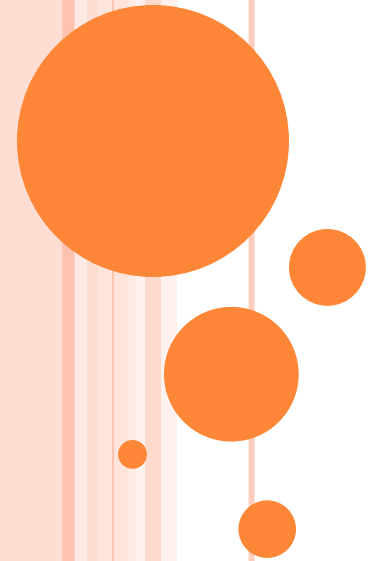
ML C59-UPGRAD IITB EPGP in ML & AI

LENDING CLUB CASE STUDY

GROUP MEMBERS:

1. AMIT RANJAN
2. GAURAV GARG



upGrad

Case Study Objectives:

•Indentify patterns and indicators that may predict loan default likelihood.

•Explore the relationship between loan status and applicant profiles, such as employment length, annual income and loan amount.

•Analyze the impact of different features on the likelihood of loan default, including verification status, purpose of the loan and grade.

•Develop insights that can be used to inform loan approval decisions, loan amounts , and interest rates.

Data Cleaning Steps:

1. **Handling Missing Values:**
   - Identify and analyze missing values in the dataset.
   - If certain columns have a significant number of missing values and are not essential for analysis, consider dropping those columns.
   - For columns with relatively few missing values, We can impute missing values using methods like mean, median, or machine learning-based imputation.

2. **Duplicate Data:**
   - Check for duplicate rows in the dataset.
   - Remove any duplicated rows to ensure each observation is unique.

3. **Data Type Conversion:**
   - Check if the data types of variables are appropriate.
   - Ensure that categorical variables are encoded properly.
   - Convert date columns to date time data type if applicable.

Data Cleaning Steps Continue…

4. Outliers Detection and Handling:

- Identify outliers in numerical variables using statistical methods.
- Decide whether to remove outliers or transform them based on the context of the analysis.

5. Handling Categorical Variables:

- Convert categorical variables to a format suitable for analysis.
- Ensure that the encoding method aligns with the analysis goals.

Data Cleaning Steps Continue...

6. Handling Irrelevant or Redundant Columns:

- Identify columns that do not contribute to the analysis objectives.
- Remove irrelevant or redundant columns to streamline the dataset.

7. Data Exploration for Anomalies:

- Conduct exploratory data analysis (EDA) to identify any unexpected patterns or anomalies that may require further investigation.

8. Documentation:

- Keep detailed records of the data cleaning steps performed for transparency and reproducibility.

Analysis

•The primary objective of this project is to comprehensively analyze and discern the influential factors within consumer attributes that contribute to the likelihood of loan default.

•Through detailed exploratory data analysis (EDA) , the aim is to uncover patterns, correlations, and insights that shed light on the key characteristics or behaviors exhibited by loan applicants that are associated with a higher tendency to default. This analysis will empower the company to make informed decisions , refine risk assessment strategies , and proactively mitigate the impact of defaults by identifying and understanding the driving forces behind them.

Exploratory Data Analysis(EDA):

Load the Dataset: loan.csv

- Imported necessary libraries and loaded dataset into a Pandas DataFrame.
- Also displayed the first five rows of a DataFrame.

In [1]:
```
import pandas as pd

# Load the loan.csv file
loan_df = pd.read_csv('loan.csv')

# Display the first few rows of the dataframe
loan_df.head()
```

```
C:\Users\amitr\AppData\Local\Temp\ipykernel_13484\1748146803.py:4: DtypeWarning: Columns (47) have mixed types. Specify dty
pe option on import or set low_memory=False.
  loan_df = pd.read_csv('loan.csv')
```

Out[1]:

	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_
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87	B	B2	...	NaN	
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83	C	C4	...	NaN	
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	84.33	C	C5	...	NaN	
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	339.31	C	C1	...	NaN	
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	67.79	B	B5	...	NaN	

5 rows × 111 columns

Exploratory Data Analysis(EDA):Cont.

Data Overview:

- Displayed the basic information about the dataset , such as the number of rows, columns, data types and summary statistics.
- Loan_df.info() is useful method to quickly get an overview of the structure and characteristics of a DataFrame
- The dataset contains a total of 111 columns and 39,717 entries. The data types include float64, int64, and object. This information will be useful for further data cleaning and analysis

```
In [7]: # Display basic information about the dataset
        print(loan_df.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
        None
```

Exploratory Data Analysis(EDA):Cont.

Data Overview:

- Loan_df.describe().T is useful method in pandas to generate descriptive statistics of a DataFrame or Series.
- It print a summary of various statistical measures for each numerical column in the DataFrame.

```
In [2]: loan_df.describe().T
```

Out[2]:

	count	mean	std	min	25%	50%	75%	max
id	39717.0	683131.913060	210694.132915	54734.0	516221.0	665665.0	837755.0	1077501.0
member_id	39717.0	850463.559408	265678.307421	70699.0	666780.0	850812.0	1047339.0	1314167.0
loan_amnt	39717.0	11219.443815	7456.670694	500.0	5500.0	10000.0	15000.0	35000.0
funded_amnt	39717.0	10947.713196	7187.238670	500.0	5400.0	9600.0	15000.0	35000.0
funded_amnt_inv	39717.0	10397.448868	7128.450439	0.0	5000.0	8975.0	14400.0	35000.0
...	...	...	...	...	...	...	...	...
tax_liens	39678.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0
tot_hi_cred_lim	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
total_bal_ex_mort	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
total_bc_limit	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
total_il_high_credit_limit	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN

87 rows × 8 columns

Exploratory Data Analysis(EDA):Cont.

Data Overview:

- The Code loan_status_counts=loan_df('loan_status'].value_count() is used to count the occurences of unique values in the 'loan_status' column of the DataFram 'loan_df'.
- Result is a new pandas series 'loan_status_counts' where the index represents unique values in the 'loan_status' column and the values represent the count of each unique value.

```
In [10]: loan_status_counts = loan_df['loan_status'].value_counts()
         loan_status_counts

Out[10]: Fully Paid     32950
         Charged Off     5627
         Current         1140
         Name: loan_status, dtype: int64
```

upGrad

Exploratory Data Analysis(EDA):Cont.

Check For Missing Values In Dataset:

- The dataset contains missing values in several columns, including “emp_title”, “emp_lenght”,”desc”,”title” and “mths_since_last_delinnq”
- Here's a summary of the missing values in some of the columns:

emp_title': 2,459 missing values
'emp_length': 1,075 missing values
'desc': 12,940 missing values
'title': 11 missing values
'mths_since_last_delinq': 25,682 missing values

```
In [15]: # Check for missing values in the dataset
         missing_values = loan_df.isnull().sum()
         missing_values

Out[15]: 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
```

Exploratory Data Analysis(EDA):Cont.

- **List down the column name which has Missing Values In Dataset:**

In [24]:

```
# columns_with_missing_values
columns_with_missing_values = loan_df.columns[loan_df.isnull().any()]

print("Columns with missing values:")
print(columns_with_missing_values)
```

```
Columns with missing values:
Index(['emp_title', 'emp_length', 'desc', 'title', 'mths_since_last_delinq',
       'mths_since_last_record', 'revol_util', 'last_pymnt_d', 'next_pymnt_d',
       'last_credit_pull_d', 'collections_12_mths_ex_med',
       'mths_since_last_major_derog', 'annual_inc_joint', 'dti_joint',
       'verification_status_joint', 'tot_coll_amt', 'tot_cur_bal',
       'open_acc_6m', 'open_il_6m', 'open_il_12m', 'open_il_24m',
       'mths_since_rcnt_il', 'total_bal_il', 'il_util', 'open_rv_12m',
       'open_rv_24m', 'max_bal_bc', 'all_util', 'total_rev_hi_lim', 'inq_fi',
       'total_cu_tl', 'inq_last_12m', 'acc_open_past_24mths', 'avg_cur_bal',
       'bc_open_to_buy', 'bc_util', 'chargeoff_within_12_mths',
       'mo_sin_old_il_acct', 'mo_sin_old_rev_tl_op', 'mo_sin_rcnt_rev_tl_op',
       'mo_sin_rcnt_tl', 'mort_acc', 'mths_since_recent_bc',
       'mths_since_recent_bc_dlq', 'mths_since_recent_inq',
       'mths_since_recent_revol_delinq', 'num_accts_ever_120_pd',
       'num_actv_bc_tl', 'num_actv_rev_tl', 'num_bc_sats', 'num_bc_tl',
       'num_il_tl', 'num_op_rev_tl', 'num_rev_accts', 'num_rev_tl_bal_gt_0',
       'num_sats', 'num_tl_120dpd_2m', 'num_tl_30dpd', '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')
```

upGrad

Exploratory Data Analysis(EDA):Cont.

Missing Values Percentage In Dataset:

- The percentage of missing values in the dataset ranges from 0% to 51.34%. The columns with missing values include "emp_title," "emp_length," "desc," and "title."

```
In [21]:
# Assuming loan_df is your DataFrame
total_missing = loan_df.isnull().sum().sum()
total_cells = loan_df.size
percentage_missing = (total_missing / total_cells) * 100

print(f'The percentage of missing values in the dataset is: {percentage_missing:.2f}%')
```

```
The percentage of missing values in the dataset is: 51.34%
```

upGrad

Exploratory Data Analysis(EDA):Cont.

Drop Columns with a significant number of missing values:

In [32]:
```
# Drop columns with a significant number of missing values
loan_df_linked_cleaned = loan_df.dropna(axis=1, thresh=0.5*len(loan_df))

# Display the first few rows of the cleaned DataFrame
loan_df_linked_cleaned.head()
```

Out[32]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	...	last_pymnt_amnt	last_credit_pull_
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87	B	B2	...	171.62	May-1
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83	C	C4	...	119.66	Sep-1
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	84.33	C	C5	...	649.91	May-1
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	339.31	C	C1	...	357.48	Apr-1
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	67.79	B	B5	...	67.79	May-1

5 rows × 54 columns

upGrad

Exploratory Data Analysis(EDA):Cont.

- **Impute missing values with mode:**

In [34]:
```
# Impute missing values in the remaining columns with mode
loan_df_linked_cleaned_imputed_mode = loan_df_linked_cleaned.fillna(loan_df_linked_cleaned.mode().iloc[0])

# Display the first few rows of the imputed DataFrame
loan_df_linked_cleaned_imputed_mode.head()
```

Out[34]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	...	last_pymnt_amnt	last_credit_pull_
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87	B	B2	...	171.62	May-1
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83	C	C4	...	119.66	Sep-1
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	84.33	C	C5	...	649.91	May-1
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	339.31	C	C1	...	357.48	Apr-1
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	67.79	B	B5	...	67.79	May-1

upGrad

Exploratory Data Analysis(EDA):Cont.

Check for Duplicate in Dataset:

- There are no duplicate records in the dataset, which is a positive sign for data quality.

```
In [25]: # Check for duplicate records
         duplicate_records = loan_df.duplicated().sum()
         duplicate_records

Out[25]: 0
```

Exploratory Data Analysis(EDA):Cont.

Clean the date columns:

- To ensure that date column contain valid date values in the correct format.

In [36]:

```
# Clean the date columns
import datetime

# Define a function to clean the date values
# This function will convert the date values to the correct format
# and handle any missing or invalid date values

def clean_date(date_str):
    try:
        # Convert the date string to a datetime object
        date_obj = datetime.datetime.strptime(date_str, '%b-%y')
        return date_obj
    except (ValueError, TypeError):
        # Handle missing or invalid date values
        return None

# Apply the clean_date function to the date columns
loan_df_linked_cleaned_imputed_mode['issue_d'] = loan_df_linked_cleaned_imputed_mode['issue_d'].apply(clean_date)
loan_df_linked_cleaned_imputed_mode['earliest_cr_line'] = loan_df_linked_cleaned_imputed_mode['earliest_cr_line'].apply(clear
loan_df_linked_cleaned_imputed_mode['last_pymnt_d'] = loan_df_linked_cleaned_imputed_mode['last_pymnt_d'].apply(clean_date)
loan_df_linked_cleaned_imputed_mode['last_credit_pull_d'] = loan_df_linked_cleaned_imputed_mode['last_credit_pull_d'].apply(c

# Display the first few rows of the cleaned date columns
loan_df_linked_cleaned_imputed_mode[['issue_d', 'earliest_cr_line', 'last_pymnt_d', 'last_credit_pull_d']].head()
```

Out[36]:

	issue_d	earliest_cr_line	last_pymnt_d	last_credit_pull_d
0	2011-12-01	1985-01-01	2015-01-01	2016-05-01
1	2011-12-01	1999-04-01	2013-04-01	2013-09-01
2	2011-12-01	2001-11-01	2014-06-01	2016-05-01
3	2011-12-01	1996-02-01	2015-01-01	2016-04-01

upGrad

Exploratory Data Analysis(EDA):Cont.

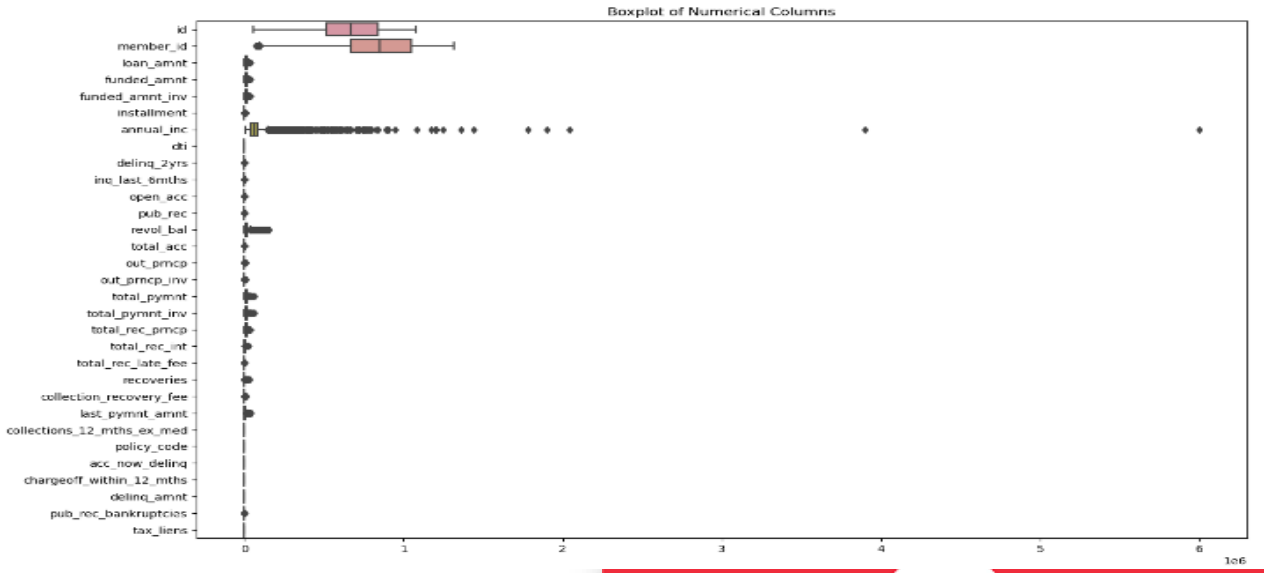
Outliers Detection and Handling :

- The boxplot provides a visual representation of the distribution of values in each numerical column and highlights the presence of outliers.

In [39]:

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

# Create boxplots for the numerical columns to visualize the outliers
plt.figure(figsize=(15, 10))
sns.boxplot(data=loan_df_linked_cleaned_imputed_mode.select_dtypes(include=[np.number]), orient="h")
plt.title('Boxplot of Numerical Columns')
plt.show()
```



upGrad

Exploratory Data Analysis(EDA):Cont.

Examine each categorical variable and identify any inconsistencies :

```
In [42]: #examine each categorical variable and identify any inconsistencies.
         loan_df_linked_cleaned_imputed_mode.select_dtypes(include='object').columns

Out[42]: Index(['term', 'int_rate', 'grade', 'sub_grade', 'emp_title', 'emp_length',
                'home_ownership', 'verification_status', 'loan_status', 'pymnt_plan',
                'url', 'desc', 'purpose', 'title', 'zip_code', 'addr_state',
                'revol_util', 'initial_list_status', 'application_type'],
               dtype='object')
```

upGrad

Exploratory Data Analysis(EDA):Cont.

Convert categorical variables to a format suitable for analysis :

In [43]:
```
#Convert categorical variables to a format suitable for analysis
loan_df_linked_cleaned_imputed_mode['sub_grade'] = loan_df_linked_cleaned_imputed_mode['sub_grade'].astype('category').cat.co
loan_df_linked_cleaned_imputed_mode['emp_title'] = loan_df_linked_cleaned_imputed_mode['emp_title'].astype('category').cat.co
loan_df_linked_cleaned_imputed_mode['home_ownership'] = loan_df_linked_cleaned_imputed_mode['home_ownership'].astype('categor
loan_df_linked_cleaned_imputed_mode['verification_status'] = loan_df_linked_cleaned_imputed_mode['verification_status'].astyp
loan_df_linked_cleaned_imputed_mode['loan_status'] = loan_df_linked_cleaned_imputed_mode['loan_status'].astype('category').ca
loan_df_linked_cleaned_imputed_mode.head()
```

Out[43]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	...	last_pymnt_amnt	last_credit_pull_
0	1077501	1296599	5000	5000	4975.0	36 months	10.65%	162.87	B	6	...	171.62	2016-05-0
1	1077430	1314167	2500	2500	2500.0	60 months	15.27%	59.83	C	13	...	119.66	2013-09-0
2	1077175	1313524	2400	2400	2400.0	36 months	15.96%	84.33	C	14	...	649.91	2016-05-0
3	1076863	1277178	10000	10000	10000.0	36 months	13.49%	339.31	C	10	...	357.48	2016-04-0
4	1075358	1311748	3000	3000	3000.0	60 months	12.69%	67.79	B	9	...	67.79	2016-05-0

5 rows × 54 columns

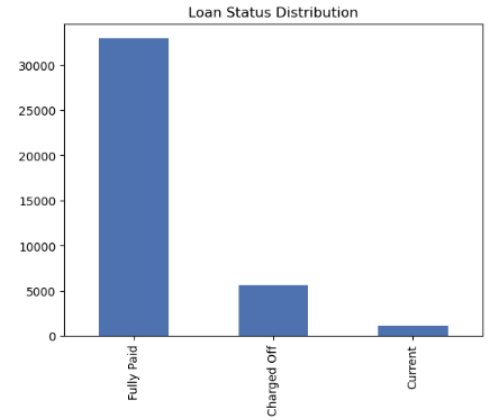
upGrad

Exploratory Data Analysis(EDA):Cont.

- **Data Overview:**
  - The loan_df['loan_status'].value_counts().plot(kind='bar', title='Loan Status Distribution')
  - value_counts(): This method counts the occurrences of each unique value in the 'loan_status' column and returns a Pandas Series.
  - plot(kind='bar'): This function is used to create a bar plot of the values in the Pandas Series. In this case, it creates a bar plot representing the counts of each unique value in the 'loan_status' column.
  - title='Loan Status Distribution': This sets the title of the bar plot to 'Loan Status Distribution'.

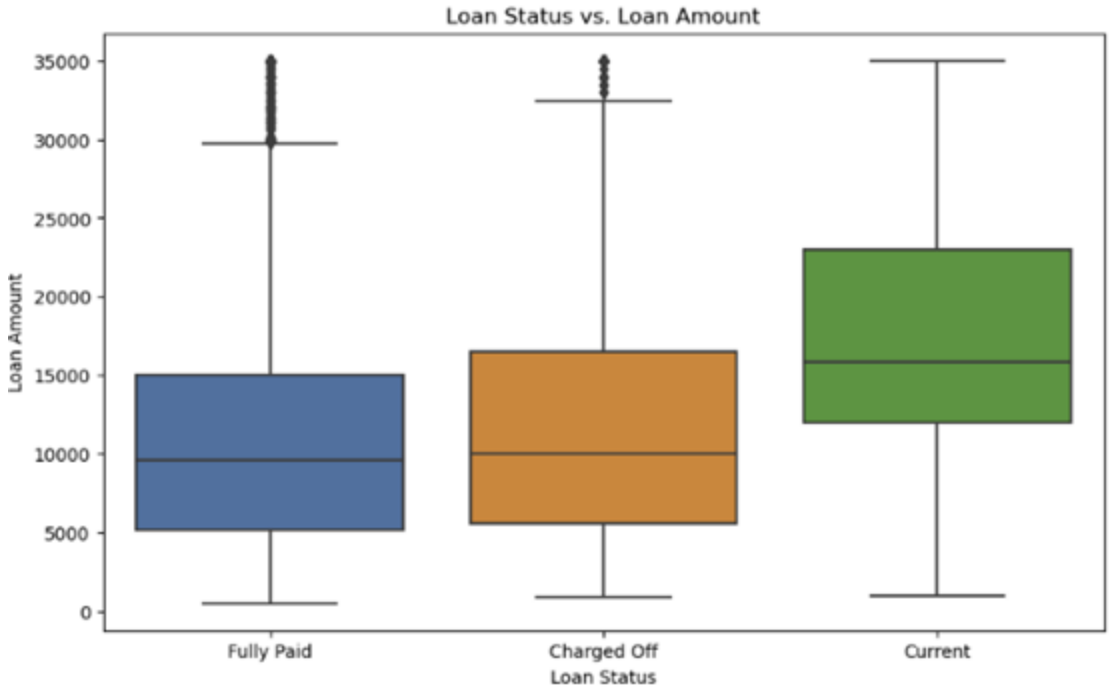
```
In [12]: # Check the distribution of the target variable
         loan_df['loan_status'].value_counts().plot(kind='bar', title='Loan Status Distribution')

Out[12]: <AxesSubplot:title={'center':'Loan Status Distribution'}>
```



upGrad

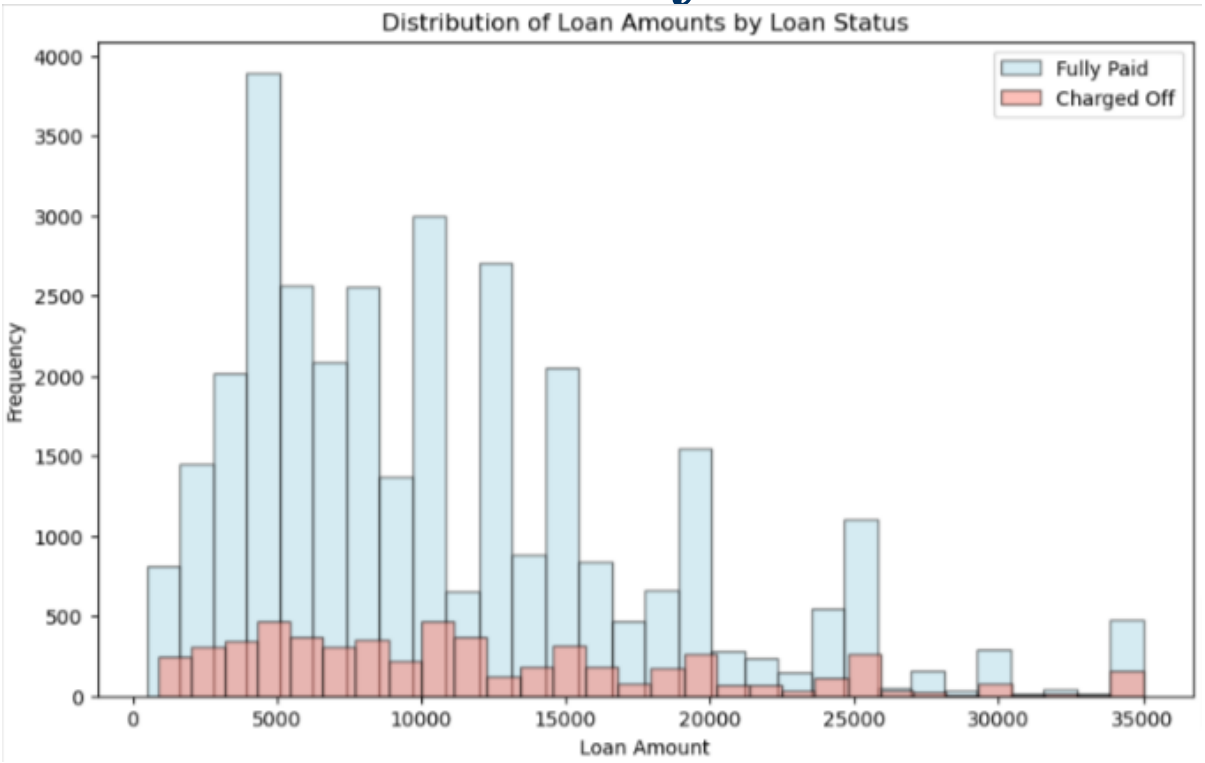
Exploratory Data Analysis(EDA):Cont.



•The boxplot visualizes the relationship between loan status and loan amount.

•This visualization provides insight into the distribution of loan amounts across different loan statuses, allowing for an initial assessment of the potential impact of loan amount on loan status.

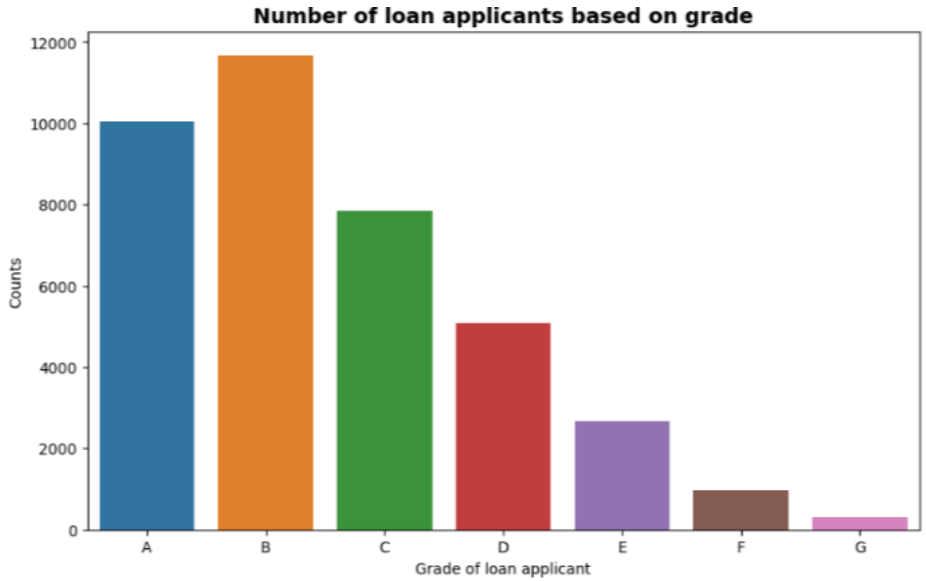
Loan Status Analysis



The histogram shows the distribution of loan amounts for loans that are "Fully Paid" and "Charged Off". The x-axis represents the loan amount, and the y-axis represents the frequency of each loan amount.

The graph provides a visual comparison of the distribution of loan amounts for fully paid and charged off loans, allowing for a better understanding of how loan amounts are related to loan status.

Loan Grade Analysis

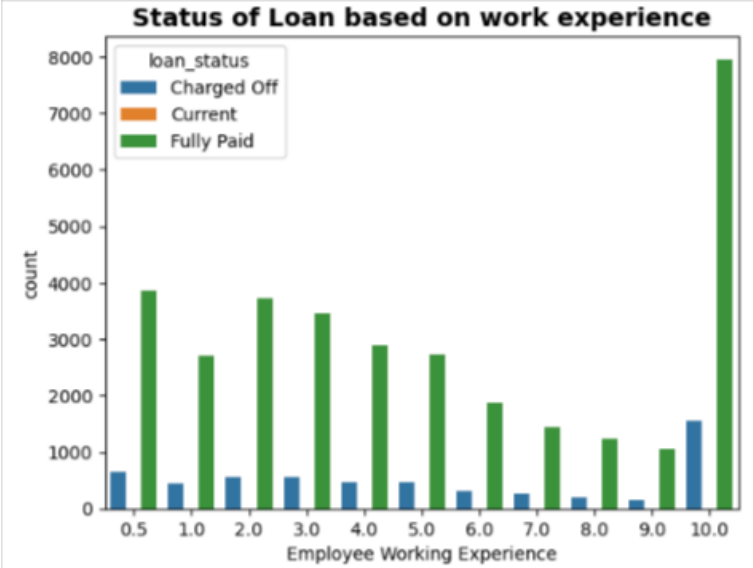


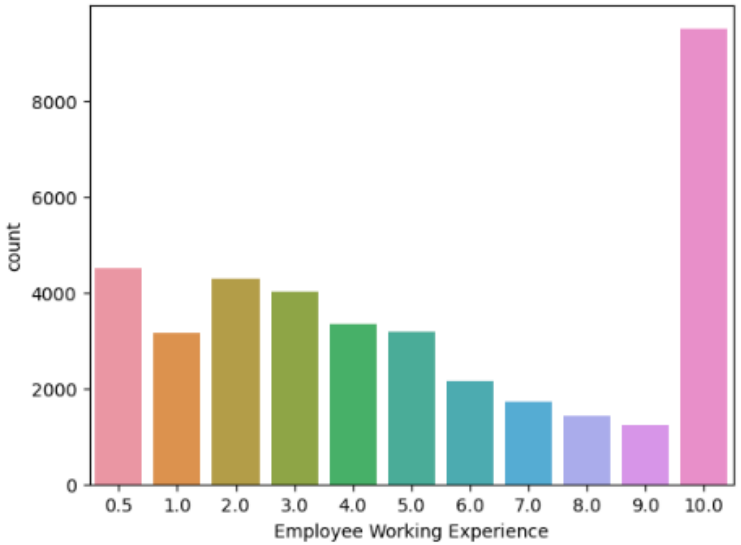
•Loans categorized as "High Quality" are associated with lower interest rates, reflecting a pattern where favorable loan terms are aligned with superior loan quality.

•This observation suggests a noteworthy correlation, indicating that loans with elevated interest rates exhibit a heightened propensity for default, emphasizing a potential risk factor associated with higher interest rate loans.

•The prevalence of loans graded as A and B signifies that the majority fall into the category of high-grade loans, highlighting a trend where a substantial portion of the loan portfolio is characterized by loans deemed to be of superior quality and lower risk.

Loan Applicants work Experience Analysis

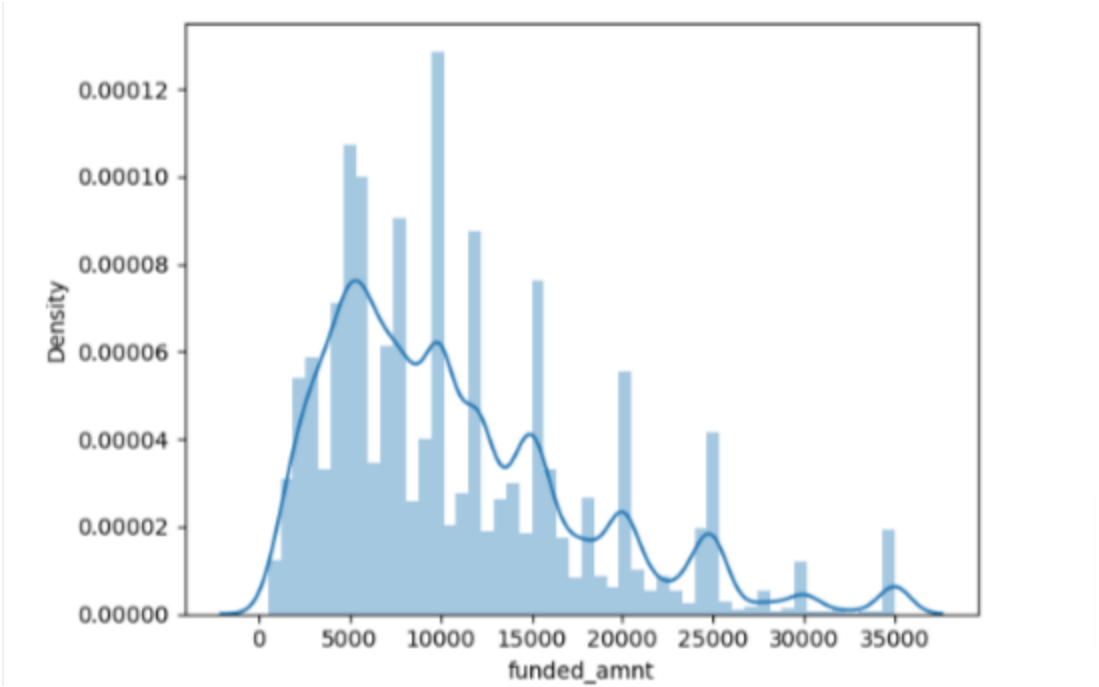




A predominant trend is observed among loan applicants, with the majority boasting over a decade of professional experience, indicating a notable concentration of seasoned employees seeking financial assistance.

Intriguingly, individuals with a decade or more of work experience display a heightened tendency to default on loans, underscoring the importance for the company to exercise caution and thorough risk assessment when approving loans for applicants within this employment experience bracket.

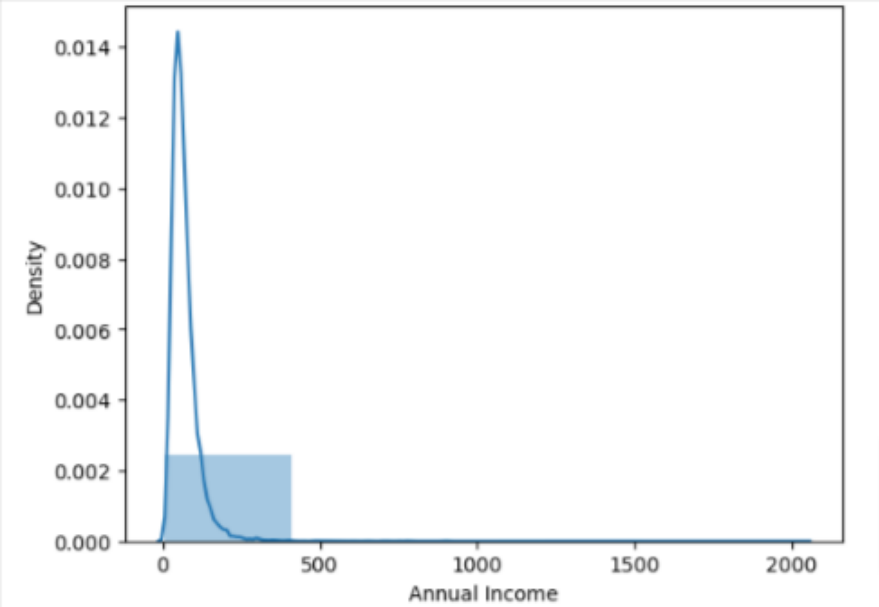
Loan Amount Analysis



The distribution of funded amounts is characterized by a left-skew, indicating that a substantial proportion of loan amounts granted falls below 7 lakhs, highlighting a concentration of loans in the lower funding range.

Notably, there is an observable trend where the likelihood of loan default tends to increase for individuals with loan amounts exceeding 7 lakhs, suggesting a higher probability of default among borrowers with larger loan obligations compared to those with amounts below this threshold.

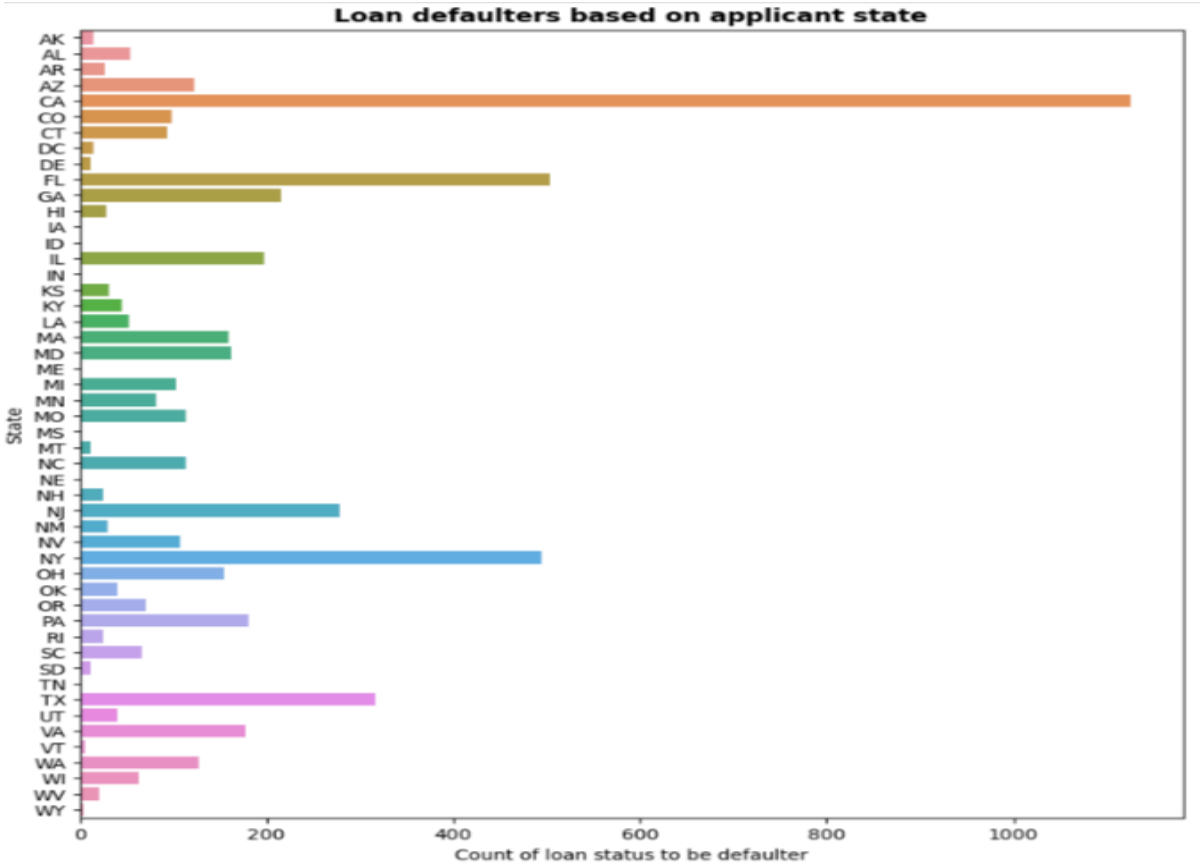
Income Analysis for the loan applicants



The resulting plot visualizes the distribution of annual income for applicants, considering only values less than 3000 lakhs. The histogram provides insights into the concentration of income values within different intervals, helping to understand the overall pattern of annual income in the specified range. Adjust the filtering condition or bin count based on your specific analysis needs.

upGrad

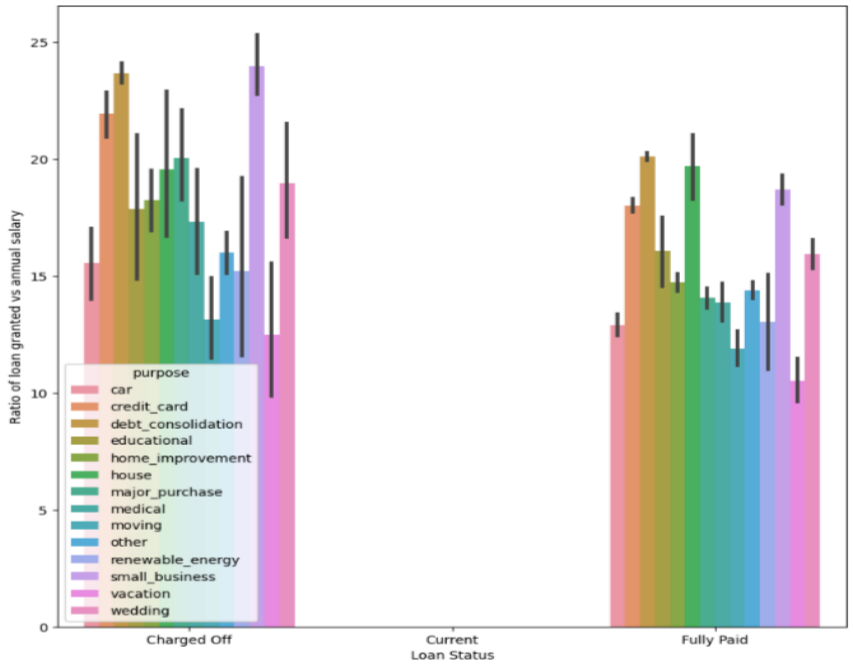
Loan Defaulters by State Analysis



Loan applicants residing in the state of California (CA) exhibit a notably higher propensity for loan default, suggesting a distinct trend in repayment behavior specific to this geographical region.

upGrad

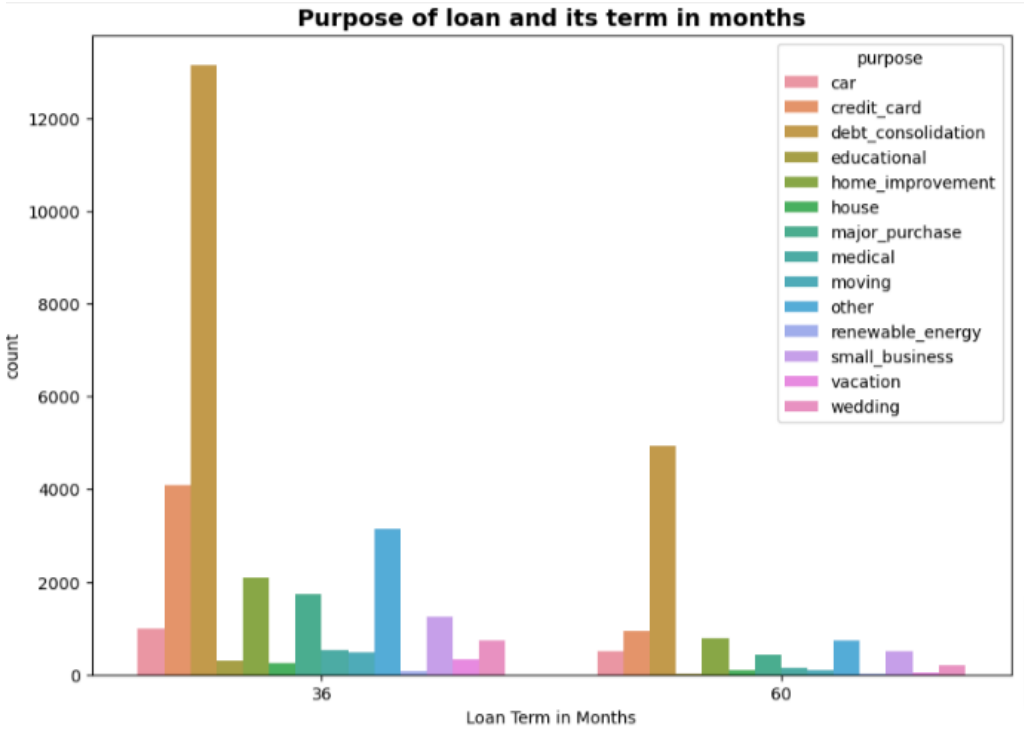
Purpose for applying Loan Analysis



1. Predominantly, loans are approved for the purpose of debt consolidation, indicating a prevalent financial need among applicants.

2. Notably, there is a discernible trend where applicants with higher income levels demonstrate a tendency to default on loans, highlighting a noteworthy association between income and loan repayment behavior.

Loan Status vs Loan Tenure

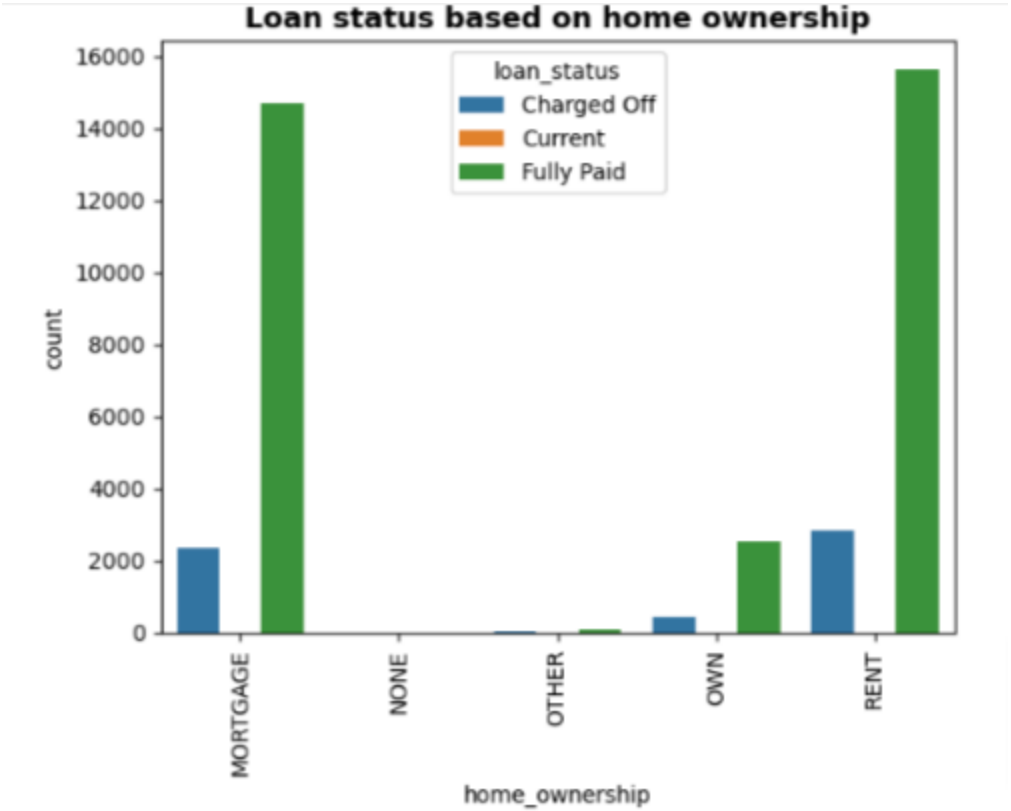


1. The majority of loans issued were for a 36-month term, indicating a prevalent choice among borrowers.

2. Interestingly, loans with a 36-month term exhibit a slightly higher propensity for default, suggesting a noteworthy trend in loan repayment patterns.

3. Notably, Debt Consolidation emerges as the predominant purpose for loan applications, with loans being approved for terms of both 36 months and 60 months, underscoring its popularity among borrowers.

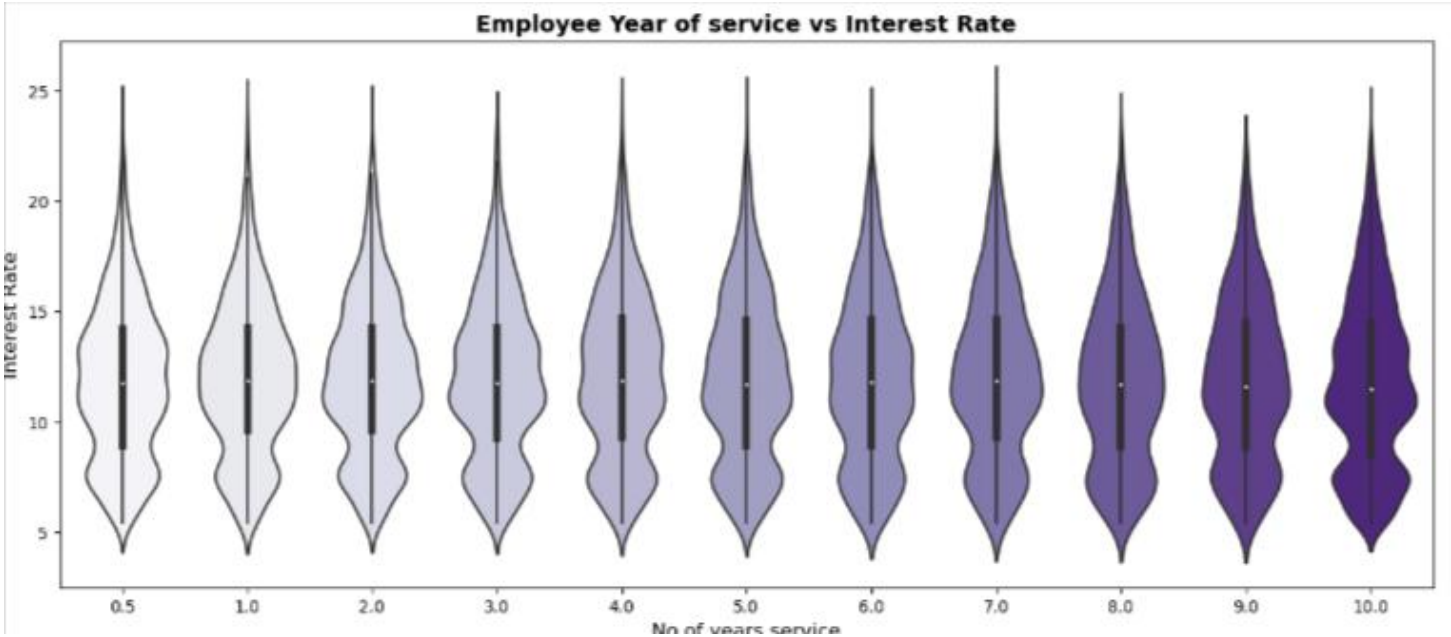
Loan Status by Home Ownership



Inference:

Applicants who either rent their homes or have a mortgage exhibit nearly identical tendencies to default on loans. The likelihood of loan default appears to be nearly equal for individuals in both housing situations.

Employee Year of Service Vs Interest Rate



1. Range of interest for applicant's year of service shows low range remains constant irrespective of years of services.
2. Maximum Interest rate is found for applicant having 7 years of service.
3. Maximum number of applicants have higher interest rate falling under 10 years of experience.

Conclusion

The analysis of the loan dataset has revealed valuable insights that can inform decision-making in the lending industry:

1. Loan Amount and Interest Rate: The relationship between loan amount and interest rate indicates that higher loan amounts tend to have higher interest rates. This suggests that thorough background checks of applicants should be conducted, especially for high-interest loans, as they have a higher likelihood of default.

2. Loan Grade and Default Risk: The segmented univariate analysis demonstrated that low-grade loans have a higher tendency to default. This highlights the importance of the grading system in predicting default risk and underscores the need for careful assessment and monitoring of low-grade loans.

Conclusion: Cont.

3. Annual Income and Loan Amount: The relationship between annual income and loan amount suggests that as the loan amount increases, the annual income also tends to increase. This insight can be used to assess the capacity of borrowers to repay higher loan amounts based on their income levels.

In conclusion, the dataset provides clear patterns and relationships that can be leveraged to understand the dynamics of loan funding and repayment. By considering the interplay between loan amount, interest rate, loan grade, and annual income, lenders can make informed decisions to mitigate default risk and ensure responsible lending practices.