

Preparation - Importing all necessary modules

```
In [70]: # import all required libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
In [71]: # Display maximum rows and columns with pandas
pd.set_option('display.max_columns', 150)
pd.set_option('display.max_rows', 100)
```

Data Collection - Loading the data sets

```
In [72]: # Load the data set
df_for_loan = pd.read_csv("loan.csv")
```

```
In [73]: df_for_loan.head()
```

```
Out[73]:
```

| | id | member_id | loan_amnt | funded_amnt | funded_amnt_inv | term | int_rate | insta |
|--|----|-----------|-----------|-------------|-----------------|------|----------|-------|
|--|----|-----------|-----------|-------------|-----------------|------|----------|-------|

| | | | | | | | | |
|---|---------|---------|------|------|--------|-----------|--------|--|
| 0 | 1077501 | 1296599 | 5000 | 5000 | 4975.0 | 36 months | 10.65% | |
|---|---------|---------|------|------|--------|-----------|--------|--|

| | | | | | | | | |
|---|---------|---------|------|------|--------|-----------|--------|--|
| 1 | 1077430 | 1314167 | 2500 | 2500 | 2500.0 | 60 months | 15.27% | |
|---|---------|---------|------|------|--------|-----------|--------|--|

| | | | | | | | | |
|---|---------|---------|------|------|--------|-----------|--------|--|
| 2 | 1077175 | 1313524 | 2400 | 2400 | 2400.0 | 36 months | 15.96% | |
|---|---------|---------|------|------|--------|-----------|--------|--|

| | | | | | | | | |
|---|---------|---------|-------|-------|---------|-----------|--------|--|
| 3 | 1076863 | 1277178 | 10000 | 10000 | 10000.0 | 36 months | 13.49% | |
|---|---------|---------|-------|-------|---------|-----------|--------|--|

| | | | | | | | | |
|---|---------|---------|------|------|--------|-----------|--------|--|
| 4 | 1075358 | 1311748 | 3000 | 3000 | 3000.0 | 60 months | 12.69% | |
|---|---------|---------|------|------|--------|-----------|--------|--|

```
In [74]: df_for_loan.columns
```

```
Out[74]: Index(['id', 'member_id', 'loan_amnt', 'funded_amnt', 'funded_amnt_inv',
        'term', 'int_rate', 'installment', 'grade', 'sub_grade',
        ...,
        'num_tl_90g_dpd_24m', 'num_tl_op_past_12m', 'pct_tl_nvr_dlq',
        'percent_bc_gt_75', 'pub_rec_bankruptcies', 'tax_liens',
        'tot_hi_cred_lim', 'total_bal_ex_mort', 'total_bc_limit',
        'total_il_high_credit_limit'],
        dtype='object', length=111)
```

```
In [75]: df_for_loan.describe()
```

```
Out[75]:
```

| | id | member_id | loan_amnt | funded_amnt | funded_amnt_inv | inst |
|--------------|--------------|--------------|--------------|--------------|-----------------|--------|
| count | 3.971700e+04 | 3.971700e+04 | 39717.000000 | 39717.000000 | 39717.000000 | 39717. |
| mean | 6.831319e+05 | 8.504636e+05 | 11219.443815 | 10947.713196 | 10397.448868 | 324. |
| std | 2.106941e+05 | 2.656783e+05 | 7456.670694 | 7187.238670 | 7128.450439 | 208. |
| min | 5.473400e+04 | 7.069900e+04 | 500.000000 | 500.000000 | 0.000000 | 15. |
| 25% | 5.162210e+05 | 6.667800e+05 | 5500.000000 | 5400.000000 | 5000.000000 | 167. |
| 50% | 6.656650e+05 | 8.508120e+05 | 10000.000000 | 9600.000000 | 8975.000000 | 280. |
| 75% | 8.377550e+05 | 1.047339e+06 | 15000.000000 | 15000.000000 | 14400.000000 | 430. |
| max | 1.077501e+06 | 1.314167e+06 | 35000.000000 | 35000.000000 | 35000.000000 | 1305. |

```
In [76]: df_for_loan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39717 entries, 0 to 39716
Columns: 111 entries, id to total_il_high_credit_limit
dtypes: float64(74), int64(13), object(24)
memory usage: 33.6+ MB
```

```
In [77]: df_for_loan.shape
```

```
Out[77]: (39717, 111)
```

Inspecting the loan data, it has been found that we have 39717 rows and 111 columns.

Now we intend to find the missing values on the data set and count along with the percentage so we understand the data in a better way.

```
In [78]: # Checking for the number of missing values on the dataset
missing_values_count = df_for_loan.isnull().sum()

# Print the count of missing values
print(missing_values_count)
```

```

id                0
member_id         0
loan_amnt         0
funded_amnt       0
funded_amnt_inv   0
...
tax_liens         39
tot_hi_cred_lim   39717
total_bal_ex_mort 39717
total_bc_limit     39717
total_il_high_credit_limit 39717
Length: 111, dtype: int64

```

```

In [79]: # Checking the percentage of missing value on the dataset
missing_percentage = (missing_values_count / len(df_for_loan)) * 100

# Displaying the missing data count and percentage together
print(pd.DataFrame({'Total Missing': missing_values_count, 'Percentage': mis

```

| | Total Missing | Percentage |
|---------------------------|---------------|------------|
| verification_status_joint | 39717 | 100.0 |
| annual_inc_joint | 39717 | 100.0 |
| mo_sin_old_rev_tl_op | 39717 | 100.0 |
| mo_sin_old_il_acct | 39717 | 100.0 |
| bc_util | 39717 | 100.0 |
| ... | ... | ... |
| delinq_amnt | 0 | 0.0 |
| policy_code | 0 | 0.0 |
| earliest_cr_line | 0 | 0.0 |
| delinq_2yrs | 0 | 0.0 |
| id | 0 | 0.0 |

[111 rows x 2 columns]

Data Cleaning - It has been found that we have greater number of null values in the Dataset. So, we will remove the null values.

```

In [80]: # Drop the columns with null value
df_for_loan.dropna(axis=1, how="any", subset=None, inplace=True)

df_for_loan.head()

```

```

Out[80]:

```

| | id | member_id | loan_amnt | funded_amnt | funded_amnt_inv | term | int_rate | insta |
|---|---------|-----------|-----------|-------------|-----------------|-----------|----------|-------|
| 0 | 1077501 | 1296599 | 5000 | 5000 | 4975.0 | 36 months | 10.65% | |
| 1 | 1077430 | 1314167 | 2500 | 2500 | 2500.0 | 60 months | 15.27% | |
| 2 | 1077175 | 1313524 | 2400 | 2400 | 2400.0 | 36 months | 15.96% | |
| 3 | 1076863 | 1277178 | 10000 | 10000 | 10000.0 | 36 months | 13.49% | |
| 4 | 1075358 | 1311748 | 3000 | 3000 | 3000.0 | 60 months | 12.69% | |

```

In [81]: # Checking and printing duplicated rows
print(f"Number of duplicate rows: {df_for_loan.duplicated().sum()}")

```

Number of duplicate rows: 0

```
In [82]: df_for_loan.shape
```

```
Out[82]: (39717, 43)
```

Let us find out the loan status for fully accepted loan

```
In [83]: df_for_loan['loan_status'].value_counts()
```

```
Out[83]: loan_status
Fully Paid      32950
Charged Off     5627
Current         1140
Name: count, dtype: int64
```

Since we want to understand the problem areas leading to default. And the Current loans are ongoing and will not add any value to our analysis so we will drop them

```
In [84]: # Exclude 'Current' loans from the dataset
df_for_loan = df_for_loan[df_for_loan['loan_status'] != 'Current']

# Verify the counts of 'loan_status'
print(df_for_loan['loan_status'].value_counts())
```

```
loan_status
Fully Paid      32950
Charged Off     5627
Name: count, dtype: int64
```

```
In [85]: df_for_loan.shape
```

```
Out[85]: (38577, 43)
```

```
In [86]: df_for_loan.head()
```

```
Out[86]:
```

| | id | member_id | loan_amnt | funded_amnt | funded_amnt_inv | term | int_rate | insta |
|---|---------|-----------|-----------|-------------|-----------------|-----------|----------|-------|
| 0 | 1077501 | 1296599 | 5000 | 5000 | 4975.0 | 36 months | 10.65% | |
| 1 | 1077430 | 1314167 | 2500 | 2500 | 2500.0 | 60 months | 15.27% | |
| 2 | 1077175 | 1313524 | 2400 | 2400 | 2400.0 | 36 months | 15.96% | |
| 3 | 1076863 | 1277178 | 10000 | 10000 | 10000.0 | 36 months | 13.49% | |
| 5 | 1075269 | 1311441 | 5000 | 5000 | 5000.0 | 36 months | 7.90% | |

```
In [87]: # Identify columns with constant values

constant_columns = df_for_loan.columns[df_for_loan.nunique() <= 1].tolist()

print("Constant columns to drop:", constant_columns)
```

```
Constant columns to drop: ['pymnt_plan', 'initial_list_status', 'out_prncp', 'out_prncp_inv', 'policy_code', 'application_type', 'acc_now_delinq', 'delinq_amnt']
```

```
In [88]: # Dropping the columns which are having the same value
df_for_loan.drop(columns=constant_columns, inplace=True)
```

```
In [89]: df_for_loan.head()
```

```
Out[89]:
```

| | id | member_id | loan_amnt | funded_amnt | funded_amnt_inv | term | int_rate | insta |
|---|---------|-----------|-----------|-------------|-----------------|-----------|----------|-------|
| 0 | 1077501 | 1296599 | 5000 | 5000 | 4975.0 | 36 months | 10.65% | |
| 1 | 1077430 | 1314167 | 2500 | 2500 | 2500.0 | 60 months | 15.27% | |
| 2 | 1077175 | 1313524 | 2400 | 2400 | 2400.0 | 36 months | 15.96% | |
| 3 | 1076863 | 1277178 | 10000 | 10000 | 10000.0 | 36 months | 13.49% | |
| 5 | 1075269 | 1311441 | 5000 | 5000 | 5000.0 | 36 months | 7.90% | |

```
In [90]: df_for_loan.shape
```

```
Out[90]: (38577, 35)
```

```
In [91]: df_for_loan = df_for_loan.astype({
    'dti': 'float',
    'funded_amnt': 'float',
    'funded_amnt_inv': 'float',
    'loan_amnt': 'float'
})
```

```
In [92]: # Convert term column to int type
df_for_loan['term'] = df_for_loan['term'].apply(lambda x: int(x.replace('months', '')))
```

```
In [93]: # Convert int_rate column to float type
df_for_loan['int_rate'] = df_for_loan['int_rate'].apply(lambda x: float(x.replace('%', '')))
```

```
In [ ]:
```

Univariate analysis

We will now perform analysis on individual variables to understand their distributions, detect outliers, and gain insights.

```
In [ ]:
```

Lets explore the distribution of continuous variables such as loan_amnt, int_rate, annual_inc and dti

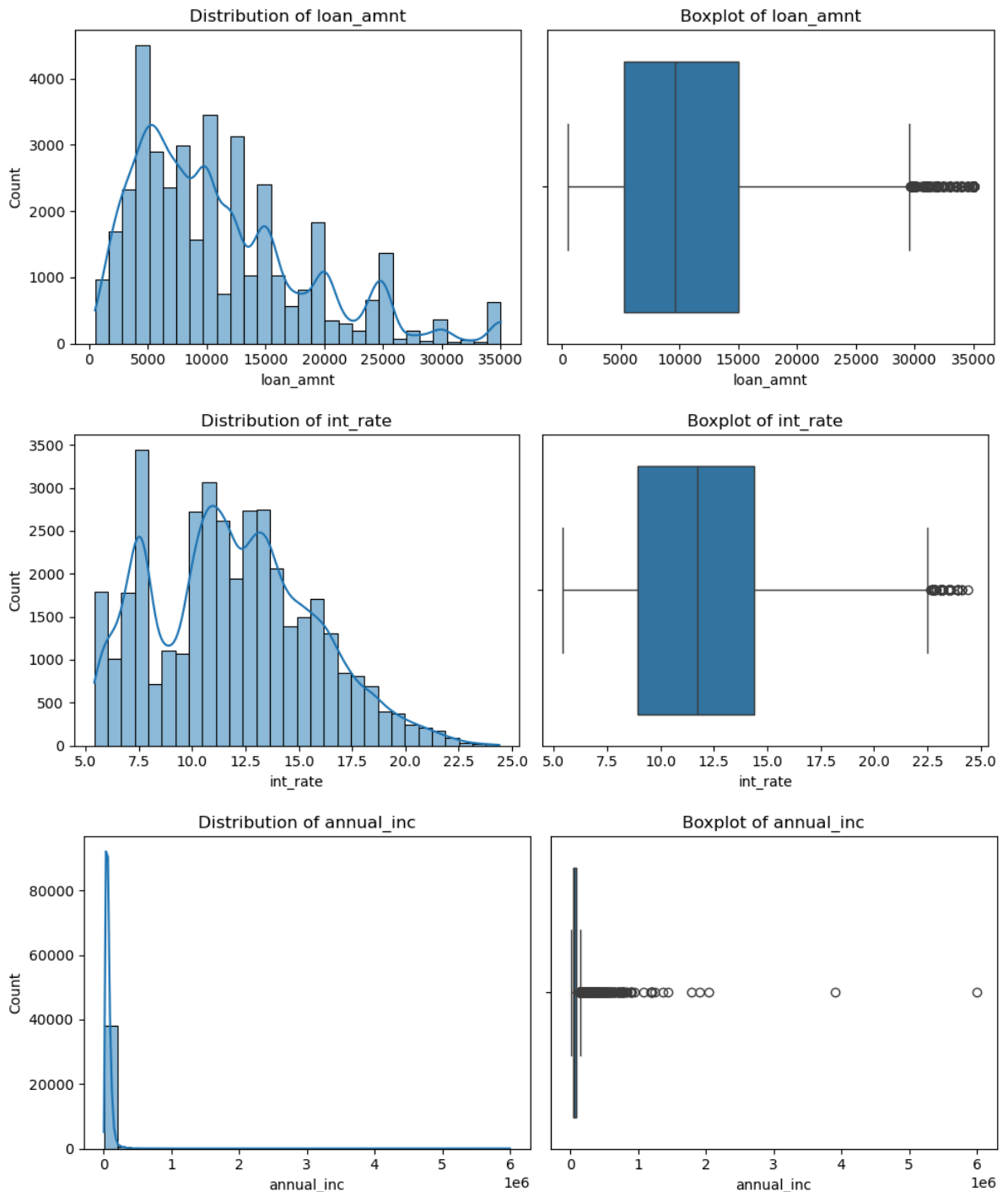
```
In [94]: # List of numerical variables to plot
numerical_vars = ['loan_amnt', 'int_rate', 'annual_inc', 'dti']

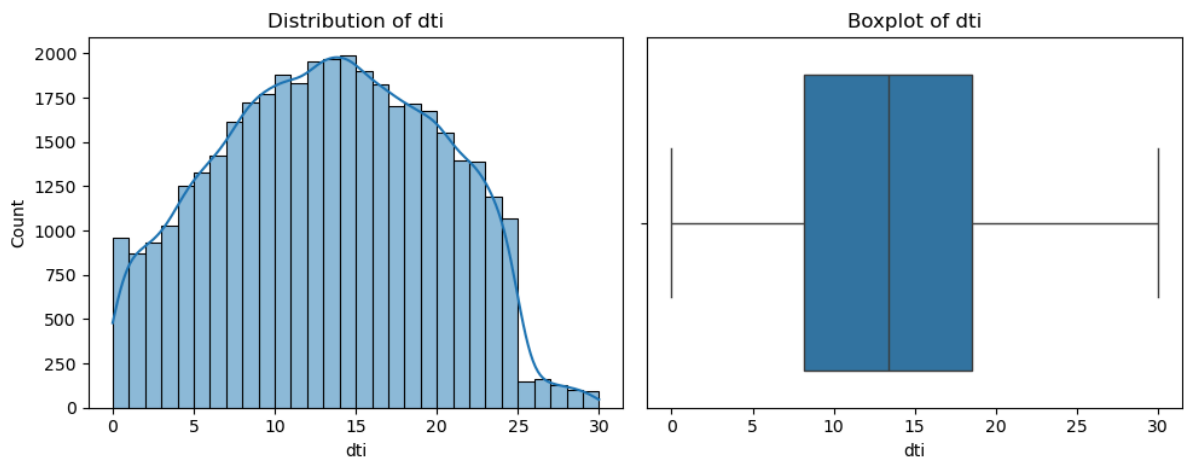
# Plot histograms and boxplots
for var in numerical_vars:
    plt.figure(figsize=(10, 4))
```

```
plt.subplot(1, 2, 1)
sns.histplot(df_for_loan[var], bins=30, kde=True)
plt.title(f'Distribution of {var}')

plt.subplot(1, 2, 2)
sns.boxplot(x=df_for_loan[var])
plt.title(f'Boxplot of {var}')

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





Following are the analysis that can be observed -

1. Loan Amount - Majority of the loans falls between 5000 to 15000 dollars range but the higher loan amount could correlate with higher risks of default hence outliers observed.
2. Interest Rate - Interest Rate are distributed across but it has been found most of it found in 5.0 to 15.0 interest rate with the peak on 7.5%
3. Annual Income - It has been found that there are heavy outliers with very high incomes so we can consider that majority of borrowers have incomes below \$100,000.
4. DTI - The Debit to Income is most distributed with most values between 10% to 15 %. Higher DTI could mean that there is more stress to the borrower

Now Lets us explore the categorical variables such as grade, sub_grade, home_ownership and purpose

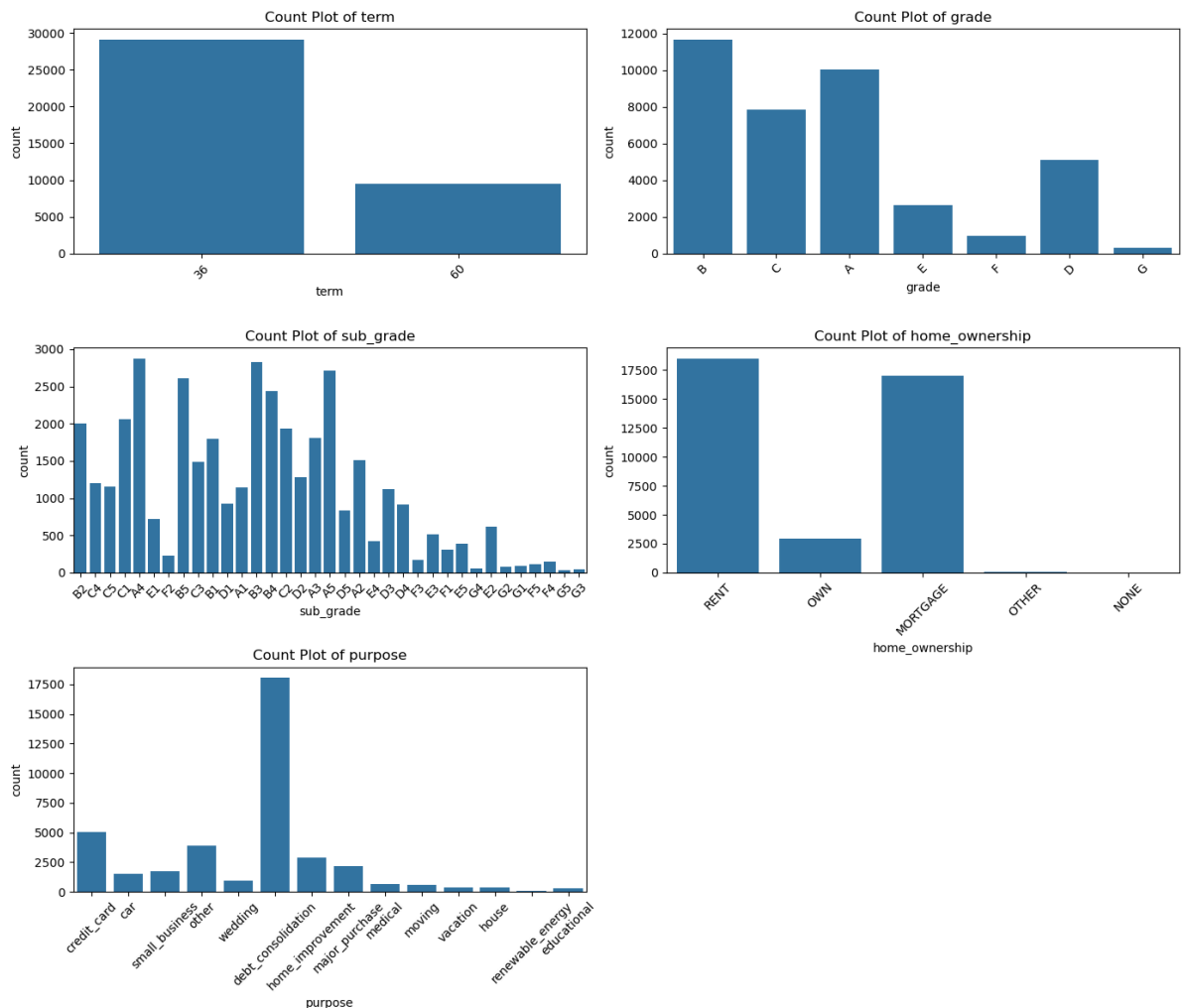
```
In [112... # List of categorical variables to plot
categorical_vars = ['term', 'grade', 'sub_grade', 'home_ownership', 'purpose']

# Plot count plots for each categorical variable (2 plots per row)
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(14, 12))
axes = axes.flatten()

for i, var in enumerate(categorical_vars):
    sns.countplot(x=var, data=df_for_loan, ax=axes[i])
    axes[i].set_title(f'Count Plot of {var}')
    axes[i].set_xticklabels(axes[i].get_xticklabels(), rotation=45)

# Remove extra subplot if necessary
if len(categorical_vars) % 2 != 0:
    fig.delaxes(axes[-1])

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



Following are the analysis that can be seen here -

1. Loan Term - Most loans taken are for shorter duration (36 months).
2. Grade - Majority of Loan are from B, C and A whereas lower Loan Grade which are D, E, F and G may have high risks associated with them.
3. Home Ownership - Most borrowers are either RENT or have mortgage. Renters may pose higher risk to default in comparison to the ones who owns it or have a mortgage on them.
4. Purpose - Debt Consolidation is the most common reasons for taking loan

Bivariate Analysis

Now let us understand the relationships of the these variables with target variables be it loan_status or loan_default.

We will analyze if the variables like purpose, interest rate, loan_amout, home ownership could result to default.

We will also analyze how these variables behave for borrowers who have been marked as Charged Off.

```
In [115... # Create a new binary target variable for loan default (1 = Charged Off, 0 = Not Charged Off)
df_for_loan['loan_default'] = df_for_loan['loan_status'].apply(lambda x: 1 if x == 'Charged Off' else 0)
```

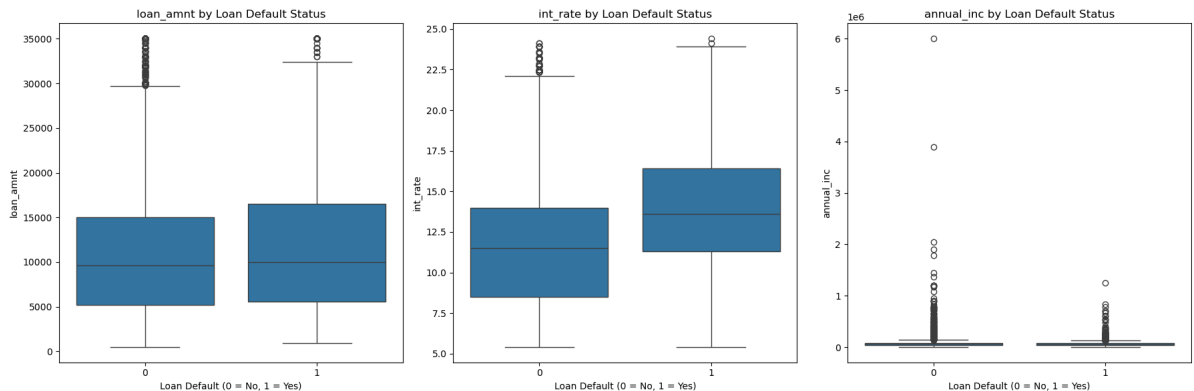


```
# List of numerical variables to plot
numerical_vars = ['loan_amnt', 'int_rate', 'annual_inc']

# Plot boxplots for numerical variables against loan default
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(18, 6)) # Adjust ncols
axes = axes.flatten()

for i, var in enumerate(numerical_vars):
    sns.boxplot(x='loan_default', y=var, data=df_for_loan, ax=axes[i])
    axes[i].set_title(f'{var} by Loan Default Status')
    axes[i].set_xlabel('Loan Default (0 = No, 1 = Yes)')
    axes[i].set_ylabel(var)

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



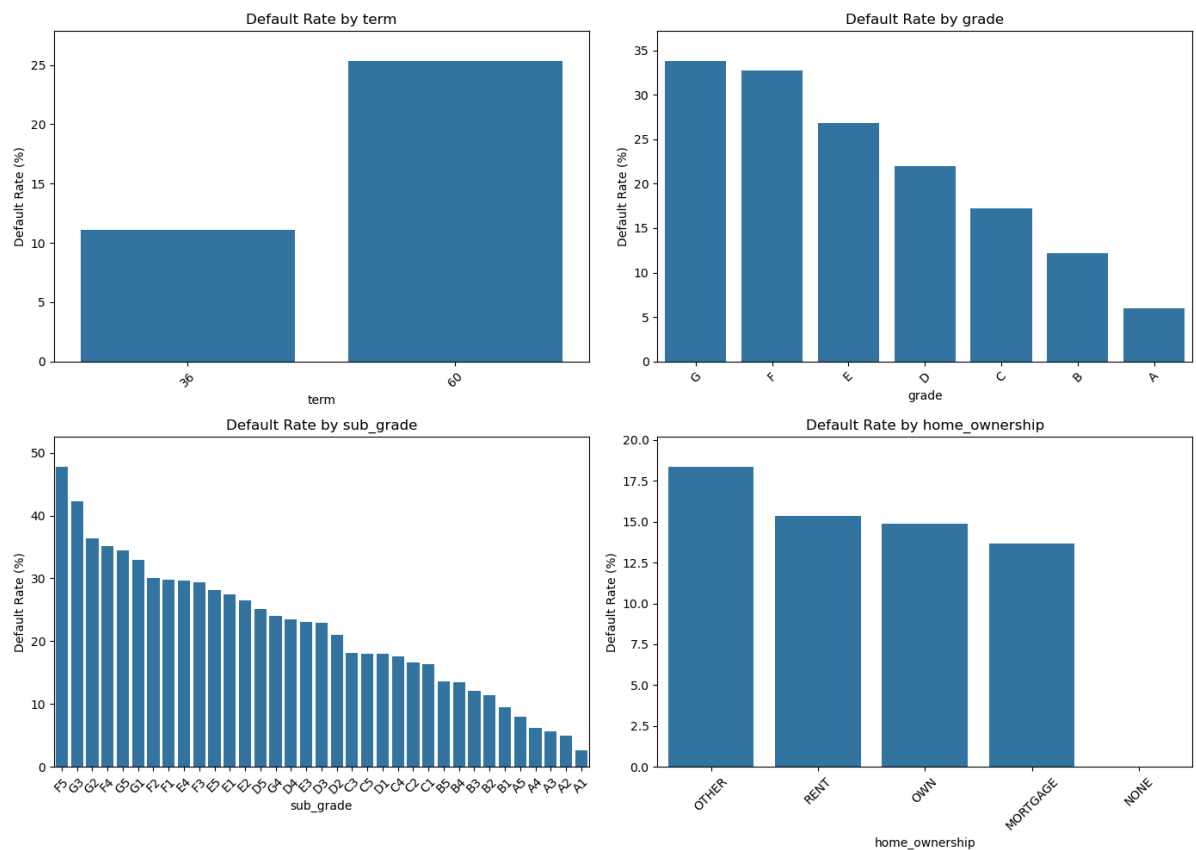
In [117...

```
# Calculate default rates for categorical variables
categorical_vars = ['term', 'grade', 'sub_grade', 'home_ownership']

# Set up a figure with multiple subplots (side by side)
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(14, 10)) # Adjust rows
axes = axes.flatten()

# Plot default rates for each categorical variable
for i, var in enumerate(categorical_vars):
    default_rate = df_for_loan.groupby(var)['loan_default'].mean().sort_values
    sns.barplot(x=default_rate.index, y=default_rate.values, ax=axes[i])
    axes[i].set_title(f'Default Rate by {var}')
    axes[i].set_ylabel('Default Rate (%)')
    axes[i].set_ylim(0, default_rate.max() * 1.1) # Set y-axis limit slightly
    axes[i].tick_params(axis='x', rotation=45)

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



Following are the analysis that can be seen -

1. Loan Term: Longer mortgage terms (60 months) are related to significantly better default quotes. This shows that longer-term loans pose extra chance.
2. Grade and Sub-Grade: Loan grade is a totally robust predictor of default danger. Lower grades (D, E, F, G) are much more likely to default, and this holds real at the sub-grade stage as well.
3. Interest Rate: Borrowers with higher interest charges are much more likely to default, reinforcing the concept that higher prices can be given to riskier borrowers.
4. Home Ownership: Renters and those classified as "OTHER" are much more likely to default than homeowners or those with mortgages.
5. Loan Purpose: Small business loans have the very best default fees, suggesting that this cause is in particular risky.

Multivariate

It will enables us to find relationships among three or more variables at a time and get a deeper understanding of their interaction.

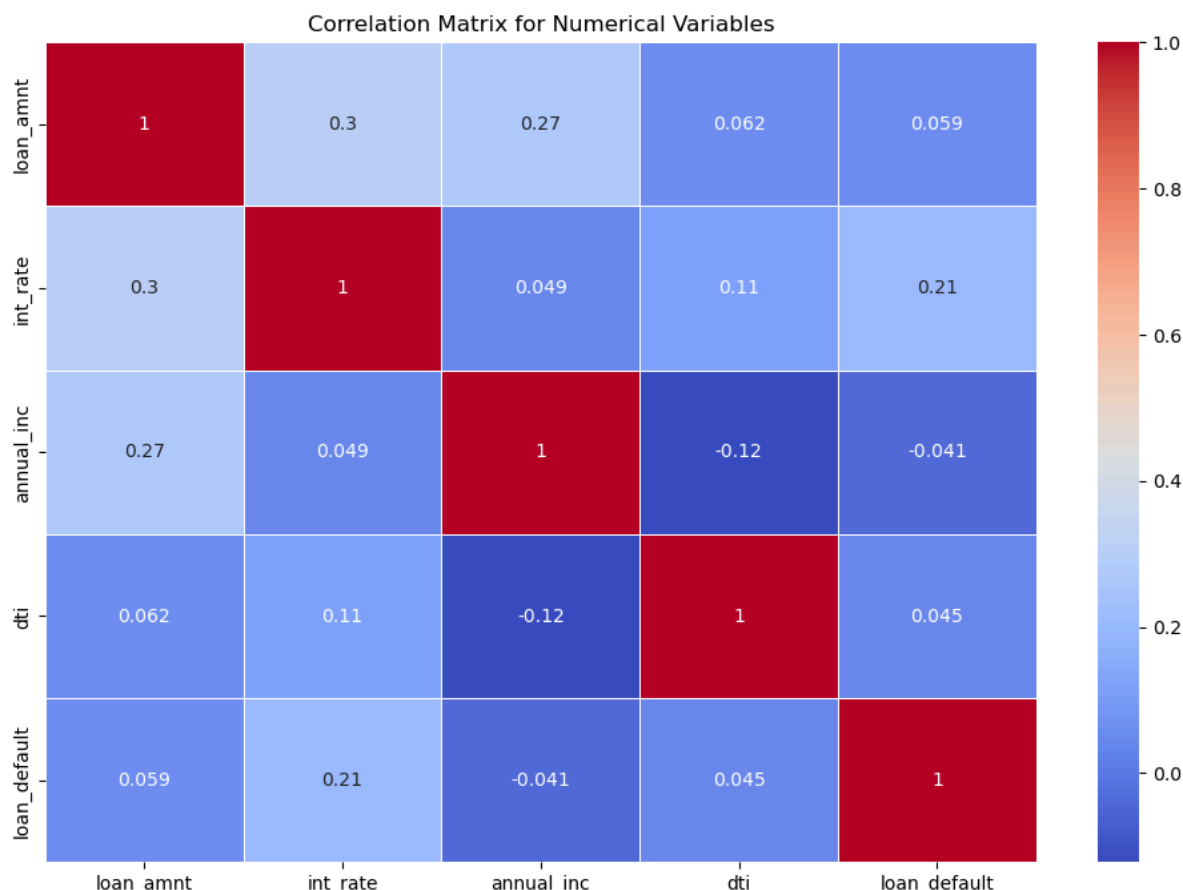
This will supply us greater sturdy insights into which mixtures of variables are most predictive of mortgage default.

A correlation matrix will assist us understand the relationships among the numerical variables and take a look at if any are specifically correlated with one another or with the intention

variable (loan_default). This can monitor multicollinearity or sturdy predictive relationships.

```
In [98]: # Correlation matrix for numerical variables
plt.figure(figsize=(12, 8))
corr_matrix = df_for_loan[['loan_amnt', 'int_rate', 'annual_inc', 'dti', 'loan_default']]

# Plotting a heatmap for the correlation matrix
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix for Numerical Variables')
plt.show()
```



Post Analyzing the Correlation Matrix for Numerical Variables, we understand that -

Loan quantity has weak positive correlation (0.059) with loan default. So we can understand that the loan amount does not influence loan default.

There is a moderate positive correlation (0.21) between the interest rate and loan default. So, we can understand that - with increase in interest rates, there is a tendency to loan default.

The correlation between DTI and loan default is weak (0.045). hence it does not give any desired relationship on this dataset

The annual income shows a slight negative correlation (-0.041) with loan default which gives us the impression that the higher-income borrowers are less likely to be default.

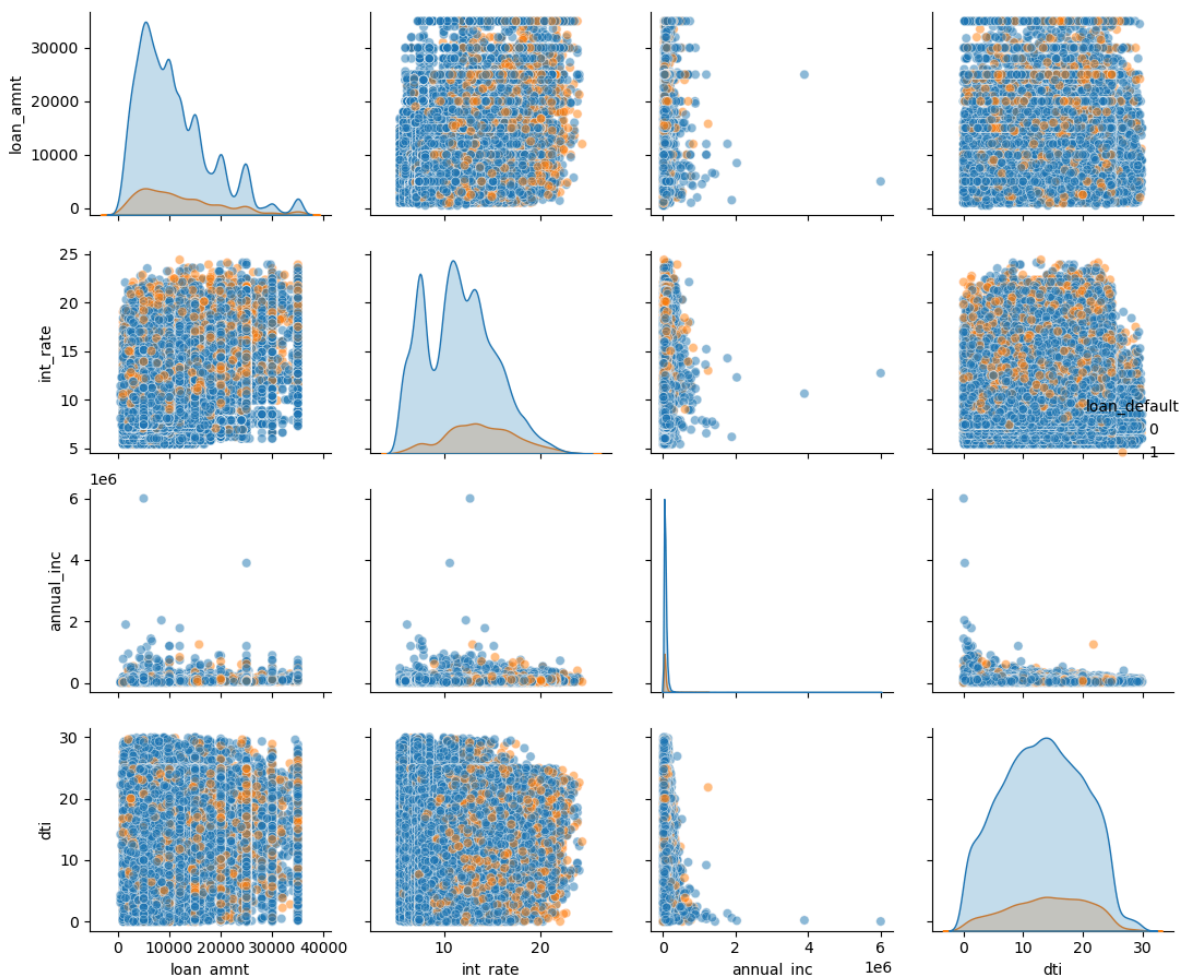
Pairplot for Numerical Variables

A pairplot can visually show how specific numerical variables interact with every other, at the equal time as highlighting the distribution of loan_default (the use of shade coding).

```
In [103... # Pairplot for numerical variables colored by loan default
pair_plot = sns.pairplot(df_for_loan[['loan_amnt', 'int_rate', 'annual_inc',
pair_plot.fig.suptitle('Pairplot for Numerical Variables and Loan Default',

# Adjusting the layout to ensure title doesn't overlap
pair_plot.fig.tight_layout()
pair_plot.fig.subplots_adjust(top=0.9)
plt.show()
```

Pairplot for Numerical Variables and Loan Default



After having analyzed the above pair plot, we have come across the below given findings -

Loan Amount: Most loans disbursed are in lower amounts (below 20,000), with few visible patterns regarding default for higher loan amounts.

Interest Rate: There is a clear indication that higher interest rates causes the maximum defaults. Borrowers with loans in higher interest rates tend to have a higher chances to default.

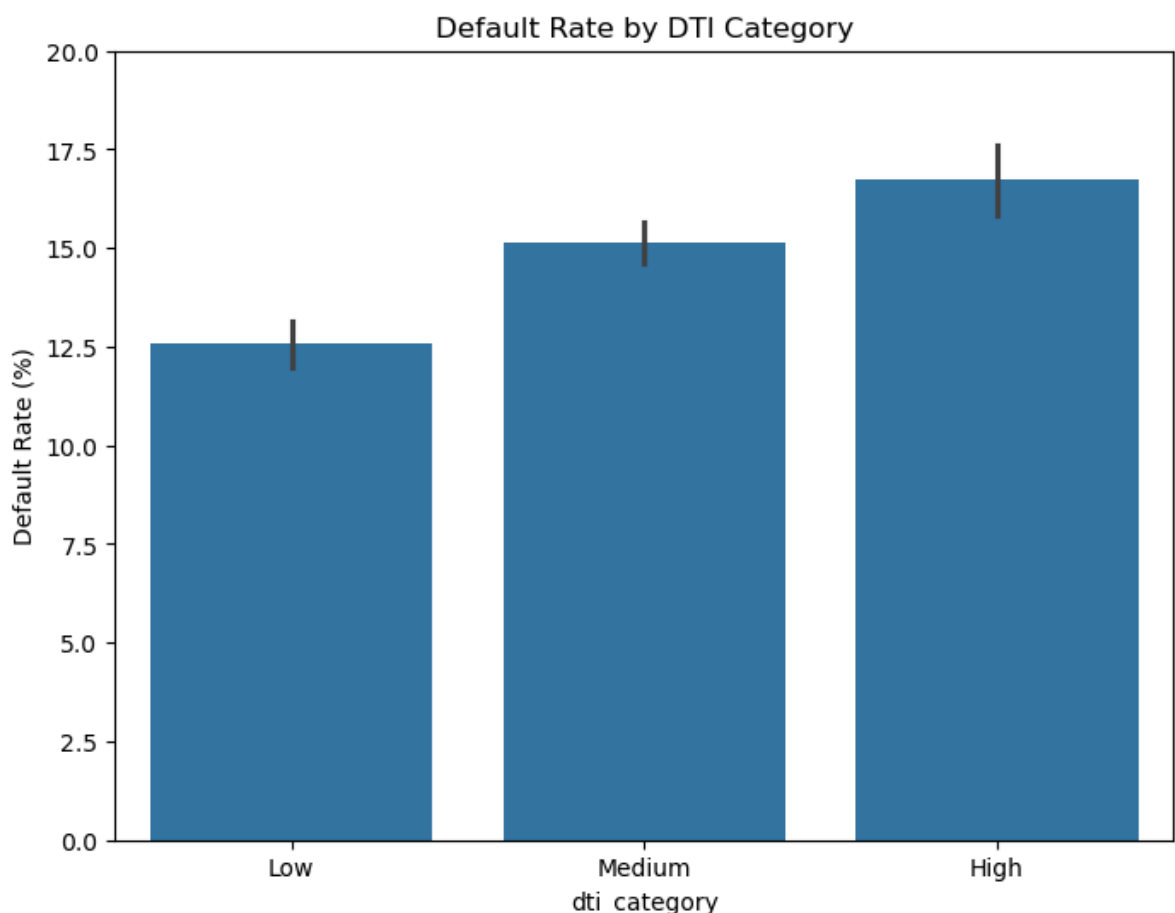
Annual Income: Most of the data is concentrated among borrowers with lower annual income, but a few outliers exist in the higher income brackets, which do not show much association with default.

DTI: With increase in DTI, there are higher chances of default, especially in the medium-to-high DTI range.

Create and Analyze DTI Categories

```
In [107... # Create categories for DTI
df_for_loan['dti_category'] = pd.cut(df_for_loan['dti'], bins=[0, 10, 20, 30], labels=['Low', 'Medium', 'High'])

# Plotting the default rates by DTI categories with percentages
plt.figure(figsize=(8, 6))
sns.barplot(x='dti_category', y='loan_default', data=df_for_loan, estimator=None)
plt.title('Default Rate by DTI Category')
plt.ylabel('Default Rate (%)') # Change the y-axis label to percentage
plt.ylim(0, 20) # Set y-axis limits to match percentage scale
plt.show()
```



The above plot do tell us the following -

The load default rate increases from Low DTI to High DTI Category.

Borrowers with a higher debt-to-income ratio are more likely to default, which aligns with typical risk assessment in finance.

It has been observed that High DTI has the highest default rate (around 17.5%), followed by Medium DTI (around 15%), and Low DTI (around 12%)

Recommendation to the business

Higher Interest Rates Signal Risk: There's a moderate link between high interest rates and loan defaults. The company should be careful when issuing loans with higher interest rates and might need to tighten its lending criteria for applicants considered to be at higher risk.

Keep an Eye on the Debt-to-Income (DTI) Ratio: Borrowers with a high DTI ratio are more likely to default on their loans. To mitigate this risk, the company should consider stricter standards for applicants with a high DTI, potentially denying loans or adjusting the terms for these borrowers.

Improving Risk Models:

Use DTI Categories: Including DTI categories in future risk assessments could help the company evaluate loan applicants more accurately. By separating applicants into groups based on their DTI—such as low, medium, or high-risk—the company can adjust its lending strategies to better fit each group's risk level.

Examine Income and Loan Outliers: Investigating unusual patterns in annual income and loan amounts could provide insights into the behaviors of borrowers who default versus those who don't default.

Interest Rate Adjustments: The company should consider offering variable interest rates based on the risk profile of the borrower. For higher-risk applicants, either deny the loan or offer it at significantly higher interest rates with stricter repayment conditions.

Income and Loan Defaults: Although the link between annual income and default rates is weak, borrowers with higher incomes tend to default less. A deeper analysis of income brackets may yield useful information for refining lending strategies.

In []: