Project Phase #1

Teammate 1: Harikumar Reddy Vengi Reddy (50518453)
Teammate 2: Deekshitha Devalla (50546172)

Problem Statements, Data Acquisition, Data Processing, Data Cleaning, EDA

LOAN APPLICANT DATA FOR CREDIT RISK ANALYSIS

Problem Statement: In the context of financial lending, the goal is to develop predictive models that evaluate an applicant's creditworthiness in respect to loans.

The purpose of this task is to predict the probability of loan default using the dataset containing applicant characteristics such as age, income, home ownership status, years in job, intended purpose for loan, loan amount, interest rate, historical time of borrower history, and records of missed payments or this would aim at giving lenders a credible tool that helps them make safer choices, reduce default risks and enhance assessment processes in general.

As one of the most important issues in the modern financial and lending system, it concerns a possibility to estimate credit risk among loan applicants.

Background of the problem, significance, and importance:

- The main reason to choose this problem is for financial stability, without assessing the credit risk it is hard to manage the risks in the financial sector.
- To reduce the defaults predicting the risk always helps, so that lenders can reduce the likelihood of defaults.
- Interest rates can be set by the lenders before they sanction the loan amount through the credit risk analysis like if the borrower with lower credit risk can be given a low loan interest rate and with a higher risk is charged with the higher interest rate.
- It is significant for various reasons, Reducing risk exposure to the barest minimum by
  addressing key risks on a continuous basis. Healthy lending environment is attained via risk prediction.
- Therefore, since the problem of credit risk assessment has wide implications on such issues as financial stability, responsible lending, economic growth etc. It constitutes a serious problem where lenders need to find the right models to predict risk effectively.
- This is crucial as it enables lenders to make informed decisions backed by data that are fair to all parties in view of creating a trustworthy and legitimate environment for loans and securities.

Data source: Dataset is taken from Kaggle: https://www.kaggle.com/datasets

Dataset Description: The dataset contains all relevant information regarding applicants of loans and their attributes.

Features are age, annual income, home ownership, employment length (in years), loan intent, loan grade, loan amount, loan interest rate, loan status, loan percent income, default history, credit history length.

Data Cleaning/Processing:

Read the csv file to risk_df.

```
risk_df = pd.read_csv('credit_risk_dataset.csv')
```

```
risk_df.shape
```

```
(32581, 12)
```

Displayed information of dataset.

```
risk_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32581 entries, 0 to 32580
Data columns (total 12 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   person_age                  32581 non-null  int64  
 1   person_income               32581 non-null  int64  
 2   person_home_ownership       32581 non-null  object 
 3   person_emp_length           31686 non-null  float64
 4   loan_intent                 32581 non-null  object 
 5   loan_grade                  32581 non-null  object 
 6   loan_amnt                   32581 non-null  int64  
 7   loan_int_rate               29465 non-null  float64
 8   loan_status                 32581 non-null  int64  
 9   loan_percent_income         32581 non-null  float64
 10  cb_person_default_on_file   32581 non-null  object 
 11  cb_person_cred_hist_length  32581 non-null  int64  
dtypes: float64(3), int64(5), object(4)
memory usage: 3.0+ MB
```

1. Data Imputation

```
[]: risk_df.isnull().sum()
```

```
[]: person_age                     0
    person_income                  0
    person_home_ownership          0
    person_emp_length            895
    loan_intent                    0
    loan_grade                     0
    loan_amnt                      0
    loan_int_rate               3116
    loan_status                    0
    loan_percent_income            0
    cb_person_default_on_file      0
    cb_person_cred_hist_length     0
    dtype: int64
```

```
risk_df['person_emp_length'].fillna(risk_df['person_emp_length'].median(), inplace=True)
risk_df.info()
risk_df.isnull().sum()
```

person_emp_length has 895 null values; those nan values are replaced by the median of that feature.

```
risk_df['loan_int_rate'].fillna(risk_df['loan_int_rate'].mean(), inplace=True)
risk_df.info()
risk_df.isnull().sum()
```

Similarly, loan_int_rate null values are also replaced by the mean value of that column.

2. Dropping Null values

```
risk_df.dropna(inplace = True)
```

```
risk_df.isnull().sum()
```

```
person_age                  0
person_income               0
person_home_ownership       0
person_emp_length           0
loan_intent                 0
loan_grade                  0
loan_amnt                   0
loan_int_rate               0
loan_status                 0
loan_percent_income         0
cb_person_default_on_file   0
cb_person_cred_hist_length  0
dtype: int64
```

Finally, we don't have any null values in the entire dataset.

3. Checking for the duplicates and removing

```
duplicates = risk_df.duplicated()
```

```
duplicates
```

```
1        False
2        False
3        False
4        False
5        False
         ...
32576    False
32577    False
32578    False
32579    False
32580    False
Length: 32571, dtype: bool
```

```
duplicates.unique()
```

```
array([False,  True])
```

```
risk_df = risk_df.drop_duplicates()
```

```
dups = risk_df.duplicated()
```

```
dups.unique()
```

```
array([False])
```

```
risk_df.shape
```

```
(32406, 22)
```

4. Data type Conversion

```
risk_df['person_income'] = risk_df['person_income'].astype('float')
```

```
risk_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32581 entries, 0 to 32580
Data columns (total 12 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   person_age                  32581 non-null  int64
 1   person_income               32581 non-null  float64
 2   person_home_ownership       32581 non-null  object
 3   person_emp_length           31686 non-null  float64
 4   loan_intent                 32581 non-null  object
 5   loan_grade                  32581 non-null  object
 6   loan_amnt                   32581 non-null  int64
 7   loan_int_rate               29465 non-null  float64
 8   loan_status                 32581 non-null  int64
 9   loan_percent_income         32581 non-null  float64
 10  cb_person_default_on_file   32581 non-null  object
 11  cb_person_cred_hist_length  32581 non-null  int64
dtypes: float64(4), int64(4), object(4)
memory usage: 3.0+ MB
```

5. String type conversion (mismatch handling)

All the object data types are converted into upper case.

```
risk_df['person_home_ownership'] = risk_df['person_home_ownership'].str.upper()
risk_df['person_home_ownership'].unique()
```

```
array(['RENT', 'OWN', 'MORTGAGE', 'OTHER'], dtype=object)
```

```
risk_df['loan_intent'] = risk_df['loan_intent'].str.upper()
risk_df["loan_intent"].unique()
```

```
array(['PERSONAL', 'EDUCATION', 'MEDICAL', 'VENTURE', 'HOMEIMPROVEMENT',
       'DEBTCONSOLIDATION'], dtype=object)
```

```
risk_df['loan_grade'] = risk_df['loan_grade'].str.upper()
risk_df["loan_grade"].unique()
```

```
array(['D', 'B', 'C', 'A', 'E', 'F', 'G'], dtype=object)
```

```
risk_df["loan_status"].unique()
```

```
array([1, 0])
```

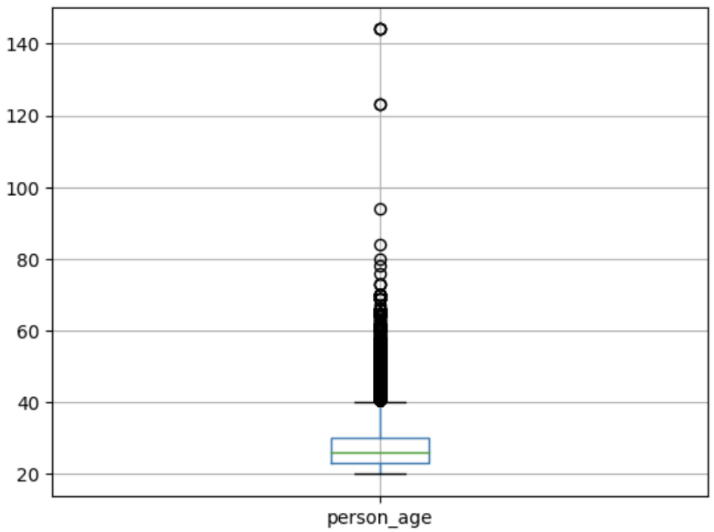
```
risk_df['cb_person_default_on_file'] = risk_df['cb_person_default_on_file'].str.upper()
risk_df["cb_person_default_on_file"].unique()
```

```
array(['Y', 'N'], dtype=object)
```

6. Detecting Outliers and removing

There are some features, where outliers are detected, those outliers are removed.
For example:
person_age column has very few outliers whose age is greater than 80 years, so these are outliers detected upon plotting a boxplot. These are removed from the dataset.

```
import seaborn as sns
figure = risk_df.boxplot(column='person_age')
plt.show()
```

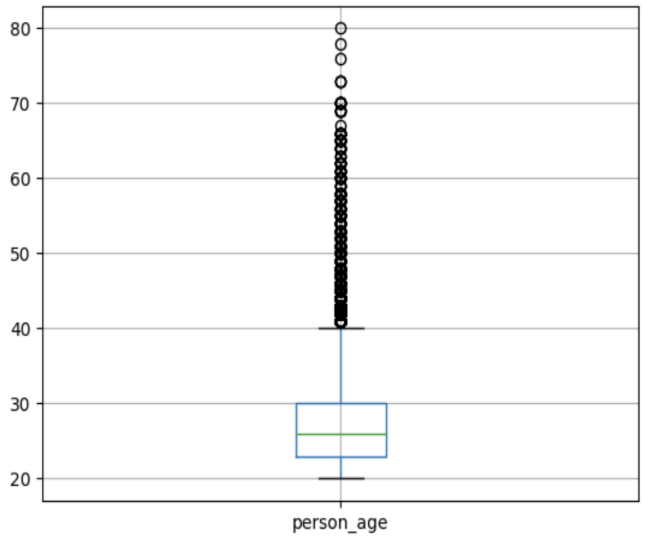


After removing outliers.

```
risk_df[risk_df['person_age'] > 80.0].count()
```

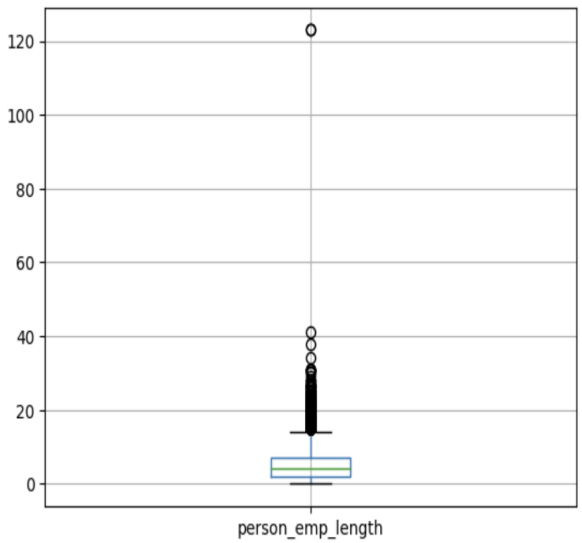
```
person_age                    0
person_income                 0
person_home_ownership         0
person_emp_length             0
loan_intent                   0
loan_grade                    0
loan_amnt                     0
loan_int_rate                 0
loan_status                   0
loan_percent_income           0
cb_person_default_on_file     0
cb_person_cred_hist_length    0
dtype: int64
```

```
figure = risk_df.boxplot(column='person_age')
plt.show()
```



Similarly, outliers are removed for other features as well.

```
figure = risk_df.boxplot(column='person_emp_length')
plt.show()
```



```
risk_df = risk_df.drop(risk_df[risk_df['person_emp_length'] > 40.0].index, axis=0)
```

```
risk_df[risk_df['person_emp_length'] > 40.0].count()
```

```
person_age                    0
person_income                 0
person_home_ownership         0
person_emp_length             0
loan_intent                   0
loan_grade                    0
loan_amnt                     0
loan_int_rate                 0
loan_status                   0
loan_percent_income           0
cb_person_default_on_file     0
cb_person_cred_hist_length    0
dtype: int64
```

7. Feature Engineering

New features are to be added representing existing features with some modifications.
person_age column data is taken and age_range a new column is added as follows:

```
risk_df['age_range'] = pd.cut(risk_df['person_age'],
                          bins=[20, 31, 41, 51, 61, 71, 81],
                          labels=['20-30', '31-40', '41-50', '51-60', '61-70', '71-80'])
risk_df['age_range']
```

```
1        20-30
2        20-30
3        20-30
4        20-30
5        20-30
         ...
32576    51-60
32577    51-60
32578    61-70
32579    51-60
32580    61-70
Name: age_range, Length: 32571, dtype: category
Categories (6, object): ['20-30' < '31-40' < '41-50' < '51-60' < '61-70' < '71-80']
```

Similarly, income_range is added upon analysing person_income data values.

```
risk_df['income_range'] = pd.cut(risk_df['person_income'],
                             bins=[0, 30000, 50000, 75000, 100000, 150000, float('inf')],
                             labels=['0-30k', '30k-50k', '50k-75k', '75k-100k', '100k-150k', '>150k'])
risk_df['income_range']
```

```
1            0-30k
2            0-30k
3          50k-75k
4          50k-75k
5            0-30k
           ...
32576      50k-75k
32577    100k-150k
32578     75k-100k
32579    100k-150k
32580      30k-50k
Name: income_range, Length: 32571, dtype: category
Categories (6, object): ['0-30k' < '30k-50k' < '50k-75k' < '75k-100k' < '100k-150k' < '>150k']
```

loan_amount_range is a new column added for loan_amnt feature.

```
risk_df['loan_amount_range'] = pd.cut(risk_df['loan_amnt'],
                                      bins=[0, 5000, 10000, 15000, 20000, 25000, 30000, float('inf')],
                                      labels=['0-5k', '5k-10k', '10k-15k', '15k-20k', '20k-25k', '25k-30k', '>30k'])
risk_df['loan_amount_range']
```

```
1           0-5k
2          5k-10k
3            >30k
4            >30k
5           0-5k
           ...
32576      5k-10k
32577     15k-20k
32578        >30k
32579     10k-15k
32580      5k-10k
Name: loan_amount_range, Length: 32571, dtype: category
Categories (7, object): ['0-5k' < '5k-10k' < '10k-15k' < '15k-20k' < '20k-25k' < '25k-30k' < '>30k']
```

8. Data Categorization

Datatype of a few features is converted into category data type and new columns are added to the data set for a proper analysis.
Ownership column is categorised into 4 categories based on their unique values.

```
risk_df['person_home_ownership_cat'] = risk_df.person_home_ownership.astype('category')
```

```
risk_df['person_home_ownership_cat'] = risk_df['person_home_ownership_cat'].cat.codes
```

```
risk_df['person_home_ownership'].unique()
```

```
array(['OWN', 'MORTGAGE', 'RENT', 'OTHER'], dtype=object)
```

```
risk_df['person_home_ownership_cat'].unique()
```

```
array([2, 0, 3, 1], dtype=int8)
```

loan_intent feature is also categorised as follows:

```
risk_df['loan_intent_cat'] = risk_df.loan_intent.astype('category')
```

```
risk_df['loan_intent_cat'] = risk_df['loan_intent_cat'].cat.codes
```

```
risk_df['loan_intent'].unique()
```

```
array(['EDUCATION', 'MEDICAL', 'VENTURE', 'PERSONAL', 'HOMEIMPROVEMENT',
       'DEBTCONSOLIDATION'], dtype=object)
```

```
risk_df['loan_intent_cat'].unique()
```

```
array([1, 3, 5, 4, 2, 0], dtype=int8)
```

Similarly, the columns which can be categorised are listed below.

```
risk_df['loan_grade_cat'] = risk_df.loan_grade.astype('category')
```

```
risk_df['loan_grade_cat'] = risk_df['loan_grade_cat'].cat.codes
```

```
risk_df['loan_grade_cat'].unique()
```

```
array([1, 2, 0, 3, 4, 5, 6], dtype=int8)
```

```
risk_df['loan_grade'].unique()
```

```
array(['B', 'C', 'A', 'D', 'E', 'F', 'G'], dtype=object)
```

```
risk_df['cb_person_default_on_file_cat'] = risk_df.cb_person_default_on_file.astype('category')
```

```
risk_df['cb_person_default_on_file_cat'] = risk_df['cb_person_default_on_file_cat'].cat.codes
```

```
risk_df['cb_person_default_on_file_cat'].unique()
```

```
array([0, 1], dtype=int8)
```

```
risk_df['cb_person_default_on_file'].unique()
```

```
array(['N', 'Y'], dtype=object)
```

9. Data Transformation

Initially, we added new features as part of feature engineering, data based on range values; those columns can also be categorised.

```
risk_df['age_range_cat'] = risk_df.age_range.astype('category')
risk_df['age_range_cat'] = risk_df['age_range_cat'].cat.codes
```

```
risk_df['income_range_cat'] = risk_df.income_range.astype('category')
risk_df['income_range_cat'] = risk_df['income_range_cat'].cat.codes
```

```
risk_df['loan_amount_range_cat'] = risk_df.loan_amount_range.astype('category')
risk_df['loan_amount_range_cat'] = risk_df['loan_amount_range_cat'].cat.codes
```

```
risk_df
```

_range	loan_intent_cat	person_home_ownership_cat	loan_grade_cat	cb_person_default_on_file_cat	age_range_cat	income_range_cat	loan_amount_range_cat
0-5k	1	2	1	0	0	0	0
5k-10k	3	0	2	0	0	0	1
>30k	3	3	2	0	0	2	6
>30k	3	3	2	1	0	2	6
0-5k	5	2	0	0	0	0	0
...	...	...	...	...	...	...	...
5k-10k	4	0	2	0	3	2	1
5k-20k	4	0	0	0	3	4	3
>30k	2	3	1	0	4	3	6
10k-15k	4	0	1	0	3	4	2
5k-10k	3	3	1	0	4	1	1

10. Data Normalisation

To scale the data in certain range, we need to normalise the features that are applicable, and the formula used is
X_normalized = (X - X_min)/(X_max - X_min)

The advantage of data normalisation is that it makes sure various properties or elements or, in general, aspects of the dataset have equal weight for a particular machine-learning algorithm or, at least as much weight as possible.

The use of multivariate techniques for analysing the data improves stability and performance of the algorithms as multivariate techniques are helpful in interpreting such data.

Selection of a normalised measurement method is based on particularities of the data and demands in a given analysis or modelling process.

```
person_emp_length_min = risk_df['person_emp_length'].min()
person_emp_length_max = risk_df['person_emp_length'].max()

print(person_emp_length_min, person_emp_length_max)
risk_df_copy = risk_df.copy()

risk_df_copy['person_emp_length_normalized'] = (risk_df['person_emp_length'] - person_emp_length_min)/(person_emp_length_max-person_emp_length
```

```
0.0 38.0
```

```
risk_df = risk_df_copy
```

```
risk_df['person_emp_length'],risk_df['person_emp_length_normalized']
```

```
(1        5.0
 2        1.0
 3        4.0
 4        8.0
 5        2.0
         ...
 32576    1.0
 32577    4.0
 32578    3.0
 32579    5.0
 32580    2.0
 Name: person_emp_length, Length: 32406, dtype: float64,
 1        0.131579
 2        0.026316
 3        0.105263
 4        0.210526
 5        0.052632
            ...
 32576    0.026316
 32577    0.105263
 32578    0.078947
 32579    0.131579
 32580    0.052632
 Name: person_emp_length_normalized, Length: 32406, dtype: float64)
```

```
loan_int_rate_min = risk_df['loan_int_rate'].min()
loan_int_rate_max = risk_df['loan_int_rate'].max()

print(loan_int_rate_min, loan_int_rate_max)
risk_df_copy = risk_df.copy()

risk_df_copy['loan_int_rate_normalized'] = (risk_df['loan_int_rate'] - loan_int_rate_min)/(loan_int_rate_max-loan_int_rate_min)
```

```
5.42 23.22
```

```
risk_df = risk_df_copy
```

```
risk_df['loan_int_rate'],risk_df['loan_int_rate_normalized']
```

```
(1        11.14
 2        12.87
 3        15.23
 4        14.27
 5         7.14
          ...
 32576    13.16
 32577     7.49
 32578    10.99
 32579    11.48
 32580     9.99
 Name: loan_int_rate, Length: 32406, dtype: float64,
 1        0.321348
 2        0.418539
 3        0.551124
 4        0.497191
 5        0.096629
            ...
 32576    0.434831
 32577    0.116292
 32578    0.312921
 32579    0.340449
 32580    0.256742
 Name: loan_int_rate_normalized, Length: 32406, dtype: float64)
```

```
cb_person_cred_hist_length_min = risk_df['cb_person_cred_hist_length'].min()
cb_person_cred_hist_length_max = risk_df['cb_person_cred_hist_length'].max()

print(cb_person_cred_hist_length_min, cb_person_cred_hist_length_max)
risk_df_copy = risk_df.copy()

risk_df_copy['cb_person_cred_hist_length_normalized'] = (risk_df['cb_person_cred_hist_length'] - cb_person_cred_hist_length_min)/(cb_person_cr
```

```
2 30
```

```
risk_df = risk_df_copy
```

```
risk_df['cb_person_cred_hist_length'],risk_df['cb_person_cred_hist_length_normalized']
```

```
(1         2
 2         3
 3         2
 4         4
 5         2
          ..
 32576    30
 32577    19
 32578    28
 32579    26
 32580    30
 Name: cb_person_cred_hist_length, Length: 32406, dtype: int64,
 1        0.000000
 2        0.035714
 3        0.000000
 4        0.071429
 5        0.000000
            ...
 32576    1.000000
 32577    0.607143
 32578    0.928571
 32579    0.857143
 32580    1.000000
 Name: cb_person_cred_hist_length_normalized, Length: 32406, dtype: float64)
```

Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach to analysing data sets with the primary goal of summarising their main characteristics, often with the aid of graphical representations. The purpose of EDA is to understand the data, discover patterns, identify trends, and gain insights before applying more complex statistical techniques or building predictive models. It is typically one of the initial steps in the data analysis process.

1. Dataset Info():

The basic analysis on the dataset after cleaning is by doing *info()* and *describe()* methods on the dataframe. The observations after data cleaning are shown below.

```
In [253]: risk_df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 32406 entries, 1 to 32580
Data columns (total 25 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   person_age                            32406 non-null  int64
 1   person_income                         32406 non-null  float64
 2   person_home_ownership                 32406 non-null  object
 3   person_emp_length                     32406 non-null  float64
 4   loan_intent                           32406 non-null  object
 5   loan_grade                            32406 non-null  object
 6   loan_amnt                             32406 non-null  int64
 7   loan_int_rate                         32406 non-null  float64
 8   loan_status                           32406 non-null  string
 9   loan_percent_income                   32406 non-null  float64
 10  cb_person_default_on_file             32406 non-null  object
 11  cb_person_cred_hist_length            32406 non-null  int64
 12  age_range                             32391 non-null  category
 13  income_range                          32406 non-null  category
 14  loan_amount_range                     32406 non-null  category
 15  person_home_ownership_cat             32406 non-null  int8
 16  loan_intent_cat                       32406 non-null  int8
 17  loan_grade_cat                        32406 non-null  int8
 18  cb_person_default_on_file_cat         32406 non-null  int8
 19  age_range_cat                         32406 non-null  int8
 20  income_range_cat                      32406 non-null  int8
 21  loan_amount_range_cat                 32406 non-null  int8
 22  person_emp_length_normalized          32406 non-null  float64
 23  loan_int_rate_normalized              32406 non-null  float64
 24  cb_person_cred_hist_length_normalized  32406 non-null  float64
dtypes: category(3), float64(7), int64(3), int8(7), object(4), string(1)
memory usage: 4.3+ MB
```

By using the above command, we can get the basic details of the dataset like datatypes of the columns, whether there are any non null values etc.

2. Dataset describe():

Using describe() method, we get all the standard details of the features like mean, median, mode, counts and standard deviation values of all the features. This is demonstrated below

```
In [254]: risk_df.describe()
```

Out[254]:

	person_age	person_income	person_emp_length	loan_amnt	loan_int_rate	loan_percent_income	cb_person_cred_hist_length	person_home_ownership
count	32406.000000	3.240600e+04	32406.000000	32406.000000	32406.000000	32406.000000	32406.000000	32406.00(
mean	27.725421	6.589523e+04	4.760507	9592.772943	11.016800	0.170250	5.809387	1.67
std	6.185625	5.251945e+04	3.978768	6321.047865	3.083114	0.106786	4.053719	1.43
min	20.000000	4.000000e+03	0.000000	500.000000	5.420000	0.000000	2.000000	0.00(
25%	23.000000	3.850000e+04	2.000000	5000.000000	8.490000	0.090000	3.000000	0.00(
50%	26.000000	5.500000e+04	4.000000	8000.000000	11.011695	0.150000	4.000000	3.00(
75%	30.000000	7.920000e+04	7.000000	12250.000000	13.110000	0.230000	8.000000	3.00(
max	80.000000	2.039784e+06	38.000000	35000.000000	23.220000	0.830000	30.000000	3.00(

3. Correlation Matrix:

A correlation matrix is a representation of correlation coefficients between variables in a data set. It's commonly used to examine the relationships between variables.

We filtered the required features to check the correlation between them and plotted the correlation matrix as shown below.

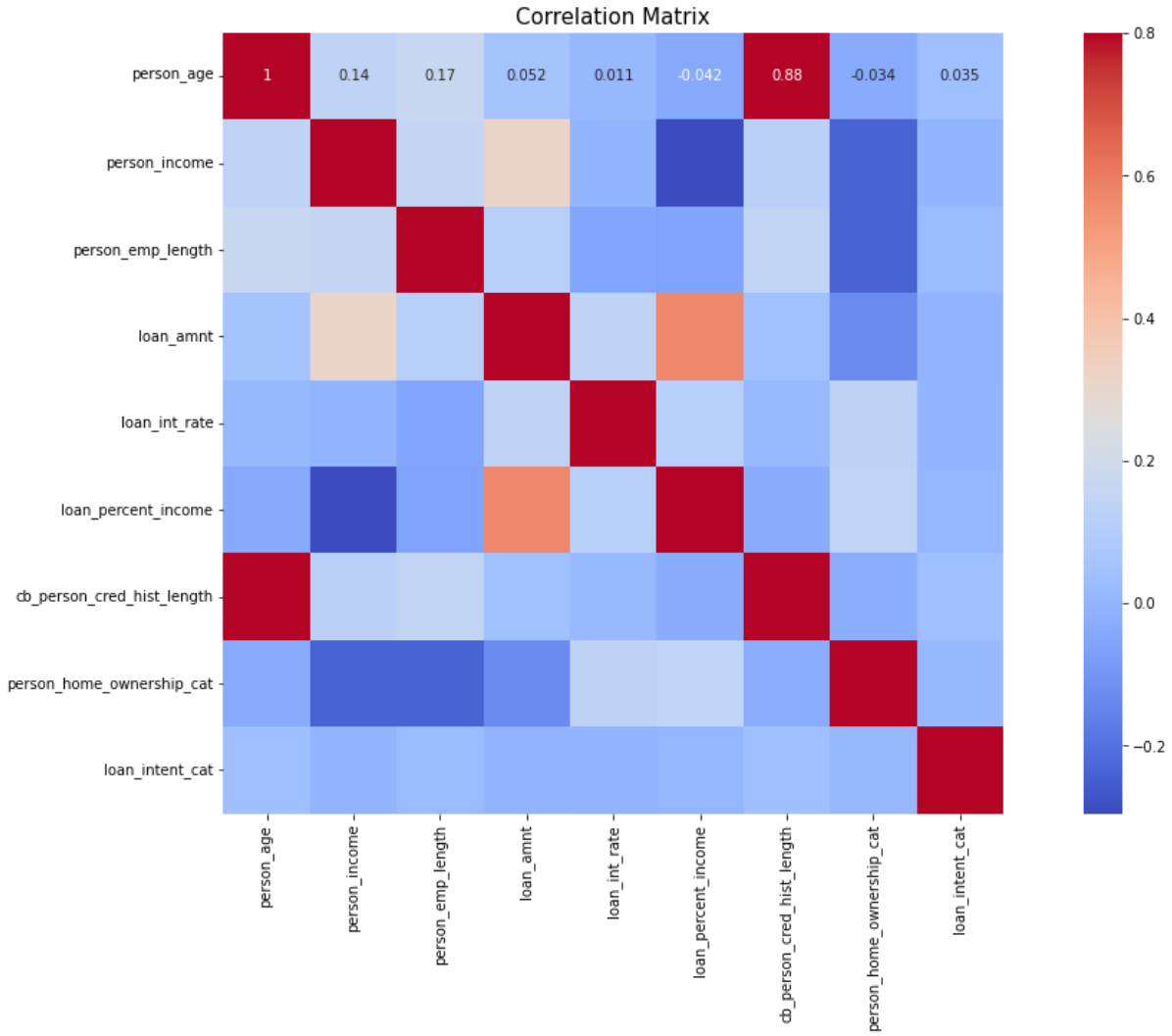
```
In [120]: feature_corelation_df = risk_df[['person_age', 'person_income', 'person_emp_length', 'loan_amnt', 'loan_int_rate', 'loa
```

```
In [121]: feature_corelation_df.corr()
```

Out[121]:

son_age	person_income	person_emp_length	loan_amnt	loan_int_rate	loan_percent_income	cb_person_cred_hist_length	person_home_ownership_cat	loan_intent_
1.000000	0.141254	0.169744	0.052262	0.011160	-0.041529	0.878580	-0.033913	0.035
0.141254	1.000000	0.155383	0.316719	-0.000942	-0.294759	0.122891	-0.235684	-0.001
0.169744	0.155383	1.000000	0.112324	-0.052811	-0.058191	0.147190	-0.235035	0.019
0.052262	0.316719	0.112324	1.000000	0.139567	0.572645	0.042408	-0.131233	-0.004
0.011160	-0.000942	-0.052811	0.139567	1.000000	0.114276	0.015327	0.132906	-0.001
0.041529	-0.294759	-0.058191	0.572645	0.114276	1.000000	-0.031170	0.141667	0.000
0.878580	0.122891	0.147190	0.042408	0.015327	-0.031170	1.000000	-0.025951	0.035
0.033913	-0.235684	-0.235035	-0.131233	0.132906	0.141667	-0.025951	1.000000	0.010
0.035238	-0.001390	0.019100	-0.004921	-0.001324	0.000812	0.035321	0.010509	1.000

```
In [126]: corelation_plot, axis = plt.subplots()
          corelation_plot.set_size_inches(20,10)
          sns.heatmap(feature_corelation_df.corr(), vmax =.8, square = True, annot = True,cmap='coolwarm' )
          plt.title('Correlation Matrix', fontsize=15);
```

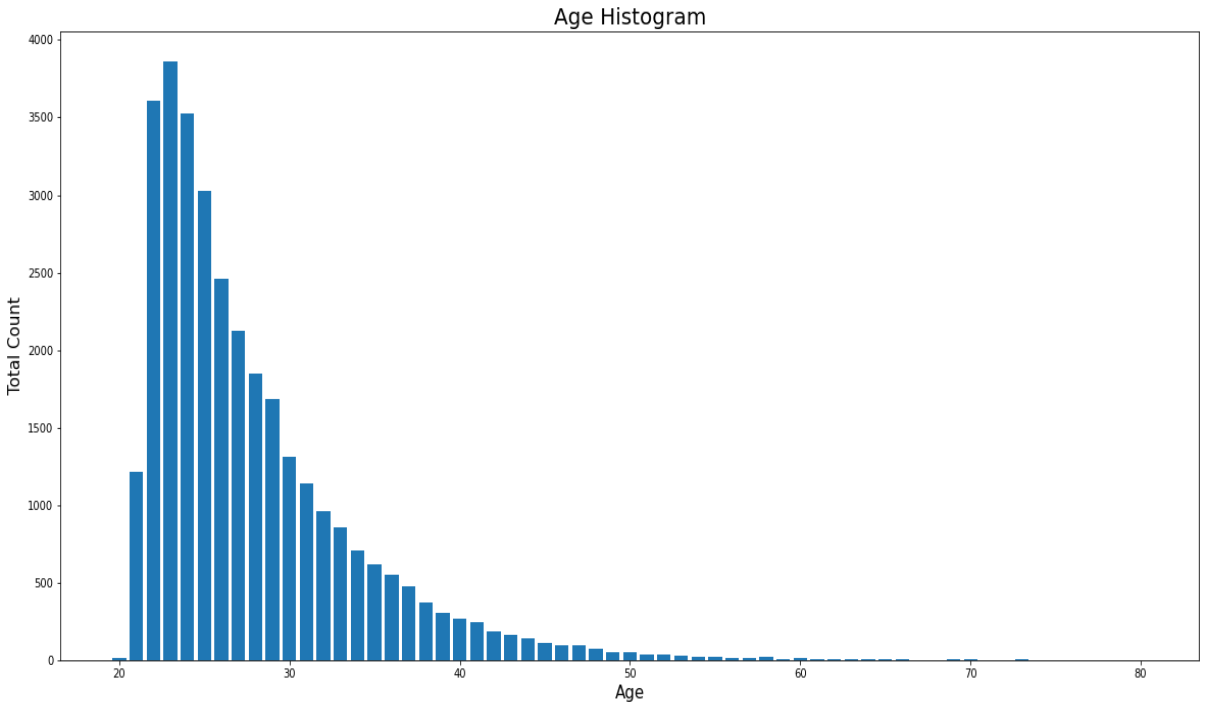


The above correlation matrix shows the relation between all the required features of the dataset, which inturn, helps in doing feature engineering.

4. Histogram of Person Age Feature :

The below analysis shows the histogram of the age of the customers, who are credit worthy and took loans

```
In [218]: age_count = risk_df['person_age'].value_counts().values
          age = risk_df['person_age'].value_counts().index
          fig = plt.figure(figsize = (20, 10))
          plt.bar(hori, verti)
          plt.title("Age Histogram", fontsize=20)
          plt.xlabel("Age", fontsize=15)
          plt.ylabel("Total Count", fontsize=15)
          plt.show()
```

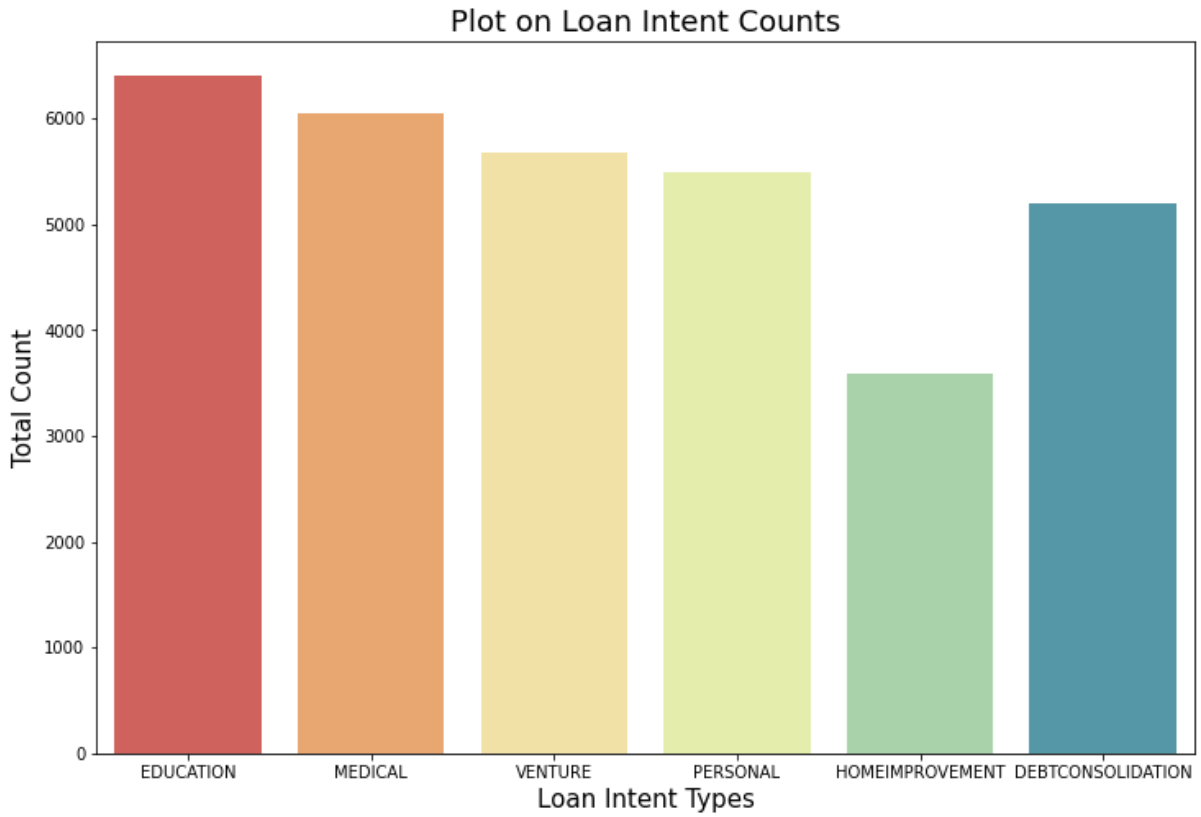


From the above histogram, we can observe that the age range of most of the customers is between 20 and 30. This shows that customers whose age is between 20 and 30 are the users who took most of the loans.

5. Loan Intent Bar Graph:

The below bar graph shows the total count of different loan intents, for which customers are taking the loan for. This information is helpful to figure out the importance of this feature in predicting whether customers defaults on the loan.

```
In [258]: plt.figure(figsize=(12,8))
          sns.countplot(x=risk_df['loan_intent'], palette = "Spectral")
          plt.title('Plot on Loan Intent Counts', fontsize=18)
          plt.xlabel('Loan Intent Types', fontsize=15)
          plt.xlabel('Total Count', fontsize=15)
          plt.show()
```

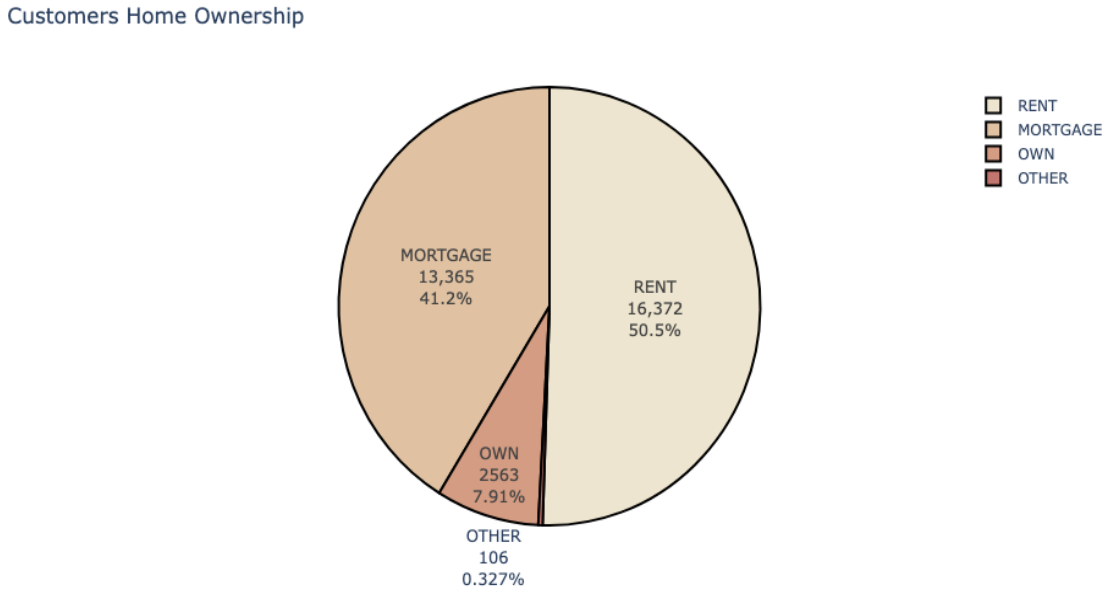


From the above graph, we can say, all these loan purposes equally contributed to the dataset, And this metric is relevant to predict, whether customer defaults on the loan or not.

6. Visualisation of Home ownership:

The below plot is to visualise the home ownership feature which is the most important feature for load default prediction. The analysis is done in pie chart as shown below.

```
In [264]: level_counts=risk_df.person_home_ownership.value_counts()
          fig=pl.pie(values=level_counts.values,
                     names=level_counts.index,
                     color_discrete_sequence=pl.colors.sequential.Brwnyl,
                     title= 'Customers Home Ownership'
                     )
          fig.update_traces(textinfo='label+percent+value', textfont_size=13,
                            marker=dict(line=dict(color='#102000', width=0.2)))

          fig.data[0].marker.line.width = 2
          fig.data[0].marker.line.color='black'
          fig.show()
```

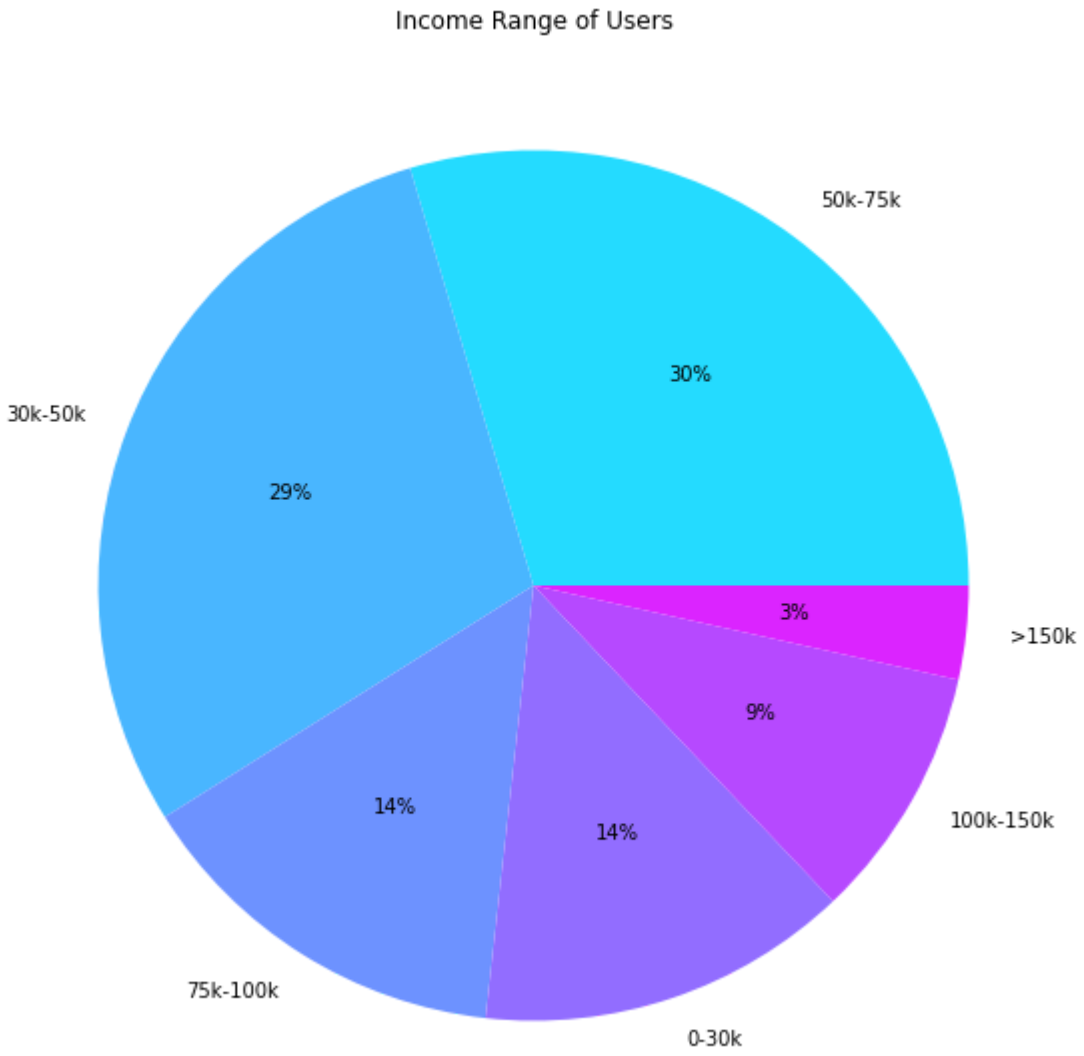


From the above plot, we can say that most of the customers around 50% are staying in rented houses. We can use this as a feature to predict the loan default rate.

7. Income Range Visualisation with Pie chart:

In the data cleaning and processing step, we've created a new feature which tells the income range of all these customers in the dataset. The below pie chart shows the percentage of total users in a particular income range.

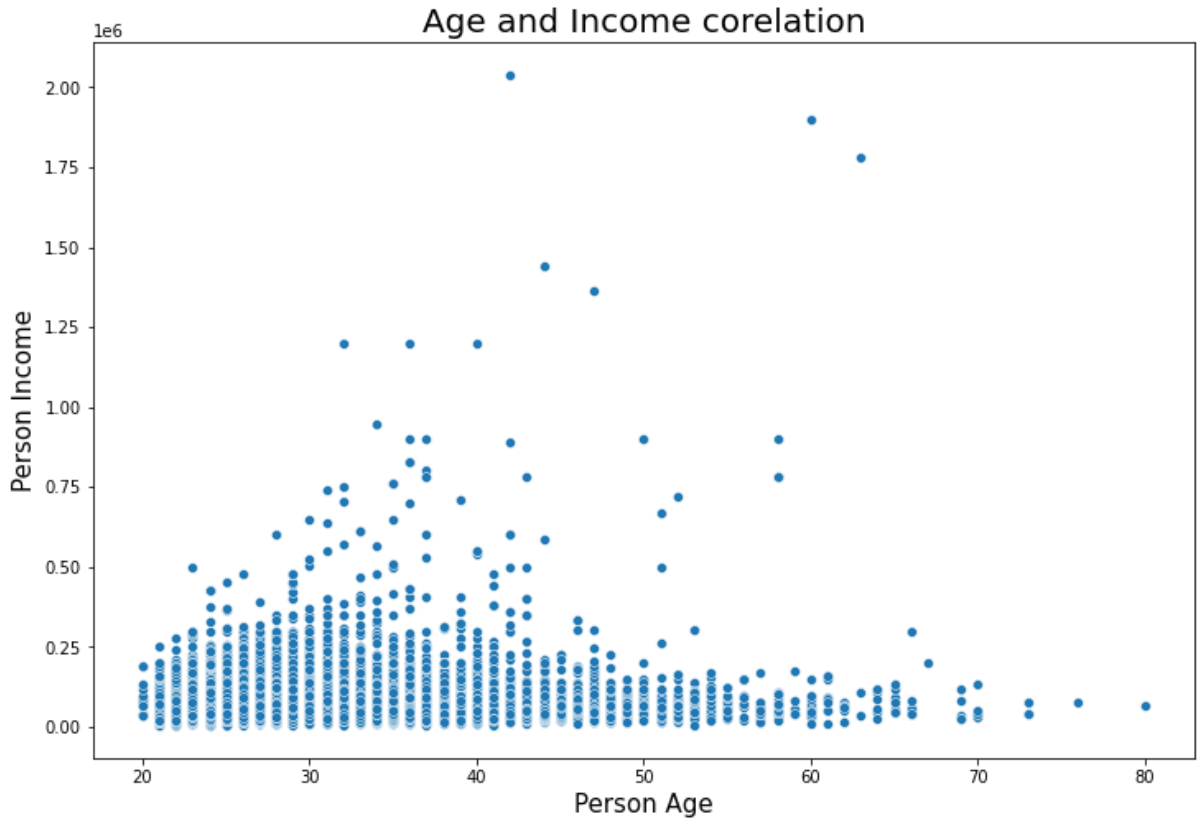
```
In [199]: income_range_counts=risk_df.income_range.value_counts()
          fig,ax=plt.subplots(1,1,figsize=(10,10))

          ax.set_title("Income Range of Users")
          ax.pie(income_range_counts.values, labels=income_range_counts.index, colors=sns.color_palette('cool'), autopct='%.0f%%'
          fig.show()
```



8. Age and Income Correlation scatter plot:

The below scatter plot shows the correlation between the age of the customers and the income of the customers, which are the most important features in predicting whether the customers defaults on the loans or not. The scatter plot shows, for every age point, what would be the income.

```
In [214]: plt.figure(figsize=(12,8))
          sns.scatterplot(data=risk_df, x="person_age", y="person_income")
          plt.title("Age and Income corelation", fontsize=20)
          plt.xlabel("Person Age", fontsize=15)
          plt.ylabel("Person Income", fontsize=15)
          plt.show()
```

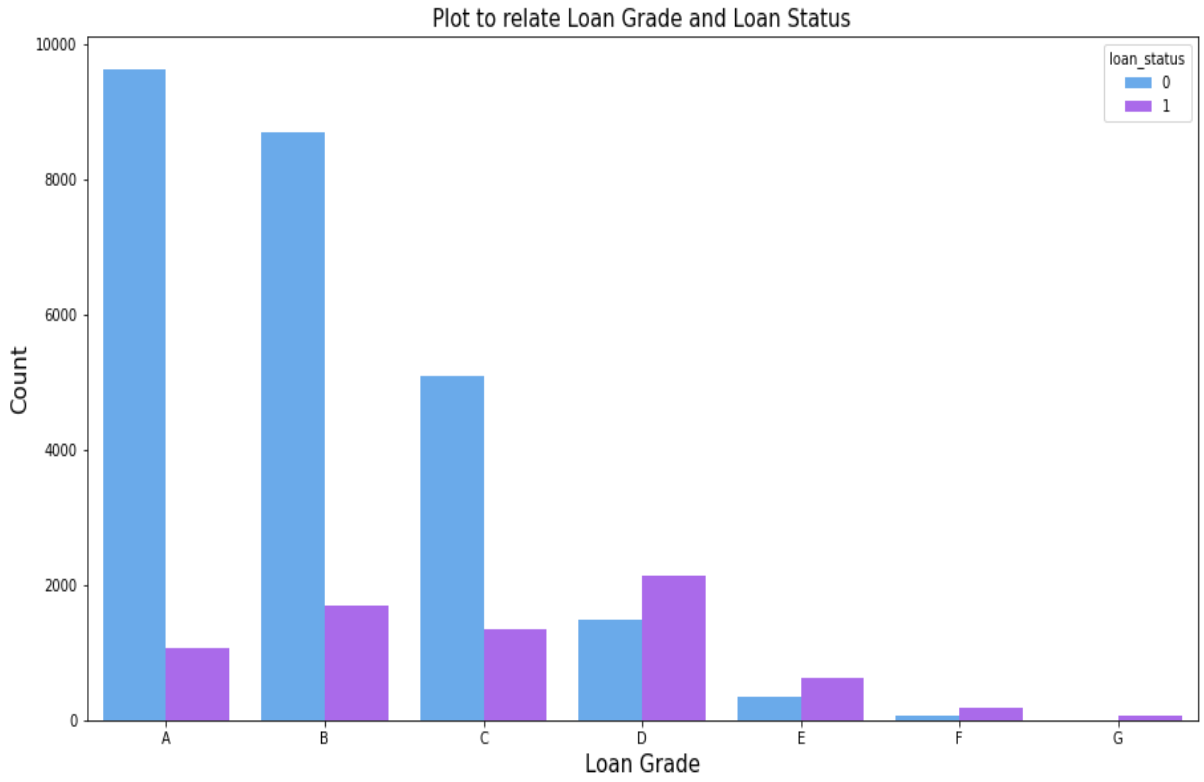


The above scatter plot shows all the points and we can observe that, most of the salary range is from 0 to 50k range.

9. Bar graph representing Loan grade and Loan status:

The below bar chart shows the relation between the loan grade, and the loan status. From this plot, we can analyse which types of loans are mostly prone to default and which are not.

```
In [260]: risk_df['loan_status'] = risk_df.loan_status.astype('string')
          plt.figure(figsize = (15,8))
          grade_counts = risk_df.groupby(['loan_grade', 'loan_status']).size().reset_index(name='count')
          grade_index = risk_df['loan_grade'].value_counts().sort_values(ascending=False).index
          sns.barplot(x='loan_grade', y='count', hue='loan_status', data=grade_counts, order=grade_index,  palette = "cool")
          plt.title('Plot to relate Loan Grade and Loan Status', fontsize=15)
          plt.xlabel('Loan Grade', fontsize=15)
          plt.ylabel('Count', fontsize=15)
          plt.show()
```

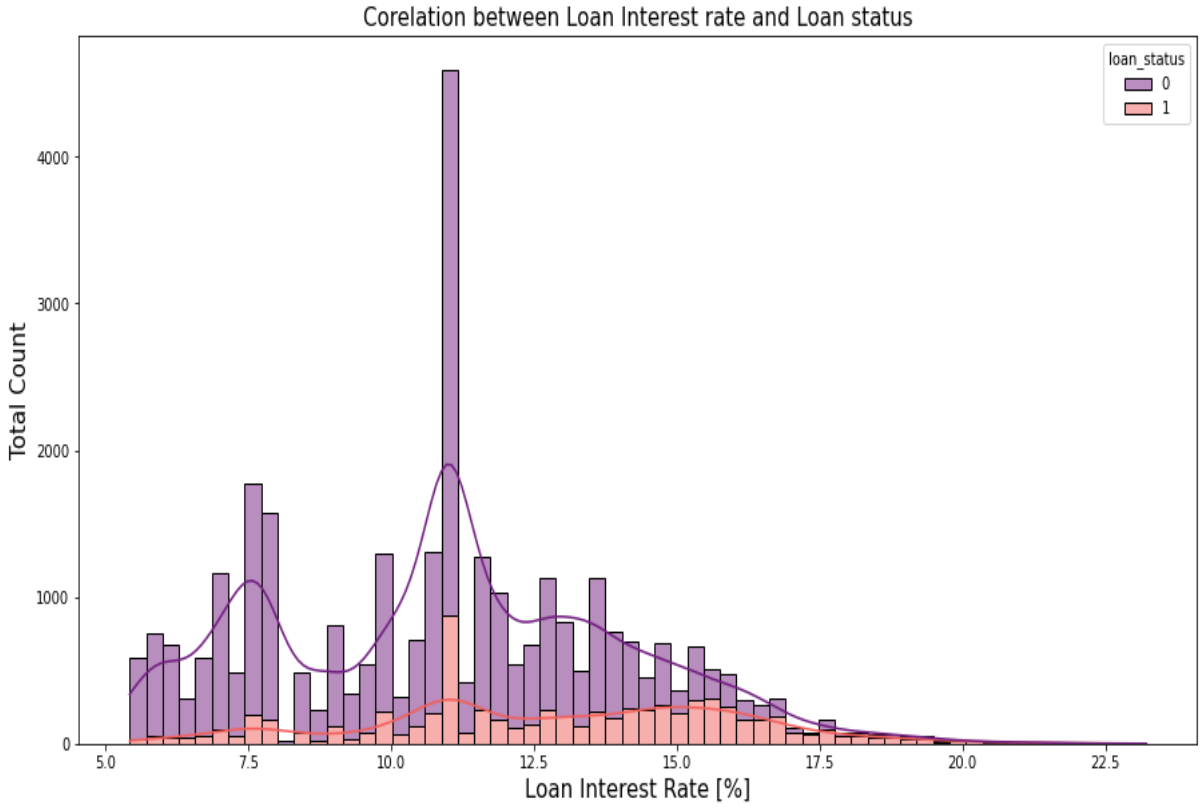


From the above plot, we can observe that, if the loan grades A,B and C the defaults are less than non defaults, and as the grade goes down, the default rate increases. That means, low grades are risky and prone to defaults.

10. Plot to relate Loan Interest Rate and loan status:

The below plot is to analyse the correlation between the loan interest rates and the loan status whether there is any relation between the loan interest rates and the loan status.

```
In [247]: plt.figure(figsize = (15,8))
          sns.histplot(data=risk_df, x='loan_int_rate', hue='loan_status', multiple='stack', kde=True, palette = "magma")
          plt.title('Corelation between Loan Interest rate and Loan status', fontsize=15)
          plt.xlabel('Loan Interest Rate [%]', fontsize=15)
          plt.ylabel('Total Count', fontsize=15)
          plt.show()
```

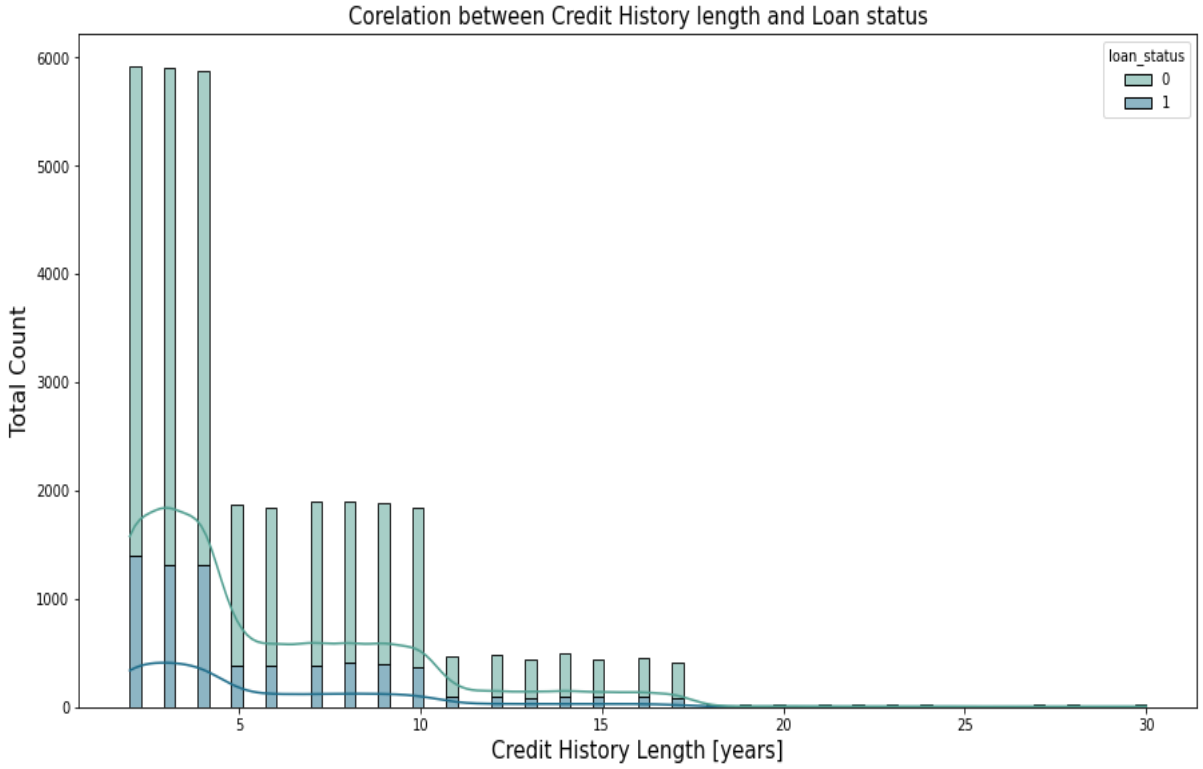


From the above plot, we can observe that, as the interest rates increase, the loan default rates are also increasing. And one more observation is that, there is a spike at an interest rate of around 11. This is also an important metric and correlation.

11. Plot to relate Credit History Length and loan status:

The below plot is to observe the correlation between the credit history length i.e, how long the customer is having the credit history. Ideally, if the credit history length is more, the loan default rate should be less. Below plot shows that.

```
In [248]: plt.figure(figsize = (15,8))
          sns.histplot(data=risk_df, x='cb_person_cred_hist_length', hue='loan_status', multiple='stack', kde=True, palette = "cr
          plt.title('Corelation between Credit History length and Loan status', fontsize=15)
          plt.xlabel('Credit History Length [years]', fontsize=15)
          plt.ylabel('Total Count', fontsize=15)
          plt.show()
```



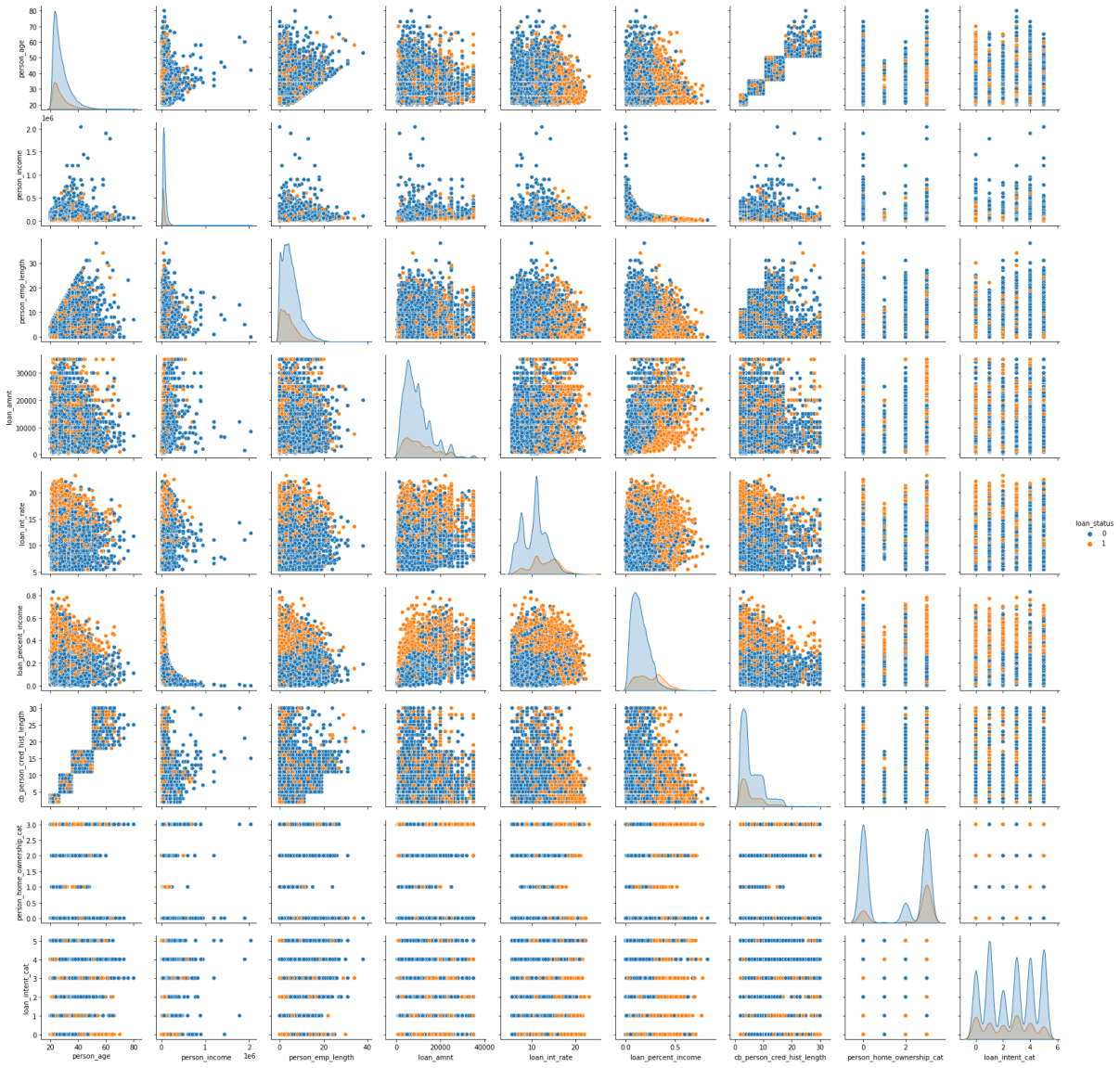
From the above plot, we can observe that most of the customers are in the range of 2-4 years of credit history length. And the default rates are decreasing as the credit history length increases.

12. Feature Pair Plot:

The feature plot shows that.the correlation between all the required features that are used in the prediction models, the relation whether between them using the pairplot functionality provided by seaborn library.

```
In [262]: feature_df = risk_df[['person_age', 'person_income', 'person_emp_length', 'loan_amnt', 'loan_int_rate', 'loan_percent_i
```

```
In [263]: sns.pairplot(feature_df, hue='loan_status')
```



The above pairplot shows for every feature, what is the impact of the loan default rates.