FINANCIAL LOAN ANALYSIS REPORT

Import Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import plotly.express as px
```

```
from google.colab import files
uploaded = files.upload()
print(uploaded)
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving financial_loan.xlsx to financial_loan.xlsx
{'financial_loan.xlsx': b'PK\x03\x04\x14\x00\x06\x00\x08\x00\x00\x00!\x00\xdd+\x8bXl\x01\x00\x00\x10\x05\x00\x00\x13\x00\x08
```

```
df = pd.read_excel('financial_loan.xlsx')
```

```
df.head(10)
```

	id	address_state	application_type	emp_length	emp_title	grade	home_ownership	issue_date	last_credit_pull_date
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	1062608	CA	INDIVIDUAL	3 years	Studio 94 Corp	C	RENT	2021-07-17	2021-03-16
6	1067441	TX	INDIVIDUAL	10+ years	American Airlines	C	MORTGAGE	2021-11-19	2021-06-14
7	1066424	PA	INDIVIDUAL	10+ years	SCI Mahanoy	A	OWN	2021-06-11	2021-07-14
8	1065254	FL	INDIVIDUAL	10+ years	Tech Data Corp	A	MORTGAGE	2021-09-02	2021-06-15
9	1064589	MI	INDIVIDUAL	10+ years	teltow contracting	B	MORTGAGE	2021-02-09	2021-03-16

10 rows × 24 columns

Metadata of Data

```
print("No. of Rows:", df.shape[0])
```

```
No. of Rows: 38576
```

```
print("No. of Columns:", df.shape[1])
```

```
No. of Columns: 24
```

```
df.info
```

```
pandas.core.frame.DataFrame.info
def info(verbose: bool | None=None, buf: WriteBuffer[str] | None=None, max_cols: int | None=None,
memory_usage: bool | str | None=None, show_counts: bool | None=None) -> None
```

```
Print a concise summary of a DataFrame.

This method prints information about a DataFrame including
the index dtype and columns, non-null values and memory usage.

Parameters
```

Data Type

```
df.dtypes
```

	0
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

```
df.describe()
```

	id	issue_date	last_credit_pull_date	last_payment_date	next_payment_date	member_id	annual_inc
count	3.857600e+04	38576	38576	38576	38576	3.857600e+04	3.857600e
mean	6.810371e+05	2021-07-16 02:31:35.562007040	2021-06-08 13:36:34.193280512	2021-06-26 09:52:08.909166080	2021-07-26 20:42:20.605557760	8.476515e+05	6.964454e
min	5.473400e+04	2021-01-01 00:00:00	2021-01-08 00:00:00	2021-01-08 00:00:00	2021-02-08 00:00:00	7.069900e+04	4.000000e
25%	5.135170e+05	2021-04-11 00:00:00	2021-04-15 00:00:00	2021-03-16 00:00:00	2021-04-16 00:00:00	6.629788e+05	4.150000e
50%	6.627280e+05	2021-07-11 00:00:00	2021-05-16 00:00:00	2021-06-14 00:00:00	2021-07-14 00:00:00	8.473565e+05	6.000000e
75%	8.365060e+05	2021-10-11 00:00:00	2021-08-13 00:00:00	2021-09-15 00:00:00	2021-10-15 00:00:00	1.045652e+06	8.320050e
max	1.077501e+06	2021-12-12 00:00:00	2022-01-20 00:00:00	2021-12-15 00:00:00	2022-01-15 00:00:00	1.314167e+06	6.000000e
std	2.113246e+05	NaN	NaN	NaN	NaN	2.668105e+05	6.429368e

Total Loan Applications

```
total_loan_application = df['id'].count()
print("Total Loan Applications:", total_loan_application)
```

```
Total Loan Applications: 38576
```

MTD Total Loan Apllications

```
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_loan_applications = mtd_data['id'].count()

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

```
MTD Total Loan Applications for December 2021: 4314
```

Total Funded Amount

```
total_funded_amount = df['loan_amount'].sum()
total_funded_millions = total_funded_amount/1000000
print("Total Funded Amount: ${:.2f}M".format(total_funded_millions))
```

```
Total Funded Amount: $435.76M
```

MTD - Total Funded Amount

```
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_millions = mtd_total_funded_amount/1000000


print(f"MTD Total Funded Amount for {latest_issue_date.strftime('%B %Y')}: ${mtd_total_funded_millions:.2f}M")
```

```
MTD Total Funded Amount for December 2021: $53.98M
```

Total Amount Recevied

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

```
Total Amount Received: $473.07M
```

MTD - Total Amount Recevied

```
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_amount_received = mtd_data['total_payment'].sum()
mtd_total_amount_received = mtd_total_amount_received/100000

print(f"MTD Total Amount Received for {latest_issue_date.strftime('%B %Y')}: ${mtd_total_amount_received:.2f}M")
```

```
MTD Total Amount Received for December 2021: $580.74M
```

Average Interest Rate

```
average_interest_rate = df['int_rate'].mean()*100
print("Average Interest Rate: {:.2f}%".format(average_interest_rate))
```

```
Average Interest Rate: 12.05%
```

Average Debt-to-Income Ratio (DTI)

```
average_dti = df['dti'].mean()*100
print("Average Debt-to-Income Ratio (DTI): {:.2f}%".format(average_dti))
```

```
Average Debt-to-Income Ratio (DTI): 13.33%
```

Good Loan Metrix

```
good_loans = df[df['loan_status'].isin(['Fully Paid', 'Current'])] #Good Loans

total_loan_applications = df['id'].count()

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

good_loan_funded_amount_millions = good_loan_funded_amount/1000000
good_loan_amount_received_millions = good_loan_amount_received/1000000

good_loans_percentage = (good_loan_applications / total_loan_applications) * 100

print("Good Loan Application:", good_loan_applications)
print(f"Good Loan Funded Amount (in Millions): ${good_loan_funded_amount_millions:.2f}M")
print(f"Good Loan Total Amount Received (in Millions): ${good_loan_amount_received_millions:.2f}M")
print("Percentage of Good Loans: {:.2f}%".format(good_loans_percentage))
```

```
Good Loan Application: 33243
Good Loan Funded Amount (in Millions): $370.22M
Good Loan Total Amount Received (in Millions): $435.79M
Percentage of Good Loans: 86.18%
```

Bad Loan Metrix

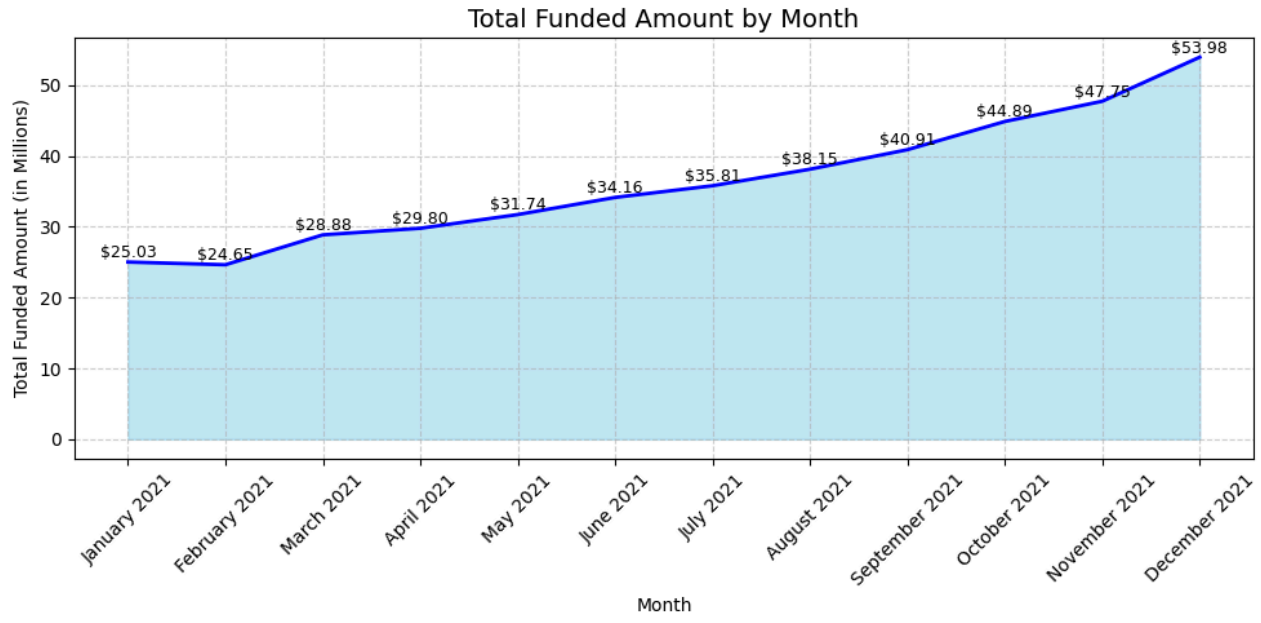
```
bad_loans = df[df['loan_status'].isin(['Charged Off'])]

total_loan_applications = df['id'].count()

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

bad_loan_funded_amount_millions = bad_loan_funded_amount/1000000
```

```
bad_loan_amount_received_millions = bad_loan_amount_received/1000000
bad_loan_percentage = (bad_loan_applications / total_loan_applications) * 100

print("Bad Loan Application:", bad_loan_applications)
print(f"Bad Loan Funded Amount (in Millions): ${bad_loan_funded_amount_millions:.2f}M")
print(f"Bad Loan Total Amount Received (in Millions): ${bad_loan_amount_received_millions:.2f}M")
print("Percentage of Bad Loans: {:.2f}%".format(bad_loan_percentage))
```

```
Bad Loan Application: 5333
Bad Loan Funded Amount (in Millions): $65.53M
Bad Loan Total Amount Received (in Millions): $37.28M
Percentage of Bad Loans: 13.82%
```

Monthly Trends by Issue date for Total Funded Amount

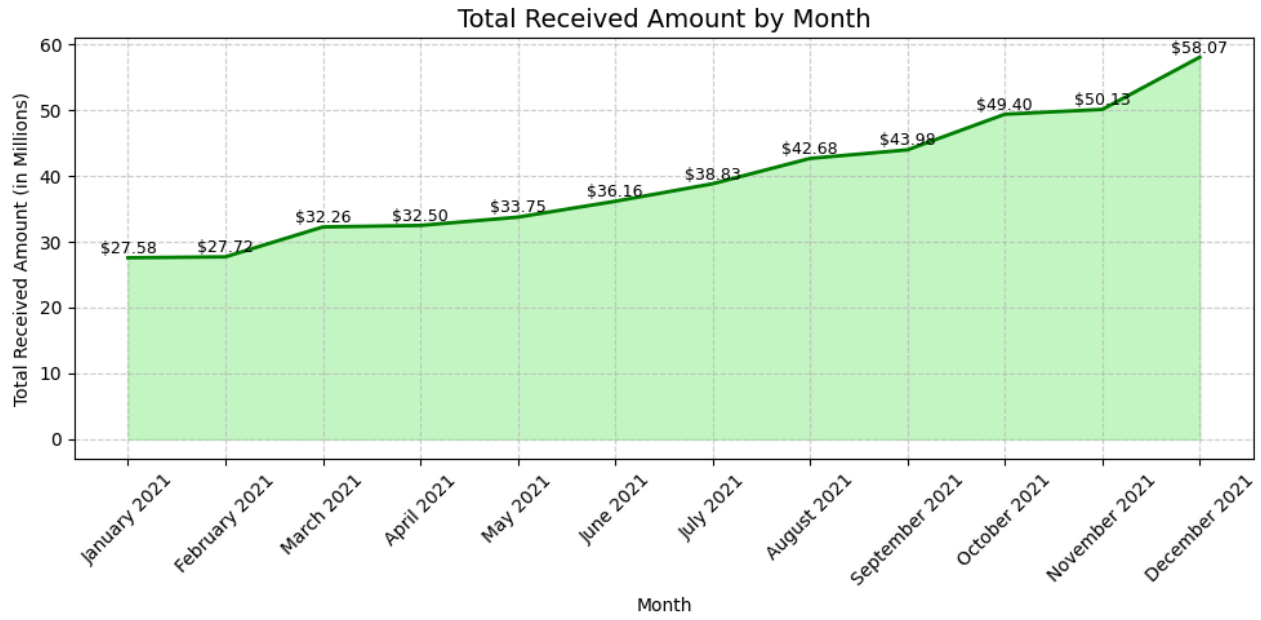
```
from numpy._core.defchararray import title
from os import name
from numpy._core.fromnumeric import sort
monthly_funded = (
    df.sort_values('issue_date')
    .assign(month_name=lambda x: x['issue_date'].dt.strftime('%B %Y'))
    .groupby('month_name', sort=False)['loan_amount']
    .sum()
    .div(1000000)
    .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'], linestyle='-', 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('Total Funded Amount (in Millions)')
plt.xticks(ticks=range(len(monthly_funded['month_name'])), labels=monthly_funded['month_name'], rotation=45)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```



Monthly Trend by Issue date for Total Amount Received

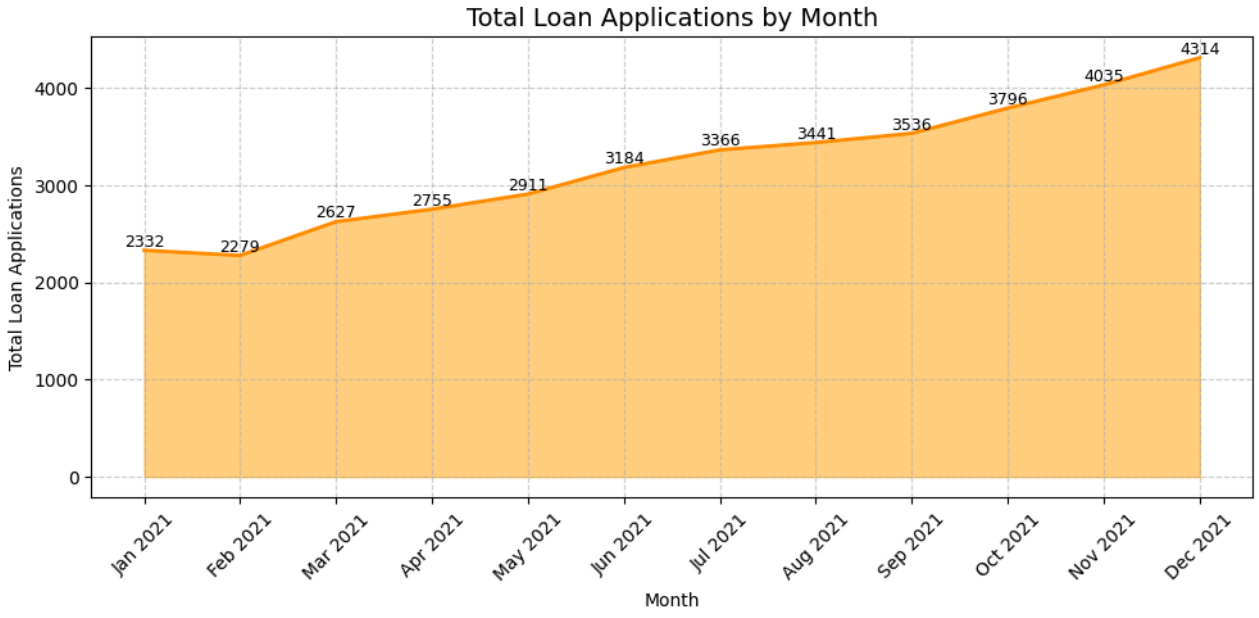
```
monthly_received = (
    df.sort_values('issue_date')
    .assign(month_name=lambda x: x['issue_date'].dt.strftime('%B %Y'))
    .groupby('month_name', sort=False)['total_payment']
    .sum()
```

```
    .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.
plt.plot(monthly_received['month_name'], monthly_received['received_amount_millions'], linestyle='-', color='green', linewi

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('Total Received Amount (in Millions)')
plt.xticks(ticks=range(len(monthly_received['month_name'])), labels=monthly_received['month_name'], rotation=45)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```



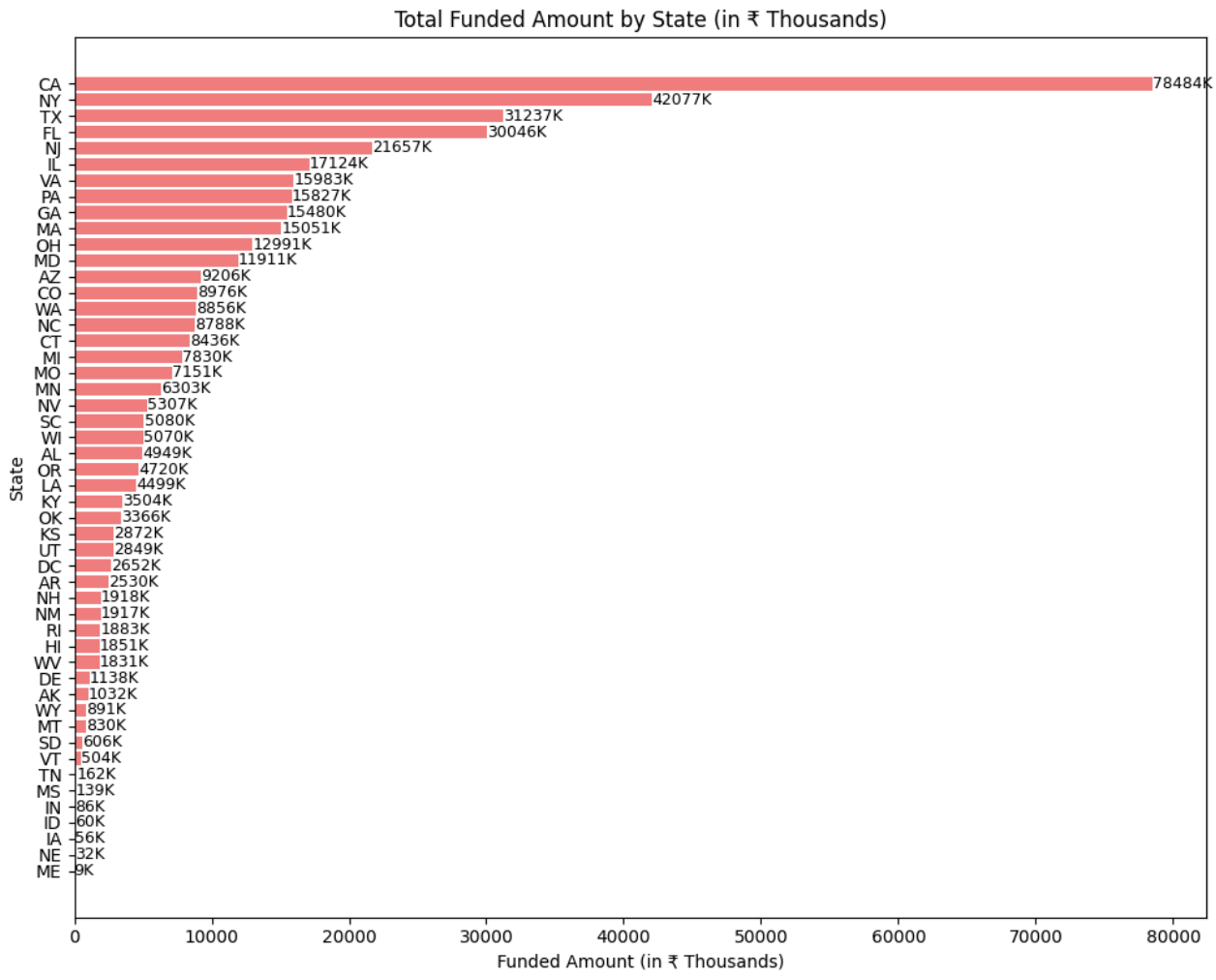
Monthly Trend by Issue Date for Total Loan Applications

```
montly_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_applications')
)

plt.figure(figsize=(10, 5))
plt.fill_between(montly_applications['month_name'], montly_applications['loan_applications'], color='orange', alpha=0.5)
plt.plot(montly_applications['month_name'], montly_applications['loan_applications'],
         color='darkorange', linewidth=2)

for i, row in montly_applications.iterrows():
    plt.text(i, row['loan_applications'] + 0.5, f"{row['loan_applications']}",
             ha='center', va='bottom', fontsize=9, rotation=0,color='black')

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



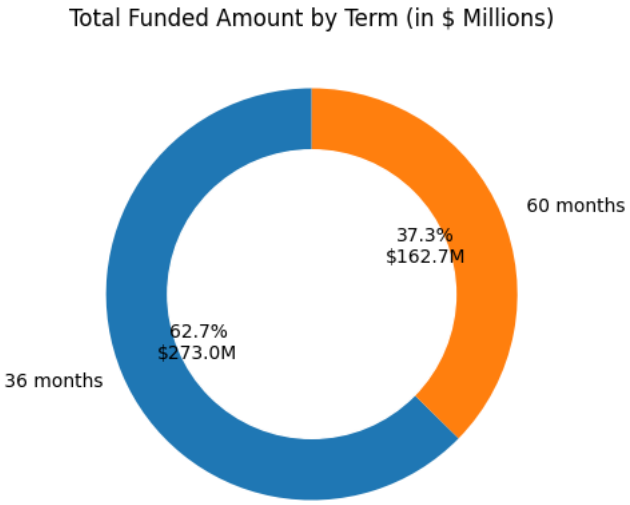
Regional Analysis by State for Total Funded Amount

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


states_reversed = state_funding_thousands.index[::-1]
amounts_reversed = state_funding_thousands.values[::-1]

plt.figure(figsize=(10, 8))
bars = plt.barh(states_reversed, amounts_reversed, color='lightcoral')

for i, bar in enumerate(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 (in ₹ Thousands)')
plt.ylabel('State')
plt.tight_layout()
plt.show()
```



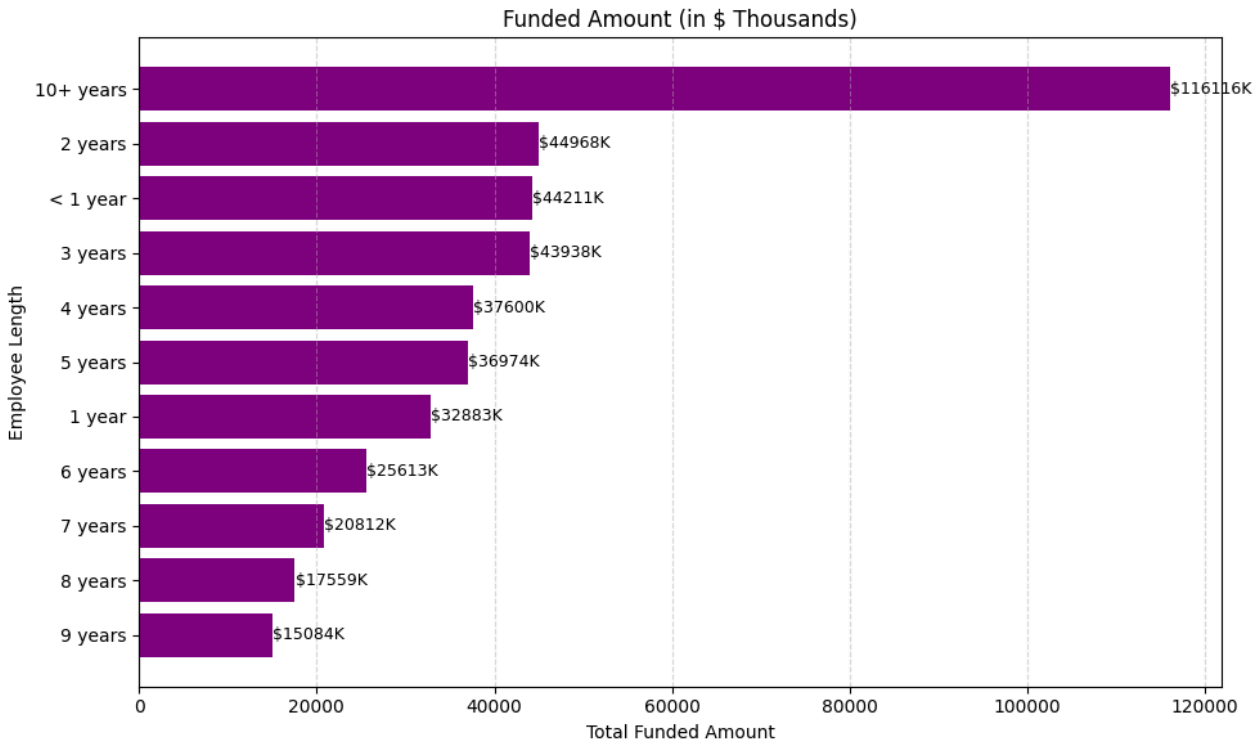
Loan Term Analysis by Total Funded Amount

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

plt.figure(figsize=(5, 5))
ax = plt.gca()
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}
)
# Use add_artist to add the circle to the axes
circle = plt.Circle((0, 0), 0.7, color='white')
ax.add_artist(circle)
plt.title('Total Funded Amount by Term (in $ Millions)')
plt.show()
```



Employee Length by Total Funded Amount

```
emp_funding_thousands = df.groupby('emp_length')['loan_amount'].sum().sort_values()/1000

plt.figure(figsize=(10, 6))
bars = plt.barh(emp_funding_thousands.index, emp_funding_thousands.values, color='purple')

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.title('Funded Amount (in $ Thousands)')
plt.xlabel('Total Funded Amount')
plt.ylabel('Employee Length')
plt.grid(axis='x', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```



Loan Purpose by Total Funded Amount

```
purpose_funding_millions = df.groupby('purpose')['loan_amount'].sum().sort_values()/1000

plt.figure(figsize=(10, 6))
bars = plt.barh(purpose_funding_millions.index, purpose_funding_millions.values, color='skyblue')

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.title('Total Funded Amount by Loan Purpose (in $ Millions)', fontsize=14)
plt.xlabel('Funded Amount ($ Millions)')
plt.ylabel('Loan Purpose')
plt.grid(axis='x', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```