In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

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

In [2]:
```
df = pd.read_excel(r"Cleaned_DA_1.xlsx")
df.head()
```

Out[2]:

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

5 rows × 33 columns

In [3]:
```
df = df.drop(columns=['Unnamed: 0'])
df.head()
```

Out[3]:

	id	address_state	application_type	emp_length	emp_title	grade	home_ownership	issue_date	last_credit_pull_date	la
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	

5 rows × 32 columns

Step-4 : Data Analysis

- Using Pandas & Plots (Matplotlib, Seaborn, PowerBI)
- Data Analysis : applying various logics(questions) on given data & observation on the output

In [5]:
```
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 [6]:
```
df[continuous].describe().transpose()
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
annual_income	38576.0	69644.540310	64293.681045	4000.0000	41500.0000	60000.0000	83200.5000	6.000000e+06
dti	38576.0	0.133274	0.066662	0.0000	0.0821	0.1342	0.1859	2.999000e-01
installment	38576.0	326.862965	209.092000	15.6900	168.4500	283.0450	434.4425	1.305190e+03
int_rate	38576.0	0.120488	0.037164	0.0542	0.0932	0.1186	0.1459	2.459000e-01
loan_amount	38576.0	11296.066855	7460.746022	500.0000	5500.0000	10000.0000	15000.0000	3.500000e+04
total_payment	38576.0	12263.348533	9051.104777	34.0000	5633.0000	10042.0000	16658.0000	5.856400e+04

In [7]:
```
df[discrete_count].describe().transpose()
```

Out[7]:

	count	mean	std	min	25%	50%	75%	max
emp_length	38576.0	4.974829	3.562833	0.0	2.0	4.0	9.0	10.0
total_acc	38576.0	22.132544	11.392282	2.0	14.0	20.0	29.0	90.0

In [8]:
```
df.columns
```

Out[8]:
```
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', 'total_acc_Cus', 'emp_length_Cus', 'Annual_income_Cus',
       'Installments_Cus', 'DTI_Cus', 'int_rate_Cus', 'loan_amount_Cus',
       'total_payment_Cus'],
      dtype='object')
```

In [9]:
```
discrete_categorical = [col for col in ['address_state', 'application_type', 'emp_title',
                                        'grade', 'home_ownership', 'loan_status',
                                        'purpose', 'sub_grade', 'term', 'verification_status']
                        if col in df.columns]

df[discrete_categorical].describe().transpose()
```

Out[9]:

	count	unique	top	freq
address_state	38576	50	CA	6894
application_type	38576	1	INDIVIDUAL	38576
emp_title	37138	28525	US Army	135
grade	38576	7	B	11674
home_ownership	38576	5	RENT	18439
loan_status	38576	3	Fully Paid	32145
purpose	38576	14	Debt consolidation	18214
sub_grade	38576	35	B3	2834
term	38576	2	36 months	28237
verification_status	38576	3	Not Verified	16464

KPI :-

In [11]:
```
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  object
 25  emp_length_Cus         38576 non-null  object
 26  Annual_income_Cus      38576 non-null  object
 27  Installments_Cus       38576 non-null  object
 28  DTI_Cus                38576 non-null  object
 29  int_rate_Cus           38576 non-null  object
 30  loan_amount_Cus        38576 non-null  object
 31  total_payment_Cus      38576 non-null  object
dtypes: float64(4), int64(6), object(22)
memory usage: 9.4+ MB
```

In [12]:
```
# KPIs
total_applications = len(df)
total_funded = df['loan_amount'].sum()
total_received = df['total_payment'].sum()
avg_int_rate = df['int_rate'].mean()
avg_dti = df['dti'].mean()
```

In [13]:
```
print("Overall KPIs:")
print(f"Total Loan Applications: {total_applications}")
print(f"Total Funded Amount: {total_funded:.2f}")
print(f"Total Amount Received: {total_received:.2f}")
print(f"Average Interest Rate: {avg_int_rate:.4f}")
print(f"Average DTI: {avg_dti:.4f}")
```

```
Overall KPIs:
Total Loan Applications: 38576
Total Funded Amount: 435757075.00
Total Amount Received: 473070933.00
Average Interest Rate: 0.1205
Average DTI: 0.1333
```

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).

In [15]:
```
df[time_series]
```

Out[15]:

	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 [16]:
```
# Convert issue_date to datetime
df['issue_date'] = pd.to_datetime(df['issue_date'], format='%d-%m-%Y')
```

In [17]:
```
# Calculate applications per month
period_apps_1 = len(df[(df['issue_date'] >= '2021-01-01') & (df['issue_date'] <= '2021-01-31')])
print(f"Total Applications (Jan 2021): {period_apps_1}")

period_apps_2 = len(df[(df['issue_date'] >= '2021-02-01') & (df['issue_date'] <= '2021-02-28')])
print(f"Total Applications (Feb 2021): {period_apps_2}")

period_apps_3 = len(df[(df['issue_date'] >= '2021-03-01') & (df['issue_date'] <= '2021-03-31')])
print(f"Total Applications (March 2021): {period_apps_3}")

period_apps_4 = len(df[(df['issue_date'] >= '2021-04-01') & (df['issue_date'] <= '2021-04-30')])
print(f"Total Applications (April 2021): {period_apps_4}")

period_apps_5 = len(df[(df['issue_date'] >= '2021-05-01') & (df['issue_date'] <= '2021-05-31')])
print(f"Total Applications (May 2021): {period_apps_5}")

period_apps_6 = len(df[(df['issue_date'] >= '2021-06-01') & (df['issue_date'] <= '2021-06-30')])
print(f"Total Applications (June 2021): {period_apps_6}")

period_apps_7 = len(df[(df['issue_date'] >= '2021-07-01') & (df['issue_date'] <= '2021-07-31')])
print(f"Total Applications (July 2021): {period_apps_7}")

period_apps_8 = len(df[(df['issue_date'] >= '2021-08-01') & (df['issue_date'] <= '2021-08-31')])
print(f"Total Applications (Aug 2021): {period_apps_8}")

period_apps_9 = len(df[(df['issue_date'] >= '2021-09-01') & (df['issue_date'] <= '2021-09-30')])
print(f"Total Applications (Sep 2021): {period_apps_9}")

period_apps_10 = len(df[(df['issue_date'] >= '2021-10-01') & (df['issue_date'] <= '2021-10-31')])
print(f"Total Applications (Oct 2021): {period_apps_10}")

period_apps_11 = len(df[(df['issue_date'] >= '2021-11-01') & (df['issue_date'] <= '2021-11-30')])
print(f"Total Applications (Nov 2021): {period_apps_11}")

period_apps_12 = len(df[(df['issue_date'] >= '2021-12-01') & (df['issue_date'] <= '2021-12-31')])
print(f"Total Applications (Dec 2021): {period_apps_12}")

# Calculate total for all 12 months
total_apps_2021 = (period_apps_1 + period_apps_2 + period_apps_3 + period_apps_4 +
                   period_apps_5 + period_apps_6 + period_apps_7 + period_apps_8 +
                   period_apps_9 + period_apps_10 + period_apps_11 + period_apps_12)
print(f"Total Applications (All of 2021): {total_apps_2021}")
```

```
Total Applications (Jan 2021): 2332
Total Applications (Feb 2021): 2279
Total Applications (March 2021): 2627
Total Applications (April 2021): 2755
Total Applications (May 2021): 2911
Total Applications (June 2021): 3184
Total Applications (July 2021): 3366
Total Applications (Aug 2021): 3441
Total Applications (Sep 2021): 3536
Total Applications (Oct 2021): 3796
Total Applications (Nov 2021): 4035
Total Applications (Dec 2021): 4314
Total Applications (All of 2021): 38576
```

In [18]:
```
df["issue_date"].dt.to_period("M")
```

Out[18]:
```
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 [19]:
```
# Extract year and month for aggregation
df["issue_month"] = df["issue_date"].dt.to_period("M")
df["issue_month"]
```

Out[19]:
```
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 [20]:
```
# Count loan applications per month
monthly_counts = df["issue_month"].value_counts().sort_index()
monthly_counts
```

Out[20]:
```
issue_month
2021-01    2332
2021-02    2279
2021-03    2627
2021-04    2755
2021-05    2911
2021-06    3184
2021-07    3366
2021-08    3441
2021-09    3536
2021-10    3796
2021-11    4035
2021-12    4314
Freq: M, Name: count, dtype: int64
```

In [21]:
```
# Compute MoM change (difference from the previous month)
mom_change = monthly_counts.diff()
mom_change
```

Out[21]:
```
issue_month
2021-01      NaN
2021-02    -53.0
2021-03    348.0
2021-04    128.0
2021-05    156.0
2021-06    273.0
2021-07    182.0
2021-08     75.0
2021-09     95.0
2021-10    260.0
2021-11    239.0
2021-12    279.0
Freq: M, Name: count, dtype: float64
```

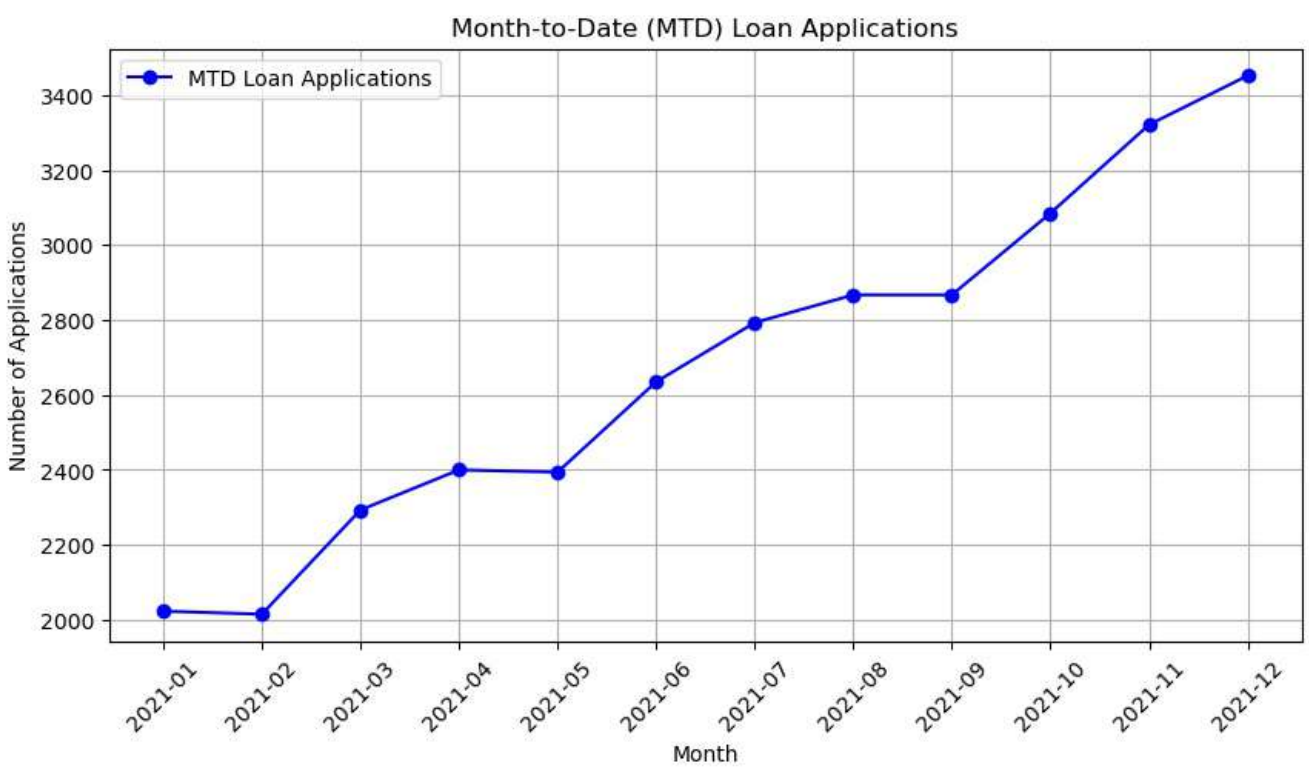
In [22]:
```
# Filter dataset
df_good = df[df["loan_status"] == "Fully Paid"]
df_bad = df[df["loan_status"] == "Charged Off"]
```

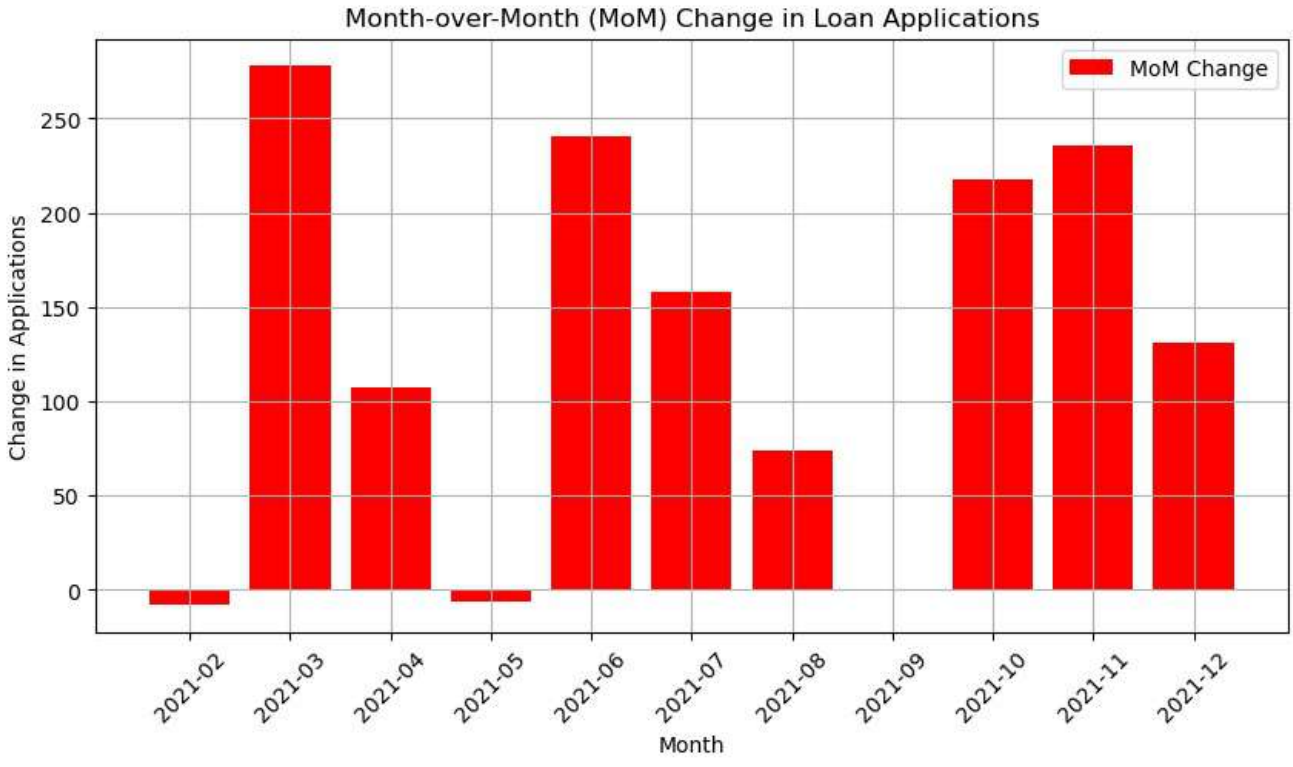
In [23]:
```
# Count loan applications per month (Good Loans)
monthly_counts_good = df_good["issue_month"].value_counts().sort_index()
mom_change_good = monthly_counts_good.diff()
```

In [24]:
```
# Count loan applications per month (Bad Loans)
monthly_counts_bad = df_bad["issue_month"].value_counts().sort_index()
mom_change_bad = monthly_counts_bad.diff()
```

In [25]:

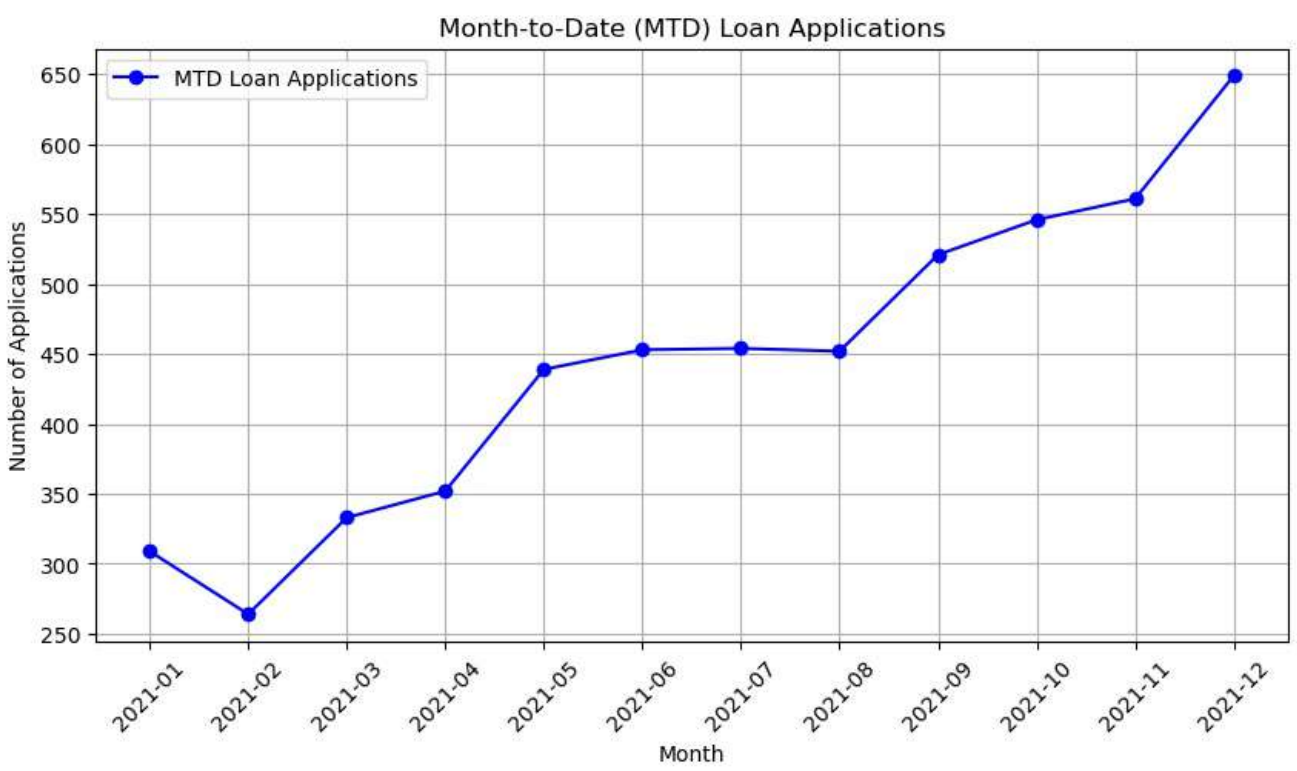
```
# For Good Loans
# Plot MTD Loan  for good loans
plt.figure(figsize=(10, 5))
plt.plot(monthly_counts_good.index.astype(str), monthly_counts_good, marker="o", label="MTD Loan Applications",
plt.xlabel("Month")
plt.ylabel("Number of Applications")
plt.title("Month-to-Date (MTD) Loan Applications")
plt.xticks(rotation=45)
plt.legend()
plt.grid()
plt.show()

# Plot MoM Change for good loans
plt.figure(figsize=(10, 5))
plt.bar(mom_change_good.index.astype(str), mom_change_good, color="r", label="MoM Change")
plt.xlabel("Month")
plt.ylabel("Change in Applications")
plt.title("Month-over-Month (MoM) Change in Loan Applications")
plt.xticks(rotation=45)
plt.legend()
plt.grid()
plt.show()
```





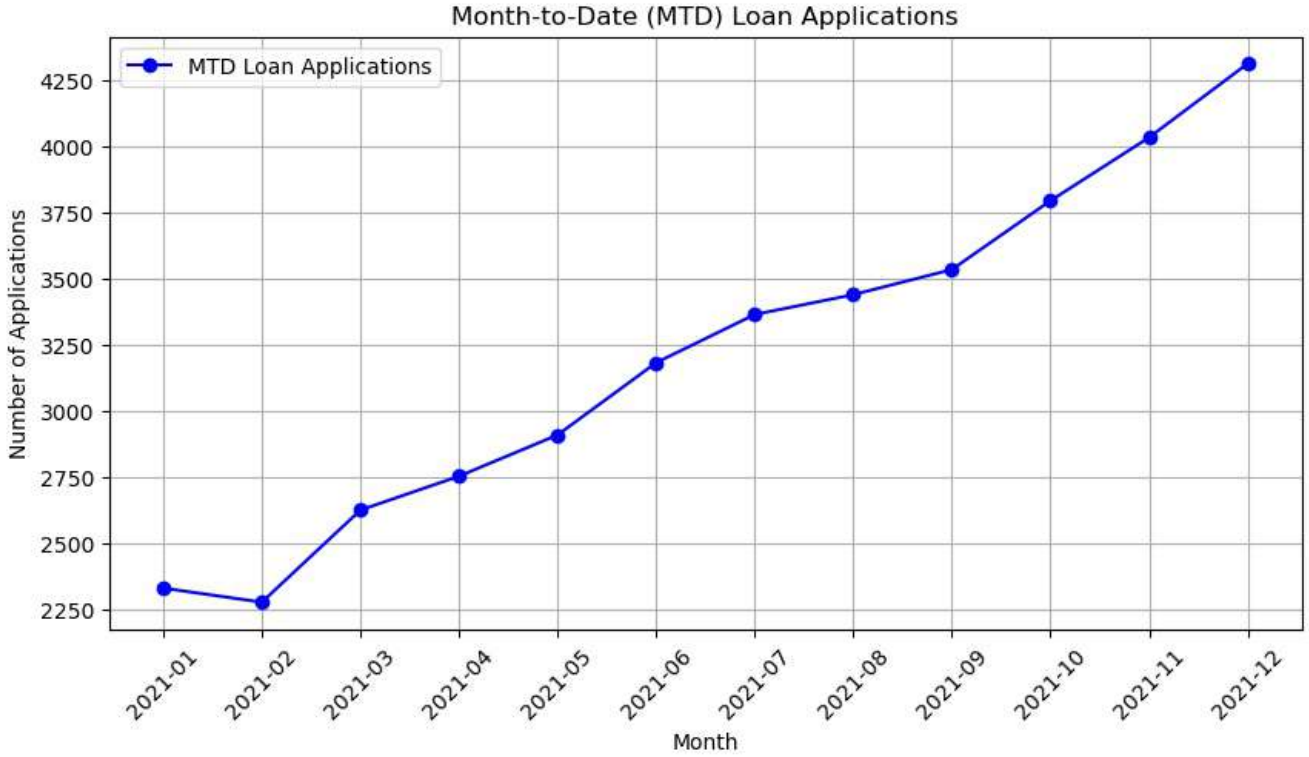
In [26]:

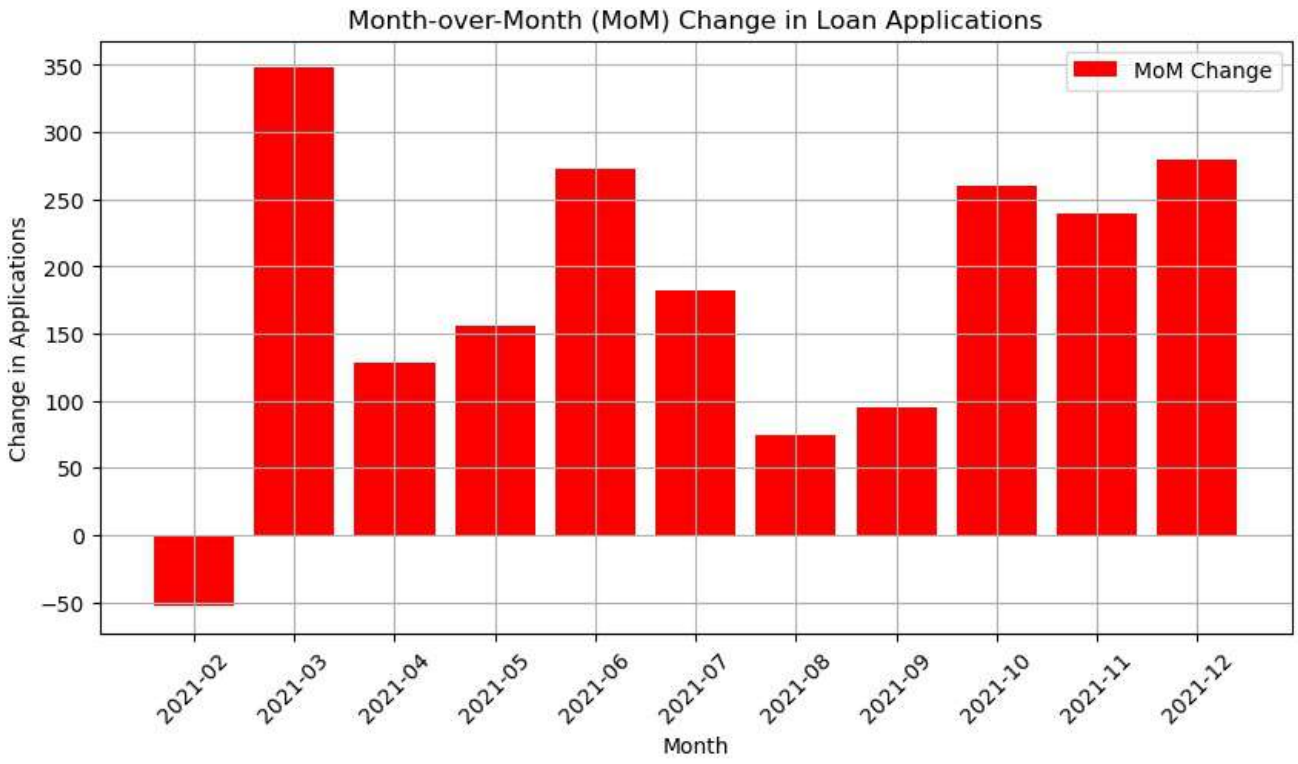
```
# For Bad Loans
# Plot MTD Loan Applications
plt.figure(figsize=(10, 5))
plt.plot(monthly_counts_bad.index.astype(str), monthly_counts_bad, marker="o", label="MTD Loan Applications", c
plt.xlabel("Month")
plt.ylabel("Number of Applications")
plt.title("Month-to-Date (MTD) Loan Applications")
plt.xticks(rotation=45)
plt.legend()
plt.grid()
plt.show()

# Plot MoM Change
plt.figure(figsize=(10, 5))
plt.bar(mom_change_bad.index.astype(str), mom_change_bad, color="r", label="MoM Change")
plt.xlabel("Month")
plt.ylabel("Change in Applications")
plt.title("Month-over-Month (MoM) Change in Loan Applications")
plt.xticks(rotation=45)
plt.legend()
plt.grid()
plt.show()
```



In [ ]:

In [27]:
```
# For over all Loan Applicants
# Plot MTD Loan Applications
plt.figure(figsize=(10, 5))
plt.plot(monthly_counts.index.astype(str), monthly_counts, marker="o", label="MTD Loan Applications", color="b"
plt.xlabel("Month")
plt.ylabel("Number of Applications")
plt.title("Month-to-Date (MTD) Loan Applications")
plt.xticks(rotation=45)
plt.legend()
plt.grid()
plt.show()

# Plot MoM Change
plt.figure(figsize=(10, 5))
plt.bar(mom_change.index.astype(str), mom_change, color="r", label="MoM Change")
plt.xlabel("Month")
plt.ylabel("Change in Applications")
plt.title("Month-over-Month (MoM) Change in Loan Applications")
plt.xticks(rotation=45)
plt.legend()
plt.grid()
plt.show()
```





Observations

- In Month-to-Date (MTD) Loan Applications **there is a increasing trend of number of Loan Applications from Feb to Dec.**
- In Month-over-Month (MoM) Change in Loan Applications **There is a decrease in loan application in April, May,August and September.**
- We can observe **increased loans in March and also in the year end months like Oct, Nov, Dec.**

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.

In [30]:
```
total_funded = df['loan_amount'].sum()
total_funded
```

Out[30]: 435757075

In [31]:
```
# calculate MTD Funded Amount
```

```
mtd_funded_amount = df.groupby("issue_month")["loan_amount"].sum()
mtd_funded_amount
```

Out[31]:
```
issue_month
2021-01    25031650
2021-02    24647825
2021-03    28875700
2021-04    29800800
2021-05    31738350
2021-06    34161475
2021-07    35813900
2021-08    38149600
2021-09    40907725
2021-10    44893800
2021-11    47754825
2021-12    53981425
Freq: M, Name: loan_amount, dtype: int64
```

In [32]:
```
# Calculate MoM change
mom_change = mtd_funded_amount.diff()
mom_change
```

Out[32]:
```
issue_month
2021-01          NaN
2021-02    -383825.0
2021-03    4227875.0
2021-04     925100.0
2021-05    1937550.0
2021-06    2423125.0
2021-07    1652425.0
2021-08    2335700.0
2021-09    2758125.0
2021-10    3986075.0
2021-11    2861025.0
2021-12    6226600.0
Freq: M, Name: loan_amount, dtype: float64
```

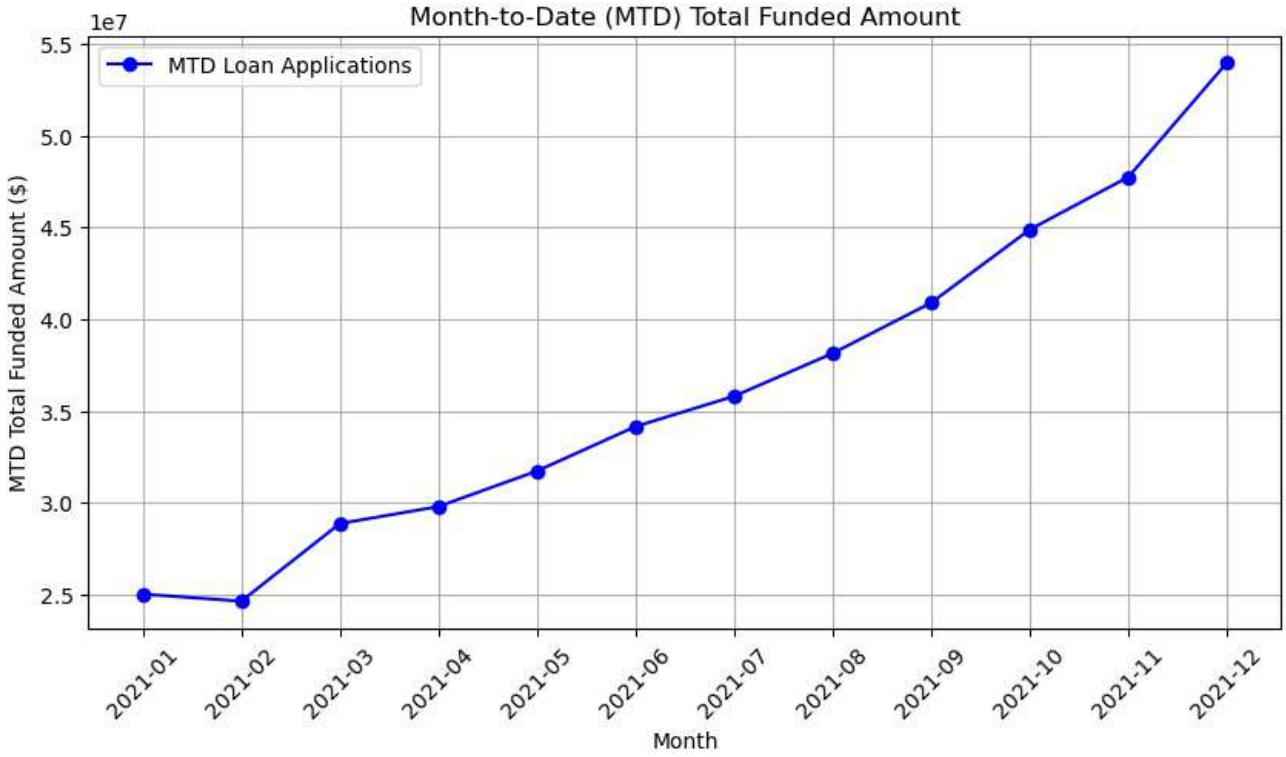
In [33]:
```
import pandas as pd

# Create DataFrame for Power BI
powerbi_df = pd.DataFrame({
    "issue_month": mtd_funded_amount.index,
    "MTD_Funded_Amount": mtd_funded_amount.values,
    "MoM_Change": mom_change.values
})

# Display the DataFrame
print(powerbi_df)
```

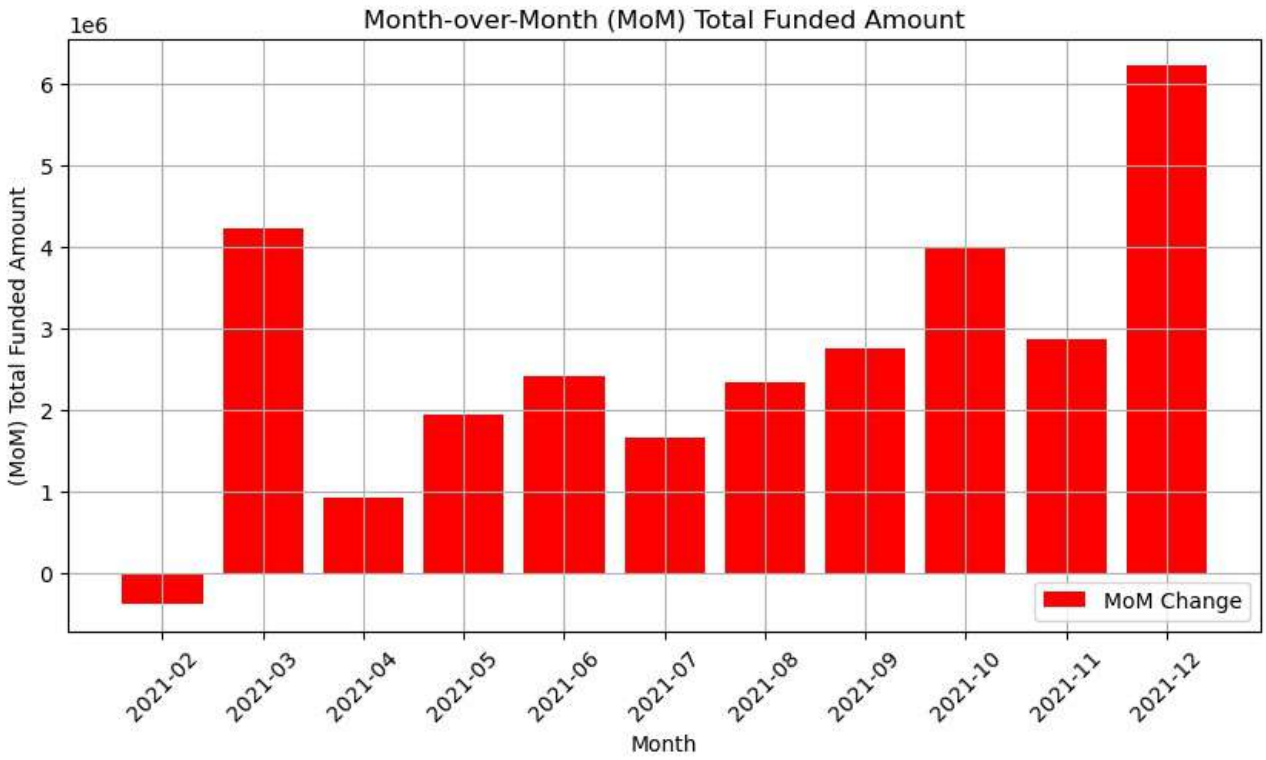
```
   issue_month  MTD_Funded_Amount  MoM_Change
0      2021-01           25031650         NaN
1      2021-02           24647825   -383825.0
2      2021-03           28875700   4227875.0
3      2021-04           29800800    925100.0
4      2021-05           31738350   1937550.0
5      2021-06           34161475   2423125.0
6      2021-07           35813900   1652425.0
7      2021-08           38149600   2335700.0
8      2021-09           40907725   2758125.0
9      2021-10           44893800   3986075.0
10     2021-11           47754825   2861025.0
11     2021-12           53981425   6226600.0
```

In [34]:
```
powerbi_df.to_csv("MoM_Total_Funded_Amount.csv")
```

In [35]:
```
# Plot Month-to-Date (MTD) Total Funded Amount
plt.figure(figsize=(10, 5))
plt.plot(mtd_funded_amount.index.astype(str), mtd_funded_amount, marker="o", label="MTD Loan Applications", colo
plt.xlabel("Month")
plt.ylabel("MTD Total Funded Amount ($)")
plt.title("Month-to-Date (MTD) Total Funded Amount")
plt.xticks(rotation=45)
plt.legend()
plt.grid()
plt.show()

# Plot MoM Change
plt.figure(figsize=(10, 5))
plt.bar(mom_change.index.astype(str), mom_change, color="r", label="MoM Change")
plt.xlabel("Month")
plt.ylabel("(MoM) Total Funded Amount")
plt.title("Month-over-Month (MoM) Total Funded Amount")
plt.xticks(rotation=45)
plt.legend()
```

```
plt.grid()
plt.show()
```





Observations

- In Month-to-Date (MTD) Total Funded Amount **we can observe that the trend only falls on Feb and then from Feb to Dec it increses**
- In We can observe **increased Total Funded AMount in March and also increases from april to june & july to Sep and also in the year end months like Oct and Dec.**

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.

In [38]:
```
total_received = df['total_payment'].sum()
```

```
total_received
```

Out[38]:
```
473070933
```

In [39]:
```
# calculate MTD Total Amount Received
mtd_total_received = df.groupby("issue_month")["total_payment"].sum()
mtd_total_received
```

Out[39]:
```
issue_month
2021-01    27578836
2021-02    27717745
2021-03    32264400
2021-04    32495533
2021-05    33750523
2021-06    36164533
2021-07    38827220
2021-08    42682218
2021-09    43983948
2021-10    49399567
2021-11    50132030
2021-12    58074380
Freq: M, Name: total_payment, dtype: int64
```

In [40]:
```
# Calculate MoM change of Total Amount Received
mom_change = mtd_total_received.diff()
mom_change
```

Out[40]:
```
issue_month
2021-01          NaN
2021-02     138909.0
2021-03    4546655.0
2021-04     231133.0
2021-05    1254990.0
2021-06    2414010.0
2021-07    2662687.0
2021-08    3854998.0
2021-09    1301730.0
2021-10    5415619.0
2021-11     732463.0
2021-12    7942350.0
Freq: M, Name: total_payment, dtype: float64
```

In [41]:
```
# Create DataFrame for Power BI
powerbi_df = pd.DataFrame({
    "issue_month": mtd_total_received.index,
    "mtd_total_received": mtd_total_received.values,
    "mom_change": mom_change.values
})

# Display the DataFrame
print(powerbi_df)
```

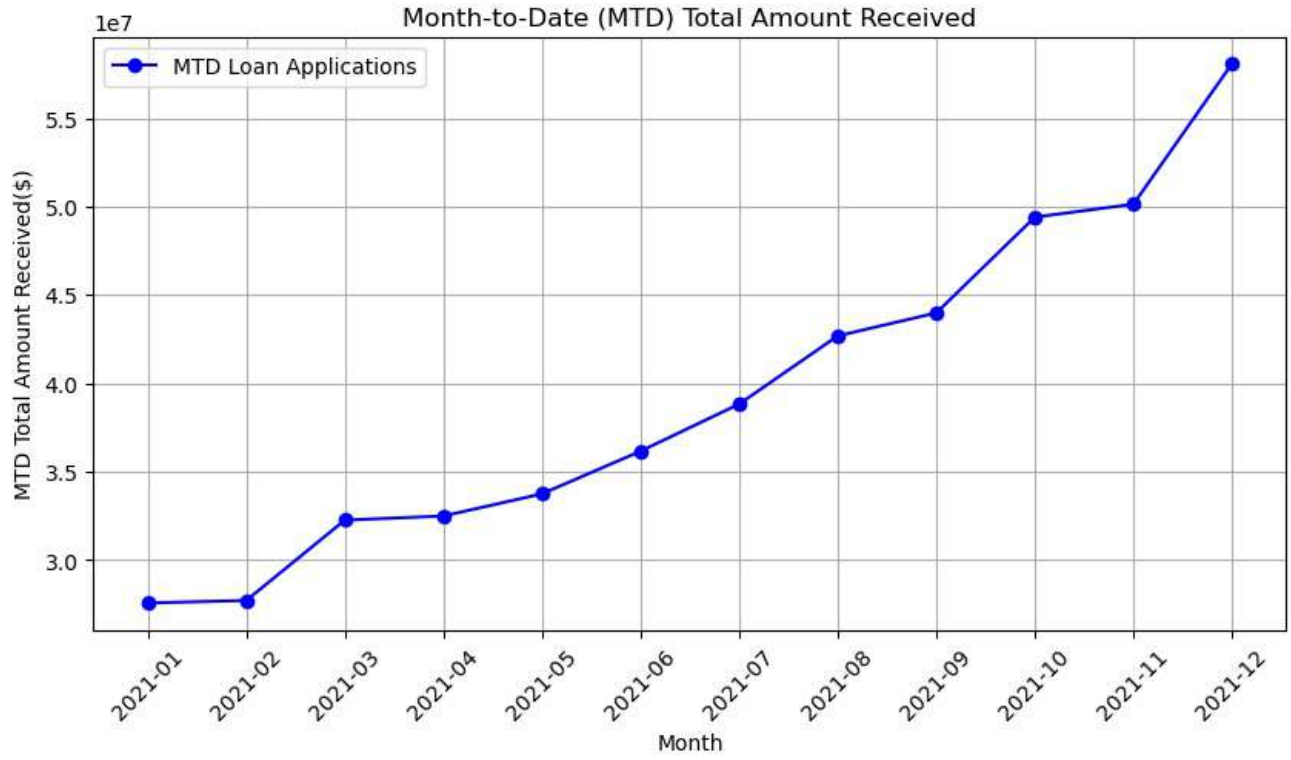
```
   issue_month  mtd_total_received  mom_change
0      2021-01            27578836         NaN
1      2021-02            27717745    138909.0
2      2021-03            32264400   4546655.0
3      2021-04            32495533    231133.0
4      2021-05            33750523   1254990.0
5      2021-06            36164533   2414010.0
6      2021-07            38827220   2662687.0
7      2021-08            42682218   3854998.0
8      2021-09            43983948   1301730.0
9      2021-10            49399567   5415619.0
10     2021-11            50132030    732463.0
11     2021-12            58074380   7942350.0
```

In [42]:
```
powerbi_df.to_csv("MoM_Total_Received.csv")
```

In [ ]:
```

```

In [43]:
```
# Plot Month-to-Date (MTD) Total Amount Received
plt.figure(figsize=(10, 5))
plt.plot(mtd_total_received.index.astype(str), mtd_total_received, marker="o", label="MTD Loan Applications", c
plt.xlabel("Month")
plt.ylabel("MTD Total Amount Received($)")
plt.title("Month-to-Date (MTD) Total Amount Received")
plt.xticks(rotation=45)
plt.legend()
plt.grid()
plt.show()

# Plot MoM Change of Total Amount Received
plt.figure(figsize=(10, 5))
```

```
plt.bar(mom_change.index.astype(str), mom_change, color="r", label="MoM Change")
plt.xlabel("Month")
plt.ylabel("(MoM) Total Amount Received")
plt.title("Month-over-Month (MoM) Total Amount Received")
plt.xticks(rotation=45)
plt.legend()
plt.grid()
plt.show()
```



Observations

- **Increasing MTD total amounts indicate a growing demand for loans or larger loan amounts being disbursed, signaling strong market conditions.**
- **Consistently increasing MoM totals from april to august suggests increased loan demand or economic recovery, where borrowers are willing to take larger loans or meet repayments.**
- **A large positive MoM change in Oct and Dec could reflect a one-time surge in loan demand, possibly due to holiday spending or a special promotion.**

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.

In [46]:
```
avg_int_rate = df['int_rate'].mean()
avg_int_rate
```

Out[46]:
```
0.12048831397760265
```

In [47]:
```
# calculate MTD Average Interest Rate
mtd_avg_int_rate = df.groupby("issue_month")["int_rate"].mean()
mtd_avg_int_rate
```

Out[47]:
```
issue_month
2021-01    0.114619
2021-02    0.117216
2021-03    0.118583
2021-04    0.117409
2021-05    0.122578
2021-06    0.122742
2021-07    0.122372
2021-08    0.123002
2021-09    0.120032
2021-10    0.120241
2021-11    0.119417
2021-12    0.123560
Freq: M, Name: int_rate, dtype: float64
```

In [48]:
```
# Calculate MoM change of Total Amount Received
mom_change = mtd_avg_int_rate.diff()
mom_change
```

Out[48]:
```
issue_month
2021-01         NaN
2021-02    0.002597
2021-03    0.001367
2021-04   -0.001174
2021-05    0.005169
2021-06    0.000164
2021-07   -0.000370
2021-08    0.000630
2021-09   -0.002970
2021-10    0.000209
2021-11   -0.000824
2021-12    0.004143
Freq: M, Name: int_rate, dtype: float64
```

In [49]:
```
# Create DataFrame for Power BI
powerbi_df = pd.DataFrame({
    "issue_month": mtd_avg_int_rate.index,
    "mtd_avg_int_rate": mtd_avg_int_rate.values,
    "mom_change": mom_change.values
})

# Display the DataFrame
print(powerbi_df)
```

```
   issue_month  mtd_avg_int_rate  mom_change
0      2021-01          0.114619         NaN
1      2021-02          0.117216    0.002597
2      2021-03          0.118583    0.001367
3      2021-04          0.117409   -0.001174
4      2021-05          0.122578    0.005169
5      2021-06          0.122742    0.000164
6      2021-07          0.122372   -0.000370
7      2021-08          0.123002    0.000630
8      2021-09          0.120032   -0.002970
9      2021-10          0.120241    0.000209
10     2021-11          0.119417   -0.000824
11     2021-12          0.123560    0.004143
```

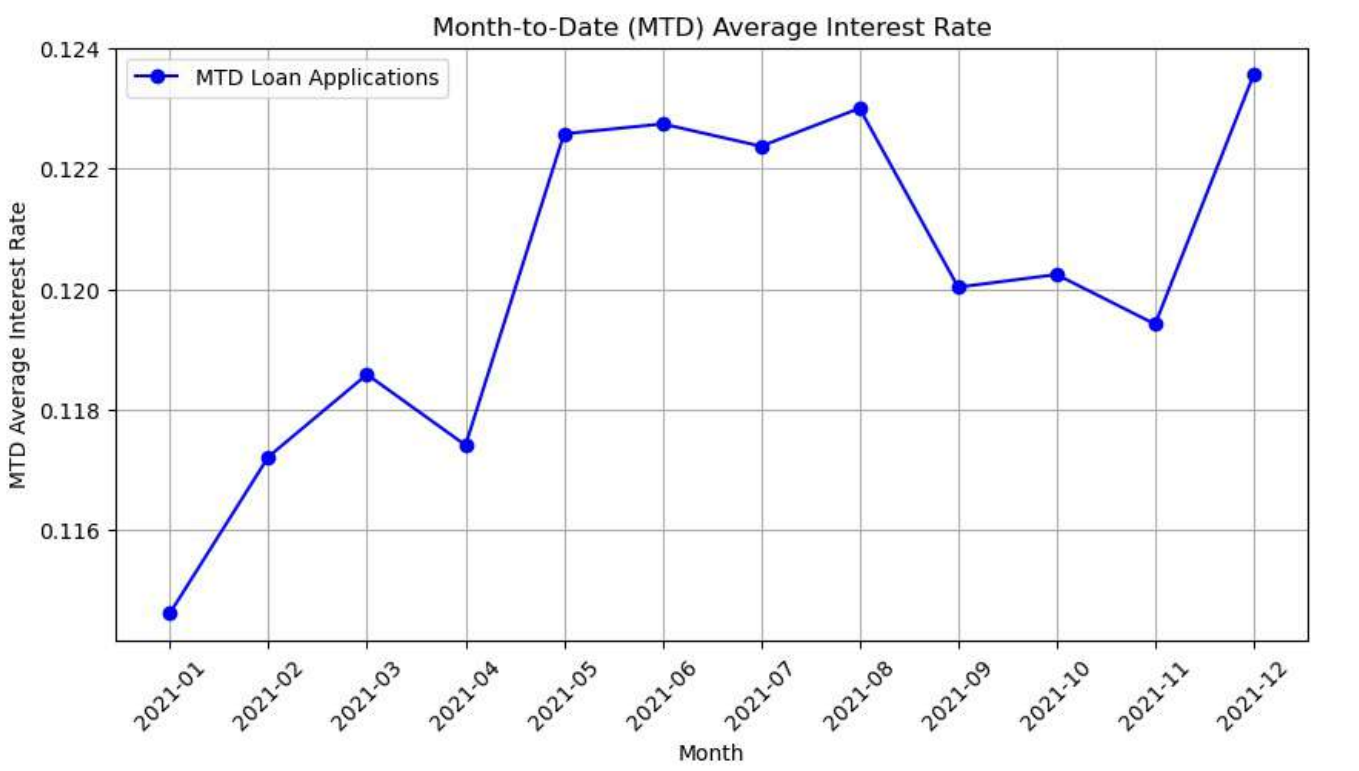
In [50]:
```
powerbi_df.to_csv("MoM_Avg_Int_Rate.csv")
```

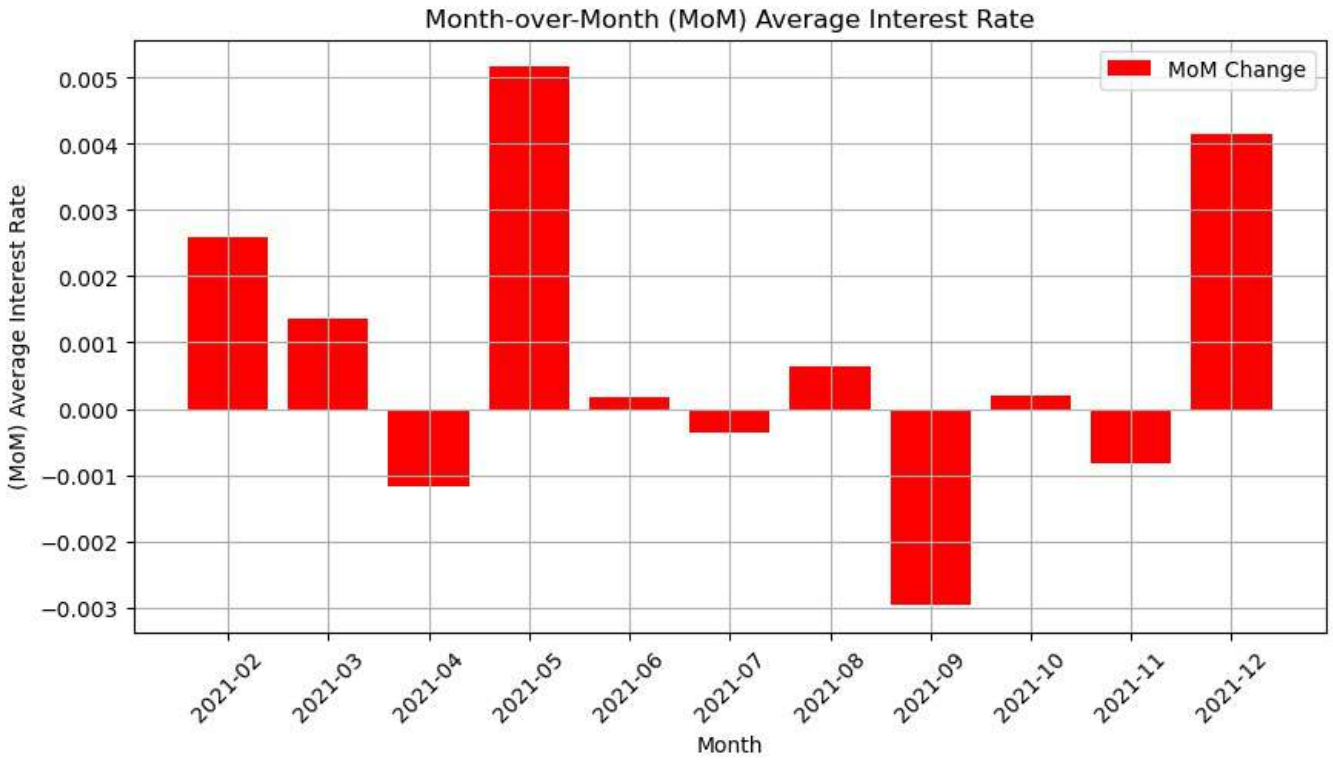
In [ ]:
```

```

In [ ]:
```

```

In [51]:
```
# Plot Month-to-Date (MTD) Total Amount Received
```

```
plt.figure(figsize=(10, 5))
plt.plot(mtd_avg_int_rate.index.astype(str), mtd_avg_int_rate, marker="o", label="MTD Loan Applications", color=
plt.xlabel("Month")
plt.ylabel("MTD Average Interest Rate")
plt.title("Month-to-Date (MTD) Average Interest Rate")
plt.xticks(rotation=45)
plt.legend()
plt.grid()
plt.show()

# Plot MoM Change of Total Amount Received
plt.figure(figsize=(10, 5))
plt.bar(mom_change.index.astype(str), mom_change, color="r", label="MoM Change")
plt.xlabel("Month")
plt.ylabel("(MoM) Average Interest Rate")
plt.title("Month-over-Month (MoM) Average Interest Rate")
plt.xticks(rotation=45)
plt.legend()
plt.grid()
plt.show()
```





Observations

- **In the month 4,9,10,11 april, september, october the is fall in MTD Average interest rate.**
- **In MoM Positive bars indicate that the interest rate has increased compared to the previous month, suggesting tightening credit conditions or rising risk factors, Negative bars indicate a decline in average interest rates, possibly due to lower risk profiles or lender competition.**

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.

In [54]:
```
avg_dti = df['dti'].mean()
avg_dti
```

Out[54]:
```
0.13327433119037743
```

In [55]:
```
# calculate MTD Average Debt-to-Income Ratio (DTI)
mtd_avg_dti = df.groupby("issue_month")["dti"].mean()
mtd_avg_dti
```

Out[55]:
```
issue_month
2021-01    0.129370
2021-02    0.134093
2021-03    0.132156
2021-04    0.132194
2021-05    0.133337
2021-06    0.132438
2021-07    0.132948
2021-08    0.133532
2021-09    0.132978
2021-10    0.134144
2021-11    0.133027
2021-12    0.136655
Freq: M, Name: dti, dtype: float64
```

In [56]:
```
# Calculate MoM change of Average Debt-to-Income Ratio (DTI)
mom_change = mtd_avg_dti.diff()
mom_change
```

Out[56]:
```
issue_month
2021-01         NaN
2021-02    0.004723
2021-03   -0.001937
2021-04    0.000037
2021-05    0.001144
2021-06   -0.000900
2021-07    0.000510
2021-08    0.000584
2021-09   -0.000554
2021-10    0.001165
2021-11   -0.001116
2021-12    0.003628
Freq: M, Name: dti, dtype: float64
```

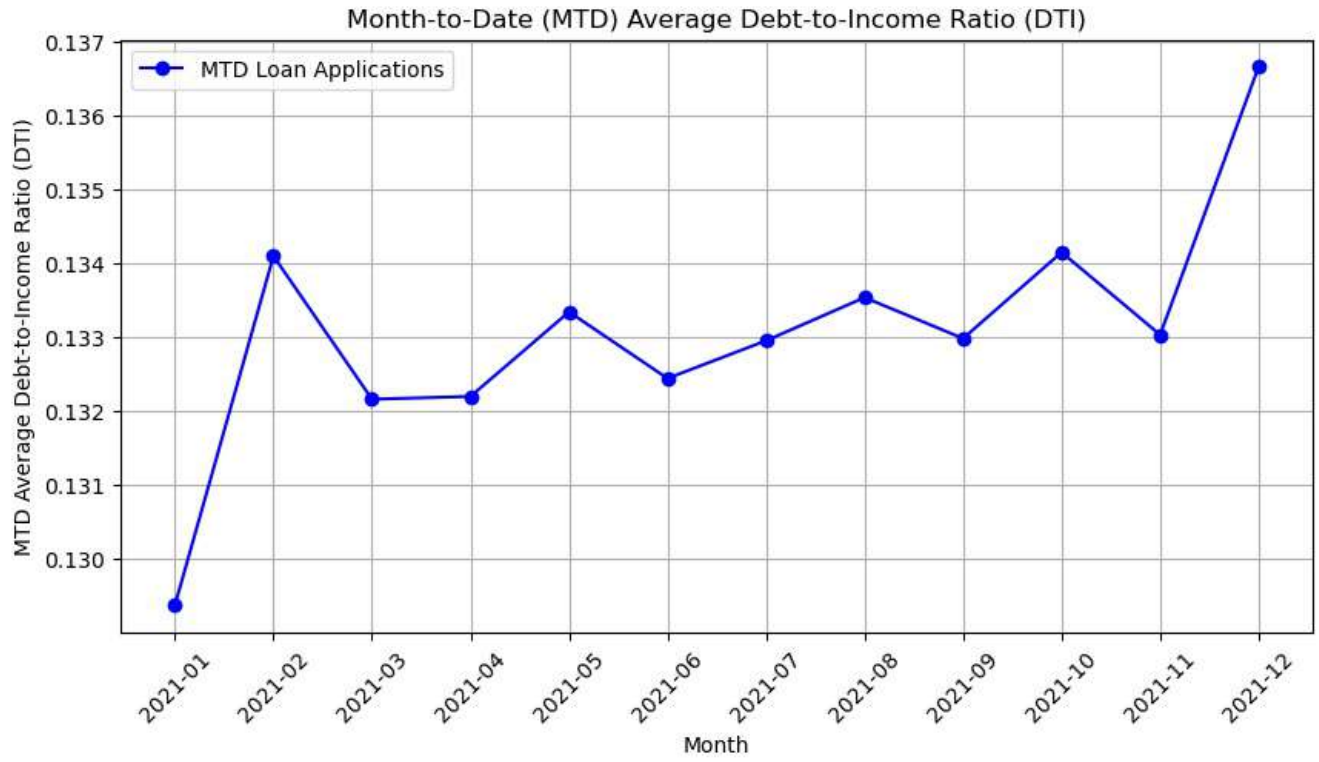
In [57]:
```
# Create DataFrame for Power BI
powerbi_df = pd.DataFrame({
    "issue_month": mtd_avg_dti.index,
    "mtd_avg_dti": mtd_avg_dti.values,
    "mom_change": mom_change.values
})

# Display the DataFrame
print(powerbi_df)
```

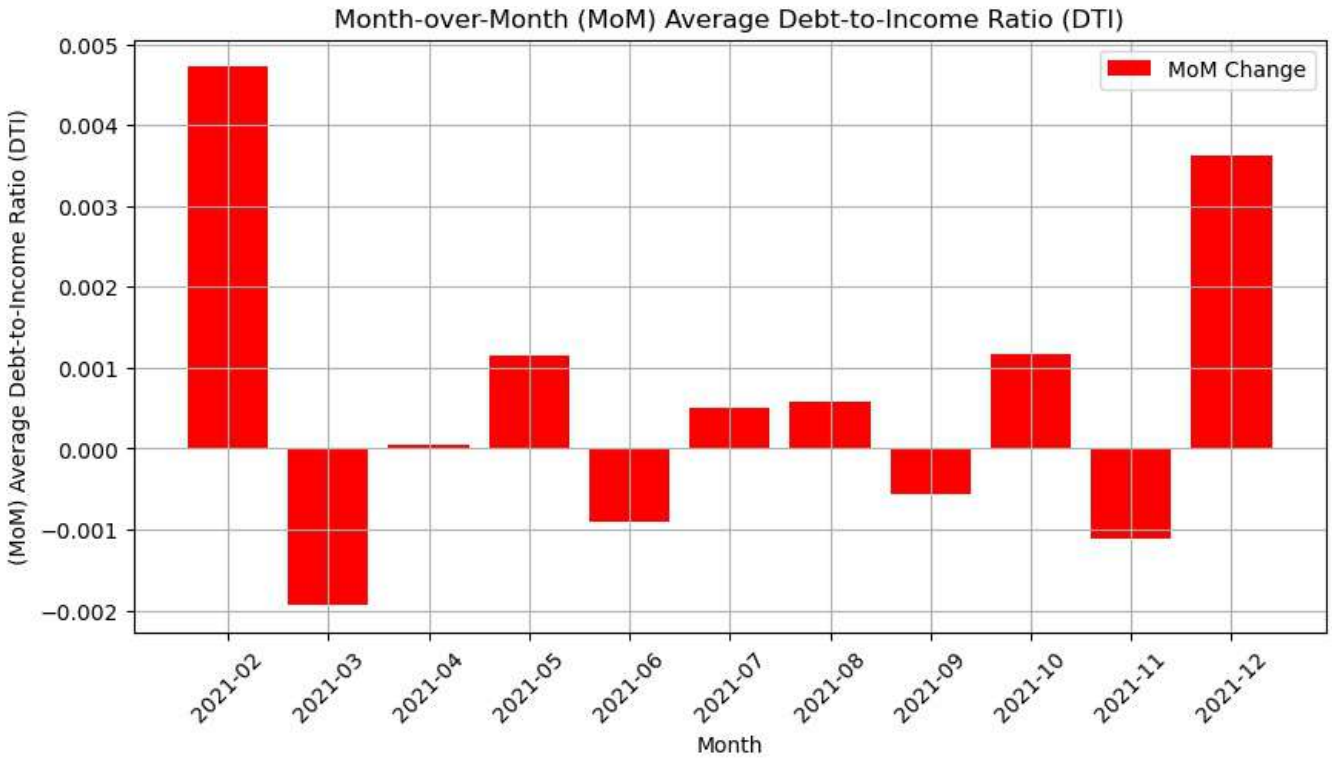
```
   issue_month  mtd_avg_dti  mom_change
0      2021-01     0.129370         NaN
1      2021-02     0.134093    0.004723
2      2021-03     0.132156   -0.001937
3      2021-04     0.132194    0.000037
4      2021-05     0.133337    0.001144
5      2021-06     0.132438   -0.000900
6      2021-07     0.132948    0.000510
7      2021-08     0.133532    0.000584
8      2021-09     0.132978   -0.000554
9      2021-10     0.134144    0.001165
10     2021-11     0.133027   -0.001116
11     2021-12     0.136655    0.003628
```

In [58]:
```
powerbi_df.to_csv("MoM_Avg_DTI.csv")
```

In [ ]:
```

```

In [60]:
```
# Plot Month-to-Date (MTD) Average Debt-to-Income Ratio (DTI)
plt.figure(figsize=(10, 5))
plt.plot(mtd_avg_dti.index.astype(str), mtd_avg_dti, marker="o", label="MTD Loan Applications", color="b")
plt.xlabel("Month")
plt.ylabel("MTD Average Debt-to-Income Ratio (DTI)")
plt.title("Month-to-Date (MTD) Average Debt-to-Income Ratio (DTI)")
plt.xticks(rotation=45)
plt.legend()
plt.grid()
plt.show()

# Plot MoM Change of Average Debt-to-Income Ratio (DTI)
plt.figure(figsize=(10, 5))
plt.bar(mom_change.index.astype(str), mom_change, color="r", label="MoM Change")
plt.xlabel("Month")
plt.ylabel("(MoM) Average Debt-to-Income Ratio (DTI)")
plt.title("Month-over-Month (MoM) Average Debt-to-Income Ratio (DTI)")
plt.xticks(rotation=45)
plt.legend()
plt.grid()
plt.show()
```





Observations

- **A rising MTD trend with positive MoM changes may signal increased financial stress, requiring careful risk assessment.**
- **There is constant increase and decrease in MoM Average DTI, Large fluctuations in MoM change could be due to policy changes, economic conditions, or seasonal spending patterns.**

Good & Bad Loans

In [63]:
```
# Good Loans (Fully Paid)
good_loans = df[df['loan_status'] == 'Fully Paid']
good_loan_apps = len(good_loans)
good_loan_pct = (good_loan_apps / total_applications * 100)
good_loan_funded = good_loans['loan_amount'].sum()
good_loan_received = good_loans['total_payment'].sum()

print(f"Good Loan Application Percentage: {good_loan_pct:.2f}%")
print(f"Good Loan Applications: {good_loan_apps}")
print(f"Good Loan Funded Amount: {good_loan_funded:.2f}")
print(f"Good Loan Total Received Amount: {good_loan_received:.2f}")
```

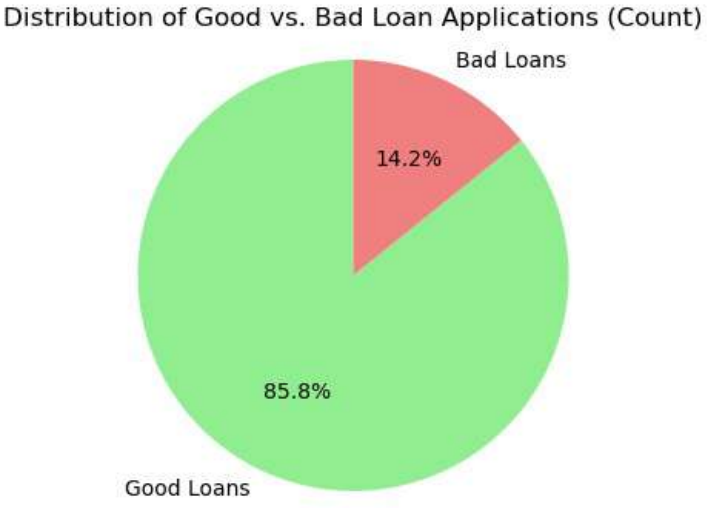
```
Good Loan Application Percentage: 83.33%
Good Loan Applications: 32145
Good Loan Funded Amount: 351358350.00
Good Loan Total Received Amount: 411586256.00
```

In [64]:
```python
# Bad Loans (Charged Off)
bad_loans = df[df['loan_status'] == 'Charged Off']
bad_loan_apps = len(bad_loans)
bad_loan_pct = (bad_loan_apps / total_applications * 100)
bad_loan_funded = bad_loans['loan_amount'].sum()
bad_loan_received = bad_loans['total_payment'].sum()

print(f"Bad Loan Application Percentage: {bad_loan_pct:.2f}%")
print(f"Bad Loan Applications: {bad_loan_apps}")
print(f"Bad Loan Funded Amount: {bad_loan_funded:.2f}")
print(f"Bad Loan Total Received Amount: {bad_loan_received:.2f}")
```

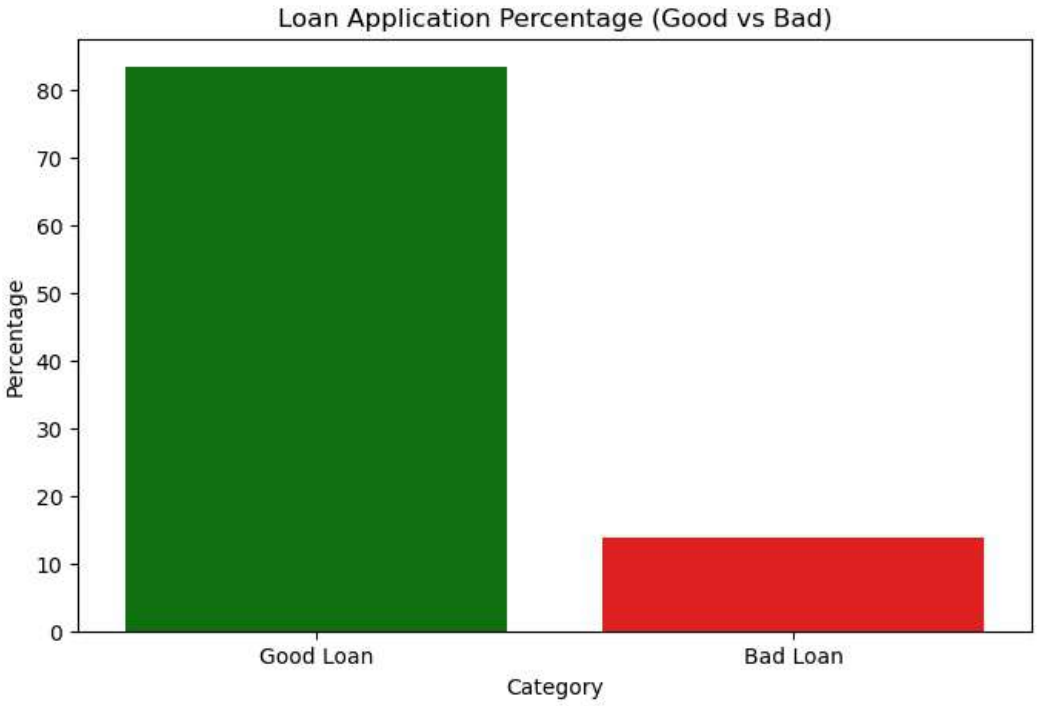
```
Bad Loan Application Percentage: 13.82%
Bad Loan Applications: 5333
Bad Loan Funded Amount: 65532225.00
Bad Loan Total Received Amount: 37284763.00
```

In [65]:
```python
# Create summary dataframe
summary_df = pd.DataFrame({
    "Category": ["Good Loan", "Bad Loan"],
    "Applications": [good_loan_apps, bad_loan_apps],
    "Funded Amount": [good_loan_funded, bad_loan_funded],
    "Total Received Amount": [good_loan_received, bad_loan_received],
    "Application Percentage": [good_loan_pct, bad_loan_pct]
})
```

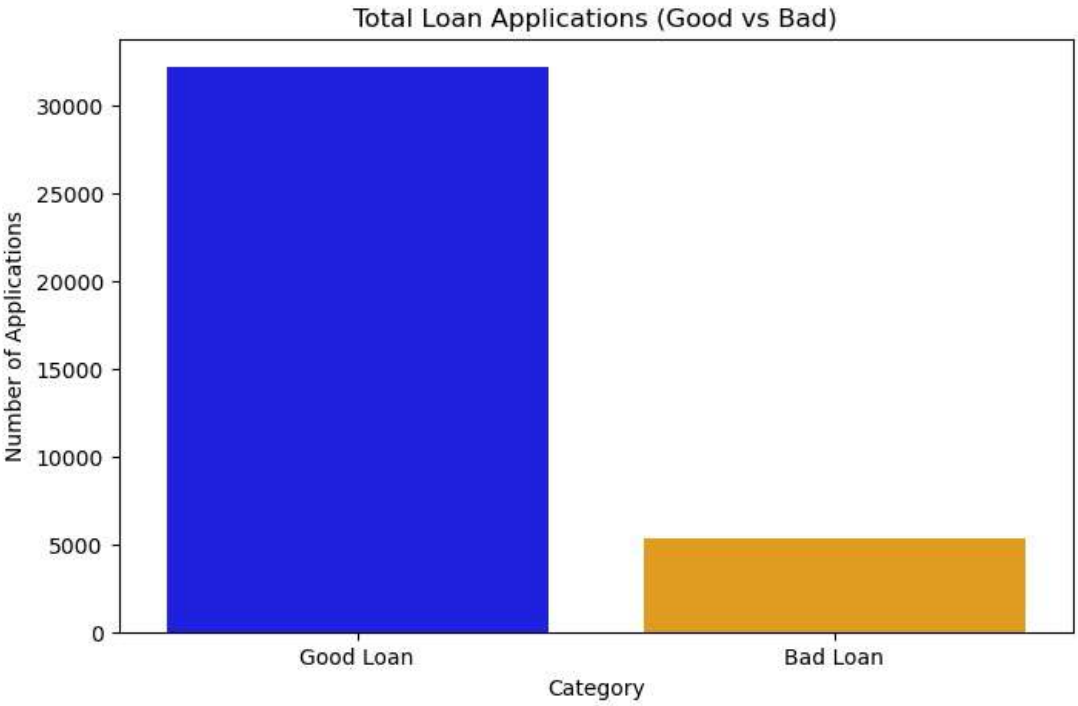
In [66]:
```python
# 1. Pie Chart of Good vs. Bad Loan Applications (Count)
labels = ['Good Loans', 'Bad Loans']
sizes = [good_loan_apps, bad_loan_apps]
colors = ['lightgreen', 'lightcoral']
plt.figure(figsize=(6, 4))
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=90)
plt.title('Distribution of Good vs. Bad Loan Applications (Count)')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```



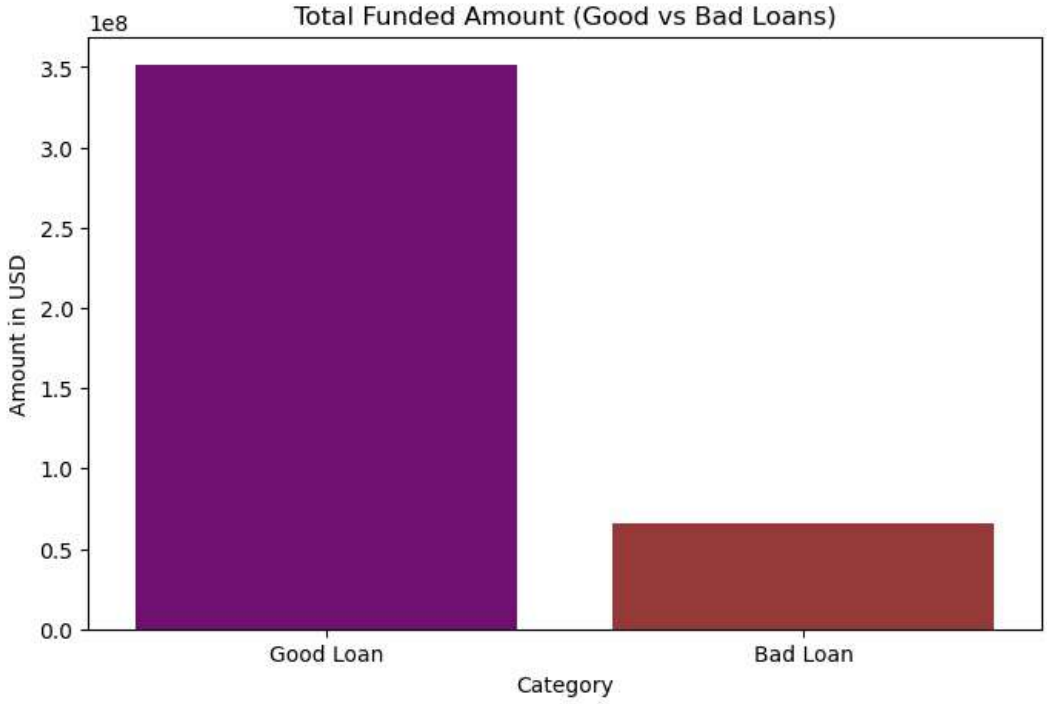
In [67]:
```python
# 2. Loan Application Percentage
plt.figure(figsize=(8,5))
sns.barplot(x="Category", y="Application Percentage", data=summary_df, palette=["green", "red"])
plt.title("Loan Application Percentage (Good vs Bad)")
plt.ylabel("Percentage")
plt.show()
```



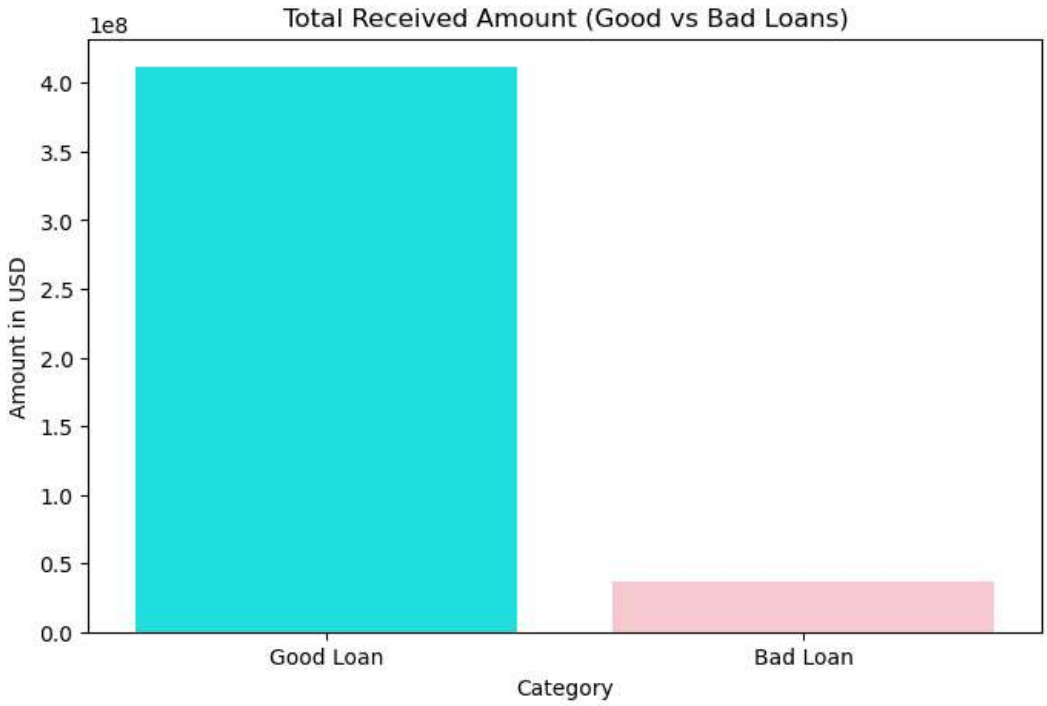
In [68]:
```
# 3. Total Loan Applications
plt.figure(figsize=(8,5))
sns.barplot(x="Category", y="Applications", data=summary_df, palette=["blue", "orange"])
plt.title("Total Loan Applications (Good vs Bad)")
plt.ylabel("Number of Applications")
plt.show()
```



In [69]:
```
# 4. Funded Amount
plt.figure(figsize=(8,5))
sns.barplot(x="Category", y="Funded Amount", data=summary_df, palette=["purple", "brown"])
plt.title("Total Funded Amount (Good vs Bad Loans)")
plt.ylabel("Amount in USD")
plt.show()
```



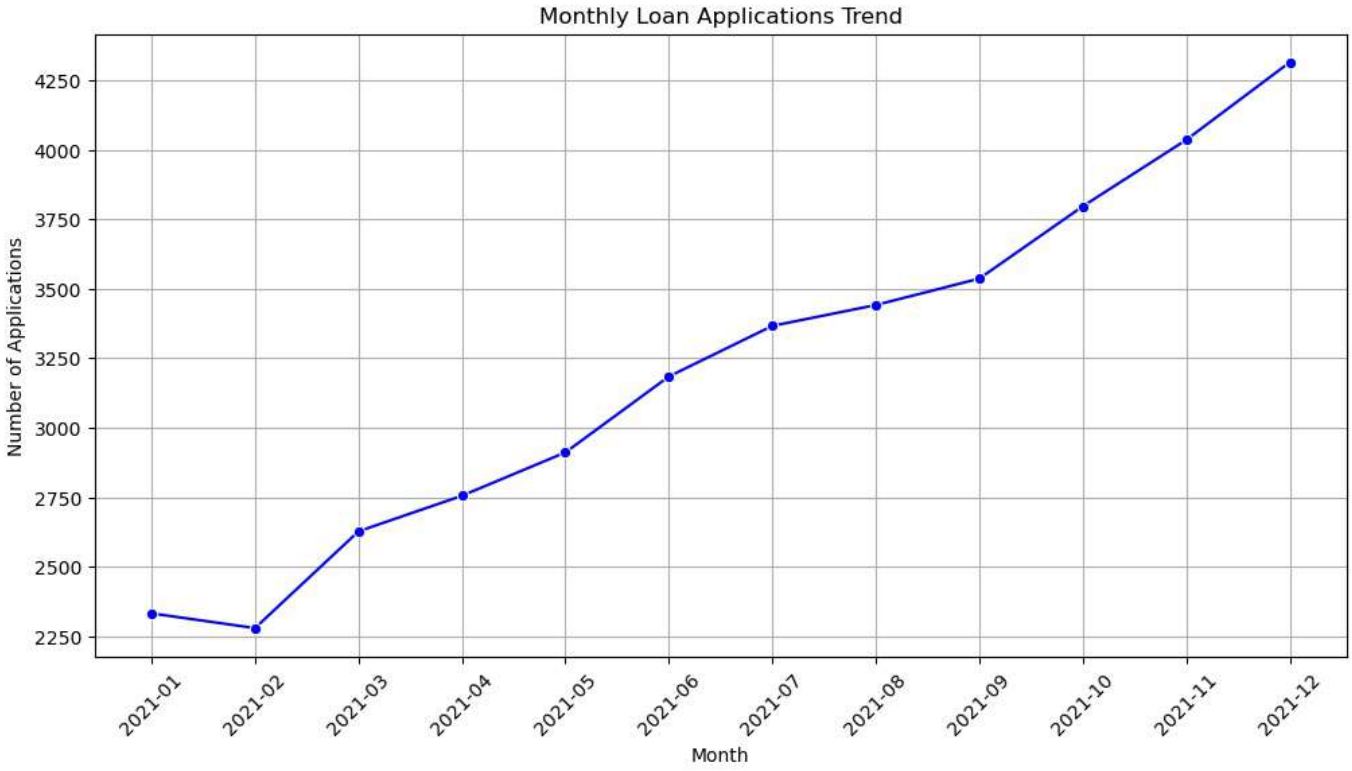
In [70]:
```
# 5. Total Received Amount
plt.figure(figsize=(8,5))
sns.barplot(x="Category", y="Total Received Amount", data=summary_df, palette=["cyan", "pink"])
plt.title("Total Received Amount (Good vs Bad Loans)")
plt.ylabel("Amount in USD")
plt.show()
```



Chart's Requirement:

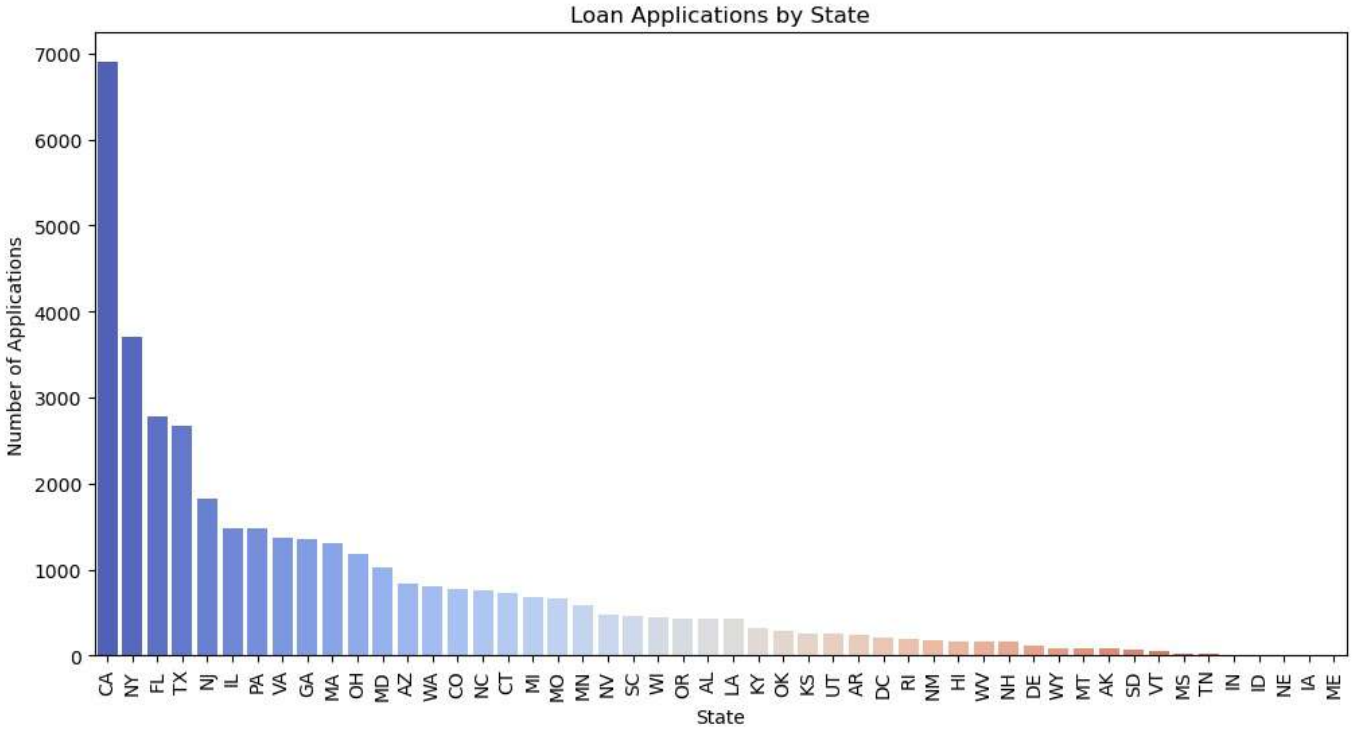
- **1. Monthly Trends by Issue Date (Line Chart): To identify seasonality and long-term trends in lending activities.**

In [72]:
```
# 1. Monthly Trends by Issue Date
monthly_trends = df['issue_month'].value_counts().sort_index()

plt.figure(figsize=(12,6))
sns.lineplot(x=monthly_trends.index.astype(str), y=monthly_trends.values, marker="o", color="blue")
plt.xticks(rotation=45)
plt.title("Monthly Loan Applications Trend")
plt.xlabel("Month")
plt.ylabel("Number of Applications")
plt.grid()
plt.show()
```



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

In [74]:
```
# 2. Regional Analysis by State
plt.figure(figsize=(12,6))
state_counts = df["address_state"].value_counts()
sns.barplot(x=state_counts.index, y=state_counts.values, palette="coolwarm")
plt.xticks(rotation=90)
plt.title("Loan Applications by State")
plt.ylabel("Number of Applications")
plt.xlabel("State")
plt.show()
```



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

In [76]:
```
df['term']
```

Out[76]:
```
0         60 months
1         36 months
2         36 months
3         60 months
4         36 months
            ...
38571     60 months
38572     60 months
38573     60 months
38574     60 months
38575     60 months
Name: term, Length: 38576, dtype: object
```

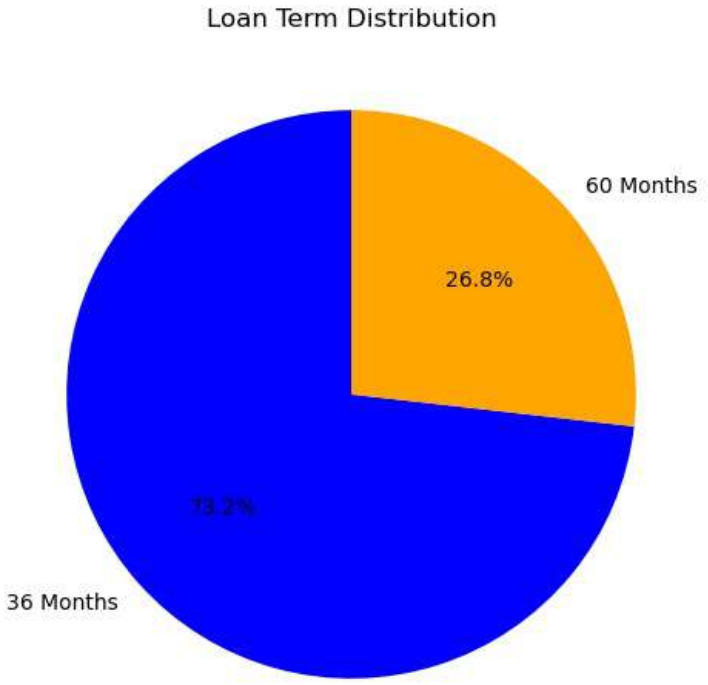
In [77]:
```
df['term'] = df['term'].astype(str)
df['term']
```

Out[77]:
```
0         60 months
1         36 months
2         36 months
3         60 months
4         36 months
            ...
38571     60 months
38572     60 months
38573     60 months
38574     60 months
38575     60 months
Name: term, Length: 38576, dtype: object
```

In [78]:
```
df['term'] = df['term'].str.replace(" months", "")
df['term']
```

Out[78]:
```
0         60
1         36
2         36
3         60
4         36
        ...
38571     60
38572     60
38573     60
38574     60
38575     60
Name: term, Length: 38576, dtype: object
```

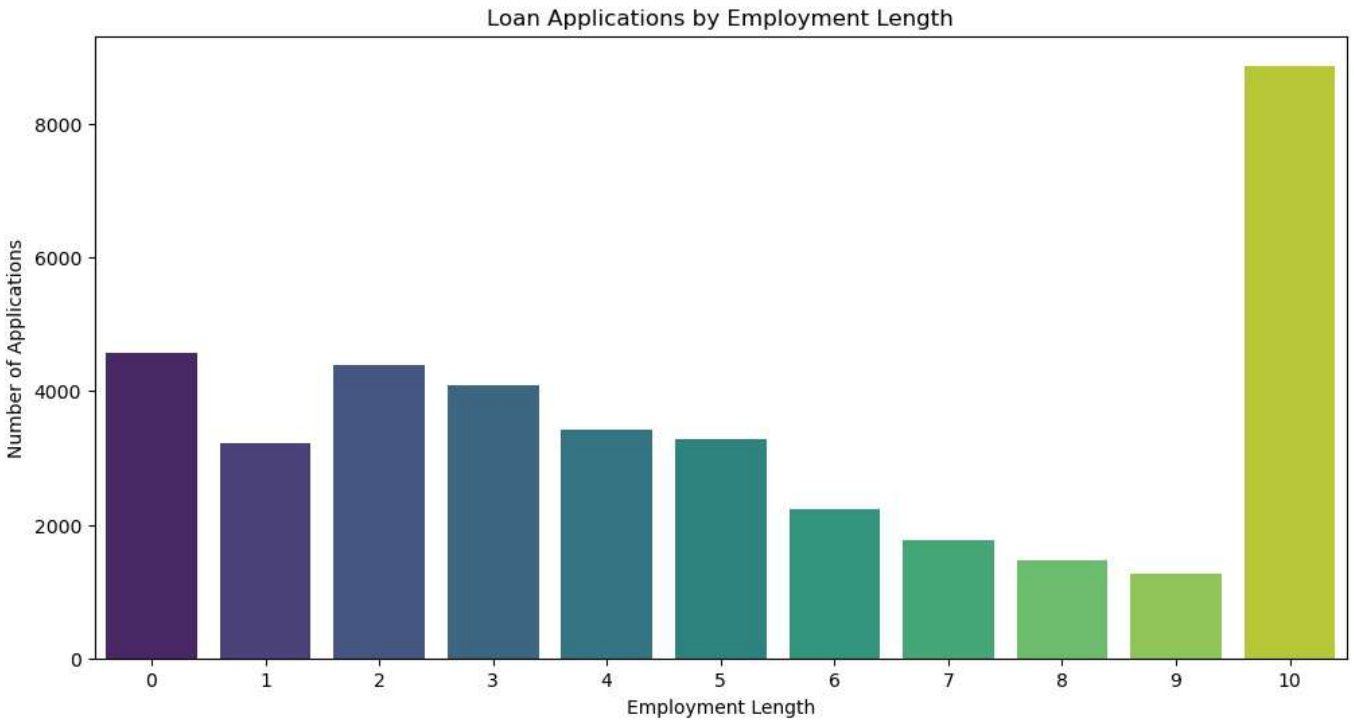
In [79]:
```
plt.figure(figsize=(6,6))
term_counts = df['term'].value_counts()
plt.pie(term_counts, labels=["36 Months", "60 Months"], autopct='%1.1f%%', colors=["blue", "orange"], startangl
plt.title("Loan Term Distribution")
plt.show()
```



- **4. Employee Length Analysis: How lending metrics are distributed among borrowers with different employment lengths,**

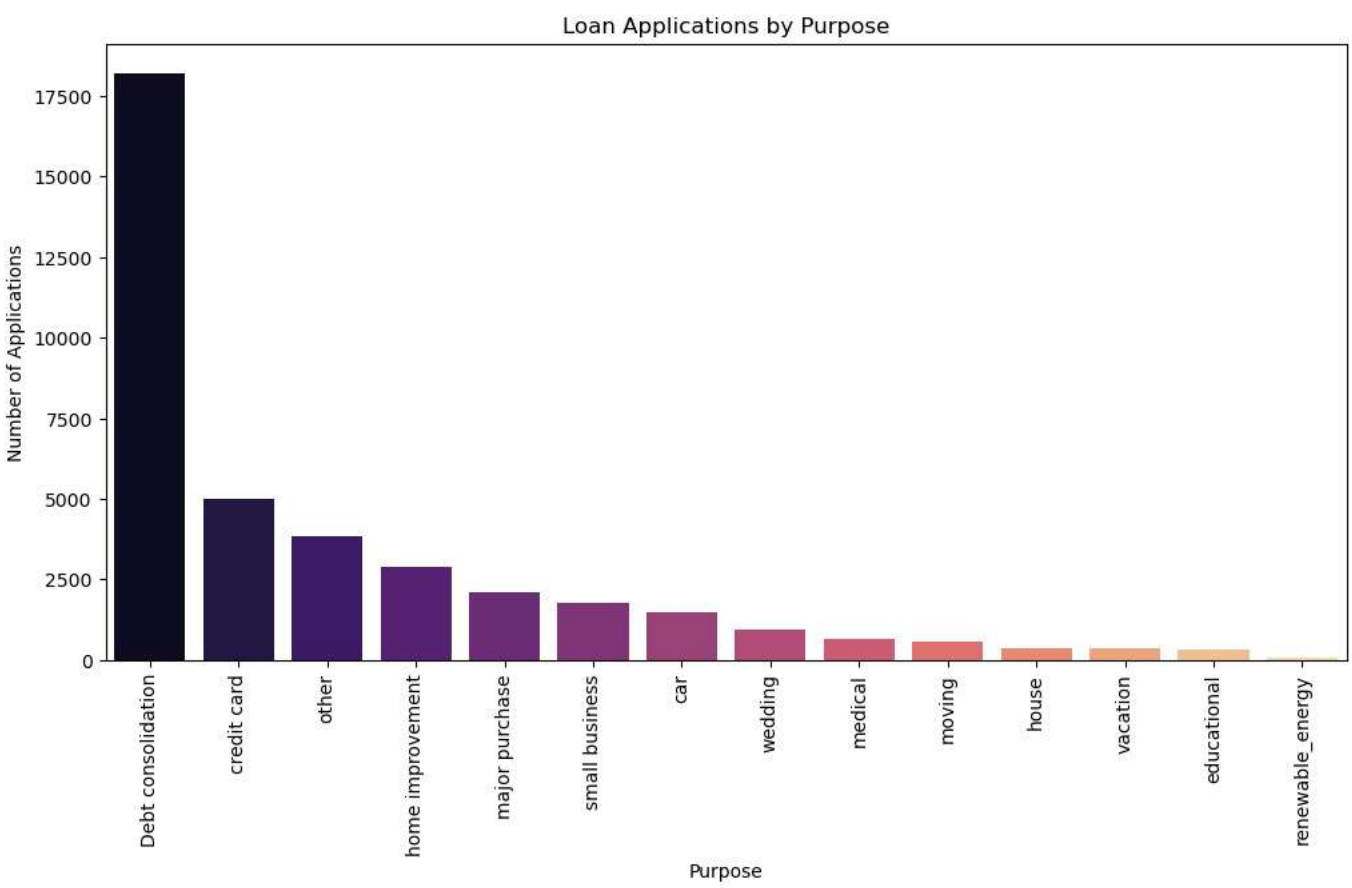
helping us assess the impact of employment history on loan applications.

In [81]:
```
# 4. Employment Length Analysis
plt.figure(figsize=(12,6))
emp_counts = df["emp_length"].value_counts().sort_index()
sns.barplot(x=emp_counts.index, y=emp_counts.values, palette="viridis")
plt.title("Loan Applications by Employment Length")
plt.ylabel("Number of Applications")
plt.xlabel("Employment Length")
plt.show()
```



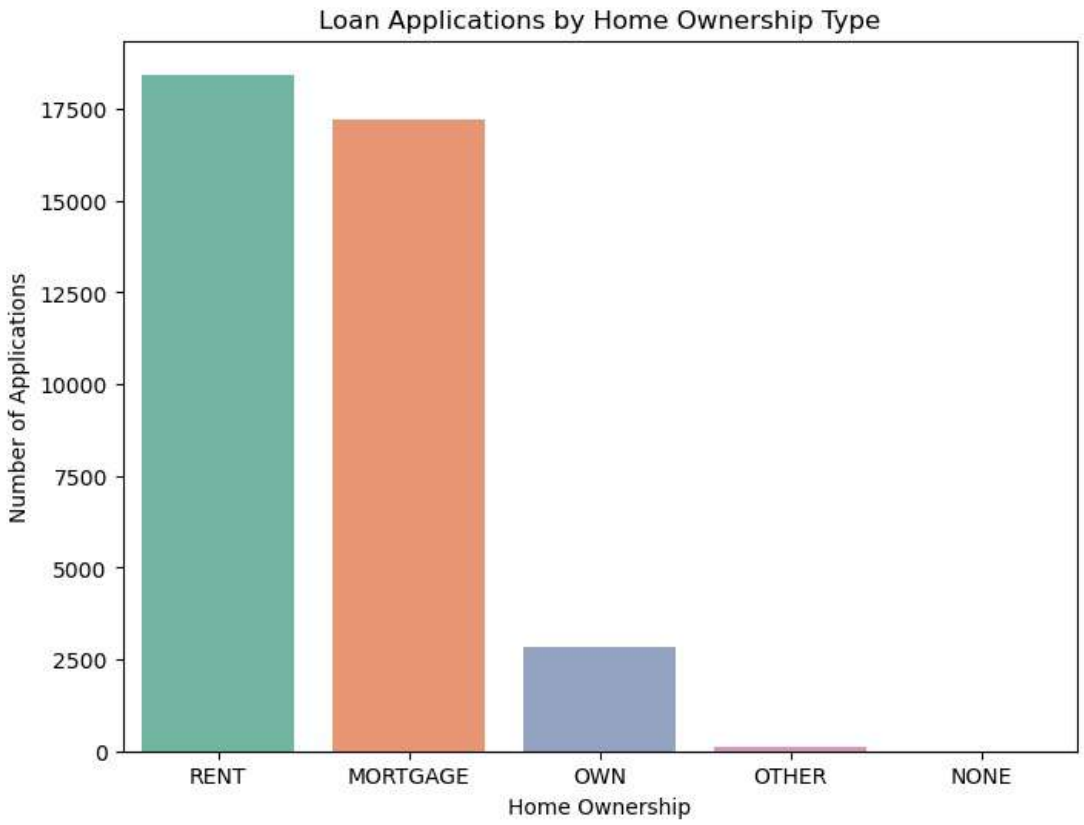
- **5. Loan Purpose Breakdown: Will provide a visual breakdown of loan metrics based on the stated purposes of loans, aiding in the understanding of the primary reasons borrowers seek financing.**

In [83]:
```
# 5. Loan Purpose Breakdown
plt.figure(figsize=(12,6))
purpose_counts = df["purpose"].value_counts()
sns.barplot(x=purpose_counts.index, y=purpose_counts.values, palette="magma")
plt.xticks(rotation=90)
plt.title("Loan Applications by Purpose")
plt.ylabel("Number of Applications")
plt.xlabel("Purpose")
plt.show()
```



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

In [85]:
```
# 6. Home Ownership Analysis
plt.figure(figsize=(8,6))
home_counts = df["home_ownership"].value_counts()
sns.barplot(x=home_counts.index, y=home_counts.values, palette="Set2")
plt.title("Loan Applications by Home Ownership Type")
plt.ylabel("Number of Applications")
plt.xlabel("Home Ownership")
plt.show()
```



In [86]:
```
df.columns
```

Out[86]:
```
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', 'total_acc_Cus', 'emp_length_Cus', 'Annual_income_Cus',
       'Installments_Cus', 'DTI_Cus', 'int_rate_Cus', 'loan_amount_Cus',
       'total_payment_Cus', 'issue_month'],
      dtype='object')
```

Plot's for Continous Data

1. **Univariate (Single Variable)**

- Histogram
- Kde plot
- Boxplot

2. **Bivariate (plot between two Variables)**

- Scatter plot
- Line plot
- Join plot
- Violin plot

3. **Multivariate (More than 2 Variables)**

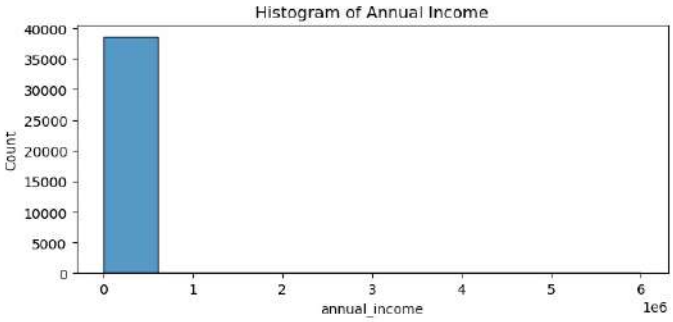
- Scatter Plot (2 continous +1 Discrete)
- Pair plot
- Heatmap

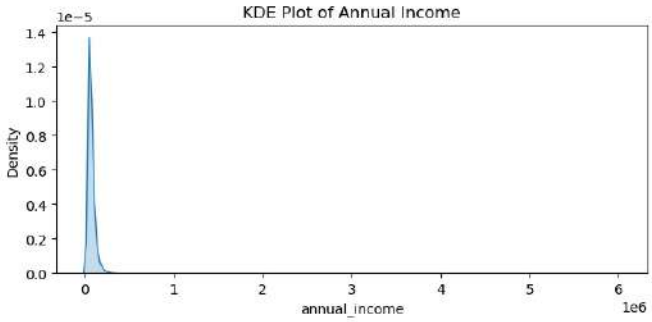
In [88]:
```
df[continuous].columns.to_list()
```

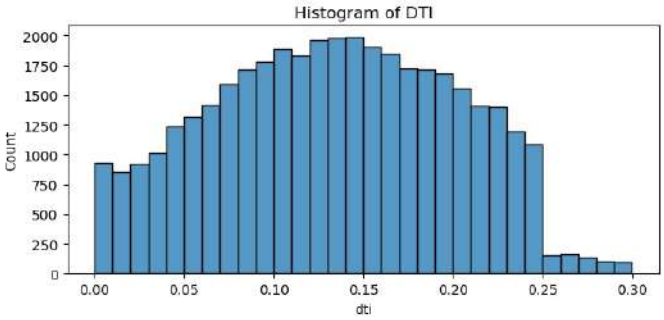
Out[88]:
```
['annual_income',
 'dti',
 'installment',
 'int_rate',
 'loan_amount',
 'total_payment']
```

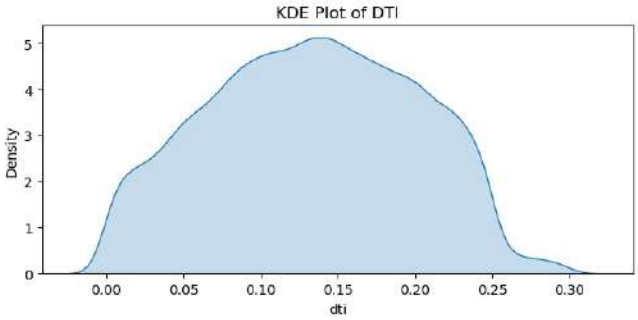
In [89]:
```
# Create a figure with a specified size
plt.figure(figsize=(14, 20))

# Plot each histogram and KDE separately using subplot()

# Annual Income
plt.subplot(6, 2, 1)
sns.histplot(df['annual_income'], bins=10)
plt.title("Histogram of Annual Income")

plt.subplot(6, 2, 2)
sns.kdeplot(df['annual_income'], fill=True)
plt.title("KDE Plot of Annual Income")

# DTI
plt.subplot(6, 2, 3)
sns.histplot(df['dti'], bins=30)
plt.title("Histogram of DTI")

plt.subplot(6, 2, 4)
sns.kdeplot(df['dti'], fill=True)
plt.title("KDE Plot of DTI")

# Installment
plt.subplot(6, 2, 5)
sns.histplot(df['installment'], bins=30)
plt.title("Histogram of Installment")

plt.subplot(6, 2, 6)
sns.kdeplot(df['installment'], fill=True)
plt.title("KDE Plot of Installment")

# Interest Rate
plt.subplot(6, 2, 7)
sns.histplot(df['int_rate'], bins=30)
plt.title("Histogram of Interest Rate")

plt.subplot(6, 2, 8)
```

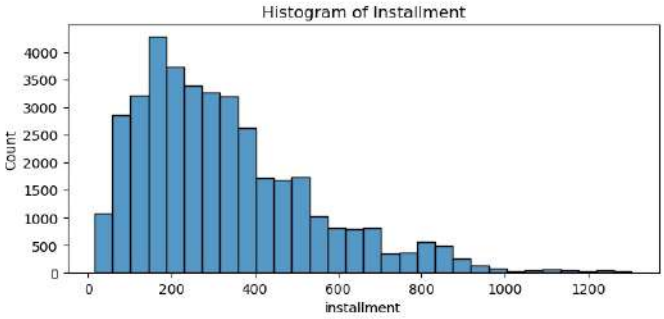
```
sns.kdeplot(df['int_rate'], fill=True)
plt.title("KDE Plot of Interest Rate")

# Loan Amount
plt.subplot(6, 2, 9)
sns.histplot(df['loan_amount'], bins=30)
plt.title("Histogram of Loan Amount")

plt.subplot(6, 2, 10)
sns.kdeplot(df['loan_amount'], fill=True)
plt.title("KDE Plot of Loan Amount")

# Total Payment
plt.subplot(6, 2, 11)
sns.histplot(df['total_payment'], bins=30)
plt.title("Histogram of Total Payment")

plt.subplot(6, 2, 12)
sns.kdeplot(df['total_payment'], fill=True)
plt.title("KDE Plot of Total Payment")

# Adjust layout and show the plots
plt.tight_layout()
plt.show()
```

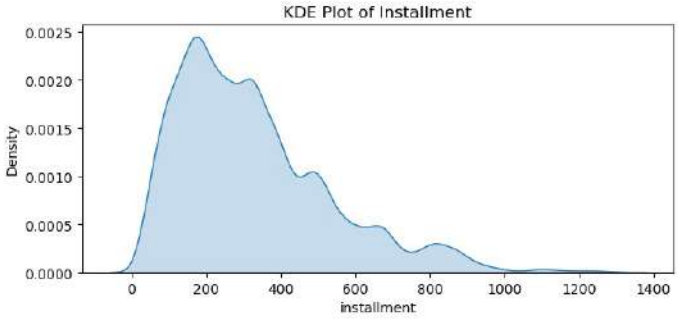


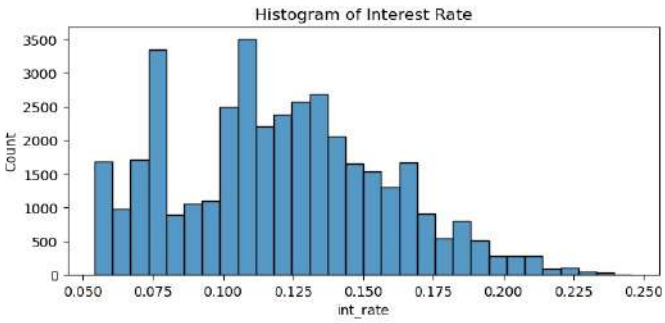


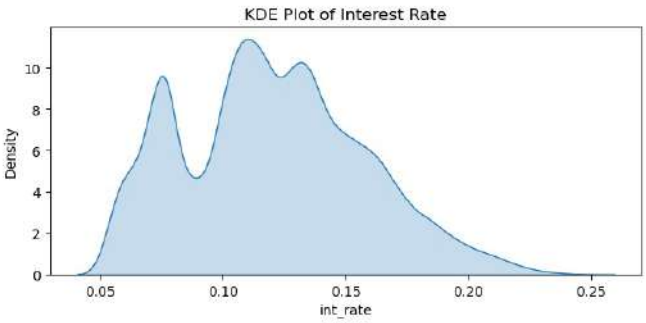


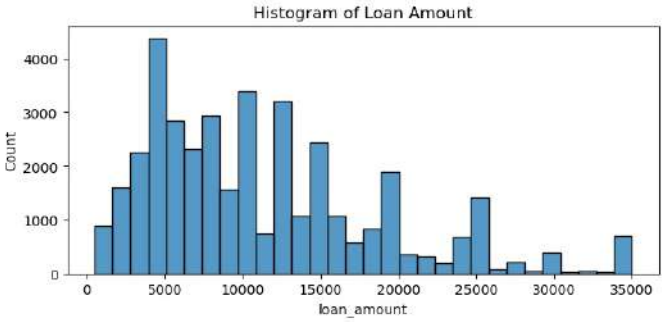


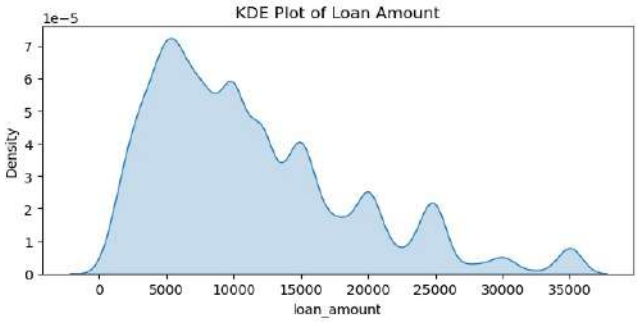


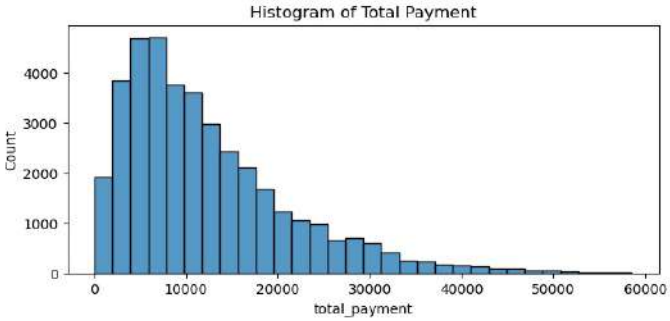


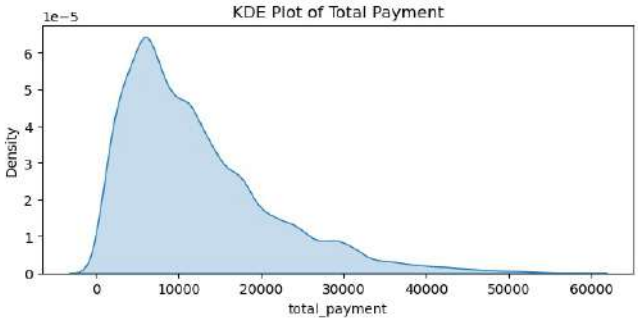












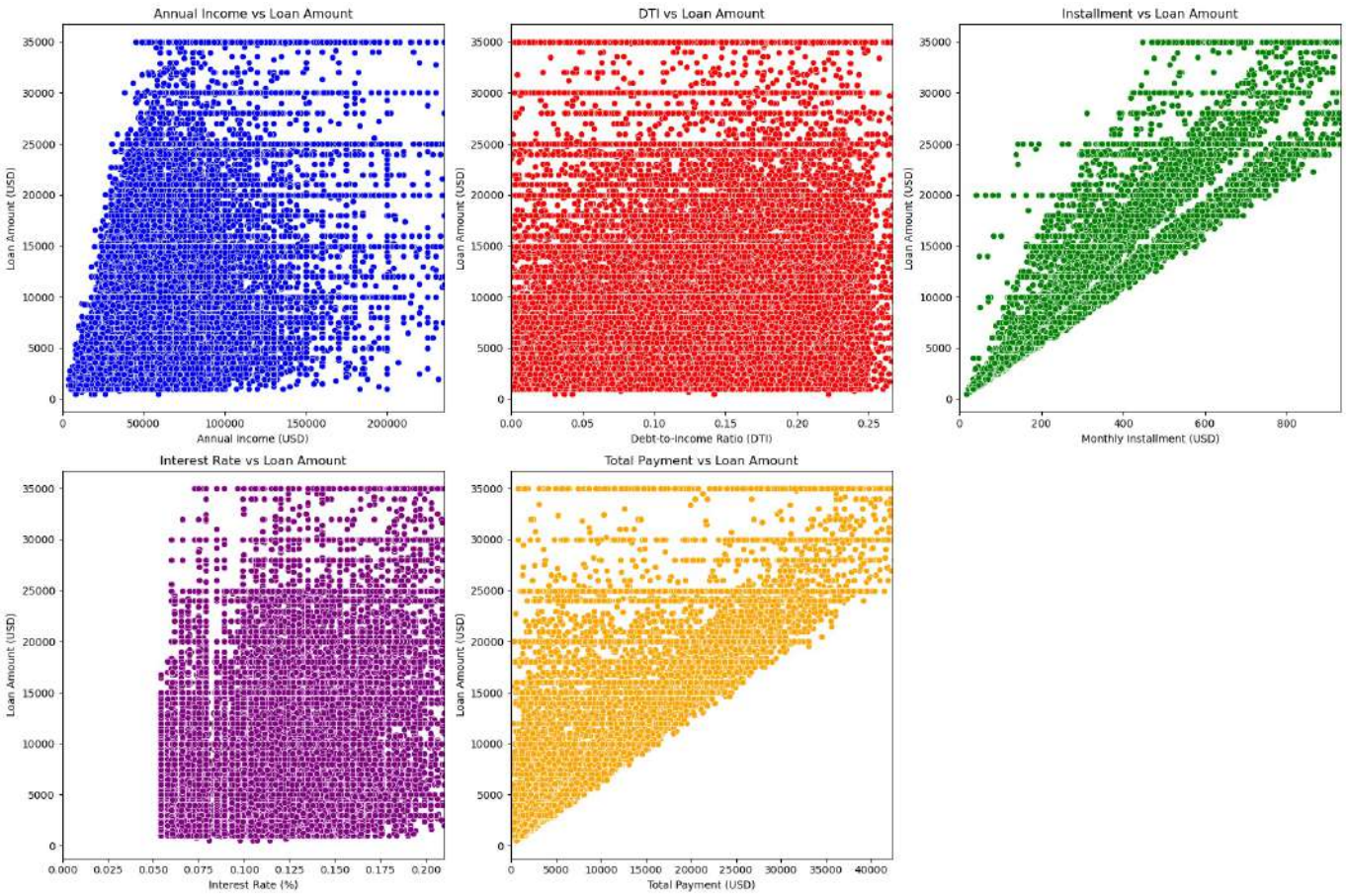
Observations

- In the AnnualIncome plot **The distribution is right-skewed, indicating that most borrowers have lower income levels while a few have very high incomes.**
- **The DTI distribution appears slightly right-skewed, meaning some borrowers have very high debt burdens relative to**

income, they fall in range of 0.23 to 0.3

- In the Installment plot **The histogram shows a peak in lower values(0 - 400), indicating that many borrowers have manageable monthly payments, A small portion of borrowers have high installment payments(400 - 1200), which could indicate larger loans or higher interest rates.**
- In the interest rate plot **The interest rate distribution shows a peak in the lower ranges, meaning many borrowers received lower interest rates.**
- In Loan Amount plot **Most loans are for smaller amounts, with a gradual decline in the number of large loans.**
- In Total_payment **The distribution is right-skewed, meaning most of the borrowers payed back with less interest, only few payed back more interest loans**

In [91]:
```
# Create a single figure with subplots (2 rows, 3 columns)
plt.figure(figsize=(18, 12))

# Scatter Plot: Annual Income vs Loan Amount
plt.subplot(2, 3, 1)
sns.scatterplot(x=df['annual_income'], y=df['loan_amount'],color="blue")
plt.title("Annual Income vs Loan Amount")
plt.xlabel("Annual Income (USD)")
plt.ylabel("Loan Amount (USD)")
plt.xlim(0, df['annual_income'].quantile(0.99))

# Scatter Plot: DTI vs Loan Amount
plt.subplot(2, 3, 2)
sns.scatterplot(x=df['dti'], y=df['loan_amount'],  color="red")
plt.title("DTI vs Loan Amount")
plt.xlabel("Debt-to-Income Ratio (DTI)")
plt.ylabel("Loan Amount (USD)")
plt.xlim(0, df['dti'].quantile(0.99))

# Scatter Plot: Installment vs Loan Amount
plt.subplot(2, 3, 3)
sns.scatterplot(x=df['installment'], y=df['loan_amount'],  color="green")
plt.title("Installment vs Loan Amount")
plt.xlabel("Monthly Installment (USD)")
plt.ylabel("Loan Amount (USD)")
plt.xlim(0, df['installment'].quantile(0.99))

# Scatter Plot: Interest Rate vs Loan Amount
plt.subplot(2, 3, 4)
sns.scatterplot(x=df['int_rate'], y=df['loan_amount'],  color="purple")
plt.title("Interest Rate vs Loan Amount")
plt.xlabel("Interest Rate (%)")
plt.ylabel("Loan Amount (USD)")
plt.xlim(0, df['int_rate'].quantile(0.99))

# Scatter Plot: Total Payment vs Loan Amount
plt.subplot(2, 3, 5)
sns.scatterplot(x=df['total_payment'], y=df['loan_amount'], color="orange")
plt.title("Total Payment vs Loan Amount")
plt.xlabel("Total Payment (USD)")
plt.ylabel("Loan Amount (USD)")
plt.xlim(0, df['total_payment'].quantile(0.99))

# Adjust layout and show the plots
plt.tight_layout()
plt.show()
```



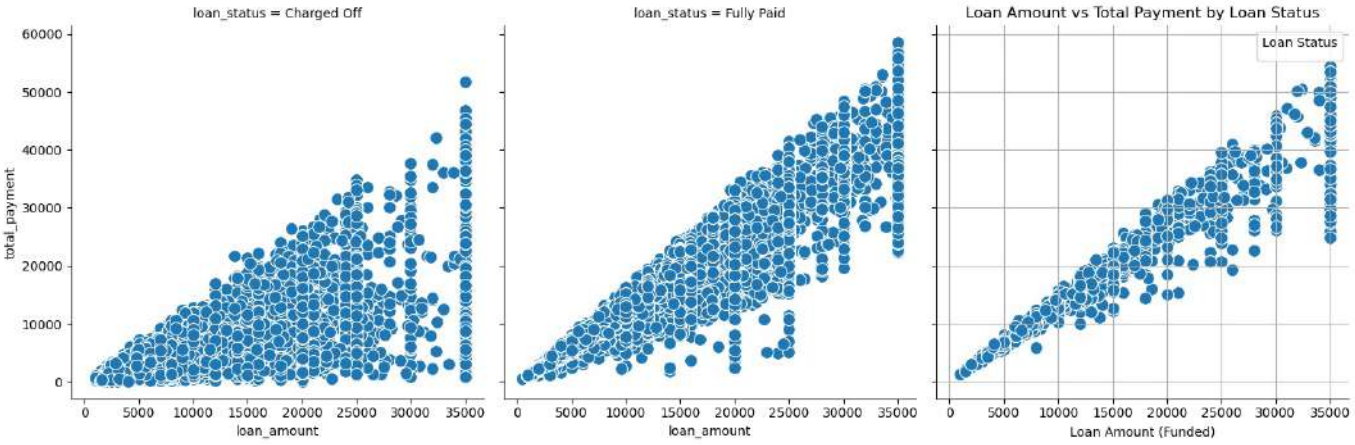
Observations

- **There is a positive correlation between annual income and loan amount, but it is not perfectly linear and Most borrowers have moderate incomes, and few high-income individuals take large loans**
- **A strong positive correlation exists higher loan amounts typically result in higher monthly installments, Some borrowers have high loan amounts but relatively small installments, suggesting longer loan terms.**
- **There is a linear correlation larger loan amounts generally lead to higher total payments.**

In [93]:
```
# Create scatter plot
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='loan_amount', y='total_payment', hue='loan_status', palette=['red', 'green', 'blue'
plt.title('Loan Amount vs Total Payment by Loan Status')
plt.xlabel('Loan Amount (Funded)')
plt.ylabel('Total Payment (Received)')
plt.legend(title='Loan Status')
plt.grid(True)
plt.show()
```

In [94]:
```
# Create scatter plot
plt.figure(figsize=(10, 6))
sns.relplot(data=df, x='loan_amount', y='total_payment', col='loan_status', s=100)
plt.title('Loan Amount vs Total Payment by Loan Status')
plt.xlabel('Loan Amount (Funded)')
plt.ylabel('Total Payment (Received)')
plt.legend(title='Loan Status')
plt.grid(True)
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignor
ed when legend() is called with no argument.
<Figure size 1000x600 with 0 Axes>
```



Observations

- **Good loans (green) cluster tightly above the 45-degree line with consistent repayment plus interest, while bad loans (red) show wide variability below the line, with larger loans like 12000 indicating higher risk and significant losses.**

In [96]:
```
df[discrete_categorical].columns.to_list()
```

Out[96]:
```
['address_state',
 'application_type',
 'emp_title',
 'grade',
 'home_ownership',
 'loan_status',
 'purpose',
 'sub_grade',
 'term',
 'verification_status']
```

Plot's for Discrete Data

1. **Univariate (Single Variable)**

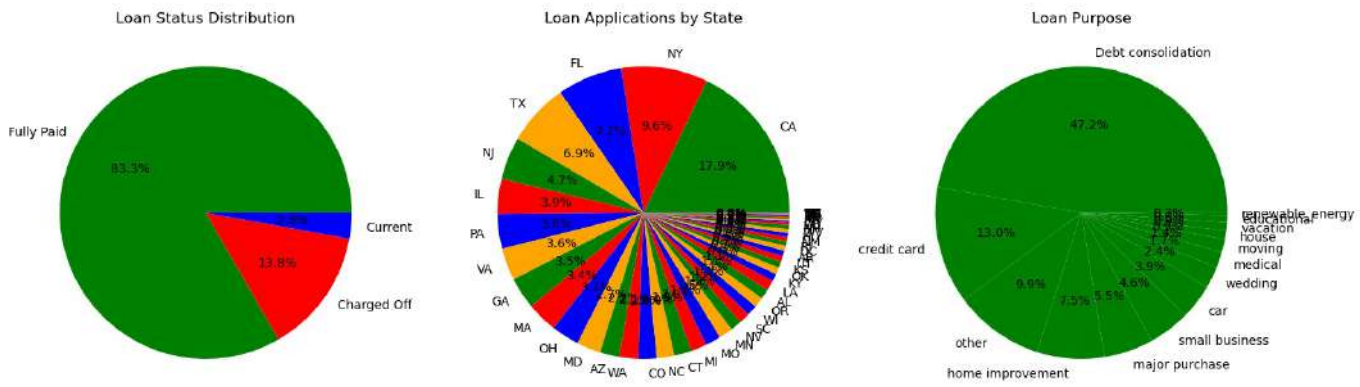
- Pie plot
- Bar plot
- Countplot

2. **Bivariate (plot between two Variables)**

- Boxplot -> one discrete variable & one continous variable

In [98]:
```
# Create a figure with subplots (1 row, 3 columns)
plt.figure(figsize=(18, 6))  # Wide figure to accommodate 3 pie charts

# 1. Pie Plot for Loan Status
plt.subplot(1, 3, 1)  # 1 row, 3 columns, 1st position
df['loan_status'].value_counts().plot.pie(autopct='%1.1f%%', colors=['green', 'red','blue'])
plt.title('Loan Status Distribution')
plt.ylabel('')  # Remove y-label for pie chart

# 2. Pie Plot for Address State
plt.subplot(1, 3, 2)  # 1 row, 3 columns, 2nd position
df['address_state'].value_counts().plot.pie(autopct='%1.1f%%', colors=['green', 'red', 'blue', 'orange'])
plt.title('Loan Applications by State')
plt.ylabel('')

# 3. Pie Plot for Purpose
plt.subplot(1, 3, 3)  # 1 row, 3 columns, 3rd position
df['purpose'].value_counts().plot.pie(autopct='%1.1f%%', colors=['green'])
plt.title('Loan Purpose')
plt.ylabel('')
```

Out[98]:
```
Text(0, 0.5, '')
```

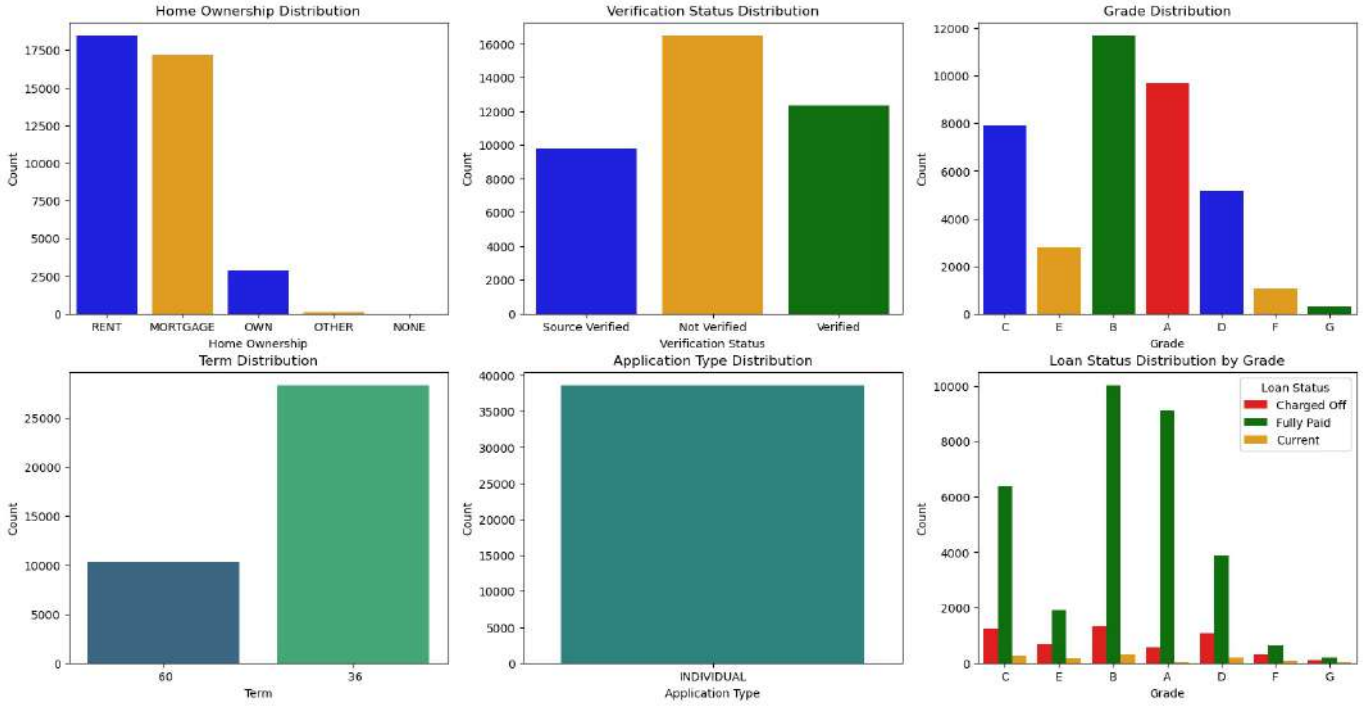


Observations

- **The Fully paid is largest segment so there is high repayment of loans but Charged Off is also significant, it suggests higher credit risk,**
- **Certain states like CA, NY, FL, TX have higher loan applications, indicating regions with higher borrowing demand.**
- **Most of the the borrowers used the loan for Debt consolidation followed by Credit card and home improvement.**

In [100…
```
# Create a figure with subplots (2 rows, 3 columns)
plt.figure(figsize=(20, 10))

# 1. Bar Plot for Home Ownership
plt.subplot(2, 3, 1)
sns.countplot(data=df, x='home_ownership', palette=['blue', 'orange'])
plt.title('Home Ownership Distribution')
plt.xlabel('Home Ownership')
plt.ylabel('Count')

# 2. Bar Plot for Verification Status
plt.subplot(2, 3, 2)
sns.countplot(data=df, x='verification_status', palette=['blue', 'orange', 'green'])
plt.title('Verification Status Distribution')
plt.xlabel('Verification Status')
plt.ylabel('Count')

# 3. Bar Plot for Grade
plt.subplot(2, 3, 3)
```

```
sns.countplot(data=df, x='grade', palette=['blue', 'orange', 'green', 'red'])
plt.title('Grade Distribution')
plt.xlabel('Grade')
plt.ylabel('Count')

# 4. Bar Plot for Term
plt.subplot(2, 3, 4)
sns.countplot(data=df, x='term', palette='viridis')
plt.title('Term Distribution')
plt.xlabel('Term')
plt.ylabel('Count')

# 5. Bar Plot for Application Type
plt.subplot(2, 3, 5)
sns.countplot(data=df, x='application_type', palette='viridis')
plt.title('Application Type Distribution')
plt.xlabel('Application Type')
plt.ylabel('Count')

# 6. Create bar plot
plt.subplot(2, 3, 6)
sns.countplot(data=df, x='grade', hue='loan_status', palette=['red', 'green', 'orange'])
plt.title('Loan Status Distribution by Grade')
plt.xlabel('Grade')
plt.ylabel('Count')
plt.legend(title='Loan Status')
plt.show()

# Adjust layout to prevent overlap
plt.tight_layout()
plt.show()
```



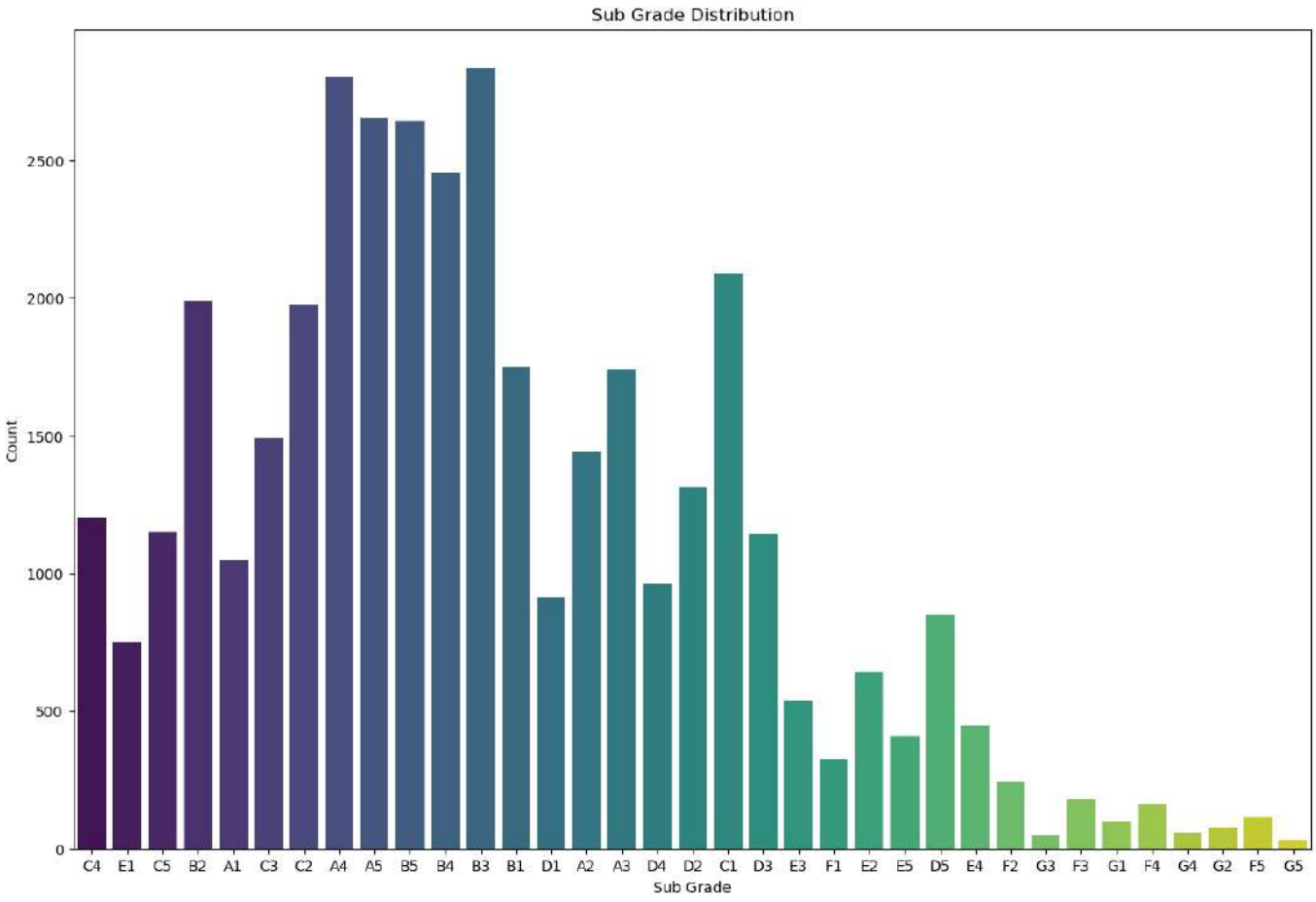

```
<Figure size 640x480 with 0 Axes>
```

In [101…

```
# Bar Plot for Sub Grade
plt.figure(figsize=(15, 10))
sns.countplot(data=df, x='sub_grade', palette='viridis')
plt.title('Sub Grade Distribution')
plt.xlabel('Sub Grade')
plt.ylabel('Count')
```

Out[101…

```
Text(0, 0.5, 'Count')
```

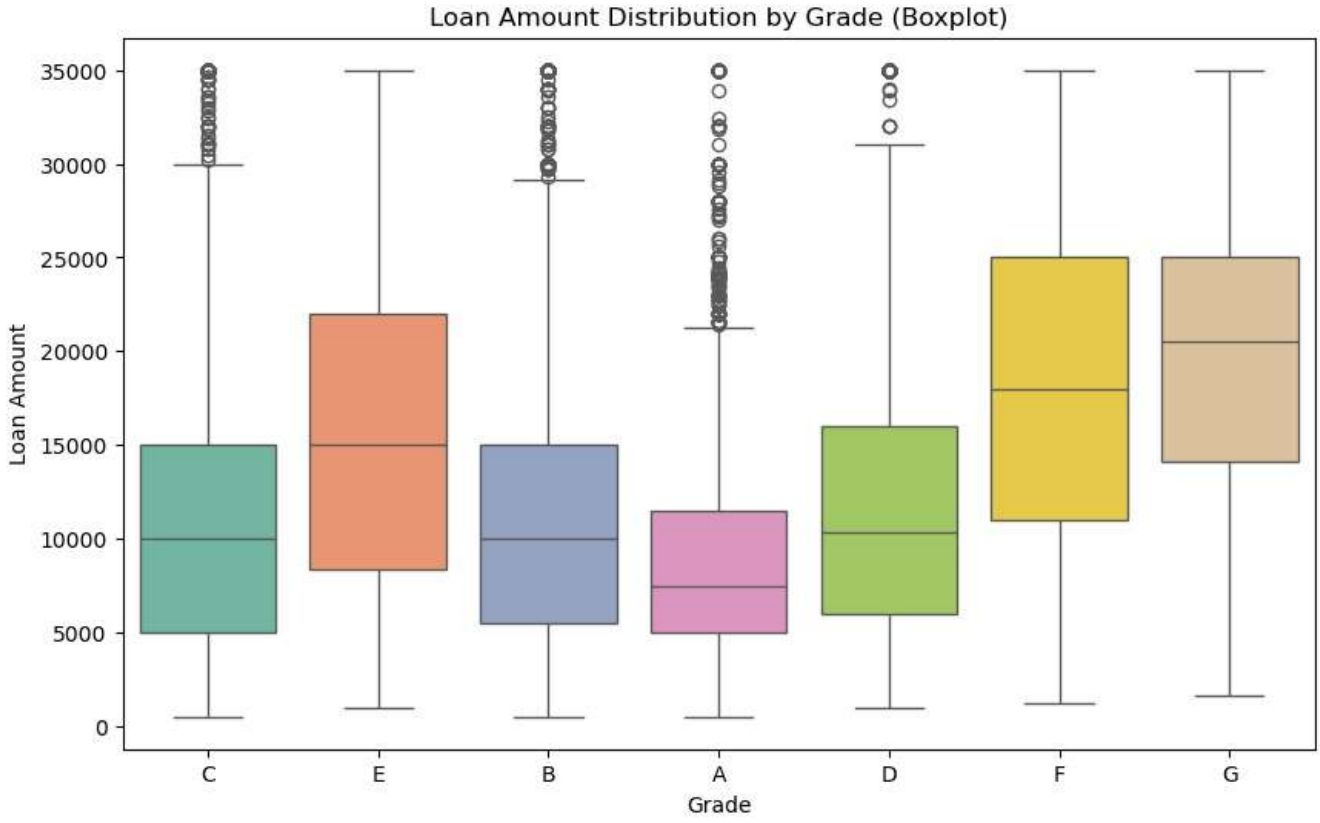


Observations

- **Renters (60%) outnumber mortgage holders (40%), suggesting a higher loan application rate among renters in this plot.**
- **There are many Not Verified status in the borrowers which induces high risk**
- **Majority of Grades are from A,B,C,D.**
- **The 36-month terms (60%) are more frequent than 60-month terms (40%), hinting at a preference for shorter loan durations by borrowers.**
- **Every loan applicant choose Individual loan**

In [103…

```
# Bivariate Plot (Discrete: 'grade', Continuous: 'loan_amount')
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='grade', y='loan_amount', palette='Set2')
plt.title('Loan Amount Distribution by Grade (Boxplot)')
plt.xlabel('Grade')
plt.ylabel('Loan Amount')
plt.show()
```

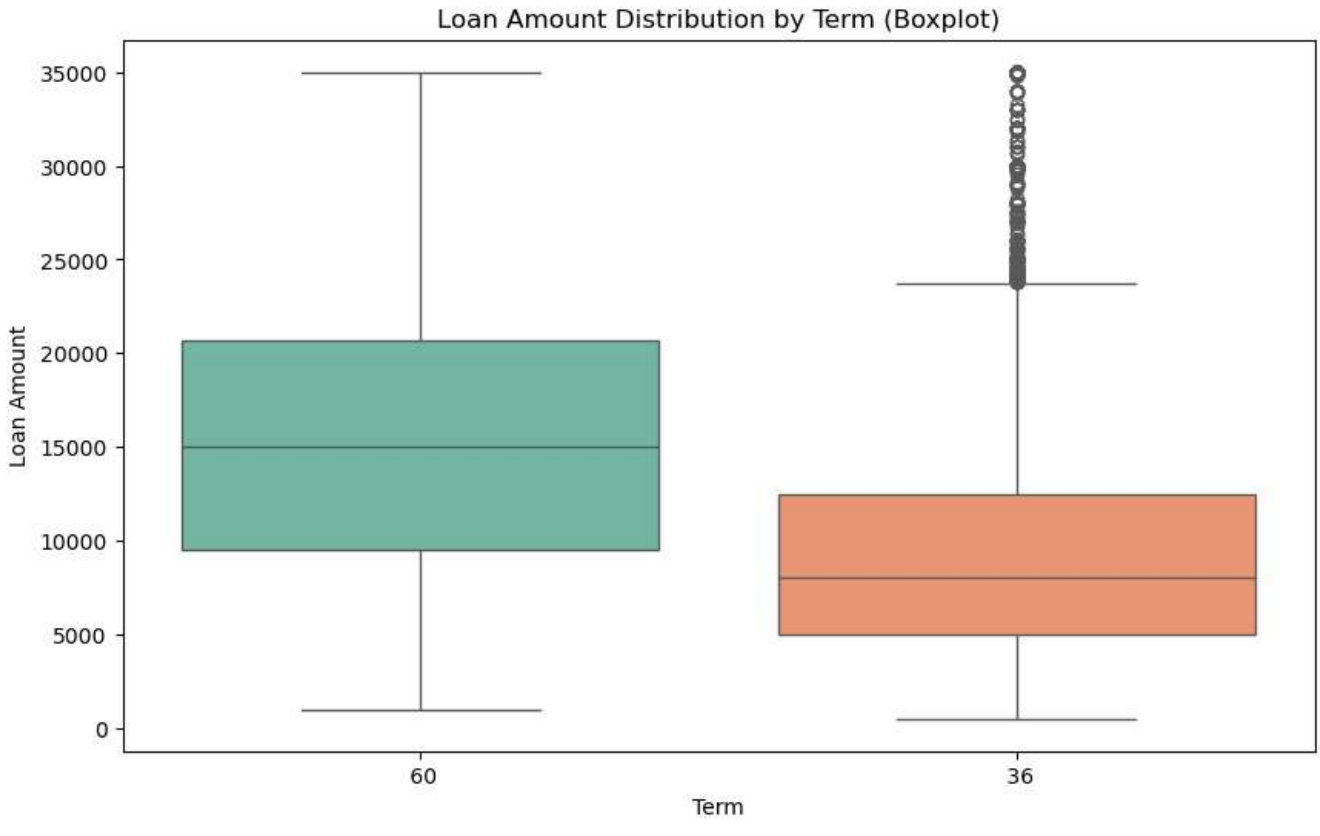


Observations

- **The median loan amount is highest in Grade G and lowest in Grade A**
- **Riskier borrowers (lower grades) tend to have higher loan amounts, possibly due to higher interest rates or greater need for funds.**

In [105…

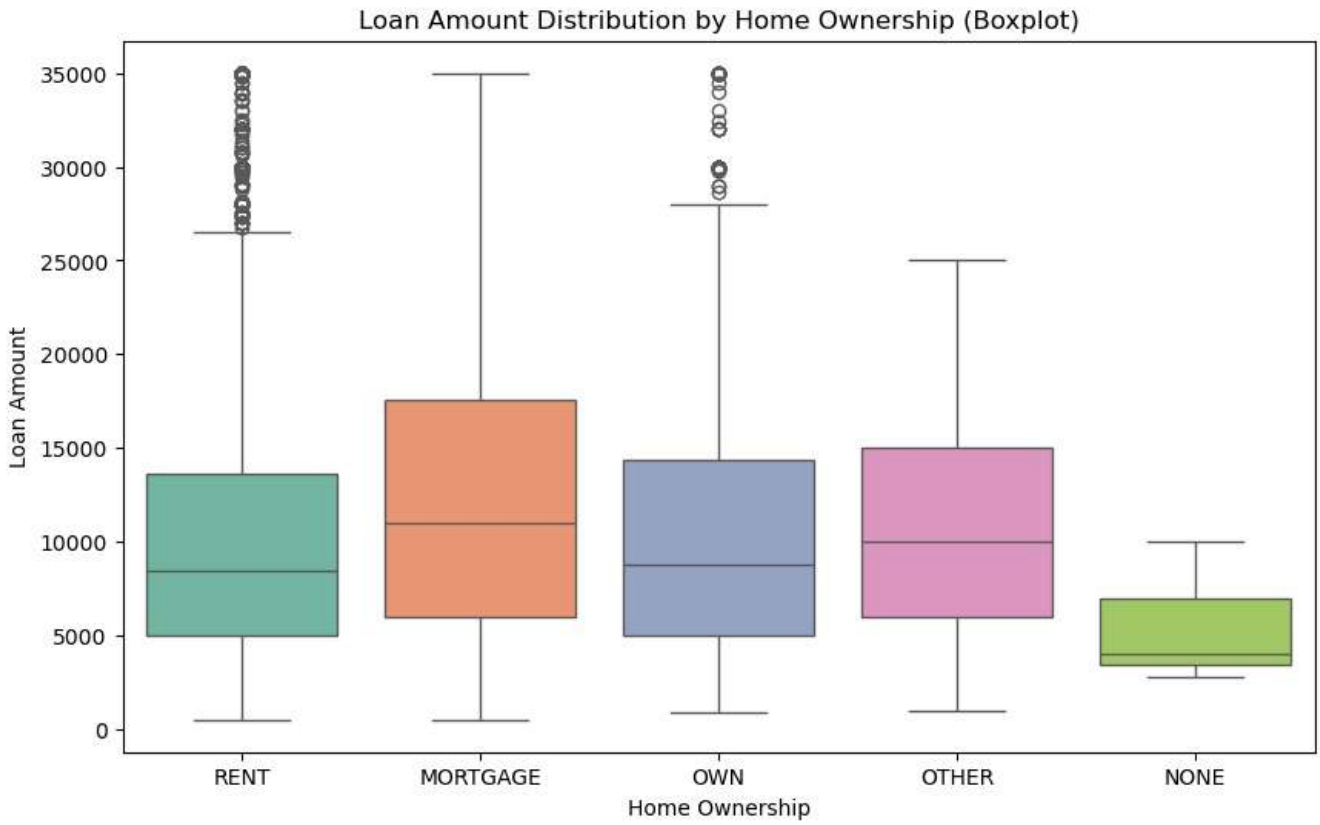
```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='term', y='loan_amount', palette='Set2')
plt.title('Loan Amount Distribution by Term (Boxplot)')
plt.xlabel('Term')
plt.ylabel('Loan Amount')
plt.show()
```



Observations

- **Longer-term loans(60 months) tend to have higher median loan amounts compared to shorter-term loans (36 months).**
- **Longer-term loans correspond to higher loan amounts, which is expected as larger amounts need more time for repayment.**

In [107…
```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='home_ownership', y='loan_amount', palette='Set2')
plt.title('Loan Amount Distribution by Home Ownership (Boxplot)')
plt.xlabel('Home Ownership')
plt.ylabel('Loan Amount')
plt.show()
```



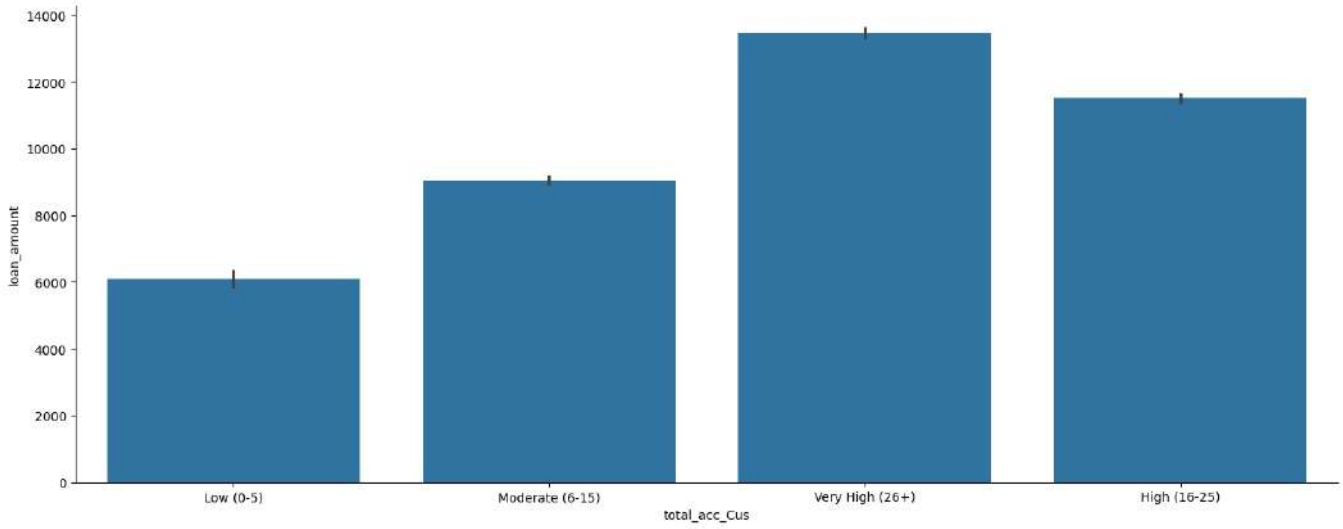
Observations

- **Mortgage holders have the highest median loan amount, meaning they tend to receive larger loans on average.**
- **Owners (who fully own their homes) have a slightly lower median loan amount compared to mortgage holders.**
- **Renters have the lowest median loan amount, suggesting that they either barely qualify for the loan or apply for smaller loans.**

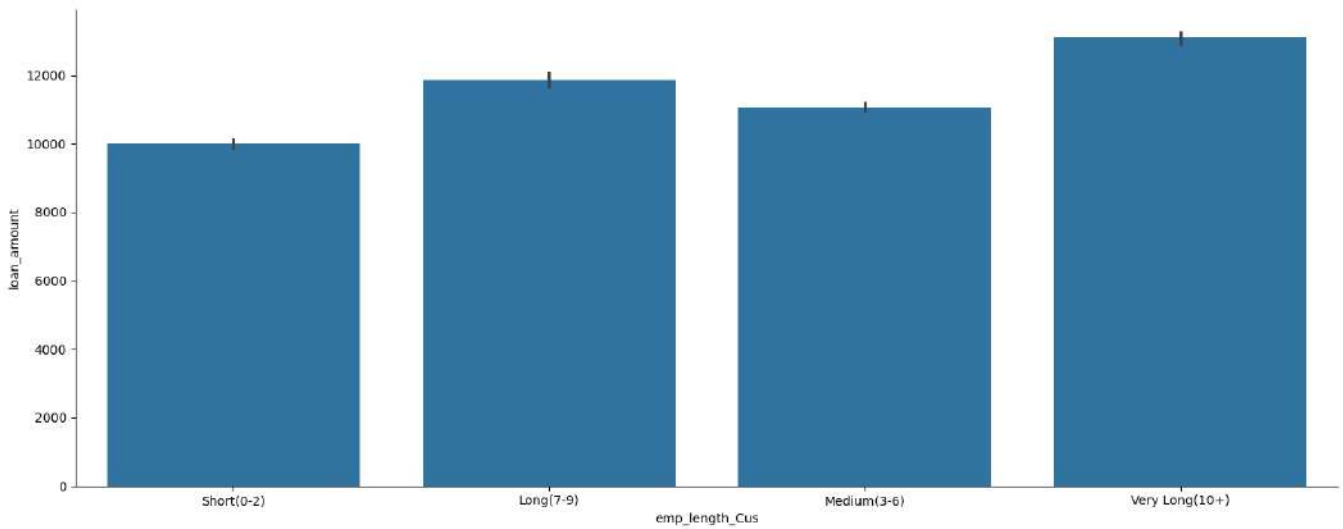
In [109…
```
df.columns.tolist()
```

Out[109…
```
['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',
 'total_acc_Cus',
 'emp_length_Cus',
 'Annual_income_Cus',
 'Installments_Cus',
 'DTI_Cus',
 'int_rate_Cus',
 'loan_amount_Cus',
 'total_payment_Cus',
 'issue_month']
```

In [110…
```
sns.catplot(y="loan_amount", x="total_acc_Cus",  kind="bar", data=df, height=6, aspect=2.5)
plt.show()
```

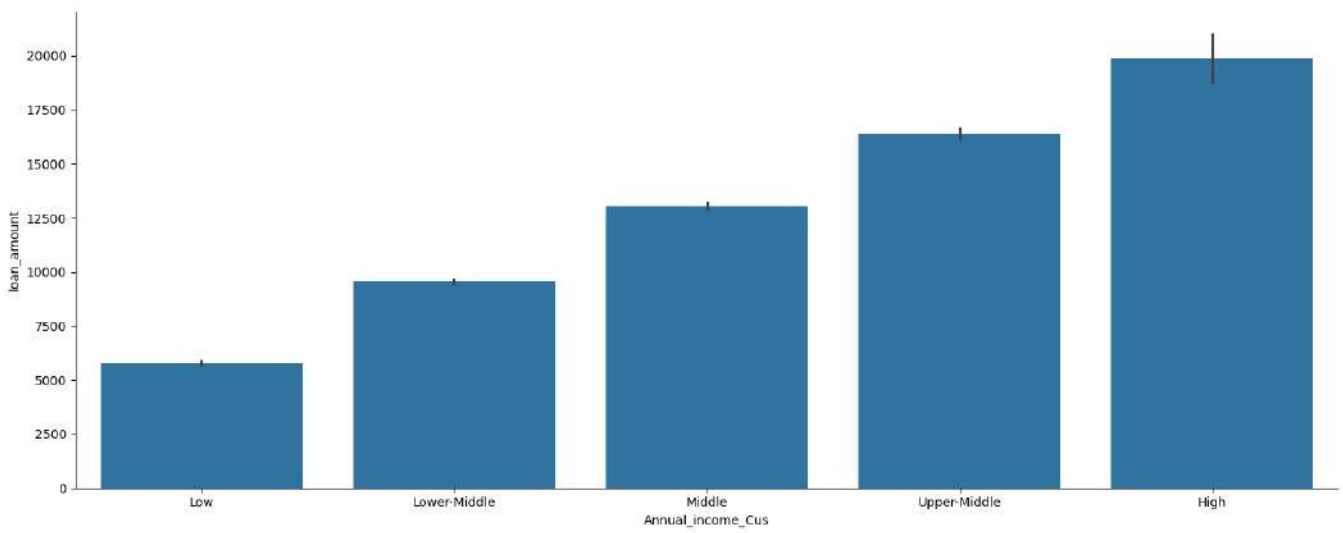


In [111…
```
sns.catplot(y="loan_amount", x="emp_length_Cus",  kind="bar", data=df, height=6, aspect=2.5)
plt.show()
```

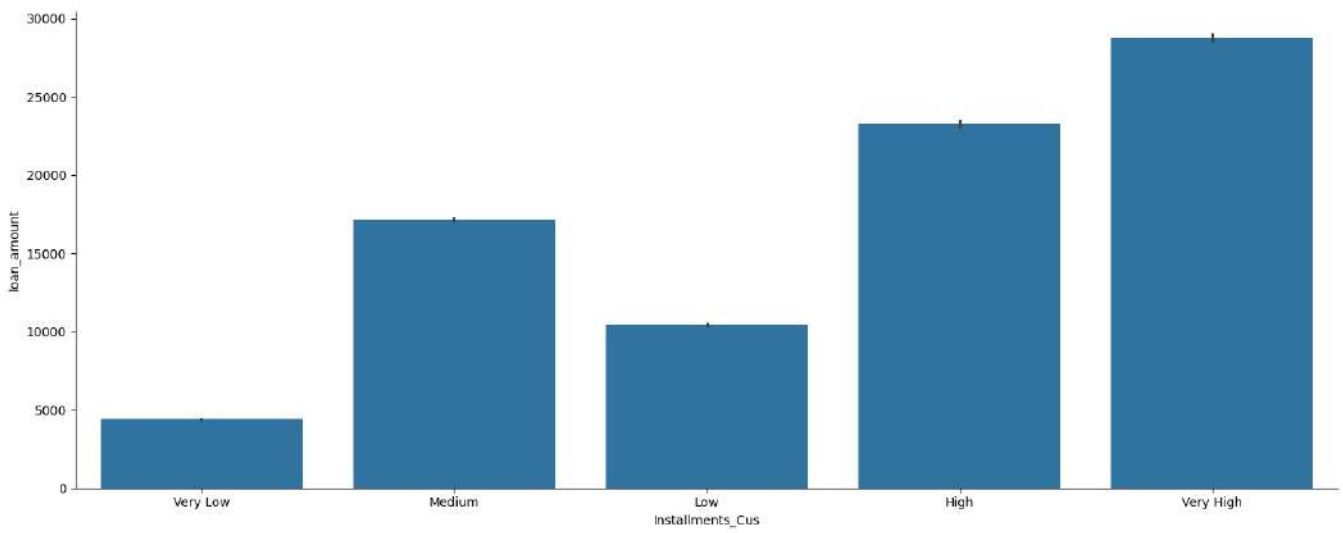


In [112…
```
sns.catplot(y="loan_amount", x="Annual_income_Cus",  kind="bar", data=df, height=6, aspect=2.5)
```

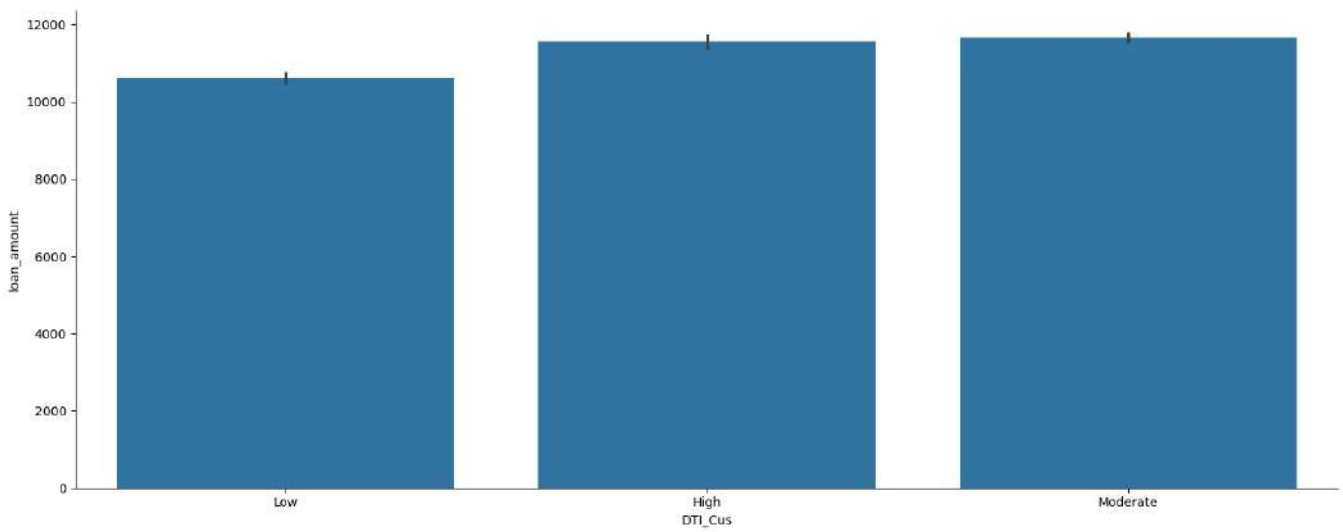
```
plt.show()
```



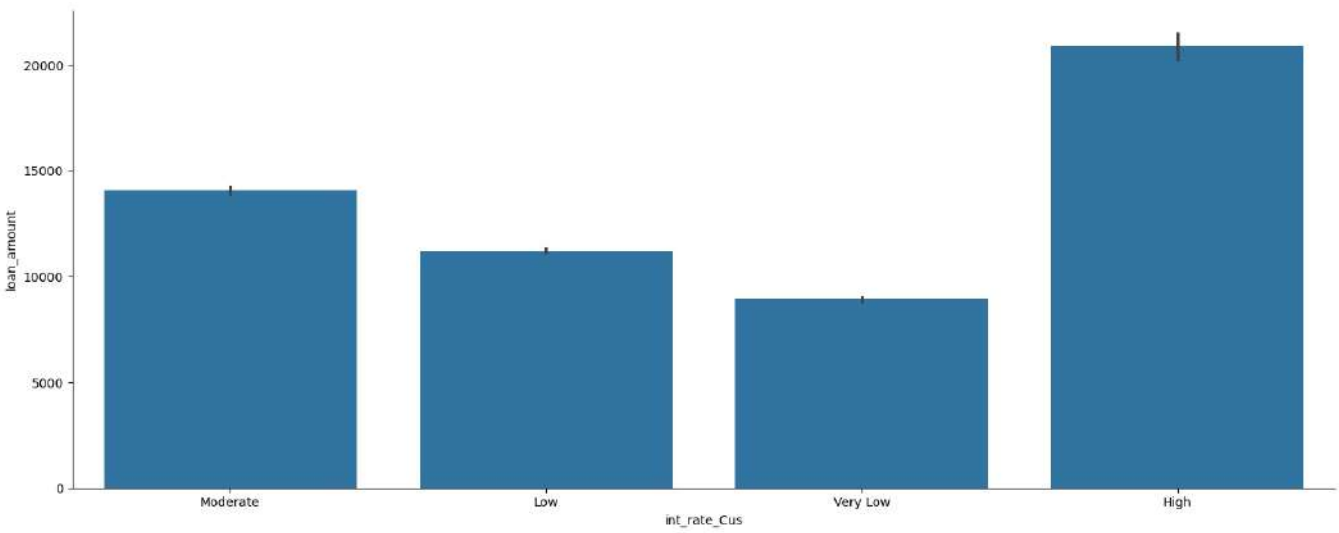
In [113…
```
sns.catplot(y="loan_amount", x="Installments_Cus",  kind="bar", data=df, height=6, aspect=2.5)
plt.show()
```



In [202…
```
sns.catplot(y="loan_amount", x="DTI_Cus",  kind="bar", data=df, height=6, aspect=2.5)
plt.show()
```



In [204…
```
sns.catplot(y="loan_amount", x="int_rate_Cus",  kind="bar", data=df, height=6, aspect=2.5)
plt.show()
```



In [206…

```
sns.catplot(y="loan_amount", x="total_payment_Cus",  kind="bar", data=df, height=6, aspect=2.5)
plt.show()
```