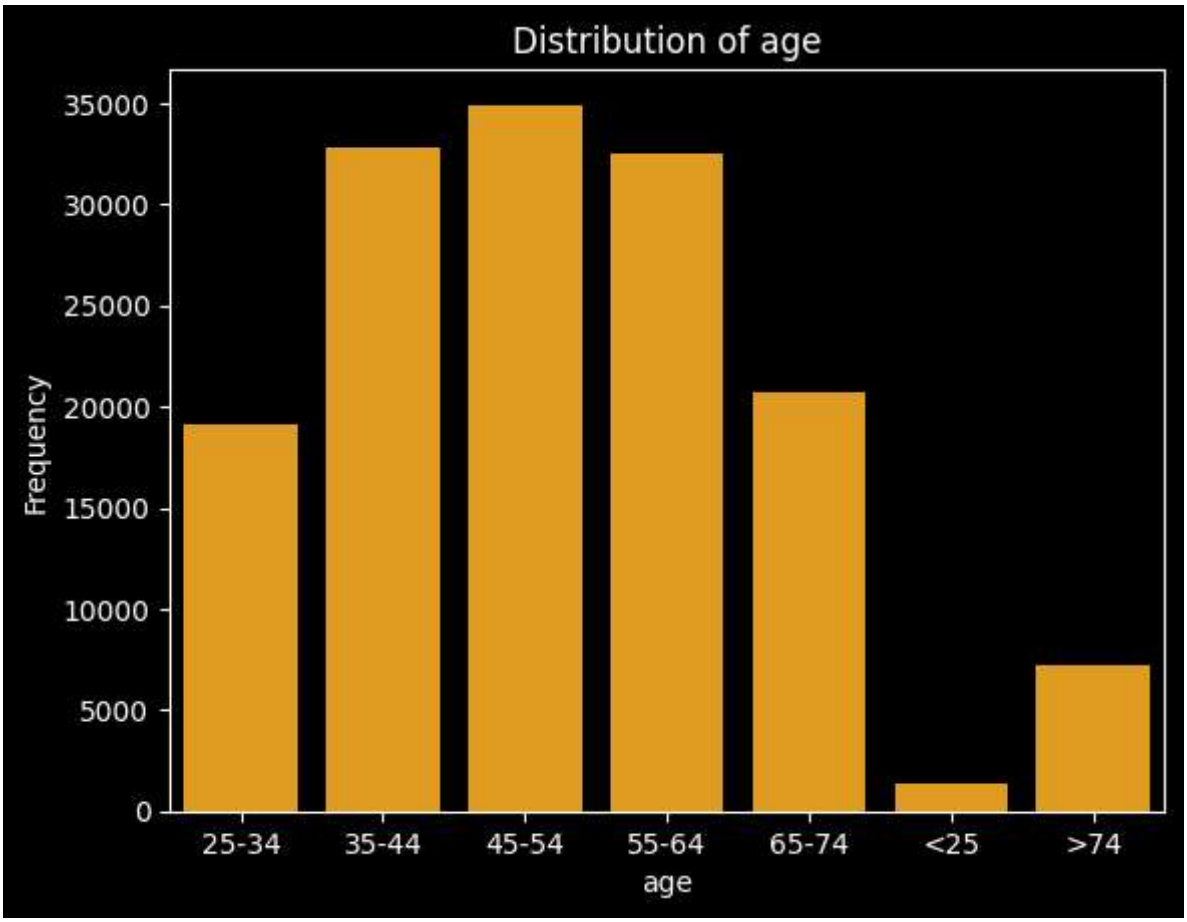
```
Details of co-applicant_credit_type column

DataType:  object

Number of Unique Values:  2

Distribution of column:

co-applicant_credit_type
CIB    74392
EXP    74278
Name: count, dtype: int64

There are no null values
Memory usage of co-applicant_credit_type column :  1189488
Updated Memory usage of  co-applicant_credit_type column :  148922
```

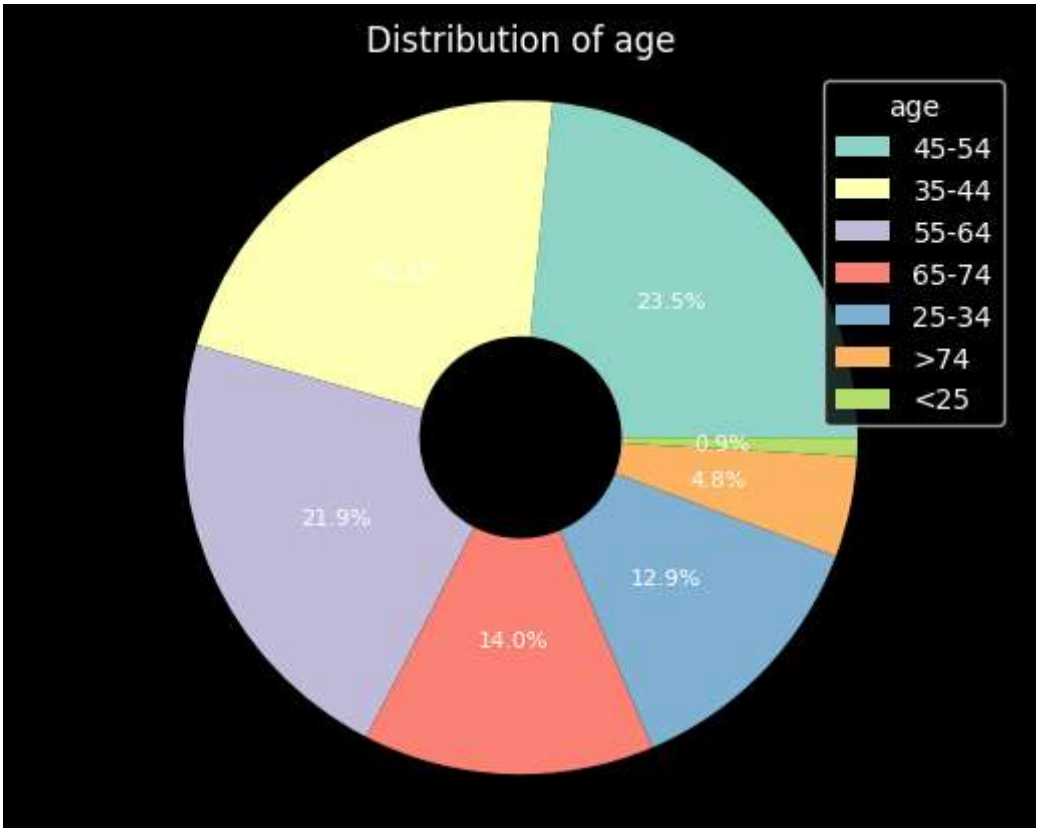




In [159…
```
column_details(df,'age')
convert_dtype(df,'age','category')
fill_with_mode(df,'age')
count_plot(df,'age')
donut_plot(df,'age')
```

```
Details of age column

DataType:  object

Number of Unique Values:  7

Distribution of column:

age
45-54    34720
35-44    32818
55-64    32534
65-74    20744
25-34    19142
>74       7175
<25       1337
Name: count, dtype: int64

There are  200  null values
Memory usage of age column :  1189488
Updated Memory usage of  age column :  149154

No. of missing values after filling with column mode: 0
```



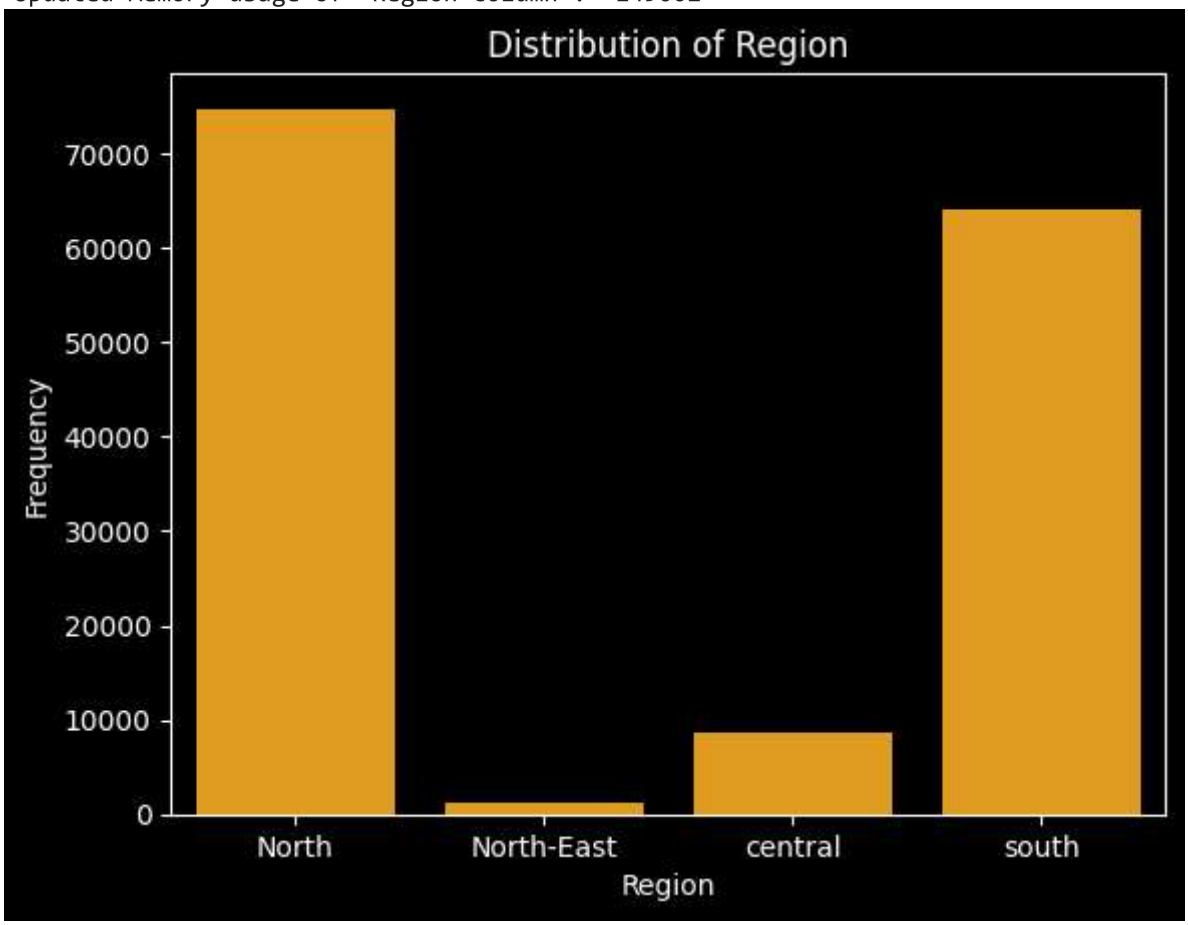


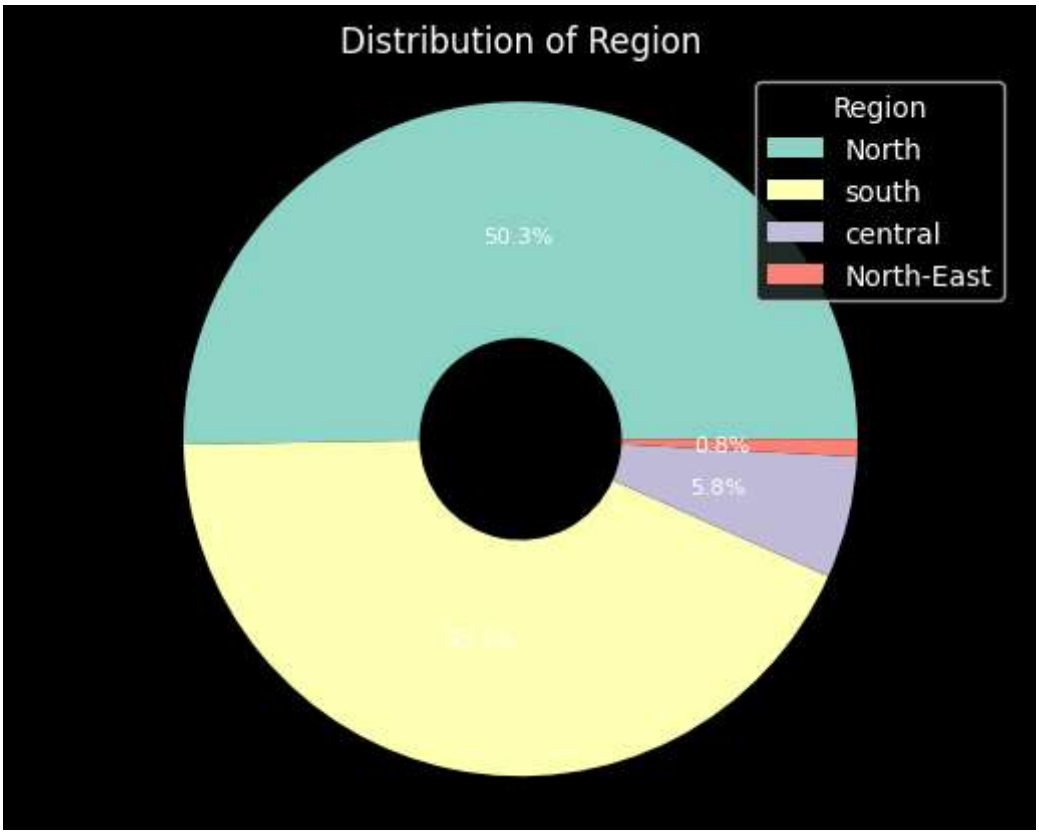
In [160…

```
column_details(df,'Region')
convert_dtype(df,'Region','category')
```

```
count_plot(df,'Region')
donut_plot(df,'Region')
```

```
Details of Region column

DataType:  object

Number of Unique Values:  4

Distribution of column:

Region
North         74722
south         64016
central        8697
North-East     1235
Name: count, dtype: int64

There are no null values
Memory usage of Region column :  1189488
Updated Memory usage of  Region column :  149002
```

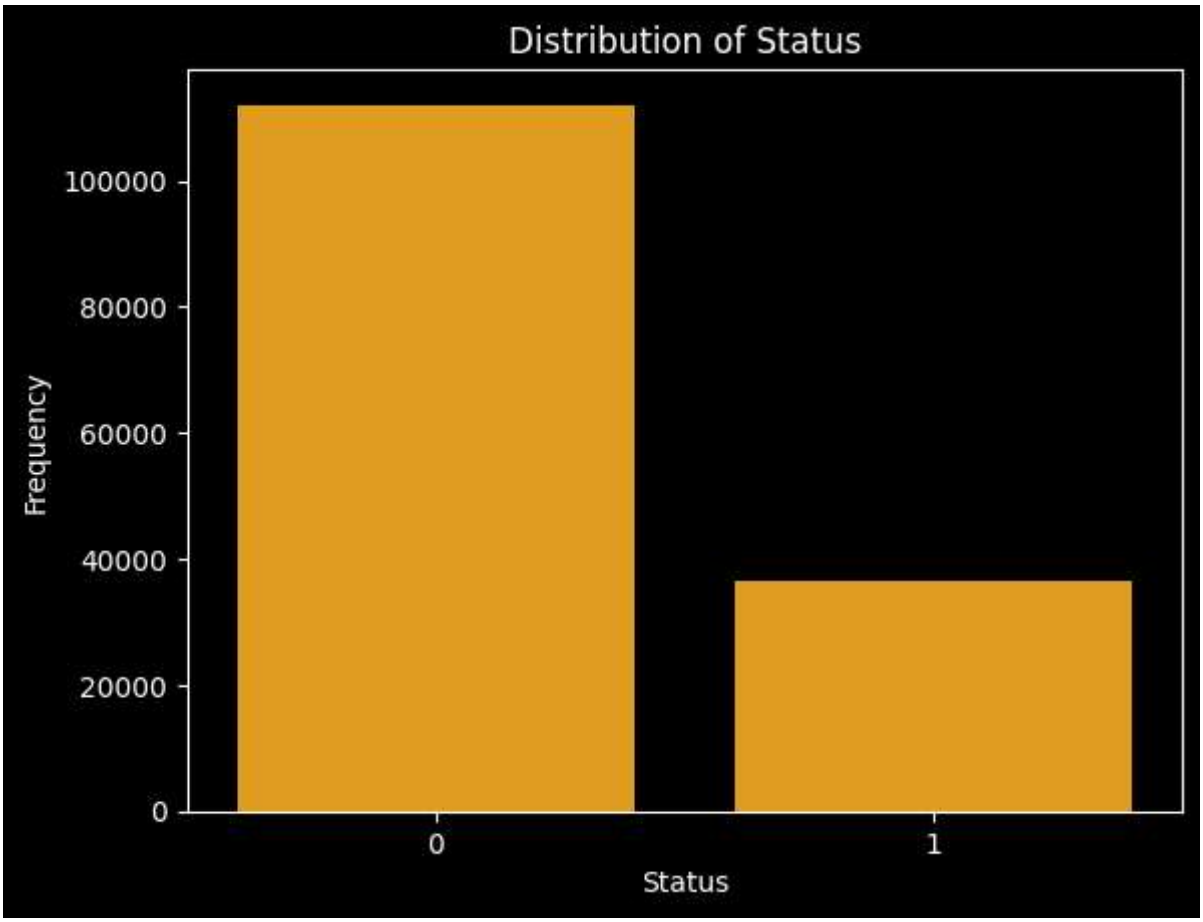


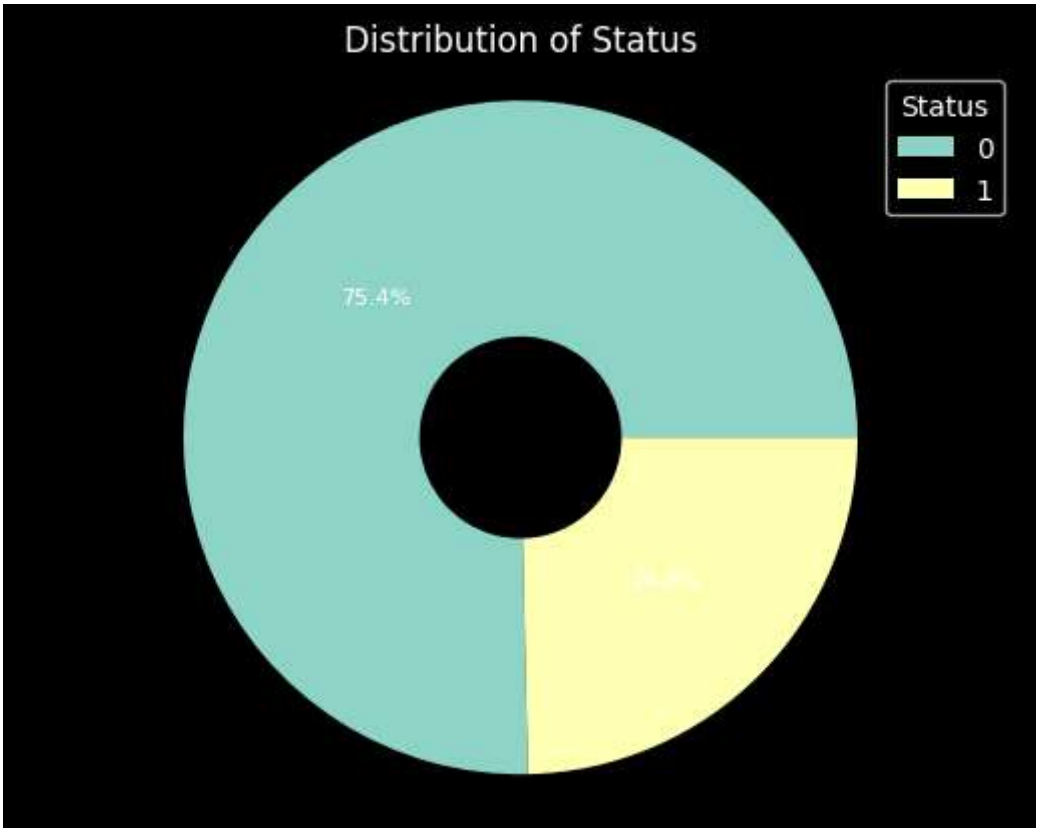


In [161…

```
column_details(df,'Status')
convert_dtype(df,'Status','category')
count_plot(df,'Status')
donut_plot(df,'Status')
```

```
Details of Status column

DataType:  int64

Number of Unique Values:  2

Distribution of column:

Status
0    112031
1     36639
Name: count, dtype: int64

There are no null values
Memory usage of Status column :  1189488
Updated Memory usage of  Status column :  148922
```



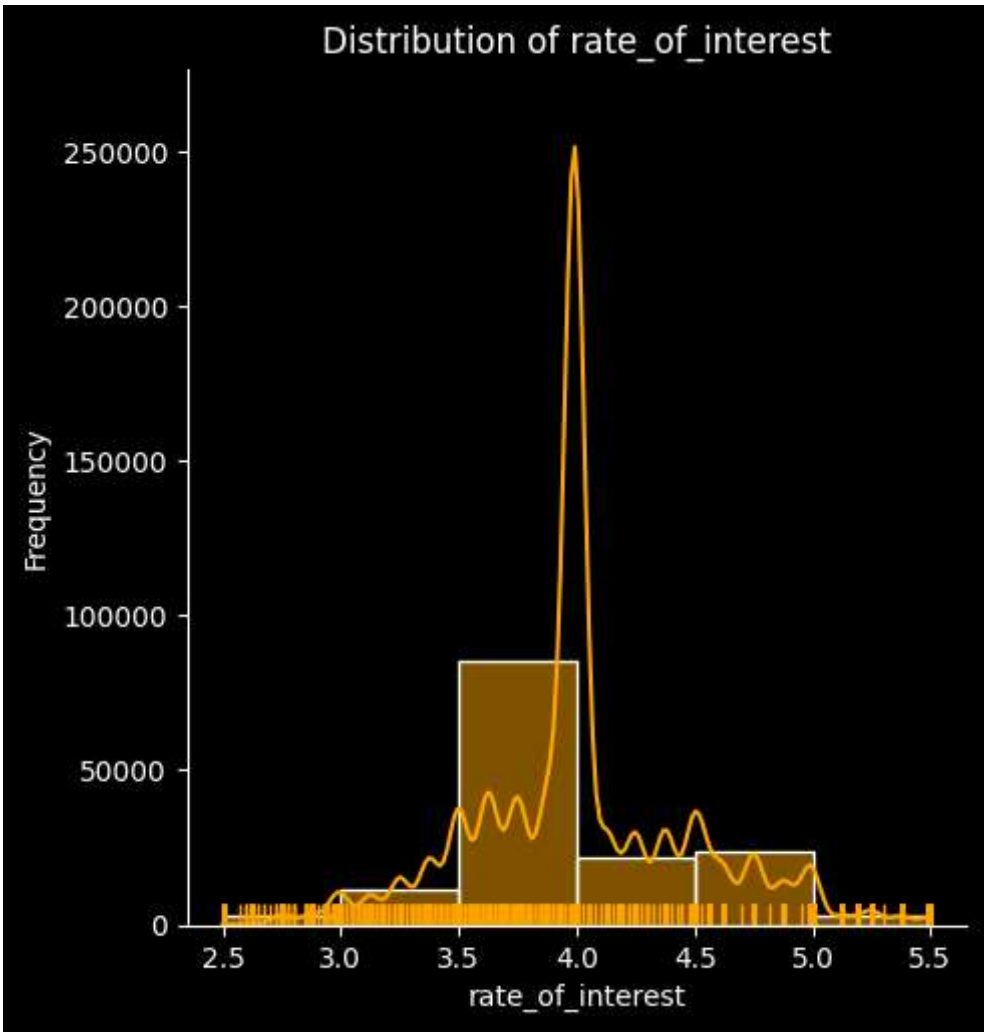


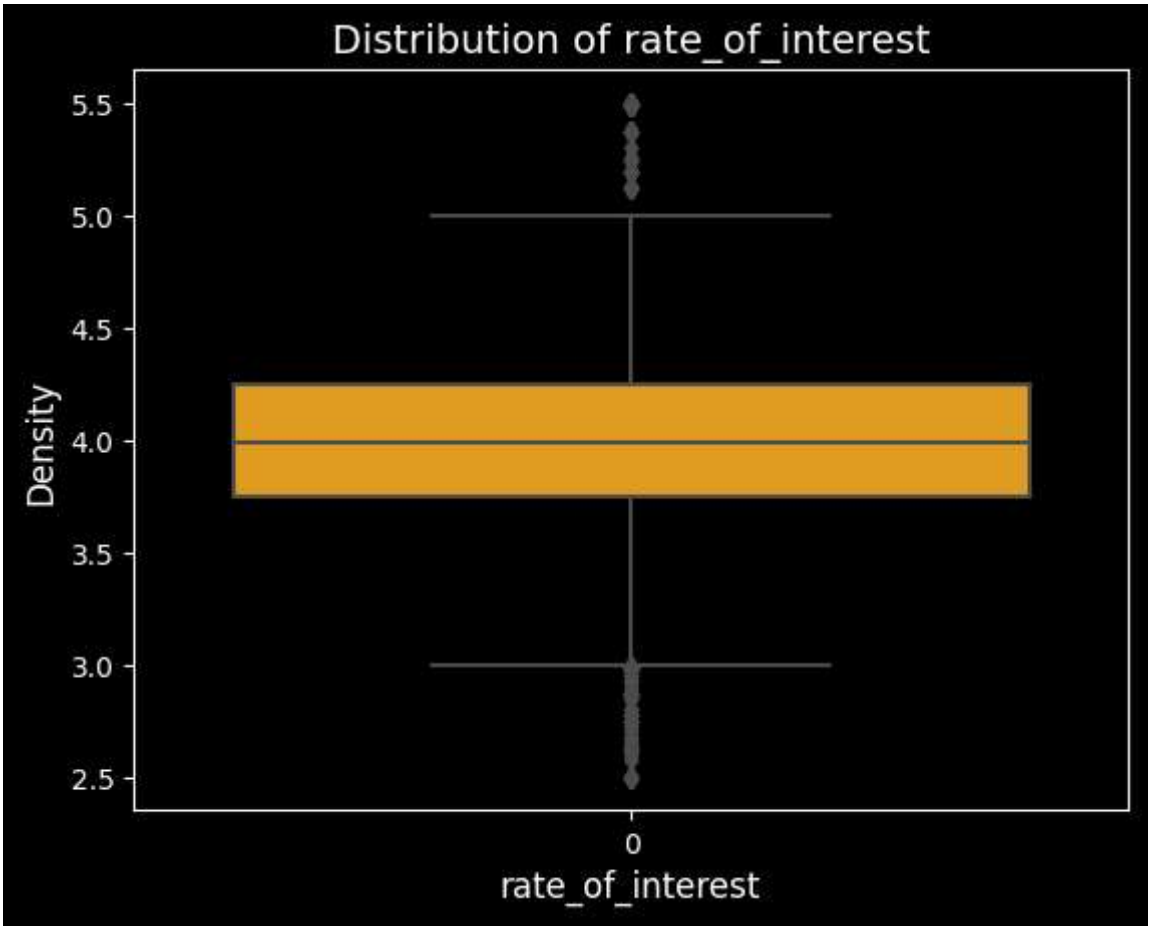
In [162…

```
column_details(df,'rate_of_interest')
outliers_treatment(df,'rate_of_interest')
fill_with_mode(df,'rate_of_interest')
convert_dtype(df,'rate_of_interest','float32')
```

```
displot(df,'rate_of_interest')
box_plot(df,'rate_of_interest')
```

```
Details of rate_of_interest column

DataType:  float64

Number of Unique Values:  131

Distribution of column:

rate_of_interest
3.990    14455
3.625     8800
3.875     8592
3.750     8474
3.500     6866
         ...
4.700        1
8.000        1
7.750        1
5.300        1
2.700        1
Name: count, Length: 131, dtype: int64

There are  36439  null values

No. of missing values after filling with column mode: 0
Memory usage of rate_of_interest column :  1189488
Updated Memory usage of  rate_of_interest column :  594808
```

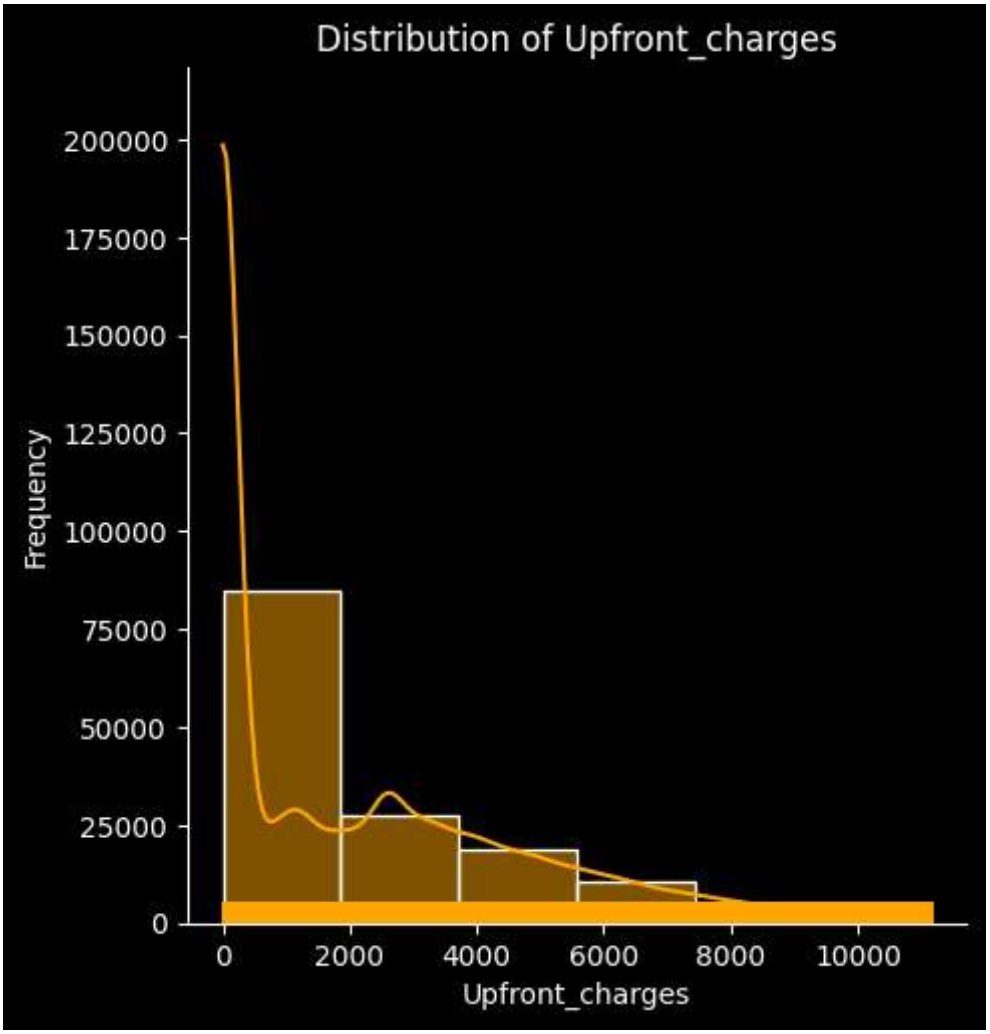

Distribution of rate_of_interest
Frequency
250000
200000
150000
100000
50000
0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
rate_of_interest

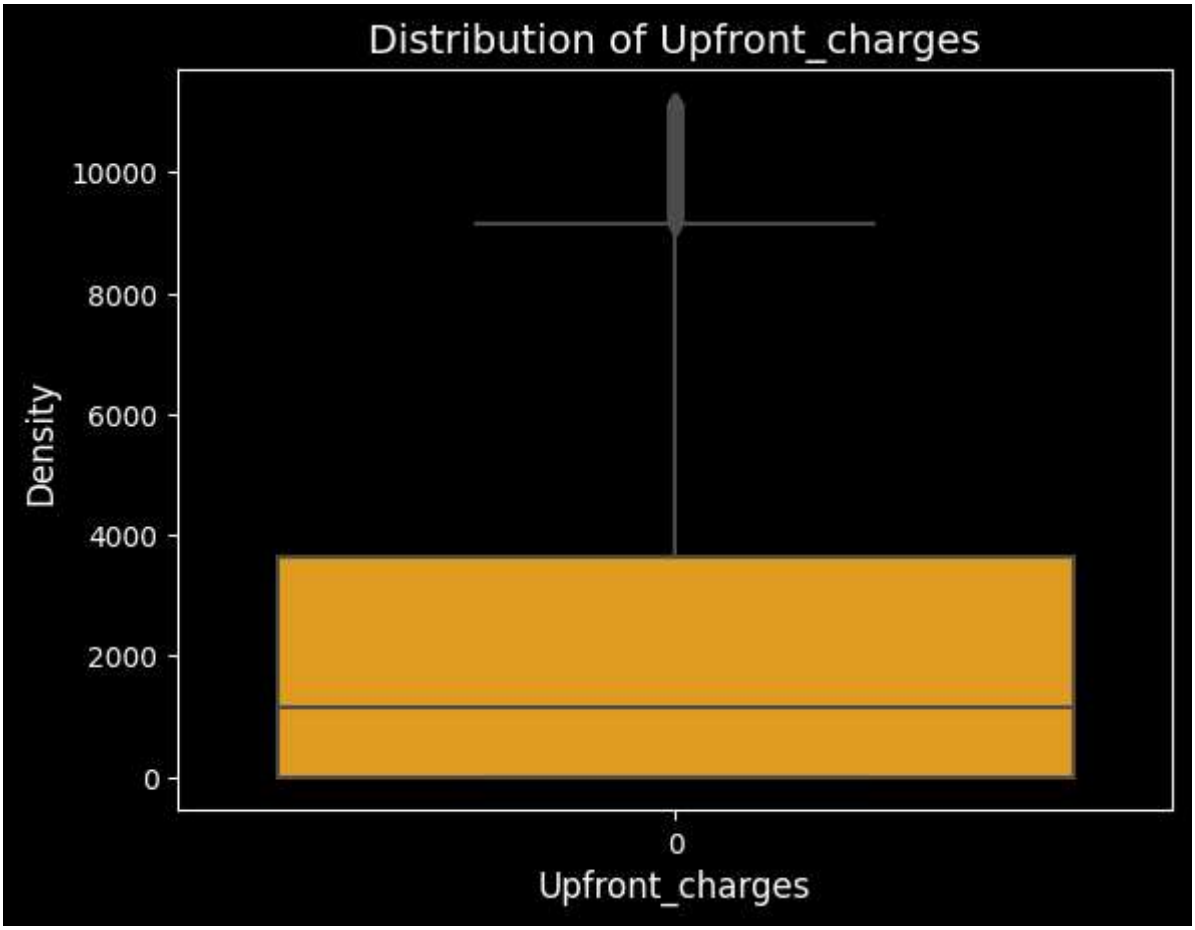


In [163…

```
column_details(df,'Upfront_charges')
outliers_treatment(df,'Upfront_charges')
fill_with_mode(df,'Upfront_charges')
convert_dtype(df,'Upfront_charges','float32')
displot(df,'Upfront_charges')
box_plot(df,'Upfront_charges')
```

```
Details of Upfront_charges column

DataType:  float64

Number of Unique Values:  58271

Distribution of column:

Upfront_charges
0.00       20770
1250.00     1184
1150.00      892
795.00       487
295.00       403
            ...
4447.72        1
3173.84        1
3421.81        1
198.96         1
4323.33        1
Name: count, Length: 58271, dtype: int64

There are  39642  null values

No. of missing values after filling with column mode: 0
Memory usage of Upfront_charges column :  1189488
Updated Memory usage of  Upfront_charges column :  594808
```

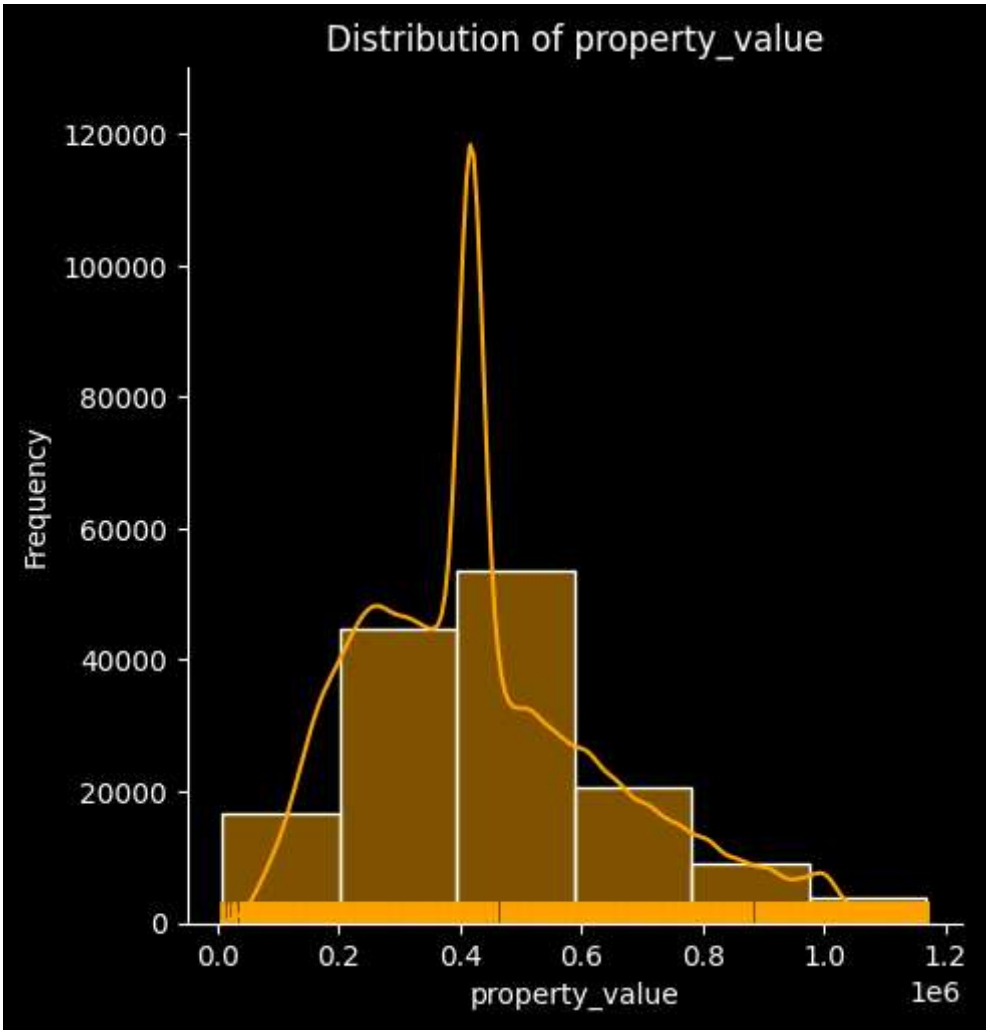

Distribution of Upfront_charges
Frequency
200000
175000
150000
125000
100000
75000
50000
25000
0
0
2000
4000
6000
8000
10000
Upfront_charges

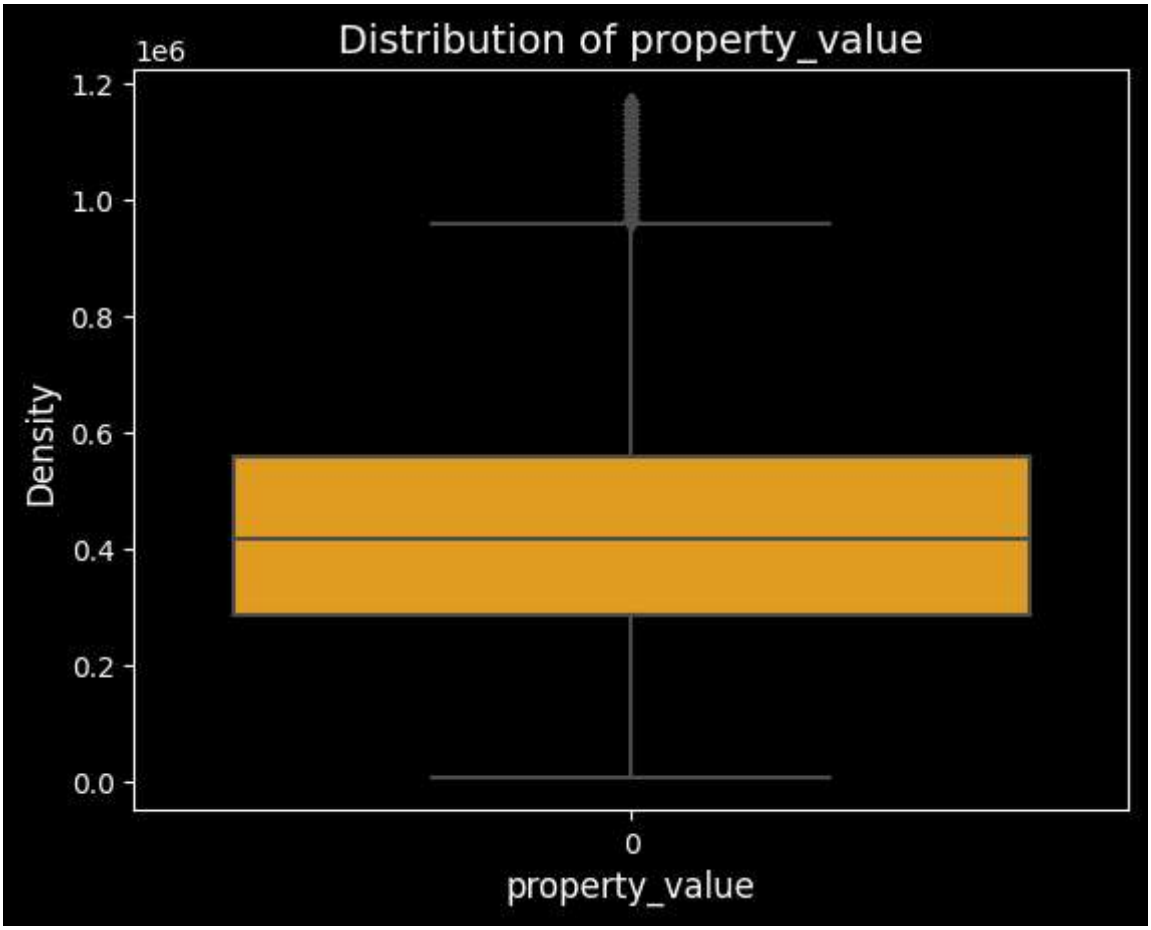


In [164…

```
column_details(df,'property_value')
outliers_treatment(df,'property_value')
fill_with_mode(df,'property_value')
convert_dtype(df,'property_value','int32')
displot(df,'property_value')
box_plot(df,'property_value')
```

```
Details of property_value column

DataType:  float64

Number of Unique Values:  385

Distribution of column:

property_value
308000.0     2792
258000.0     2763
358000.0     2679
408000.0     2537
328000.0     2524
             ...
4648000.0       1
3878000.0       1
5758000.0       1
2618000.0       1
2698000.0       1
Name: count, Length: 385, dtype: int64

There are  15098  null values

No. of missing values after filling with column mode: 0
Memory usage of property_value column :  1189488
Updated Memory usage of  property_value column :  594808
```

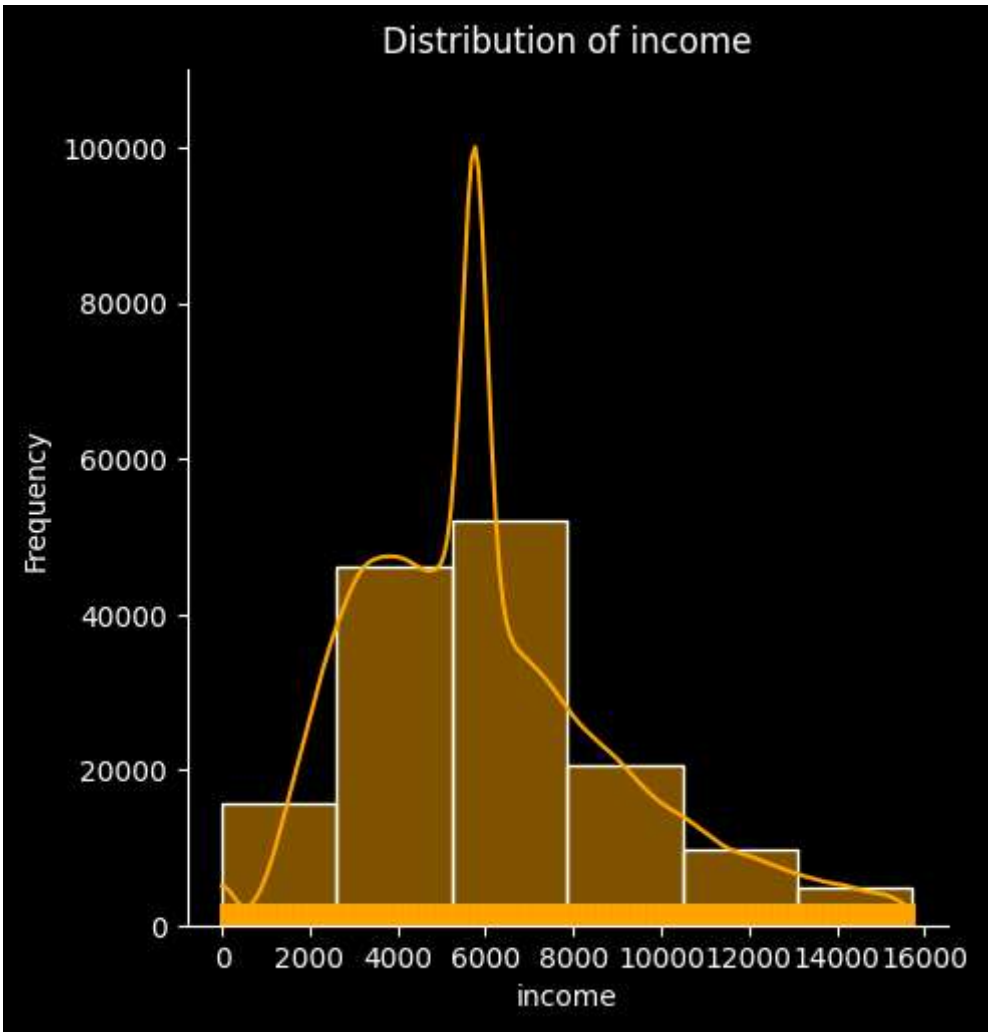

Distribution of property_value
Frequency
120000
100000
80000
60000
40000
20000
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
property_value
1e6

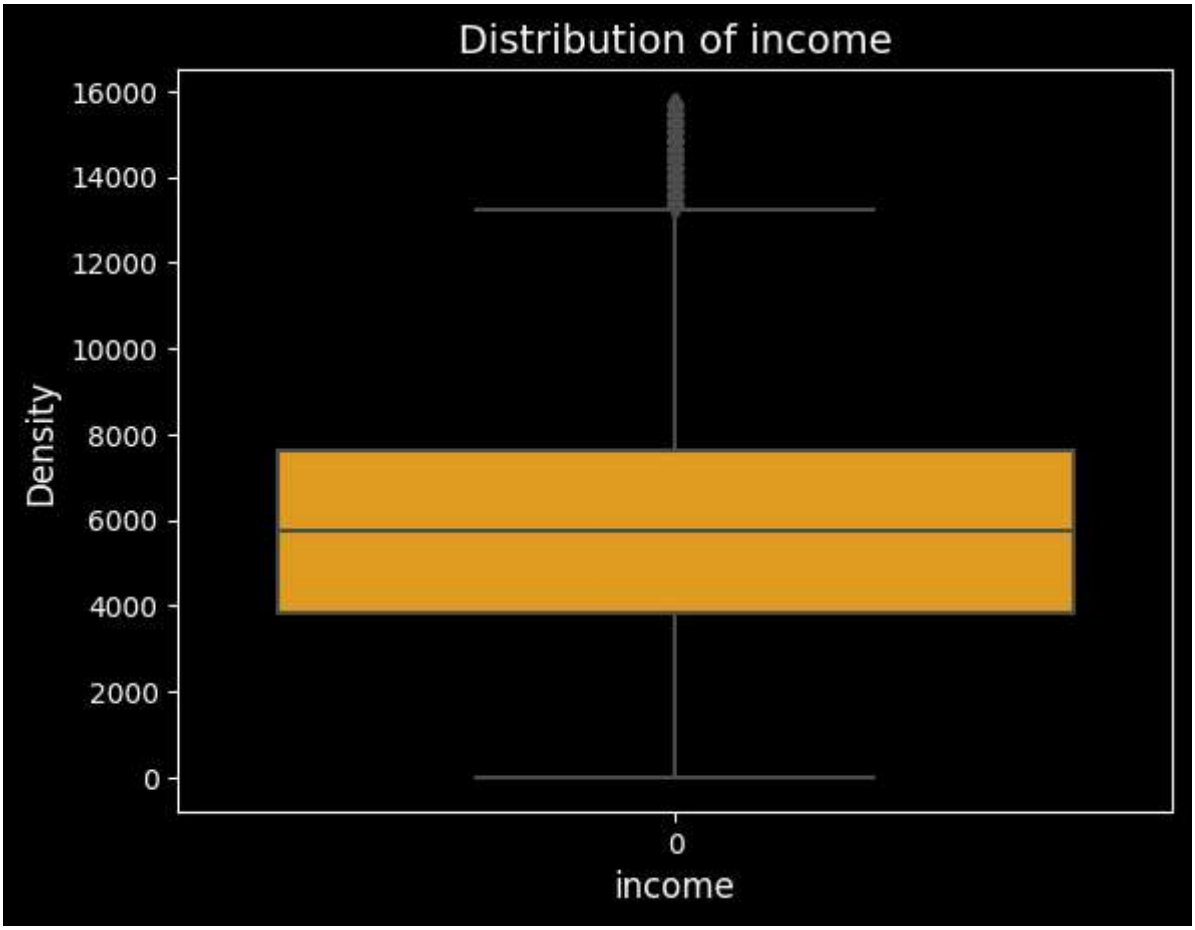


In [165…

```
column_details(df,'income')
outliers_treatment(df,'income')
fill_with_mode(df,'income')
convert_dtype(df,'income','int32')
displot(df,'income')
box_plot(df,'income')
```

```
Details of income column

DataType:  float64

Number of Unique Values:  1001

Distribution of column:

income
0.0         1260
3600.0      1250
4200.0      1243
4800.0      1191
3120.0      1168
            ...
45300.0        1
154440.0       1
137760.0       1
145560.0       1
79920.0        1
Name: count, Length: 1001, dtype: int64

There are  9150  null values

No. of missing values after filling with column mode: 0
Memory usage of income column :  1189488
Updated Memory usage of  income column :  594808
```

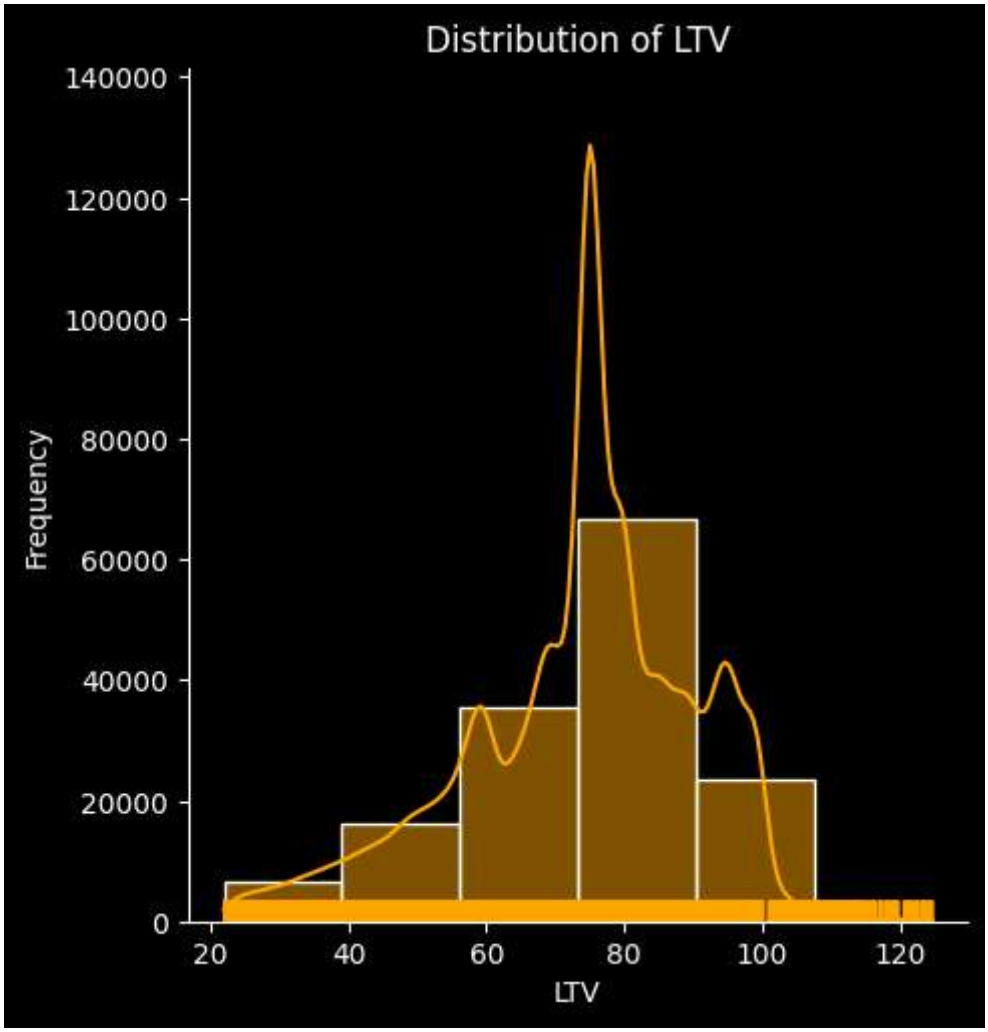

Distribution of income
Frequency
100000
80000
60000
40000
20000
0
0
2000
4000
6000
8000
10000
12000
14000
16000
income

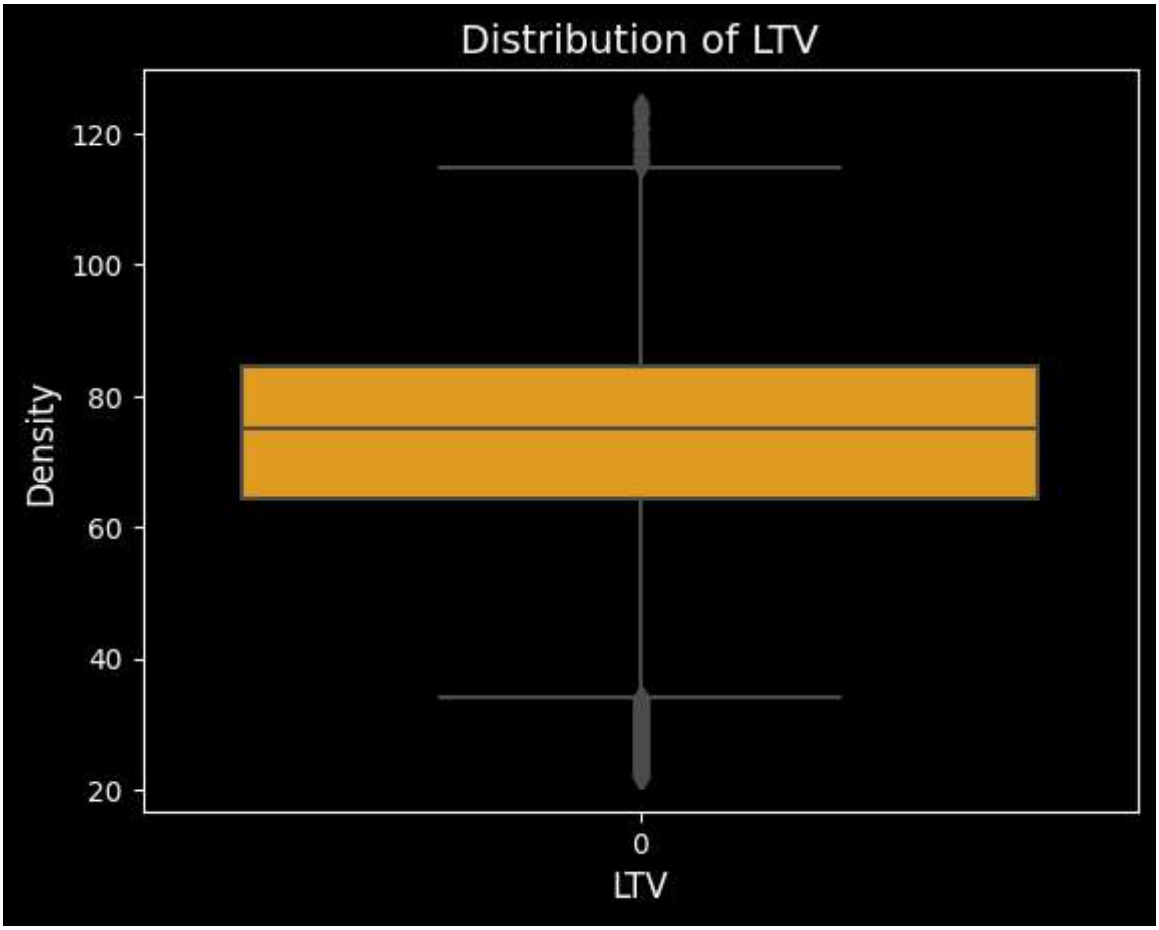


In [166…

```
column_details(df,'LTV')
outliers_treatment(df,'LTV')
fill_with_mode(df,'LTV')
convert_dtype(df,'LTV','float32')
displot(df,'LTV')
box_plot(df,'LTV')
```

```
Details of LTV column

DataType:  float64

Number of Unique Values:  8484

Distribution of column:

LTV
81.250000    530
91.666667    499
80.038760    380
80.032468    328
94.956140    322
             ...
33.598066      1
13.618746      1
55.981445      1
50.175070      1
32.802013      1
Name: count, Length: 8484, dtype: int64

There are  15098  null values

No. of missing values after filling with column mode: 0
Memory usage of LTV column :  1189488
Updated Memory usage of  LTV column :  594808
```


Distribution of LTV
Frequency
140000
120000
100000
80000
60000
40000
20000
0
20
40
60
80
100
120
LTV



In [167…
```
replace_dict = {1: 'Defaulter', 0: 'Normal'}
df['Status'] = df['Status'].replace(replace_dict)
```

In [168…
```
replace_dict_region = {'south': 'South', 'central': 'Central'}
df['Region'] = df['Region'].replace(replace_dict_region)
```

In [169…
```
df.head(5)
```

Out[169]:

	ID	year	loan_limit	Gender	loan_type	loan_purpose	business_or_commercial	loa
0	24890	2019	cf	Sex Not Available	type1	p1	nob/c	
1	24891	2019	cf	Male	type2	p1	b/c	
2	24892	2019	cf	Male	type1	p1	nob/c	
3	24893	2019	cf	Male	type1	p4	nob/c	
4	24894	2019	cf	Joint	type1	p1	nob/c	

Feature Engineering

Loan to income ratio

In [170…
```
df['loan_to_income_ratio'] = df.apply(lambda x: 0 if x['income'] == 0 else x['loan_
```

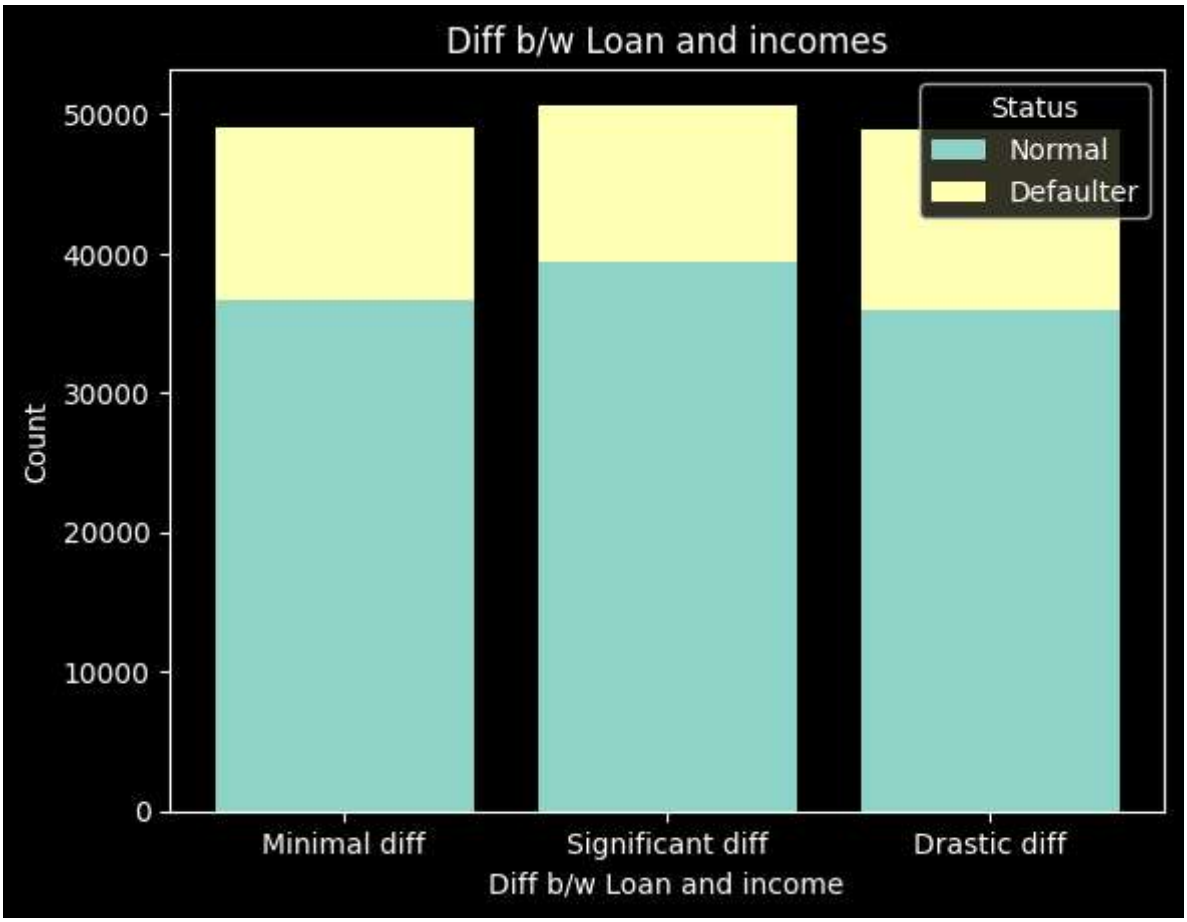
In [171…
```
df['loan_to_income_ratio'] = pd.qcut(df['loan_to_income_ratio'], q=[0, 0.33, 0.67,
df.head()
```

Out[171]:

	ID	year	loan_limit	Gender	loan_type	loan_purpose	business_or_commercial	loa
0	24890	2019	cf	Sex Not Available	type1	p1	nob/c	
1	24891	2019	cf	Male	type2	p1	b/c	
2	24892	2019	cf	Male	type1	p1	nob/c	
3	24893	2019	cf	Male	type1	p4	nob/c	
4	24894	2019	cf	Joint	type1	p1	nob/c	

5 rows × 21 columns

In [172…
```
counts = df.groupby(['loan_to_income_ratio', 'Status']).size().unstack(fill_value=0

# Create a stem plot-like bar plot
fig, ax = plt.subplots()

for i, col in enumerate(counts.columns):
    ax.bar(counts.index, counts[col], bottom=counts.iloc[:, :i].sum(axis=1), label=

# Customize plot
ax.set_xlabel('Diff b/w Loan and income')
ax.set_ylabel('Count')
ax.set_title('Diff b/w Loan and incomes')
ax.legend(title='Status')

plt.show()
```



Credit score category

In [173…
```
df['Credit_category'] = pd.qcut(df['Credit_Score'], q=[0, 0.25,0.5,0.75, 1], labels
df.head()
```

Out[173]:

	ID	year	loan_limit	Gender	loan_type	loan_purpose	business_or_commercial	loa
0	24890	2019	cf	Sex Not Available	type1	p1	nob/c	
1	24891	2019	cf	Male	type2	p1	b/c	
2	24892	2019	cf	Male	type1	p1	nob/c	
3	24893	2019	cf	Male	type1	p4	nob/c	
4	24894	2019	cf	Joint	type1	p1	nob/c	

5 rows × 22 columns

Income category

In [174…
```
df['Income_category'] = pd.qcut(df['income'], q=[0, 0.33, 0.67, 1], labels=['Low In
df.head()
```

Out[174]:

	ID	year	loan_limit	Gender	loan_type	loan_purpose	business_or_commercial	loa
0	24890	2019	cf	Sex Not Available	type1	p1	nob/c	
1	24891	2019	cf	Male	type2	p1	b/c	
2	24892	2019	cf	Male	type1	p1	nob/c	
3	24893	2019	cf	Male	type1	p4	nob/c	
4	24894	2019	cf	Joint	type1	p1	nob/c	

5 rows × 23 columns

LTV category

In [175…
```
df['LTV_category'] = pd.qcut(df['LTV'], q=[0.0, 0.2,0.4,0.6,0.8, 1.0], labels=['Low
df.head()
```

Out[175]:

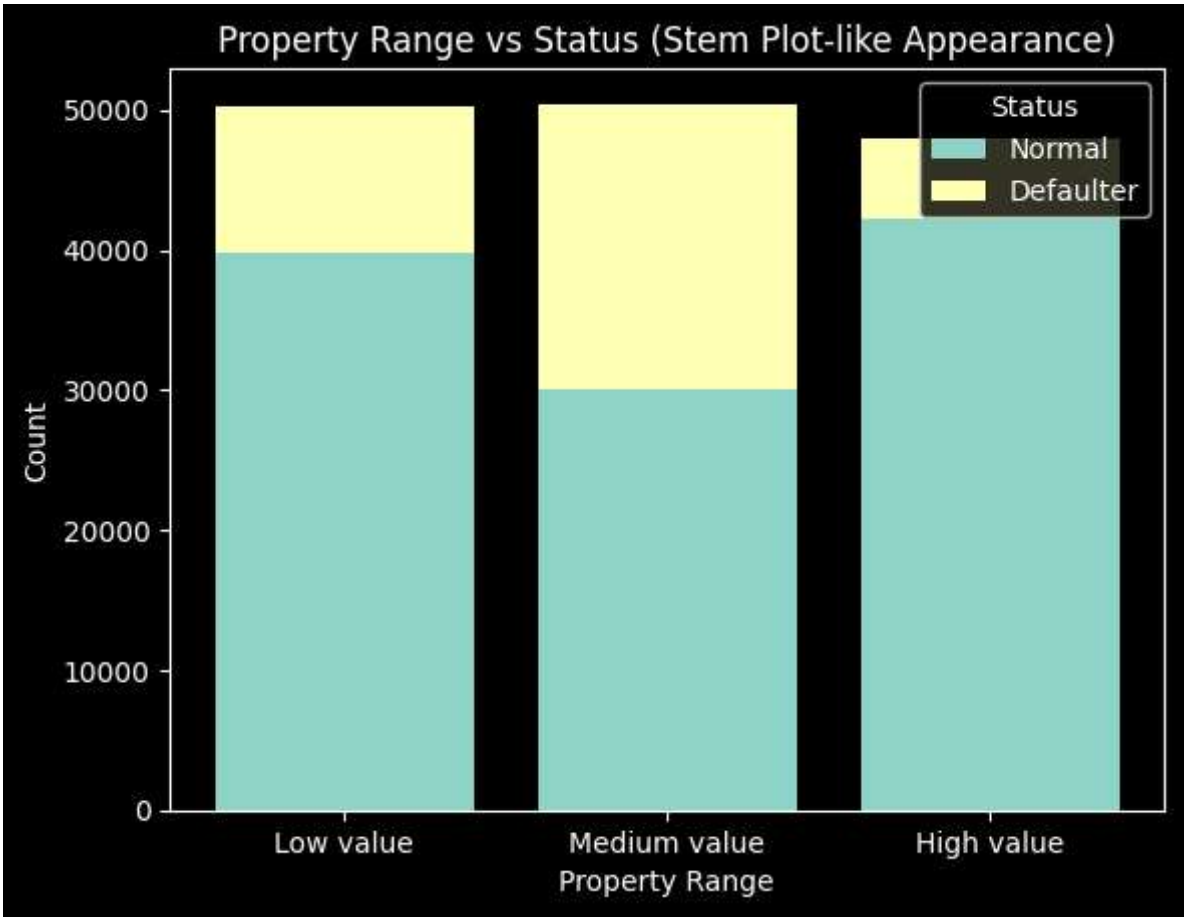
	ID	year	loan_limit	Gender	loan_type	loan_purpose	business_or_commercial	loa
0	24890	2019	cf	Sex Not Available	type1	p1	nob/c	
1	24891	2019	cf	Male	type2	p1	b/c	
2	24892	2019	cf	Male	type1	p1	nob/c	
3	24893	2019	cf	Male	type1	p4	nob/c	
4	24894	2019	cf	Joint	type1	p1	nob/c	

5 rows × 24 columns

Property to loan amount category

In [176…
```
df['property_range'] = pd.qcut(df['property_value'], q=[0, 0.33, 0.67, 1], labels=[
```

In [177…
```
counts = df.groupby(['property_range', 'Status']).size().unstack(fill_value=0)

# Create a stem plot-like bar plot
fig, ax = plt.subplots()

for i, col in enumerate(counts.columns):
    ax.bar(counts.index, counts[col], bottom=counts.iloc[:, :i].sum(axis=1), label=

# Customize plot
ax.set_xlabel('Property Range')
ax.set_ylabel('Count')
ax.set_title('Property Range vs Status (Stem Plot-like Appearance)')
ax.legend(title='Status')

plt.show()
```



Loan amount category

In [178…
```
df['loan_amount_category'] = pd.qcut(df['loan_amount'], q=[0, 0.33, 0.67, 1], label
```

In [179…
```
df.head()
```

Out[179]:

	ID	year	loan_limit	Gender	loan_type	loan_purpose	business_or_commercial	loa
0	24890	2019	cf	Sex Not Available	type1	p1	nob/c	
1	24891	2019	cf	Male	type2	p1	b/c	
2	24892	2019	cf	Male	type1	p1	nob/c	
3	24893	2019	cf	Male	type1	p4	nob/c	
4	24894	2019	cf	Joint	type1	p1	nob/c	

5 rows × 26 columns

Property to loan ratio

In [180…
```
df['property_to_loan_ratio'] = df['property_value']-df['loan_amount']
```

In [181…
```
df['property_to_loan_ratio'] = pd.qcut(df['property_to_loan_ratio'], q=[0, 0.33, 0.
df.head()
```

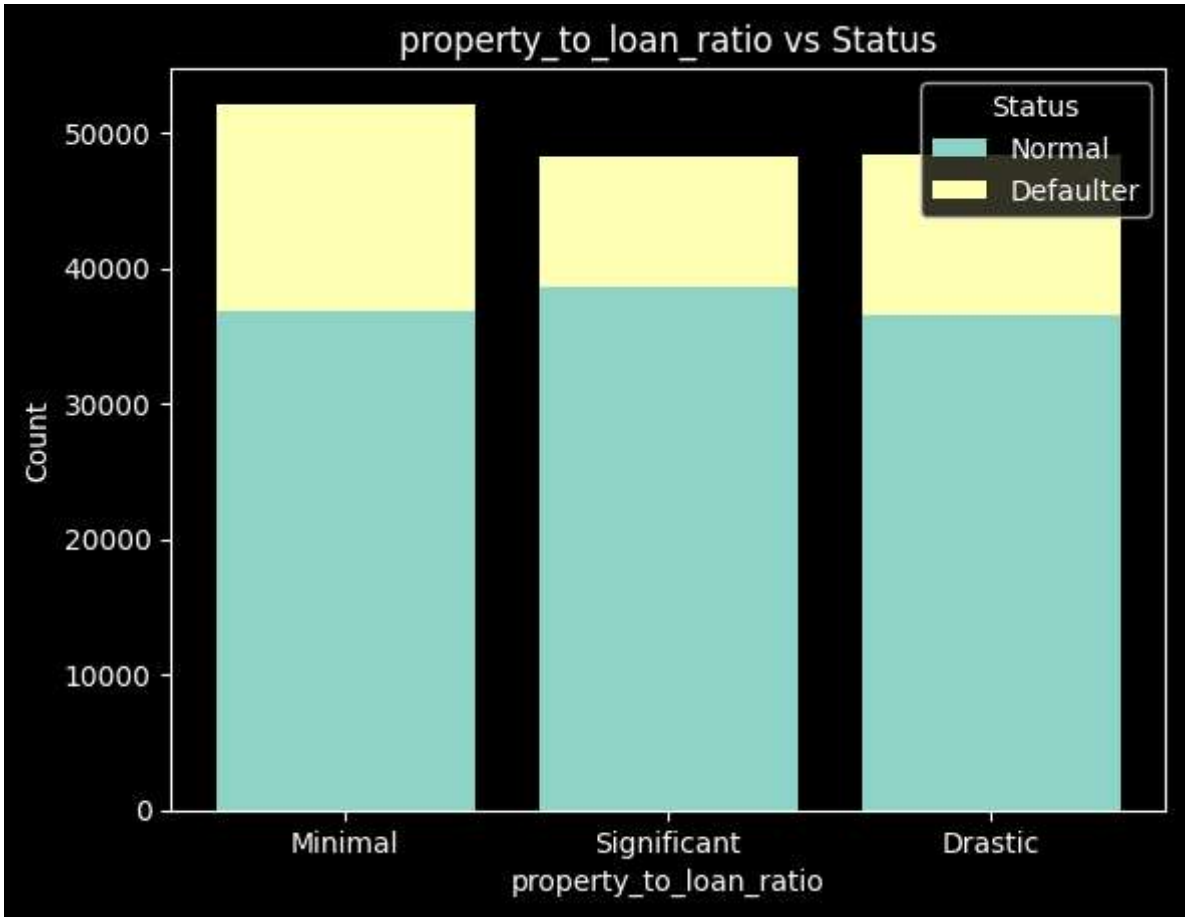
Out[181]:

	ID	year	loan_limit	Gender	loan_type	loan_purpose	business_or_commercial	loa
0	24890	2019	cf	Sex Not Available	type1	p1	nob/c	
1	24891	2019	cf	Male	type2	p1	b/c	
2	24892	2019	cf	Male	type1	p1	nob/c	
3	24893	2019	cf	Male	type1	p4	nob/c	
4	24894	2019	cf	Joint	type1	p1	nob/c	

5 rows × 27 columns

In [182…
```
counts = df.groupby(['property_to_loan_ratio', 'Status']).size().unstack(fill_value

# Create a stem plot-like bar plot
fig, ax = plt.subplots()

for i, col in enumerate(counts.columns):
    ax.bar(counts.index, counts[col], bottom=counts.iloc[:, :i].sum(axis=1), label=
```

```
# Customize plot
ax.set_xlabel('property_to_loan_ratio')
ax.set_ylabel('Count')
ax.set_title('property_to_loan_ratio vs Status')
ax.legend(title='Status')

plt.show()
```



In [ ]:

In [183…

```
df.head()
```

Out[183]:

	ID	year	loan_limit	Gender	loan_type	loan_purpose	business_or_commercial	loa
0	24890	2019	cf	Sex Not Available	type1	p1	nob/c	
1	24891	2019	cf	Male	type2	p1	b/c	
2	24892	2019	cf	Male	type1	p1	nob/c	
3	24893	2019	cf	Male	type1	p4	nob/c	
4	24894	2019	cf	Joint	type1	p1	nob/c	

5 rows × 27 columns

In [184…
```
df.dtypes
```

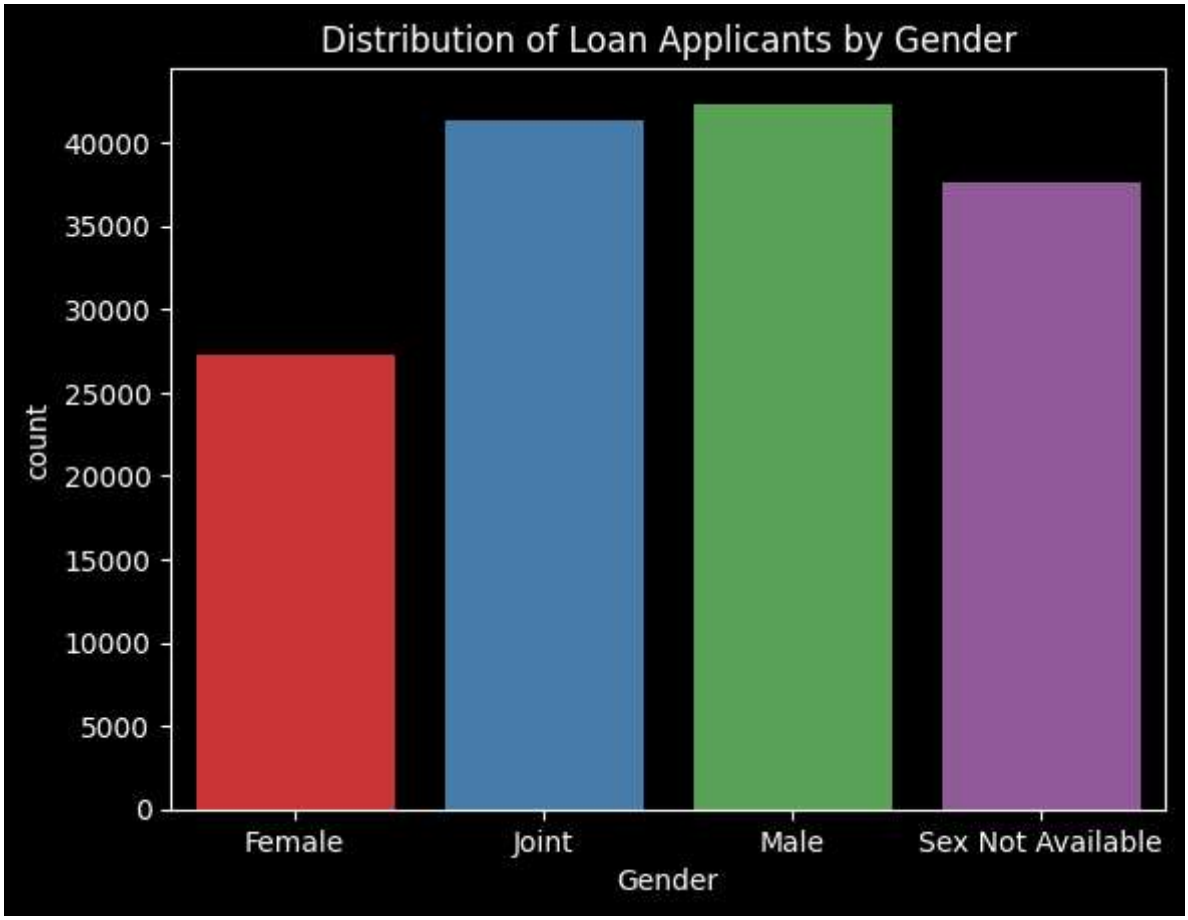
Out[184]:
```
ID                          int64
year                        int64
loan_limit               category
Gender                   category
loan_type                category
loan_purpose             category
business_or_commercial   category
loan_amount                 int32
rate_of_interest          float32
Upfront_charges           float32
property_value              int32
occupancy_type           category
income                      int32
credit_type              category
Credit_Score                int16
co-applicant_credit_type category
age                      category
LTV                       float32
Region                   category
Status                   category
loan_to_income_ratio     category
Credit_category          category
Income_category          category
LTV_category             category
property_range           category
loan_amount_category     category
property_to_loan_ratio   category
dtype: object
```

Demographic analysis

In [185…
```
# Distribution of loan applicants based on gender
sns.countplot(x='Gender', data=df, palette='Set1')
```

```
plt.title('Distribution of Loan Applicants by Gender')
```

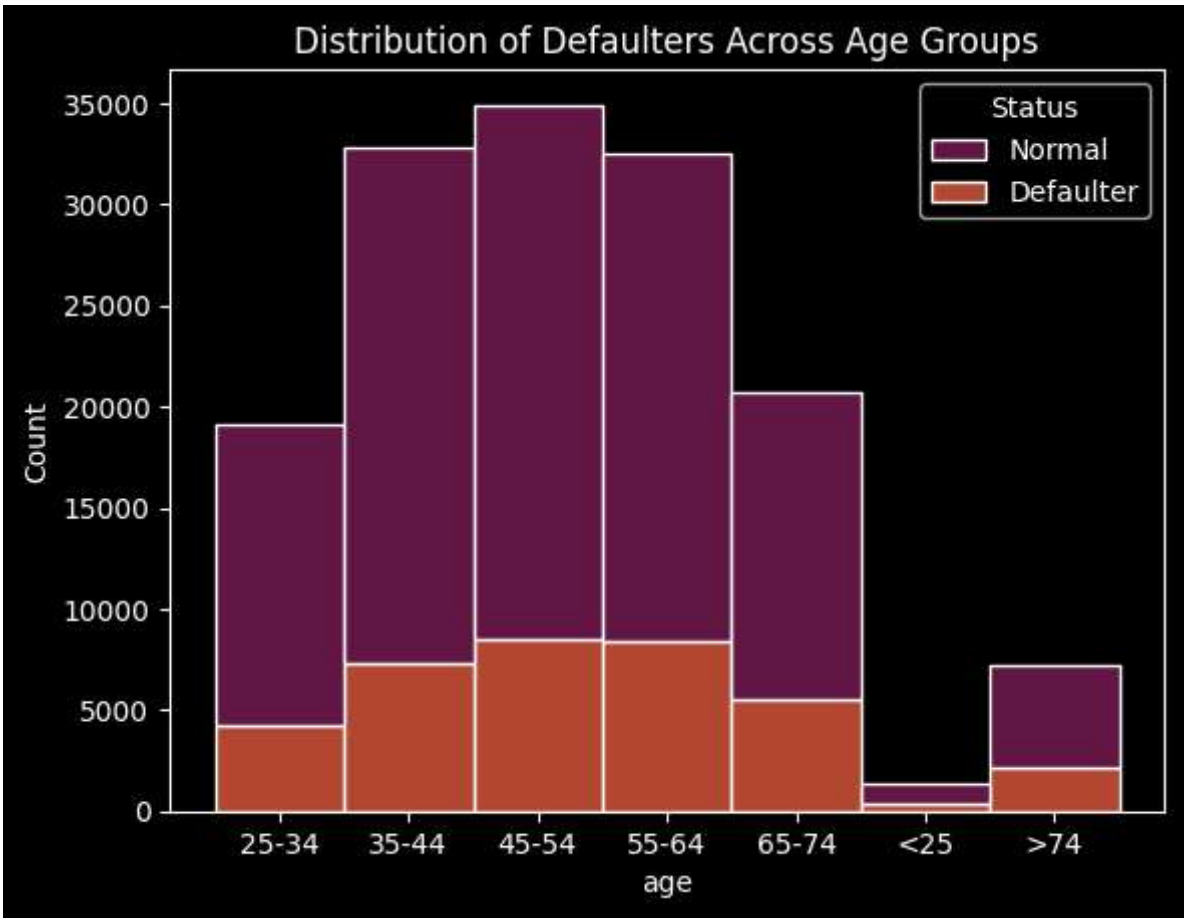
Out[185]: Text(0.5, 1.0, 'Distribution of Loan Applicants by Gender')



In [186…

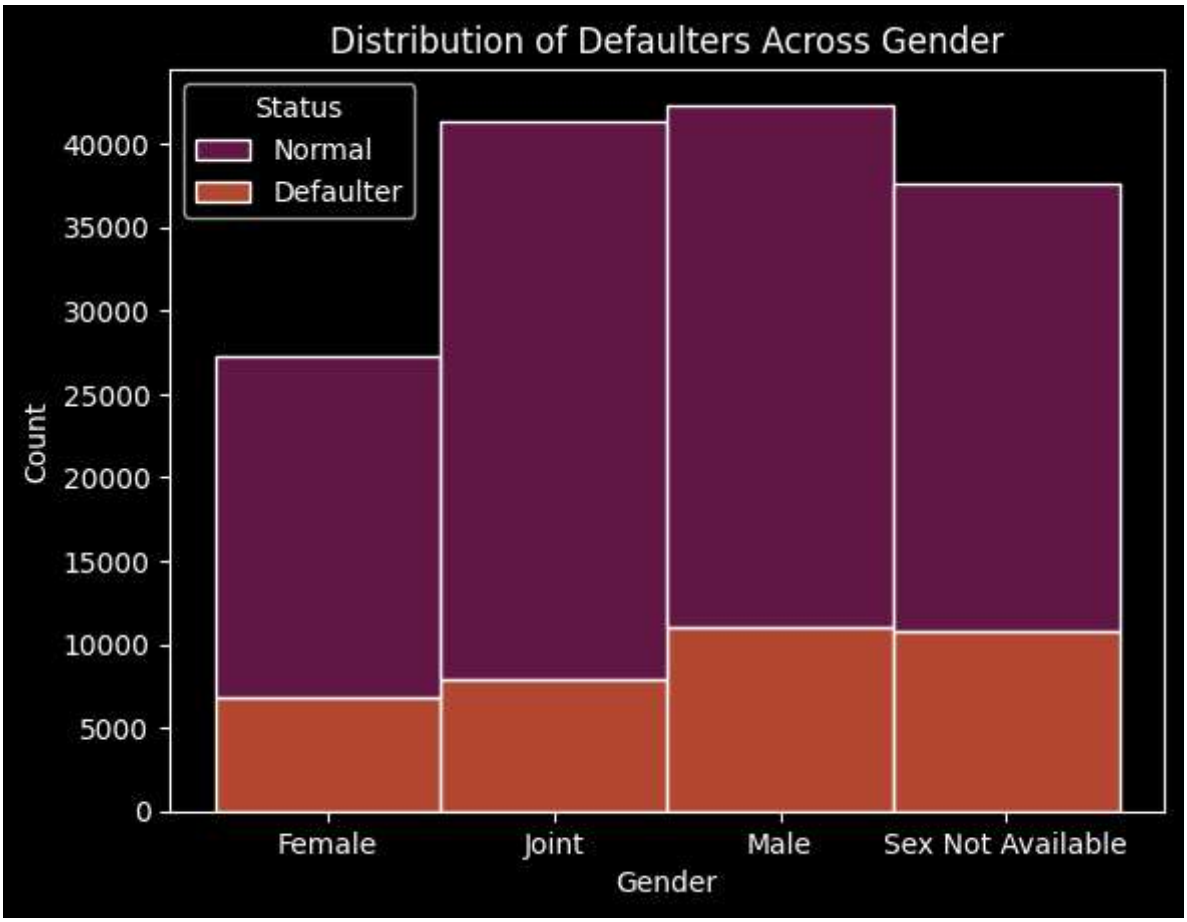
```
# Distribution of defaulters across different age groups
sns.histplot(x='age', data=df, hue='Status', multiple='stack', palette='rocket', bi
plt.title('Distribution of Defaulters Across Age Groups')
```

Out[186]: Text(0.5, 1.0, 'Distribution of Defaulters Across Age Groups')



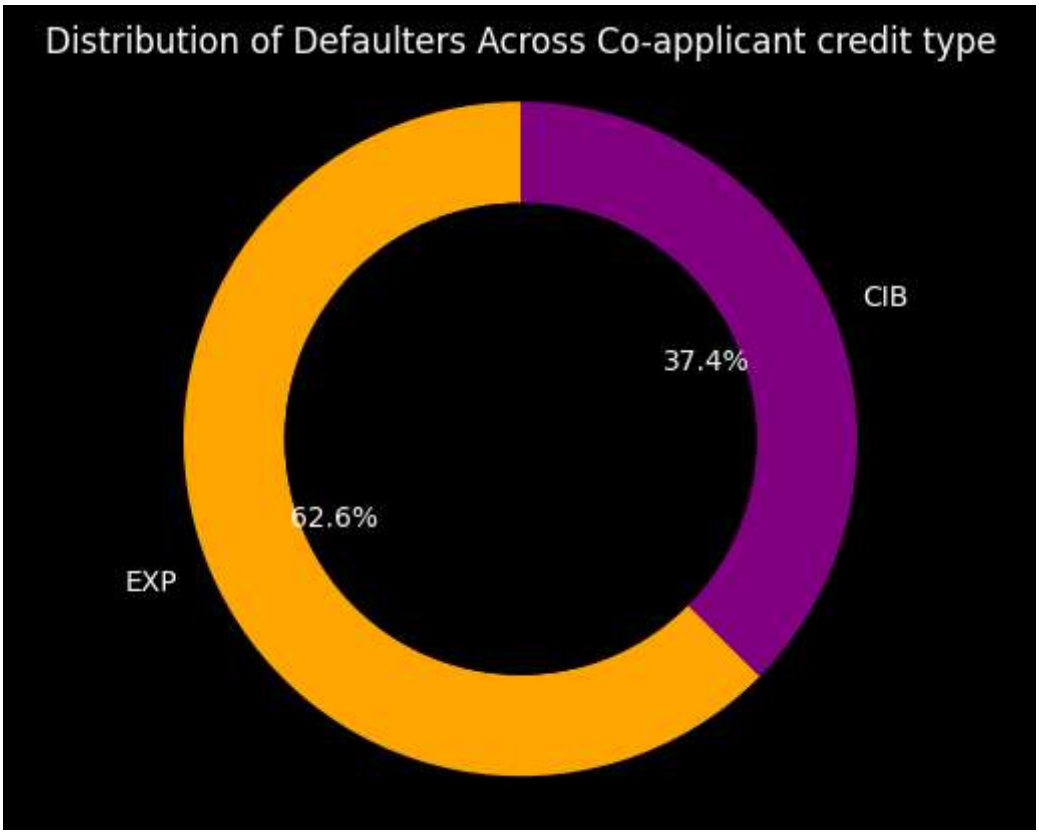
In [187…
```
# Distribution of defaulters across different age groups
sns.histplot(x='Gender', data=df, hue='Status', multiple='stack', palette='rocket',
plt.title('Distribution of Defaulters Across Gender')
```

Out[187]:
```
Text(0.5, 1.0, 'Distribution of Defaulters Across Gender')
```



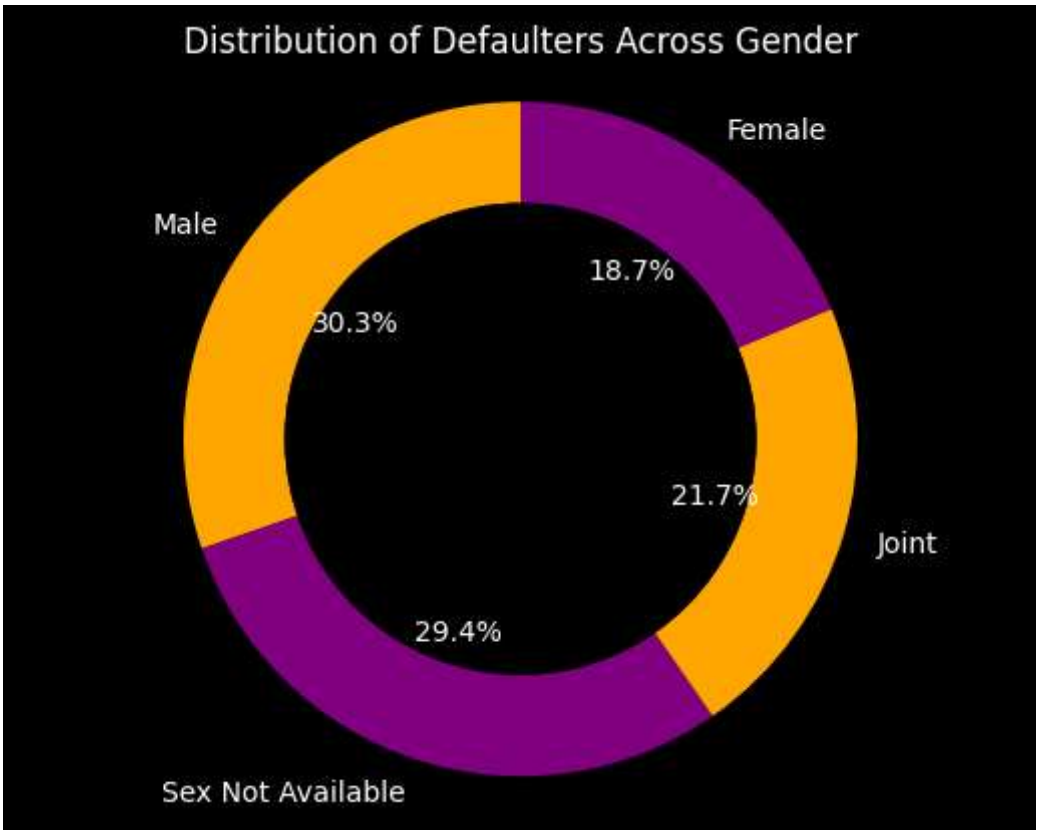
In [188…

```
gender_counts = df[df['Status'] == 'Defaulter']['co-applicant_credit_type'].value_c

# Create a donut chart
fig, ax = plt.subplots()
ax.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%', startangle=90,

# Draw a circle in the center to create a donut chart
centre_circle = plt.Circle((0, 0), 0.7, fc='black')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

# Equal aspect ratio ensures that the pie chart is circular
ax.axis('equal')

# Set title
plt.title('Distribution of Defaulters Across Co-applicant credit type')

# Show the donut chart
plt.show()
```



In [189…

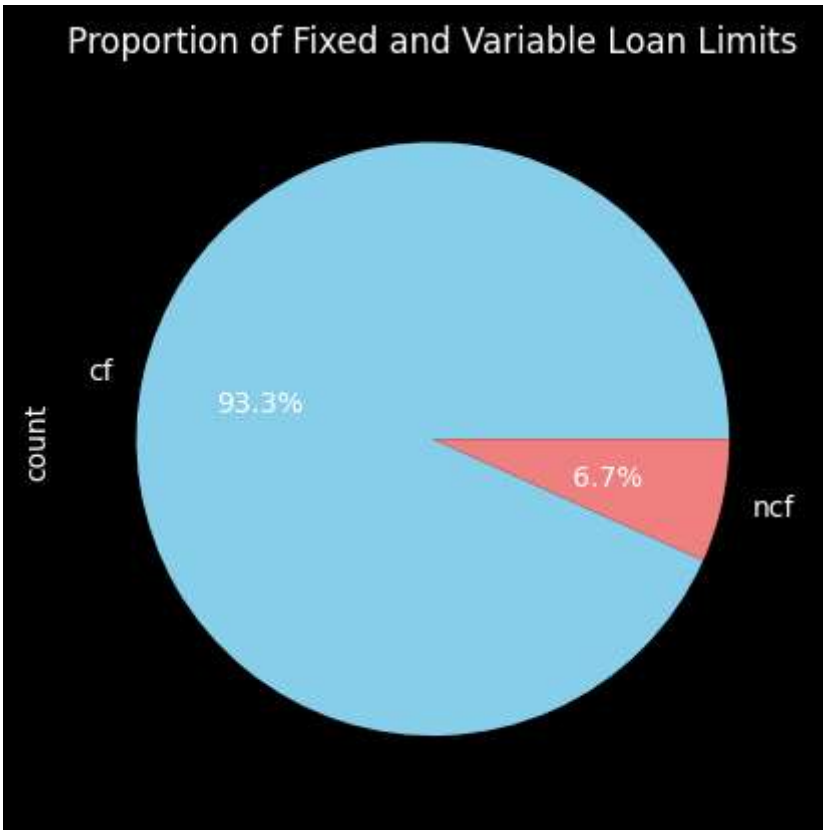
```
gender_counts = df[df['Status'] == 'Defaulter']['Gender'].value_counts()

# Create a donut chart
fig, ax = plt.subplots()
ax.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%', startangle=90,

# Draw a circle in the center to create a donut chart
centre_circle = plt.Circle((0, 0), 0.7, fc='black')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

# Equal aspect ratio ensures that the pie chart is circular
ax.axis('equal')

# Set title
plt.title('Distribution of Defaulters Across Gender')

# Show the donut chart
plt.show()
```



Loan Characteristics

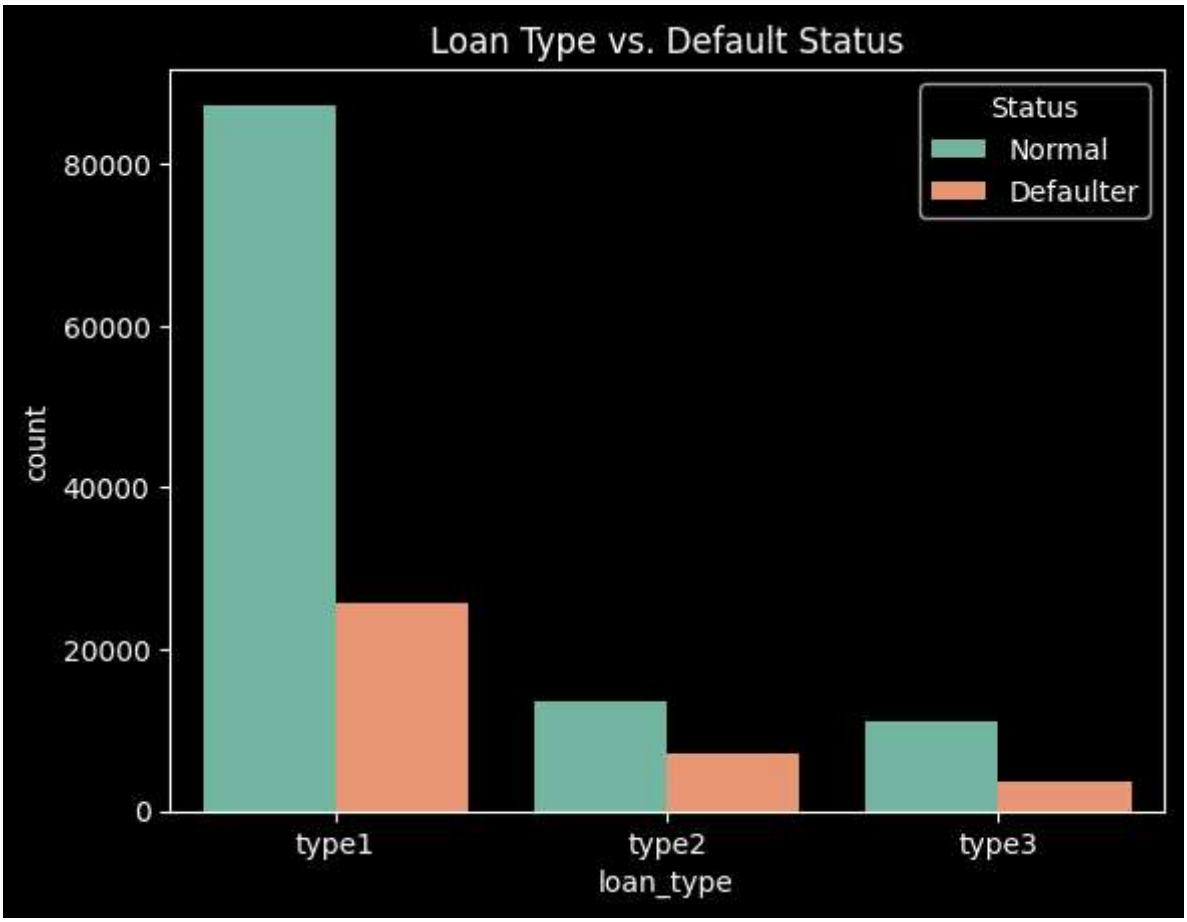
In [190…
```
#Proportion of fixed and variable loan limits
df['loan_limit'].value_counts().plot.pie(autopct='%1.1f%%', colors=['skyblue', 'lig
plt.title('Proportion of Fixed and Variable Loan Limits')
```

Out[190]:
```
Text(0.5, 1.0, 'Proportion of Fixed and Variable Loan Limits')
```



In [191…
```
#Loan types associated with a higher likelihood of default
sns.countplot(x='loan_type', data=df, hue='Status', palette='Set2')
plt.title('Loan Type vs. Default Status')
```

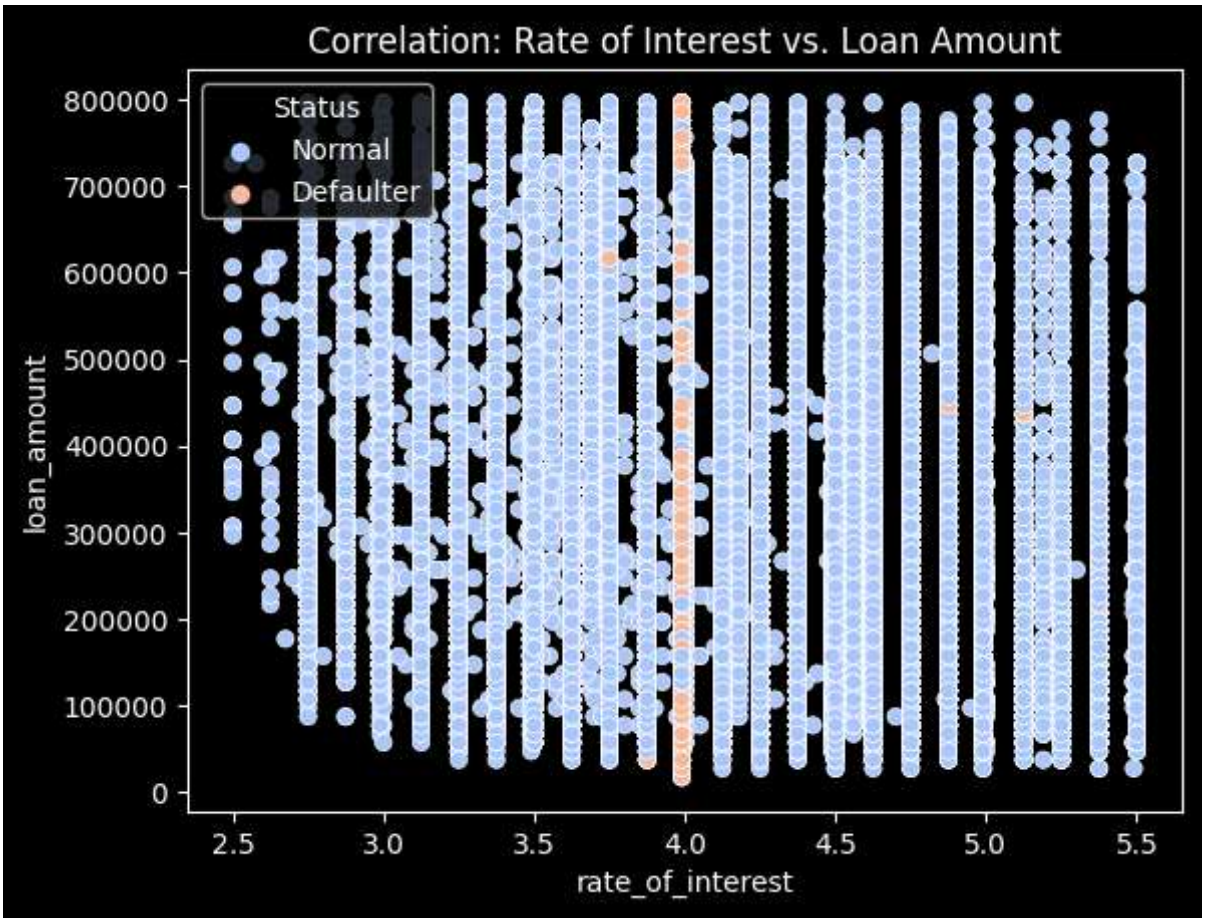
Out[191]:
```
Text(0.5, 1.0, 'Loan Type vs. Default Status')
```



In [192…
```
# Correlation between rate of interest and loan amount
sns.scatterplot(x='rate_of_interest', y='loan_amount', data=df, hue='Status', palet
plt.title('Correlation: Rate of Interest vs. Loan Amount')
```

Out[192]:
```
Text(0.5, 1.0, 'Correlation: Rate of Interest vs. Loan Amount')
```

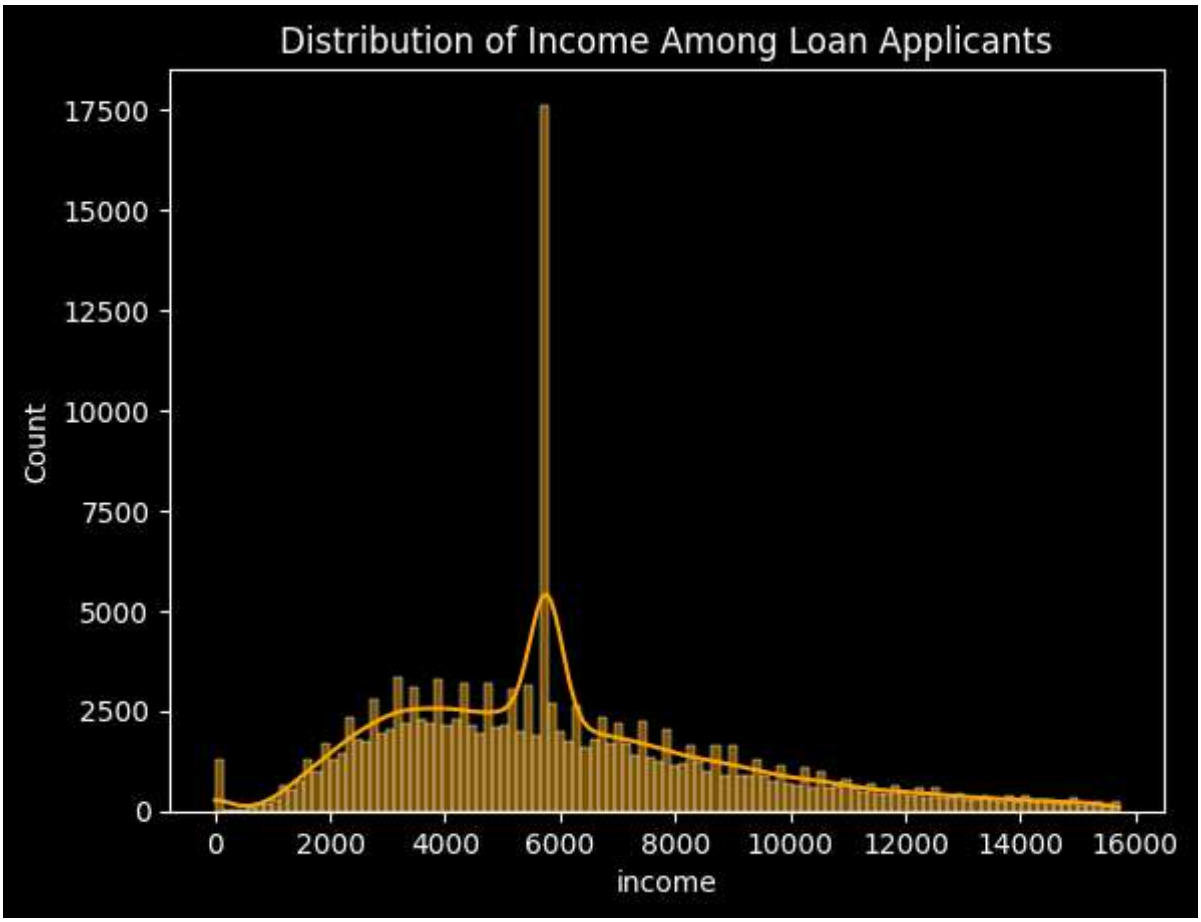
```
C:\Users\sooriyapriya\AppData\Local\Programs\Python\Python310\lib\site-packages\IPyt
hon\core\events.py:89: UserWarning: Creating legend with loc="best" can be slow with
large amounts of data.
  func(*args, **kwargs)
```



Financial assessment

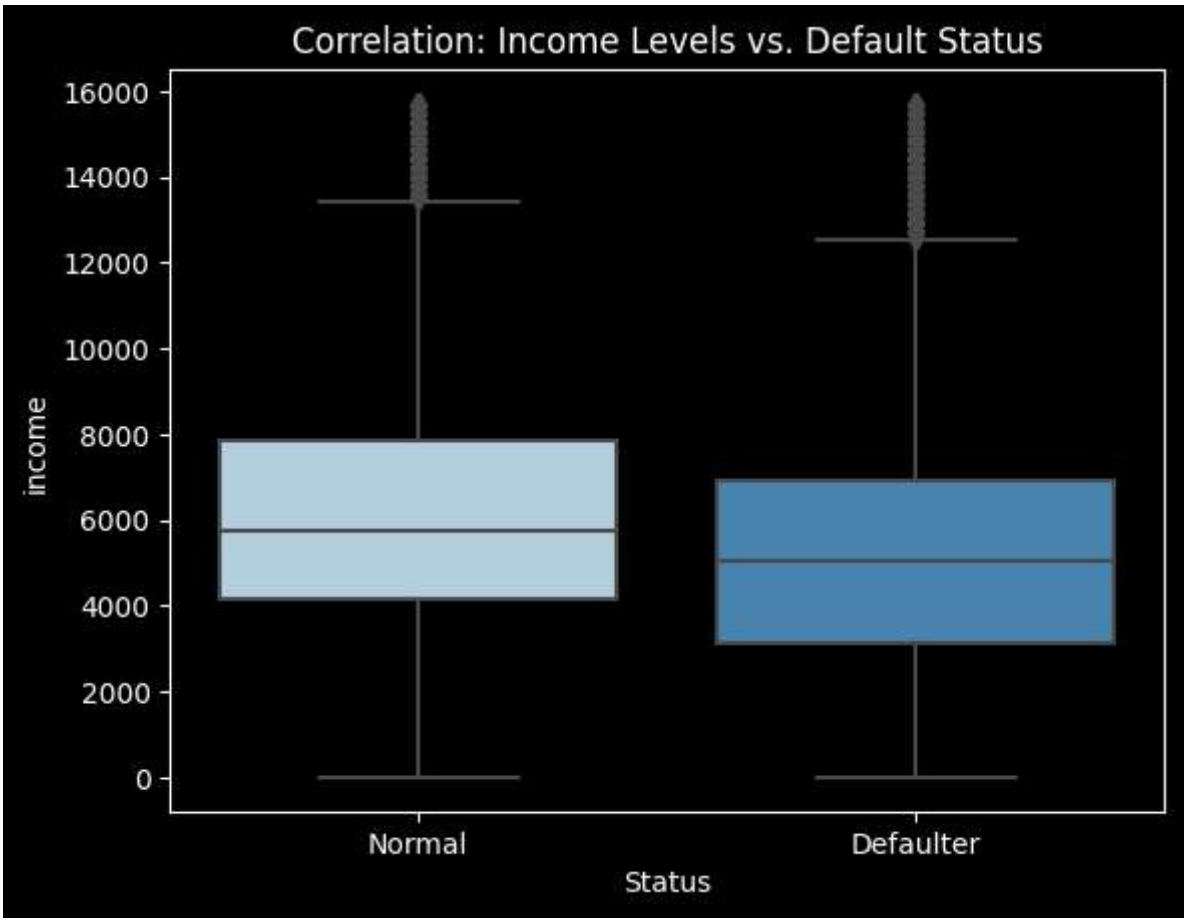
In [193…
```
#Distribution of income among loan applicants
sns.histplot(x='income', data=df, kde=True, color='orange')
plt.title('Distribution of Income Among Loan Applicants')
```

Out[193]:
```
Text(0.5, 1.0, 'Distribution of Income Among Loan Applicants')
```



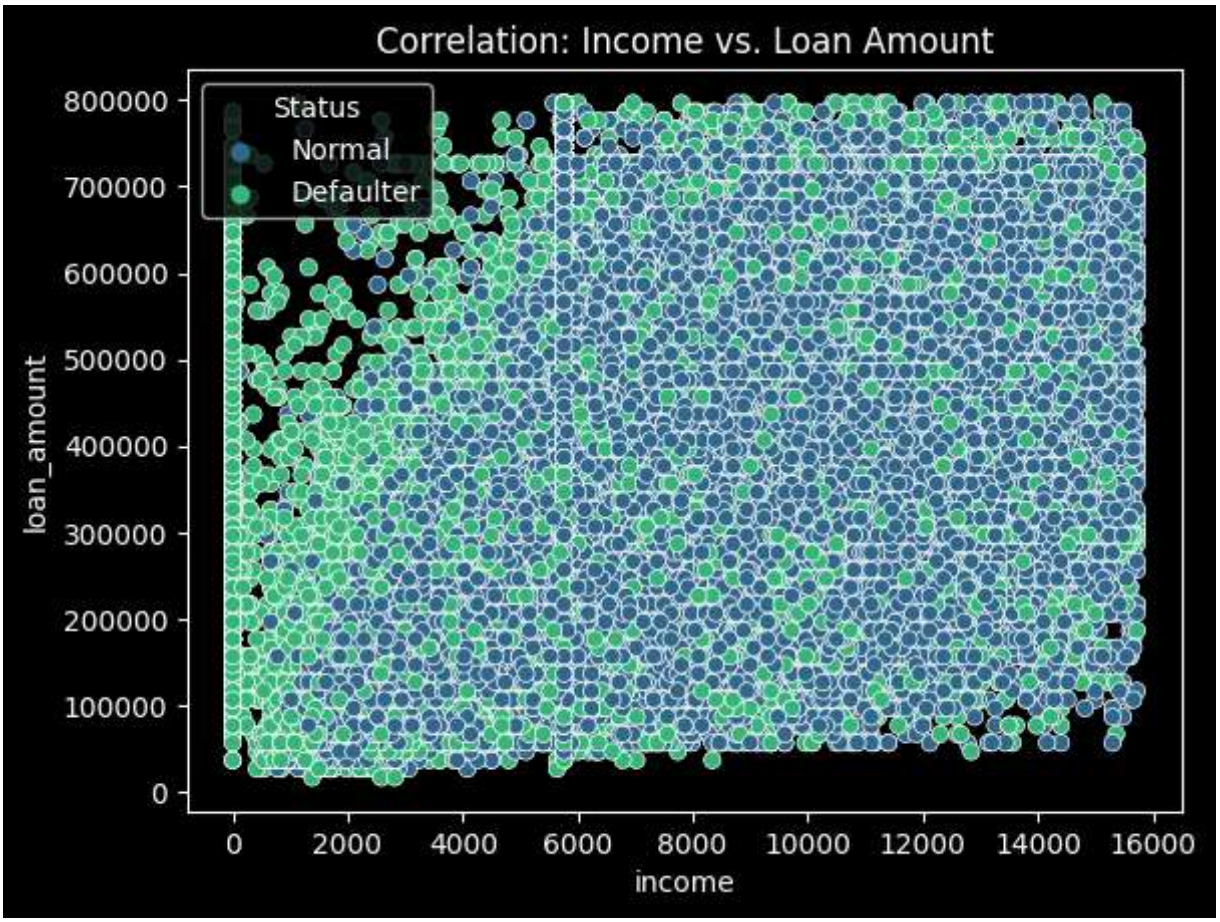
In [194…
```
#Correlation between income levels and default status
sns.boxplot(x='Status', y='income', data=df, palette='Blues')
plt.title('Correlation: Income Levels vs. Default Status')
```

Out[194]:
```
Text(0.5, 1.0, 'Correlation: Income Levels vs. Default Status')
```



In [195…
```
#Variation in loan amount based on income
sns.scatterplot(x='income', y='loan_amount', data=df, hue='Status', palette='viridi
plt.title('Correlation: Income vs. Loan Amount')
```

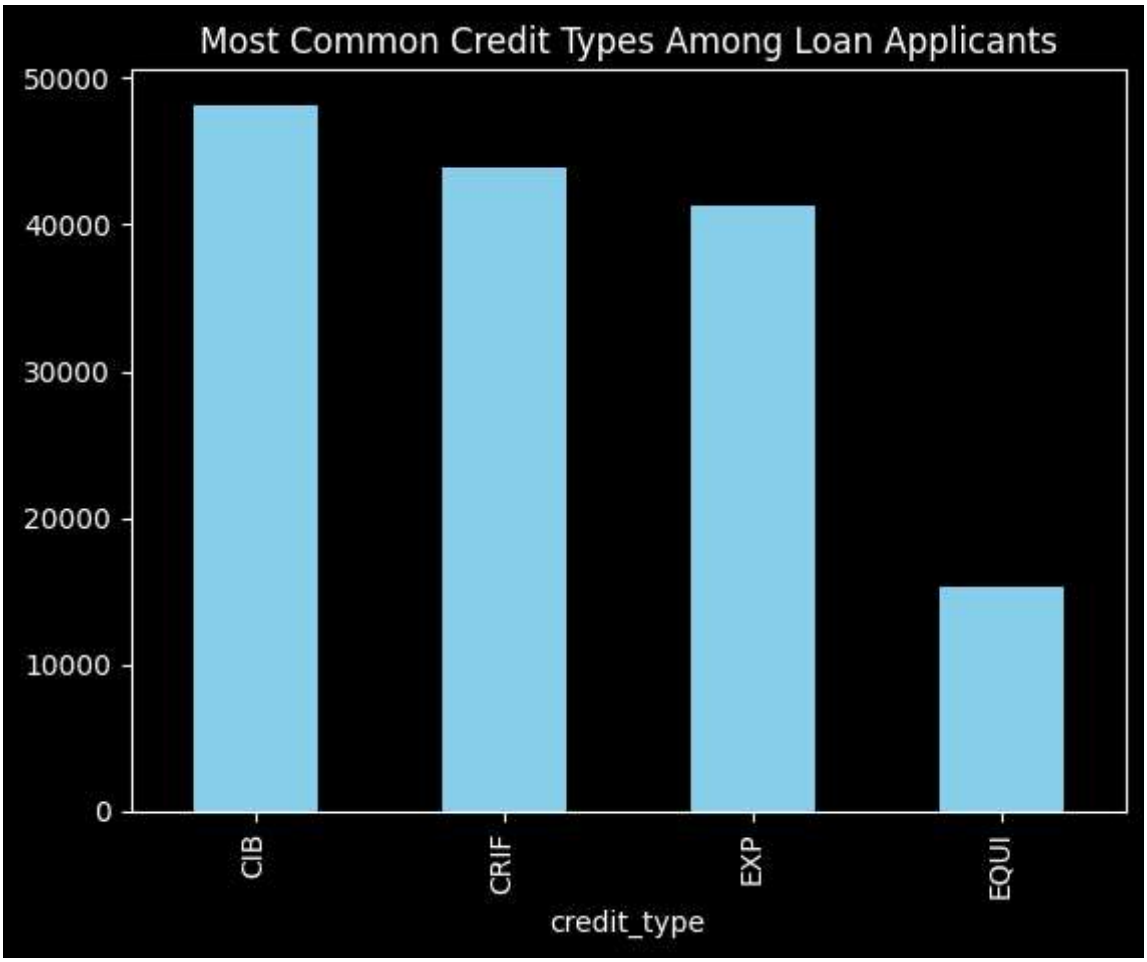
Out[195]:
```
Text(0.5, 1.0, 'Correlation: Income vs. Loan Amount')
```



Credit profiling

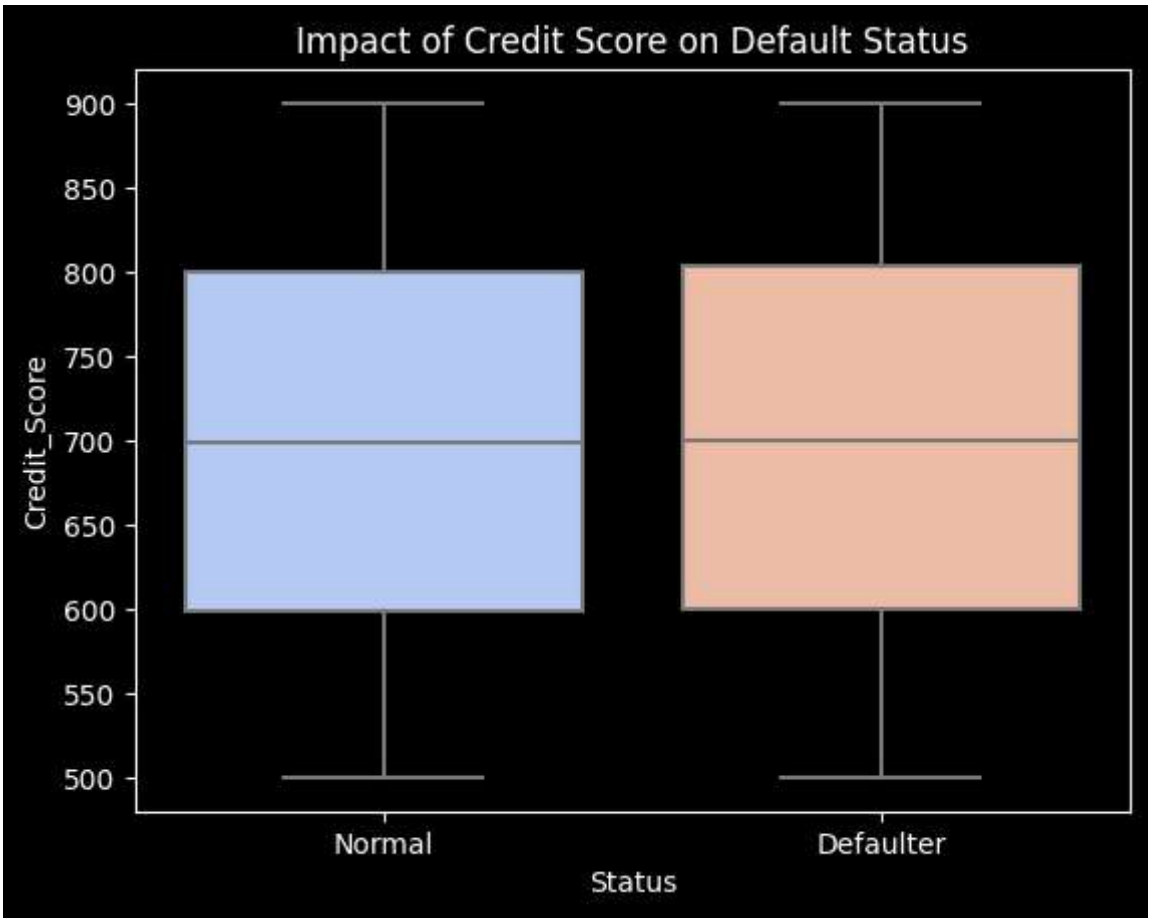
In [196…
```
# Most common credit types among loan applicants
df['credit_type'].value_counts().plot.bar(color='skyblue')
plt.title('Most Common Credit Types Among Loan Applicants')
```

Out[196]:
```
Text(0.5, 1.0, 'Most Common Credit Types Among Loan Applicants')
```



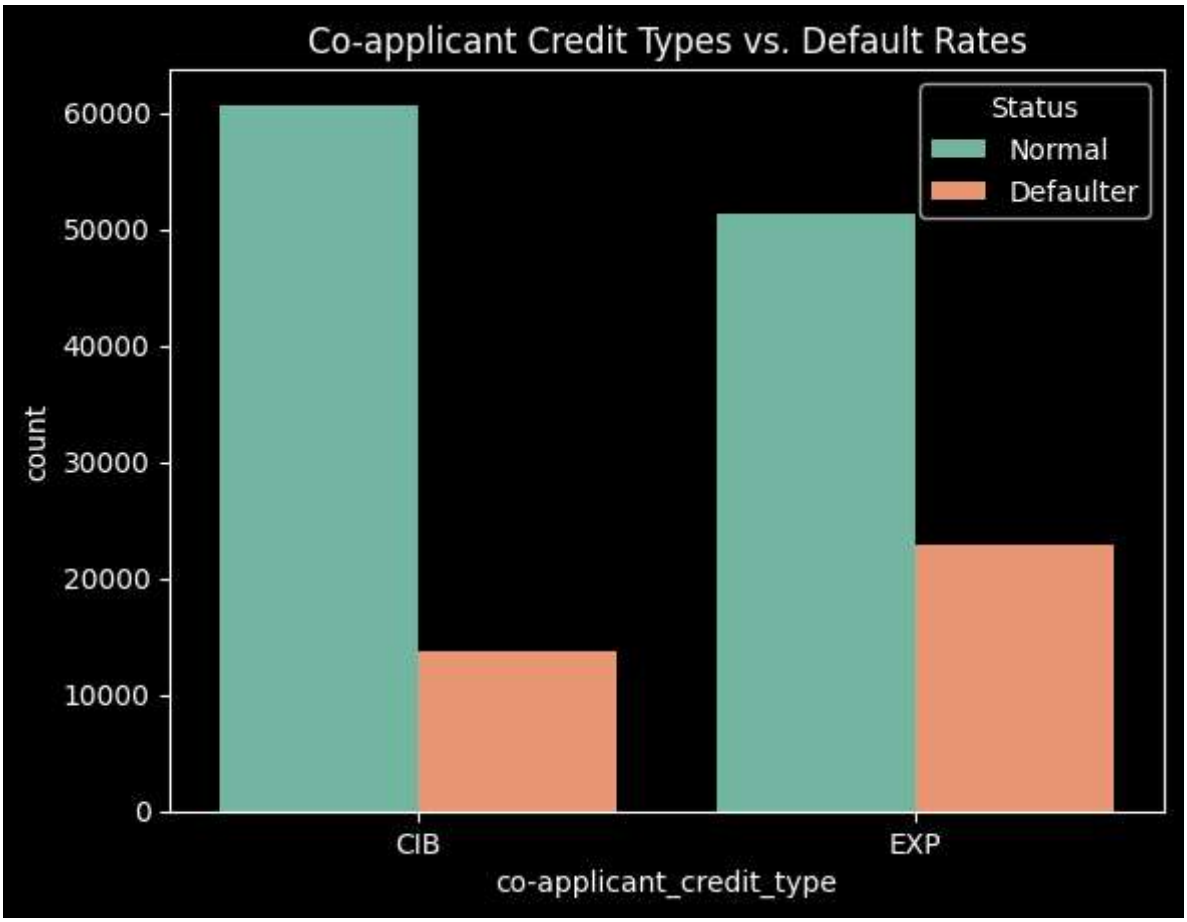
In [197…
```
#Impact of credit score on the likelihood of default
sns.boxplot(x='Status', y='Credit_Score', data=df, palette='coolwarm')
plt.title('Impact of Credit Score on Default Status')
```

Out[197]:
```
Text(0.5, 1.0, 'Impact of Credit Score on Default Status')
```



In [198…
```
#Patterns in co-applicant credit types and default rates
sns.countplot(x='co-applicant_credit_type', data=df, hue='Status', palette='Set2')
plt.title('Co-applicant Credit Types vs. Default Rates')
```

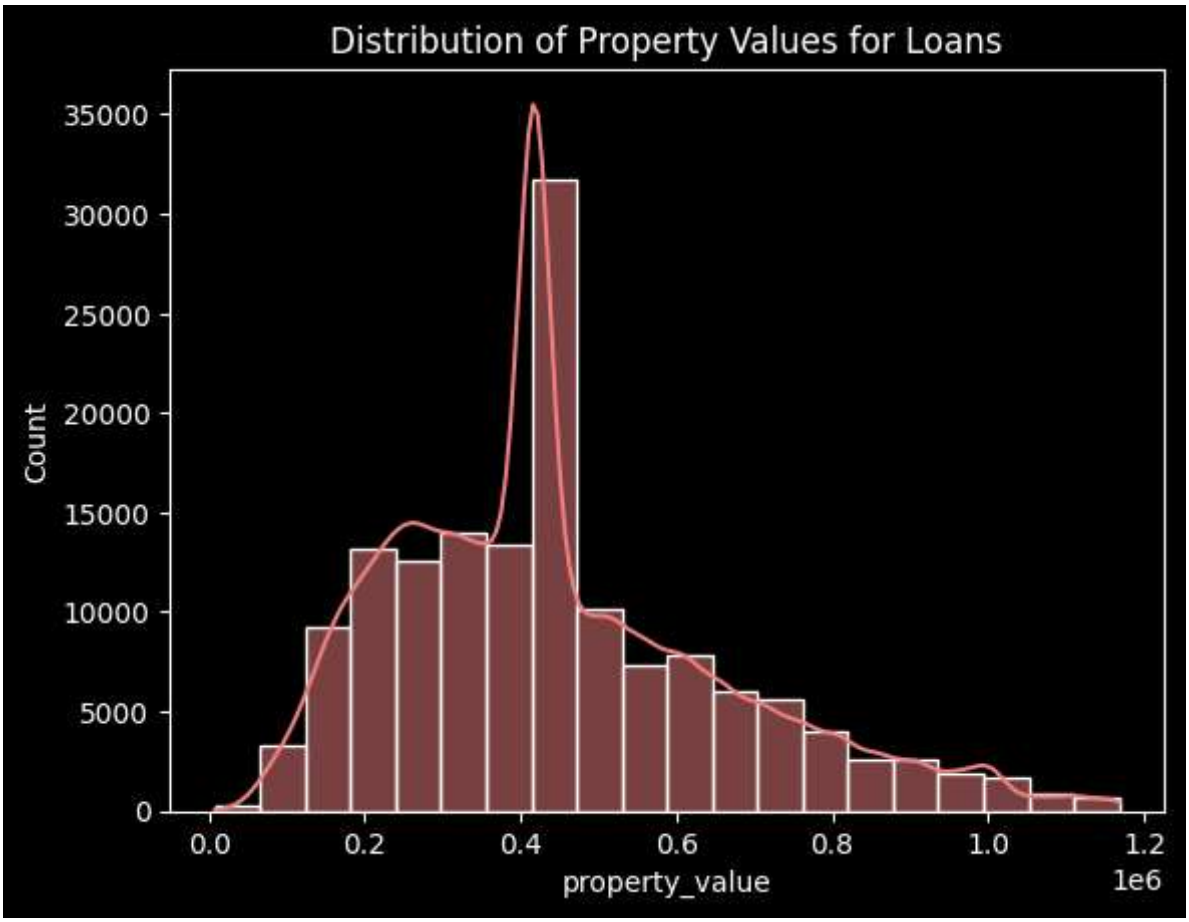
Out[198]:
```
Text(0.5, 1.0, 'Co-applicant Credit Types vs. Default Rates')
```



Property Analysis

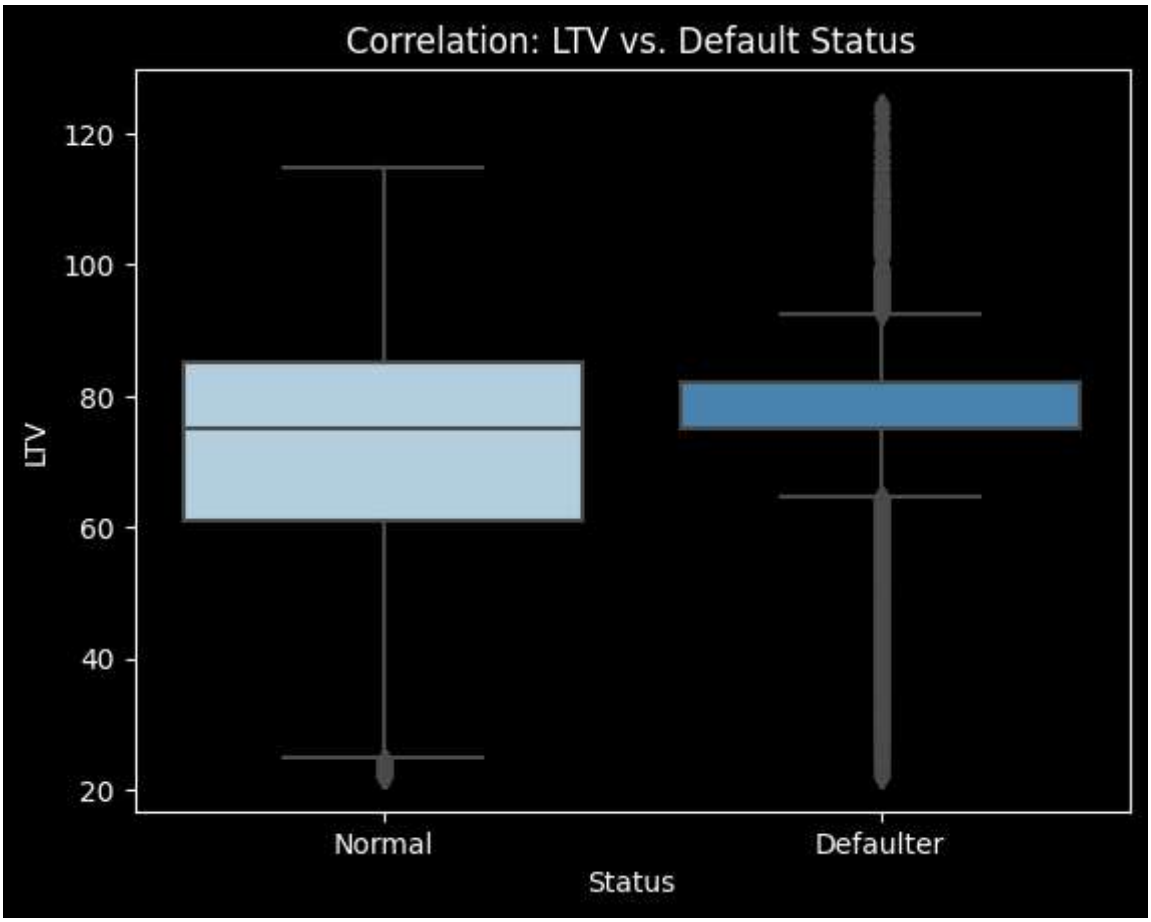
In [199…
```
#Distribution of property values for which loans are taken
sns.histplot(x='property_value', data=df, bins=20, kde=True, color='lightcoral')
plt.title('Distribution of Property Values for Loans')
```

Out[199]:
```
Text(0.5, 1.0, 'Distribution of Property Values for Loans')
```



In [200…
```
#Correlation between LTV and default status
sns.boxplot(x='Status', y='LTV', data=df, palette='Blues')
plt.title('Correlation: LTV vs. Default Status')
```

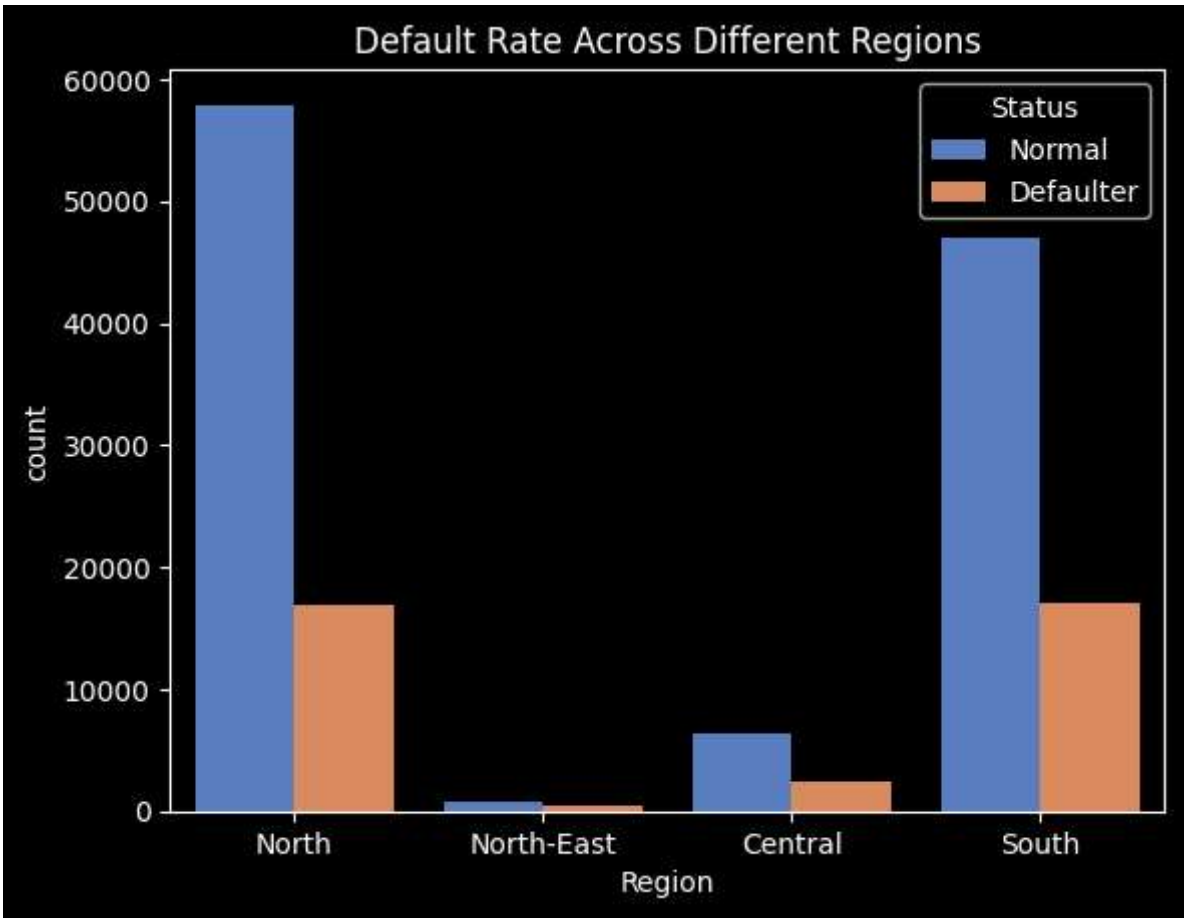
Out[200]:
```
Text(0.5, 1.0, 'Correlation: LTV vs. Default Status')
```



Geographical Analysis

In [201…
```
#Default rate variation across different regions
sns.countplot(x='Region', data=df, hue='Status', palette='muted')
plt.title('Default Rate Across Different Regions')
```

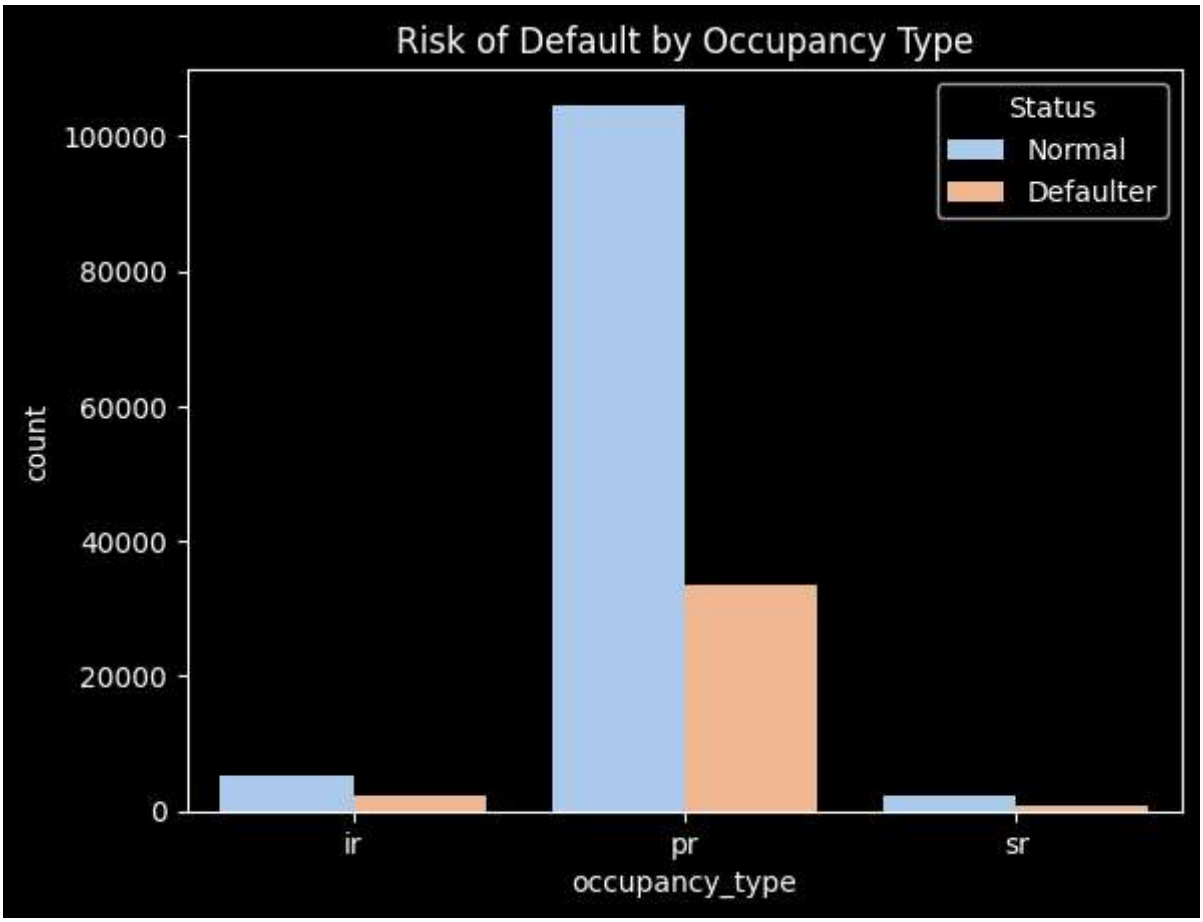
Out[201]:
```
Text(0.5, 1.0, 'Default Rate Across Different Regions')
```



Occupancy Type and Purpose

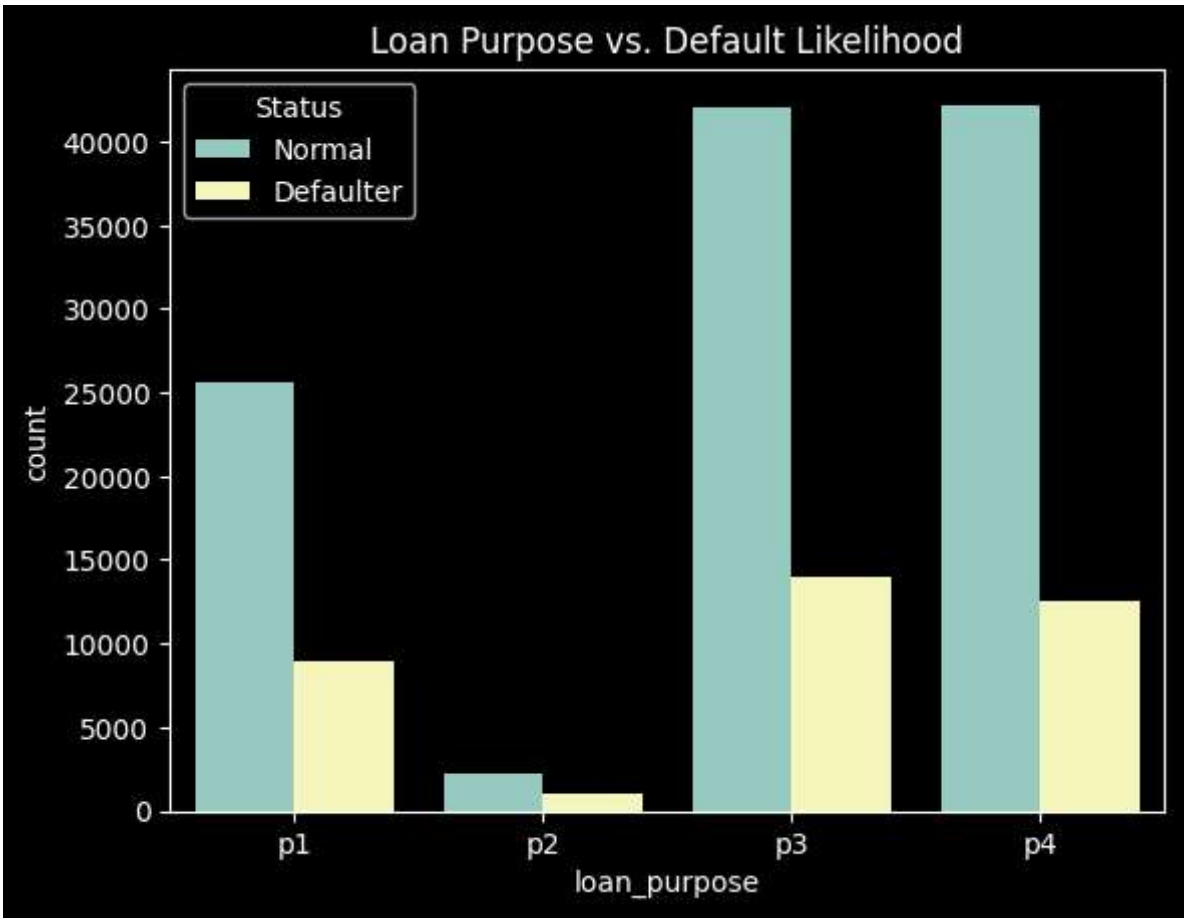
In [202…
```
#Risk of default based on occupancy types
sns.countplot(x='occupancy_type', data=df, hue='Status', palette='pastel')
plt.title('Risk of Default by Occupancy Type')
```

Out[202]:
```
Text(0.5, 1.0, 'Risk of Default by Occupancy Type')
```



In [203…
```
#Impact of loan purpose on default likelihood
sns.countplot(x='loan_purpose', data=df, hue='Status', palette='Set3')
plt.title('Loan Purpose vs. Default Likelihood')
```

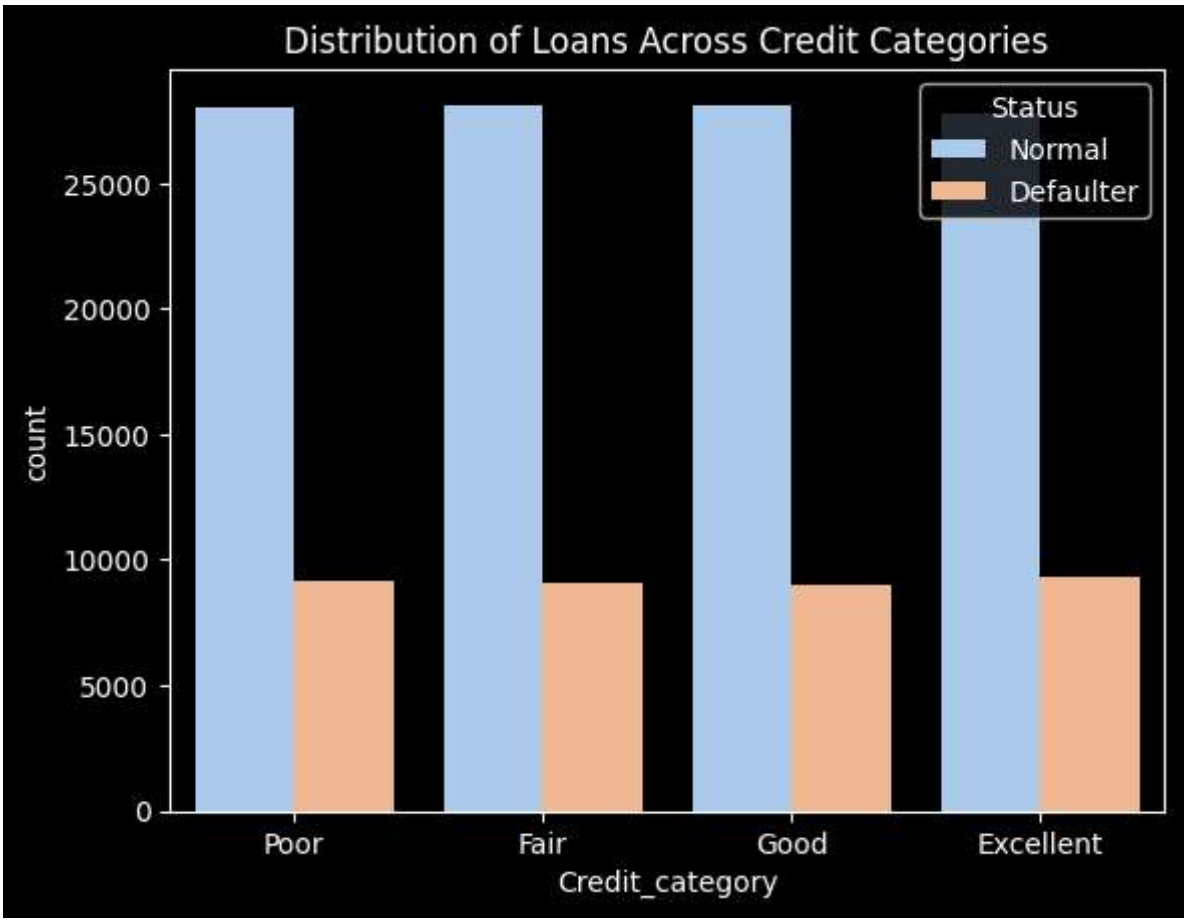
Out[203]:
```
Text(0.5, 1.0, 'Loan Purpose vs. Default Likelihood')
```



Loan Categories

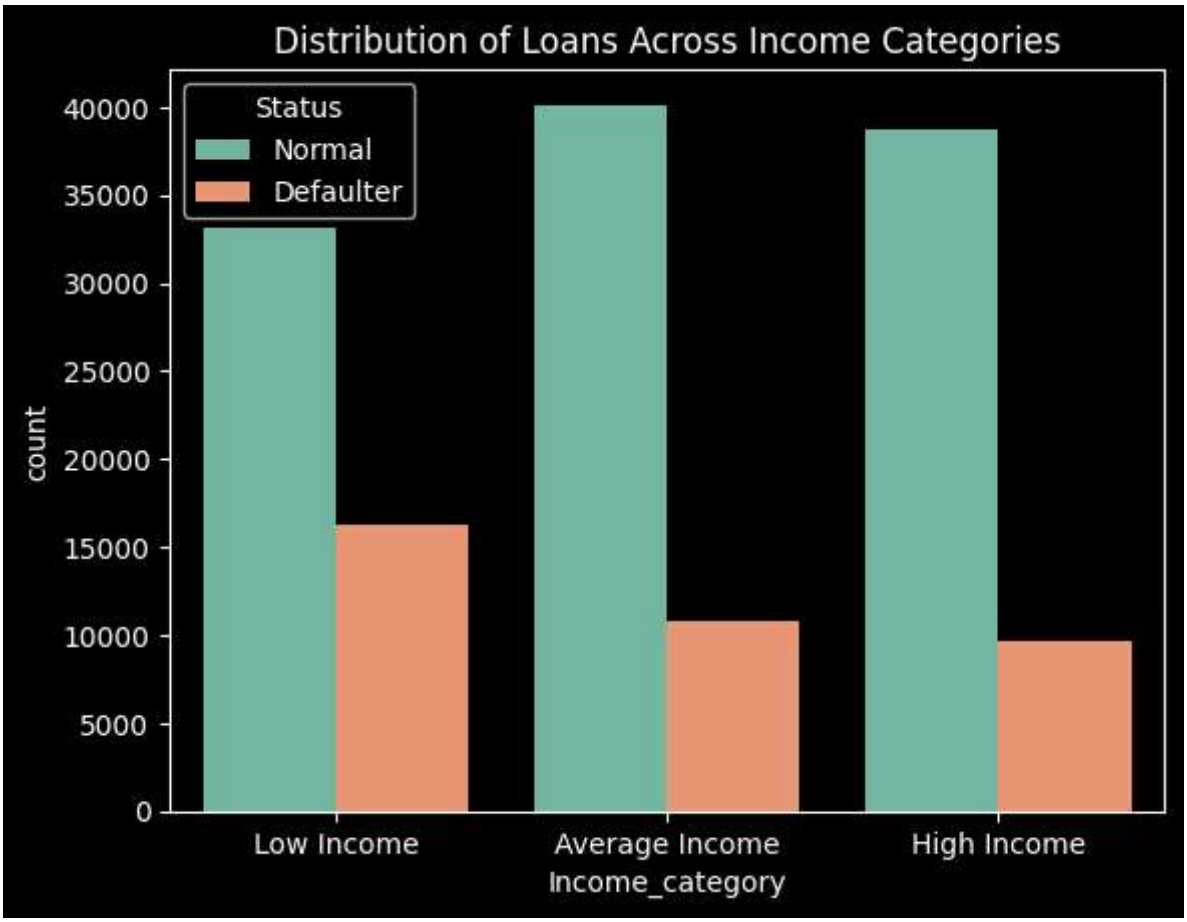
In [204…
```
# How are loans distributed across different categories (e.g., credit category, inc
sns.countplot(x='Credit_category', data=df, hue='Status', palette='pastel')
plt.title('Distribution of Loans Across Credit Categories')
```

Out[204]:
```
Text(0.5, 1.0, 'Distribution of Loans Across Credit Categories')
```



In [205…
```
# Correlation between loan categories and default rates
sns.countplot(x='Income_category', data=df, hue='Status', palette='Set2')
plt.title('Distribution of Loans Across Income Categories')
```

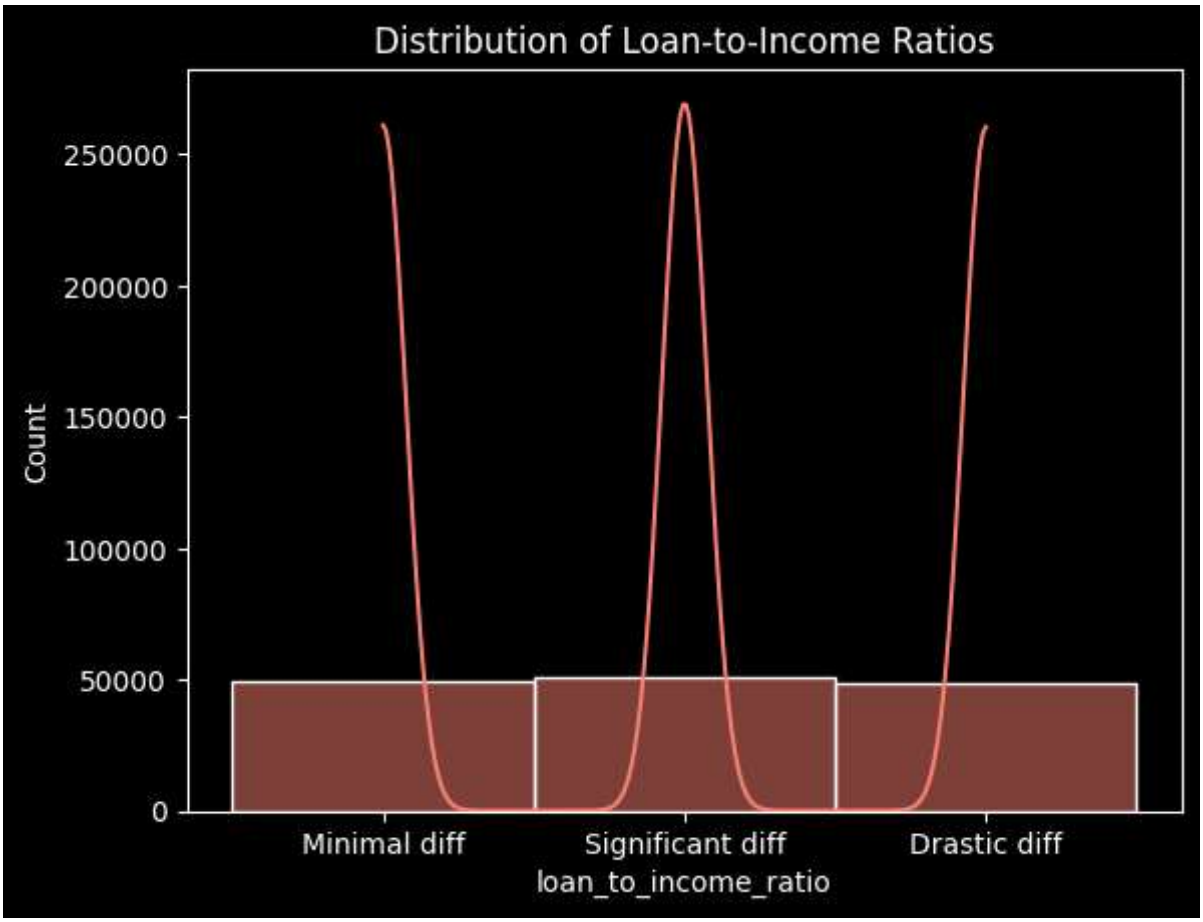
Out[205]:
```
Text(0.5, 1.0, 'Distribution of Loans Across Income Categories')
```



Loan-to-Income Ratio

In [206…
```
#Distribution of loan-to-income ratios among loan applicants
sns.histplot(x='loan_to_income_ratio', data=df, bins=20, kde=True, color='salmon')
plt.title('Distribution of Loan-to-Income Ratios')
```

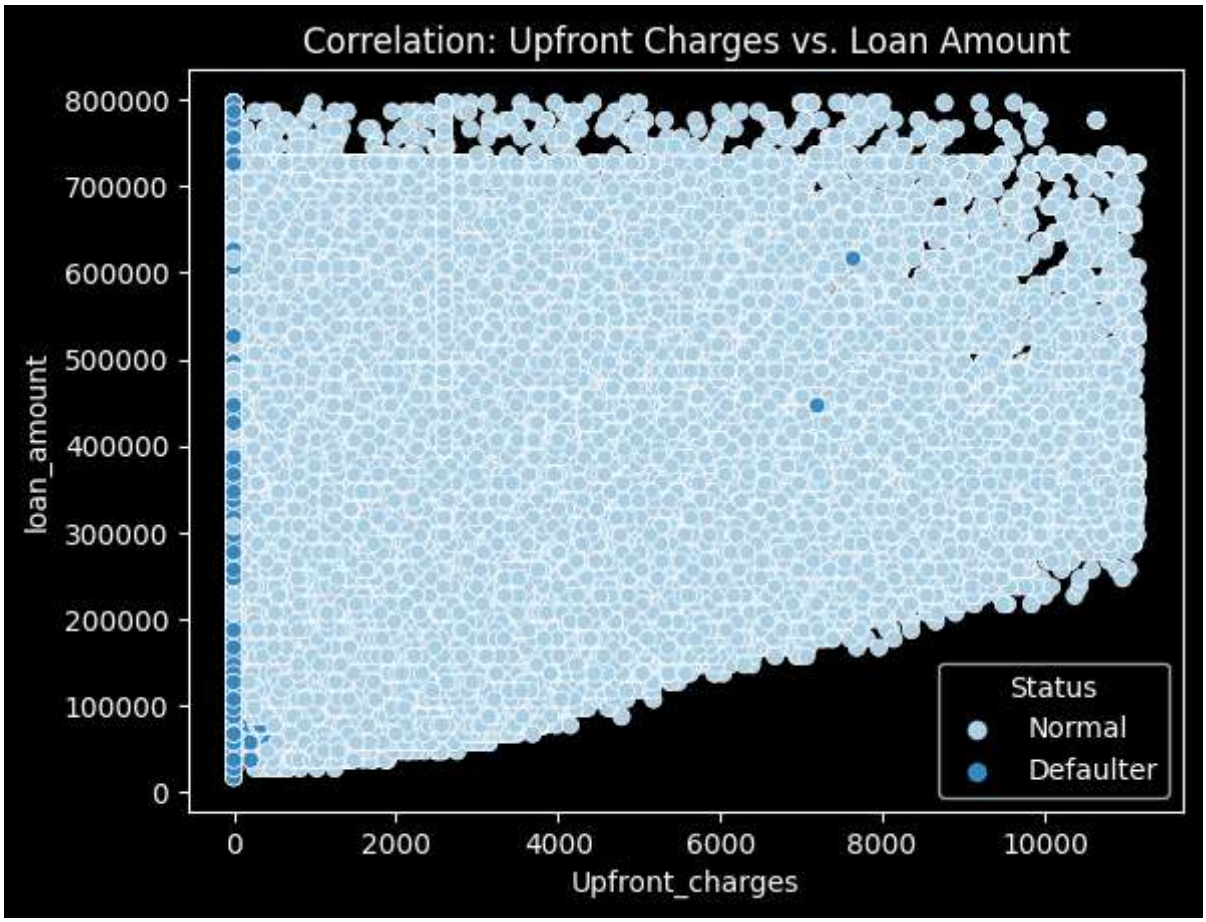
Out[206]:
```
Text(0.5, 1.0, 'Distribution of Loan-to-Income Ratios')
```



Upfront Charges and Financial Behavior

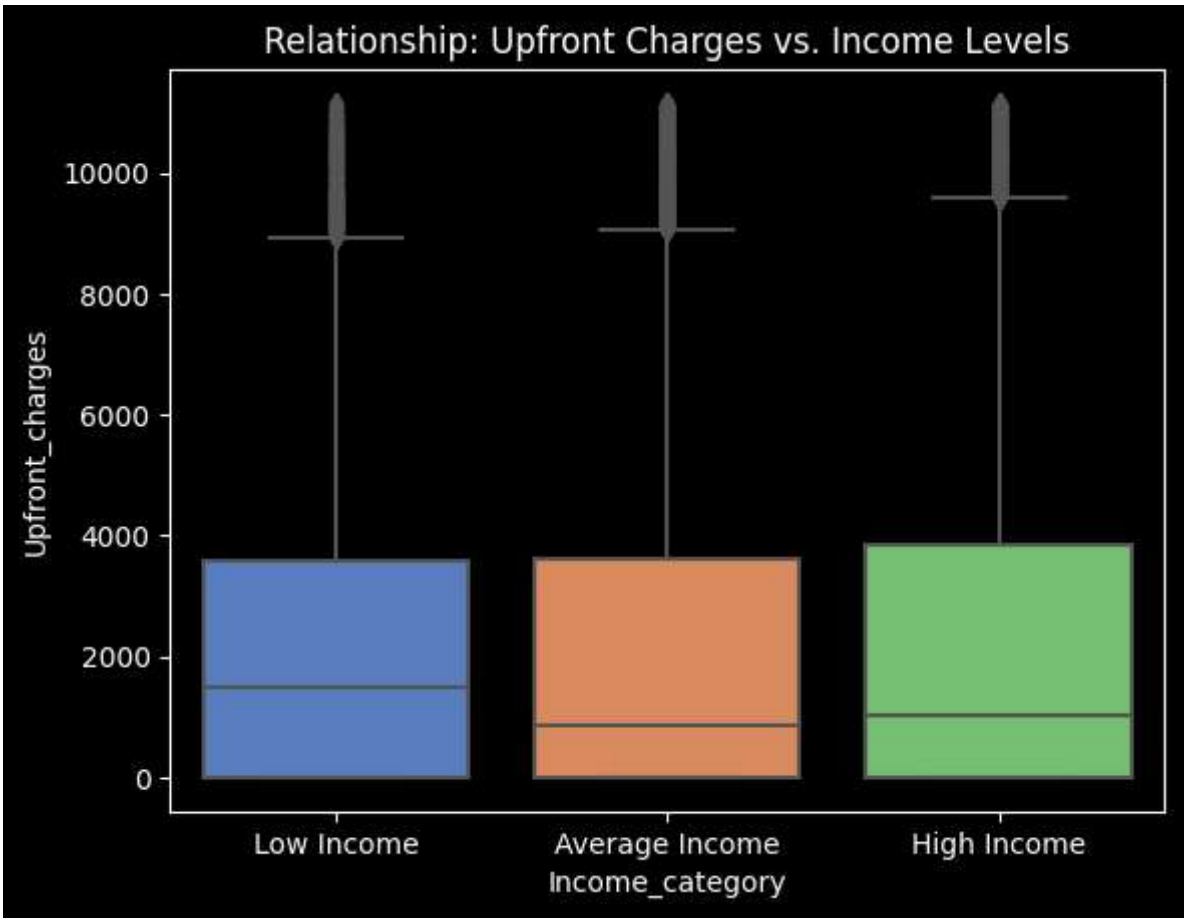
In [207…
```
#Correlation between upfront charges and the default status of applicants
sns.scatterplot(x='Upfront_charges', y='loan_amount', data=df, hue='Status', palett
plt.title('Correlation: Upfront Charges vs. Loan Amount')
```

Out[207]:
```
Text(0.5, 1.0, 'Correlation: Upfront Charges vs. Loan Amount')
```



In [208…
```
# Relationship between upfront charges and income levels
sns.boxplot(x='Income_category', y='Upfront_charges', data=df, palette='muted')
plt.title('Relationship: Upfront Charges vs. Income Levels')
```

Out[208]:
```
Text(0.5, 1.0, 'Relationship: Upfront Charges vs. Income Levels')
```



Default Rates Across Different Regions:

Null Hypothesis (H0): There is no significant difference in default rates across different regions.

Alternative Hypothesis (H1): There is a significant difference in default rates across different regions.

Statistical Test: Chi-square test for independence (Categorical vs. Categorical).

In [209…

```
from scipy.stats import chi2_contingency
contingency_table = pd.crosstab(df['Region'], df['Status'])

chi2, p, _, _ = chi2_contingency(contingency_table)

alpha = 0.05
if p < alpha:
    print("Reject the null hypothesis: There is a significant difference in default
else:
    print("Fail to reject the null hypothesis: No significant difference in default
```

```
Reject the null hypothesis: There is a significant difference in default rates acros
s regions.
```

Exploring the Relationship Between Age Groups and Default Rates

Null Hypothesis (H0): There is no significant difference in default rates across different age groups.

Alternative Hypothesis (H1): There is a significant difference in default rates across different age groups.

Statistical Test: Chi-Square Test for Independence

In [210…
```
import pandas as pd
from scipy.stats import chi2_contingency

contingency_table = pd.crosstab(df['age'], df['Status'])

chi2, p, _, _ = chi2_contingency(contingency_table)

print(f"Chi-Square Value: {chi2}")
print(f"P-value: {p}")

alpha = 0.05
if p < alpha:
    print("Reject the null hypothesis: There is a significant difference in default
else:
    print("Fail to reject the null hypothesis: No significant difference in default
```

```
Chi-Square Value: 368.10143385295123
P-value: 2.001490250998746e-76
Reject the null hypothesis: There is a significant difference in default rates acros
s age groups.
```

Correlation Between Loan Amount and Upfront Charges:

Null Hypothesis (H0): There is no correlation between loan amount and upfront charges.

Alternative Hypothesis (H1): There is a correlation between loan amount and upfront charges.

Statistical Test: Pearson correlation coefficient (Continuous vs. Continuous).

In [211…
```
from scipy.stats import pearsonr

corr_coefficient, p_value = pearsonr(df['loan_amount'], df['Upfront_charges'])

alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: There is a correlation between loan amount a
else:
    print("Fail to reject the null hypothesis: No correlation between loan amount a
```

```
Reject the null hypothesis: There is a correlation between loan amount and upfront c
harges.
```

Comparing Default Rates Between Different Credit Categories:

Null Hypothesis (H0): There is no significant difference in default rates between different credit categories.

Alternative Hypothesis (H1): There is a significant difference in default rates between different credit categories.

Statistical Test: One-way ANOVA (Analysis of Variance) or Kruskal-Wallis test (if assumptions are not met).

In [212…
```
# Create a contingency table for credit categories and default status
contingency_table = pd.crosstab(df['Credit_category'], df['Status'])

# Perform the chi-square test for independence
chi2_stat, p_value, _, _ = chi2_contingency(contingency_table)

# Check the p-value against alpha
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: There is a significant difference in default
else:
    print("Fail to reject the null hypothesis: No significant difference in default
```

```
Reject the null hypothesis: There is a significant difference in default rates betwe
en credit categories.
```

Comparing Loan Amounts Across Different Income Categories:

Null Hypothesis (H0): There is no significant difference in loan amounts across different income categories.

Alternative Hypothesis (H1): There is a significant difference in loan amounts across different income categories.

Statistical Test: Kruskal-Wallis test.

In [213…
```
from scipy.stats import kruskal

# Group data by Income Category
groups_income = [df[df['Income_category'] == category]['loan_amount'] for category

# Perform Kruskal-Wallis test
statistic_income, p_value_income = kruskal(*groups_income)

# Check the p-value against alpha
alpha = 0.05
if p_value_income < alpha:
    print("Reject the null hypothesis: There is a significant difference in loan am
else:
    print("Fail to reject the null hypothesis: No significant difference in loan am
```

```
Reject the null hypothesis: There is a significant difference in loan amounts across
income categories.
```

Testing the Equality of Variance for Loan Amounts Between Default and Non-Default Groups:

Null Hypothesis (H0): There is no significant difference in variance of loan amounts between default and non-default groups.

Alternative Hypothesis (H1): There is a significant difference in variance of loan amounts between default and non-default groups.

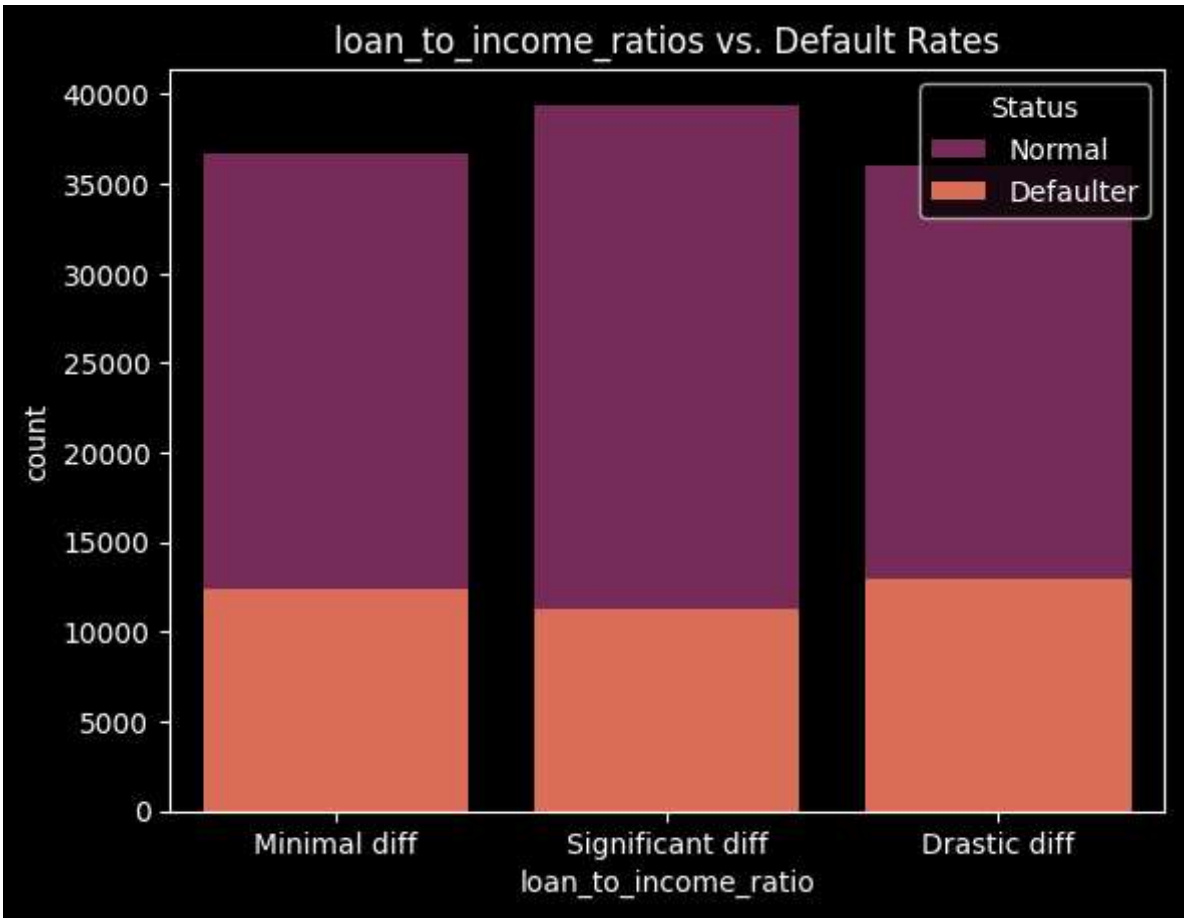
Statistical Test: Levene's test (Test for equality of variances).

In [214…
```
from scipy.stats import levene

# Perform Levene's test for equality of variances
statistic_levene, p_value_levene = levene(df[df['Status'] == 1]['loan_amount'], df[

# Check the p-value against alpha
if p_value_levene < alpha:
    print("Reject the null hypothesis: There is a significant difference in varianc
else:
    print("Fail to reject the null hypothesis: No significant difference in varianc
```

```
Fail to reject the null hypothesis: No significant difference in variance of loan am
ounts between default and non-default groups.
C:\Users\sooriyapriya\AppData\Local\Programs\Python\Python310\lib\site-packages\nump
y\core\fromnumeric.py:3504: RuntimeWarning: Mean of empty slice.
  return _methods._mean(a, axis=axis, dtype=dtype,
C:\Users\sooriyapriya\AppData\Local\Programs\Python\Python310\lib\site-packages\nump
y\core\_methods.py:129: RuntimeWarning: invalid value encountered in scalar divide
  ret = ret.dtype.type(ret / rcount)
```

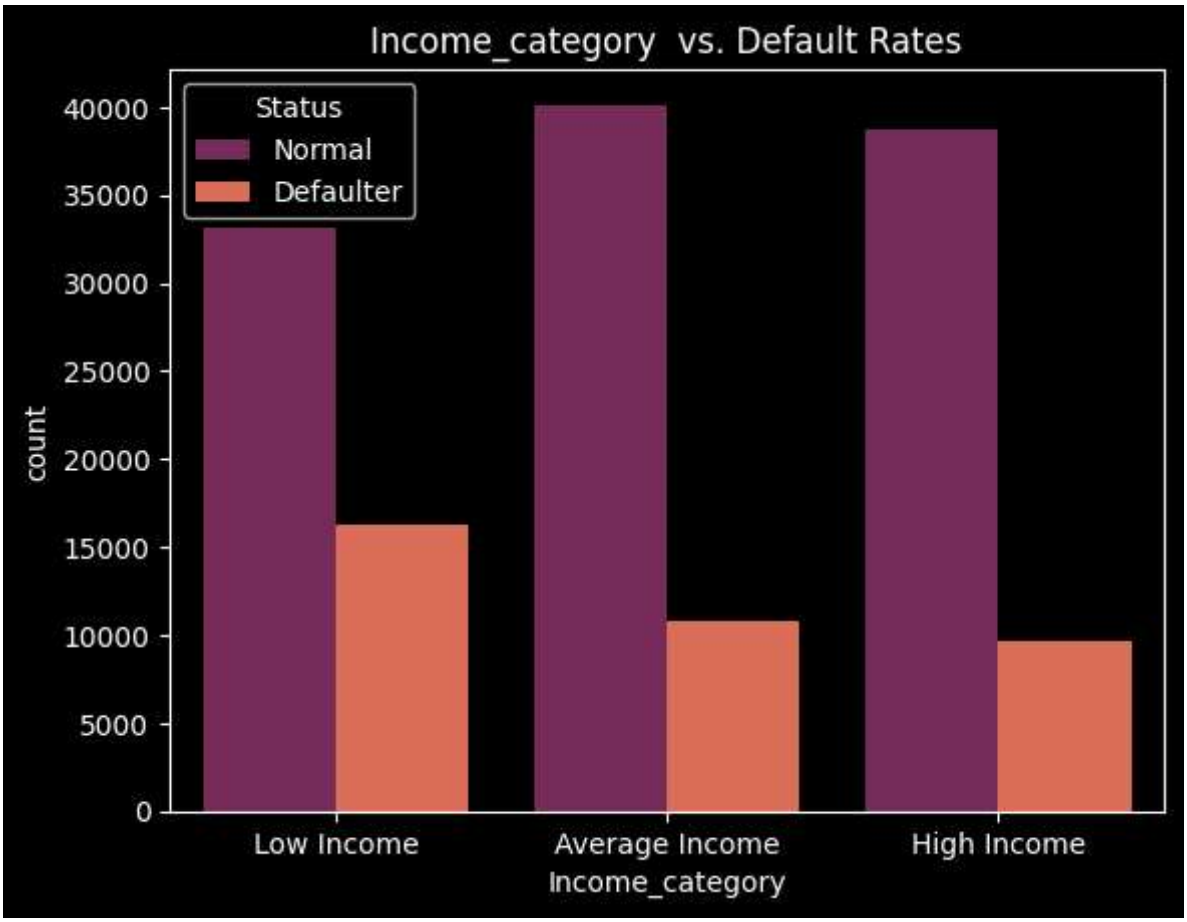
In [216…
```
sns.countplot(x='loan_to_income_ratio', hue='Status', data=df, dodge=False, palette
plt.title('loan_to_income_ratios vs. Default Rates')
```

Out[216]:
```
Text(0.5, 1.0, 'loan_to_income_ratios vs. Default Rates')
```



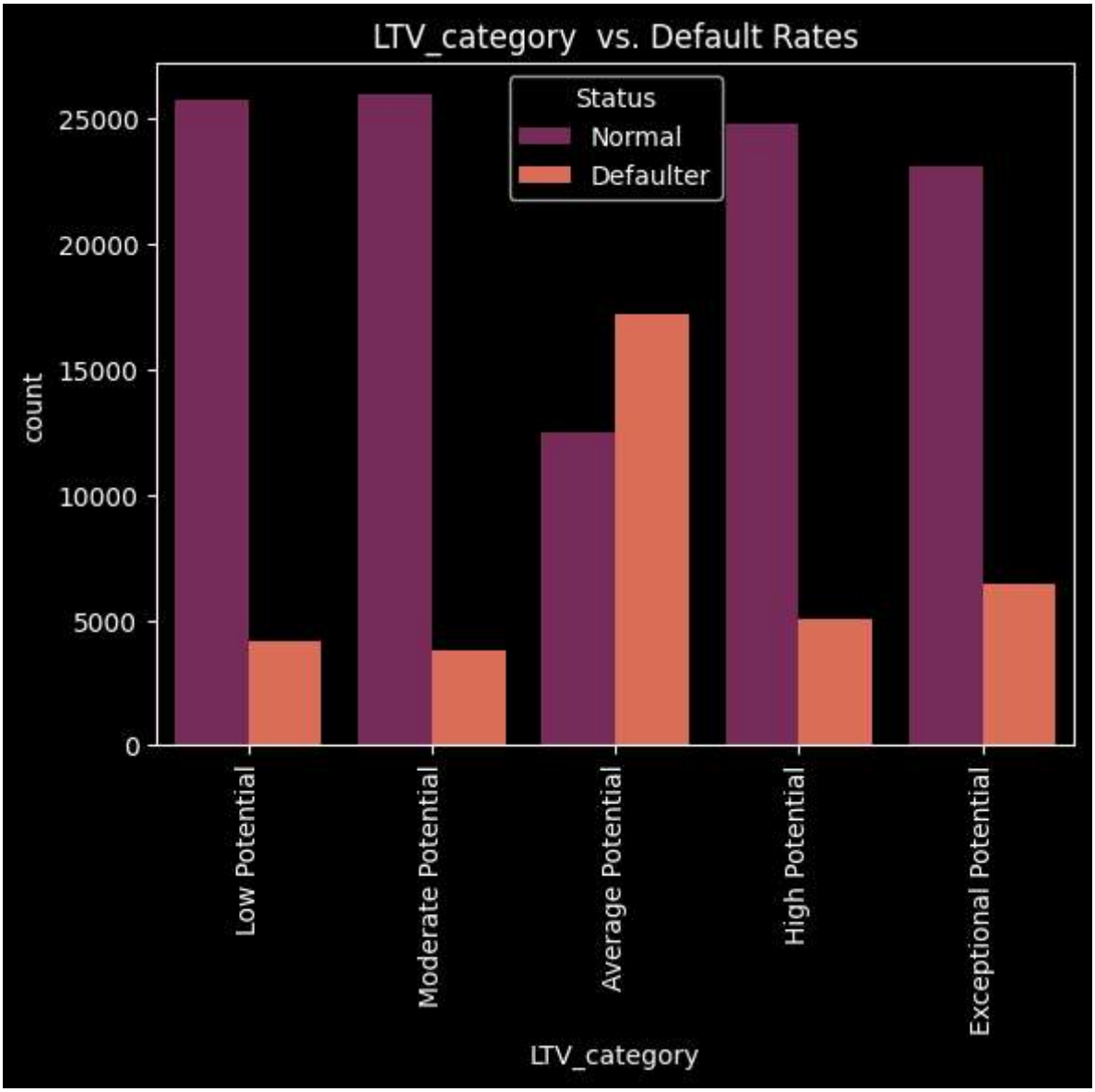
In [228…
```
sns.countplot(x='Income_category', data=df, hue='Status', palette='rocket')
plt.title('Income_category  vs. Default Rates')
```

Out[228]:
```
Text(0.5, 1.0, 'Income_category  vs. Default Rates')
```



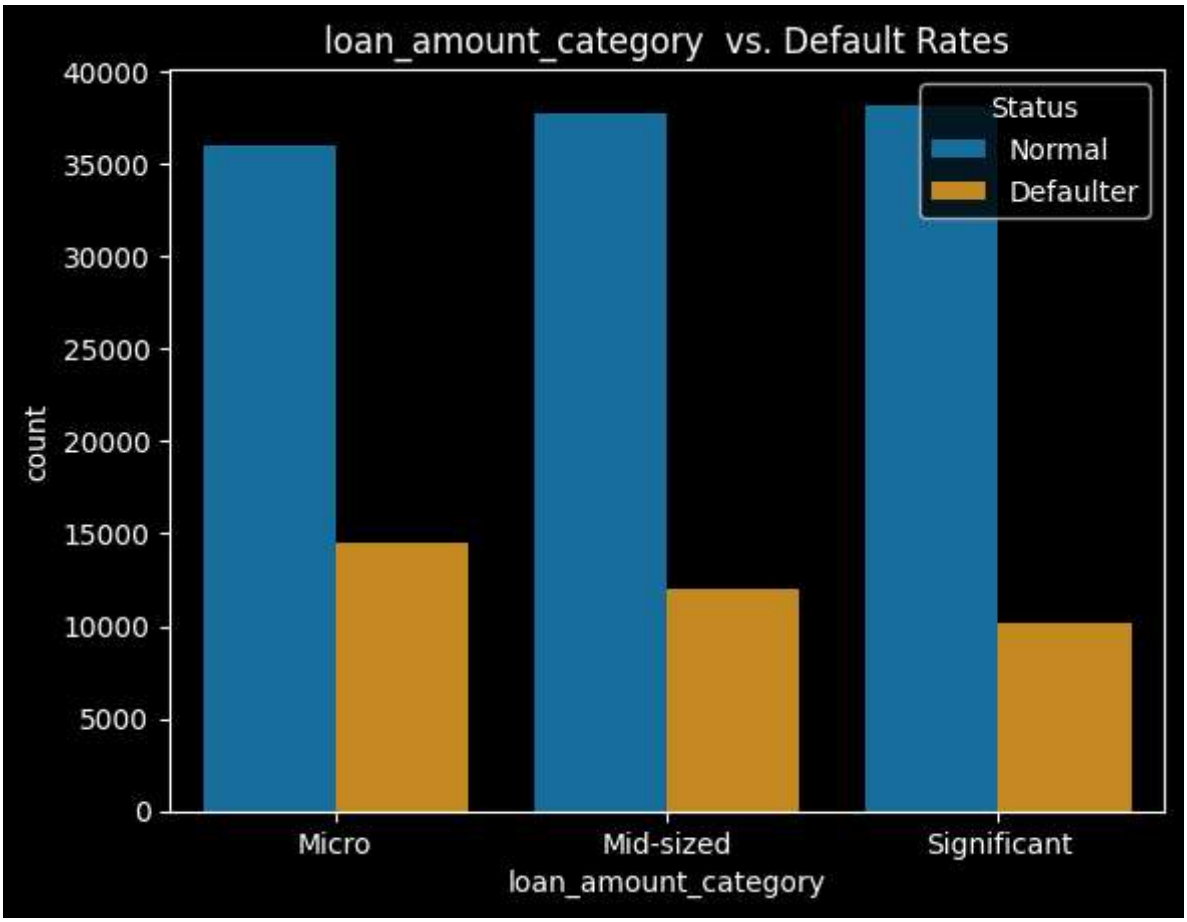
In [227…
```
sns.countplot(x='LTV_category', data=df, hue='Status', palette='rocket')
plt.title('LTV_category  vs. Default Rates')
plt.xticks(rotation='vertical')
```

Out[227]:
```
(array([0, 1, 2, 3, 4]),
 [Text(0, 0, 'Low Potential'),
  Text(1, 0, 'Moderate Potential'),
  Text(2, 0, 'Average Potential'),
  Text(3, 0, 'High Potential'),
  Text(4, 0, 'Exceptional Potential')])
```



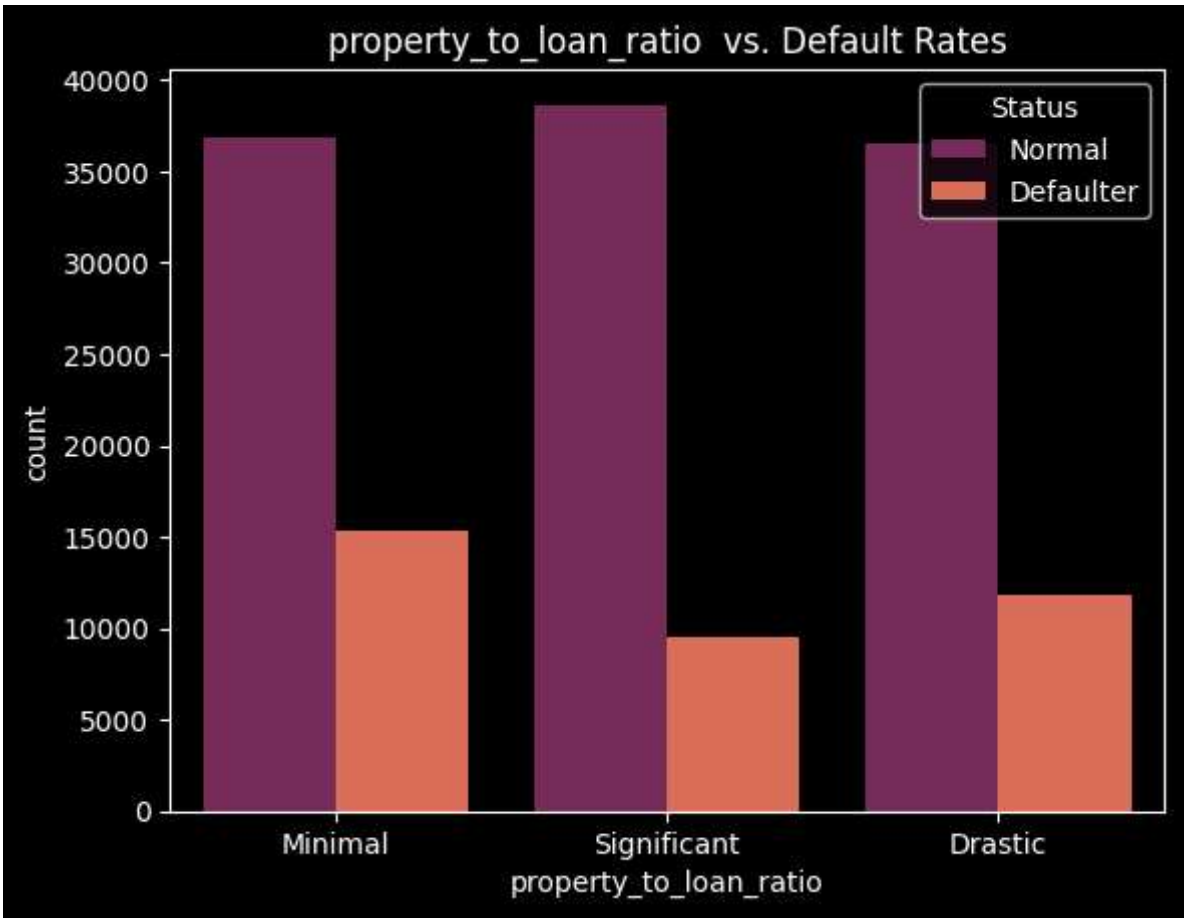
In [218…
```
sns.countplot(x='loan_amount_category', data=df, hue='Status', palette='colorblind'
plt.title('loan_amount_category  vs. Default Rates')
```

Out[218]:
```
Text(0.5, 1.0, 'loan_amount_category  vs. Default Rates')
```



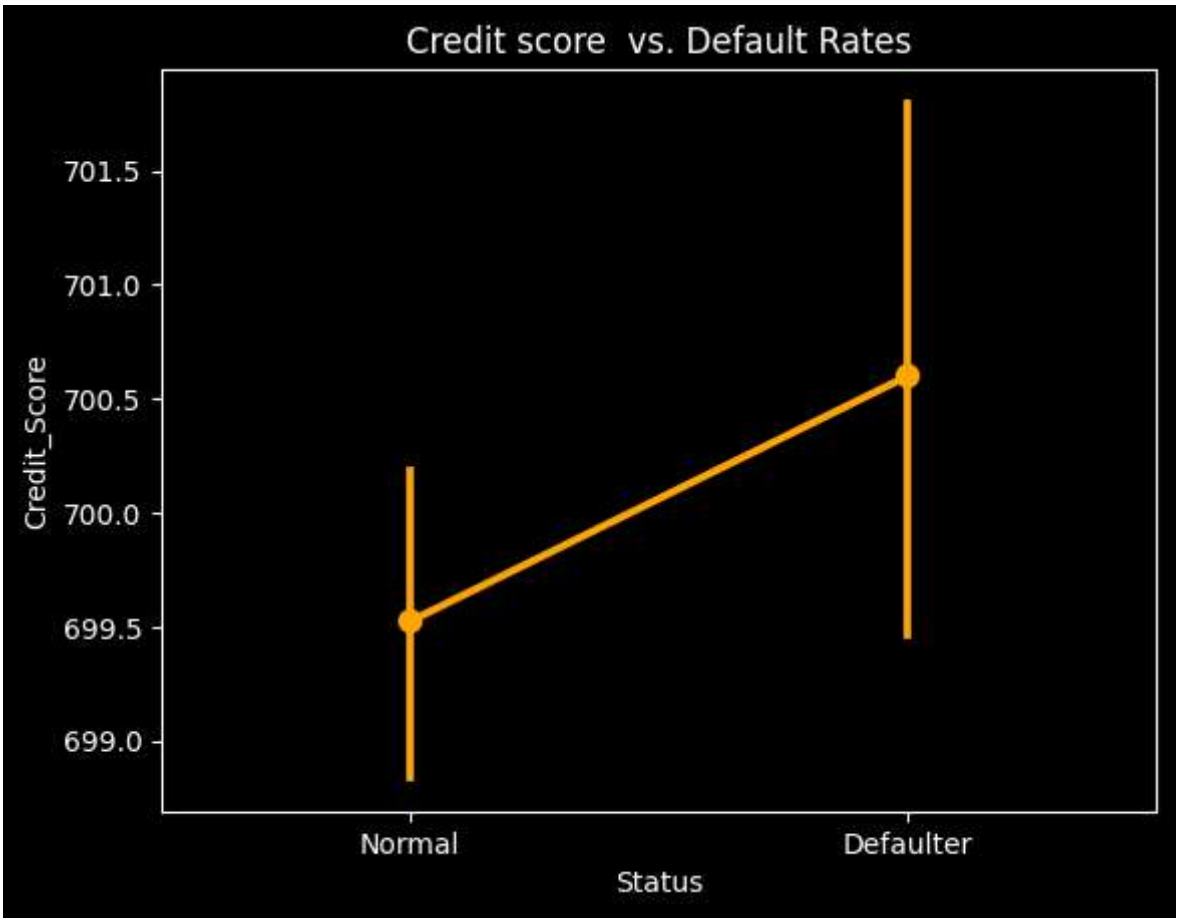
In [226…
```
sns.countplot(x='property_to_loan_ratio', data=df, hue='Status', palette='rocket')
plt.title('property_to_loan_ratio  vs. Default Rates')
```

Out[226]:
```
Text(0.5, 1.0, 'property_to_loan_ratio  vs. Default Rates')
```



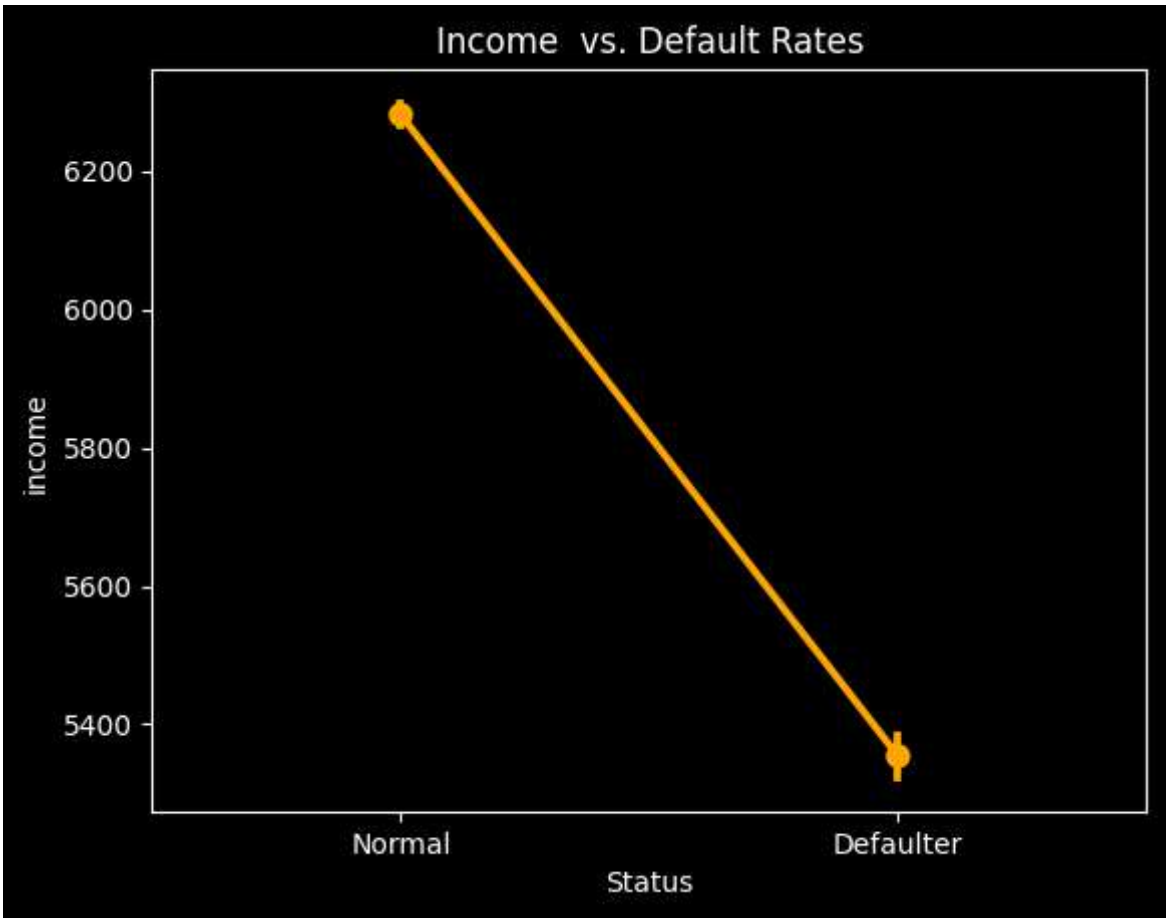
In [225…

```
sns.pointplot(x='Status', y='Credit_Score', data=df, color='orange')  # You can cha
plt.title('Credit score  vs. Default Rates')
plt.show()
```



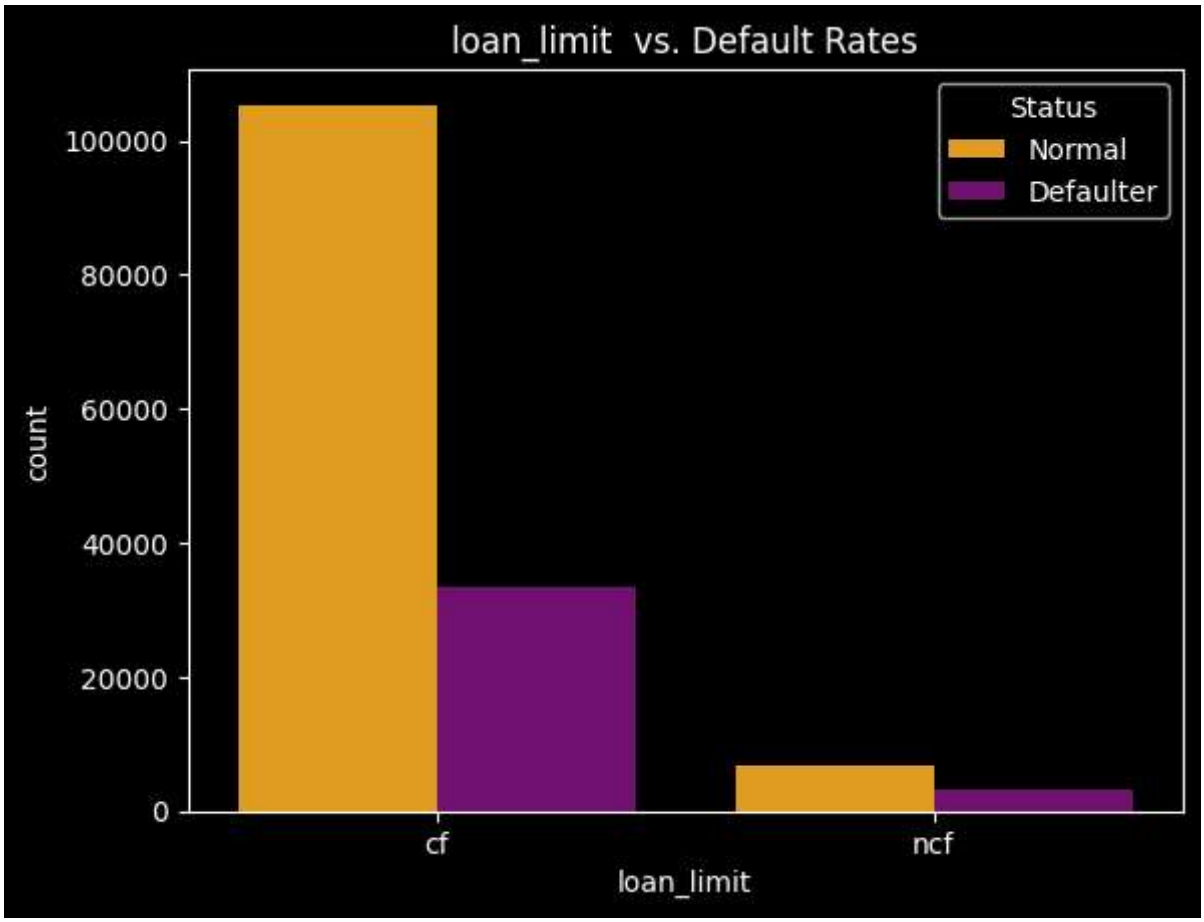
In [223…

```
sns.pointplot(x='Status', y='income', data=df, color='orange')  # You can change 's
plt.title('Income  vs. Default Rates')
plt.show()
```



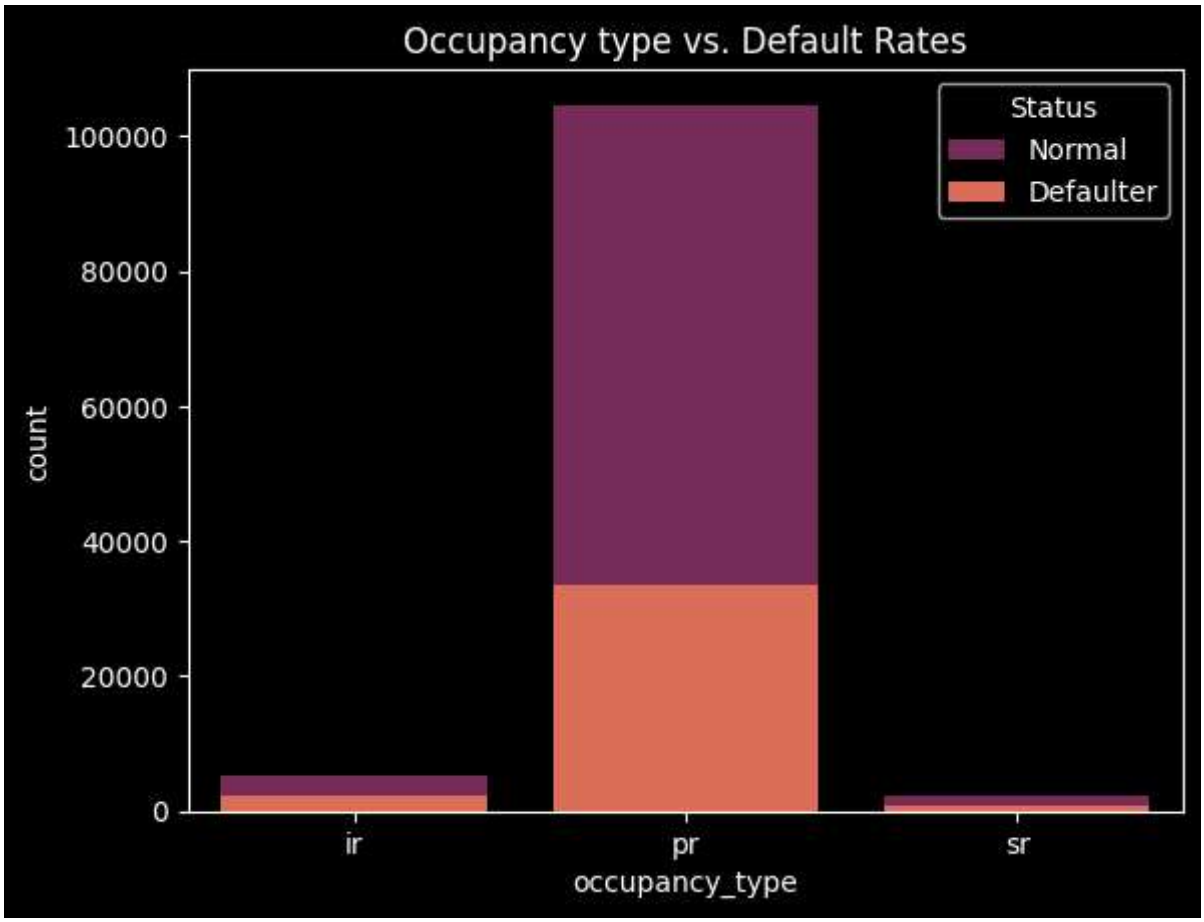
In [224…
```
sns.countplot(x='loan_limit', data=df, hue='Status', palette=['orange','purple'])
plt.title('loan_limit  vs. Default Rates')
```

Out[224]:
```
Text(0.5, 1.0, 'loan_limit  vs. Default Rates')
```



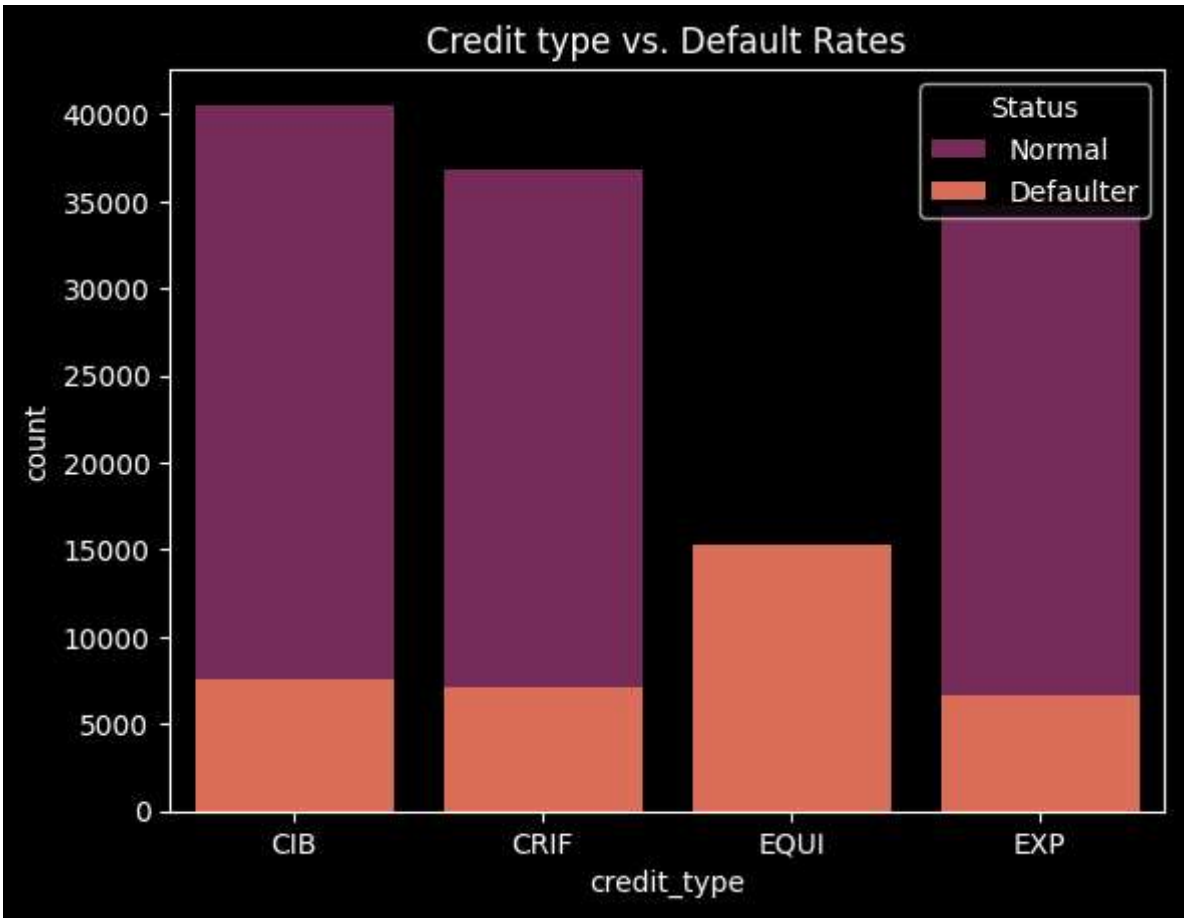
In [219…

```
# Stacked bar chart for two categorical variables
sns.countplot(x='occupancy_type', hue='Status', data=df, palette='rocket', dodge=Fa
plt.title('Occupancy type vs. Default Rates')
plt.show()
```



In [221…

```
# Stacked bar chart for two categorical variables
sns.countplot(x='credit_type', hue='Status', data=df, palette='rocket', dodge=False
plt.title('Credit type vs. Default Rates')
plt.show()
```



Insights

There are a total of 148670 unique IDs present in the datset.

all these entries are from a specific year 2019.

The common datatypes were int64, float64 and obejct which were converted into category, int32, int 16, float32.

The columns loan_limit, loan_purpose, rate_of_interest, upfront_charges, propert_value, income, age, LTV had null values which were filled with mode of respective columns

The numeric columns contained outliers which were processed using inter quartile range and filling with median value of respective column

features like loan to income ration, credit category, income category, LTV category, property range, loan amount category, property to loan ratio were created using existing columns

The data vizualisation of these columns revealed patterns among defaulted payments

- There is no significant difference in the mean of credit score between customers with normal repayment behaviour(699) and defaulters(700)

- The customers with credit type EQUI show more defaults when compared to other credit types (CIB, CRIF AND EXP)

- The customers with loan limit ncf show slightly more defaults when compared to cf loan limit.

- The occupancy type ir have normal repayment customers equivalent to defaulters whereas pr and sr types have comparitively less defaulters.

- There is a significant difference in default rates across younger age groups and older age groups compared to middle aged customers.

- Female customers are less likely to be defaulters compared to male customers and joint customers show less default compared to males.

- Customers with co-applicant with credit type CIB are less likely to be default in their payments when compared to EXP

- Customers who have rate of interest of 3.8 to 4 percent show more default overall.

- Customers who paid very low upfront charges defaulted more on thier repayments.

- There is a significant difference in loan amounts across income and default rate across credit.

- the regions North and south show way more defaults comapred to central and north east regions.

- The customers with type I loan and having significant and drastic difference in their property to loan ratio show normal repayment behaviour.

Recommendations:

Credit Score:

Although there is no significant difference in the mean credit score between normal and defaulting customers, it's crucial to maintain a minimum credit score requirement for loan approval.

Credit Type:

Consider re-evaluating the creditworthiness of customers with EQUI credit type, possibly by implementing stricter approval criteria for this category.

Loan Limit:

Review the approval process for loans with ncf loan limits and assess whether adjustments are needed to reduce the likelihood of defaults.

Occupancy Type:

Re-evaluate the approval criteria for customers with ir occupancy type to reduce default rates, and consider offering more favorable terms for pr and sr types.

Age Groups:

Implement targeted strategies for customer education and risk management for younger and older age groups to reduce default rates.

Gender and Joint Applicants:

Consider tailoring loan products or terms specifically for male customers, and encourage joint applications. This could include providing incentives or lower interest rates for joint applications.

Co-applicant Credit Type:

Encourage co-applicants with CIB credit type, as they seem to be less likely to default compared to EXP.

Interest Rate and Upfront Charges:

Evaluate the impact of interest rates and upfront charges on default rates. Consider optimizing these factors to find a balance that ensures affordability for customers while minimizing defaults.

Loan Amounts and Regions:

Review the approval criteria for loan amounts, ensuring they align with income levels, and implement regional-specific risk management strategies, especially for North and South regions where defaults are higher.

Loan Type and Property to Loan Ratio:

Analyze the factors contributing to the significant and drastic difference in the property to loan ratio for type I loans. Consider adjustments to the approval process or terms for this loan type.

In [ ]: