BANK LOAN ANALYSIS REPORT

Upgrading libraries

In [1]:
```
!pip install --upgrade numpy pandas matplotlib seaborn plotly
```

```
Requirement already satisfied: numpy in c:\users\priya\anaconda3\lib\site-packages (1.26.4)
Collecting numpy
  Using cached numpy-2.3.3-cp312-cp312-win_amd64.whl.metadata (60 kB)
Requirement already satisfied: pandas in c:\users\priya\anaconda3\lib\site-packages (2.3.2)
Requirement already satisfied: matplotlib in c:\users\priya\anaconda3\lib\site-packages (3.10.6)
Requirement already satisfied: seaborn in c:\users\priya\anaconda3\lib\site-packages (0.13.2)
Requirement already satisfied: plotly in c:\users\priya\anaconda3\lib\site-packages (6.3.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\priya\anaconda3\lib\site-packages (from pandas) (2.9.0.p
ost0)
Requirement already satisfied: pytz>=2020.1 in c:\users\priya\anaconda3\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\priya\anaconda3\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\priya\anaconda3\lib\site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\priya\anaconda3\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\priya\anaconda3\lib\site-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\priya\anaconda3\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\priya\anaconda3\lib\site-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=8 in c:\users\priya\anaconda3\lib\site-packages (from matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\priya\anaconda3\lib\site-packages (from matplotlib) (3.1.2)
Requirement already satisfied: narwhals>=1.15.1 in c:\users\priya\anaconda3\lib\site-packages (from plotly) (2.5.0)
Requirement already satisfied: six>=1.5 in c:\users\priya\anaconda3\lib\site-packages (from python-dateutil>=2.8.2->panda
s) (1.16.0)
```

Importing Libraries

In [5]:
```
import numpy as np, pandas as pd, matplotlib, seaborn as sns, plotly

print("NumPy:", np.__version__)
print("Pandas:", pd.__version__)
print("Matplotlib:", matplotlib.__version__)
print("Seaborn:", sns.__version__)
print("Plotly:", plotly.__version__)
```

```
NumPy: 1.26.4
Pandas: 2.3.2
Matplotlib: 3.10.6
Seaborn: 0.13.2
Plotly: 6.3.0
```

In [7]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import plotly.express as px
```

In [9]:
```
df = pd.read_excel("C:/Users/Priya/Downloads/manish doc/data analytics/bank loan project/data/financial_loan.xlsx")
```

In [11]:
```
df.head()
```

Out[11]:

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

5 rows × 24 columns

In [13]:
```
df.tail()
```

Out[13]:

	id	address_state	application_type	emp_length	emp_title	grade	home_ownership	issue_date	last_credit_pull_date	las
38571	803452	NJ	INDIVIDUAL	< 1 year	Joseph M Sanzari Company	C	MORTGAGE	2021-07-11	2021-05-16	
38572	970377	NY	INDIVIDUAL	8 years	Swat Fame	C	RENT	2021-10-11	2021-04-16	
38573	875376	CA	INDIVIDUAL	5 years	Anaheim Regional Medical Center	D	RENT	2021-09-11	2021-05-16	
38574	972997	NY	INDIVIDUAL	5 years	Brooklyn Radiology	D	RENT	2021-10-11	2021-05-16	
38575	682952	NY	INDIVIDUAL	4 years	Allen Edmonds	F	RENT	2021-07-11	2021-05-16	

5 rows × 24 columns

Metadata of data

In [21]:
```
print( "No. of rows and coloumns:", df.shape)
```

```
No. of rows and coloumns: (38576, 24)
```

In [23]:
```
df.dtypes
```

Out[23]:
```
id                          int64
address_state              object
application_type           object
emp_length                 object
emp_title                  object
grade                      object
home_ownership             object
issue_date         datetime64[ns]
last_credit_pull_date      datetime64[ns]
last_payment_date          datetime64[ns]
loan_status                object
next_payment_date          datetime64[ns]
member_id                   int64
purpose                    object
sub_grade                  object
term                       object
verification_status        object
annual_income             float64
dti                       float64
installment               float64
int_rate                  float64
loan_amount                 int64
total_acc                   int64
total_payment               int64
dtype: object
```

Total Loan Applications

In [28]:
```
Total_Loan_Application = df['id'].count()
print( "Total Loan Application:", Total_Loan_Application)
```

```
Total Loan Application: 38576
```

MTD Total Loan Application

In [36]:
```
latest_issue_date = df['issue_date'].max()
latest_year = latest_issue_date.year
latest_month = latest_issue_date.month

mtd_data = df[(df['issue_date'].dt.year == latest_year) & (df['issue_date'].dt.month == latest_month)]

mtd_loan_applications = mtd_data['id'].count()

print(f"MTD Loan Applications(for {latest_issue_date.strftime('%B %Y')}):{mtd_loan_applications}")
```

```
MTD Loan Applications(for December 2021):4314
```

Total Funded Amount

In [46]:
```
Total_fund_amount = df["loan_amount"].sum()
total_funded_amount_millions = Total_fund_amount/ 1000000
print("Total funded amount: ${:.2f}M".format(total_funded_amount_millions))
```

```
Total funded amount: $435.76M
```

MTD Total Funded Amount

In [55]:
```
latest_issue_date = df['issue_date'].max()
latest_year = latest_issue_date.year
latest_month = latest_issue_date.month

mtd_data = df[(df['issue_date'].dt.year == latest_year) & (df['issue_date'].dt.month == latest_month)]

mtd_total_funded_amount = mtd_data['loan_amount'].sum()
mtd_total_funded_amount_millions = mtd_total_funded_amount/1000000
print(" MTD Total Funded Amount: ${:.2f}M".format(mtd_total_funded_amount_millions))
```

```
MTD Total Funded Amount: $53.98M
```

Total Amount Received

In [62]:
```
Total_amount_received = df["total_payment"].sum()
total_amount_recived_millions = Total_amount_received/ 1000000
print("Total amount received: ${:.2f}M".format(total_amount_recived_millions))
```

```
Total amount received: $473.07M
```

MTD Total Amount Received

In [64]:
```
latest_issue_date = df['issue_date'].max()
latest_year = latest_issue_date.year
latest_month = latest_issue_date.month

mtd_data = df[(df['issue_date'].dt.year == latest_year) & (df['issue_date'].dt.month == latest_month)]

mtd_amount_received = mtd_data['total_payment'].sum()
mtd_total_amount_received_millions = mtd_amount_received/1000000
print(" MTD Total Amount Received: ${:.2f}M".format(mtd_total_amount_received_millions))
```

```
MTD Total Amount Received: $58.07M
```

Average Interest rate

In [72]:
```
Average_Interest_rate = df['int_rate'].mean()*100
print(" Average interest rate: {:.2f}%" .format(Average_Interest_rate))
```

```
Average interest rate: 12.05%
```

Average DTI ratio

In [76]:
```
Average_DTI_ratio = df['dti'].mean()*100
print(" Average DTI ratio: {:.2f}%" .format(Average_DTI_ratio))
```

```
Average DTI ratio: 13.33%
```

Good Loan Metrics

In [83]:
```
good_loans = df[df['loan_status'].isin(["Fully Paid", "Current"])]

good_loan_applications = good_loans['id'].count()
good_loan_funded_amount = good_loans['loan_amount'].sum()
good_loan_received = good_loans['total_payment'].sum()

good_loan_funded_amount_millions = good_loan_funded_amount/1000000
good_loan_received_millions = good_loan_received/1000000

good_loan_percentage = (good_loan_applications / Total_Loan_Application) * 100

print(" Good Loan Applications:",good_loan_applications)
print(" Good Loan Funded Amount(in millions): ${:.2f}M".format(good_loan_funded_amount_millions))
```

```
print(" Good Loan Amount Received (in millions): ${:.2f}M".format(good_loan_received_millions))
print(" Percentage of Good Loan Applications: {:.2f}%".format(good_loan_percentage))
```

```
Good Loan Applications: 33243
Good Loan Funded Amount(in millions): $370.22M
Good Loan Amount Received (in millions): $435.79M
Percentage of Good Loan Applications: 86.18%
```

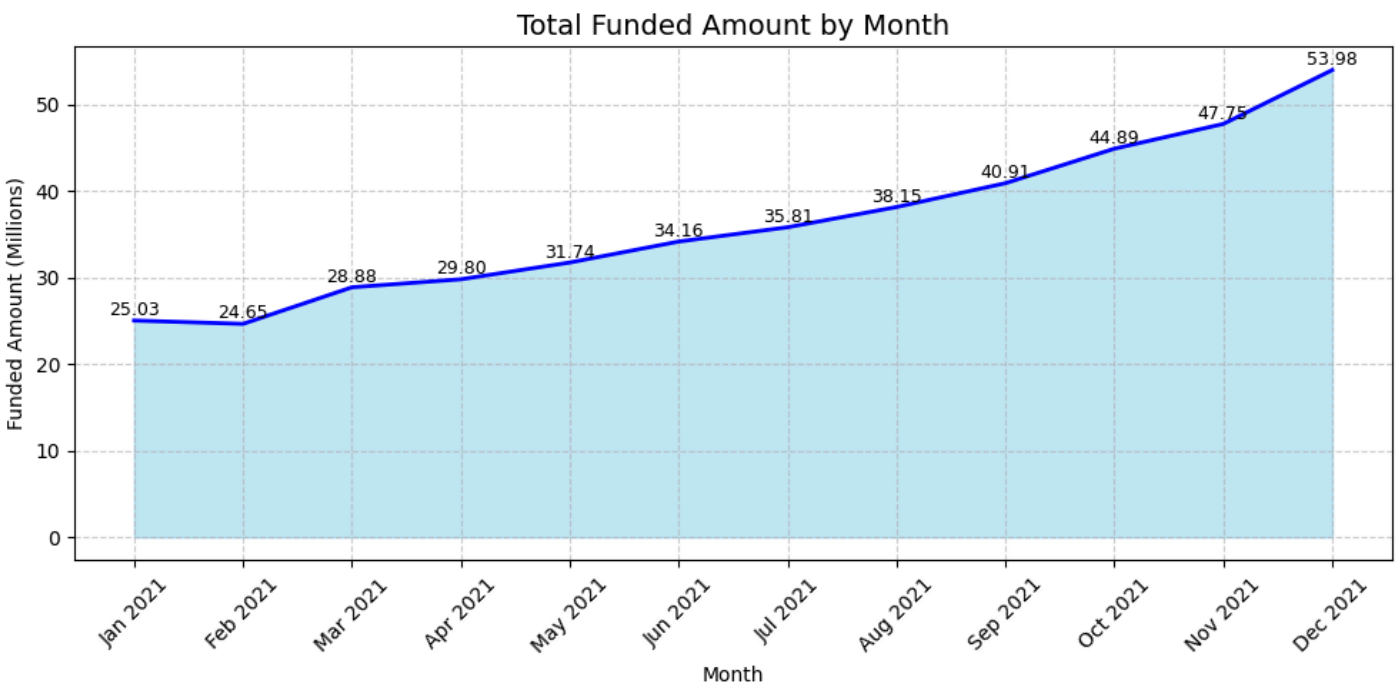
Bad Loan Metrics

In [88]:
```
bad_loans = df[df['loan_status'].isin(["Charged Off"])]

bad_loan_applications = bad_loans['id'].count()
bad_loan_funded_amount = bad_loans['loan_amount'].sum()
bad_loan_received = bad_loans['total_payment'].sum()

bad_loan_funded_amount_millions = bad_loan_funded_amount/1000000
bad_loan_received_millions = bad_loan_received/1000000

bad_loan_percentage = (bad_loan_applications / Total_Loan_Application) * 100

print(" Bad Loan Applications:",bad_loan_applications)
print(" Bad Loan Funded Amount(in millions): ${:.2f}M".format(bad_loan_funded_amount_millions))
print(" Bad Loan Amount Received (in millions): ${:.2f}M".format(bad_loan_received_millions))
print(" Percentage of Bad Loan Applications: {:.2f}%".format(bad_loan_percentage))
```

```
Bad Loan Applications: 5333
Bad Loan Funded Amount(in millions): $65.53M
Bad Loan Amount Received (in millions): $37.28M
Percentage of Bad Loan Applications: 13.82%
```

Monthly Trends by Issue Date for Total funded amount

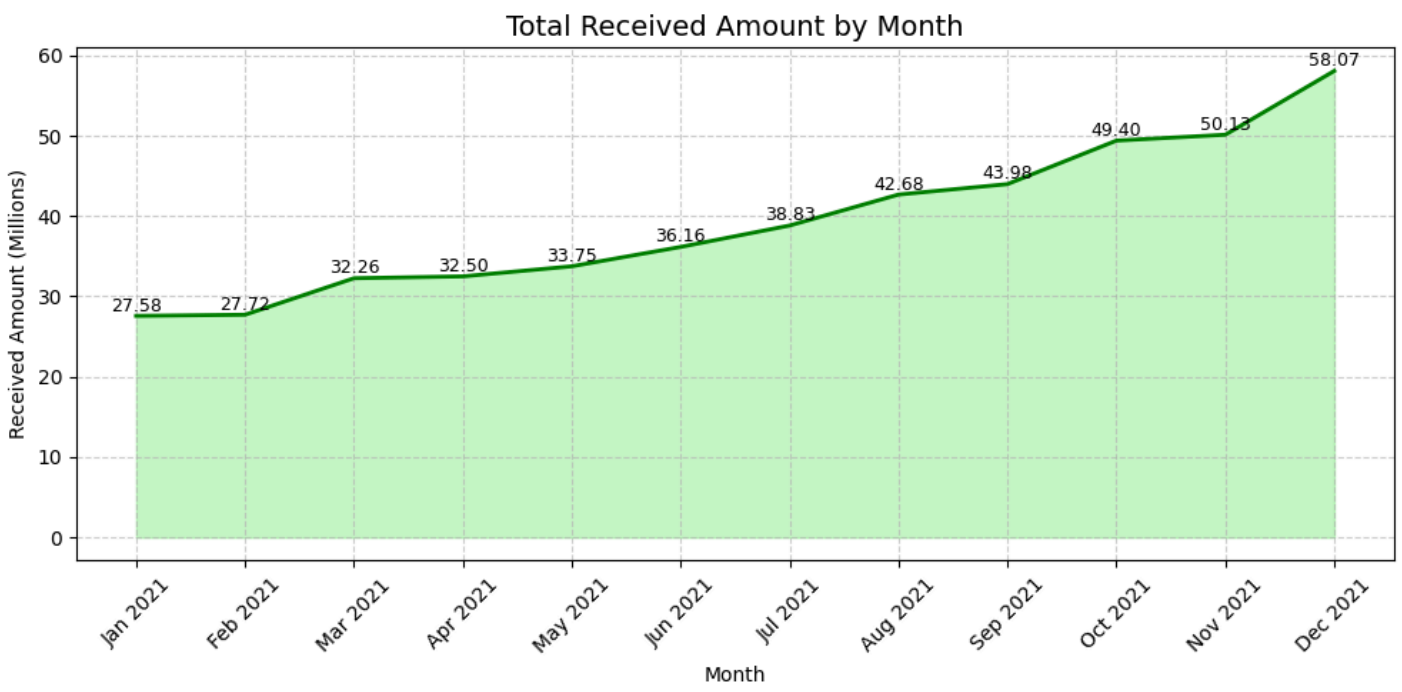
In [95]:
```
monthly_funded = (
    df.sort_values('issue_date')
      .assign(month_name=lambda x: x['issue_date'].dt.strftime('%b %Y'))
      .groupby('month_name', sort=False)['loan_amount']
      .sum()
      .div(1000000)
      .reset_index(name='loan_amount_millions')
)

plt.figure(figsize=(10, 5))

plt.fill_between(
    monthly_funded['month_name'],
    monthly_funded['loan_amount_millions'],
    color='skyblue',
    alpha=0.5
)

plt.plot(
    monthly_funded['month_name'],
    monthly_funded['loan_amount_millions'],
    color='blue',
    linewidth=2
)

for i, row in monthly_funded.iterrows():
    plt.text(
        i,
        row['loan_amount_millions'] + 0.1,
        f"{row['loan_amount_millions']:.2f}",
        ha='center',
        va='bottom',
        fontsize=9,
        rotation=0,
        color='black'
    )

plt.title('Total Funded Amount by Month', fontsize=14)
plt.xlabel('Month')
plt.ylabel('Funded Amount (Millions)')
plt.xticks(ticks=range(len(monthly_funded)), labels=monthly_funded['month_name'], rotation=45)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```



Monthly Trends by Issue Date for Total Amount Received

In [107…

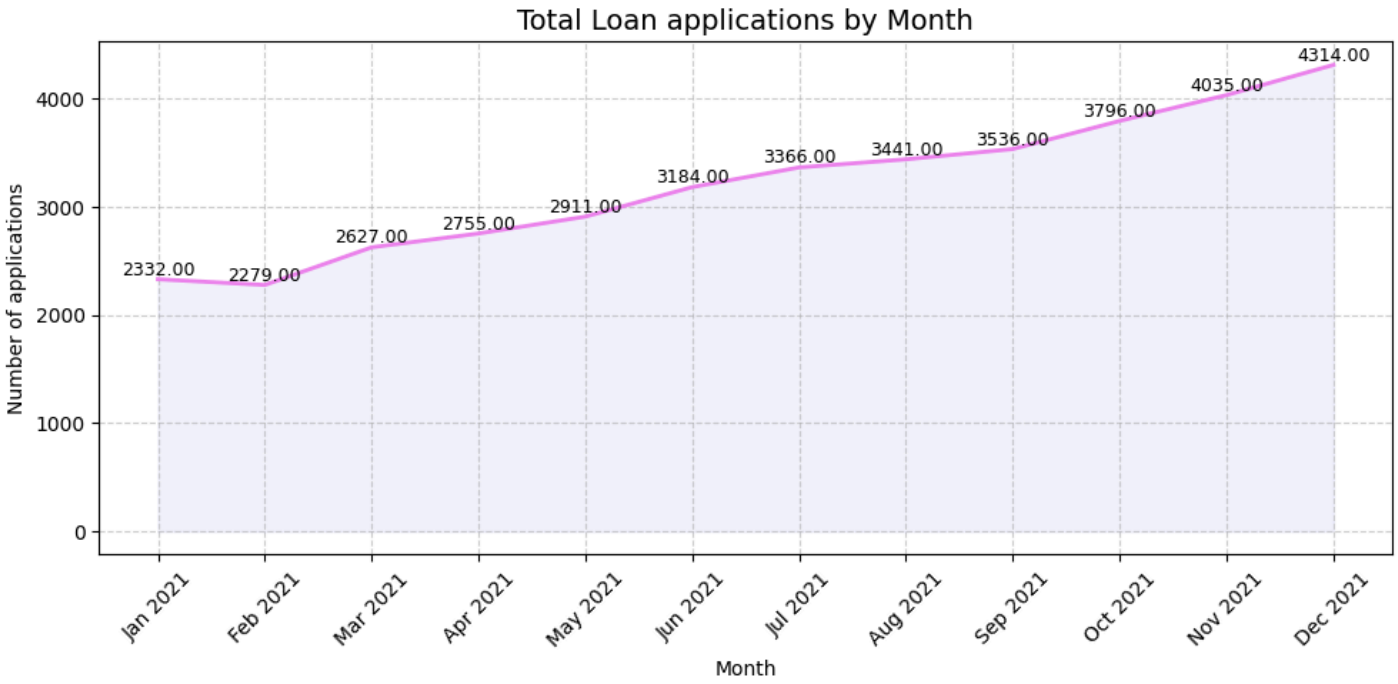
```
monthly_received = (
    df.sort_values('issue_date')
      .assign(month_name=lambda x: x['issue_date'].dt.strftime('%b %Y'))
      .groupby('month_name', sort=False)['total_payment']
      .sum()
      .div(1000000)
      .reset_index(name='received_amount_millions')
)

plt.figure(figsize=(10, 5))

plt.fill_between(
    monthly_received['month_name'],
    monthly_received['received_amount_millions'],
    color='lightgreen',
    alpha=0.5
)

plt.plot(
    monthly_received['month_name'],
    monthly_received['received_amount_millions'],
    color='green',
    linewidth=2
)

for i, row in monthly_received.iterrows():
    plt.text(
        i,
        row['received_amount_millions'] + 0.1,
        f"{row['received_amount_millions']:.2f}",
        ha='center',
        va='bottom',
        fontsize=9,
        rotation=0,
        color='black'
    )

plt.title('Total Received Amount by Month', fontsize=14)
plt.xlabel('Month')
plt.ylabel('Received Amount (Millions)')
plt.xticks(ticks=range(len(monthly_received)), labels=monthly_received['month_name'], rotation=45)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```



Monthly Trends by Issue Date for Total Loan Applications

In [110…

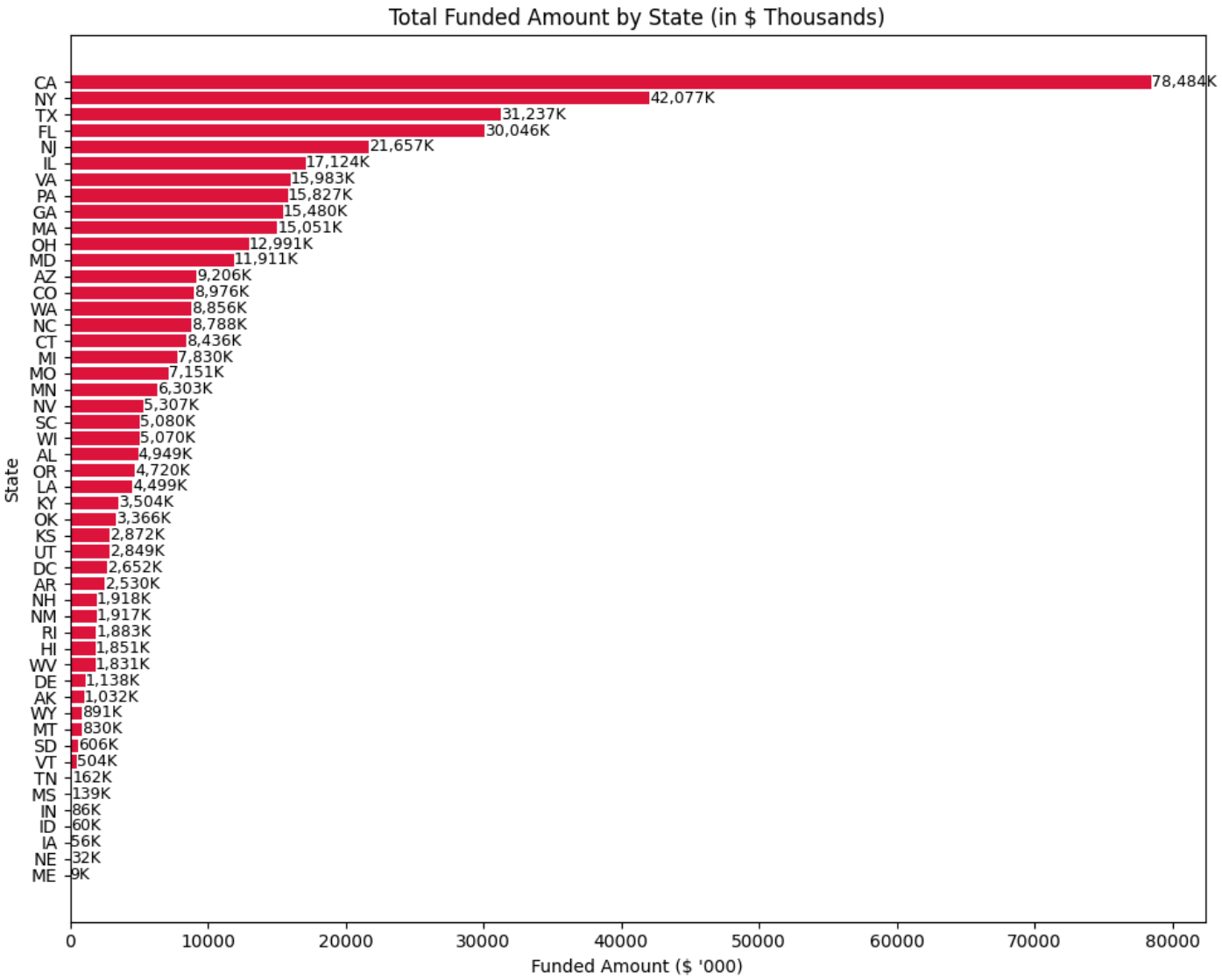
```
monthly_applications = (
    df.sort_values('issue_date')
      .assign(month_name=lambda x: x['issue_date'].dt.strftime('%b %Y'))
      .groupby('month_name', sort=False)['id']
      .count()
      .reset_index(name='loan_application_count')
)

plt.figure(figsize=(10, 5))

plt.fill_between(
    monthly_applications['month_name'],
    monthly_applications['loan_application_count'],
    color='lavender',
    alpha=0.5
)

plt.plot(
    monthly_applications['month_name'],
    monthly_applications['loan_application_count'],
    color='violet',
    linewidth=2
)

for i, row in monthly_applications.iterrows():
    plt.text(
        i,
        row['loan_application_count'] + 0.1,
        f"{row['loan_application_count']:.2f}",
        ha='center',
        va='bottom',
        fontsize=9,
        rotation=0,
        color='black'
    )

plt.title('Total Loan applications by Month', fontsize=14)
plt.xlabel('Month')
plt.ylabel('Number of applications')
plt.xticks(ticks=range(len(monthly_applications)), labels=monthly_applications['month_name'], rotation=45)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```



Regional Analysis by State of total funded amount

In [148…

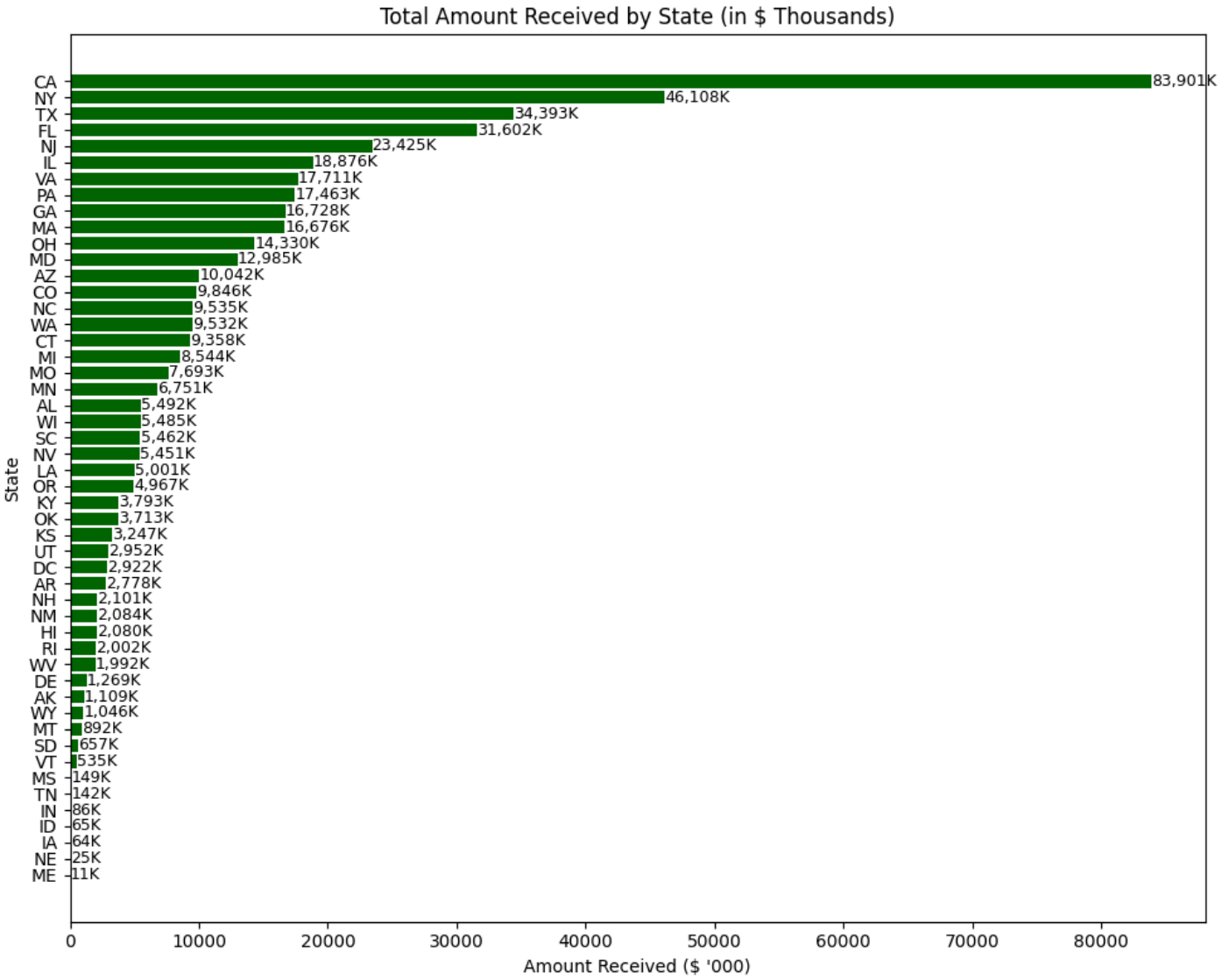
```
state_funding = df.groupby('address_state')['loan_amount'].sum().sort_values(ascending=True)
state_funding_thousand = state_funding/1000

plt.figure(figsize=(10,8))
bars = plt.barh(state_funding_thousand.index, state_funding_thousand.values,color='crimson')

for bar in bars:
    width = bar.get_width()
    plt.text(width + 10, bar.get_y() + bar.get_height() / 2,
             f'{width:,.0f}K', va='center' , fontsize = 9)

plt.title('Total Funded Amount by State (in $ Thousands)')
plt.xlabel('Funded Amount ($ \'000)')
plt.ylabel('State')
plt.tight_layout()
plt.show()
```



Regional Analysis by State of total amount received

In [150…

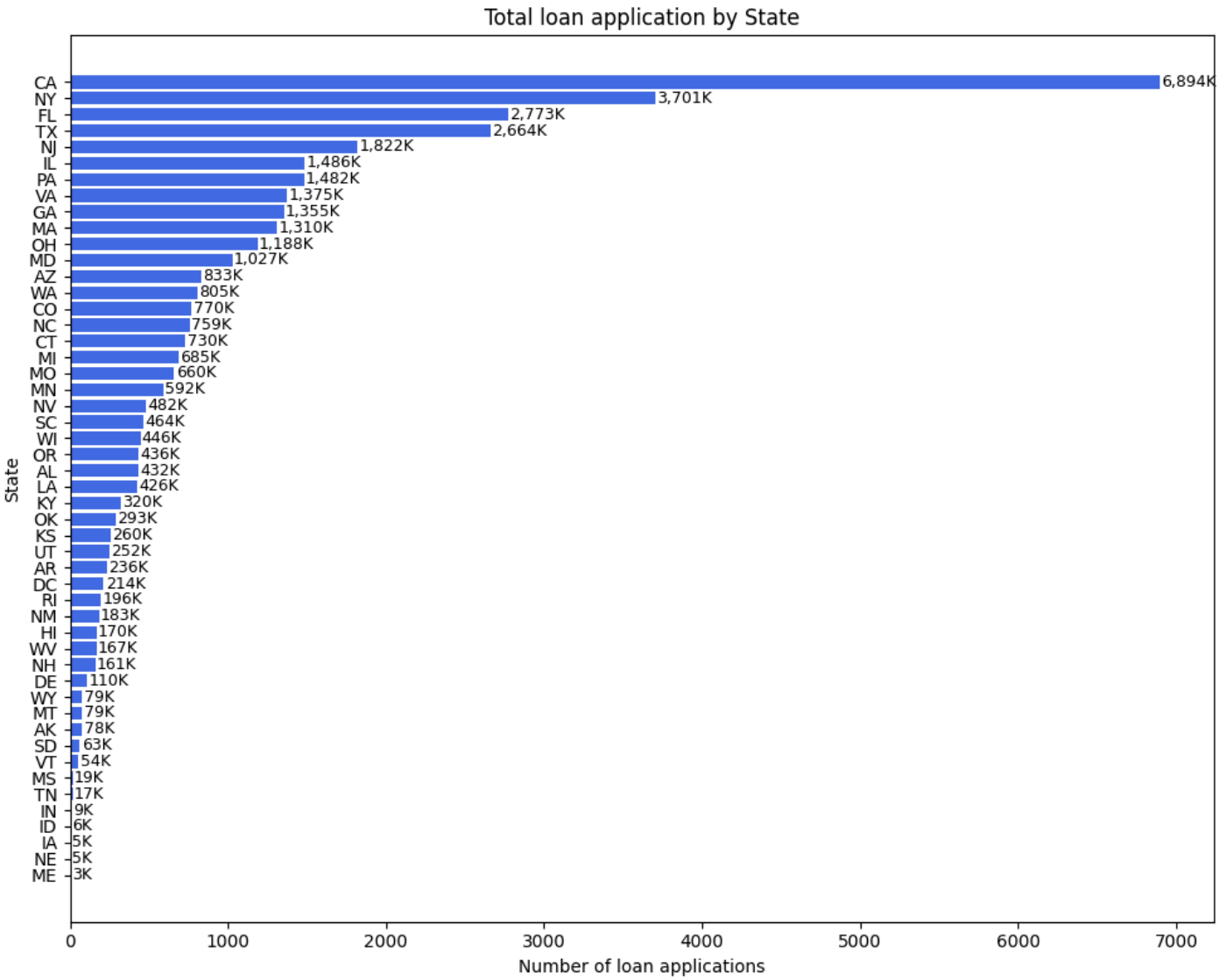
```
state_amount_received = df.groupby('address_state')['total_payment'].sum().sort_values(ascending=True)
state_amount_received_thousand = state_amount_received/1000

plt.figure(figsize=(10,8))
bars = plt.barh(state_amount_received_thousand.index, state_amount_received_thousand.values,color='darkgreen')

for bar in bars:
    width = bar.get_width()
    plt.text(width + 10, bar.get_y() + bar.get_height() / 2,
             f'{width:,.0f}K', va='center' , fontsize = 9)

plt.title('Total Amount Received by State (in $ Thousands)')
plt.xlabel('Amount Received ($ \'000)')
plt.ylabel('State')
plt.tight_layout()
plt.show()
```



Regional Analysis by State of total loan Applications

In [141…

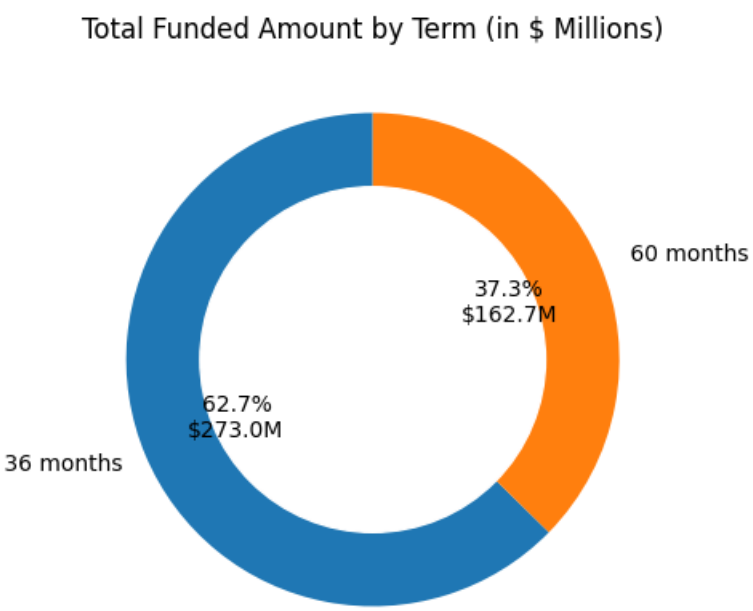
```
state_loan_app = df.groupby('address_state')['id'].count().sort_values(ascending=True)


plt.figure(figsize=(10,8))
bars = plt.barh(state_loan_app.index, state_loan_app.values,color='royalblue')

for bar in bars:
    width = bar.get_width()
    plt.text(width + 10, bar.get_y() + bar.get_height() / 2,
             f'{width:,.0f}K', va='center' , fontsize = 9)

plt.title('Total loan application by State')
plt.xlabel('Number of loan applications')
plt.ylabel('State')
plt.tight_layout()
plt.show()
```



Loan Term Analysis by total Fund Amount

In [146…

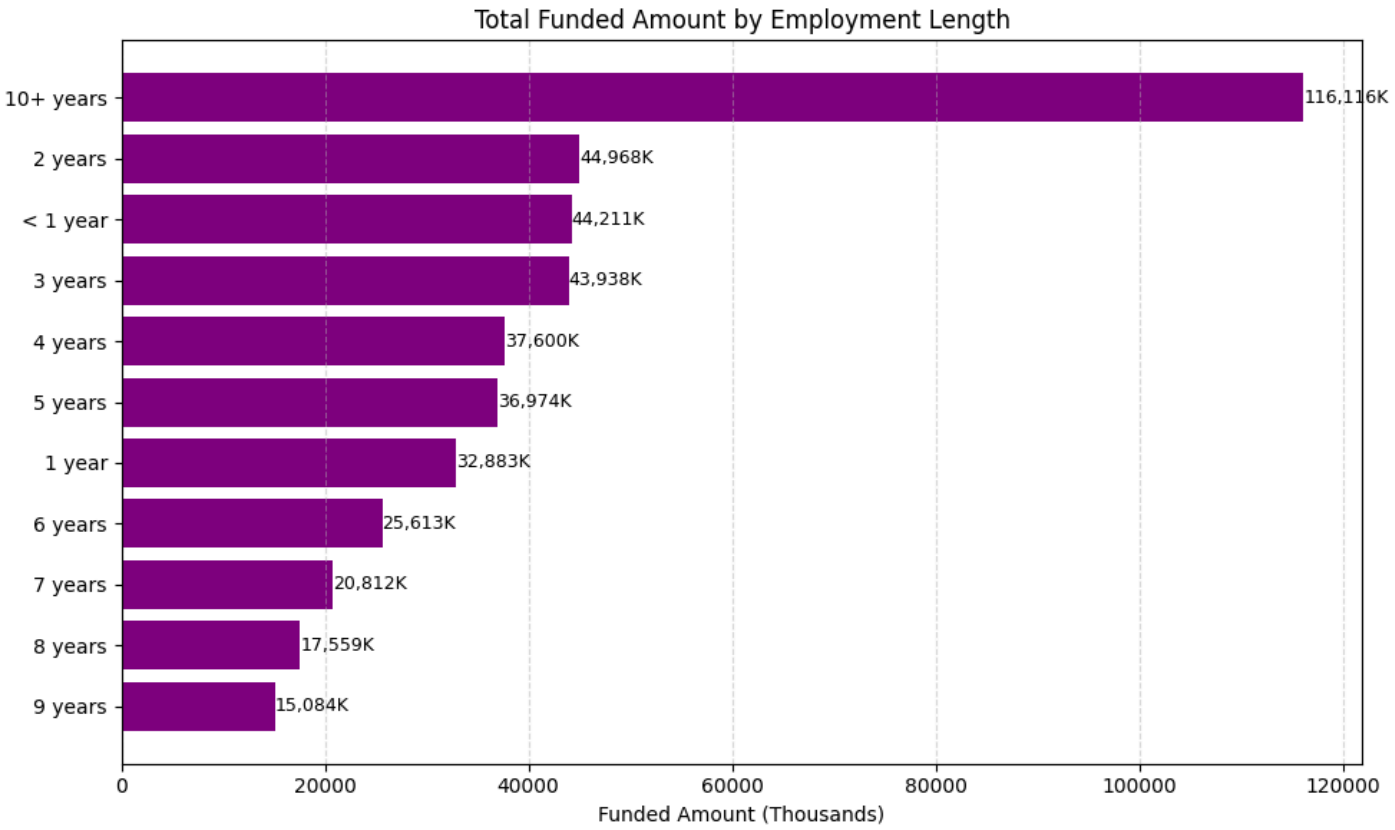
```
term_funding_millions = df.groupby('term')['loan_amount'].sum() / 1000000

plt.figure(figsize=(5, 5))

plt.pie(
    term_funding_millions,
    labels=term_funding_millions.index,
    autopct=lambda p: f"{p:.1f}%\n${p*sum(term_funding_millions)/100:.1f}M",
    startangle=90,
    wedgeprops={'width': 0.4}
)

# Add a white circle in the center (to make it a donut chart)
plt.gca().add_artist(plt.Circle((0, 0), 0.70, color='white'))

plt.title("Total Funded Amount by Term (in $ Millions)")
plt.show()
```



Employee length by total Funded Amount

In [155…

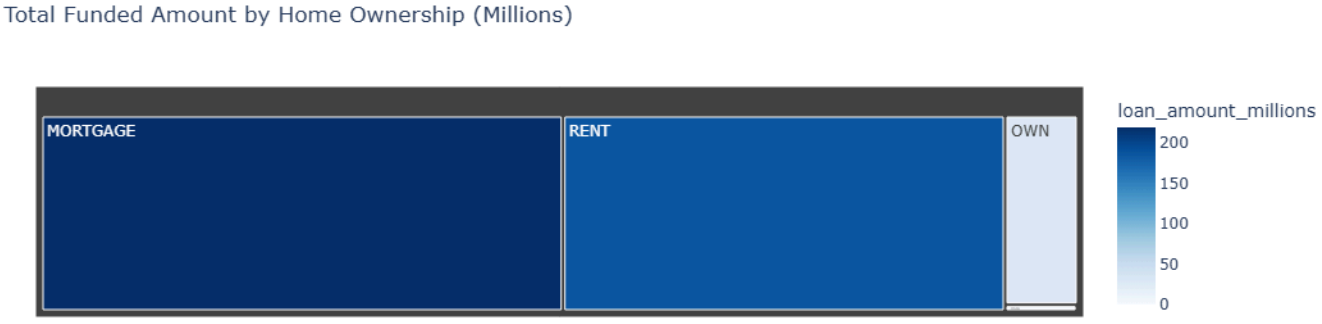
```python
# Group by employment length and calculate total funded amount (in thousands)
emp_funding_thousands = df.groupby('emp_length')['loan_amount'].sum().sort_values() / 1000

plt.figure(figsize=(10, 6))

# Create horizontal bar chart
bars = plt.barh(emp_funding_thousands.index, emp_funding_thousands, color='purple')

# Add data labels to each bar
for bar in bars:
    width = bar.get_width()
    plt.text(
        width + 5,
        bar.get_y() + bar.get_height() / 2,
        f"{width:,.0f}K",
        va='center',
        fontsize=9
    )

plt.xlabel("Funded Amount (Thousands)")
plt.title("Total Funded Amount by Employment Length")
plt.grid(axis='x', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```



Home ownership by total Funded Amount

In [159…

```
import plotly.express as px

# Group by home ownership and calculate loan amount in millions
home_funding = df.groupby('home_ownership')['loan_amount'].sum().reset_index()
home_funding['loan_amount_millions'] = home_funding['loan_amount'] / 1000000

# Create treemap
fig = px.treemap(
    home_funding,
    path=['home_ownership'],
    values='loan_amount_millions',
    color='loan_amount_millions',
    color_continuous_scale='Blues',
    title='Total Funded Amount by Home Ownership (Millions)'
)

fig.show()
```



In [ ]: