Lending Club Case Study Notebook

Introduction

Goal

```
How data can be used minimize the risk of losing money
while lending to customers.
```

Context of Problem

```
This company is the largest online loan marketplace,
facilitating personal loans, business loans, and financing
of
medical procedures. Borrowers can easily access lower
interest rate loans through a fast online interface.
```

Business Problem:

```
Like most other lending companies, lending loans to
‘risky’ applicants is the largest source of financial loss
(called credit loss). Credit loss is the amount of money
lost by the lender when the borrower refuses to pay or
runs away with the money owed. In other words, borrowers
who default cause the largest amount of loss to the
lenders. In this case, the customers labelled as 'charged-
off' are the 'defaulters'.
```

Target:

```
If one is able to identify these risky loan applicants,
then such loans can be reduced thereby cutting down the
amount of credit loss. Identification of such applicants
using EDA is the aim of this case study.
```

Risk associated with the problem

```
- If the applicant is likely to repay the loan, then not
approving loan is a loss of business (rejecting loans for
non - default).

- If the applicant is not likely to repay the loan, then
approving loan may lead to financial loss (approving loans
for default). The given dataset contains information about
past loans and each row represents the loan details of the
applicants.
```

Datset:

```
Dataset contains loan data for all loans issued through
the time period 2007 t0 2011.
```

In [1]:
```
%load_ext autoreload
%autoreload 2
```

In [2]:
```
!pip install -r requirements.txt
```

```
Requirement already satisfied: matplotlib==3.9.2 in /Users/shashank.khande
lwal/miniconda3/envs/ln_case_study/lib/python3.12/site-packages (from -r r
equirements.txt (line 1)) (3.9.2)
Requirement already satisfied: numpy==2.0.1 in /Users/shashank.khandelwal/
miniconda3/envs/ln_case_study/lib/python3.12/site-packages (from -r requir
ements.txt (line 2)) (2.0.1)
Requirement already satisfied: openpyxl==3.1.5 in /Users/shashank.khandelw
al/miniconda3/envs/ln_case_study/lib/python3.12/site-packages (from -r req
uirements.txt (line 3)) (3.1.5)
Requirement already satisfied: pandas==2.2.2 in /Users/shashank.khandelwa
l/miniconda3/envs/ln_case_study/lib/python3.12/site-packages (from -r requ
irements.txt (line 4)) (2.2.2)
Requirement already satisfied: scipy==1.14.0 in /Users/shashank.khandelwa
l/miniconda3/envs/ln_case_study/lib/python3.12/site-packages (from -r requ
irements.txt (line 5)) (1.14.0)
Requirement already satisfied: seaborn==0.13.2 in /Users/shashank.khandelw
al/miniconda3/envs/ln_case_study/lib/python3.12/site-packages (from -r req
uirements.txt (line 6)) (0.13.2)
Requirement already satisfied: contourpy>=1.0.1 in /Users/shashank.khandel
wal/miniconda3/envs/ln_case_study/lib/python3.12/site-packages (from matpl
otlib==3.9.2->-r requirements.txt (line 1)) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /Users/shashank.khandelwal/
miniconda3/envs/ln_case_study/lib/python3.12/site-packages (from matplotli
b==3.9.2->-r requirements.txt (line 1)) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /Users/shashank.khande
lwal/miniconda3/envs/ln_case_study/lib/python3.12/site-packages (from matp
lotlib==3.9.2->-r requirements.txt (line 1)) (4.53.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /Users/shashank.khande
lwal/miniconda3/envs/ln_case_study/lib/python3.12/site-packages (from matp
lotlib==3.9.2->-r requirements.txt (line 1)) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /Users/shashank.khandelw
al/miniconda3/envs/ln_case_study/lib/python3.12/site-packages (from matplo
tlib==3.9.2->-r requirements.txt (line 1)) (24.1)
Requirement already satisfied: pillow>=8 in /Users/shashank.khandelwal/min
iconda3/envs/ln_case_study/lib/python3.12/site-packages (from matplotlib==
3.9.2->-r requirements.txt (line 1)) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /Users/shashank.khandel
wal/miniconda3/envs/ln_case_study/lib/python3.12/site-packages (from matpl
otlib==3.9.2->-r requirements.txt (line 1)) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /Users/shashank.kha
ndelwal/miniconda3/envs/ln_case_study/lib/python3.12/site-packages (from m
atplotlib==3.9.2->-r requirements.txt (line 1)) (2.9.0.post0)
Requirement already satisfied: et-xmlfile in /Users/shashank.khandelwal/mi
```

```
niconda3/envs/ln_case_study/lib/python3.12/site-packages (from openpyxl==
3.1.5->-r requirements.txt (line 3)) (1.1.0)
Requirement already satisfied: pytz>=2020.1 in /Users/shashank.khandelwal/
miniconda3/envs/ln_case_study/lib/python3.12/site-packages (from pandas==
2.2.2->-r requirements.txt (line 4)) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in /Users/shashank.khandelwa
l/miniconda3/envs/ln_case_study/lib/python3.12/site-packages (from pandas=
=2.2.2->-r requirements.txt (line 4)) (2024.1)
Requirement already satisfied: six>=1.5 in /Users/shashank.khandelwal/mini
conda3/envs/ln_case_study/lib/python3.12/site-packages (from python-dateut
il>=2.7->matplotlib==3.9.2->-r requirements.txt (line 1)) (1.16.0)
```

In [3]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

## Import helpers function
from helpers import load_loan_data
```

In [4]:

```
loan_data_df = load_loan_data()
loan_data_df.head()
```

```
/Users/shashank.khandelwal/Library/CloudStorage/OneDrive-Logility/pcode/Le
ndingClubCaseStudy/helpers.py:5: DtypeWarning: Columns (47) have mixed typ
es. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv('loan.csv')
```

Out[4]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_r
0	1077501	1296599	5000	5000	4975.0	36 months	10.6
1	1077430	1314167	2500	2500	2500.0	60 months	15.2
2	1077175	1313524	2400	2400	2400.0	36 months	15.9
3	1076863	1277178	10000	10000	10000.0	36 months	13.4
4	1075358	1311748	3000	3000	3000.0	60 months	12.6

5 rows × 111 columns

Data Cleanup

As per above warning, it seems that data in column 47 is having mixed dtypes. Lets Analyse it and fix it.

In [5]:
```
# Print the column name at index 47
column_name = loan_data_df.columns[47]
print(column_name)
print(loan_data_df[column_name].to_list()[:20])
```

```
next_pymnt_d
[nan, nan, nan, nan, 'Jun-16', nan, nan, nan, nan, nan, nan, nan, nan, na
n, nan, nan, nan, nan, nan, nan]
```

So, as per above this issue is because the column contains data in string format and nan value. We can ignore it for now.

Data Cleaning steps

1. Drop columns with all null values
2. Drop columns where null value percentage is > 60
3. Drop columns with all 0 values
4. Convert term into int
5. Remove % from columns int_rate and revol_util
6. Removing duplicate rows from the dataframe
7. Correcting Data Types and Deriving New Columns
8. Removing the outliers
9. Consider only rows where loan_status != Current as ongoing loans can't be used to determine credit worthiness
10.

Drop columns with all null values if any

In [6]:
```
# Drop columns with all null values
print('Total columns Before drop:',loan_data_df.shape)
loan_data = loan_data_df.dropna(axis=1, how='all')
print('Total columns After drop:', loan_data.shape)
```

```
Total columns Before drop: (39717, 111)
Total columns After drop: (39717, 57)
```

Drop columns where null value percentage is > 60

In [7]:
```
# Drop columns where null value percentage is > 60

null_perc = loan_data.isnull().mean() * 100
columns_to_drop = null_perc[null_perc > 60].index
print('Columns with more than 60% null data :', columns_to_drop.values)
```

```
Columns with more than 60% null data : ['mths_since_last_delinq' 'mths_sin
ce_last_record' 'next_pymnt_d']
```

In [8]:
```
# Removing column with 60% or more null values as it will reduce the impa
```

```
loan_data = loan_data.loc[:,loan_data.isnull().sum()/loan_data.shape[0]*1
# Shape of the dataframe after removing columns
print(loan_data.shape)
```

```
(39717, 54)
```

Drop columns with all 0 values if any

In [9]:
```
# Drop columns with all 0 values
zero_cols = loan_data.columns[(loan_data==0).all(axis=0)]
print('Columns with all 0 values : ',zero_cols.values)
loan_data.drop(columns=zero_cols, inplace=True)
print('Total columns After drop:', loan_data.shape)
```

```
Columns with all 0 values :  ['acc_now_delinq' 'delinq_amnt']
Total columns After drop: (39717, 52)
```

In [10]:
```
# Convert term into int

loan_data.term = loan_data.term.apply(lambda x: int(x.replace(' months',
loan_data.head()
```

Out[10]:

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

5 rows × 52 columns

Remove % from columns int_rate and revol_util

In [11]:
```
# Remove % from columns int_rate and revol_util
print('Before removal')
loan_data.head()
```

```
Before removal
```

Out[11]:

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

5 rows × 52 columns

In [12]:
```
loan_data['int_rate'] = loan_data['int_rate'].str.rstrip('%').astype(floa
loan_data['revol_util'] = loan_data['revol_util'].str.rstrip('%').astype(
print('After removal')
loan_data.head()
```

```
After removal
```

Out[12]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rat
0	1077501	1296599	5000	5000	4975.0	36	10.6
1	1077430	1314167	2500	2500	2500.0	60	15.2
2	1077175	1313524	2400	2400	2400.0	36	15.9
3	1076863	1277178	10000	10000	10000.0	36	13.4
4	1075358	1311748	3000	3000	3000.0	60	12.6

5 rows × 52 columns

In [13]:
```
# Checking for missing values across the rows
print((loan_data.isnull().sum(axis=1)).max())
```

```
6
```

As the max number of missing value in row is very low compared to the count of columns(54 after removing irrelevant columns), we can move ahead with process as the impact is insignificant.

In [14]:
```
print(loan_data.columns)
```

```
Index(['id', 'member_id', 'loan_amnt', 'funded_amnt', 'funded_amnt_inv',
       'term', 'int_rate', 'installment', 'grade', 'sub_grade', 'emp_titl
e',
       'emp_length', 'home_ownership', 'annual_inc', 'verification_statu
s',
       'issue_d', 'loan_status', 'pymnt_plan', 'url', 'desc', 'purpose',
       'title', 'zip_code', 'addr_state', 'dti', 'delinq_2yrs',
       'earliest_cr_line', 'inq_last_6mths', 'open_acc', 'pub_rec',
       'revol_bal', 'revol_util', 'total_acc', 'initial_list_status',
       'out_prncp', 'out_prncp_inv', 'total_pymnt', 'total_pymnt_inv',
       'total_rec_prncp', 'total_rec_int', 'total_rec_late_fee', 'recoveri
es',
       'collection_recovery_fee', 'last_pymnt_d', 'last_pymnt_amnt',
       'last_credit_pull_d', 'collections_12_mths_ex_med', 'policy_code',
       'application_type', 'chargeoff_within_12_mths', 'pub_rec_bankruptci
es',
       'tax_liens'],
      dtype='object')
```

Removing the irrelevant columns

In [15]:

```python
# Removing irrelevant columns which are calculated after loan is approved
## The columns removed are customer behaviour variables and are calculate
loan_data=loan_data.drop(['delinq_2yrs','earliest_cr_line','inq_last_6mth
# Removing desc,emp_title,desc as they have no significance to the analys
loan_data=loan_data.drop(['title','emp_title','desc','url'],axis=1)
# Removing zip_code as it is a masked data and cannot be used as input fo
loan_data=loan_data.drop(['zip_code'],axis=1)
# Removing member_id as it is a duplicate index column and is not require
loan_data=loan_data.drop(['member_id'],axis=1)
# Removing funded_amnt_inv as it is a internal data and is calculated aft
loan_data=loan_data.drop(['funded_amnt_inv'],axis=1)
# Shape of the dataframe after removing columns
print(loan_data.shape)
```

```
(39717, 26)
```

Removed the above columns as they are customer behavior variables and are not available at time of decision and hence not useful for analysis.

Checking columns for irrelevant data which has no impact to analysis(having very few unqiue values)

In [16]:

```python
# Checking columns for irrelevant data which has no impact to analysis(ha
print(loan_data.nunique().sort_values(ascending=True))
```

```
tax_liens                         1
chargeoff_within_12_mths          1
policy_code                       1
collections_12_mths_ex_med        1
initial_list_status               1
pymnt_plan                        1
term                              2
verification_status               3
pub_rec_bankruptcies              3
loan_status                       3
home_ownership                    5
grade                             7
emp_length                       11
purpose                          14
sub_grade                        35
addr_state                       50
issue_d                          55
int_rate                        371
loan_amnt                       885
funded_amnt                    1041
revol_util                     1089
dti                            2868
annual_inc                     5318
installment                   15383
revol_bal                     21711
id                            39717
dtype: int64
```

As there are many columns with 1 unique value and null values, we have removed them as they are not relevant to the analysis.

In [17]:
```
# Removing irrelevant columns which contain 1 unique value
loan_data = loan_data.loc[:,loan_data.nunique()>1]
# Shape of the dataframe after removing columns
print(loan_data.shape)
```

```
(39717, 20)
```

Removing and fixing the null values

In [18]:
```
# Checking for missing values across the dataframe
print(loan_data.isnull().sum().sort_values(ascending=False))
```

```
emp_length              1075
pub_rec_bankruptcies     697
revol_util                50
verification_status        0
revol_bal                  0
dti                        0
addr_state                 0
purpose                    0
loan_status                0
issue_d                    0
id                         0
loan_amnt                  0
home_ownership             0
sub_grade                  0
grade                      0
installment                0
int_rate                   0
term                       0
funded_amnt                0
annual_inc                 0
dtype: int64
```

emp_length 1075 pub_rec_bankruptcies 697 The above columns has null values which can be removed or fixed depending on the relevance of the column to objective of the analysis.

In [19]:
```
# Checking values in emp_length columns for feasibility of inserting null
print(loan_data.emp_length.value_counts())
```

```
emp_length
10+ years    8879
< 1 year     4583
2 years      4388
3 years      4095
4 years      3436
5 years      3282
1 year       3240
6 years      2229
7 years      1773
8 years      1479
9 years      1258
Name: count, dtype: int64
```

In [20]:
```
# Checking values in pub_rec_bankruptcies columns for feasibility of inse
print(loan_data.pub_rec_bankruptcies.value_counts())
```

```
pub_rec_bankruptcies
0.0    37339
1.0     1674
2.0        7
Name: count, dtype: int64
```

In [21]:
```
# Removing null values in emp_title and emp_length columns
loan_data = loan_data.dropna(subset=['emp_length'])
# Shape of the dataframe after removing columns
print(loan_data.shape)
```

```
(38642, 20)
```

In [22]:
```
# Inserting 0 for null values in pub_rec_bankruptcies column
loan_data.fillna({'pub_rec_bankruptcies': 0},inplace=True)
```

In [23]:
```
# Checking for missing values across the dataframe
print(loan_data.isnull().sum())
```

```
id                       0
loan_amnt                0
funded_amnt              0
term                     0
int_rate                 0
installment              0
grade                    0
sub_grade                0
emp_length               0
home_ownership           0
annual_inc               0
verification_status      0
issue_d                  0
loan_status              0
purpose                  0
addr_state               0
dti                      0
revol_bal                0
revol_util              47
pub_rec_bankruptcies     0
dtype: int64
```

Removing duplicate rows from the dataframe

In [24]:
```
# Removing duplicate rows in the dataframe
loan_data = loan_data.drop_duplicates()
# Shape of the dataframe after removing duplicate rows
print(loan_data.shape)

# No duplicate rows found in the dataframe
```

```
(38642, 20)
```

Correcting Data Types and Deriving New Columns

In [25]:
```
# Checking information about the dataframe
print(loan_data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 38642 entries, 0 to 39716
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    38642 non-null  int64  
 1   loan_amnt             38642 non-null  int64  
 2   funded_amnt           38642 non-null  int64  
 3   term                  38642 non-null  int64  
 4   int_rate              38642 non-null  float64
 5   installment           38642 non-null  float64
 6   grade                 38642 non-null  object 
 7   sub_grade             38642 non-null  object 
 8   emp_length            38642 non-null  object 
 9   home_ownership        38642 non-null  object 
 10  annual_inc            38642 non-null  float64
 11  verification_status   38642 non-null  object 
 12  issue_d               38642 non-null  object 
 13  loan_status           38642 non-null  object 
 14  purpose               38642 non-null  object 
 15  addr_state            38642 non-null  object 
 16  dti                   38642 non-null  float64
 17  revol_bal             38642 non-null  int64  
 18  revol_util            38595 non-null  float64
 19  pub_rec_bankruptcies  38642 non-null  float64
dtypes: float64(6), int64(5), object(9)
memory usage: 6.2+ MB
None
```

In [26]:
```
# Correcting data type and format for columns in the dataframe
## Derving more columns with the conversion of data type
loan_data.term = loan_data.term.apply(lambda x: int(str(x).replace(' mont
loan_data.grade=loan_data.grade.astype('category')
loan_data.sub_grade=loan_data.sub_grade.astype('category')
loan_data.emp_length=loan_data.emp_length.apply(lambda x: x.replace('year
loan_data.home_ownership=loan_data.home_ownership.astype('category')
loan_data.verification_status=loan_data.verification_status.astype('categ
loan_data.issue_d=pd.to_datetime(loan_data.issue_d,format='%b-%y')
loan_data['issue_year']=pd.to_datetime(loan_data.issue_d,format='%b-%y').
loan_data['issue_month']=pd.to_datetime(loan_data.issue_d,format='%b-%y')
loan_data.purpose=loan_data.purpose.astype('category')
loan_data.addr_state=loan_data.addr_state.astype('category')
```

In [27]:
```
# Setting decimal point limit for all data
for x in loan_data.columns:
    if(loan_data[x].dtype=='float64'):
      loan_data[x]=loan_data[x].round(2)

loan_data.head()
```

Out[27]:

	id	loan_amnt	funded_amnt	term	int_rate	installment	grade	sub_grad
0	1077501	5000	5000	36	10.65	162.87	B	E
1	1077430	2500	2500	60	15.27	59.83	C	C
2	1077175	2400	2400	36	15.96	84.33	C	C
3	1076863	10000	10000	36	13.49	339.31	C	C
4	1075358	3000	3000	60	12.69	67.79	B	E

5 rows × 22 columns

In [ ]:

In [28]:

```
# Consider only rows where loan_status != Current as ongoing loans can't

loan_data = loan_data[loan_data['loan_status'] != 'Current']
loan_data['loan_status'].value_counts()
```

Out[28]:

```
loan_status
Fully Paid     32145
Charged Off     5399
Name: count, dtype: int64
```

As the data has been cleaned, fixed and filtered as per requirement, we can select columns required for analysis and move ahead with the analysis.

In [29]:

```
# selecting columns based on domain knowledge
## Id, Loan Amount, Term of loan, Interest Rate, Grade, Sub Grade, Emp Le
loan_data = loan_data[['id','loan_amnt','term','int_rate','grade','sub_gr
# Shape of the dataframe after removing columns
loan_data.shape
```

Out[29]: (37544, 17)

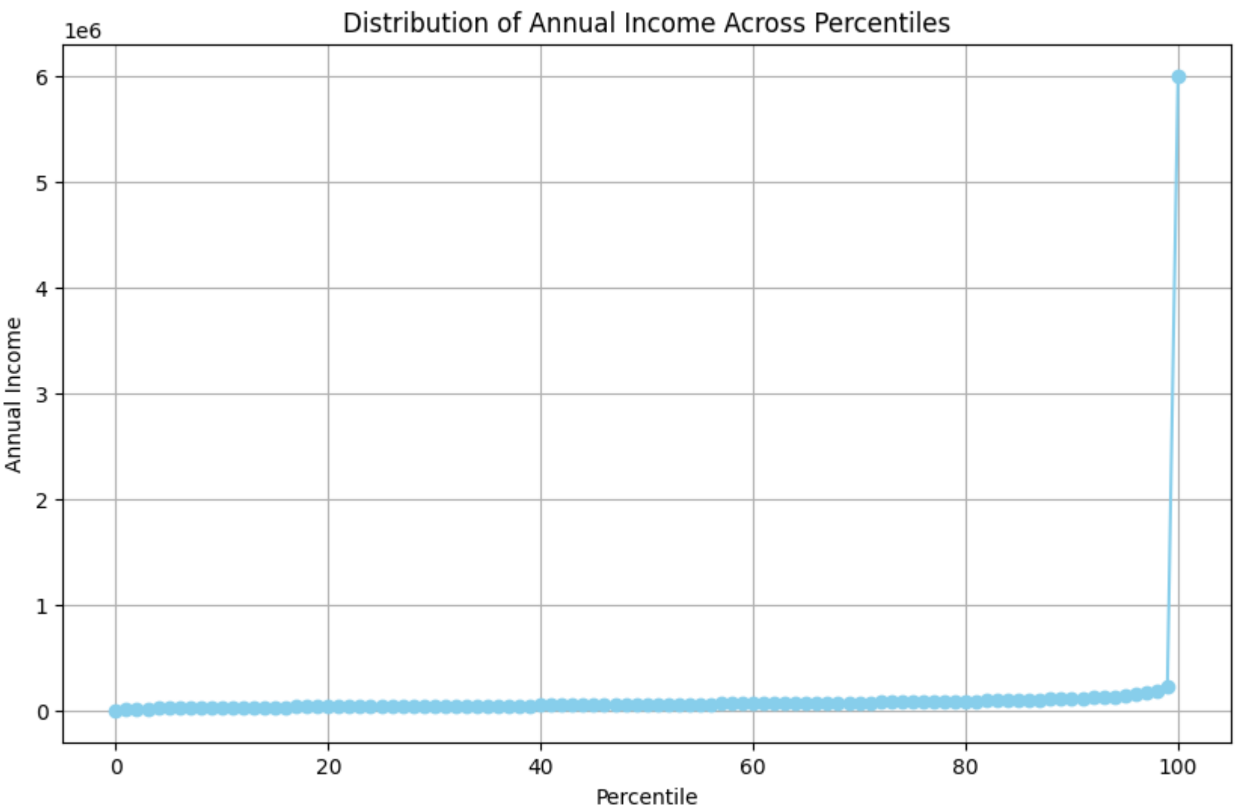
In [30]:

```
# Divding the column as per categorical and numerical
cat_cols = ['term','grade','sub_grade','emp_length','home_ownership','ver
cont_cols=['loan_amnt','int_rate','annual_inc','dti','pub_rec_bankruptcie
id_cols=['id']
result_cols=['loan_status']
```

In [ ]:

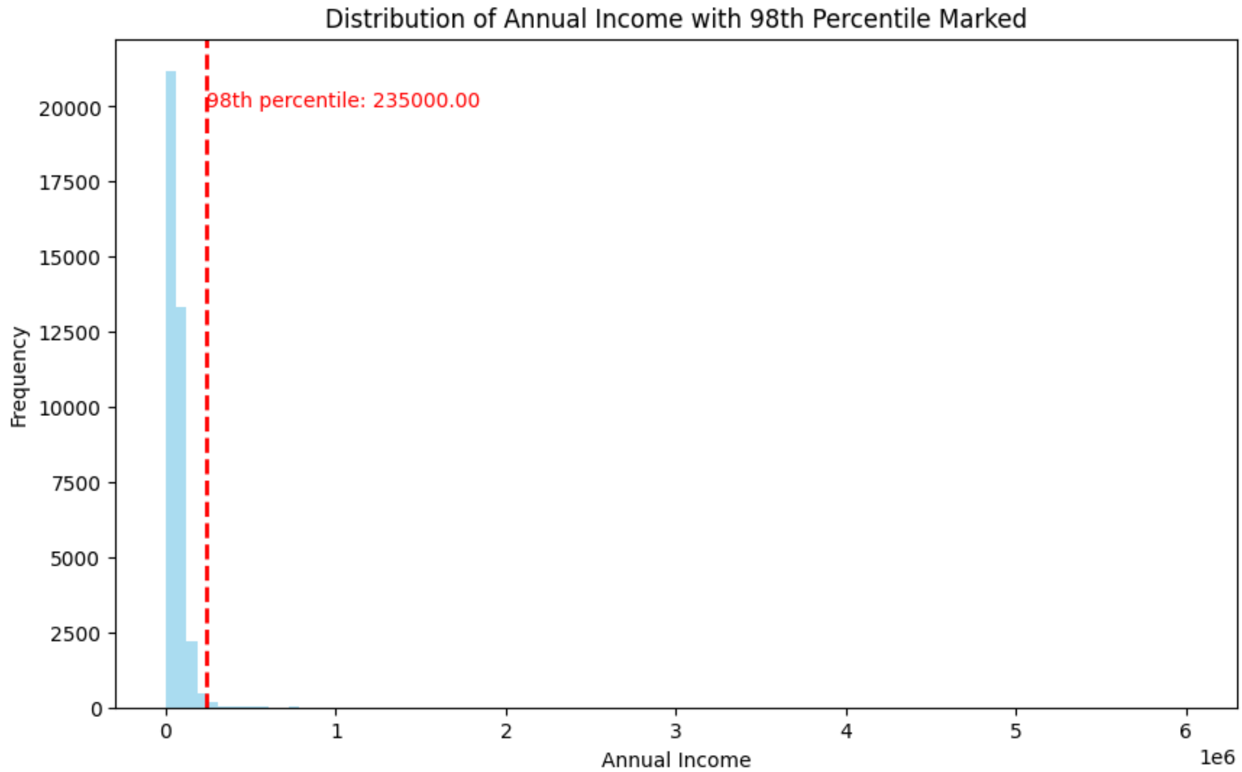
Data Analysis After Cleanup

In [31]:
```
# Identify potential outliers
# Calculate the percentiles
percentiles = np.percentile(loan_data['annual_inc'], np.arange(0, 101, 1)

# Plotting the distribution of annual_inc in percentiles
plt.figure(figsize=(10, 6))
plt.plot(np.arange(0, 101, 1), percentiles, marker='o', color='skyblue')
plt.xlabel('Percentile')
plt.ylabel('Annual Income')
plt.title('Distribution of Annual Income Across Percentiles')
plt.grid(True)
plt.show()
```



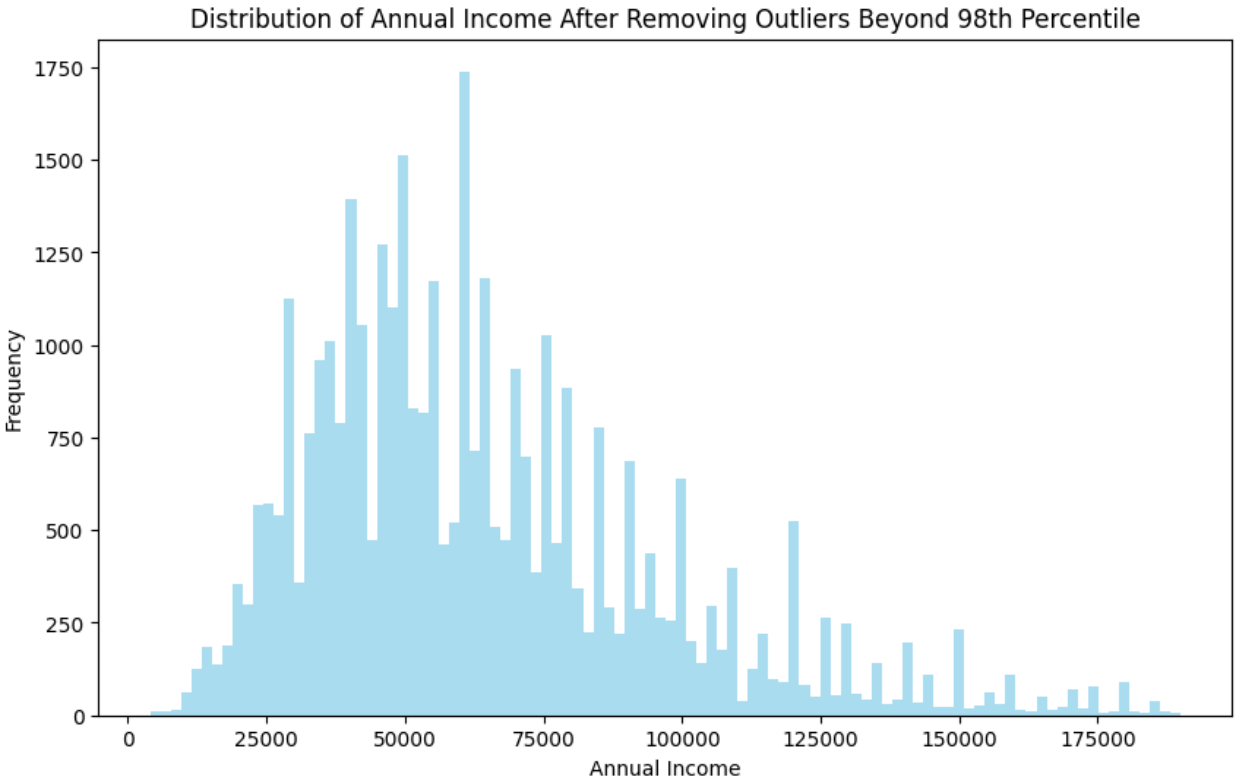
Outlier seems to be after 98th percentile

In [32]:
```
# Calculate the 98th percentile
percentile_98 = loan_data['annual_inc'].quantile(0.99)

# Plotting the distribution of annual_inc
plt.figure(figsize=(10, 6))
plt.hist(loan_data['annual_inc'], bins=100, color='skyblue', alpha=0.7)
plt.axvline(percentile_98, color='red', linestyle='dashed', linewidth=2)
plt.text(percentile_98, plt.ylim()[1]*0.9, f'98th percentile: {percentile
plt.xlabel('Annual Income')
plt.ylabel('Frequency')
plt.title('Distribution of Annual Income with 98th Percentile Marked')
plt.show()
```

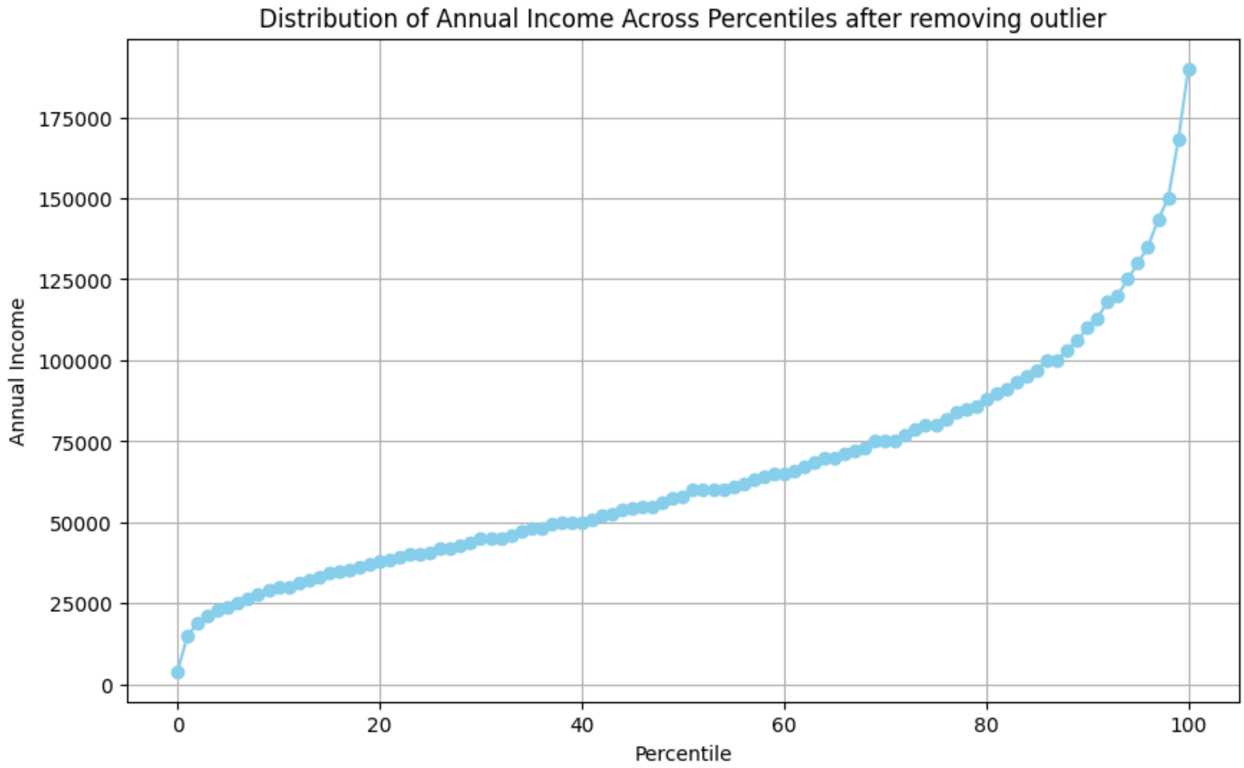


Remove outliers beyond the 98th percentile:

In [33]:
```
# Calculate the 99th percentile
percentile_98 = loan_data['annual_inc'].quantile(0.98)

# Remove outliers beyond the 99th percentile
loan_data = loan_data[loan_data['annual_inc'] <= percentile_98]

# Plotting the distribution of annual_inc after removing outliers
plt.figure(figsize=(10, 6))
plt.hist(loan_data['annual_inc'], bins=100, color='skyblue', alpha=0.7)
plt.xlabel('Annual Income')
plt.ylabel('Frequency')
plt.title('Distribution of Annual Income After Removing Outliers Beyond 9
plt.show()
```



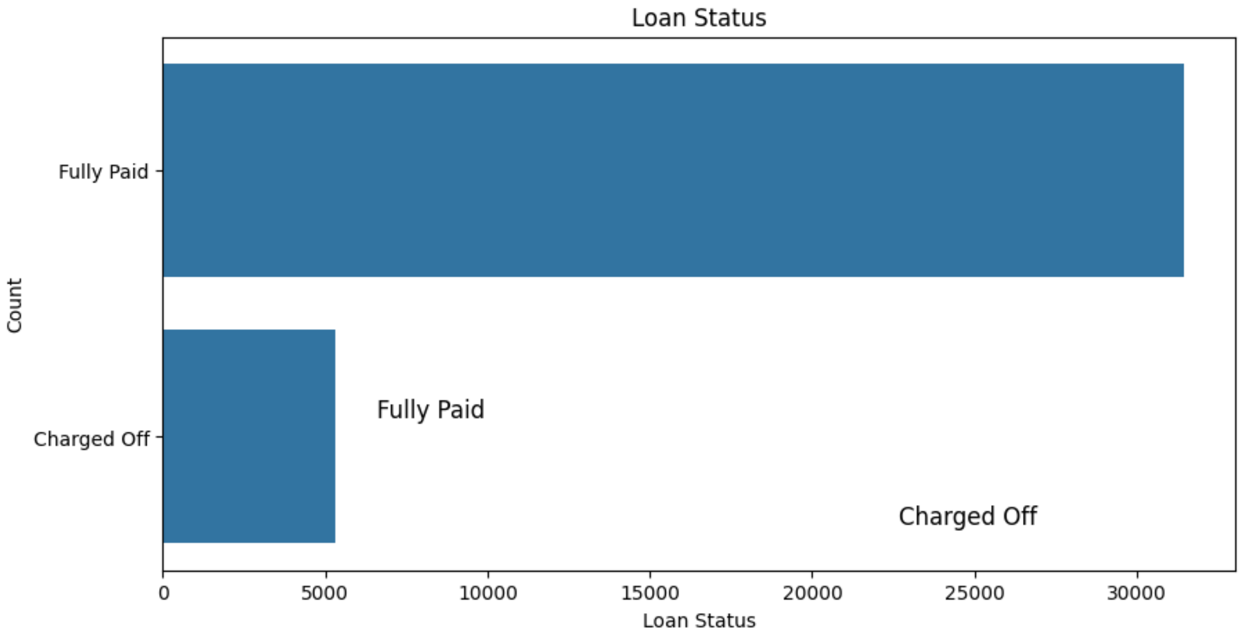
In [34]:
```
# Calculate the percentiles
percentiles = np.percentile(loan_data['annual_inc'], np.arange(0, 101, 1)

# Plotting the distribution of annual_inc in percentiles
plt.figure(figsize=(10, 6))
plt.plot(np.arange(0, 101, 1), percentiles, marker='o', color='skyblue')
plt.xlabel('Percentile')
plt.ylabel('Annual Income')
plt.title('Distribution of Annual Income Across Percentiles after removin
plt.grid(True)
plt.show()
```



Univariate Analysis

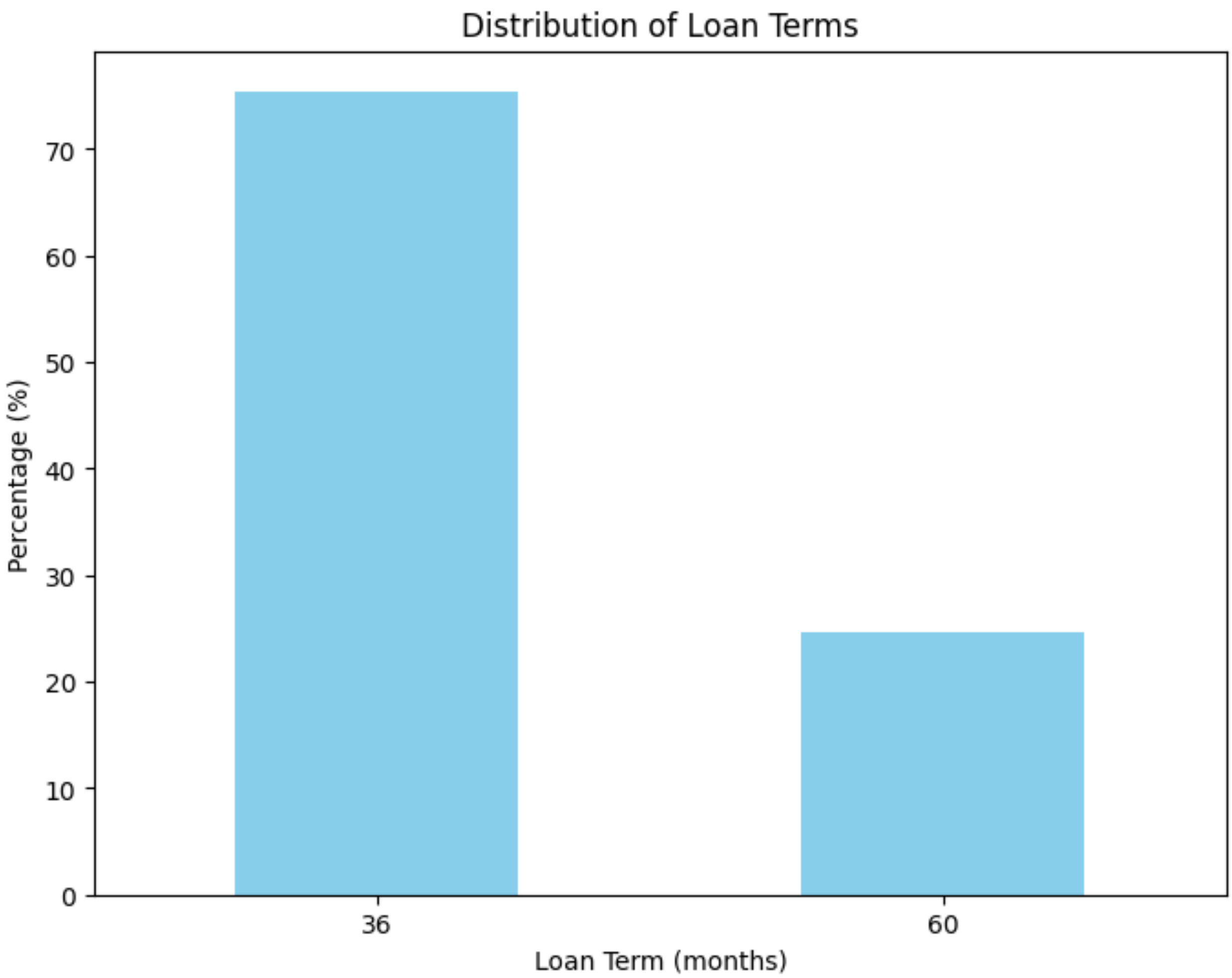
In [35]:
```
# Loan status
print(loan_data.loan_status.value_counts()*100/loan_data.loan_status.coun
# 0=Fully Paid, 1=Charged Off
plt.figure(figsize=(10,5))
ax=sns.countplot(loan_data.loan_status)
ax.annotate('Fully Paid',xy=(0.25,0.3),xycoords='axes fraction',horizonta
ax.annotate('Charged Off',xy=(0.75,0.1),xycoords='axes fraction',horizont
ax.set_title('Loan Status')
ax.set_xlabel('Loan Status')
ax.set_ylabel('Count')
plt.show()
```

```
loan_status
Fully Paid     85.532967
Charged Off    14.467033
Name: count, dtype: float64
```



Inference: Defaulted loan are low in numbers compared to Fully Paid.

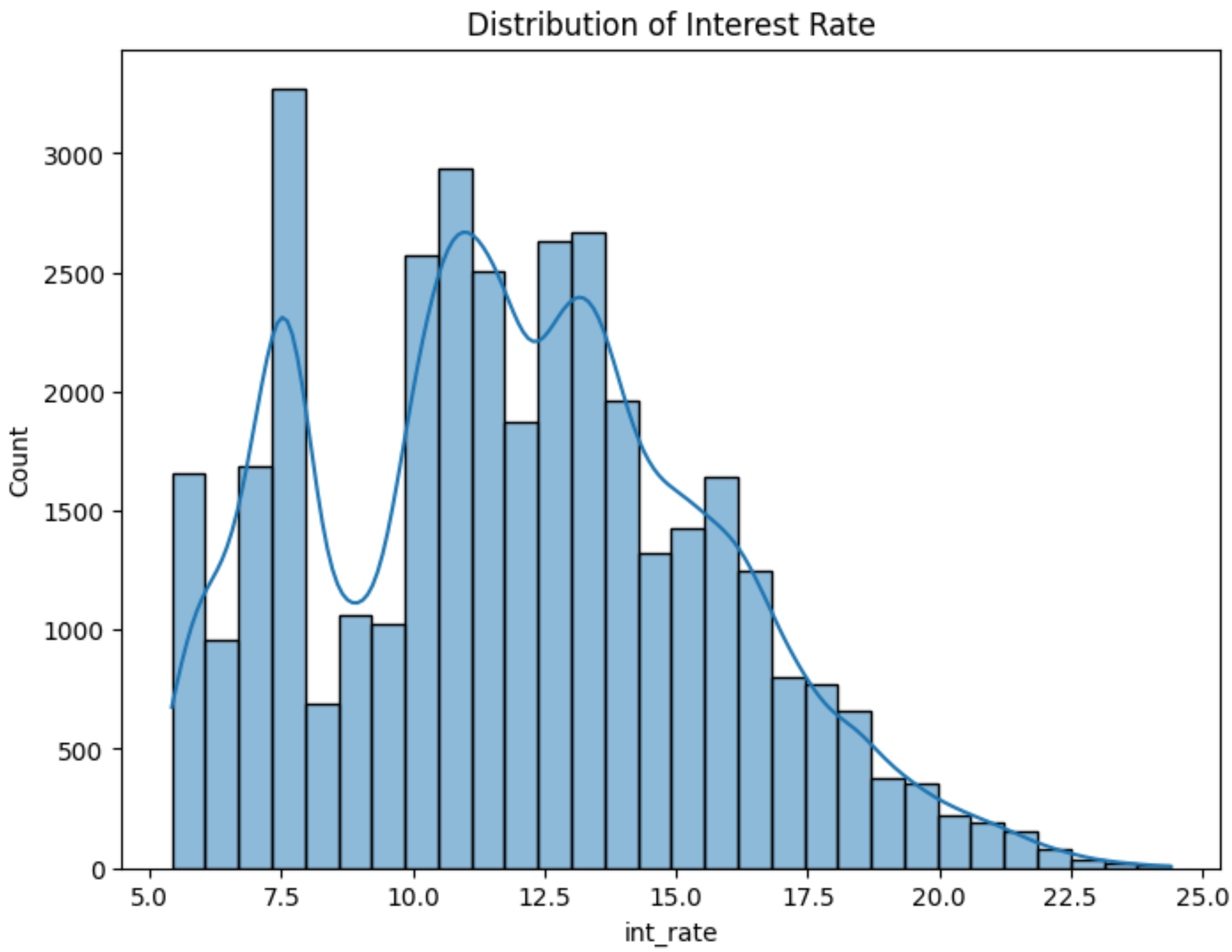
In [36]:
```
# Term of loan
term_counts = loan_data.term.value_counts(normalize=True) * 100

# Plotting
plt.figure(figsize=(8, 6))
term_counts.plot(kind='bar', color='skyblue')
plt.title('Distribution of Loan Terms')
plt.xlabel('Loan Term (months)')
plt.ylabel('Percentage (%)')
plt.xticks(rotation=0)
plt.show()
```



Inference: More than half of the loan taken has term of 36 months compared to 60 months.

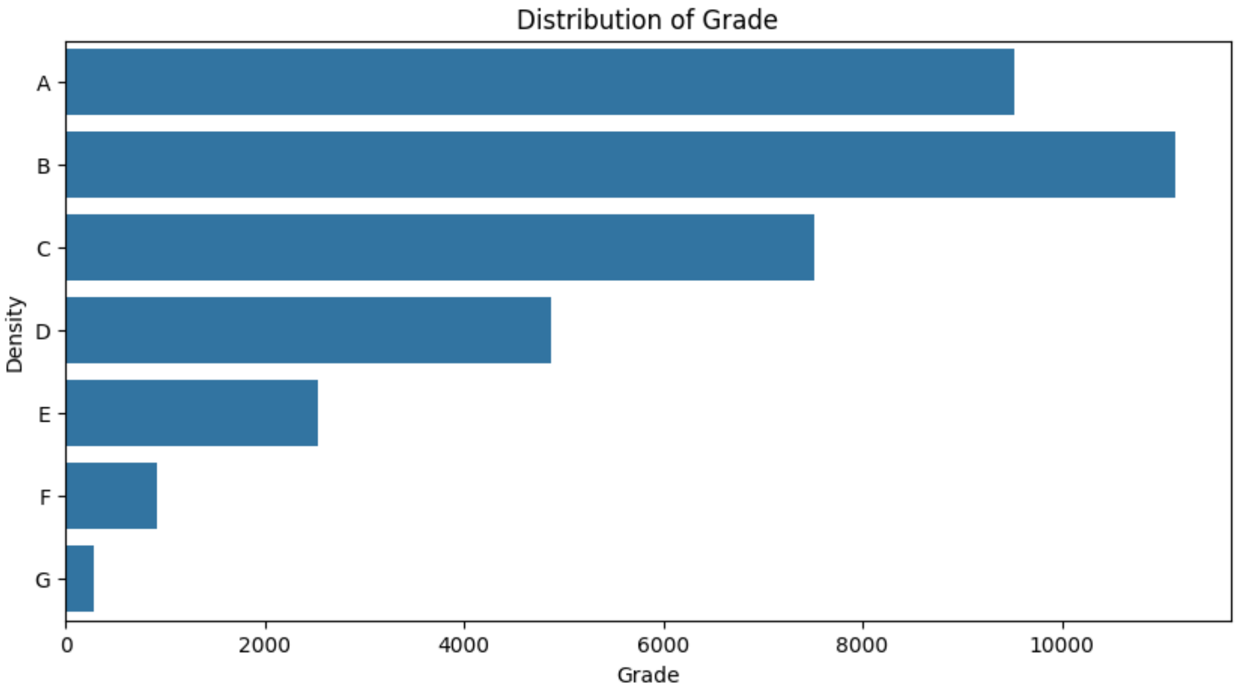
In [37]:
```
# Interest Rate Distribution (`int_rate`)
int_rate_stats = loan_data['int_rate'].describe()
print("\nInterest Rate Statistics:")
print(int_rate_stats)

plt.figure(figsize=(8, 6))
sns.histplot(loan_data['int_rate'], bins=30, kde=True)
plt.title('Distribution of Interest Rate')
plt.show()
```

```
Interest Rate Statistics:
count    36794.000000
mean        11.940170
std          3.672825
min          5.420000
25%          8.940000
50%         11.830000
75%         14.350000
max         24.400000
Name: int_rate, dtype: float64
```



Inference: The interest rate is more crowded around 5-10 and 10-15 with a drop near 10.

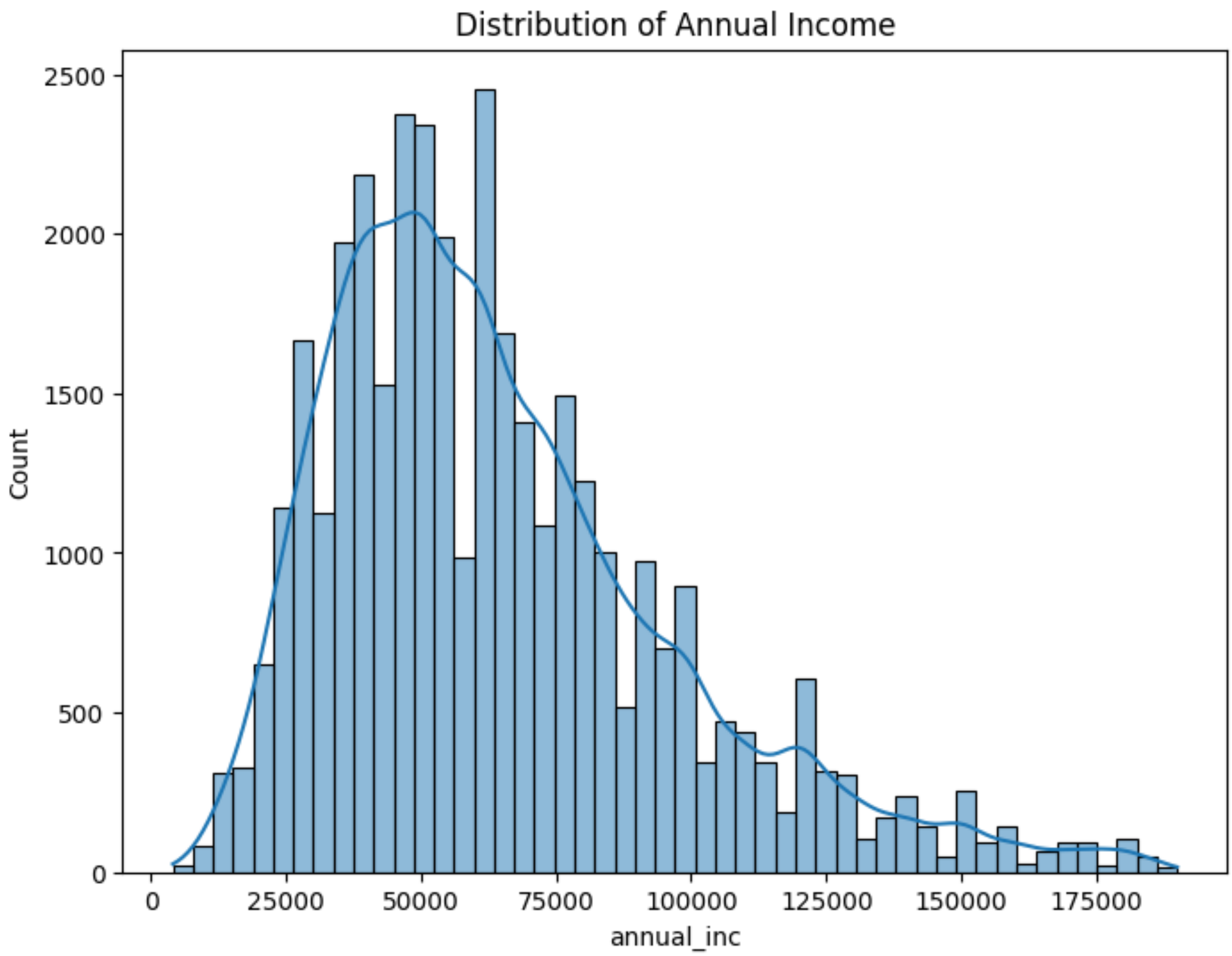
In [38]:
```
# Distribution of Greade
plt.figure(figsize=(10,5))
sns.countplot(loan_data.grade)
plt.xlabel('Grade')
plt.ylabel('Density')
plt.title('Distribution of Grade')
plt.show()
```



Inference: A large amount of loans are with grade 'A' and 'B' commpared to rest showing most loans are high grade loans.

In [39]:
```
# Distribution of Annual Income (`annual_inc`)
annual_inc_stats = loan_data['annual_inc'].describe()
print("\nAnnual Income Statistics:")
print(annual_inc_stats)

plt.figure(figsize=(8, 6))
sns.histplot(loan_data['annual_inc'], bins=50, kde=True)
plt.title('Distribution of Annual Income')
plt.show()
```

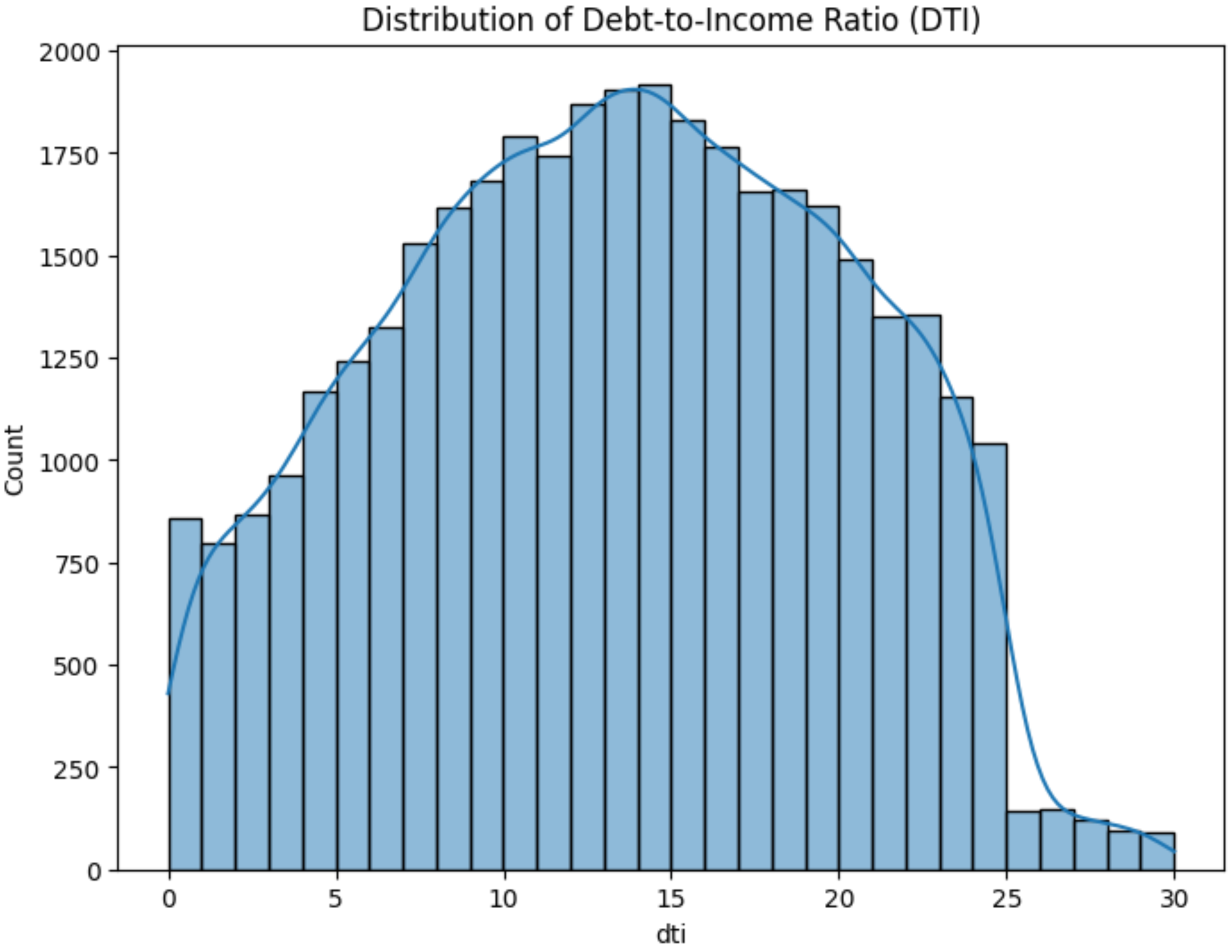
```
Annual Income Statistics:
count     36794.000000
mean      64483.097778
std       32321.576824
min        4000.000000
25%       40500.000000
50%       58000.000000
75%       80000.000000
max      189996.000000
Name: annual_inc, dtype: float64
```



Inference: Annual Income shows left skewed normal distribution thus we can say that the majority of burrowers have very low annual income compared to rest.

In [40]:
```
# Distribution of Debt-to-Income Ratio (`dti`)

dti_stats = loan_data['dti'].describe()
print("\nDebt-to-Income Ratio Statistics:")
print(dti_stats)

plt.figure(figsize=(8, 6))
sns.histplot(loan_data['dti'], bins=30, kde=True)
plt.title('Distribution of Debt-to-Income Ratio (DTI)')
plt.show()
```

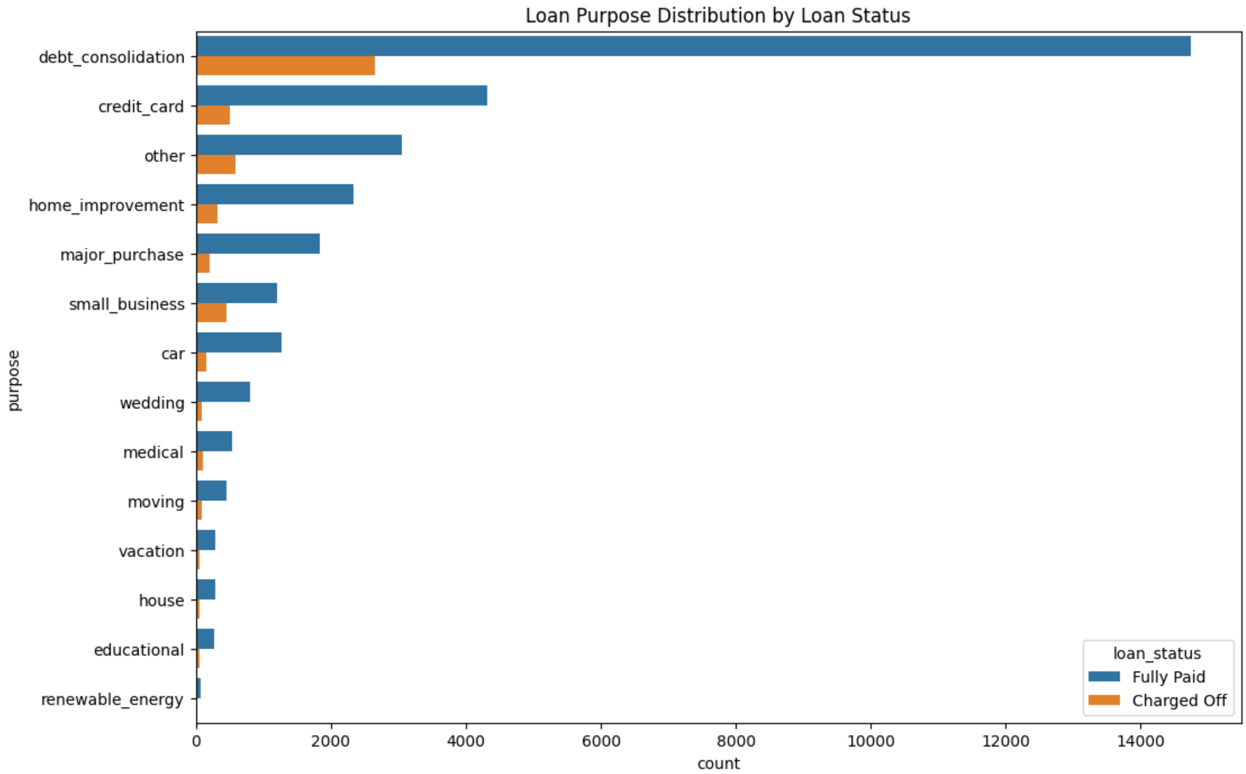
```
Debt-to-Income Ratio Statistics:
count    36794.000000
mean        13.378698
std          6.644522
min          0.000000
25%          8.280000
50%         13.490000
75%         18.620000
max         29.990000
Name: dti, dtype: float64
```



Inference: Majority of the borrowers have very large debt compared to the income registerd, concentrated in the 10-15 DTI ratio.

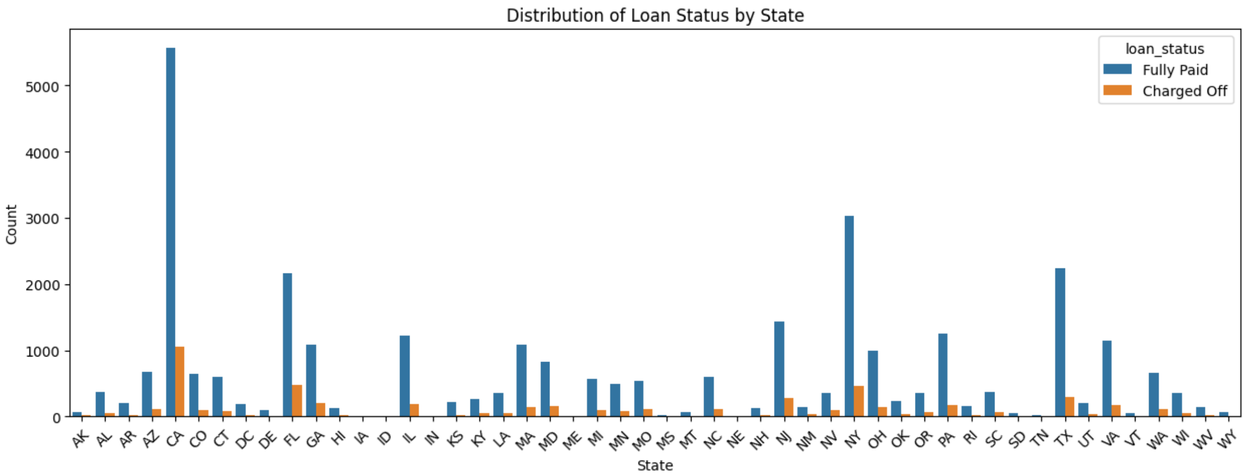
In [41]:
```
# Purpose of the Loan (`purpose`)
purpose_counts = loan_data['purpose'].value_counts()
print("\nLoan Purpose Counts:")
print(purpose_counts)

plt.figure(figsize=(12, 8))
sns.countplot(y='purpose', data=loan_data, hue='loan_status', order=loan_
plt.title('Loan Purpose Distribution by Loan Status')
plt.show()
```

```
Loan Purpose Counts:
purpose
debt_consolidation    17405
credit_card            4815
other                  3638
home_improvement       2651
major_purchase         2037
small_business         1656
car                    1429
wedding                 897
medical                 636
moving                  543
vacation                345
house                   340
educational             313
renewable_energy         89
Name: count, dtype: int64
```



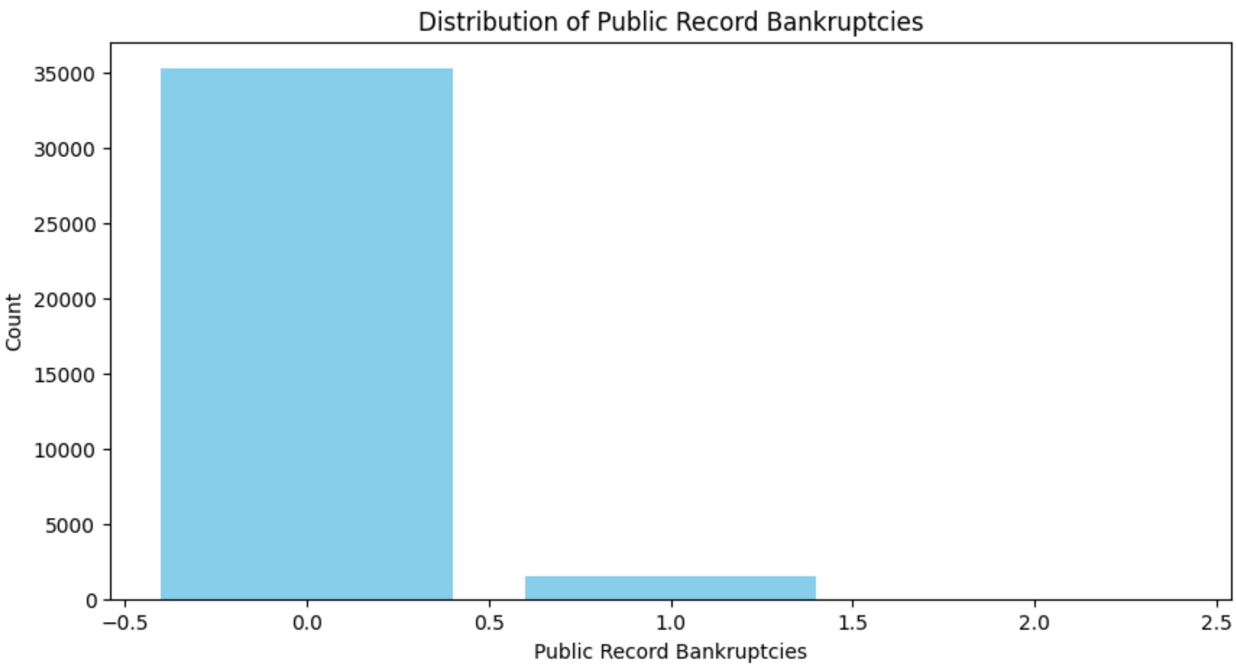
Inference: A large percentage of loans are taken for debt consolidation followed by credit card.

In [42]:
```
# Distribution of addr_state
plt.figure(figsize=(15, 5))
sns.countplot(x='addr_state', hue='loan_status', data=loan_data)
plt.xlabel('State')
plt.ylabel('Count')
plt.title('Distribution of Loan Status by State')
plt.xticks(rotation=45)  # Rotate x labels if there are many states
plt.show()
```



Inference: Majority of the borrowers are from the large urban cities like california, new york, texas, florida etc.

In [43]:
```
# Distribution of pub_rec_bankruptcies
# Count the occurrences of each unique value in pub_rec_bankruptcies
pub_rec_bankruptcies_counts = loan_data['pub_rec_bankruptcies'].value_cou

# Plotting without seaborn
plt.figure(figsize=(10, 5))
plt.bar(pub_rec_bankruptcies_counts.index, pub_rec_bankruptcies_counts.va
plt.xlabel('Public Record Bankruptcies')
plt.ylabel('Count')
plt.title('Distribution of Public Record Bankruptcies', fontsize=12)
plt.show()
```



Inference: Majority of the borrowers have no record of Public Recorded Bankruptcy.

In [44]:
```
# Distribution of issue_month
```

```
# Count the occurrences of each unique value in issue_month grouped by lo
issue_month_counts = loan_data.groupby(['issue_month', 'loan_status']).si

# Plotting without seaborn
plt.figure(figsize=(10, 5))
issue_month_counts.plot(kind='bar', stacked=True, color=['skyblue', 'salm
plt.xlabel('Issue Month')
plt.ylabel('Count')
plt.title('Distribution of Loan Issue Month by Loan Status', fontsize=12)
plt.xticks(rotation=45)
plt.legend(title='Loan Status')
plt.show()
```

In [ ]:

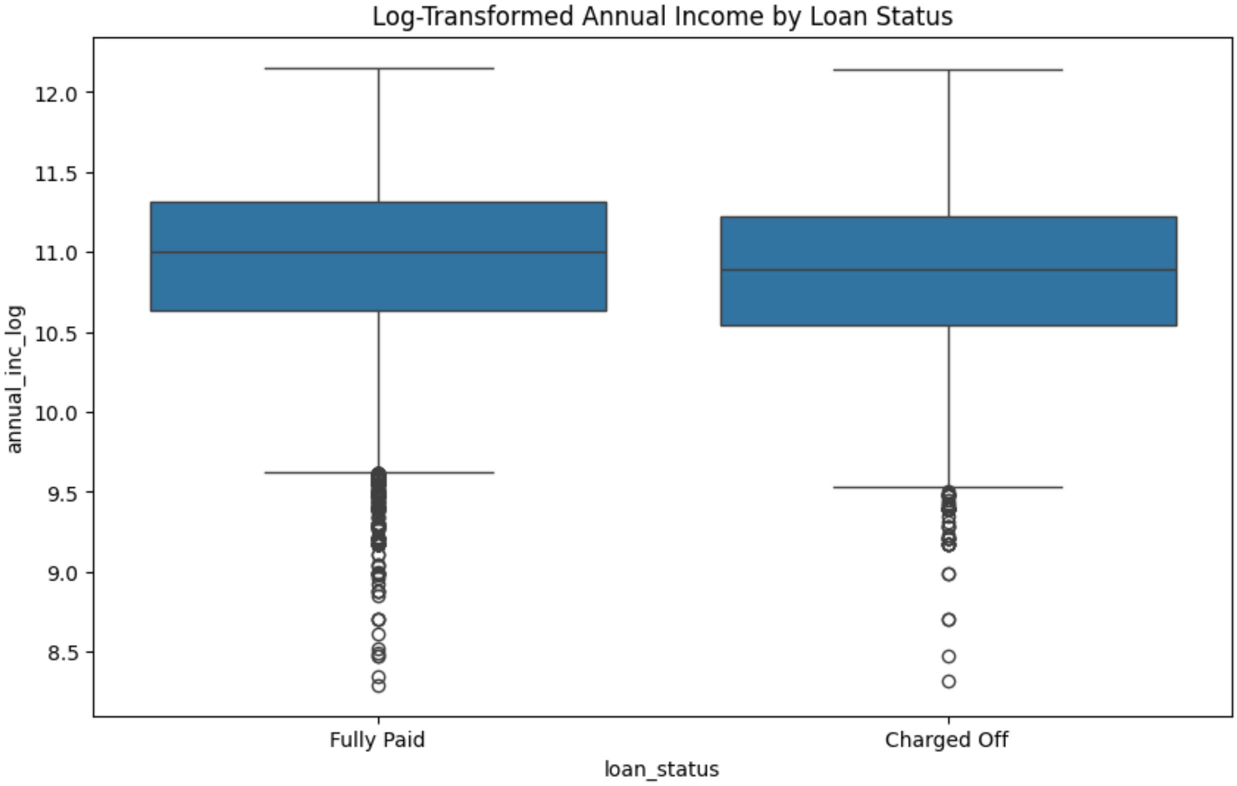
Inference: Majority of the loans are given in last quarter of the year.

Bivariate Analysis

In [45]:

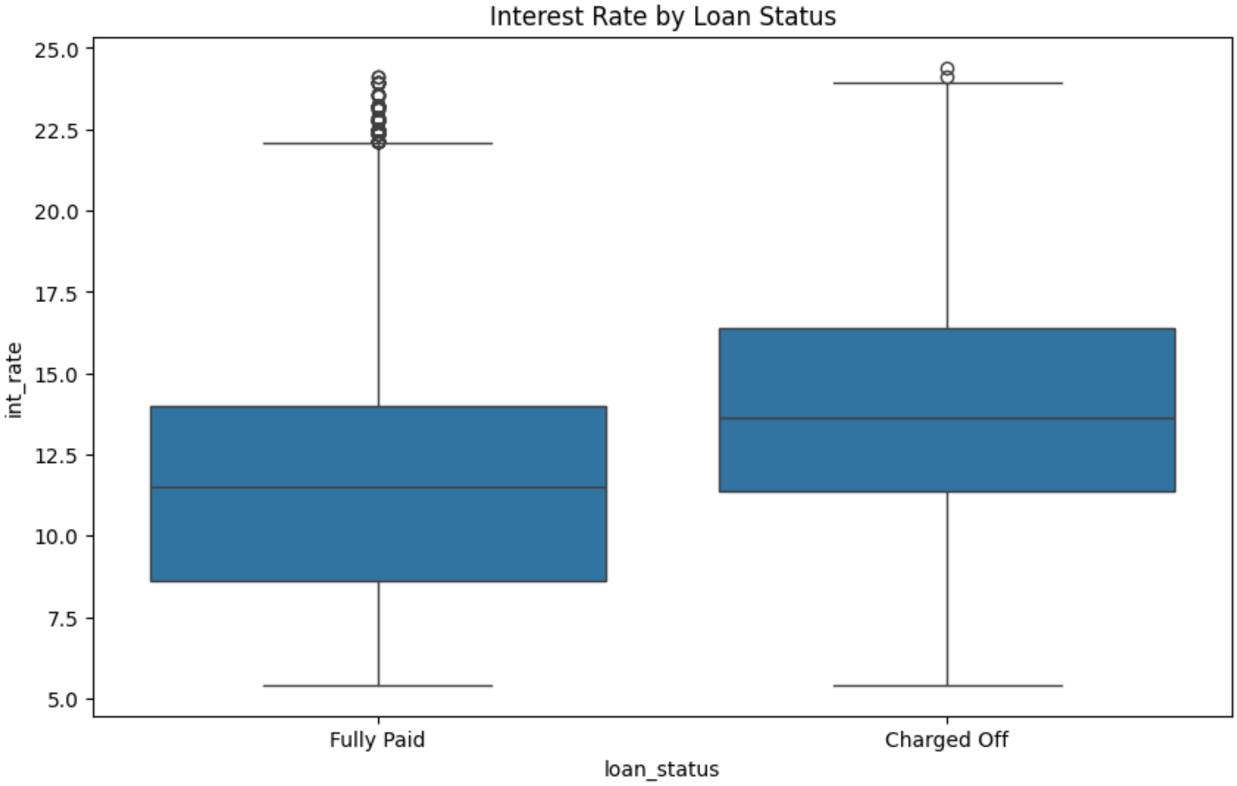
```
# Annual Income (log-transformed) vs Loan Status
loan_data['annual_inc_log'] = np.log1p(loan_data['annual_inc'])
annual_inc_log_stats = loan_data['annual_inc_log'].describe()
print("\nLog-Transformed Annual Income Statistics:")
print(annual_inc_log_stats)

plt.figure(figsize=(10, 6))
sns.boxplot(x='loan_status', y='annual_inc_log', data=loan_data)
plt.title('Log-Transformed Annual Income by Loan Status')
plt.show()
```

```
Log-Transformed Annual Income Statistics:
count    36794.000000
mean        10.950507
std          0.509289
min          8.294300
25%         10.609082
50%         10.968216
75%         11.289794
max         12.154764
Name: annual_inc_log, dtype: float64
```



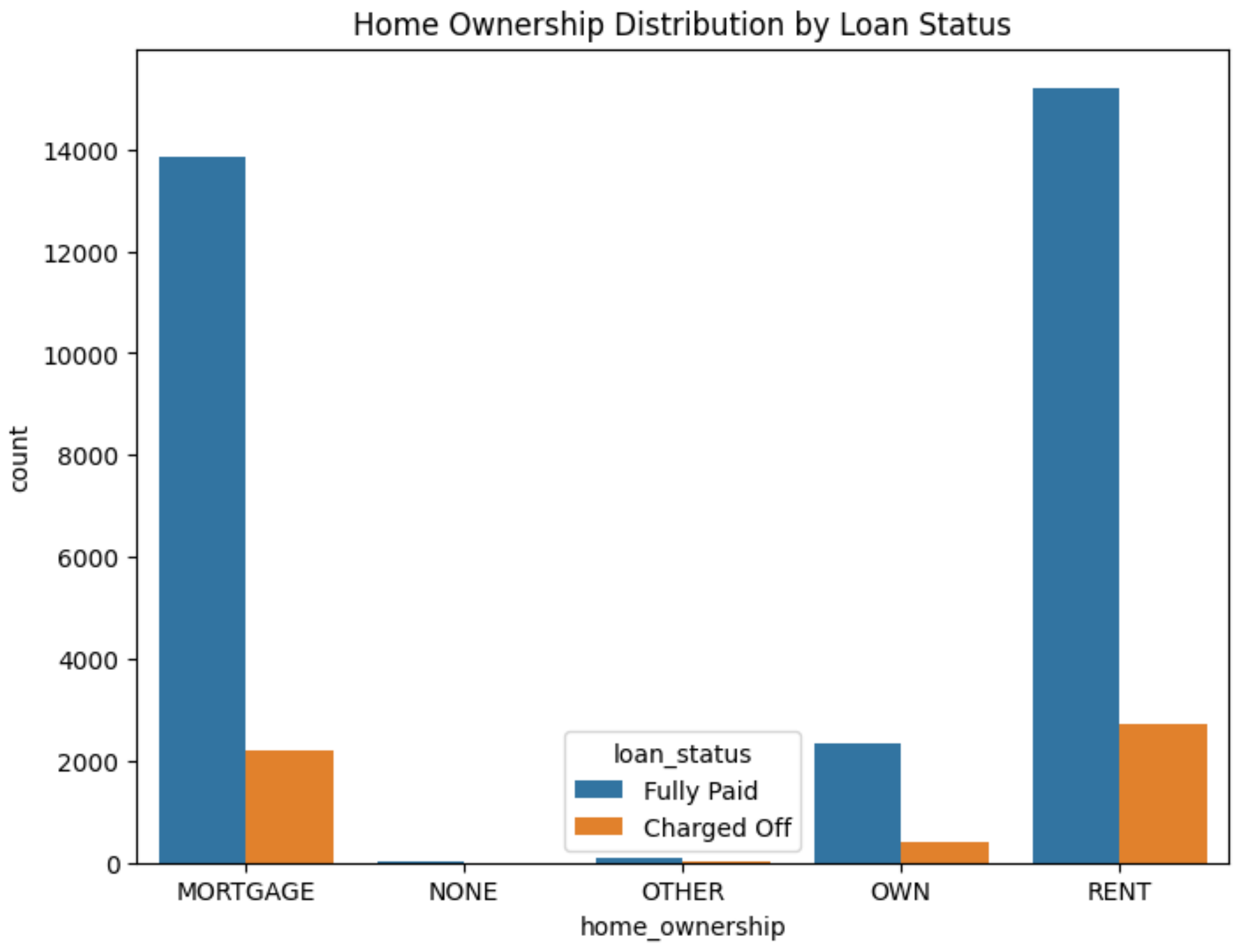
Inference: The mean and 25% percentile are same for both but we see larger 75% percentile in the defaulted loan which indicate large amount of loan has higher chance of defaulting.

In [46]:
```
# Interest Rate vs Loan Status
plt.figure(figsize=(10, 6))
sns.boxplot(x='loan_status', y='int_rate', data=loan_data)
plt.title('Interest Rate by Loan Status')
plt.show()
```



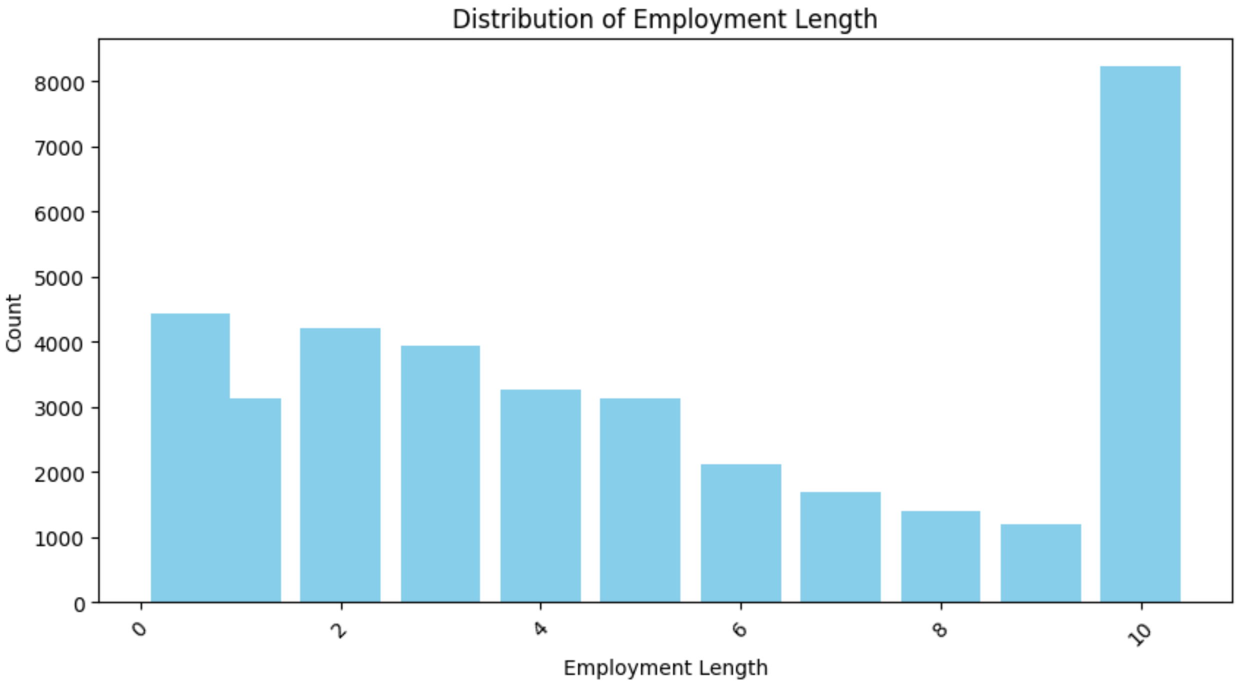
In [47]:
```
# Home Ownership Distribution by Loan Status (`home_ownership`)
home_ownership_counts = loan_data['home_ownership'].value_counts()
print("\nHome Ownership Counts:")
print(home_ownership_counts)

plt.figure(figsize=(8, 6))
sns.countplot(x='home_ownership', data=loan_data, hue='loan_status')
plt.title('Home Ownership Distribution by Loan Status')
plt.show()
```

```
Home Ownership Counts:
home_ownership
RENT        17923
MORTGAGE    16052
OWN          2721
OTHER          95
NONE            3
Name: count, dtype: int64
```



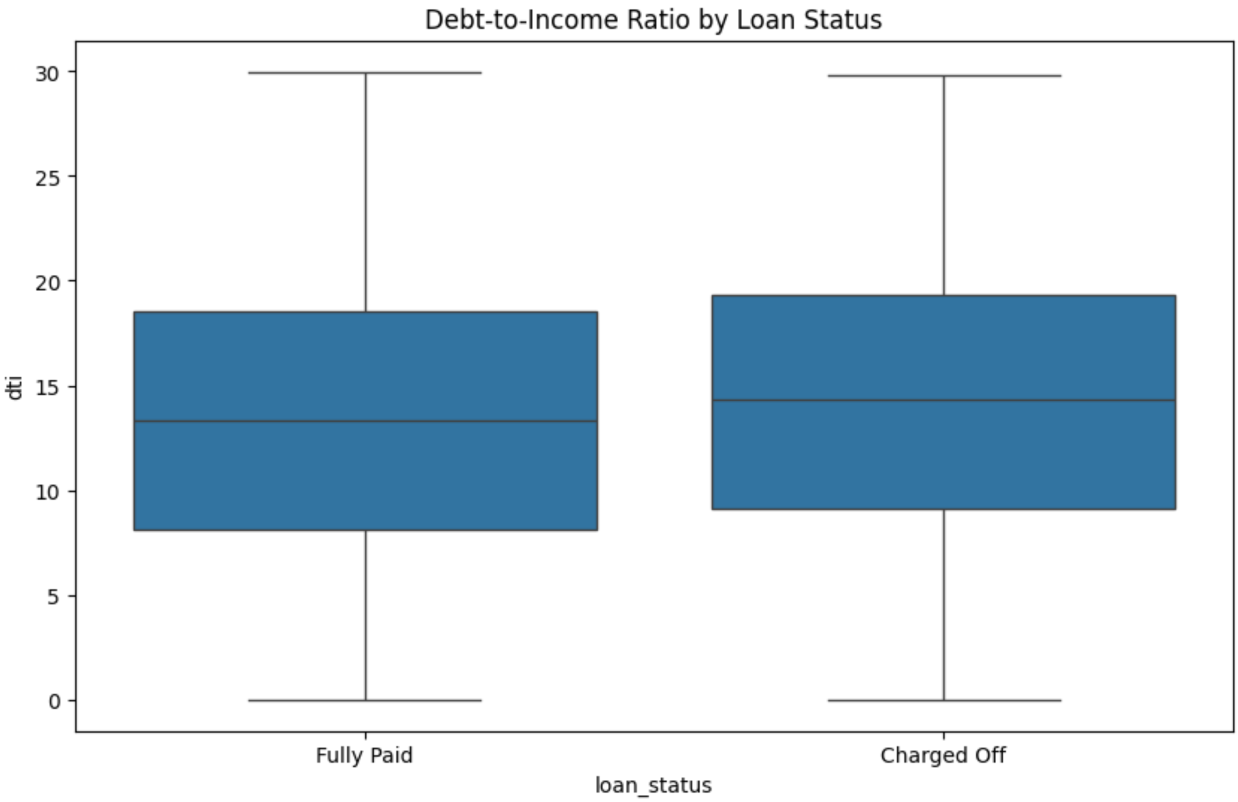
Inference: Majority of borrowsers don't posses property and are on mortage or rent.

In [48]:
```
# Employment Length Distribution by Loan Status (`emp_length`)
# Count the occurrences of each employment length
emp_length_counts = loan_data['emp_length'].value_counts().sort_index()

# Plotting
plt.figure(figsize=(10, 5))
plt.bar(emp_length_counts.index, emp_length_counts.values, color='skyblue
plt.xlabel('Employment Length')
plt.ylabel('Count')
plt.title('Distribution of Employment Length', fontsize=12)
plt.xticks(rotation=45)  # Rotate x labels for better readability
plt.show()
```



Inference: Majority of borrowsers have working experience greater than 10 years.

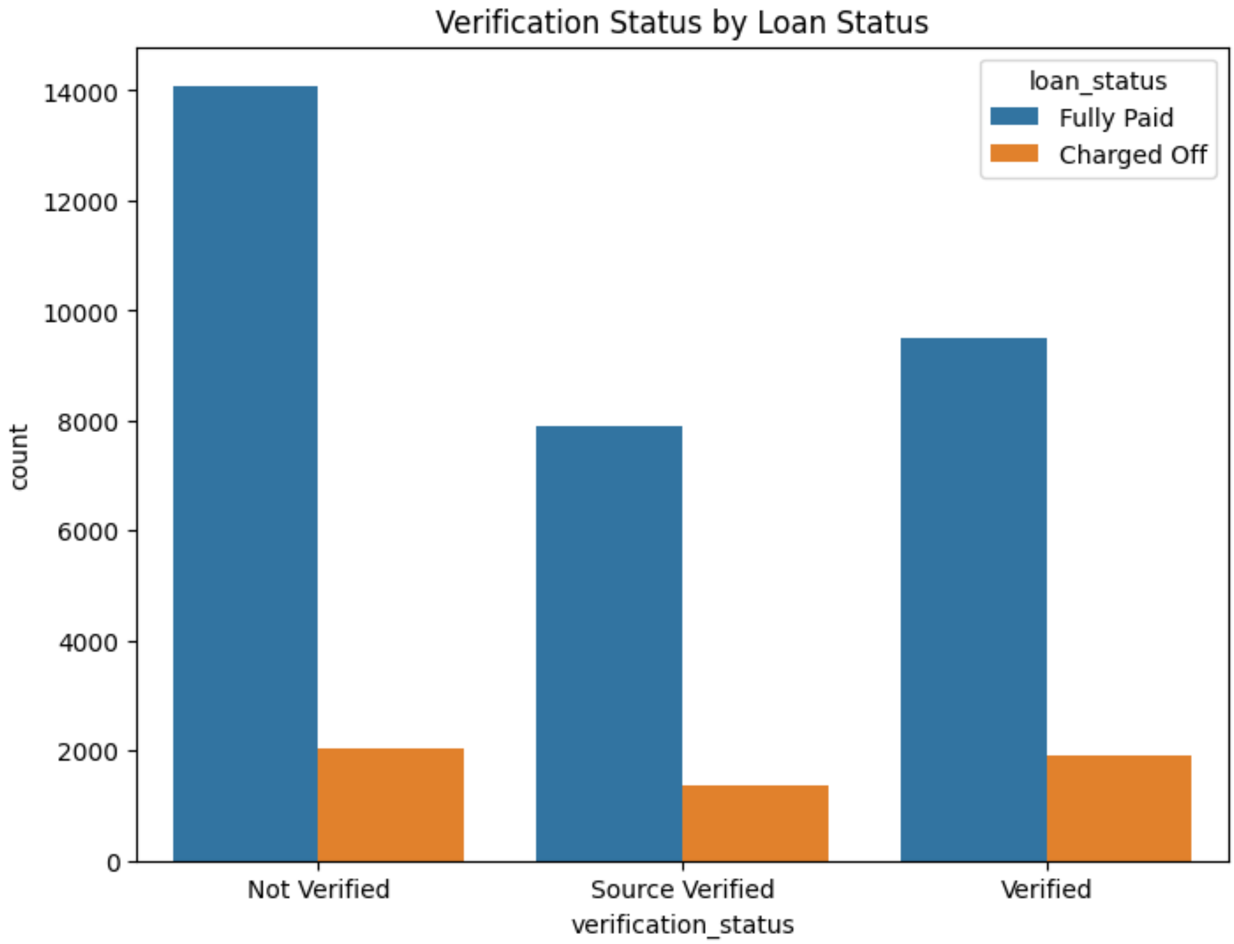
In [49]:
```
# DTI vs Loan Status
plt.figure(figsize=(10, 6))
sns.boxplot(x='loan_status', y='dti', data=loan_data)
plt.title('Debt-to-Income Ratio by Loan Status')
plt.show()
```



In [50]:
```
# Verification Status by Loan Status (`verification_status`)
verification_status_counts = loan_data['verification_status'].value_count
```

```
print("\nVerification Status Counts:")
print(verification_status_counts)

plt.figure(figsize=(8, 6))
sns.countplot(x='verification_status', data=loan_data, hue='loan_status')
plt.title('Verification Status by Loan Status')
plt.show()
```

```
Verification Status Counts:
verification_status
Not Verified       16117
Verified           11410
Source Verified     9267
Name: count, dtype: int64
```



In [51]:
```
# Calculate the counts of each verification status
# Calculate total loans
total_loans = len(loan_data)

# Calculate the counts and percentages for verification status
verification_status_counts = loan_data['verification_status'].value_count
verified_count = verification_status_counts['Verified'] + verification_st
not_verified_count = verification_status_counts['Not Verified']

verified_percentage = (verified_count / total_loans) * 100
not_verified_percentage = (not_verified_count / total_loans) * 100
```

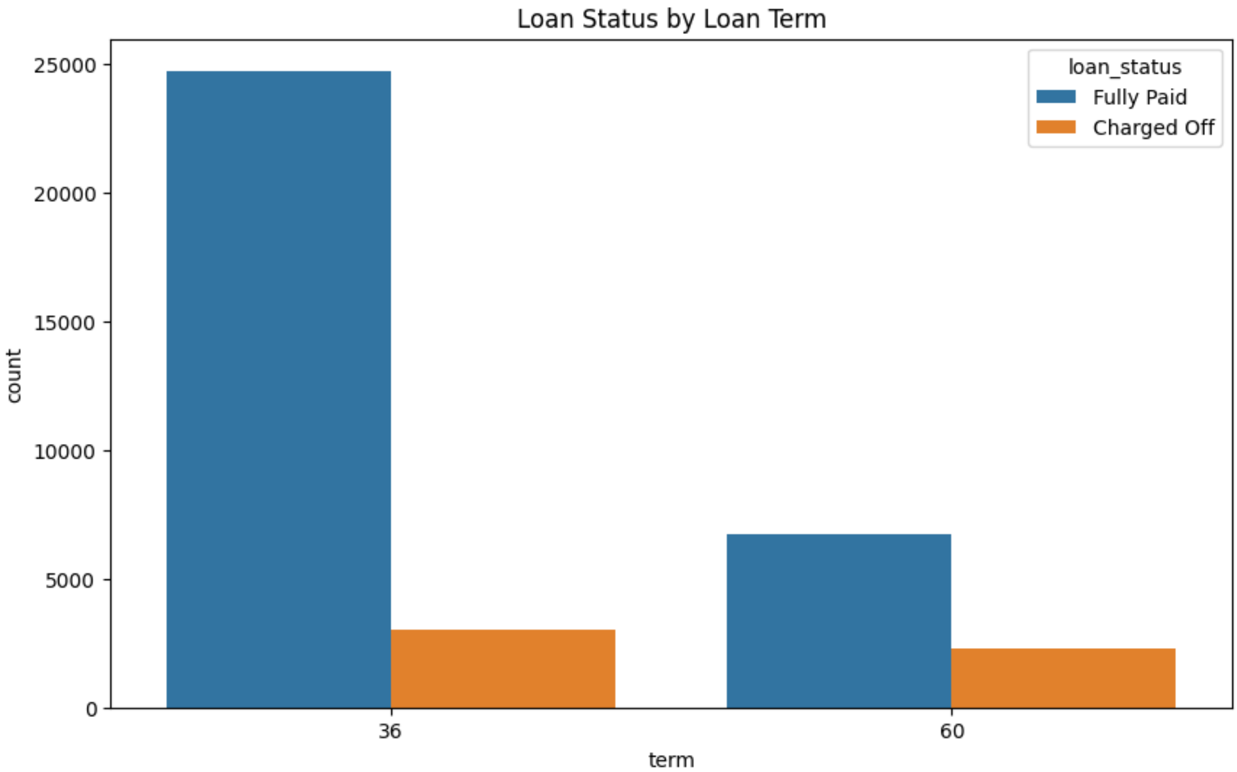
```
# Calculate charged-off loans within each group
charged_off_verified = loan_data[(loan_data['loan_status'] == 'Charged Of
                                  (loan_data['verification_status'].isin([
charged_off_not_verified = loan_data[(loan_data['loan_status'] == 'Charge
                                      (loan_data['verification_status'] ==

charged_off_verified_percentage = (charged_off_verified / verified_count)
charged_off_not_verified_percentage = (charged_off_not_verified / not_ver

# Print the results
print(f"Total Loans: {total_loans}")
print(f"Verified Loans: {verified_count} ({verified_percentage:.2f}%)")
print(f"Not Verified Loans: {not_verified_count} ({not_verified_percentag
print(f"Charged Off from Verified Loans: {charged_off_verified} ({charged
print(f"Charged Off from Not Verified Loans: {charged_off_not_verified} (
```

```
Total Loans: 36794
Verified Loans: 20677 (56.20%)
Not Verified Loans: 16117 (43.80%)
Charged Off from Verified Loans: 3294 (15.93%)
Charged Off from Not Verified Loans: 2029 (12.59%)
```

Inference: About 57% of the borrowers are verified by the company or have source verified. But it does not conclude anything as numbers are very closed.

In [52]:
```
# Analyze default rates by loan term
term_vs_status = pd.crosstab(loan_data['term'], loan_data['loan_status'],
print("\n--- Loan Term vs Loan Status ---")
print(term_vs_status)

plt.figure(figsize=(10, 6))
sns.countplot(x='term', data=loan_data, hue='loan_status')
plt.title('Loan Status by Loan Term')
plt.show()
```

```
--- Loan Term vs Loan Status ---
loan_status  Charged Off  Fully Paid
term
36              0.109242    0.890758
60              0.253457    0.746543
```

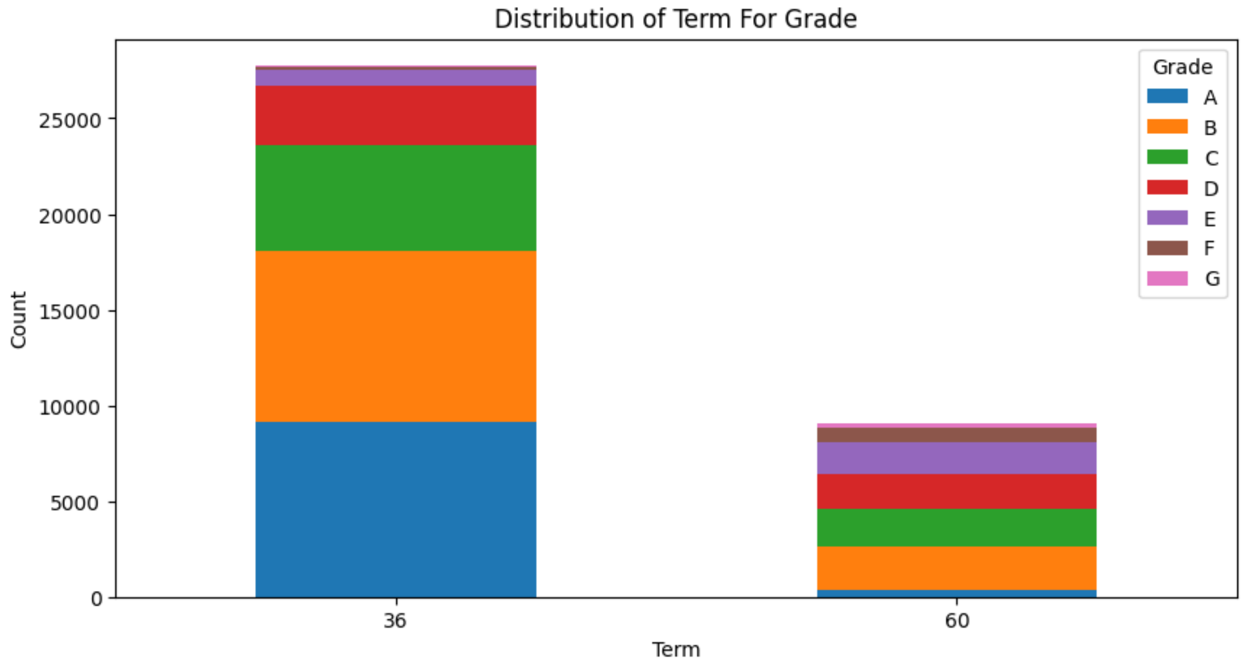


In [ ]:

Inference: The 60 month term has higher chance of defaulting than 36 month term whereas the 36 month term has higher chance of fully paid loan.

In [53]:
```
# Group the data by 'term' and 'grade' and count occurrencesInference: Th
term_grade_counts = loan_data.groupby(['term', 'grade']).size().unstack(f

# Plotting without seaborn
plt.figure(figsize=(10, 5))
term_grade_counts.plot(kind='bar', stacked=True, ax=plt.gca())
plt.xlabel('Term')
plt.ylabel('Count')
plt.title('Distribution of Term For Grade', fontsize=12)
plt.xticks(rotation=0)
plt.legend(title='Grade')
plt.show()
```

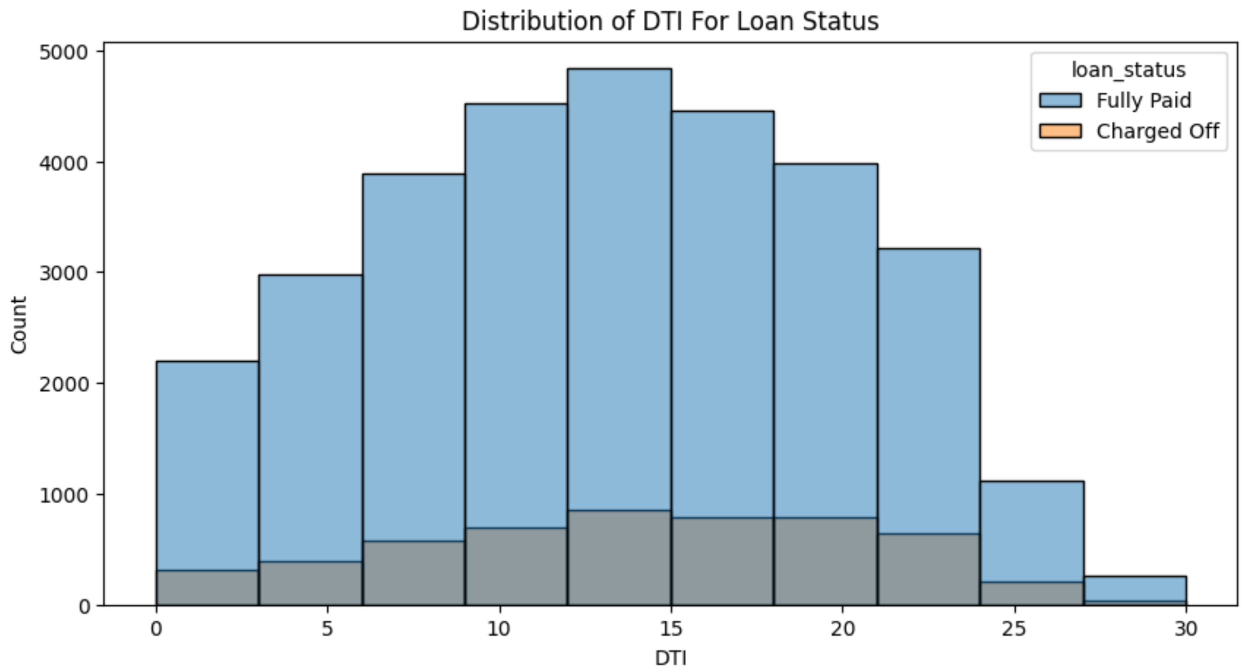
```
/var/folders/4v/6mdnm6fs2nn5qvhflh0fjzy80000gn/T/ipykernel_40097/262126539
4.py:2: FutureWarning: The default of observed=False is deprecated and wil
l be changed to True in a future version of pandas. Pass observed=False to
retain current behavior or observed=True to adopt the future default and s
ilence this warning.
  term_grade_counts = loan_data.groupby(['term', 'grade']).size().unstack(
fill_value=0)
```



Inference: The loans in 36 month term majorily consist of grade A and B loans whereas the loans in 60 month term mostly consist of grade B, C and D loans.

In [ ]:

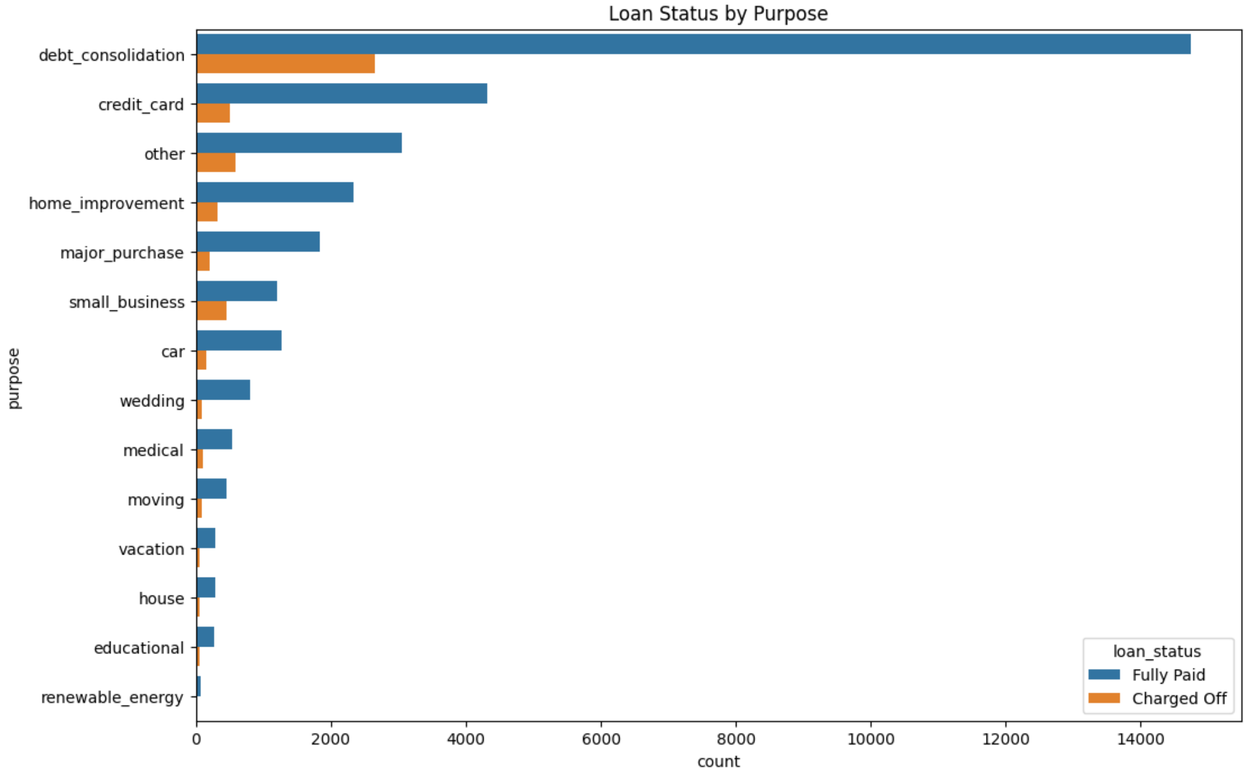
In [54]:
```
# Distribution of DTI based on Grade
plt.figure(figsize=(10,5))
sns.histplot(data=loan_data,x='dti',hue='loan_status',bins=10)
plt.xlabel('DTI')
plt.ylabel('Count')
plt.title('Distribution of DTI For Loan Status',fontsize=12)
plt.show()
```



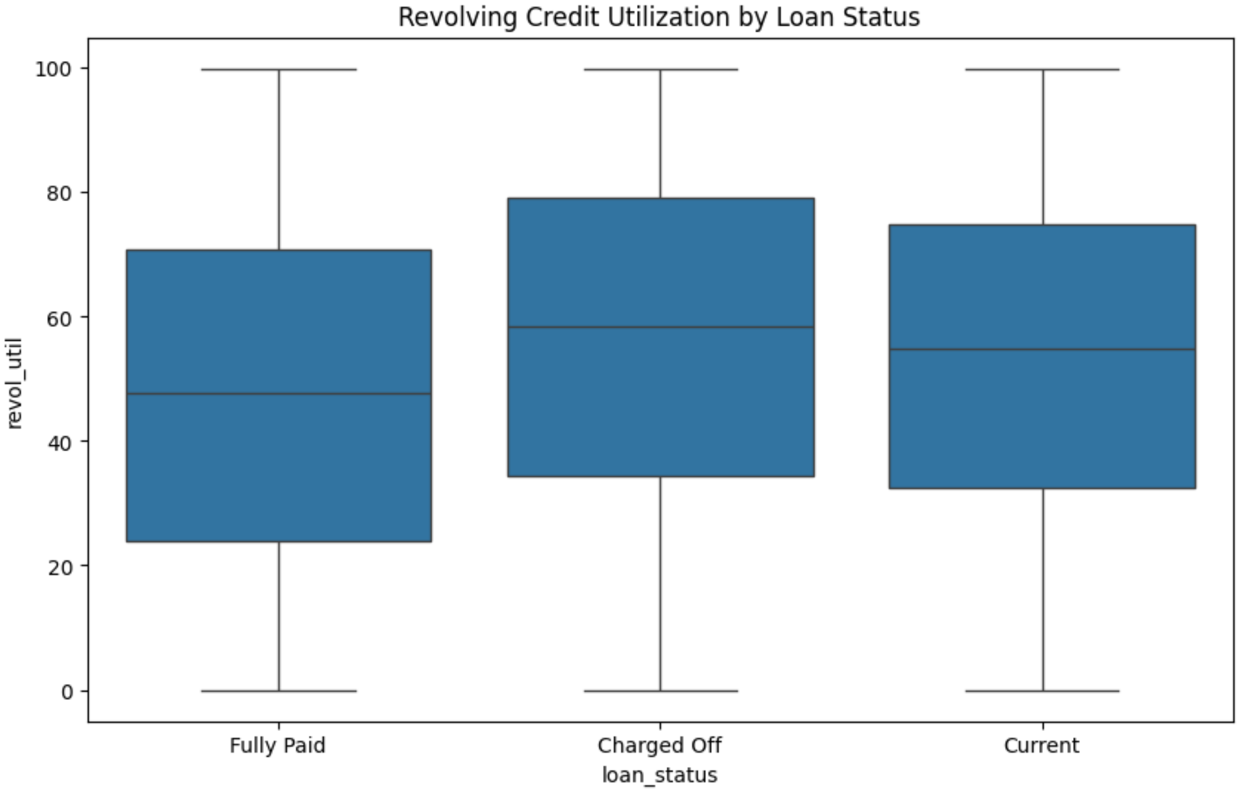
Inference: The Loan Status varies with DTI ratio, we can see that the loans in DTI ratio 10-15 have higher number of defaulted loan but higher dti has higher chance of defaulting.

In [55]:
```
# Analyze default rates by loan purpose
purpose_vs_status = pd.crosstab(loan_data['purpose'], loan_data['loan_sta
print("\n--- Purpose vs Loan Status ---")
print(purpose_vs_status)

plt.figure(figsize=(12, 8))
sns.countplot(y='purpose', data=loan_data, hue='loan_status', order=loan_
plt.title('Loan Status by Purpose')
plt.show()
```

```
--- Purpose vs Loan Status ---
loan_status          Charged Off  Fully Paid
purpose
car                     0.108467    0.891533
credit_card             0.104258    0.895742
debt_consolidation      0.152542    0.847458
educational             0.162939    0.837061
home_improvement        0.119955    0.880045
house                   0.167647    0.832353
major_purchase          0.102111    0.897889
medical                 0.158805    0.841195
moving                  0.151013    0.848987
other                   0.159428    0.840572
renewable_energy        0.191011    0.808989
small_business          0.275362    0.724638
vacation                0.144928    0.855072
wedding                 0.101449    0.898551
```



In [59]:
```
# Bivariate Analysis: Relationship between Revolving Credit Utilization a
revol_util_vs_status = loan_data.groupby('loan_status')['revol_util'].des
# Initialize a dictionary to store the logs
bivariate_logs = {}
bivariate_logs['Revolving Credit Utilization vs Loan Status'] = revol_uti
plt.figure(figsize=(10, 6))
sns.boxplot(x='loan_status', y='revol_util', data=loan_data)
plt.title('Revolving Credit Utilization by Loan Status')
plt.show()
```



Insights from Univariate Analysis

- The number of defaulted loan is 7 times less than the number of fully paid loan.
- The majority of loan has a term of 36 months compared to 60 months.
- The interest rate is more crowded around 5-10 and 10-15 with a drop near 10.
- A large amoutn of loans are with grade 'A' and 'B' commpared to rest showing most loans are high grade loans.
- Majority of borrowsers have working experience greater than 10 years.
- Majority of borrowsers don't posses property and are on mortage or rent.
- About 50% of the borrowers are verified by the company or have source verified.
- Annual Income shows left skewed normal distribution thus we can say that the majorityof burrowers

```
have very low annual income compared to rest.
•       A large percentage of loans are taken for debt
consolidation followed by credit card.
•       Majority of the borrowers are from the large urban
cities like california, new york, texas, florida etc.
•       Majority of the borrowers have very large debt
compared to the income registerd, concentrated in the 10-
15 DTI ratio.
•       Majority of the borrowers have no record of Public
Recorded Bankruptcy.
•       Majority of the loans are given in last quarter of
the year.
•       The number of loans approved increases with the
time at expontential rate, thus we can say that the loan
approval rate is increasing with the time.
```

Insights from Bivariate and Segmented Analysis

1. Interest Rate vs Loan Status

   • Insight: Borrowers who defaulted (Charged Off) had a higher average interest rate (13.82%) compared to those who fully paid their loans (11.61%). This suggests that higher interest rates are associated with a higher risk of default. • Action: Interest rate should be considered a key feature when predicting loan defaults.

2. Home Ownership vs Loan Status

   • Insight: Majority of borrowsers don't posses property and are on mortage or rent.

3. Employment Length vs Loan Status

   • Insight: There isn't a stark difference in default rates based on employment length. However, borrowers with 10+ years of employment had slightly higher default rates (14.99%) compared to other employment lengths. • Action: While employment length should be included in the model, it may not be as strong a predictor as interest rates or loan term.

4. Debt-to-Income Ratio vs Loan Status

   • Insight: Borrowers who defaulted had a slightly higher average DTI (14.00%) compared to those who fully paid their loans (13.15%). This suggests that higher DTI ratios are associated with higher risk. • Action: DTI should be used as a feature in predicting defaults, though the difference is not very large.

5. Verification Status by Loan Status

• Insight: Inference: About 57% of the borrowers are verified by the company or have source verified. But it does not conclude anything as numbers are very closed.

6. Term vs Loan Status

   • Insight: Loans with a 60-month term had a higher default rate (22.6%) compared to loans with a 36-month term (11.1%). Additionally, loans with a 60-month term also had a higher proportion of loans currently being paid (10.7%) compared to 36-month loans (0%). • Action: Loan term is an important variable in understanding loan performance, with longer-term loans being riskier.

7. Distribution of Grade vs Term

   • Insight: The loans in 36 month term majorily consist of grade A and B loans whereas the loans in 60 month term mostly consist of grade B, C and D loans.

8. Distribution of DTI For Loan Status • Insight: The Loan Status varies with DTI ratio, we can see that the loans in DTI ratio 10-15 have higher number of defaulted loan but higher dti has higher chance of defaulting. • Action: More Loan should be given to less DTI then 10 .

9. Purpose vs Loan Status

   • Insight: • The purpose of the loan significantly affects the default rate. Small business loans have the highest default rate (25.9%), while major purchases and weddings have the lowest (10.1%). • Educational, medical, and moving expenses also have relatively high default rates, indicating these purposes are riskier. • Action: Loan purpose is an important categorical variable and should be treated as a key feature in the model, particularly focusing on high-risk purposes like small business loans.

10. Home Ownership vs Loan Status

   • Insight: • The default rate is slightly higher for renters (15.0%) compared to those with a mortgage (13.2%) or who own their home (14.5%). • This suggests that homeownership provides some stability, although the differences are not as stark as in other segments. • Action: While home ownership status has some predictive value, it may not be as strong a predictor as income segment or loan term.

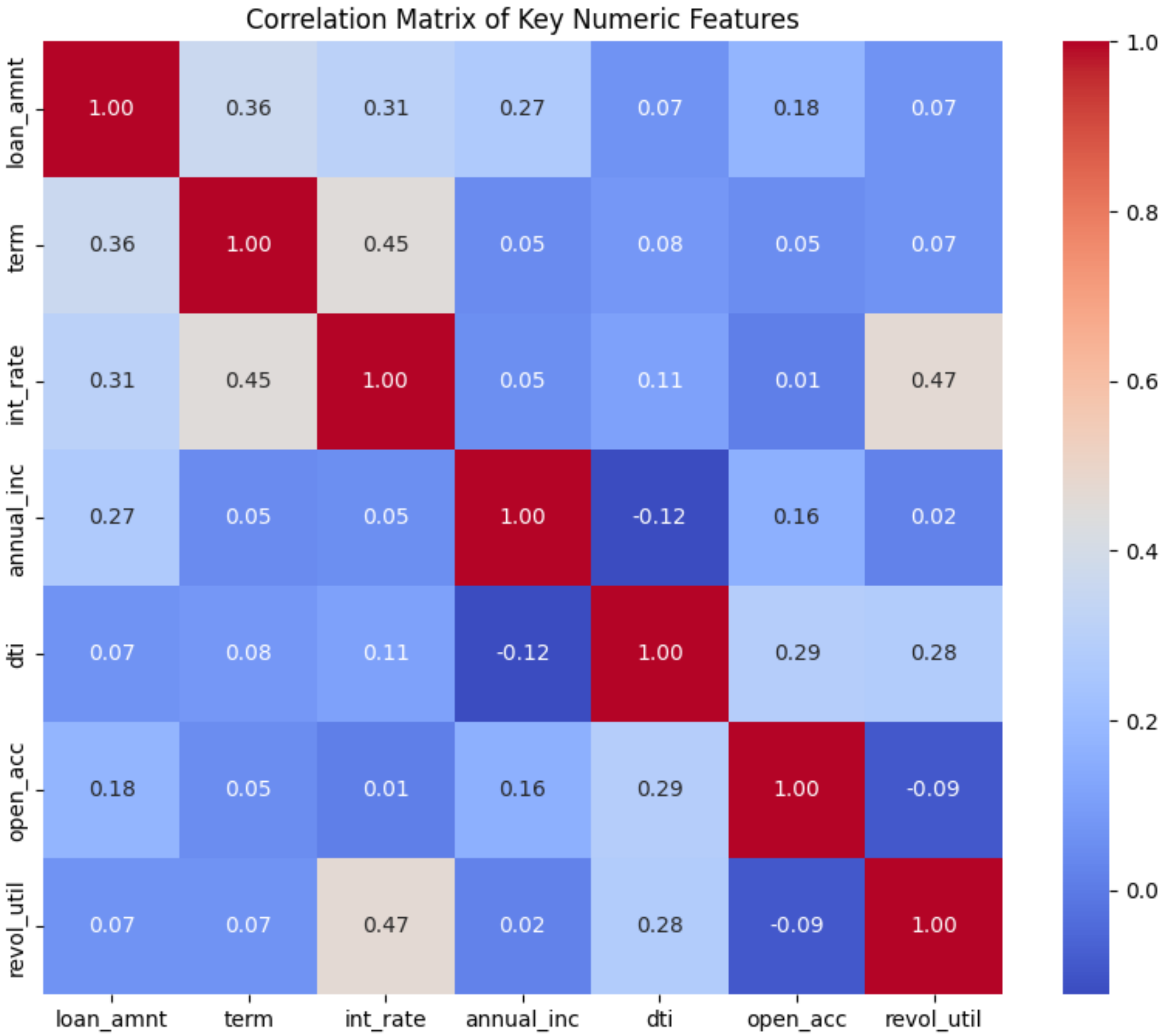
11. Revolving Credit Utilization vs Loan Status

   • Insight: Borrowers who defaulted had higher average revolving credit utilization (55.6%) compared to those who fully paid their loans (47.5%). Higher credit utilization is clearly associated with higher default risk. • Action: Revolving credit utilization is a key feature that should be included in the predictive model.

Correlation Matrix based on 3 variables

In [56]:
```
# Load the dataset
loan_data = pd.read_csv('loan.csv')

# Clean the data: Extract numeric values from the `term` column and conve
loan_data['term'] = loan_data['term'].apply(lambda x: int(x.strip().split
loan_data['int_rate'] = loan_data['int_rate'].str.replace('%', '').astype
loan_data['revol_util'] = loan_data['revol_util'].str.replace('%', '').as

# Select relevant numeric features for correlation analysis
numeric_features = ['loan_amnt', 'term', 'int_rate', 'annual_inc', 'dti',

# Calculate correlation matrix
corr_matrix = loan_data[numeric_features].corr()

# Plot the correlation heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix of Key Numeric Features')
plt.show()

# Print the correlation matrix for reference
print(corr_matrix)
```

```
/var/folders/4v/6mdnm6fs2nn5qvhflh0fjzy80000gn/T/ipykernel_40097/278088519
1.py:2: DtypeWarning: Columns (47) have mixed types. Specify dtype option
on import or set low_memory=False.
  loan_data = pd.read_csv('loan.csv')
```


Correlation Matrix of Key Numeric Features
loan_amnt
term
int_rate
annual_inc
dti
open_acc
revol_util
1.00 0.36 0.31 0.27 0.07 0.18 0.07
0.36 1.00 0.45 0.05 0.08 0.05 0.07
0.31 0.45 1.00 0.05 0.11 0.01 0.47
0.27 0.05 0.05 1.00 -0.12 0.16 0.02
0.07 0.08 0.11 -0.12 1.00 0.29 0.28
0.18 0.05 0.01 0.16 0.29 1.00 -0.09
0.07 0.07 0.47 0.02 0.28 -0.09 1.00
1.0
0.8
0.6
0.4
0.2
0.0

```
              loan_amnt      term  int_rate  annual_inc       dti  open_acc  \
loan_amnt      1.000000  0.361036  0.309415    0.271149  0.066439  0.177168
term           0.361036  1.000000  0.451699    0.046675  0.082426  0.050769
int_rate       0.309415  0.451699  1.000000    0.053185  0.111162  0.010395
annual_inc     0.271149  0.046675  0.053185    1.000000 -0.122732  0.158200
dti            0.066439  0.082426  0.111162   -0.122732  1.000000  0.288045
open_acc       0.177168  0.050769  0.010395    0.158200  0.288045  1.000000
revol_util     0.066149  0.069834  0.467168    0.017926  0.277951 -0.089891

              revol_util
loan_amnt       0.066149
term            0.069834
int_rate        0.467168
annual_inc      0.017926
dti             0.277951
open_acc       -0.089891
revol_util      1.000000
```

Insights:

1. Loan Amount and Term: There's a moderate positive correlation (0.36) between loan amount and loan term, indicating that larger loans are more likely to have longer terms.
2. Interest Rate and Term: A moderate positive correlation (0.45) exists between interest rate and loan term, suggesting that longer-term loans tend to have higher interest rates.
3. Interest Rate and Revolving Credit Utilization: A relatively strong positive correlation (0.47) between interest rate and revolving credit utilization indicates that borrowers with higher credit utilization tend to receive higher interest rates.
4. Weak Correlations: Most other correlations are weak, indicating that the variables contribute independently to the likelihood of default without much redundancy.

Action:

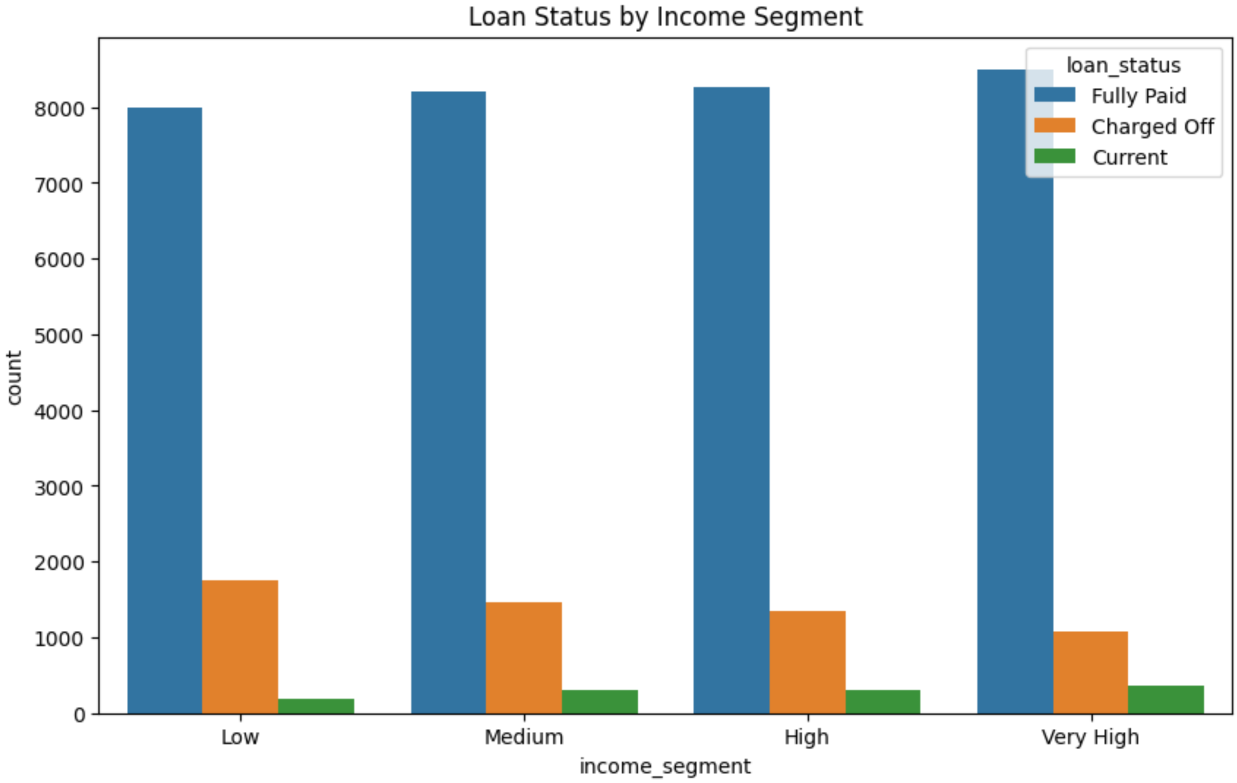
1. Given the moderate correlation between interest rate and revolving credit utilization, both features should be included in any analysis, but be aware of potential multicollinearity.
2. The weak correlations between most features suggest that they capture different aspects of borrower risk, making them all valuable in a holistic risk assessment.

Derived Metrics based Analysis

In [57]:
```
# Segment borrowers into income quartiles
loan_data['income_segment'] = pd.qcut(loan_data['annual_inc'], 4, labels=
```

```
# Analyze default rates by income segment
income_segment_vs_status = pd.crosstab(loan_data['income_segment'], loan_
print("\n--- Income Segment vs Loan Status ---")
print(income_segment_vs_status)

plt.figure(figsize=(10, 6))
sns.countplot(x='income_segment', data=loan_data, hue='loan_status')
plt.title('Loan Status by Income Segment')
plt.show()
```

```
--- Income Segment vs Loan Status ---
loan_status     Charged Off   Current  Fully Paid
income_segment
Low                0.177039  0.018127    0.804834
Medium             0.145984  0.030522    0.823494
High               0.135886  0.030208    0.833906
Very High          0.107765  0.035955    0.856280
```



Recommendations

Major Driving factor which can be used to predict the chance of defaulting and avoiding Credit Loss:

- DTI
- Grades
- Verification Status
- Annual income
- Pub_rec_bankruptcies

Other considerations for 'defaults' :

- Burrowers not from large urban cities like california, new york, texas, florida etc.
- Burrowers having annual income in the range 50000-100000.
- Burrowers having Public Recorded Bankruptcy.
- Burrowers with least grades like E,F,G which indicates high risk.
- Burrowers with very high Debt to Income value.
- Burrowers with working experience 10+ years.

Incorporate Findings

To incorporate the findings into decision-making using specific numbers or ranges, you can set thresholds and ranges based on the statistical analysis we've conducted. Here's how you can proceed:

1. Income-Based Loan Approval Criteria

Income Segment Thresholds

```
•       Low Income Segment: Borrowers with annual income
below approximately $40,404 (25th percentile, from the
earlier annual_inc.describe() output) fall into the Low
income segment.
•       Decision Criteria:
•       Loan Amount: Limit the loan amount for borrowers
in the Low income segment to a maximum of $10,000. This
cap reduces exposure to risk.
•       Interest Rate: Apply a higher interest rate, e.g.,
2% above the average rate, for borrowers in this segment
to compensate for the higher risk.
•       Loan Term: Offer only 36-month terms to borrowers
in this segment. Discourage or disallow 60-month terms due
to their higher default rates.
•       Very High Income Segment: Borrowers with annual
income above approximately $82,300 (75th percentile) fall
into the Very High income segment.
•       Decision Criteria:
•       Loan Amount: Allow higher loan amounts, up to the
maximum loan amount available ($35,000).
•       Interest Rate: Offer a lower interest rate, e.g.,
1% below the average rate, to encourage borrowing from
lower-risk borrowers.
•       Loan Term: Allow flexibility in choosing between
36 and 60 months, with possibly better terms (e.g., lower
fees or rates) for 36-month loans.
```

2. Loan Term Criteria

Term-Based Adjustments

- 60-Month Term:
- Income Requirement: Require a minimum annual income of $60,000 for approval of a 60-month loan term. This threshold is set above the median income to ensure that longer-term loans are extended only to more financially stable borrowers.
- Interest Rate Adjustment: For 60-month loans, add an additional 0.5% to 1% on the interest rate compared to 36-month loans to account for the higher risk of default.
- 36-Month Term:
- Income Flexibility: Offer loans to borrowers with incomes as low as $40,000 but apply stricter limits on the loan amount and interest rate if the income is below $50,000.
- Interest Rate: Offer standard interest rates or potentially lower rates to encourage shorter-term borrowing.

3. Purpose-Based Adjustments

High-Risk Loan Purposes

- Small Business Loans:
- Default Rate: With a default rate of approximately 25.9%, impose stringent requirements:
- Minimum Income: Require a minimum income of $75,000.
- Collateral: Demand collateral or a co-signer to mitigate risk.
- Loan Amount: Cap the loan amount at $15,000 for small business purposes.
- Other High-Risk Purposes (e.g., Educational, Medical):
- Minimum Income: Set a minimum income threshold of $50,000.
- Interest Rate: Apply a risk premium of 1.5% to 2% above the standard rate.
- Loan Term: Limit the term to 36 months unless the borrower has an income exceeding $80,000, in which case a 60-month term might be considered.

Low-Risk Loan Purposes

- Credit Card Consolidation:
- Default Rate: With a lower default rate (~10.6%), offer favorable terms:
- Interest Rate: Provide competitive rates,

```
potentially below the average, to encourage consolidation.
•       Loan Amount: Allow up to $35,000, especially if
the borrower’s income is above $60,000.
•       Major Purchases/Weddings:
•       Terms: Offer standard or slightly favorable terms,
with interest rates 0.5% below average for those in the
High or Very High income segments.
```

4. Revolving Credit Utilization

Utilization-Based Risk Adjustment

```
•       High Utilization (Above 50%):
•       Interest Rate: Increase the interest rate by 1%
for borrowers with credit utilization above 50%.
•       Loan Amount: Reduce the maximum loan amount
available by 20% to 30% for high utilization borrowers.
•       Credit Monitoring: Implement more frequent
monitoring of borrowers’ credit utilization if they are
approved for a loan.
•       Low Utilization (Below 30%):
•       Incentives: Offer better terms, such as reduced
interest rates (0.5% to 1% below average) or higher loan
amounts (up to $35,000), for borrowers with lower
utilization rates.
```

5. Implementation and Monitoring

   • Regular Reviews: Conduct quarterly reviews of loan performance data to assess whether these criteria effectively reduce default rates. Adjust thresholds and rates as necessary.

In [ ]: