Loan Approval Factors Analysis (Finance)

Problem Statement:

1. The objective of this project is to analyze how demographic factors (age, gender, education), financial indicators (income, employment experience, home ownership), and credit attributes (credit score, credit history, previous defaults) influence loan approval outcomes. Using Python, NumPy, Pandas, and hypothesis testing, the project aims to identify significant factors affecting loan approval decisions, evaluate the relationship between borrower characteristics and default risk, and provide insights that can help financial institutions in credit risk assessment and policy-making.

```
import numpy as np
import pandas as pd
```

```
df=pd.read_csv('/content/loan_data.csv')
```

To get the idea about the dataset

```
df.head()
```

	person_age	person_gender	person_education	person_income	person_emp_exp	person_home_ownership	loan_amnt	loan_intent	loan_int_rate	loan_percent_income	cb_person_cred_hist_length	credit_score	previous_loan_defaults_on_file	loan_status
0	22.0	female	Master	71948.0	0	RENT	35000.0	PERSONAL	16.02	0.49	3.0	561	No	1
1	21.0	female	High School	12282.0	0	OWN	1000.0	EDUCATION	11.14	0.08	2.0	504	Yes	0
2	25.0	female	High School	12438.0	3	MORTGAGE	5500.0	MEDICAL	12.87	0.44	3.0	635	No	1
3	23.0	female	Bachelor	79753.0	0	RENT	35000.0	MEDICAL	15.23	0.44	2.0	675	No	1
4	24.0	male	Master	66135.0	1	RENT	35000.0	MEDICAL	14.27	0.53	4.0	586	No	1

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45000 entries, 0 to 44999
Data columns (total 14 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   person_age                      45000 non-null  float64
 1   person_gender                   45000 non-null  object 
 2   person_education                45000 non-null  object 
 3   person_income                   45000 non-null  float64
 4   person_emp_exp                  45000 non-null  int64  
 5   person_home_ownership           45000 non-null  object 
 6   loan_amnt                       45000 non-null  float64
 7   loan_intent                     45000 non-null  object 
 8   loan_int_rate                   45000 non-null  float64
 9   loan_percent_income             45000 non-null  float64
 10  cb_person_cred_hist_length      45000 non-null  float64
 11  credit_score                    45000 non-null  int64  
 12  previous_loan_defaults_on_file  45000 non-null  object 
 13  loan_status                     45000 non-null  int64  
dtypes: float64(6), int64(3), object(5)
memory usage: 4.8+ MB
```

From this we would get to know that we have to convert 'person_gender', 'person_education', 'person_home_ownership', 'loan_intent ', 'previous_loan_defaults_on_file ', 'loan_status ' into categories

```
categorical_cols = [
    'person_gender',
    'person_education',
    'person_home_ownership',
    'loan_intent',
    'previous_loan_defaults_on_file',
    'loan_status'
]

df[categorical_cols] = df[categorical_cols].astype('category')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45000 entries, 0 to 44999
Data columns (total 14 columns):
 #   Column                          Non-Null Count  Dtype   
---  ------                          --------------  -----   
 0   person_age                      45000 non-null  float64 
 1   person_gender                   45000 non-null  category
 2   person_education                45000 non-null  category
 3   person_income                   45000 non-null  float64 
 4   person_emp_exp                  45000 non-null  int64   
 5   person_home_ownership           45000 non-null  category
 6   loan_amnt                       45000 non-null  float64 
 7   loan_intent                     45000 non-null  category
 8   loan_int_rate                   45000 non-null  float64 
 9   loan_percent_income             45000 non-null  float64 
 10  cb_person_cred_hist_length      45000 non-null  float64 
 11  credit_score                    45000 non-null  int64   
 12  previous_loan_defaults_on_file  45000 non-null  category
 13  loan_status                     45000 non-null  category
dtypes: category(6), float64(6), int64(2)
memory usage: 3.0 MB
```

So,we have converted all the essential data-types into category

```
#to check if any null values are present in the dataset
df.isnull().sum()
```

	0
person_age	0
person_gender	0
person_education	0
person_income	0
person_emp_exp	0
person_home_ownership	0
loan_amnt	0
loan_intent	0
loan_int_rate	0
loan_percent_income	0
cb_person_cred_hist_length	0
credit_score	0
previous_loan_defaults_on_file	0
loan_status	0

dtype: int64

NULL values are not present in the data

```
#summary statistics for numeric columns
df.describe()
```

	person_age	person_income	person_emp_exp	loan_amnt	loan_int_rate	loan_percent_income	cb_person_cred_hist_length	credit_score
count	45000.000000	4.500000e+04	45000.000000	45000.000000	45000.000000	45000.000000	45000.000000	45000.000000
mean	27.764178	8.031905e+04	5.410333	9583.157556	11.006606	0.139725	5.867489	632.608756
std	6.045108	8.042250e+04	6.063532	6314.886691	2.978808	0.087212	3.879702	50.435865
min	20.000000	8.000000e+03	0.000000	500.000000	5.420000	0.000000	2.000000	390.000000
25%	24.000000	4.720400e+04	1.000000	5000.000000	8.590000	0.070000	3.000000	601.000000
50%	26.000000	6.704800e+04	4.000000	8000.000000	11.010000	0.120000	4.000000	640.000000
75%	30.000000	9.578925e+04	8.000000	12237.250000	12.990000	0.190000	8.000000	670.000000
max	144.000000	7.200766e+06	125.000000	35000.000000	20.000000	0.660000	30.000000	850.000000

There are some issues with the data-set, it contains outliers like,

1. person_age max = 144
2. person_income max = 7,200,766
3. person_emp_exp max = 125

```
# Remove outliers
for col in ['person_age','person_income','person_emp_exp','loan_amnt']:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5*IQR
    upper = Q3 + 1.5*IQR
    df = df[(df[col] >= lower) & (df[col] <= upper)]
```

```
df.describe()
```

	person_age	person_income	person_emp_exp	loan_amnt	loan_int_rate	loan_percent_income	cb_person_cred_hist_length	credit_score
count	38544.000000	38544.000000	38544.000000	38544.000000	38544.000000	38544.000000	38544.000000	38544.000000
mean	26.605879	69052.325394	4.251219	8490.399388	10.914487	0.137932	5.209656	630.845657
std	4.111754	31598.062920	4.079549	4898.562706	2.947116	0.083506	3.003094	50.089002
min	20.000000	8000.000000	0.000000	500.000000	5.420000	0.010000	2.000000	390.000000
25%	23.000000	44525.000000	1.000000	4997.000000	8.490000	0.080000	3.000000	600.000000
50%	26.000000	63376.000000	3.000000	7500.000000	11.010000	0.120000	4.000000	638.000000
75%	29.000000	87473.500000	7.000000	12000.000000	12.980000	0.180000	7.000000	668.000000
max	39.000000	166754.000000	16.000000	22500.000000	20.000000	0.660000	17.000000	762.000000

So,the extreme values are gone

```
df.shape
```

```
(38544, 14)
```

We,have 38544 rows and 14 columns

Descriptive Analysis

```
print(df['loan_amnt'].mean())
```

```
8490.399387712743
```

Avg loan amount taken by applicants is 8490.39 USD

```
print(df['loan_amnt'].quantile(0.50))
```

```
7500.0
```

median loan amount taken by applicants is 7500 USD

```
commen_intent=df['loan_intent'].value_counts()
commen_intent
```

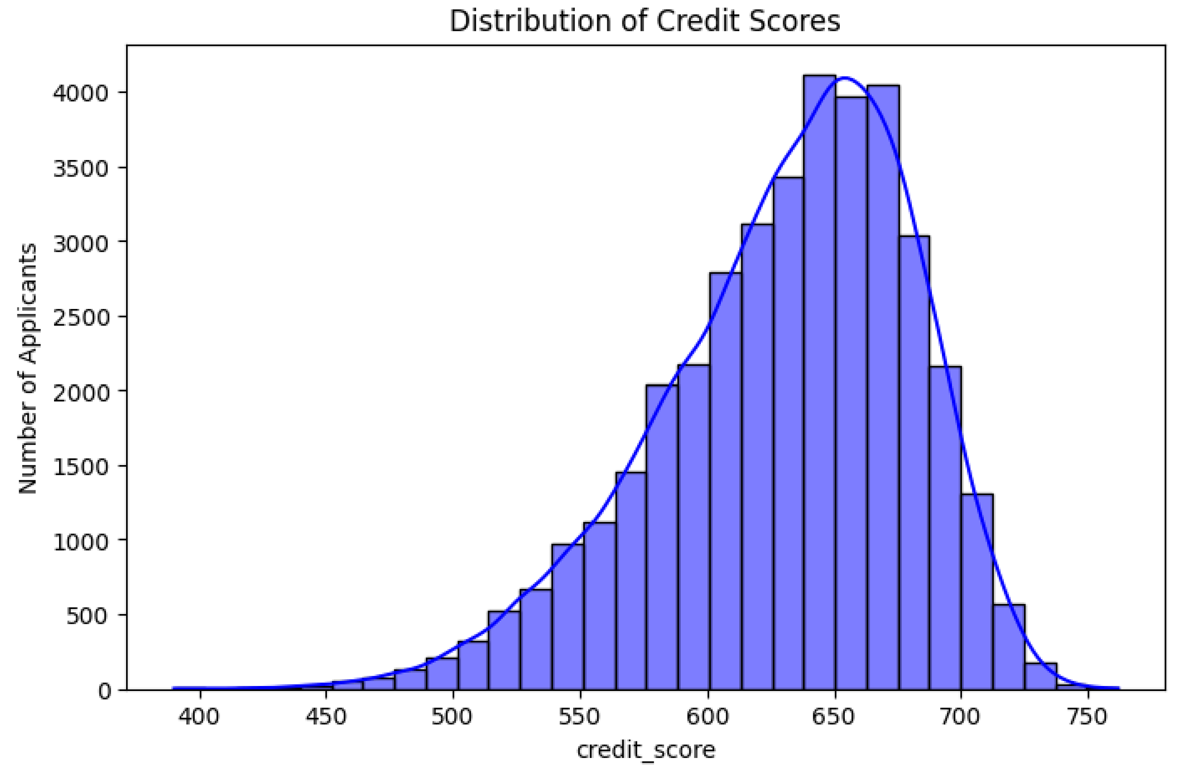
loan_intent	count
EDUCATION	8013
MEDICAL	7458
VENTURE	6718
PERSONAL	6348
DEBTCONSOLIDATION	6087
HOMEIMPROVEMENT	3920

dtype: int64

So,the most frequent loan intent is EDUCATION

```
#To get the distribution of credit scores across applicants we will draw histogram
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8,5))
sns.histplot(df['credit_score'],bins=30,kde=True,color='blue')
plt.title("Distribution of Credit Scores")
plt.xlabel('credit_score')
plt.ylabel("Number of Applicants")
plt.show()
```



Most credit scores usually cluster around 600–700 (average borrowers).

Very few applicants fall in extremely low (<500) or very high (>750) ranges.

Our data is negatively skewed

```
df.groupby('person_gender')['loan_amnt'].mean()
```

```
/tmp/ipython-input-3815762641.py:1: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  df.groupby('person_gender')['loan_amnt'].mean()
```

person_gender	loan_amnt
female	8466.478710
male	8510.138081

dtype: float64

So, female's avg loan_amount is 8466.47 and male's avg loan_amount is 8510.13.

Male apply for loan_amount slightly higher than female

```
df['person_home_ownership'].value_counts()
```

person_home_ownership	count
RENT	21062
MORTGAGE	14860
OWN	2526
OTHER	96

dtype: int64

1. Most applicants are renters (RENT = 21,062)
2. Mortgage holders form the second-largest group (MORTGAGE = 14,860)
3. Small percentage fully own homes (OWN = 2,526)
4. Other category is negligible (OTHER = 96)

Comparative Analysis

```
# Group by previous loan defaults and calculate mean interest rate
avg_int_rate = df.groupby("previous_loan_defaults_on_file")["loan_int_rate"].mean()
avg_int_rate
```

```
/tmp/ipython-input-1182975215.py:2: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  avg_int_rate = df.groupby("previous_loan_defaults_on_file")["loan_int_rate"].mean()
```

	loan_int_rate
previous_loan_defaults_on_file	
No	11.454365
Yes	10.402818

dtype: float64

Normally, we expect people with defaults to have higher interest rates (because lenders consider them risky).

But here, applicants without defaults are paying higher interest rates on average.

This result suggests interest rate is not strongly aligned with default history in this dataset.

Lenders might be using other factors (like credit score, income, or loan purpose) more heavily to decide the rate.

```
#Now we check for the avg credit score
avg_credit_score = df.groupby("previous_loan_defaults_on_file")["credit_score"].mean()
avg_credit_score
```

```
/tmp/ipython-input-3265572340.py:2: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  avg_credit_score = df.groupby("previous_loan_defaults_on_file")["credit_score"].mean()
```

	credit_score
previous_loan_defaults_on_file	
No	640.136764
Yes	622.040022

dtype: float64

Applicants without defaults have higher credit scores (640 vs 622). People with defaults are not only riskier but also have weaker credit scores, making them less attractive for approval. Even though defaulted applicants had slightly lower loan interest rates, their credit scores are significantly lower.

```
#Now we compare the avg incomes of different education category
avg_income_per_education= df.groupby('person_education')['person_income'].mean()
avg_income_per_education
```

```
/tmp/ipython-input-1468667398.py:2: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  avg_income_per_education= df.groupby('person_education')['person_income'].mean()
```

	person_income
person_education	
Associate	68706.815891
Bachelor	69311.518791
Doctorate	70344.009547
High School	68558.400635
Master	69850.500410

dtype: float64

Higher education levels generally correspond to higher average incomes.

Doctorate holders earn the most, followed by Masters and Bachelors.

The gap between High School (68,558) and Doctorate (70,344) is only about 1,786.

This suggests that in this dataset, education has only a modest effect on income.

```
# Now we check for the loan percent income  higher for younger applicants (<25) compared to older ones

df['age_grp']= df['person_age'].apply(lambda x: '<25' if x <25 else '25+')

avg_lpi_by_age=df.groupby('age_grp')['loan_percent_income'].mean()
avg_lpi_by_age
```

	loan_percent_income
age_grp	
25+	0.136247
<25	0.140577

dtype: float64

Younger borrowers (<25) take on loans that are a slightly larger share of their income compared to older borrowers.

This makes sense: younger people usually have lower income levels, so the same loan amount eats up more of their salary.

here,loan_percent_income = (Loan Amount / Applicant’s Income)

```
avg_loan_by_home = df.groupby("person_home_ownership")["loan_amnt"].mean()
avg_loan_by_home
```

```
/tmp/ipython-input-3389841605.py:1: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  avg_loan_by_home = df.groupby("person_home_ownership")["loan_amnt"].mean()
```

	loan_amnt
person_home_ownership	
MORTGAGE	9163.732100
OTHER	10487.833333
OWN	7849.724070
RENT	8083.071883

dtype: float64

Here,

People with a MORTGAGE already have a housing loan, so they may request larger additional loans (education, medical, etc.).

The OTHER category shows the largest average, but since this group is usually very small, the number may not be very reliable because of outlier effect

People who fully own their home request the smallest loans on average (7,850). Makes sense: they already have strong assets and need less borrowing.

RENT applicants borrow moderate amounts (8,083). Likely because they don't have property as collateral, lenders might limit the loan size.

Correlation

```
# We calculate correlation between income and loan amount
corr = df["person_income"].corr(df["loan_amnt"])
corr
```

```
np.float64(0.323219255144122)
```

Positive correlation

Since it's > 0, there is a positive relationship: Higher income applicants tend to request higher loan amounts.

```
#to check if loan_int_rate increases as the loan_percent_income rises.

corr1=df['loan_int_rate'].corr(df['loan_percent_income'])
corr1
```

```
np.float64(0.10716958506582809)
```

Weak positive correlation

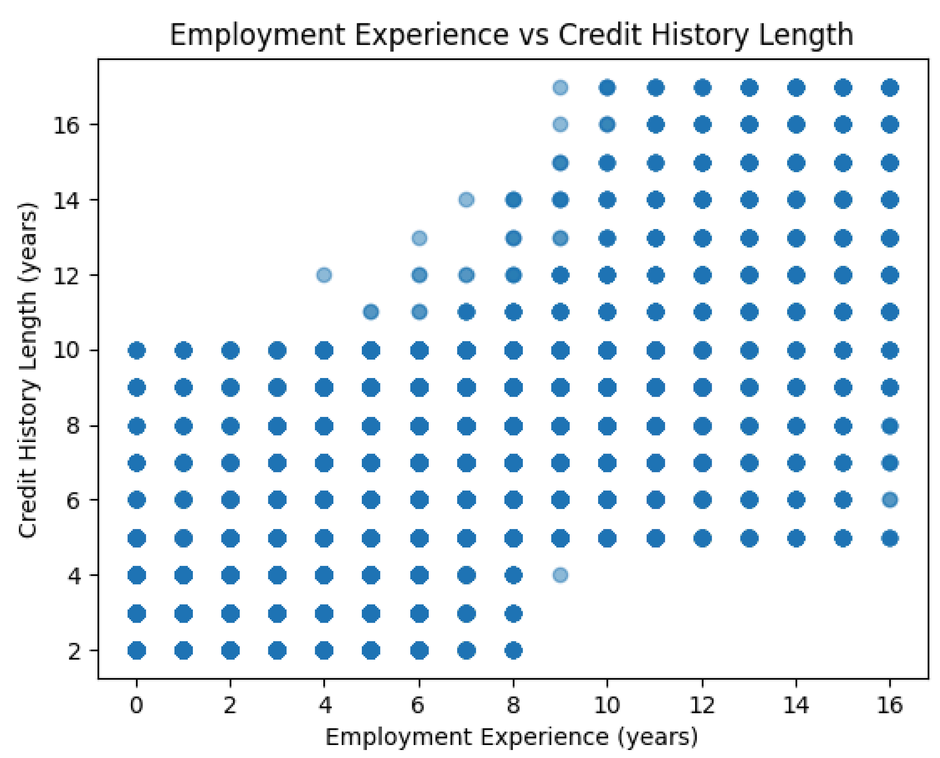
Since the correlation is > 0, there is a slight tendency for loan interest rate to increase as loan_percent_income rises.

However, 0.107 is very weak, so the relationship is minimal.

```
corr = df["person_emp_exp"].corr(df["cb_person_cred_hist_length"])
print("Correlation between Employment Experience and Credit History Length:", corr)
```

```
Correlation between Employment Experience and Credit History Length: 0.7483278978023503
```

```
#scatterplot to visualize
plt.scatter(df["person_emp_exp"], df["cb_person_cred_hist_length"], alpha=0.5)
plt.xlabel("Employment Experience (years)")
plt.ylabel("Credit History Length (years)")
plt.title("Employment Experience vs Credit History Length")
plt.show()
```



Strong positive correlation

Since the correlation is 0.748, there is a strong relationship between employment experience and credit history length.

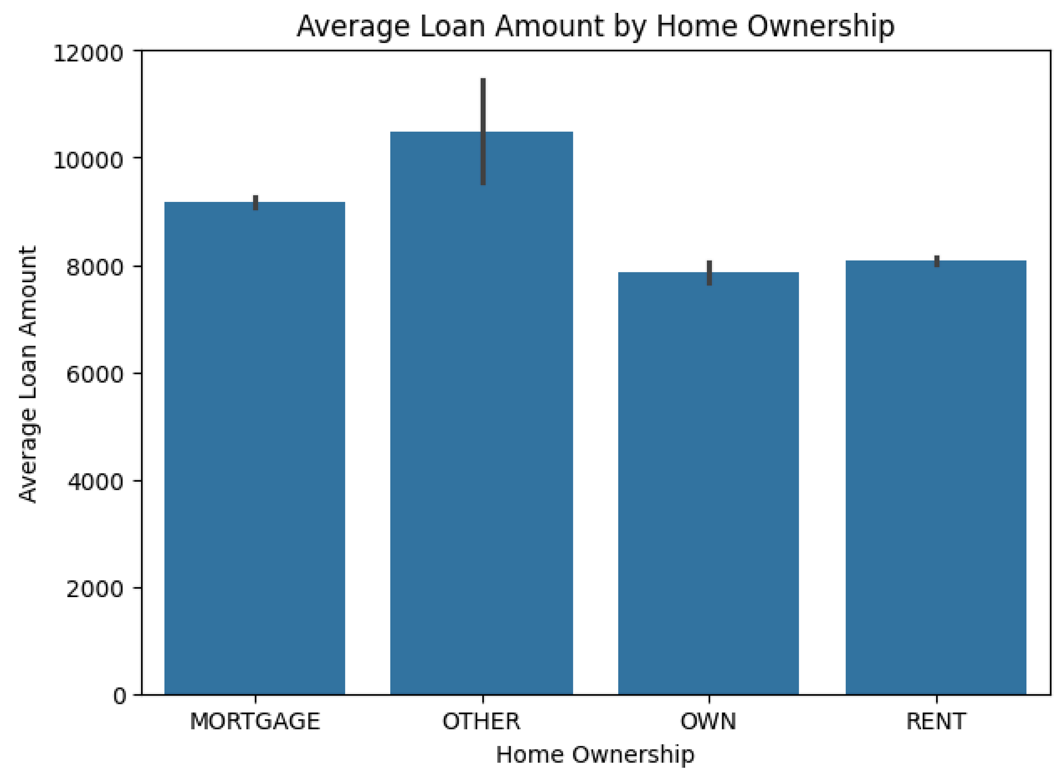
In simple terms: the more years someone has worked, the longer their credit history tends to be.

visualizations

**Loan Amount by Home Ownership (Barplot)

```
plt.figure(figsize=(7,5))
sns.barplot(x="person_home_ownership", y="loan_amnt", data=df)
plt.xlabel("Home Ownership")
plt.ylabel("Average Loan Amount")
```

```
plt.title("Average Loan Amount by Home Ownership")
plt.show()
```



The black line on a bar plot shows the uncertainty of the mean, giving you an idea of how much the data varies in that category.

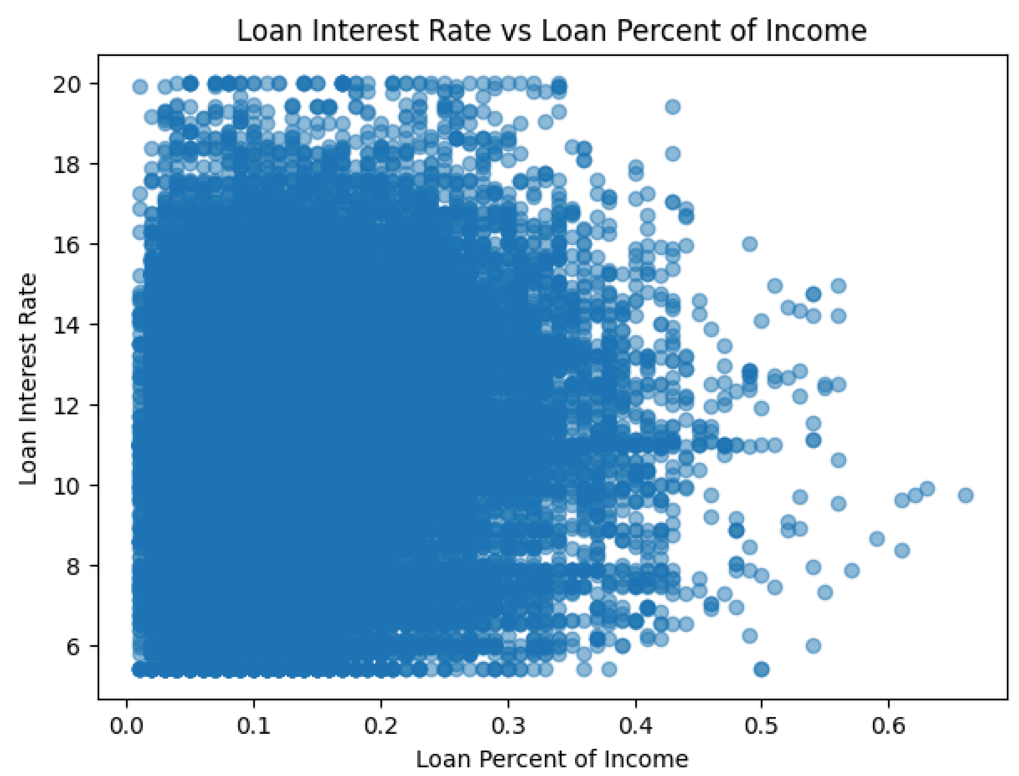
(i.e., the range in which the true mean likely falls)

Longer lines : more variability in the data.

Shorter lines : data points are close to the mean, less variability.

**Loan Interest Rate vs Loan Percent of Income (Scatter Plot)

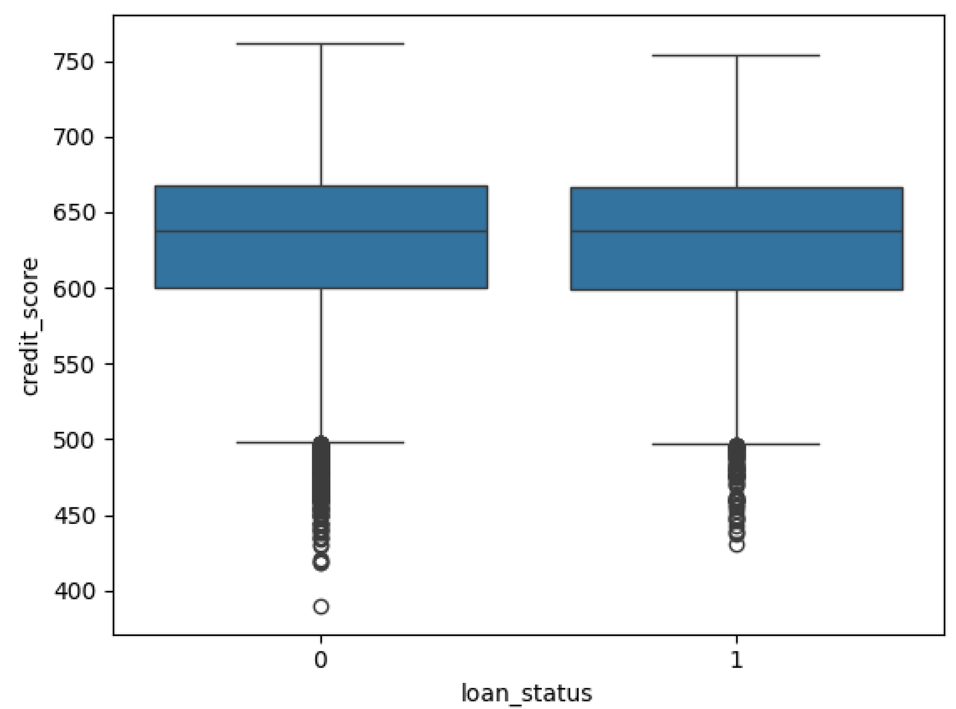
```
plt.figure(figsize=(7,5))
plt.scatter(df["loan_percent_income"], df["loan_int_rate"], alpha=0.5, )
plt.xlabel("Loan Percent of Income")
plt.ylabel("Loan Interest Rate")
plt.title("Loan Interest Rate vs Loan Percent of Income")
plt.show()
```



**Credit Score Distribution by Loan Status (Boxplot)

```
sns.boxplot(x="loan_status", y="credit_score", data=df)
```

```
<Axes: xlabel='loan_status', ylabel='credit_score'>
```



Presence of Outliers:

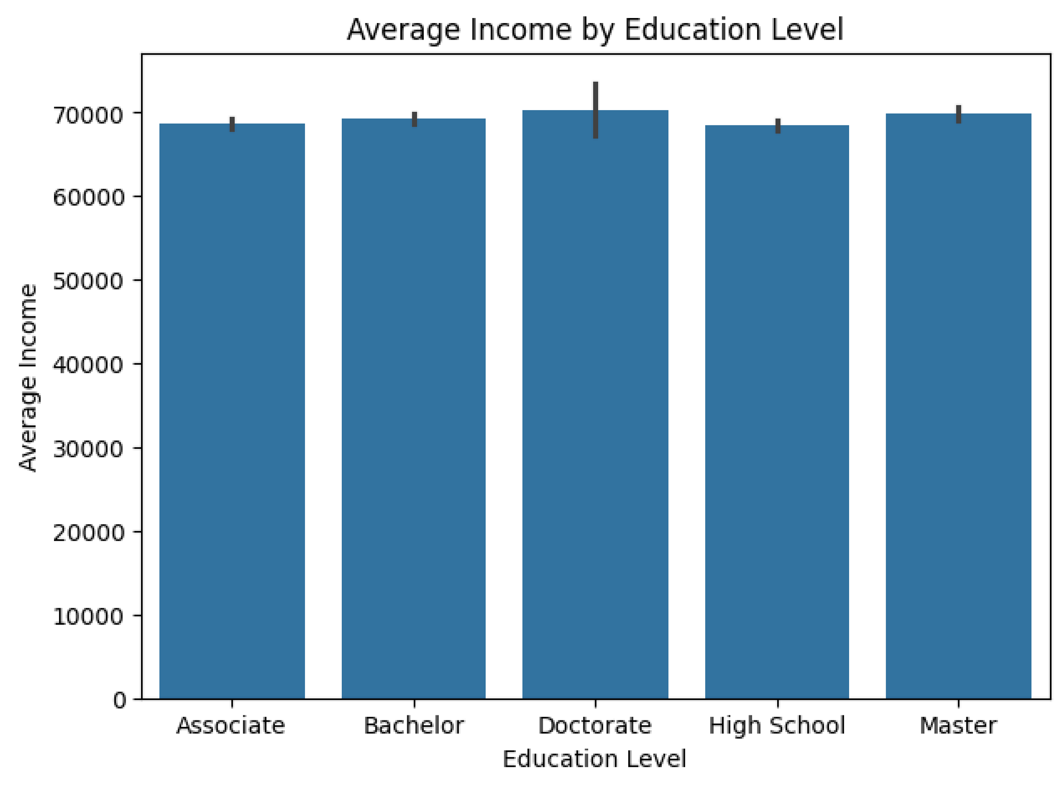
There are outliers on the lower end for both rejected and approved loans.

Outliers in rejected loans might represent applicants with very poor credit history.

Outliers in approved loans could indicate cases where other factors outweighed the lower credit score.

**Average Income by Education Level (Barplot)

```
plt.figure(figsize=(7,5))
sns.barplot(x="person_education", y="person_income", data=df)
plt.xlabel("Education Level")
plt.ylabel("Average Income")
plt.title("Average Income by Education Level")
plt.show()
```



Hypothesis testing

1] For,Gender vs Loan Approval

both are categorical

so, we use chi-squared test here

Null Hypothesis (H0): Loan approval is independent of gender (gender has no effect).

Alternative Hypothesis (H1): Loan approval is dependent on gender (gender does have an effect).

```
from scipy.stats import chi2_contingency

contingency = pd.crosstab(df['person_gender'], df['loan_status'])

# Chi-square test
chi2, p, dof, expected = chi2_contingency(contingency)

print("Chi-square Statistic:", chi2)
print("Degrees of Freedom:", dof)
print("p-value:", p)
```

```
Chi-square Statistic: 0.11107132517758744
Degrees of Freedom: 1
p-value: 0.738927728476593
```

So here p-value is 0.738927728476593 which is greater than significance level 0.05

from this we can conclude that we fail to reject H0

so,Loan approval is independent of gender (gender has no effect).

2] for,Credit Score vs Loan Status

one is numerical and one is categorical

so,we use independent t-test here

H0 (Null): The mean credit score is the same for approved and not approved loans.

H1 (Alternative): The mean credit score differs between approved and not approved loans.

```
from scipy.stats import ttest_ind

approved=df[df['loan_status']==1]['credit_score']
rejected=df[df['loan_status']==0]['credit_score']

t,p=ttest_ind(rejected,approved)
print('t-stats:',t)
print('pvalue:',p)
```

```
t-stats: 1.1843533714069354
pvalue: 0.2362805134352665
```

The t-test result gave a p-value of 0.236, which is greater than the 0.05 threshold. This means there is no statistically significant difference in mean credit scores between applicants whose loans were approved and those whose loans were not approved. In other words, in this dataset, credit score does not appear to strongly influence loan approval decisions.

3]Home ownership vs loan approval

H0: Loan approval is independent of home ownership.

H1: Loan approval is dependent on home ownership.

```
contingency = pd.crosstab(df['person_home_ownership'], df['loan_status'])

chi2, p, dof, expected = chi2_contingency(contingency)

print("Chi-square Statistic:", chi2)
print("p-value:", p)
print("Degrees of Freedom:", dof)
```

```
Chi-square Statistic: 2399.7489923092617
p-value: 0.0
Degrees of Freedom: 3
```

The Chi-Square test gave a statistic of 2399.75 with 3 degrees of freedom and a p-value ≈ 0.0. This means we reject the null hypothesis and conclude that home ownership type and loan approval status are not independent. In other words, loan approval is significantly associated with whether a person rents, owns, or has a mortgage.

4] Loan_intent vs intrest_rate

H0: Mean interest rate is the same across loan intent categories.

H1: At least one loan intent category has a different mean interest rate.

```
from scipy.stats import f_oneway

groups = df.groupby('loan_intent')['loan_int_rate'].apply(list)
f_stat, p_val = f_oneway(*groups)
print('f-stat:',f_stat)
print('p_val:',p_val)
```

```
f-stat: 7.378105909188488
p_val: 6.347173757789848e-07
/tmp/ipython-input-2557624856.py:3: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  groups = df.groupby('loan_intent')['loan_int_rate'].apply(list)
```

The ANOVA test for loan intent vs interest rate produced a p-value of 6.3e-07, which is far below 0.05. This means we reject the null hypothesis and conclude that mean interest rates differ significantly across loan intents

Key Insights

1. **Applicant Demographics**

Age: Most applicants are young (20–30 years). Younger borrowers (<25) spend a slightly larger share of income on loans.

Gender: Loan approval is independent of gender (p = 0.73). Males request marginally higher loan amounts than females.

Education: Higher education correlates with slightly higher income, but the income gap between High School and Doctorate is small (~$1,700 difference).

2. **Financial Profile**

Income & Loan Amount : Positive correlation (0.32). Higher income tends higher loan request.

Employment vs Credit History: Strong correlation (0.75). More job experience tends longer credit history.

Home Ownership: Strong effect on approval (Chi-square significant). Mortgage holders and renters differ notably from owners.

3. **Loan Characteristics**

Loan Amounts: Median loan : 7500; mean: 8,490.

Loan Intent: Education loans are most common, followed by Medical and Venture. Interest rates vary significantly by loan purpose (ANOVA p < 0.001).

Loan Percent of Income: Younger borrowers have higher ratios (riskier debt burden).

4. **Credit Profile**

Credit Scores: Clustered around 600–700. Few applicants in very low (<500) or very high (>750) ranges.

Defaults: Applicants with defaults have lower credit scores (622 vs 640) but paradoxically lower interest rates. Suggests credit score weighs more than default history in rate assignment.

Approval vs Credit Score: Surprisingly, approval is not significantly different across credit score groups (p = 0.23).

Recommendations

For Banks & Financial Institutions

1] Reassess Loan Approval Criteria

Credit score alone is not differentiating approvals; consider integrating loan percent income and employment stability more strongly.

Use home ownership type as a weighted factor in approval, since it shows strong correlation with loan outcomes.

2] Improve Risk-based Pricing

Current interest rate assignment seems inconsistent (defaults have lower rates). Align pricing better with actual risk by combining credit score, income-to-loan ratio, and past defaults.

3] Focus on Young Borrowers

Younger applicants (<25) take riskier loans (higher % of income). Implement stricter checks (e.g., co-signer requirement, capped loan size).

4] Product Differentiation by Loan Intent

Education loans are dominant. Offer custom loan products with flexible repayment terms for students.

Medical and debt consolidation loans also large — tailor products with risk-adjusted pricing.

For Policy & Risk Management

1] Encourage Credit Building Early

Since employment experience and credit history are strongly related, incentivize young workers to start building credit responsibly.

2] Outlier & Fraud Detection

Extreme values in income, age, and experience (e.g., 144 years old, income > $7M) suggest possible data quality issues or fraud. Stronger KYC/validation rules are needed.

3] Educate Borrowers

Many loans are taken for education/medical reasons — institutions can run financial literacy programs to help borrowers manage debt-to-income ratios.

4] Alternative Credit Scoring Models

Since traditional credit score is not fully predictive of approval in this dataset, lenders could explore machine learning models using multiple borrower attributes.