Bank Loan Analysis | Python Project

Problem Statement

BRD 1: SUMMARY

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.
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 metric.
3. **Total Amount Received:** Tracking the total amount received from borrowers is essential for assessing the bank's cash flow and loan repayment. We should analyse the Month-to-Date (MTD) Total Amount Receive.
4. **Average Interest Rate:** Calculating the average interest rate across all loans which will provide insights into our lending portfolio's overall cost.
5. **Average Debt-to-Income Ratio (DTI):** Evaluating the average DTI for our borrowers helps us gauge their financial health. We need to compute the average DTI for all loans.

In [3]:
```
#importing necessary libs for the project
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 [4]:
```
#importing the required dataset
df = pd.read_csv(" *** PATH OF THE DATASET *** ")

# print the first 5 rows of the dataset/ for last 5 rows: df.tail(5)
df.head(5)
```

Out[4]:

	id	address_state	application_type	emp_length	emp_title	grade	home_ownership	issue_date	last_credit_pull_date	last_payment_date	...	sub_grade	term	verification_status	annual_inc
0	1077430	GA	INDIVIDUAL	< 1 year	Ryder	C	RENT	2/11/2021	9/13/2021	4/13/2021	...	C4	60 months	Source Verified	30(
1	1072053	CA	INDIVIDUAL	9 years	MKC Accounting	E	RENT	1/1/2021	12/14/2021	1/15/2021	...	E1	36 months	Source Verified	48(
2	1069243	CA	INDIVIDUAL	4 years	Chemat Technology Inc	C	RENT	1/5/2021	12/12/2021	1/9/2021	...	C5	36 months	Not Verified	50(
3	1041756	TX	INDIVIDUAL	< 1 year	barnes distribution	B	MORTGAGE	2/25/2021	12/12/2021	3/12/2021	...	B2	60 months	Source Verified	42(
4	1068350	IL	INDIVIDUAL	10+ years	J&J Steel Inc	A	MORTGAGE	1/1/2021	12/14/2021	1/15/2021	...	A1	36 months	Verified	83(

5 rows × 24 columns

In [5]:
```
#Metadata of the data

print("Number of Rows:", df.shape[0])       # 0 for rows
print("Number of Columns:", df.shape[1])    # 1 for columns
```

```
Number of Rows: 38576
Number of Columns: 24
```

In [6]:
```
#Types of data within Dataset

df.dtypes
```

Out[6]:
```
id                       int64
address_state           object
application_type        object
emp_length              object
emp_title               object
grade                   object
home_ownership          object
issue_date              object
last_credit_pull_date   object
last_payment_date       object
loan_status             object
next_payment_date       object
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
```

In [7]:
```
#Summary of the dataset

df.describe()
```

Out[7]:

	id	member_id	annual_income	dti	installment	int_rate	loan_amount	total_acc	total_payment
count	3.857600e+04	3.857600e+04	3.857600e+04	38576.000000	38576.000000	38576.000000	38576.000000	38576.000000	38576.000000
mean	6.810371e+05	8.476515e+05	6.964454e+04	0.133274	326.862965	0.120488	11296.066855	22.132544	12263.348533
std	2.113246e+05	2.668105e+05	6.429368e+04	0.066662	209.092000	0.037164	7460.746022	11.392282	9051.104777
min	5.473400e+04	7.069900e+04	4.000000e+03	0.000000	15.690000	0.054200	500.000000	2.000000	34.000000
25%	5.135170e+05	6.629788e+05	4.150000e+04	0.082100	168.450000	0.093200	5500.000000	14.000000	5633.000000
50%	6.627280e+05	8.473565e+05	6.000000e+04	0.134200	283.045000	0.118600	10000.000000	20.000000	10042.000000
75%	8.365060e+05	1.045652e+06	8.320050e+04	0.185900	434.442500	0.145900	15000.000000	29.000000	16658.000000
max	1.077501e+06	1.314167e+06	6.000000e+06	0.299900	1305.190000	0.245900	35000.000000	90.000000	58564.000000

1. Total Loan Applications

In [9]:
```
total_loan_applications = df['id'].count()

print(f"Number of total applications: {total_loan_applications}")
```

```
Number of total applications: 38576
```

1.1. Month-To-Date Total Loan Applications

In [11]:
```
# ✅ Convert issue_date to datetime first:

df['issue_date'] = pd.to_datetime(df['issue_date'])
```

In [12]:
```
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
```

2. Total Funded Amount

In [14]:
```
total_funded_amount = df['loan_amount'].sum()

total_funded_amount_in_millions = total_funded_amount/1000000

print(f"Total Funded Amount: {total_funded_amount_in_millions:.2f}M$")
```

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

2.1. Month-To-Date Total Funded Amount

In [16]:
```
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_in_millions = mtd_total_funded_amount/1000000

print(f"MTD Total Funded Amount: {mtd_total_funded_amount_in_millions:.2f}M$")
```

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

3. Total Amount Received

In [18]:
```
total_amount_received = df['total_payment'].sum()

total_amount_received_in_millions = total_amount_received/1000000

print(f"Total Amount Received: {total_amount_received_in_millions:.2f}M$")
```

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

3.1 Month-To-Date Total Amount Received

In [20]:
```
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_in_millions = mtd_total_amount_received/1000000

print(f"MTD Total Amount Received: {mtd_total_amount_received_in_millions:.2f}M$")
```

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

4. Average Interest Rate

In [22]:
```
average_interest_rate = df['int_rate'].mean()*100
print(f"Average Interest Rate: {average_interest_rate:.2f}%")
```

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

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

In [24]:
```
average_DTI = df['dti'].mean()*100
print(f"Average DTI: {average_DTI:.2f}%")
```

```
Average DTI: 13.33%
```

Good Loan Metrics

In [26]:
```
good_loan = df[df['loan_status'].isin(["Fully Paid", "Current"])]

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

good_loan_funded_amount = good_loan['loan_amount'].sum()
good_loan_received = good_loan['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_applications) * 100

print(f"Good Loan Applications:               {good_loan_applications} Applicants")
print(f"Good Loan Funded Amount (Millions):   {good_loan_funded_amount_millions:.2f} M$")
print(f"Good Loan Received Amount (Millions): {good_loan_received_millions:.2f} M$")
print(f"Percentage of Good Loan Applications: {good_loan_percentage:.2f}%")
```

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

Bad Loan Metrics

In [28]:
```python
bad_loan = df[df['loan_status'].isin(["Charged Off"])]

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

bad_loan_funded_amount = bad_loan['loan_amount'].sum()
bad_loan_received = bad_loan['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_applications) * 100

print(f"{'Bad Loan Applications':<36}: {bad_loan_applications:>4} Applicants")
print(f"{'Bad Loan Funded Amount (Millions)':<36}: {bad_loan_funded_amount_millions:>5,.2f} M$")
print(f"{'Bad Loan Received Amount (Millions)':<36}: {bad_loan_received_millions:>5,.2f} M$")
print(f"{'Percentage of Bad Loan Applications':<36}: {bad_loan_percentage:>4,.2f}%")

#                             ^^^^^ Second Way to Print the Values ^^^^^
```

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

Problem Statement

BRD 2: Overview

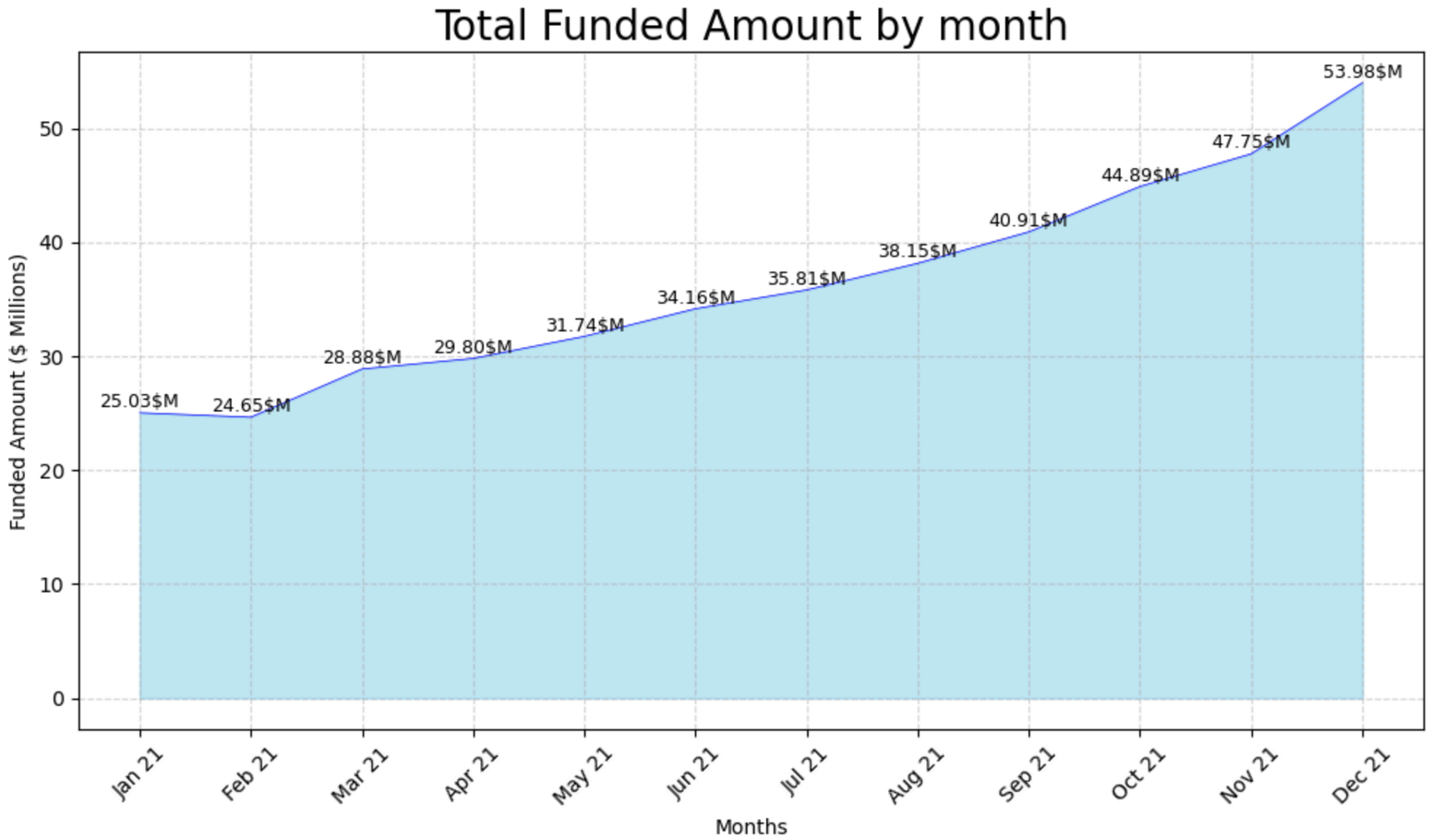
Charts

1. **Monthly Trends by Issue Date (Line/ Area Chart):** To identify seasonality and long-term trends in lending activities.
2. **Regional Analysis by State (Bar Chart):** To identify regions with significant lending activity and assess regional disparities.
3. **Loan Term Analysis (Donut Chart):** To allow the client to understand the distribution of loans across various term lengths.
4. **Employee Length Analysis (Bar Chart):** How lending metrics are distributed among borrowers with different employment lengths, helping us assess the impact of employment history on loan applications.
5. **Loan Purpose Breakdown (Bar Chart):** Will provide a visual breakdown of loan metrics based on the stated purposes of loans, aiding in the understanding of the primary reasons borrowers seek financing.
6. **Home Ownership Analysis (Tree/ Heat Map):** For a hierarchical view of how home ownership impacts loan applications and disbursements.

Metrics to be shown: 'Total Loan Applications,' 'Total Funded Amount,' and 'Total Amount Received'

1. Monthly Trends by Issue Date for Total Funded Amount

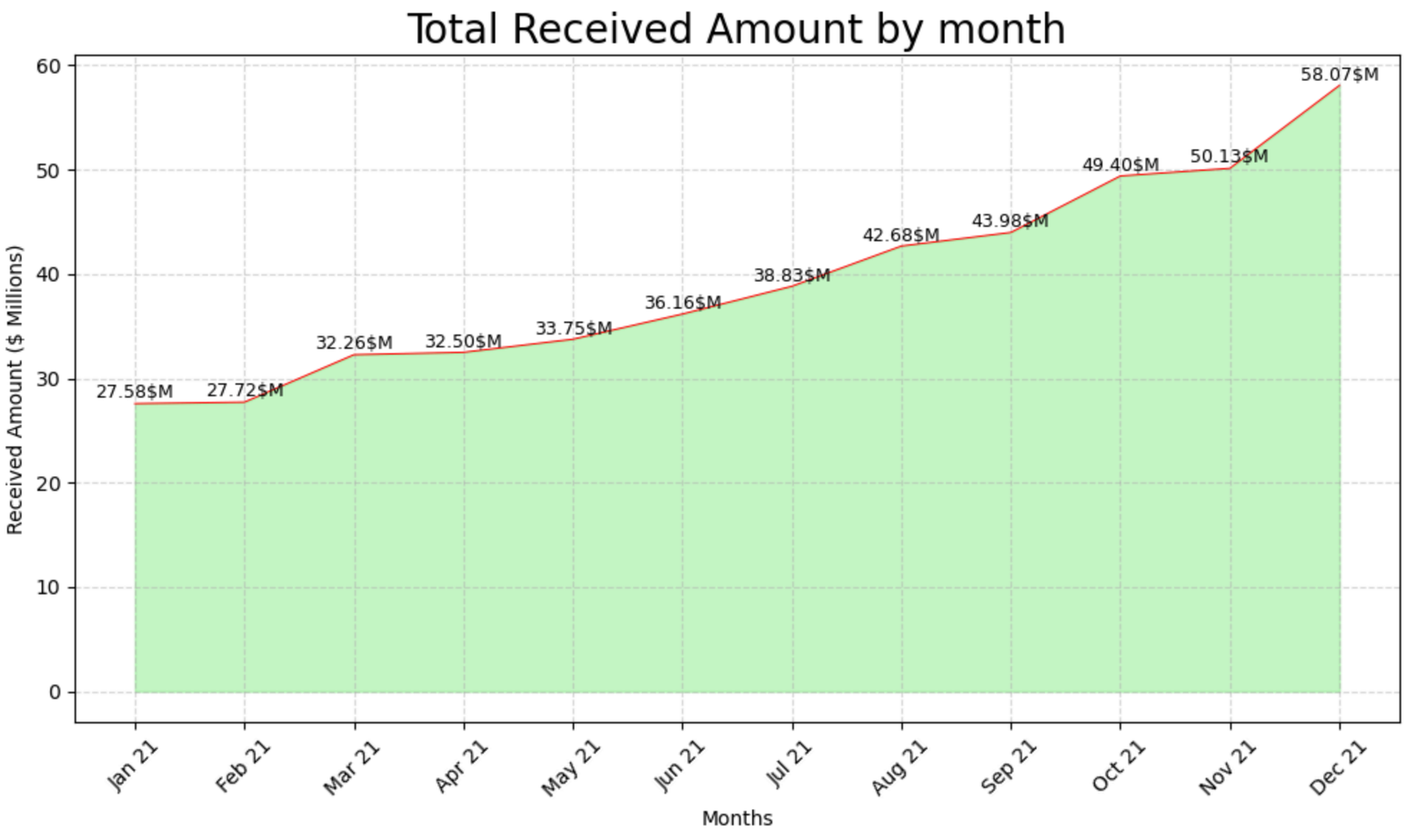
In [31]:
```
monthly_funded = (
df.sort_values('issue_date')
  .assign(month_name=lambda x: x['issue_date'].dt.strftime('%b %y'))
  .groupby('month_name', sort=False)['loan_amount']
  .sum()
  .div(1000000)
  .reset_index(name='loan_amount_millions')
)

plt.figure(figsize=(10,6))
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=0.5)

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

plt.title('Total Funded Amount by month', fontsize = 20)
plt.xlabel('Months')
plt.ylabel('Funded Amount ($ Millions)')
plt.xticks(ticks=range(len(monthly_funded)), label=monthly_funded['month_name'], rotation = 45)
plt.grid(True, linestyle = '--', alpha = 0.5)
plt.tight_layout()
plt.show()
```



1.1. Monthly Trends by Issue Date for Total Amount Received

In [33]:
```
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=(10,6))
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='red', linewidth=0.7)

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

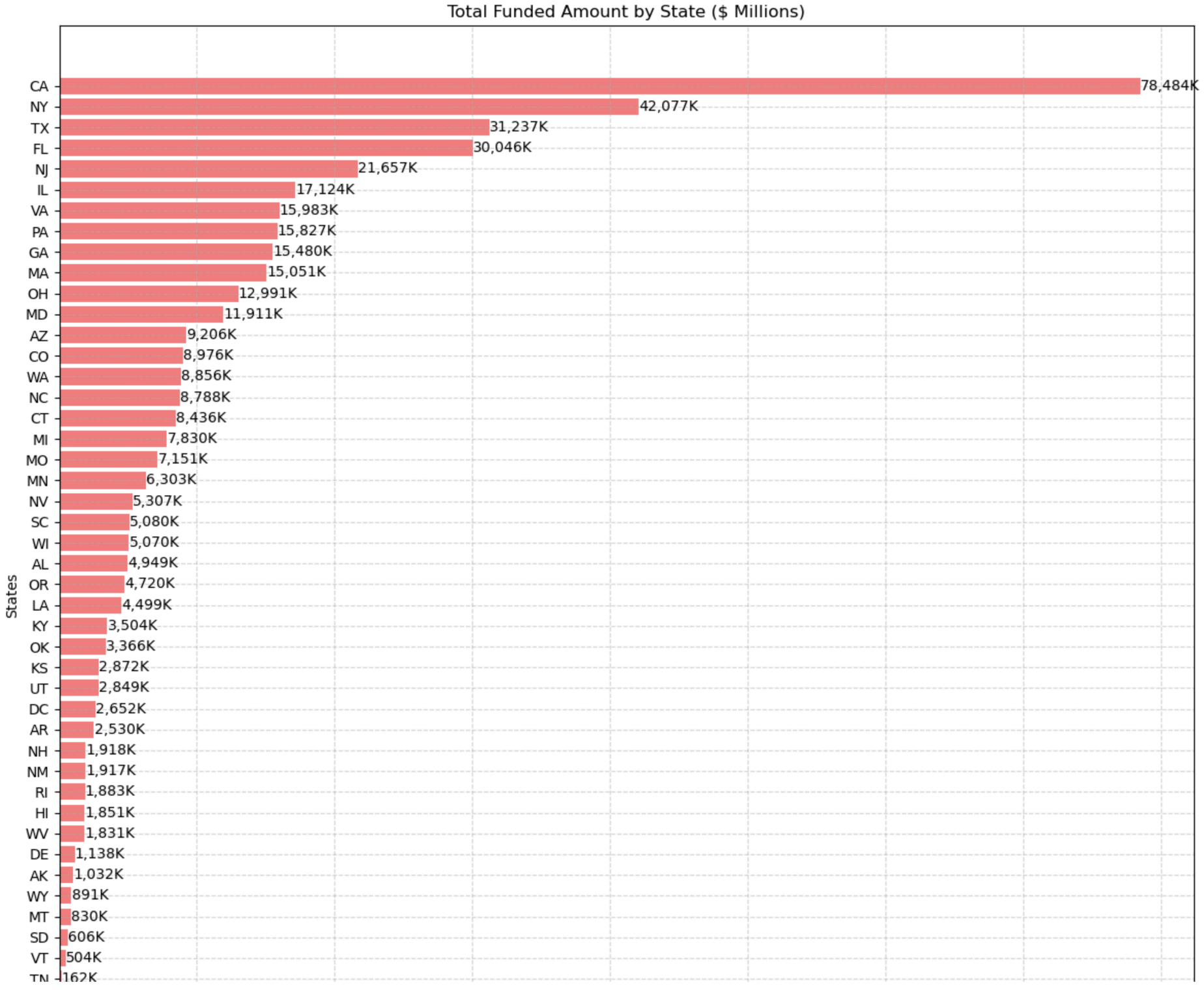
```
plt.title('Total Received Amount by month', fontsize = 20)
plt.xlabel('Months')
plt.ylabel('Received Amount ($ Millions)')
plt.xticks(ticks=range(len(monthly_received)), labels=monthly_received['month_name'], rotation = 45)
plt.grid(True, linestyle = '--', alpha = 0.5)
plt.tight_layout()
plt.show()
```

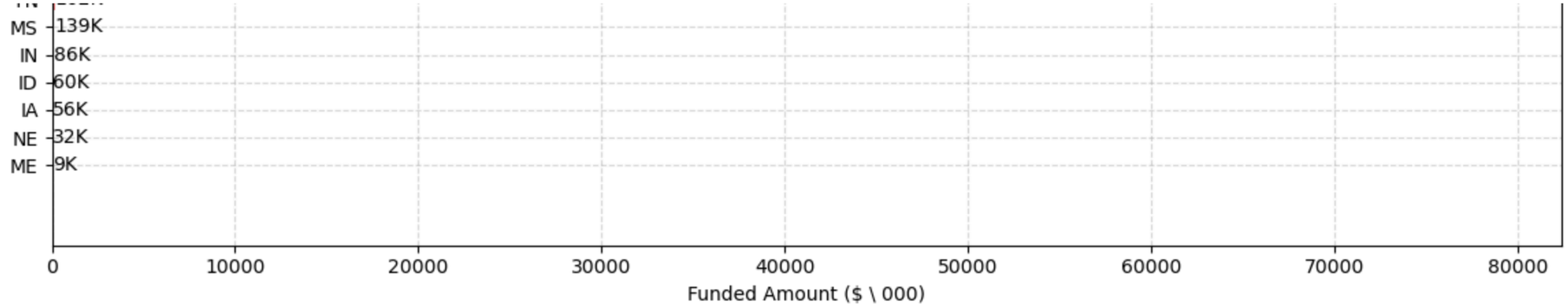


2. Regional Analysis by State for Total Funded Amount

In [35]:
```
state_funding = df.groupby('address_state')['loan_amount'].sum().sort_values(ascending = True)
state_funding_thousands = state_funding / 1000

plt.figure(figsize=(12,12))
bars = plt.barh(state_funding_thousands.index, state_funding_thousands.values, color='lightcoral')

for bar in bars:
    width = bar.get_width()
    plt.text(width + 10, bar.get_y() + bar.get_height() / 2,
```

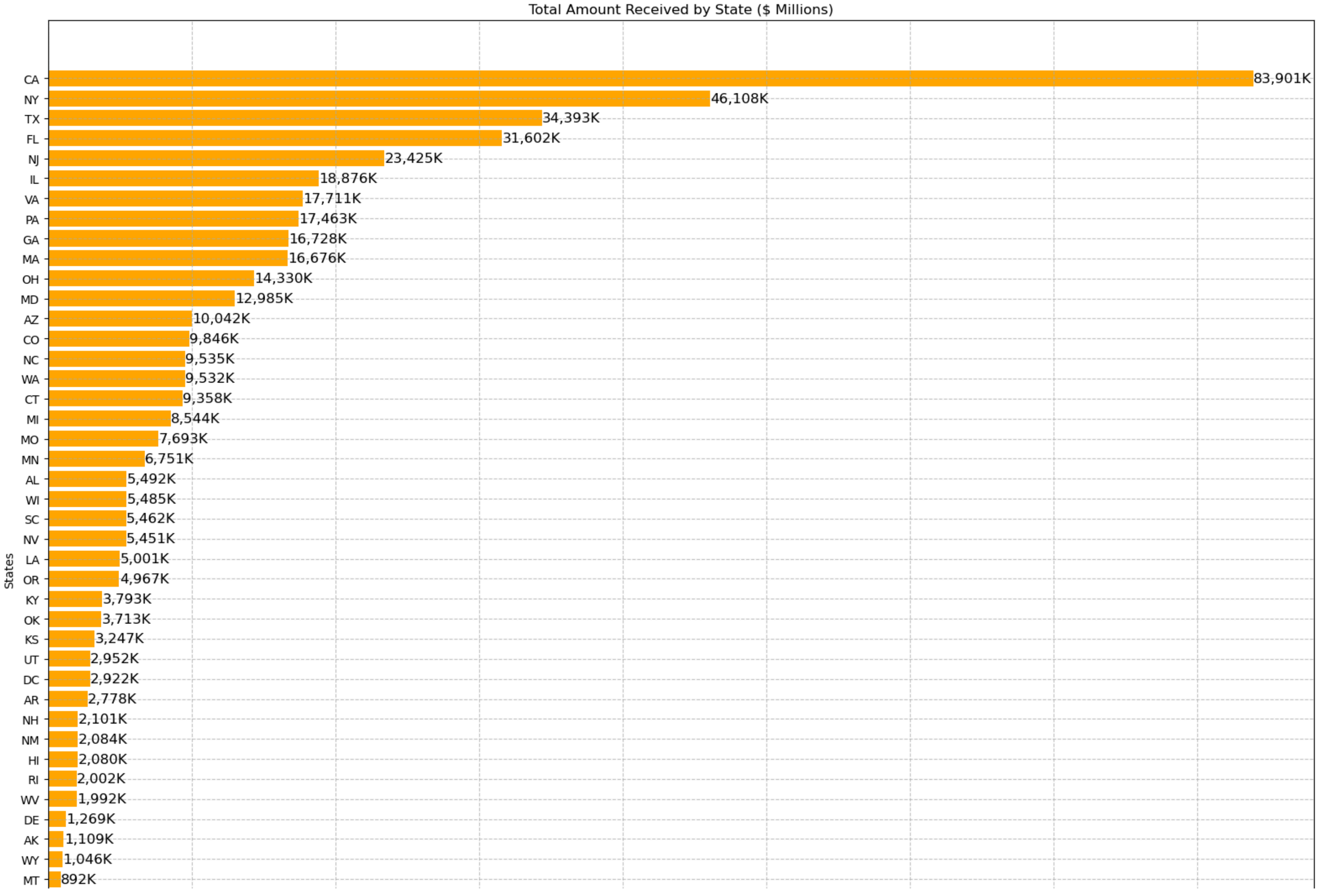
```
            f'{width:,.0f}K', va = 'center', fontsize = 10)

plt.title('Total Funded Amount by State ($ Millions)')
plt.xlabel('Funded Amount ($ \ 000)')
plt.ylabel('States')
plt.grid(True, linestyle = '--', alpha = 0.5)
plt.tight_layout()
plt.show()
```

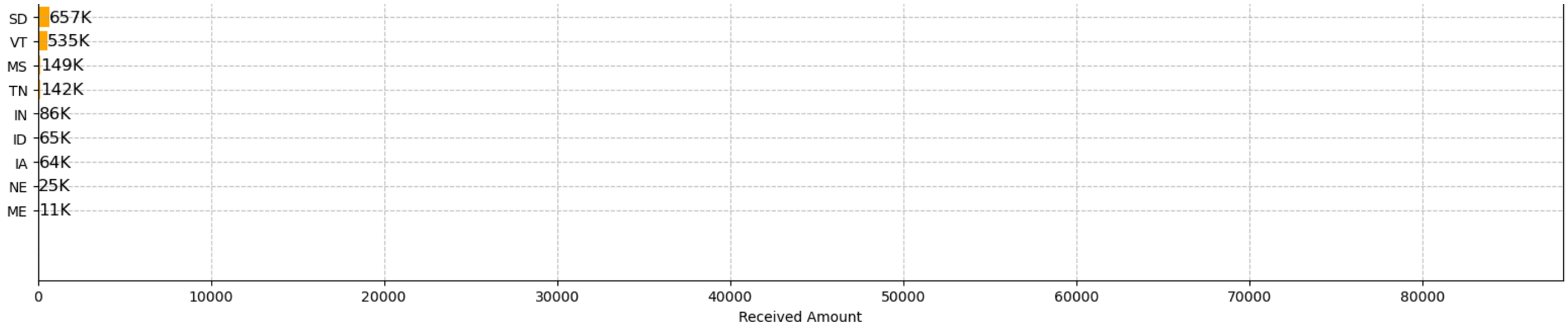

Total Funded Amount by State ($ Millions)
States
CA 78,484K
NY 42,077K
TX 31,237K
FL 30,046K
NJ 21,657K
IL 17,124K
VA 15,983K
PA 15,827K
GA 15,480K
MA 15,051K
OH 12,991K
MD 11,911K
AZ 9,206K
CO 8,976K
WA 8,856K
NC 8,788K
CT 8,436K
MI 7,830K
MO 7,151K
MN 6,303K
NV 5,307K
SC 5,080K
WI 5,070K
AL 4,949K
OR 4,720K
LA 4,499K
KY 3,504K
OK 3,366K
KS 2,872K
UT 2,849K
DC 2,652K
AR 2,530K
NH 1,918K
NM 1,917K
RI 1,883K
HI 1,851K
WV 1,831K
DE 1,138K
AK 1,032K
WY 891K
MT 830K
SD 606K
VT 504K
TN 162K



2.1. Regional Analysis by State for Total Amount Received

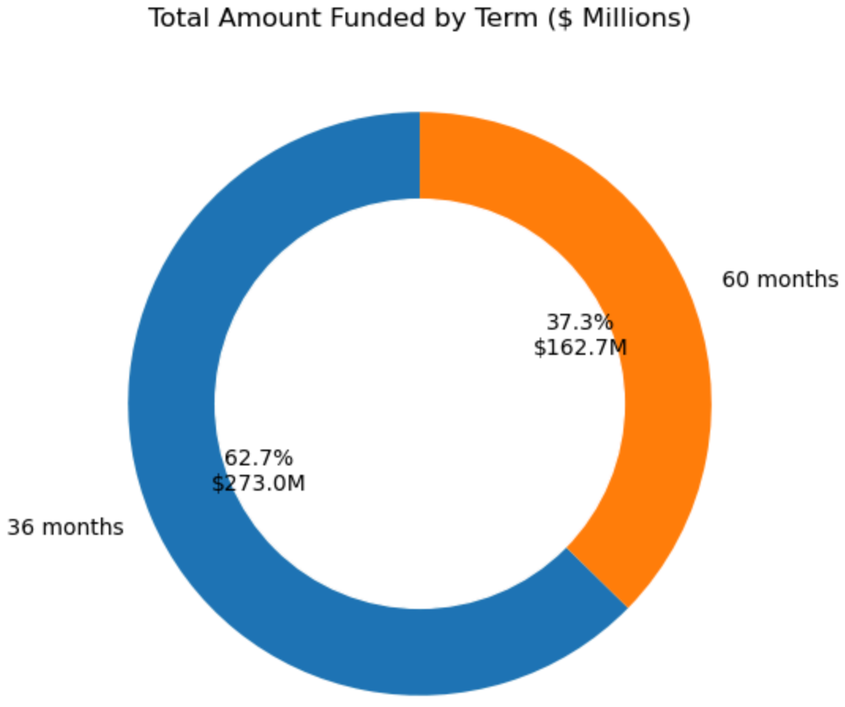
In [37]:
```
state_received = df.groupby('address_state')['total_payment'].sum().sort_values(ascending = True)
state_received_thousands = (state_received / 1000).round(2)


plt.figure(figsize=(16,14))
bars = plt.barh(state_received_thousands.index, state_received_thousands.values, color='orange')

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 = 12)

plt.title('Total Amount Received by State ($ Millions)')
plt.xlabel('Received Amount')
plt.ylabel('States')
plt.grid(True, linestyle = '--', alpha = 0.7)
plt.tight_layout()
plt.show()
```


Total Amount Received by State ($ Millions)
States
CA 83,901K
NY 46,108K
TX 34,393K
FL 31,602K
NJ 23,425K
IL 18,876K
VA 17,711K
PA 17,463K
GA 16,728K
MA 16,676K
OH 14,330K
MD 12,985K
AZ 10,042K
CO 9,846K
NC 9,535K
WA 9,532K
CT 9,358K
MI 8,544K
MO 7,693K
MN 6,751K
AL 5,492K
WI 5,485K
SC 5,462K
NV 5,451K
LA 5,001K
OR 4,967K
KY 3,793K
OK 3,713K
KS 3,247K
UT 2,952K
DC 2,922K
AR 2,778K
NH 2,101K
NM 2,084K
HI 2,080K
RI 2,002K
WV 1,992K
DE 1,269K
AK 1,109K
WY 1,046K
MT 892K



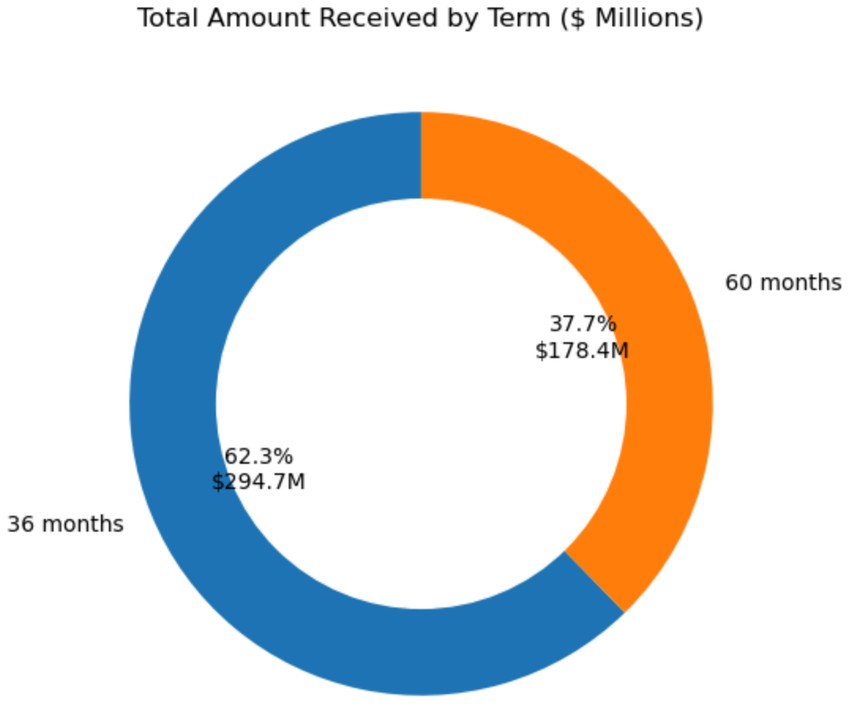
3. Loan Term Analysis for Total Amount Funded

In [39]:
```
term_funding_millions = df.groupby ('term')['loan_amount'].sum() / 1000000

plt.figure(figsize=(6,6))
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' : 0.9}
)

plt.gca().add_artist(plt.Circle((0,0), 0.70, color = 'white'))
plt.title("Total Amount Funded by Term ($ Millions)")
plt.show()
```



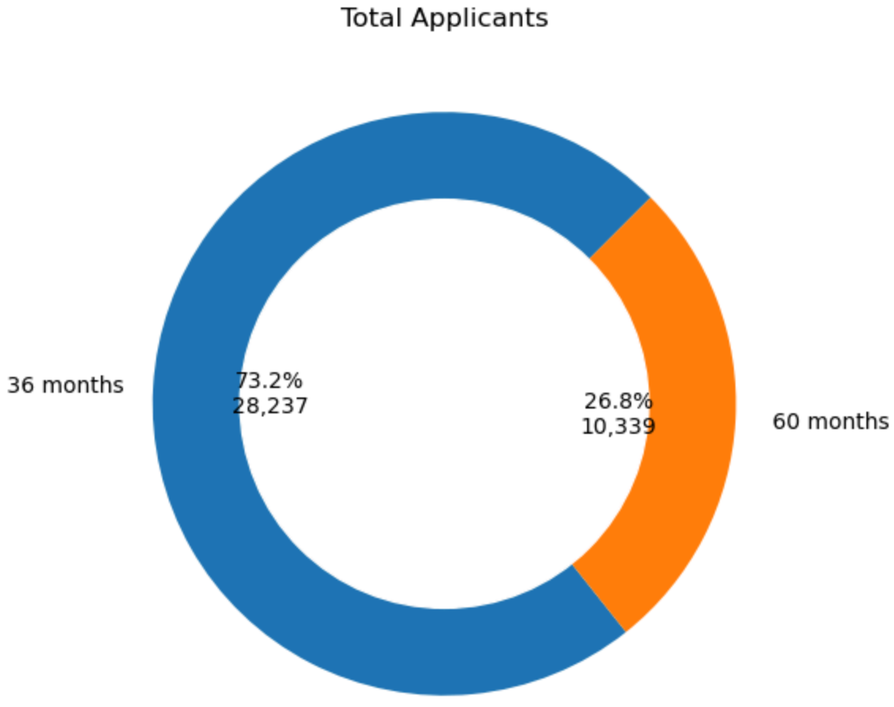
3.1 Loan Term Analysis for Total Amount Received

In [41]:
```
term_received_millions = df.groupby ('term')['total_payment'].sum() / 1000000

plt.figure(figsize=(6,6))
plt.pie(
    term_received_millions,
    labels=term_received_millions.index,
    autopct=lambda p: f"{p:.1f}%\n${p*sum(term_received_millions)/100:.1f}M",
    startangle = 90,
    wedgeprops = {'width' : 0.9}
)

plt.gca().add_artist(plt.Circle((0,0), 0.70, color = 'white'))
plt.title("Total Amount Received by Term ($ Millions)")
plt.show()
```



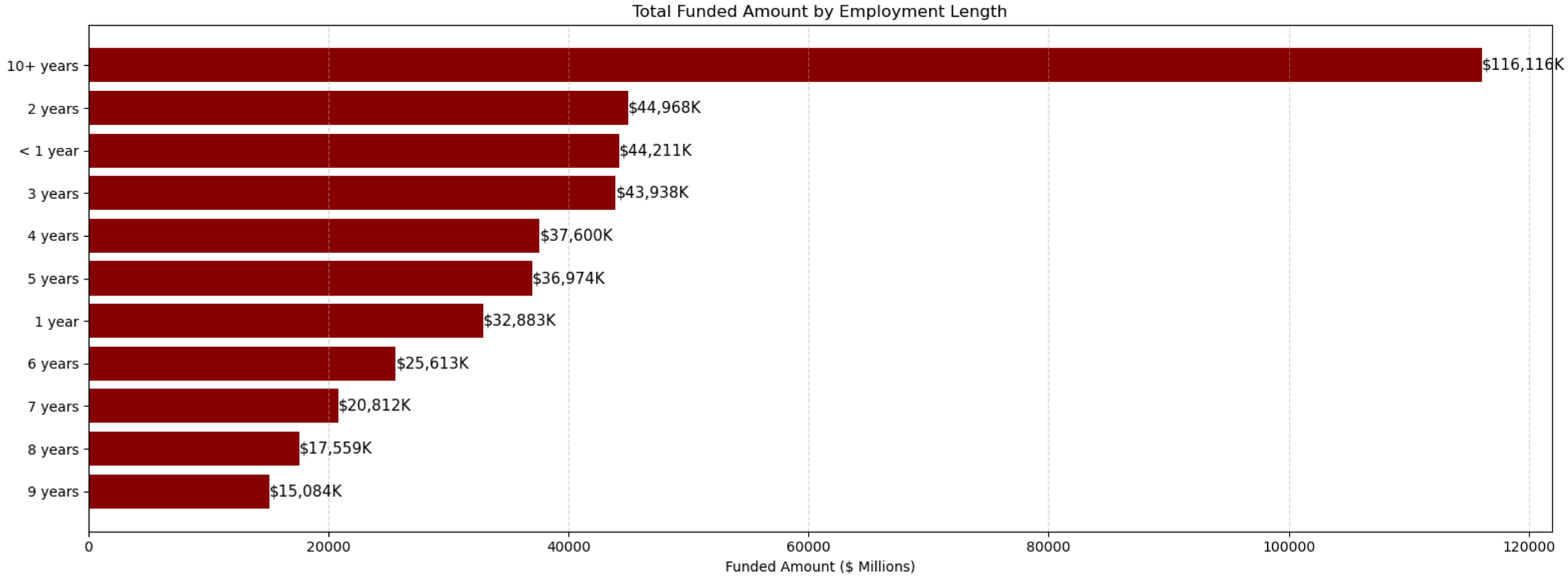
3.2 Loan Term Analysis for Total Applicants

In [43]:
```
term_total_applicants = df.groupby('term')['id'].count()

plt.figure(figsize=(6,6))
plt.pie(
    term_total_applicants,
    labels=term_total_applicants.index,
    autopct=lambda p: f"{p:.1f}%\n{p*sum(term_total_applicants)/100:,.0f}",
    startangle=45,
    wedgeprops={'width': 1}
)

plt.gca().add_artist(plt.Circle((0,0), 0.70, color='white'))
plt.title("Total Applicants")
plt.show()
```



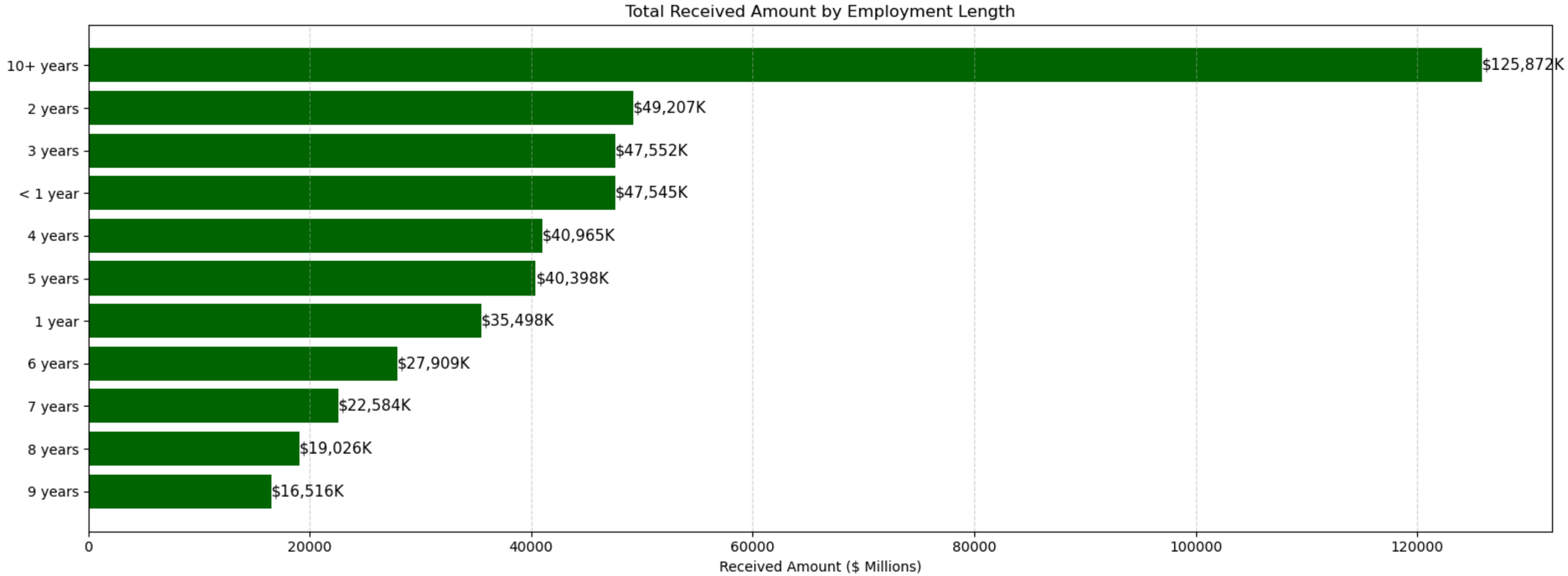
4. Employee Length Analysis for Total Amount Funded

In [45]:
```
emp_funding = df.groupby('emp_length')['loan_amount'].sum().sort_values()/1000

plt.figure(figsize=(16,6))
bars = plt.barh(emp_funding.index, emp_funding, color = 'darkred')

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 = 11)

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



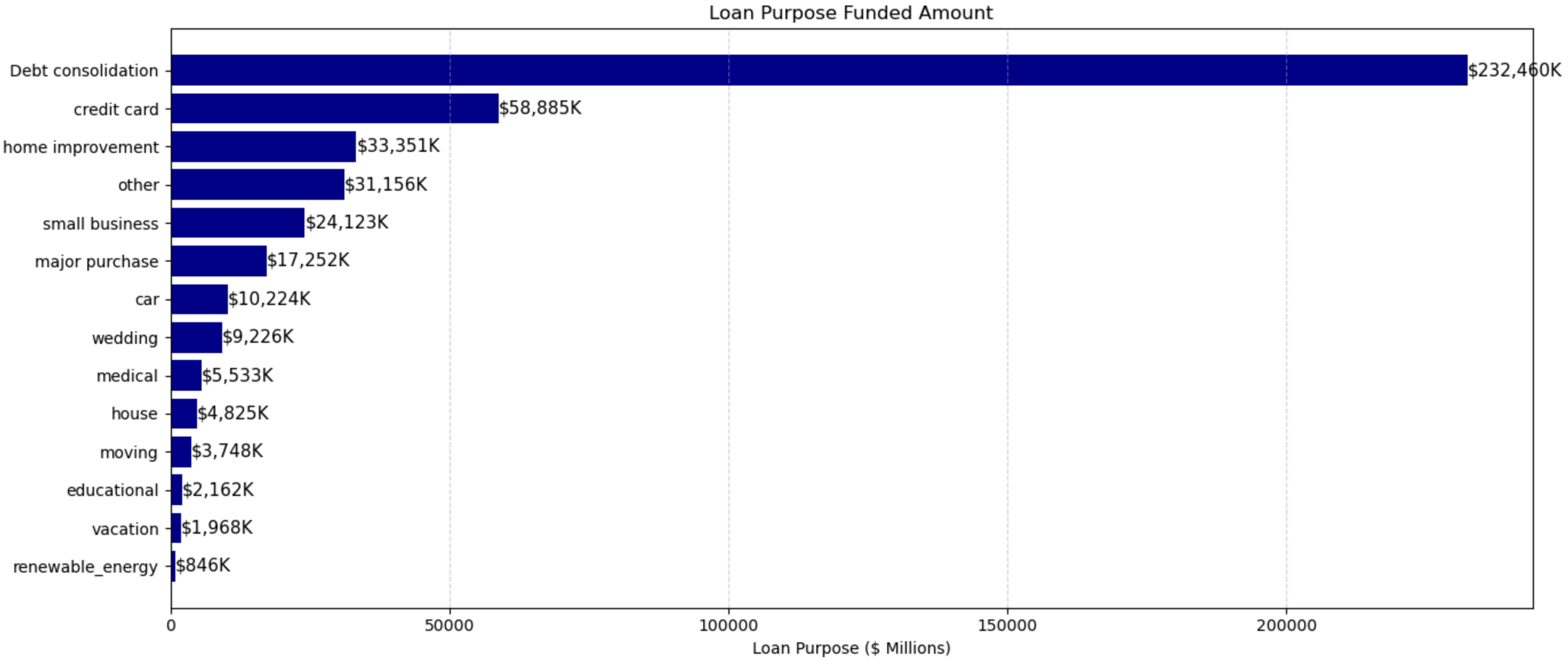
4.1. Employee Length Analysis for Total Amount Received

In [47]:
```
emp_received = df.groupby('emp_length')['total_payment'].sum().sort_values()/1000

plt.figure(figsize=(16,6))
bars = plt.barh(emp_received.index, emp_received, color = 'darkgreen')

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 = 11)

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



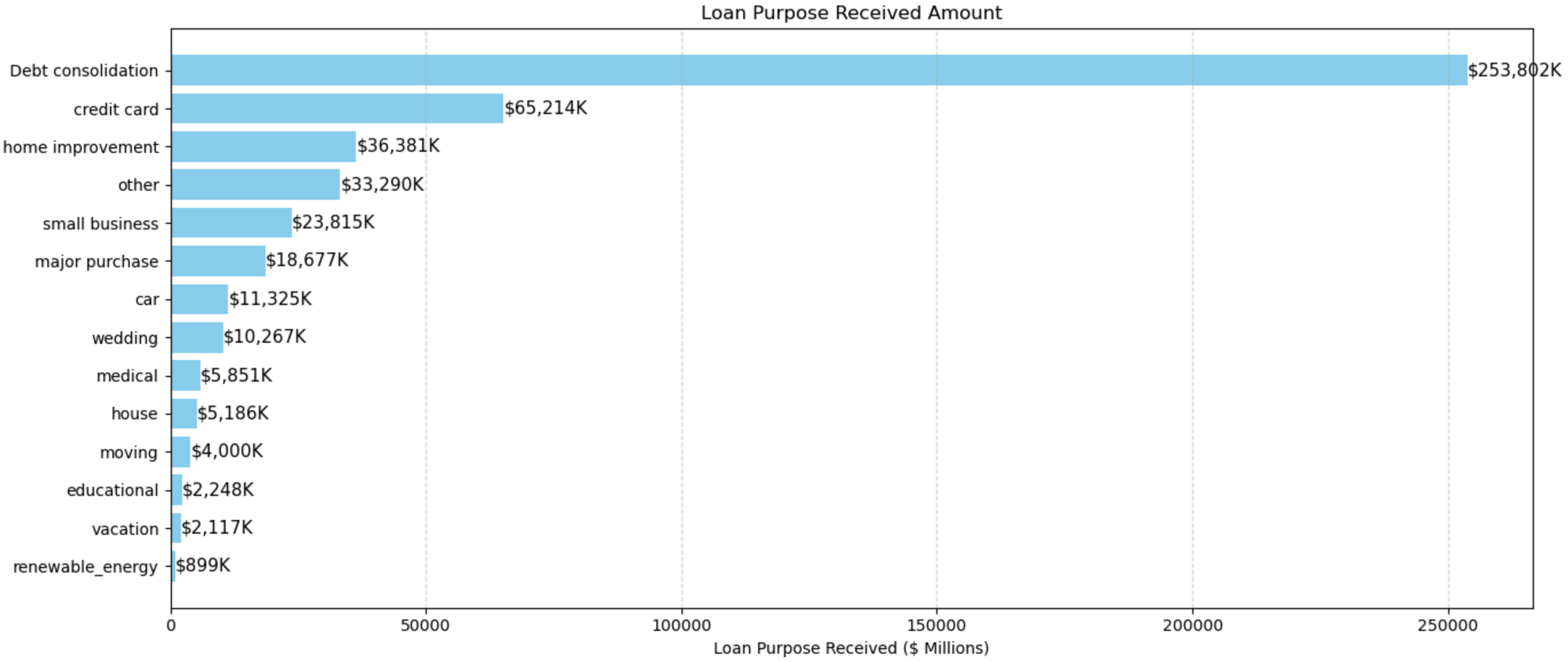
5. Loan Purpose Breakdown for Total Amount Funded

In [49]:
```
purpose_funding = df.groupby('purpose')['loan_amount'].sum().sort_values()/1000

plt.figure(figsize=(14,6))
bars = plt.barh(purpose_funding.index, purpose_funding, color = 'darkblue')

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 = 11)

plt.xlabel("Loan Purpose ($ Millions)")
plt.title("Loan Purpose Funded Amount")
plt.grid(axis = 'x', linestyle = '--', alpha = 0.5)
plt.tight_layout()
plt.show()
```



5.1. Loan Purpose Breakdown for Total Amount Received

In [51]:
```
purpose_funding_received = df.groupby('purpose')['total_payment'].sum().sort_values()/1000

plt.figure(figsize=(14,6))
bars = plt.barh(purpose_funding_received.index, purpose_funding_received, 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 = 11)

plt.xlabel("Loan Purpose Received ($ Millions)")
plt.title("Loan Purpose Received Amount")
plt.grid(axis = 'x', linestyle = '--', alpha = 0.5)
plt.tight_layout()
plt.show()
```



6. Home Ownership Analysis by Total Funded Amount

In [53]:
```
# pip install notebook --upgrade

# import plotly.io as pio
# pio.renderers.default = 'notebook'  # or 'iframe' or 'plotly_mimetype'
```

In [54]:
```
home_funding = df.groupby('home_ownership')['loan_amount'].sum().reset_index()
home_funding['loan_amount_millions'] = home_funding['loan_amount'] / 1000000

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

```
fig.show()
```

6.1. Home Ownership Analysis by Total Received Amount

In [56]:
```
home_funding_received = df.groupby('home_ownership')['total_payment'].sum().reset_index()
home_funding_received['loan_amount_millions'] = home_funding_received['total_payment'] / 1000000

fig = px.treemap(
      home_funding,
      path = ['home_ownership'],
      values = 'loan_amount_millions',
      color = 'loan_amount_millions',
      color_continuous_scale = 'Viridis',
      title = 'Total Received Amount from Home Ownership ($ Millions)'
)

fig.show()
```

Total Received Amount from Home Ownership ($ Millions)