Data Analytics Project --> Bank Loan

- Understand the Business Problem
- Data Understanding
- Data Cleaning
- Data Analysis
- Presentation

Step-1 :- Bussiness Problem Understanding

Problem Statement:

- **In order to gain a comprehensive overview of our lending operations and monitor the performance of loans, we aim to create a grid view report categorized by Loan Status.**
- By providing insights into metrics such as
  - Total Loan Applications
  - Total Funded Amount
  - Total Amount Received
  - Month-to-Date (MTD) Funded Amount
  - MTD Amount Received
  - Average Interest Rate
  - Average Debt-to-Income Ratio (DTI)
- This Analytics Report will empower us to make data-driven decisions and assess the health of our loan portfolio.

Key Performance Indicators (KPIs) Requirements:

1. Total Loan Applications:

- We need to calculate the total number of loan applications received during a specified period. Additionally, it is essential to monitor the Month-to-Date(MTD) Loan Applications and track changes Month-over-Month (MoM).

2. Total Funded Amount:

- Understanding the total amount of funds disbursed as loans is crucial. We also want to keep an eye on the MTD Total Funded Amount and analyse the

Month-over-Month (MoM) changes in this metric.

3. Total Amount Received:

- Tracking the total amount received from borrowers is essential for assessing the bank's cash flow and loan repayment. We should analyse the Month-to-

Date (MTD) Total Amount Received and observe the Month-over-Month(MoM) changes.

4. Average Interest Rate:

- Calculating the average interest rate across all loans, MTD, and monitoring

the Month-over-Month (MoM) variations in interest rates will provide insights into our lending portfolio's overall cost.

5. Average Debt-to-Income Ratio (DTI):

- Evaluating the average DTI for our borrowers helps us gauge their financial

health. We need to compute the average DTI for all loans, MTD, and track Month-over-Month (MoM) fluctuations.

Good Loans:

- **1. Good Loan Application Percentage**
- **2. Good Loan Applications**
- **3. Good Loan Funded Amount**
- **4. Good Loan Total Received Amount**

Bad Loans:

- **5. Bad Loan Application Percentage**

- **6. Bad Loan Applications**
- **7. Bad Loan Funded Amount**
- **8. Bad Loan Total Received Amount**

Chart's Requirement:

1. Monthly Trends by Issue Date (Line Chart): To identify seasonality and long-term trends in lending activities

2. Regional Analysis by State : To identify regions with significant lending activity and assess regional disparities

3. Loan Term Analysis : To allow the client to understand the distribution of loans across various term lengths.

4. Employee Length Analysis: How lending metrics are distributed among borrowers with different employment lengths, helping us assess the impact of employment history on loan applications.

5. Loan Purpose Breakdown: Will provide a visual breakdown of loan metrics based on the stated purposes of loans, aiding in the understanding of the primary reasons borrowers seek financing.

6. Home Ownership Analysis : For a hierarchical view of how home ownership impacts loan applications and disbursements.

Step-2.1 : Load Data

In [8]:

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

import warnings
warnings.simplefilter("ignore")
```

In [9]:

```
df = pd.read_csv(r"D:\DATA SCIENCE\33. Projects\1. Data Analytics project\Bank Loan\financial_loan.csv")

df.head()
```

Out[9]:

	id	address_state	application_type	emp_length	emp_title	grade	home_ownership	issue_date	last_credit_pull_date	la
0	1077430	GA	INDIVIDUAL	< 1 year	Ryder	C	RENT	11-02-2021	13-09-2021	
1	1072053	CA	INDIVIDUAL	9 years	MKC Accounting	E	RENT	01-01-2021	14-12-2021	
2	1069243	CA	INDIVIDUAL	4 years	Chemat Technology Inc	C	RENT	05-01-2021	12-12-2021	
3	1041756	TX	INDIVIDUAL	< 1 year	barnes distribution	B	MORTGAGE	25-02-2021	12-12-2021	
4	1068350	IL	INDIVIDUAL	10+ years	J&J Steel Inc	A	MORTGAGE	01-01-2021	14-12-2021	

5 rows × 24 columns

Dropped Columns:

id & member_id → Removed because they are just unique identifiers.

next_payment_date → Removed because it’s future-oriented and not needed for past loan analysis.

Step-2.2 : Data Understanding

- We understand the each & every column name very clearly (do research)
- understand the dataset by applying info(), shape, dtypes, columns
- list the continous, discrete categorial, discrete count
- Observe the data

In [12]:

```
df.columns
```

Out[12]:
```
Index(['id', 'address_state', 'application_type', 'emp_length', 'emp_title',
       'grade', 'home_ownership', 'issue_date', 'last_credit_pull_date',
       'last_payment_date', 'loan_status', 'next_payment_date', 'member_id',
       'purpose', 'sub_grade', 'term', 'verification_status', 'annual_income',
       'dti', 'installment', 'int_rate', 'loan_amount', 'total_acc',
       'total_payment'],
      dtype='object')
```

In [13]:
```
df["loan_status"].value_counts()
```

Out[13]:
```
loan_status
Fully Paid     32145
Charged Off     5333
Current         1098
Name: count, dtype: int64
```

- no wrong space in column name

Understanding of columns

1) id --> Unique identifier for each loan application.

2)address_state --> The state where the borrower resides.

3)application_type --> Type of loan application (Individual or Joint), Determines whether a loan is issued to a single borrower or multiple borrowers (which can impact risk).

4)emp_length --> Length of employment (e.g., <1 year, 10+ years), Longer employment indicates financial stability, reducing default risk.

5)emp_title --> The job title of the borrower, Can help analyze loan trends for different professions

6)grade --> A credit rating assigned to a borrower (A to G, where A is the best), Used to assess borrower risk—higher grades indicate lower risk.

7)home_ownership --> It indicates the housing situation of the borrower at the time of applying for a loan. It provides insights into whether the borrower owns, rents, or has a mortgage on their home.

```
         -  RENT –> The borrower is renting their home.
         -  MORTGAGE –> The borrower owns the home but has an active mortgage (i.e., still
paying off a loan on the property).
         -  OWN –> The borrower fully owns the home with no mortgage debt.
         -  OTHER –> Any homeownership status that does not fall into the above categories.
         -  NONE –> The borrower has no homeownership status, possibly indicating homelessness
or unconventional living situations.
```

8)issue_date --> The date when the loan was issued, Helps analyze monthly loan trends and seasonal borrowing behavior.

9)last_credit_pull_date --> The most recent date a credit check was done on the borrower, Helps track creditworthiness changes over time.

10)last_payment_date --> The last date when the borrower made a payment, Helps track repayment patterns and loan status (delinquency, defaults).

11)loan_status --> Describes the current state of the loan.

```
- Fully Paid  --> Loan is successfully repaid.
- Charged Off --> Loan is defaulted and written off.
- Current     --> Loan is still active and being repaid.
```

12)next_payment_date --> The scheduled date for the borrower’s next loan payment.

13)member_id --> A unique identifier for the borrower (if applicable), Helps track multiple loans by the same borrower.

14)purpose --> The reason why the borrower is taking the loan

15)sub_grade --> Each grade is further divided into sub-grades from 1 to 5 (e.g., A1, A2, A3, A4, A5), A1 is the best (lowest risk, lowest interest rate), and G5 is the worst (highest risk, highest interest rate).

- Sub-grades help lenders determine loan risk and pricing more accurately. Borrowers with better sub-grades get lower interest rates, while higher-risk borrowers get higher rates or may even be denied a loan.

16)term --> he duration of the loan repayment (typically in months).

```
Usefulness:
  - Shorter terms generally mean higher monthly payments but less interest paid overall.
  - Longer terms may be riskier due to extended financial obligations.
```

17)verification_status --> Indicates whether the borrower's income and employment details were verified.

Verification Status	Explanation	Risk Level
Not Verified	The borrower's income and employment details were not independently verified by the lender.	**High Risk**
Source Verified	Some documents (such as pay stubs or tax returns) were checked, but full verification was not completed.	**Medium Risk**
Verified	The borrower's income, employment, and financial details were fully verified through official documents.	**Low Risk**

18)annual_income --> The borrower's self-reported annual income.

```
Usefulness:
   - Used to calculate Debt-to-Income (DTI) Ratio.
   - Helps assess repayment capacity.
```

19)dti -->Measures a borrower's total monthly debt payments relative to their income

- Debt-to-Income (DTI) Ratio Formula, **DTI = (Total Monthly Debt Payments / Monthly Gross Income) * 100.**
- High DTI (>40%) means borrower has high financial obligations, increasing risk.
- Low DTI (<30%) suggests better financial stability.

20)installment -->The fixed monthly payment for the loan.

- Installment = [Loan Amount+Interest] / Loan Term (in months)

```
Usefulness:
   - Helps in cash flow analysis for borrowers and lenders.
```

21)int_rate --> The percentage of interest charged on the loan amount.

```
Usefulness:
   - Higher interest rates indicate higher lending risk.
   - Used for loan pricing and profitability analysis.
```

22)loan_amount --> The total amount borrowed by the applicant.

```
Usefulness:
   - Helps in analyzing average loan sizes across different borrower groups.
```

23)total_acc --> the total number of credit accounts a borrower has across all financial institutions. This includes:Credit cards, Personal loans,Mortgages, Auto loans, Student loans, Retail store credit accounts.

Risk Category	`total_acc` Range	Interpretation
High Risk	`1 - 5` accounts	Limited credit history; may struggle to manage loans.
Medium Risk	`6 - 20` accounts	Established credit history; moderate financial stability.
Low Risk	`21+` accounts	Long credit history, good account management, and financial stability.

```
Usefulness:
     - Helps assess credit history and experience with debt.
     - A higher number of accounts may indicate good financial management or high debt exposure.
```

24)total_payment --> The total amount repaid by the borrower (including interest).

```
Usefulness:
    - Helps assess loan performance.
    - Used in revenue tracking for lenders.
```

In [16]:

```
df.columns.tolist()
```

Out[16]:
```
['id',
 'address_state',
 'application_type',
 'emp_length',
 'emp_title',
 'grade',
 'home_ownership',
 'issue_date',
 'last_credit_pull_date',
 'last_payment_date',
 'loan_status',
 'next_payment_date',
 'member_id',
 'purpose',
 'sub_grade',
 'term',
 'verification_status',
 'annual_income',
 'dti',
 'installment',
 'int_rate',
 'loan_amount',
 'total_acc',
 'total_payment']
```

In [17]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38576 entries, 0 to 38575
Data columns (total 24 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   id                     38576 non-null  int64  
 1   address_state          38576 non-null  object 
 2   application_type       38576 non-null  object 
 3   emp_length             38576 non-null  object 
 4   emp_title              37138 non-null  object 
 5   grade                  38576 non-null  object 
 6   home_ownership         38576 non-null  object 
 7   issue_date             38576 non-null  object 
 8   last_credit_pull_date  38576 non-null  object 
 9   last_payment_date      38576 non-null  object 
 10  loan_status            38576 non-null  object 
 11  next_payment_date      38576 non-null  object 
 12  member_id              38576 non-null  int64  
 13  purpose                38576 non-null  object 
 14  sub_grade              38576 non-null  object 
 15  term                   38576 non-null  object 
 16  verification_status    38576 non-null  object 
 17  annual_income          38576 non-null  float64
 18  dti                    38576 non-null  float64
 19  installment            38576 non-null  float64
 20  int_rate               38576 non-null  float64
 21  loan_amount            38576 non-null  int64  
 22  total_acc              38576 non-null  int64  
 23  total_payment          38576 non-null  int64  
dtypes: float64(4), int64(5), object(15)
memory usage: 7.1+ MB
```

Step-2.3 : Data Exploration

In [19]:
```
continuous = ["annual_income","dti","installment","int_rate","loan_amount","total_payment"]

discrete_count = ["emp_length","total_acc"]

discrete_categorical = ["address_state","application_type","emp_title","grade","home_ownership","loan_status","

time_series = ["issue_date","last_credit_pull_date","last_payment_date","next_payment_date"]

unique = ["id", "member_id"]
```

In [20]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38576 entries, 0 to 38575
Data columns (total 24 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    38576 non-null  int64  
 1   address_state         38576 non-null  object 
 2   application_type      38576 non-null  object 
 3   emp_length            38576 non-null  object 
 4   emp_title             37138 non-null  object 
 5   grade                 38576 non-null  object 
 6   home_ownership        38576 non-null  object 
 7   issue_date            38576 non-null  object 
 8   last_credit_pull_date  38576 non-null  object 
 9   last_payment_date     38576 non-null  object 
 10  loan_status           38576 non-null  object 
 11  next_payment_date     38576 non-null  object 
 12  member_id             38576 non-null  int64  
 13  purpose               38576 non-null  object 
 14  sub_grade             38576 non-null  object 
 15  term                  38576 non-null  object 
 16  verification_status   38576 non-null  object 
 17  annual_income         38576 non-null  float64
 18  dti                   38576 non-null  float64
 19  installment           38576 non-null  float64
 20  int_rate              38576 non-null  float64
 21  loan_amount           38576 non-null  int64  
 22  total_acc             38576 non-null  int64  
 23  total_payment         38576 non-null  int64  
dtypes: float64(4), int64(5), object(15)
memory usage: 7.1+ MB
```

In [21]:
```
# Continuous   --> Float or int
# Count        --> int
# Categorical  --> object


# Wrong data types   --> [3,21,23]
# timeseries --> 7,8,9,11,
```

In [22]:
```
df[continuous].describe()
```

Out[22]:

	annual_income	dti	installment	int_rate	loan_amount	total_payment
count	3.857600e+04	38576.000000	38576.000000	38576.000000	38576.000000	38576.000000
mean	6.964454e+04	0.133274	326.862965	0.120488	11296.066855	12263.348533
std	6.429368e+04	0.066662	209.092000	0.037164	7460.746022	9051.104777
min	4.000000e+03	0.000000	15.690000	0.054200	500.000000	34.000000
25%	4.150000e+04	0.082100	168.450000	0.093200	5500.000000	5633.000000
50%	6.000000e+04	0.134200	283.045000	0.118600	10000.000000	10042.000000
75%	8.320050e+04	0.185900	434.442500	0.145900	15000.000000	16658.000000
max	6.000000e+06	0.299900	1305.190000	0.245900	35000.000000	58564.000000

In [23]:
```
df[discrete_count].describe()
```

Out[23]:

	total_acc
count	38576.000000
mean	22.132544
std	11.392282
min	2.000000
25%	14.000000
50%	20.000000
75%	29.000000
max	90.000000

In [24]:
```
df[discrete_categorical].describe()
```

Out[24]:

	address_state	application_type	emp_title	grade	home_ownership	loan_status	purpose	sub_grade	term	verificat
count	38576	38576	37138	38576	38576	38576	38576	38576	38576	
unique	50	1	28525	7	5	3	14	35	2	
top	CA	INDIVIDUAL	US Army	B	RENT	Fully Paid	Debt consolidation	B3	36 months	N
freq	6894	38576	135	11674	18439	32145	18214	2834	28237	

In [25]:
```
df["emp_length"].unique()
```

Out[25]:
```
array(['< 1 year', '9 years', '4 years', '10+ years', '3 years',
       '5 years', '1 year', '6 years', '2 years', '7 years', '8 years'],
      dtype=object)
```

In [26]:
```
df["emp_length"].value_counts()
```

Out[26]:
```
emp_length
10+ years    8870
< 1 year     4575
2 years      4382
3 years      4088
4 years      3428
5 years      3273
1 year       3229
6 years      2228
7 years      1772
8 years      1476
9 years      1255
Name: count, dtype: int64
```

In [27]:
```
df["total_acc"].unique()
```

Out[27]:
```
array([ 4, 11,  9, 28, 30, 23, 31, 21, 33, 13,  3, 15, 18, 14,  8,  7, 20,
       39, 24, 10, 19, 27,  6, 16, 45, 25,  5, 43, 29, 22, 41, 35, 44, 36,
       17, 26, 37, 32, 47, 52, 42, 46, 12, 50, 34, 59, 38, 63, 49, 48, 61,
       51, 55, 40, 53, 62, 58, 67, 54, 57, 56, 70,  2, 64, 60, 80, 79, 71,
       66, 65, 69, 90, 68, 74, 75, 87, 78, 72, 77, 81, 76, 73],
      dtype=int64)
```

In [28]:
```
df["total_acc"].value_counts()
```

Out[28]:
```
total_acc
16    1435
15    1420
17    1408
14    1405
20    1398
      ...
68       1
90       1
69       1
71       1
73       1
Name: count, Length: 82, dtype: int64
```

In [29]:
```
df["address_state"].unique()
```

Out[29]:
```
array(['GA', 'CA', 'TX', 'IL', 'PA', 'FL', 'MI', 'RI', 'NY', 'MD', 'WI',
       'NV', 'UT', 'WA', 'NH', 'HI', 'MA', 'OK', 'NJ', 'OH', 'AZ', 'CT',
       'MN', 'CO', 'TN', 'VA', 'MO', 'DE', 'NM', 'LA', 'AR', 'KY', 'NC',
       'SC', 'WV', 'KS', 'WY', 'OR', 'AL', 'VT', 'MS', 'DC', 'MT', 'SD',
       'AK', 'IN', 'ME', 'ID', 'NE', 'IA'], dtype=object)
```

In [30]:
```
df["address_state"].value_counts()
```

Out[30]:
```
address_state
CA    6894
NY    3701
FL    2773
TX    2664
NJ    1822
IL    1486
PA    1482
VA    1375
GA    1355
MA    1310
OH    1188
MD    1027
AZ     833
WA     805
CO     770
NC     759
CT     730
MI     685
MO     660
MN     592
NV     482
SC     464
WI     446
OR     436
AL     432
LA     426
KY     320
OK     293
KS     260
UT     252
AR     236
DC     214
RI     196
NM     183
HI     170
WV     167
NH     161
DE     110
WY      79
MT      79
AK      78
SD      63
VT      54
MS      19
TN      17
IN       9
ID       6
NE       5
IA       5
ME       3
Name: count, dtype: int64
```

In [31]:
```
df["application_type"].unique()
```

Out[31]:
```
array(['INDIVIDUAL'], dtype=object)
```

In [32]:
```
df["application_type"].value_counts()
```

Out[32]:
```
application_type
INDIVIDUAL    38576
Name: count, dtype: int64
```

In [33]:
```
df["emp_title"].unique()
```

Out[33]:
```
array(['Ryder', 'MKC Accounting', 'Chemat Technology Inc', ...,
       'Anaheim Regional Medical Center', 'Brooklyn Radiology',
       'Allen Edmonds'], dtype=object)
```

In [34]:
```
df["emp_title"].nunique()
```

Out[34]:
```
28525
```

In [35]:
```
df["emp_title"].value_counts()
```

Out[35]:
```
emp_title
US Army                             135
Bank of America                     109
IBM                                  67
AT&T                                 63
Wells Fargo                          57
                                   ...
Emeril's Delmonico's                  1
The Shafer Law Group                  1
U.S navy                              1
Wellspring Healthcare Services        1
Allen Edmonds                         1
Name: count, Length: 28525, dtype: int64
```

In [36]:
```
df["grade"].unique()
```

Out[36]:
```
array(['C', 'E', 'B', 'A', 'D', 'F', 'G'], dtype=object)
```

In [37]:
```
df["grade"].value_counts()
```

Out[37]:
```
grade
B    11674
A     9689
C     7904
D     5182
E     2786
F     1028
G      313
Name: count, dtype: int64
```

In [38]:
```
df["home_ownership"].unique()
```

Out[38]:
```
array(['RENT', 'MORTGAGE', 'OWN', 'OTHER', 'NONE'], dtype=object)
```

In [39]:
```
df["home_ownership"].value_counts()
```

Out[39]:
```
home_ownership
RENT        18439
MORTGAGE    17198
OWN          2838
OTHER          98
NONE            3
Name: count, dtype: int64
```

In [40]:
```
df["loan_status"].unique()
```

Out[40]:
```
array(['Charged Off', 'Fully Paid', 'Current'], dtype=object)
```

In [41]:
```
df["loan_status"].value_counts()
```

Out[41]:
```
loan_status
Fully Paid     32145
Charged Off     5333
Current         1098
Name: count, dtype: int64
```

In [42]:
```
df["purpose"].unique()
```

Out[42]:
```
array(['car', 'credit card', 'Debt consolidation', 'educational',
       'home improvement', 'house', 'major purchase', 'medical', 'moving',
       'other', 'renewable_energy', 'small business', 'vacation',
       'wedding'], dtype=object)
```

In [43]:
```
df["purpose"].value_counts()
```

Out[43]:
```
purpose
Debt consolidation    18214
credit card            4998
other                  3824
home improvement       2876
major purchase         2110
small business         1776
car                    1497
wedding                 928
medical                 667
moving                  559
house                   366
vacation                352
educational             315
renewable_energy         94
Name: count, dtype: int64
```

In [44]:
```
df["sub_grade"].unique()
```

Out[44]:
```
array(['C4', 'E1', 'C5', 'B2', 'A1', 'C3', 'C2', 'A4', 'A5', 'B5', 'B4',
       'B3', 'B1', 'D1', 'A2', 'A3', 'D4', 'D2', 'C1', 'D3', 'E3', 'F1',
       'E2', 'E5', 'D5', 'E4', 'F2', 'G3', 'F3', 'G1', 'F4', 'G4', 'G2',
       'F5', 'G5'], dtype=object)
```

In [45]:
```
df["sub_grade"].value_counts()
```

Out[45]:
```
sub_grade
B3    2834
A4    2803
A5    2654
B5    2644
B4    2455
C1    2089
B2    1990
C2    1972
B1    1751
A3    1740
C3    1490
A2    1440
D2    1314
C4    1202
C5    1151
D3    1144
A1    1052
D4     960
D1     913
D5     851
E1     750
E2     640
E3     538
E4     448
E5     410
F1     325
F2     243
F3     182
F4     163
F5     115
G1     101
G2      78
G4      56
G3      48
G5      30
Name: count, dtype: int64
```

In [46]:
```
df["term"].unique()
```

Out[46]:
```
array([' 60 months', ' 36 months'], dtype=object)
```

In [47]:
```
df["term"].value_counts()
```

Out[47]:
```
term
36 months    28237
60 months    10339
Name: count, dtype: int64
```

In [48]:
```
df["verification_status"].unique()
```

Out[48]:
```
array(['Source Verified', 'Not Verified', 'Verified'], dtype=object)
```

In [49]:
```
df["verification_status"].value_counts()
```

Out[49]:
```
verification_status
Not Verified       16464
Verified           12335
Source Verified     9777
Name: count, dtype: int64
```

Step-3 : Data Preprocessing

(i) Data Cleaning

- **1) Drop Unique/unnecessary Columns**
- Dropped "id", "member_id" columns because they are unique so cannot find insights through those columns
- Dropped "emp_title" column because it has too many unique job titles, difficult to analyze

In [52]:
```
# df = df.drop(columns = ["id","member_id","emp_title"])
# df.head()
```

- **2) Convert Wrong Data**

In [54]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38576 entries, 0 to 38575
Data columns (total 24 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   id                     38576 non-null  int64
 1   address_state          38576 non-null  object
 2   application_type       38576 non-null  object
 3   emp_length             38576 non-null  object
 4   emp_title              37138 non-null  object
 5   grade                  38576 non-null  object
 6   home_ownership         38576 non-null  object
 7   issue_date             38576 non-null  object
 8   last_credit_pull_date  38576 non-null  object
 9   last_payment_date      38576 non-null  object
 10  loan_status            38576 non-null  object
 11  next_payment_date      38576 non-null  object
 12  member_id              38576 non-null  int64
 13  purpose                38576 non-null  object
 14  sub_grade              38576 non-null  object
 15  term                   38576 non-null  object
 16  verification_status    38576 non-null  object
 17  annual_income          38576 non-null  float64
 18  dti                    38576 non-null  float64
 19  installment            38576 non-null  float64
 20  int_rate               38576 non-null  float64
 21  loan_amount            38576 non-null  int64
 22  total_acc              38576 non-null  int64
 23  total_payment          38576 non-null  int64
dtypes: float64(4), int64(5), object(15)
memory usage: 7.1+ MB
```

In [55]:
```
# Continuous   --> Float or int
# Count        --> int
# Categorical  --> object


# Wrong data types   --> [3,21,23]
# timeseries --> 7,8,9,11,
```

In [56]:
```
df["emp_length"].unique()
```

Out[56]:
```
array(['< 1 year', '9 years', '4 years', '10+ years', '3 years',
       '5 years', '1 year', '6 years', '2 years', '7 years', '8 years'],
      dtype=object)
```

In [57]:
```
df["emp_length"] = df["emp_length"].replace({"< 1 year" : 0, "9 years" : 9, "4 years" : 4, "10+ years" : 10,
                                  "3 years" : 3, "5 years" : 5, "1 year" : 1, "6 years" : 6,
                                  "2 years" : 2, "7 years" : 7, "8 years" : 8})
```

In [58]:
```
df["emp_length"].unique()
```

Out[58]:
```
array([ 0,  9,  4, 10,  3,  5,  1,  6,  2,  7,  8], dtype=int64)
```

- **3) Convert wrong data types**

In [60]:
```
df["total_payment"].unique()
```

Out[60]:
```
array([ 1009,  3939,  3522, ..., 31870, 35721, 33677], dtype=int64)
```

In [61]:
```
df["loan_amount"].unique()
```

Out[61]:
```
array([ 2500,  3000, 12000,  4500,  3500,  8000,  6000,  5500, 24000,
        4125,  5400, 11200,  5000,  3050, 10000,  2225,  4000,  7000,
        9000,  4800,  6300,  4750,  1850,  4200,  7200,  2400,  7500,
        5550, 22000,  3200, 11000,  4400,  8500,  2000,  7400,  5650,
        1800,  6500, 15000,  8700,  5600,  4600,  3800, 16000,  1300,
        7800,  5900,  3600,  2100,  4975,  1925,  1500,  7750,  9600,
        3900, 12975,  5950,  5100,  5200,  1200,  4650,  1450,  3250,
        3300,  1700,  5525, 18000,  1750,  5375,  9500,  7600,  6400,
        9900,  1000, 10400, 23500, 22600, 23600, 13100,  5800, 10800,
        1900,  8400,  3075,  6200, 11500,  4350,  4150,  4900,  6125,
        2425,  1600,  7100,  8900, 14000, 12250,  3700, 17000,  2550,
        6250, 14400,  8200,  9250,  3375,  1675,  8600,  2800,  3525,
        8800,  2250,  4375,  1275,  5050, 25000,  9800,  6600,  8250,
```

```
 2825,  5975,  3350, 20000, 19000,  2200, 14750,  9575, 13250,
 2350, 10625,  1400, 12800, 16800,  5750,  8975,  5275,  5850,
 2275, 14800,  5300, 20400, 16500,  2950,  6800,  6775,  9925,
 2475,  7275, 17500,  9100,  2650, 10500, 22800,  6900, 21000,
14900,  2875,  3825,  7875,  4625, 11900,  2900,  4250, 13000,
10200, 15950,  8100, 13200,  6100, 16675, 12600,  6075,  9175,
 5150,  5625, 12500,  8650,  8750,  7250,  5575,  4700,  2300,
 5700, 28000,  3100, 13500, 14500, 15850, 15600, 12325,  4950,
15325,  7575, 17600, 10750,  6275,  9975,  6700, 16600, 10600,
19200, 16200, 21600, 15500, 12450, 18500, 13800, 30000,  4550,
10550,  9350,  6350, 32000,  5125, 12400, 25600, 20975, 11800,
15300,  8850, 11100, 13475, 11625, 19600,  4475, 26800, 11300,
 4100, 11700, 26000, 22500, 24500, 21400, 35000, 22400, 14575,
 7125, 10375, 20050, 24925, 12375,  7150, 17250, 13225, 11775,
16400, 10075, 20125,  6950, 23000,  4050,  1950, 21500,  9200,
 4325, 13575,  6625, 16250,  8950, 14100, 19750,  7650,  3675,
 9525, 19950,  5875,  3975, 24625,  2675, 14275, 14300, 15450,
 9300,   750,  6225, 17950, 12750, 13975, 16075, 10675, 18650,
 7900, 15875, 10475,  2325, 19150,  2700,  9075,  3125,  6575,
11050,  3150,  3625,  9400, 10250,  7475,  3425, 20800, 15200,
10050, 18300, 13950,  9750, 23450, 14950, 10175,  8300, 16750,
18800,  1250, 19500, 10650,  8550, 15250, 10700, 17525, 21125,
 8150, 11600,  9700, 13600, 20500,  8575, 12900, 14700, 12700,
 2750,  7775, 12100, 10825,   725, 10950, 11250,  3725,  9050,
 1375,  4450,  3400,  6650,  6925,  1475, 10150,  6475,  4575,
17400,  2375,  6150,  2050,  2725, 18200,  9150,  7050, 18600,
 9425,  3850,  4850,  8450,  7925,  9450, 10725,  2600,  3750,
13300, 11550, 13650,  4300,  8350,  7950,  1150,  5925, 12350,
27250, 13750,  7350,  8675,  7700, 25850,  1050,  2850, 17325,
14550,  1425,  6975, 12775, 12025, 11400, 18150, 11075, 18250,
16775,  7850,  8325, 25975, 31000,  1125,  9875,  6550,  8775,
 2450,  6750,  9950,  4025,  6850,  1350, 11450, 10300, 17700,
11975, 18225,  4675,  4925, 19400, 17475,  6025, 15700, 21250,
 2150, 11525, 23750, 17200, 22950, 17750,  5475, 14600,  1875,
19650, 13275,  7450, 19550,  7675, 15175,  9275, 23100, 16300,
10875, 13700, 23700, 22100, 12200, 12875, 16700, 10850,  5250,
13125,  5225,  9125, 16950, 12725, 26375, 27400,  4775, 34000,
17800, 33250, 17050, 19800, 18400, 23325, 25475, 12300, 29000,
24375, 27500,  8175, 18550, 14250, 16875,  7300, 27600,  3450,
 6325,  3650, 15050,  5825, 24250, 17625, 20900, 15625, 15400,
19125, 14475,  8050, 10225,  8875,  1825, 23575, 30600,  8125,
13400, 12550,  9550, 15350, 33500, 11850, 16450,  4275, 26300,
15075,  9325, 14125, 12650,  8075, 26850, 29700, 21725, 24575,
31500, 22250, 30800,  6450, 17850, 20675, 30750, 10975, 22750,
21225, 27000, 15900, 33425, 19700, 18950, 31300, 27300, 28600,
24600, 23200, 17150, 12675, 25450, 14875, 25900, 28100, 21850,
23975, 31825, 26500, 17100, 21200, 30400, 25875, 20250, 17675,
16425, 18825,  6375,  7325, 13350, 22475, 29500, 11875, 15550,
16100, 10525, 19775, 22200, 27050, 28625, 27575, 19075, 11225,
 8525, 10325, 19275,  7725,  8275, 10275,  7025,  5175,  5075,
 8475, 14975,  5425, 19900,  7550, 10100,   800, 12125, 13025,
22550,  5450, 17350, 24750,  2975,  4225,  2925, 13450,  7375,
 3275,  5325, 14825,  6825,  3950,  1525, 11650, 11325, 13675,
 7975,  3225, 13075, 18050, 12150, 24800, 14200, 14675, 20600,
11425, 11275, 11125, 10125, 24150, 15275, 20700, 13050,  6425,
17875, 10900,  9225,  9475,  6675,  4175,  3550,  8725, 15800,
 3775, 15750,  7525,  7075, 22650, 10025, 14850, 11575,  4075,
 1775,  8625,  2775,  4725,  8225,  5775, 11025, 11750, 13900,
14150, 15775,  4875,  2525, 31050, 16525,   700, 10450, 13425,
11175,  1550,  4425, 13150,  9850, 12625,  6725,  8025,  5350,
33000, 18725,  9375,  3325,  3025,  9775, 18325, 10575,  5025,
14650, 24400,  7425, 18750, 12225, 23275, 14075,  3575, 10925,
 2125, 14625, 23050, 12275, 15825, 28250, 15150, 32400,  7225,
15675, 14525, 14725, 12075,  8825,  1325, 10425, 24175,  5725,
 1625, 26400,  9725, 17725, 13775,  2575, 13375, 23525, 33950,
17450, 12925, 21100, 19725, 21825, 16725, 23800, 21650, 19450,
14350,  9625, 20475, 18125, 27175, 29800, 23400, 20375, 29550,
17975, 21450, 17900, 19425, 19850, 13625, 15125, 21700, 32500,
23475, 10775, 15575, 11475, 20200, 29850, 23850, 29100, 12950,
16050, 34800,  7175, 18900, 11725,  9650, 13550, 12175,  8375,
14175, 11675, 33600, 15425, 12050,  1100, 12825, 23350, 19575,
13725, 10350, 28800, 16225, 16550, 31150, 23075, 17275, 26025,
 4525, 25500, 12425,  1650, 21575, 20950, 22125, 22325, 26250,
32350, 25200, 21750, 19300, 34475, 20300, 24650, 20450, 31200,
21425, 28200, 22900, 18975, 17425, 28750, 34525, 21350, 23675,
34675, 22875, 19250, 18275, 29900, 21300, 19875, 17075, 32775,
32875, 23425, 32250, 18875, 31725, 29375, 22575, 21625, 16350,
13325, 25375, 27525, 31325, 14225, 16275, 27200, 18575, 29175,
28500, 30500, 24700, 31025, 17225, 31700, 26200,  3875, 32275,
  900, 22350, 11375, 12475, 13175, 24200, 19925, 17375, 16325,
11350, 32525, 30100, 29275,  7625, 13850, 15650, 17300, 17175,
21800, 31800, 14050, 24100,  5675,  1075,  3925,  3475, 29300,
 2625,  2075,  3175, 19975,   500,   950, 20150, 15975,  6525,
```

```
       17925, 12850, 22300,  6175,  6875, 25300, 15025, 31400,  9825,
       19475, 25400, 30225, 34200, 27700, 17825, 24975], dtype=int64)
```

In [62]:
```
df["annual_income"].dtype
```

Out[62]:
```
dtype('float64')
```

- **4) Drop Duplicates**
- There are no duplicates

In [64]:
```
df[df.duplicated()]
```

Out[64]:

id	address_state	application_type	emp_length	emp_title	grade	home_ownership	issue_date	last_credit_pull_date	last_payme

0 rows × 24 columns

- **5) Missing Values**

- There are no missing values

In [66]:
```
df.isnull().sum()/ len(df)
```

Out[66]:
```
id                       0.000000
address_state            0.000000
application_type         0.000000
emp_length               0.000000
emp_title                0.037277
grade                    0.000000
home_ownership           0.000000
issue_date               0.000000
last_credit_pull_date    0.000000
last_payment_date        0.000000
loan_status              0.000000
next_payment_date        0.000000
member_id                0.000000
purpose                  0.000000
sub_grade                0.000000
term                     0.000000
verification_status      0.000000
annual_income            0.000000
dti                      0.000000
installment              0.000000
int_rate                 0.000000
loan_amount              0.000000
total_acc                0.000000
total_payment            0.000000
dtype: float64
```

- **6) Outliers**

In [68]:
```
df[continuous]
```

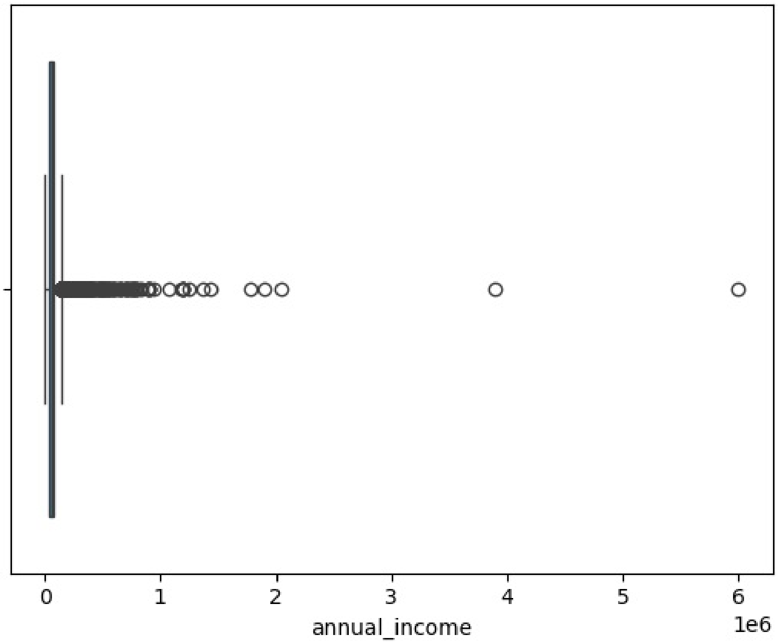
Out[68]:

	annual_income	dti	installment	int_rate	loan_amount	total_payment
0	30000.0	0.0100	59.83	0.1527	2500	1009
1	48000.0	0.0535	109.43	0.1864	3000	3939
2	50000.0	0.2088	421.65	0.1596	12000	3522
3	42000.0	0.0540	97.06	0.1065	4500	4911
4	83000.0	0.0231	106.53	0.0603	3500	3835
...	...	...	...	...	...	...
38571	100000.0	0.1986	551.64	0.1299	24250	31946
38572	50000.0	0.0458	579.72	0.1349	25200	31870
38573	65000.0	0.1734	627.93	0.1749	25000	35721
38574	368000.0	0.0009	612.72	0.1825	24000	33677
38575	80000.0	0.0600	486.86	0.2099	18000	27679

38576 rows × 6 columns

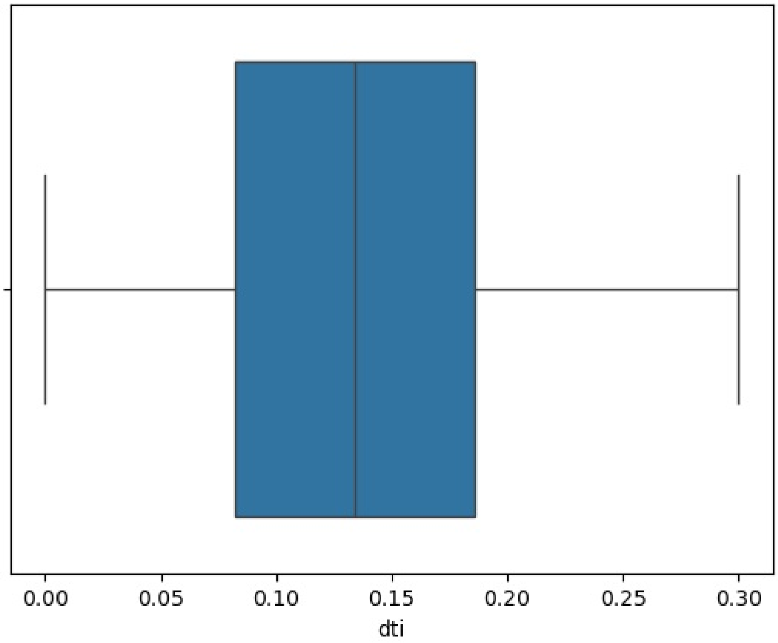
In [69]:
```
sns.boxplot(x = df["annual_income"])
```

Out[69]: <Axes: xlabel='annual_income'>



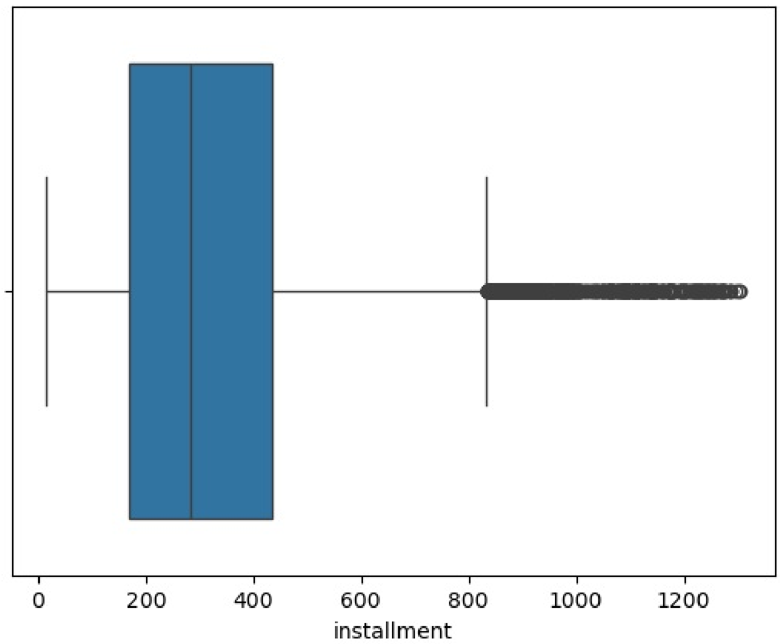
In [70]:
```
sns.boxplot(x = df["dti"])
```

Out[70]: <Axes: xlabel='dti'>



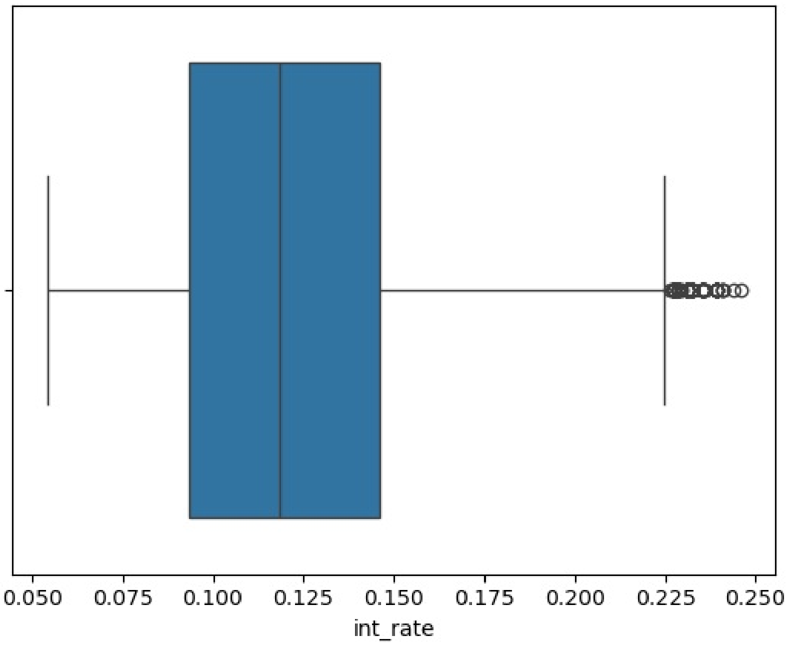
In [71]:
```
sns.boxplot(x = df["installment"])
```

Out[71]: <Axes: xlabel='installment'>



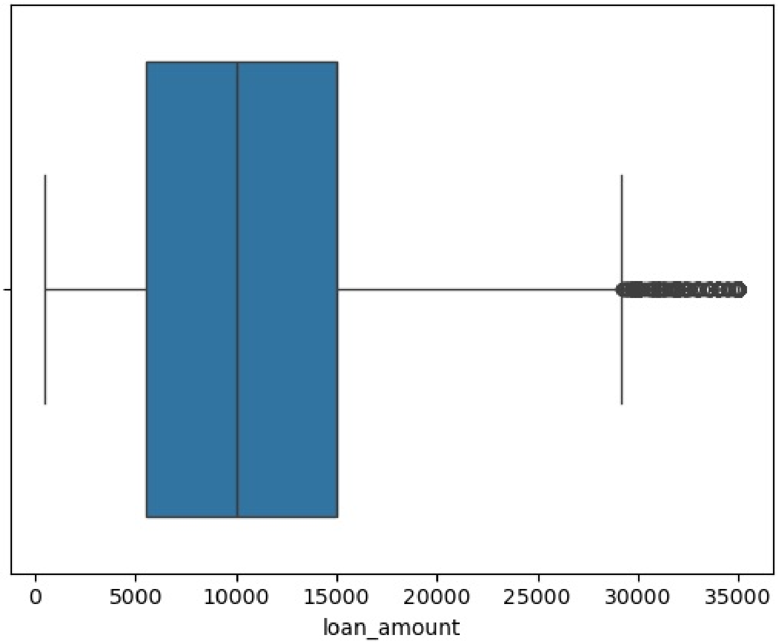

```
In [72]: sns.boxplot(x = df["int_rate"])
```

```
Out[72]: <Axes: xlabel='int_rate'>
```



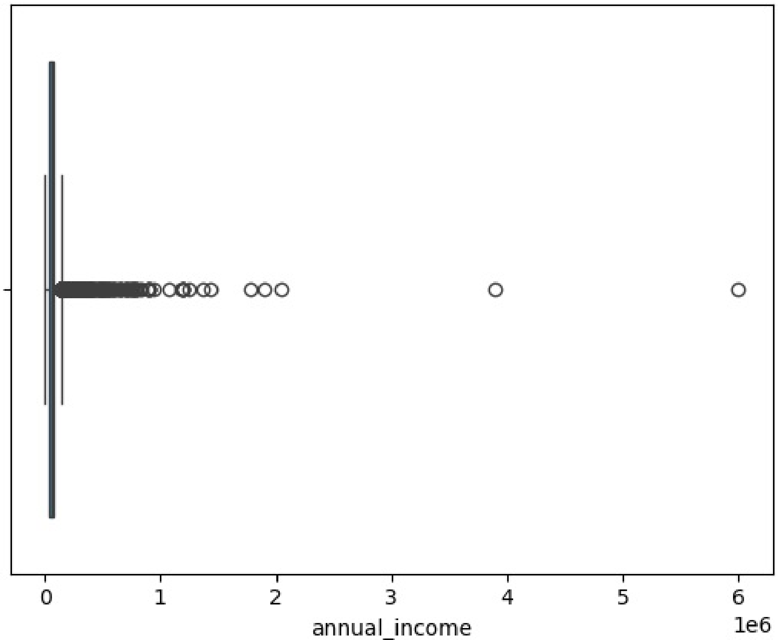

```
In [73]: sns.boxplot(x = df["loan_amount"])
```

```
Out[73]: <Axes: xlabel='loan_amount'>
```



In [74]:
```
sns.boxplot(x = df["annual_income"])
```

Out[74]: <Axes: xlabel='annual_income'>



In [75]:
```
# retrain outiers                          why ?--- > valid(Logical) answer
```

ii) Data Manipulation

In [77]:
```
df[discrete_count].describe()
```

Out[77]:

	emp_length	total_acc
count	38576.000000	38576.000000
mean	4.974829	22.132544
std	3.562833	11.392282
min	0.000000	2.000000
25%	2.000000	14.000000
50%	4.000000	20.000000
75%	9.000000	29.000000
max	10.000000	90.000000

In [78]:
```
df["total_acc_Cus"] = pd.cut(df["total_acc"],
```

```
                              bins = [0, 5, 15, 25, df["total_acc"].max()],
                              labels = ["Low (0-5)", "Moderate (6-15)", "High (16-25)", "Very High (26+)"])
```

In [79]:
```
df["emp_length_Cus"] = pd.cut(df["emp_length"],
                              bins = [0, 2, 6, 9, 10],
                              labels = ["Short(0-2)", "Medium(3-6)", "Long(7-9)", "Very Long(10+)"],
                              include_lowest = True)
```

In [80]:
```
df[continuous].describe()
```

Out[80]:

	annual_income	dti	installment	int_rate	loan_amount	total_payment
count	3.857600e+04	38576.000000	38576.000000	38576.000000	38576.000000	38576.000000
mean	6.964454e+04	0.133274	326.862965	0.120488	11296.066855	12263.348533
std	6.429368e+04	0.066662	209.092000	0.037164	7460.746022	9051.104777
min	4.000000e+03	0.000000	15.690000	0.054200	500.000000	34.000000
25%	4.150000e+04	0.082100	168.450000	0.093200	5500.000000	5633.000000
50%	6.000000e+04	0.134200	283.045000	0.118600	10000.000000	10042.000000
75%	8.320050e+04	0.185900	434.442500	0.145900	15000.000000	16658.000000
max	6.000000e+06	0.299900	1305.190000	0.245900	35000.000000	58564.000000

In [81]:
```
df["Annual_income_Cus"] = pd.cut(df["annual_income"],
                                 bins = [0, 30000, 60000, 100000, 250000, df["annual_income"].max()],
                                 labels = ["Low", "Lower-Middle", "Middle", "Upper-Middle", "High"])
```

In [82]:
```
df["Installments_Cus"] = pd.cut(df["installment"],
                                bins = [0, 200, 400, 600, 800, df["installment"].max()],
                                labels = ["Very Low", "Low", "Medium", "High", "Very High"])
```

In [83]:
```
df["DTI_Cus"] = pd.cut(df["dti"],
                       bins = [0, 0.1, 0.2, 0.3],
                       labels = ["Low", "Moderate", "High"],
                       include_lowest = True)
```

In [84]:
```
df["int_rate_Cus"] = pd.cut(df["int_rate"],
                            bins = [0.05, 0.10, 0.15, 0.20, 0.25],
                            labels = ["Very Low", "Low", "Moderate", "High"])
```

In [85]:
```
df["loan_amount_Cus"] = pd.cut(df["loan_amount"],
                               bins = [0, 5000, 10000, 15000, 20000, 35000],
                               labels = ["Very Small", "Small", "Medium", "Large", "Very Large"])
```

In [86]:
```
df["total_payment_Cus"] = pd.cut(df["total_payment"],
                                 bins = [0, 5000, 10000, 15000, 25000, df["total_payment"].max()],
                                 labels = ["Very Low", "Low", "Medium", "High", "Very High"])
```

In [87]:
```
df
```

Out[87]:

	id	address_state	application_type	emp_length	emp_title	grade	home_ownership	issue_date	last_credit_pull_date
0	1077430	GA	INDIVIDUAL	0	Ryder	C	RENT	11-02-2021	13-09-2021
1	1072053	CA	INDIVIDUAL	9	MKC Accounting	E	RENT	01-01-2021	14-12-2021
2	1069243	CA	INDIVIDUAL	4	Chemat Technology Inc	C	RENT	05-01-2021	12-12-2021
3	1041756	TX	INDIVIDUAL	0	barnes distribution	B	MORTGAGE	25-02-2021	12-12-2021
4	1068350	IL	INDIVIDUAL	10	J&J Steel Inc	A	MORTGAGE	01-01-2021	14-12-2021
...	...	...	...	...	...	...	...	...	..
38571	803452	NJ	INDIVIDUAL	0	Joseph M Sanzari Company	C	MORTGAGE	11-07-2021	16-05-2021
38572	970377	NY	INDIVIDUAL	8	Swat Fame	C	RENT	11-10-2021	16-04-2021
38573	875376	CA	INDIVIDUAL	5	Anaheim Regional Medical Center	D	RENT	11-09-2021	16-05-2021
38574	972997	NY	INDIVIDUAL	5	Brooklyn Radiology	D	RENT	11-10-2021	16-05-2021
38575	682952	NY	INDIVIDUAL	4	Allen Edmonds	F	RENT	11-07-2021	16-05-2021

38576 rows × 32 columns

In [88]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38576 entries, 0 to 38575
Data columns (total 32 columns):
 #   Column                 Non-Null Count  Dtype   
---  ------                 --------------  -----   
 0   id                     38576 non-null  int64   
 1   address_state          38576 non-null  object  
 2   application_type       38576 non-null  object  
 3   emp_length             38576 non-null  int64   
 4   emp_title              37138 non-null  object  
 5   grade                  38576 non-null  object  
 6   home_ownership         38576 non-null  object  
 7   issue_date             38576 non-null  object  
 8   last_credit_pull_date  38576 non-null  object  
 9   last_payment_date      38576 non-null  object  
 10  loan_status            38576 non-null  object  
 11  next_payment_date      38576 non-null  object  
 12  member_id              38576 non-null  int64   
 13  purpose                38576 non-null  object  
 14  sub_grade              38576 non-null  object  
 15  term                   38576 non-null  object  
 16  verification_status    38576 non-null  object  
 17  annual_income          38576 non-null  float64 
 18  dti                    38576 non-null  float64 
 19  installment            38576 non-null  float64 
 20  int_rate               38576 non-null  float64 
 21  loan_amount            38576 non-null  int64   
 22  total_acc              38576 non-null  int64   
 23  total_payment          38576 non-null  int64   
 24  total_acc_Cus          38576 non-null  category
 25  emp_length_Cus         38576 non-null  category
 26  Annual_income_Cus      38576 non-null  category
 27  Installments_Cus       38576 non-null  category
 28  DTI_Cus                38576 non-null  category
 29  int_rate_Cus           38576 non-null  category
 30  loan_amount_Cus        38576 non-null  category
 31  total_payment_Cus      38576 non-null  category
dtypes: category(8), float64(4), int64(6), object(14)
memory usage: 7.4+ MB
```

In [ ]:
```
df.to_excel("Cleaned_DA_1.xlsx")
```

In [89]:
```
df[time_series]
```

Out[89]:

	issue_date	last_credit_pull_date	last_payment_date	next_payment_date
0	11-02-2021	13-09-2021	13-04-2021	13-05-2021
1	01-01-2021	14-12-2021	15-01-2021	15-02-2021
2	05-01-2021	12-12-2021	09-01-2021	09-02-2021
3	25-02-2021	12-12-2021	12-03-2021	12-04-2021
4	01-01-2021	14-12-2021	15-01-2021	15-02-2021
...	...	...	...	...
38571	11-07-2021	16-05-2021	16-05-2021	16-06-2021
38572	11-10-2021	16-04-2021	16-05-2021	16-06-2021
38573	11-09-2021	16-05-2021	16-05-2021	16-06-2021
38574	11-10-2021	16-05-2021	16-05-2021	16-06-2021
38575	11-07-2021	16-05-2021	16-05-2021	16-06-2021

38576 rows × 4 columns

In [90]:

```
# Convert issue_date to datetime
df['issue_date'] = pd.to_datetime(df['issue_date'], format='%d-%m-%Y')
```

In [91]:

```
df["issue_date"].dt.to_period("M")
```

Out[91]:

```
0        2021-02
1        2021-01
2        2021-01
3        2021-02
4        2021-01
           ...
38571    2021-07
38572    2021-10
38573    2021-09
38574    2021-10
38575    2021-07
Name: issue_date, Length: 38576, dtype: period[M]
```

In [164…

```
# Extract year and month for aggregation
df["issue_month"] = df["issue_date"].dt.to_period("M")
df["issue_month"]
```

Out[164…

```
0        2021-02
1        2021-01
2        2021-01
3        2021-02
4        2021-01
           ...
38571    2021-07
38572    2021-10
38573    2021-09
38574    2021-10
38575    2021-07
Name: issue_month, Length: 38576, dtype: period[M]
```

In [ ]:

```
df.to_excel("Cleaned_DA_2.xlsx")
```