BANK LOAN ANALYSIS - FINANCE DOMAIN

In [2]:
```
#importing necessary dependencies

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 [3]:
```
#loading the dataset

df = pd.read_excel("financial_loan_data_excel.xlsx")
df
```

Out[3]:

	id	address_state	application_type	emp_length	emp_title	grade	home_
0	1077430	GA	INDIVIDUAL	< 1 year	Ryder	C	
1	1072053	CA	INDIVIDUAL	9 years	MKC Accounting	E	
2	1069243	CA	INDIVIDUAL	4 years	Chemat Technology Inc	C	
3	1041756	TX	INDIVIDUAL	< 1 year	barnes distribution	B	
4	1068350	IL	INDIVIDUAL	10+ years	J&J Steel Inc	A	
...	...	...	...	...	...	...	
38571	803452	NJ	INDIVIDUAL	< 1 year	Joseph M Sanzari Company	C	
38572	970377	NY	INDIVIDUAL	8 years	Swat Fame	C	
38573	875376	CA	INDIVIDUAL	5 years	Anaheim Regional Medical Center	D	
38574	972997	NY	INDIVIDUAL	5 years	Brooklyn Radiology	D	
38575	682952	NY	INDIVIDUAL	4 years	Allen Edmonds	F	

38576 rows × 24 columns

In [4]:
```
#first five records

df.head()
```

Out[4]:

	id	address_state	application_type	emp_length	emp_title	grade	home_own
0	1077430	GA	INDIVIDUAL	< 1 year	Ryder	C	
1	1072053	CA	INDIVIDUAL	9 years	MKC Accounting	E	
2	1069243	CA	INDIVIDUAL	4 years	Chemat Technology Inc	C	
3	1041756	TX	INDIVIDUAL	< 1 year	barnes distribution	B	MOR
4	1068350	IL	INDIVIDUAL	10+ years	J&J Steel Inc	A	MOR

5 rows × 24 columns

In [5]:
```
#last five records

df.tail()
```

Out[5]:

	id	address_state	application_type	emp_length	emp_title	grade	home_ov
38571	803452	NJ	INDIVIDUAL	< 1 year	Joseph M Sanzari Company	C	MC
38572	970377	NY	INDIVIDUAL	8 years	Swat Fame	C	
38573	875376	CA	INDIVIDUAL	5 years	Anaheim Regional Medical Center	D	
38574	972997	NY	INDIVIDUAL	5 years	Brooklyn Radiology	D	
38575	682952	NY	INDIVIDUAL	4 years	Allen Edmonds	F	

5 rows × 24 columns

In [6]:
```
#no of rows and columns

df.shape
```

Out[6]:
```
(38576, 24)
```

In [7]:
```
#brief informations about the dataset

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38576 entries, 0 to 38575
Data columns (total 24 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   id                     38576 non-null  int64
 1   address_state          38576 non-null  object
 2   application_type       38576 non-null  object
 3   emp_length             38576 non-null  object
 4   emp_title              37138 non-null  object
 5   grade                  38576 non-null  object
 6   home_ownership         38576 non-null  object
 7   issue_date             38576 non-null  datetime64[ns]
 8   last_credit_pull_date  38576 non-null  datetime64[ns]
 9   last_payment_date      38576 non-null  datetime64[ns]
 10  loan_status            38576 non-null  object
 11  next_payment_date      38576 non-null  datetime64[ns]
 12  member_id              38576 non-null  int64
 13  purpose                38576 non-null  object
 14  sub_grade              38576 non-null  object
 15  term                   38576 non-null  object
 16  verification_status    38576 non-null  object
 17  annual_income          38576 non-null  float64
 18  dti                    38576 non-null  float64
 19  installment            38576 non-null  float64
 20  int_rate               38576 non-null  float64
 21  loan_amount            38576 non-null  int64
 22  total_acc              38576 non-null  int64
 23  total_payment          38576 non-null  int64
dtypes: datetime64[ns](4), float64(4), int64(5), object(11)
memory usage: 7.1+ MB
```

In [8]:
```
#statistical summary on numerical data

df.describe()
```

Out[8]:

	id	issue_date	last_credit_pull_date	last_payment_date	next_p
count	3.857600e+04	38576	38576	38576	
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	20:42:
min	5.473400e+04	2021-01-01 00:00:00	2021-01-08 00:00:00	2021-01-08 00:00:00	2021-0
25%	5.135170e+05	2021-04-11 00:00:00	2021-04-15 00:00:00	2021-03-16 00:00:00	2021-0
50%	6.627280e+05	2021-07-11 00:00:00	2021-05-16 00:00:00	2021-06-14 00:00:00	2021-0
75%	8.365060e+05	2021-10-11 00:00:00	2021-08-13 00:00:00	2021-09-15 00:00:00	2021-1
max	1.077501e+06	2021-12-12 00:00:00	2022-01-20 00:00:00	2021-12-15 00:00:00	2022-0
std	2.113246e+05	NaN	NaN	NaN	

BUSINESS PROBLEMS

In [9]:
```
#total loan applications

total_loan_applications = df.id.count()
print("Total Loan Applications:",total_loan_applications)
```

```
Total Loan Applications: 38576
```

In [10]:
```
#MTD total loan applications

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 =
mtd_loan_applications = mtd_data.id.count()

print("MTD Loan Applications:",mtd_loan_applications)
```

```
MTD Loan Applications: 4314
```

In [11]:
```
#total funded amount

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

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

In [12]:
```
#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 =
mtd_funded_amount = mtd_data.loan_amount.sum()
mtd_funded_amount_in_millions = mtd_funded_amount / 1000000

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

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

In [13]:
```
#total received amount

total_received_amount = df.total_payment.sum()
total_received_amount_in_millions = total_received_amount / 1000000
print("Total Received Amount: ${:.2f}M" . format(total_received_amount_in_millio
```

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

In [14]:
```
#MTD total received 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 =
mtd_received_amount = mtd_data.total_payment.sum()
```

```
mtd_received_amount_in_millions = mtd_received_amount / 1000000

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

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

In [15]:
```
#average interest rate

avg_int_rate = df.int_rate.mean()*100
print("Avg Interest Rate: {:.2f}%".format(avg_int_rate))
```

```
Avg Interest Rate: 12.05%
```

In [16]:
```
#average dti

avg_dti = df.dti.mean()*100
print("Avg DTI: {:.2f}%".format(avg_dti))
```

```
Avg DTI: 13.33%
```

In [17]:
```
#good loan metrics

good_loans = df[df.loan_status.isin(["Fully Paid","Current"])]

total_loan_applications = df.id.count()

good_loan_applications = good_loans["id"].count()
good_loan_funded_amount = good_loans["loan_amount"].sum()
good_loan_received_amount = good_loans["total_payment"].sum()

good_loan_funded_amount_in_millions = good_loan_funded_amount / 1000000
good_loan_received_amount_in_millions = good_loan_received_amount / 1000000

good_loan_percentage = good_loan_applications / total_loan_applications * 100

print("Good Loan Applications:",good_loan_applications)
print("Good Loan Funded Amount (in Millions): ${:.2f}M".format(good_loan_funded_
print("Good Loan Received Amount (in Millions): ${:.2f}M".format(good_loan_recei
print("Percentage of Good Loan Applications: {:.2f}%".format(good_loan_percentag
```

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

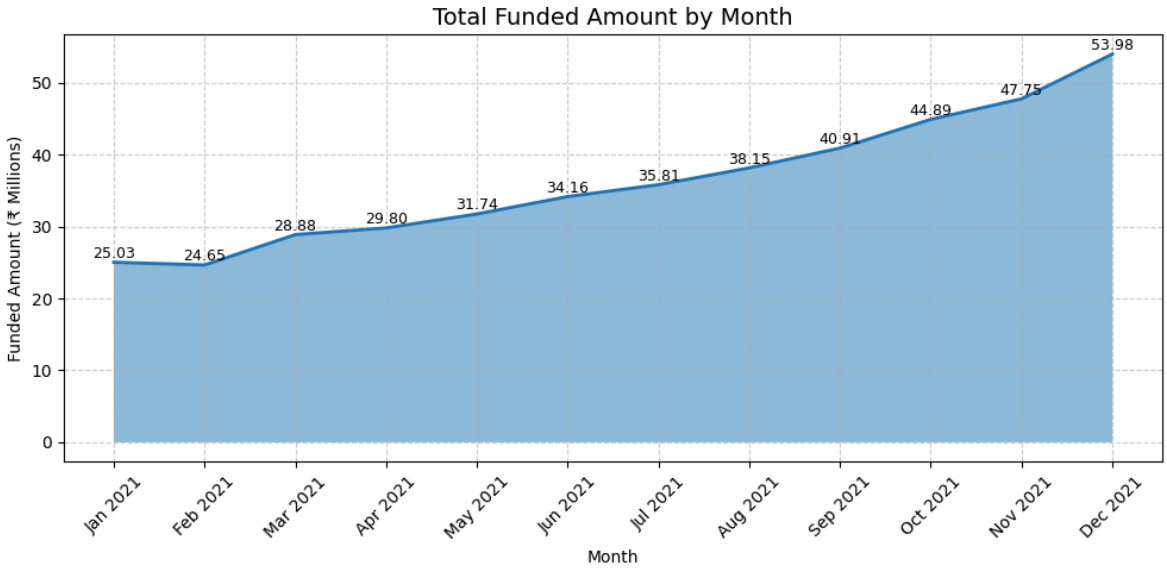
In [18]:
```
#bad loan metrics

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_received_amount = bad_loans["total_payment"].sum()

bad_loan_funded_amount_in_millions = bad_loan_funded_amount / 1000000
bad_loan_received_amount_in_millions = bad_loan_received_amount / 1000000

bad_loan_percentage = bad_loan_applications / total_loan_applications * 100

print("Bad Loan Applications:",bad_loan_applications)
print("Bad Loan Funded Amount (in Millions): ${:.2f}M".format(bad_loan_funded_am
```

```
print("Bad Loan Received Amount (in Millions): ${:.2f}M".format(bad_loan_receive
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 Received Amount (in Millions): $37.28M
Percentage of Bad Loan Applications: 13.82%
```

In [19]:

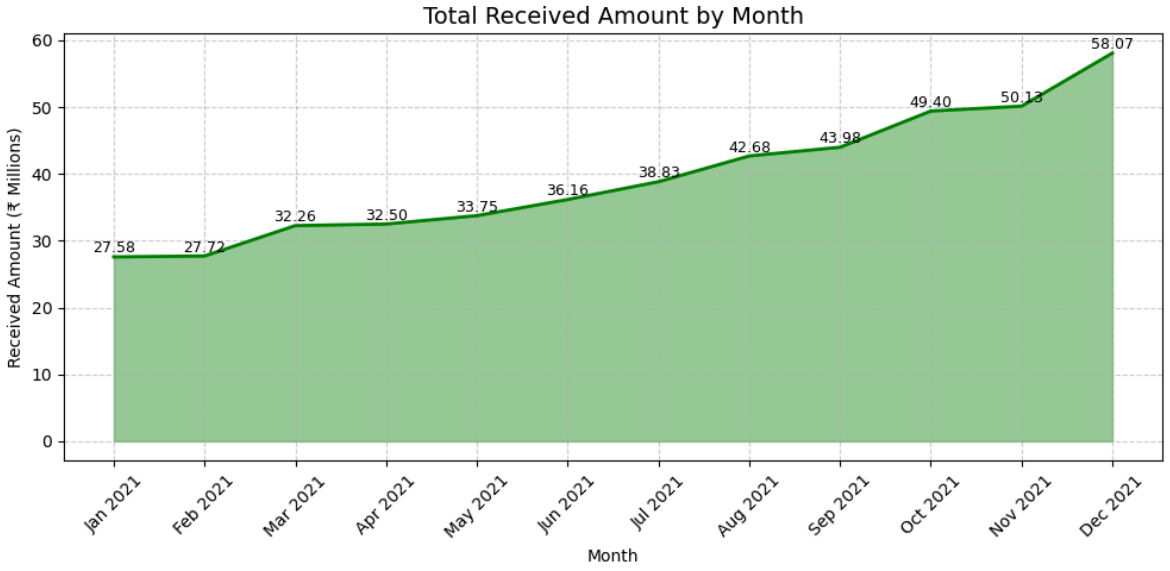
```
# ===== Monthly funded amount calculation =====

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(1_000_000)   # convert to millions
      .reset_index(name='loan_amount_millions')
)

# ===== Plot =====

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

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

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

# Add value labels
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
    )

# Titles and labels
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()
```



In [20]:

```python
# ===== Monthly received amount calculation =====

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(1_000_000)
      .reset_index(name='received_amount_millions')
)

# ===== Plot in GREEN =====

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

plt.fill_between(
    monthly_received['month_name'],
    monthly_received['received_amount_millions'],
    color='green',
    alpha=0.4
)

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

# Value labels
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
    )

plt.title('Total Received Amount by Month', fontsize=14)
plt.xlabel('Month')
```

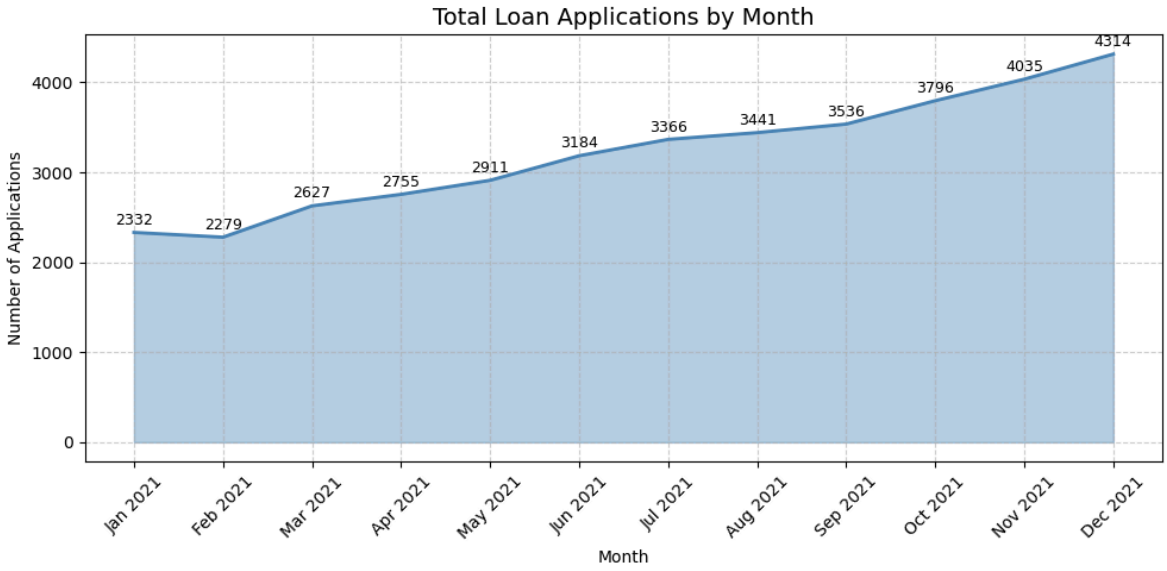
```
plt.ylabel('Received Amount (₹ Millions)')

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()
```



In [21]:

```
# Monthly aggregation
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='total_applications')
)

# ===== Plot =====

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

plt.fill_between(
    monthly_applications['month_name'],
    monthly_applications['total_applications'],
    color='steelblue',
    alpha=0.4
)

plt.plot(
    monthly_applications['month_name'],
    monthly_applications['total_applications'],
    color='steelblue',
    linewidth=2
)

# Value labels
for i, row in monthly_applications.iterrows():
    plt.text(
```

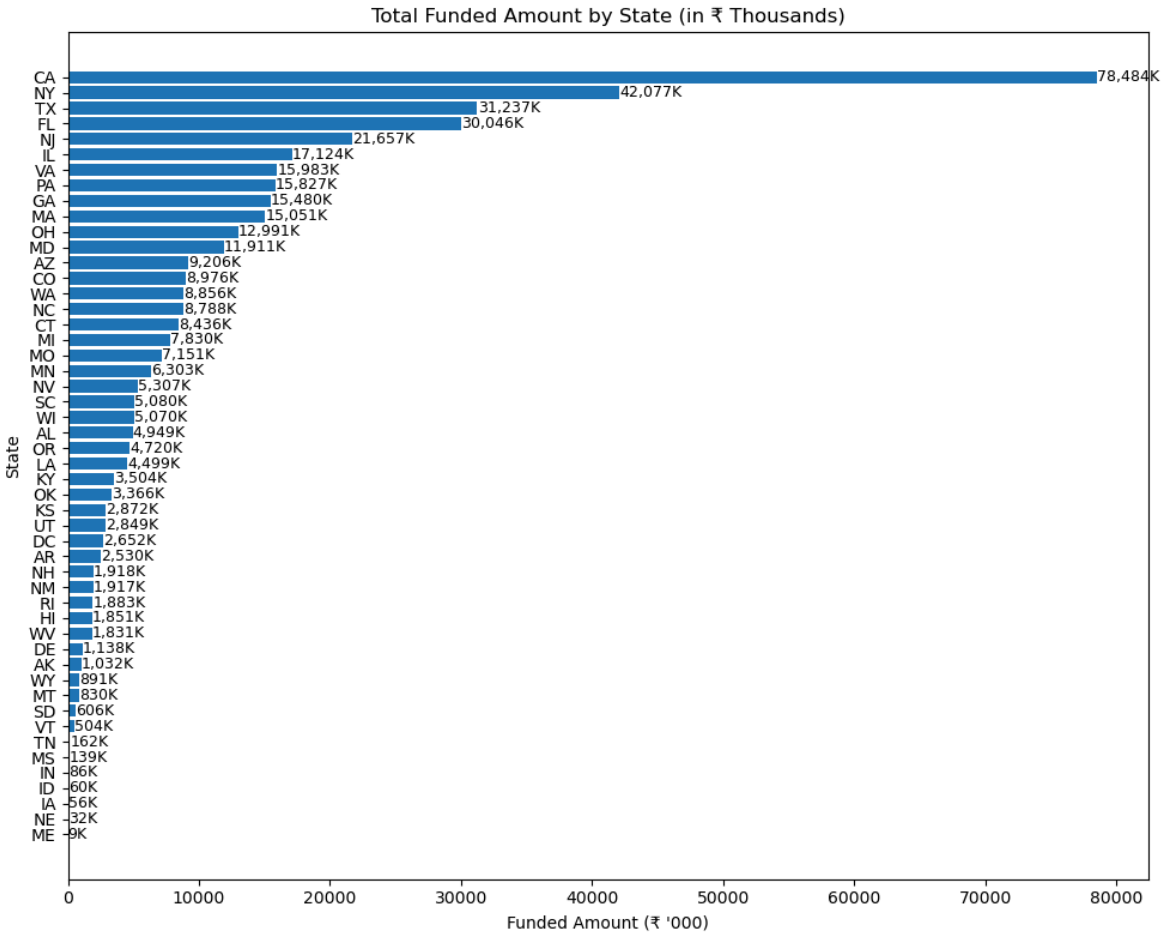
```
        i,
        row['total_applications'] + 50,
        f"{row['total_applications']}",
        ha='center',
        va='bottom',
        fontsize=9
    )

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()
```



In [22]:

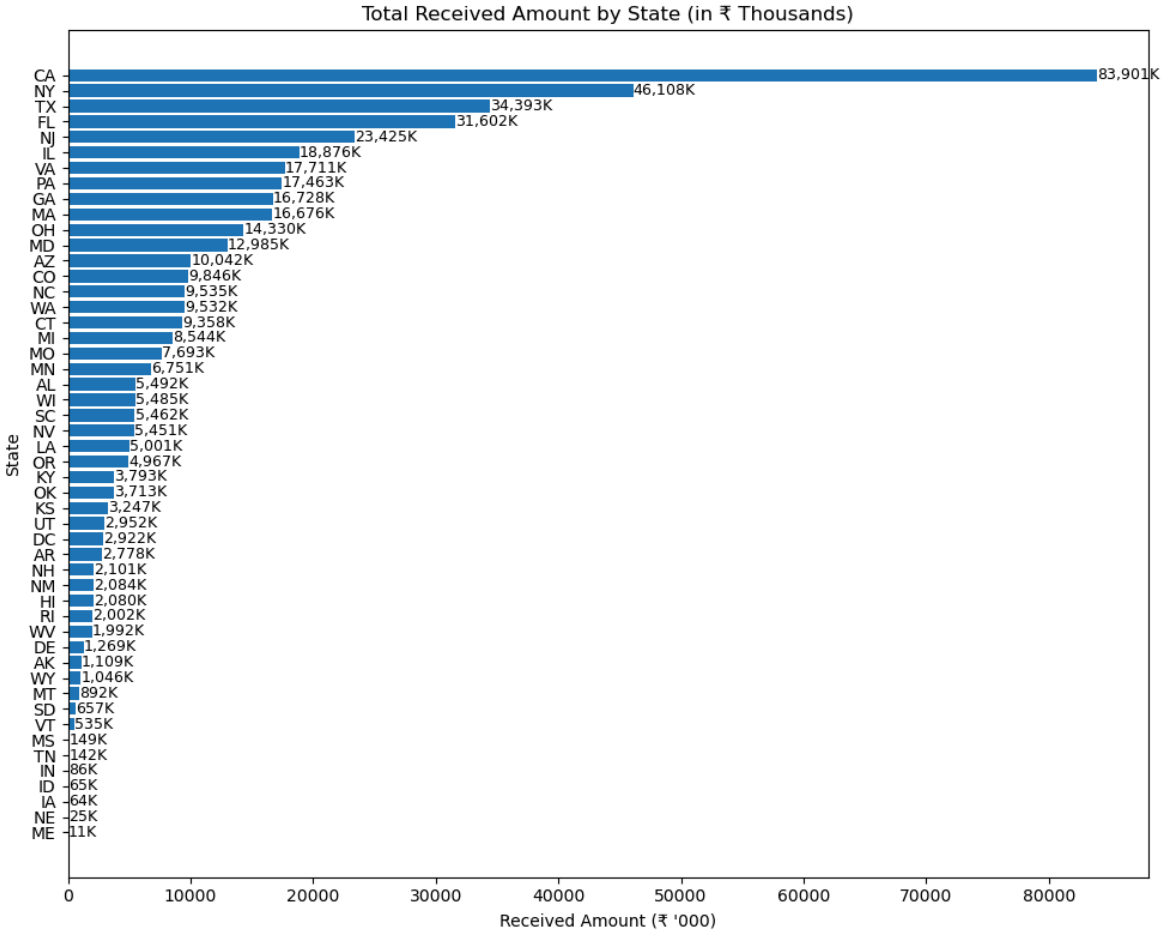
```
# ===== Aggregate funded amount by state =====

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

# Convert to thousands
state_funding_thousands = state_funding / 1000

# ===== Plot =====

plt.figure(figsize=(10, 8))

bars = plt.barh(
    state_funding_thousands.index,
    state_funding_thousands.values
)
```

```
# Add value labels on bars
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
    )

# Titles and labels
plt.title('Total Funded Amount by State (in ₹ Thousands)')
plt.xlabel('Funded Amount (₹ \'000)')
plt.ylabel('State')

plt.tight_layout()
plt.show()
```



In [23]:
```
# ===== Aggregate received amount by state =====

state_received = (
    df.groupby('address_state')['total_payment']
      .sum()
      .sort_values(ascending=True)
)

# Convert to thousands
state_received_thousands = state_received / 1000

# ===== Plot =====
```

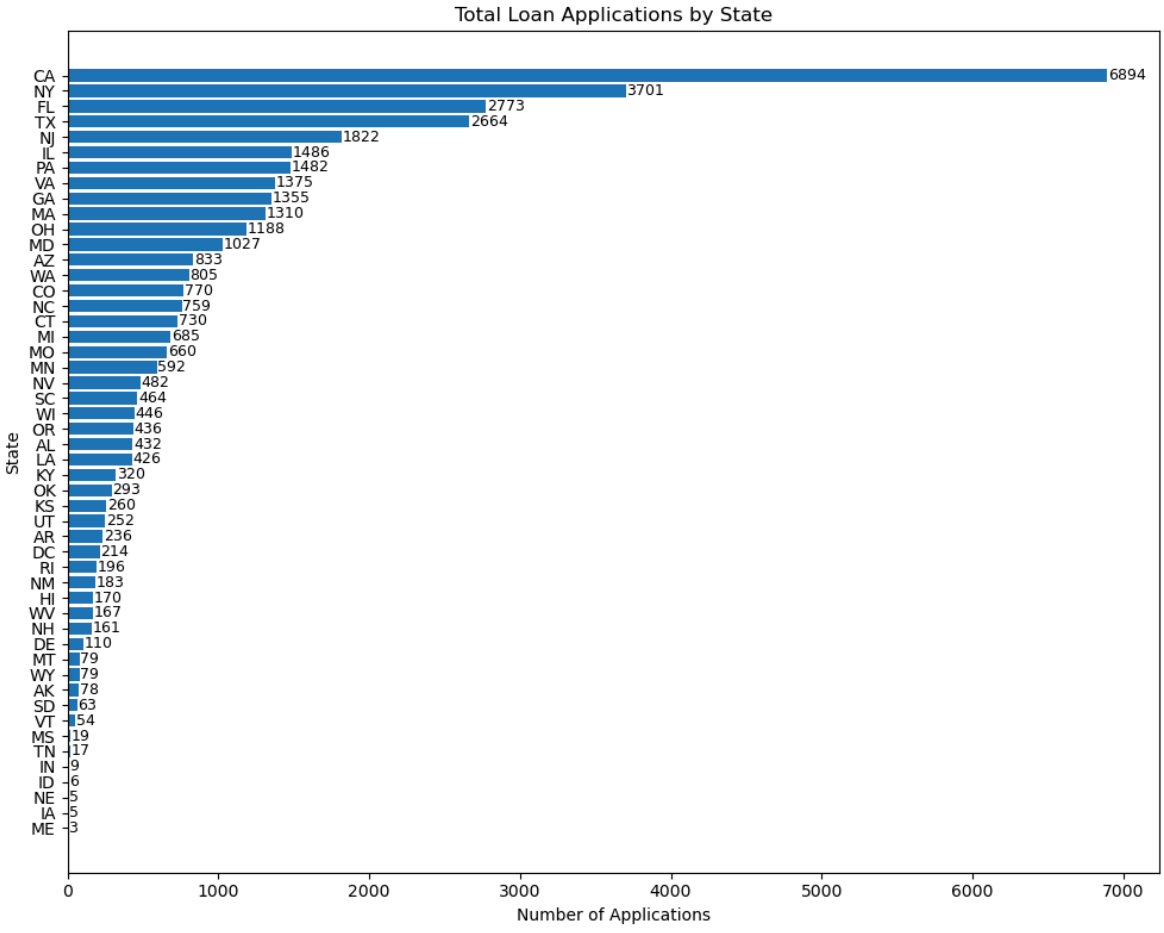
```
plt.figure(figsize=(10, 8))

bars = plt.barh(
    state_received_thousands.index,
    state_received_thousands.values
)

# Add value labels
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 Received Amount by State (in ₹ Thousands)')
plt.xlabel('Received Amount (₹ \'000)')
plt.ylabel('State')

plt.tight_layout()
plt.show()
```



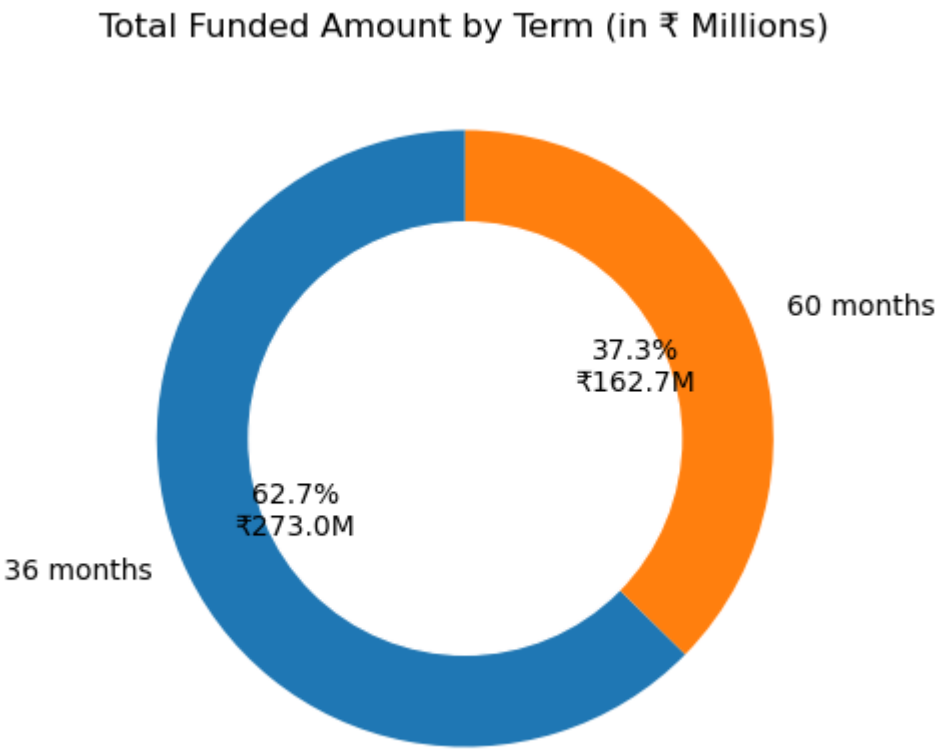
In [24]:

```
# ===== Count loan applications by state =====

state_applications = (
    df.groupby('address_state')['id']
      .count()
```

```
      .sort_values(ascending=True)
)

# ===== Plot =====

plt.figure(figsize=(10, 8))

bars = plt.barh(
    state_applications.index,
    state_applications.values
)

# Add value labels
for bar in bars:
    width = bar.get_width()
    plt.text(
        width + 5,
        bar.get_y() + bar.get_height() / 2,
        f'{int(width)}',
        va='center',
        fontsize=9
    )

plt.title('Total Loan Applications by State')
plt.xlabel('Number of Applications')
plt.ylabel('State')

plt.tight_layout()
plt.show()
```

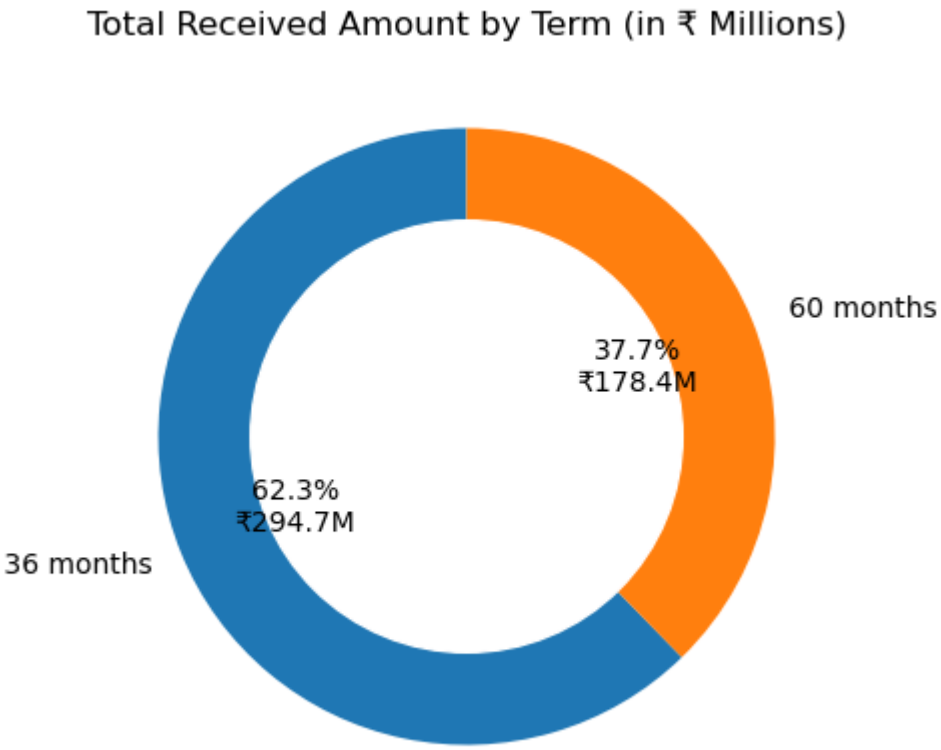


In [25]:
```
# ===== Aggregate funded amount by term =====

term_funding_millions = (
    df.groupby('term')['loan_amount']
      .sum()
      / 1_000_000
)

# ===== Plot donut chart =====

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}
)

# White circle to make donut
plt.gca().add_artist(plt.Circle((0, 0), 0.70, color='white'))

plt.title('Total Funded Amount by Term (in ₹ Millions)')
plt.show()
```



In [26]:
```
# ===== Aggregate received amount by term =====

term_received_millions = (
    df.groupby('term')['total_payment']
      .sum()
      / 1_000_000
)
```

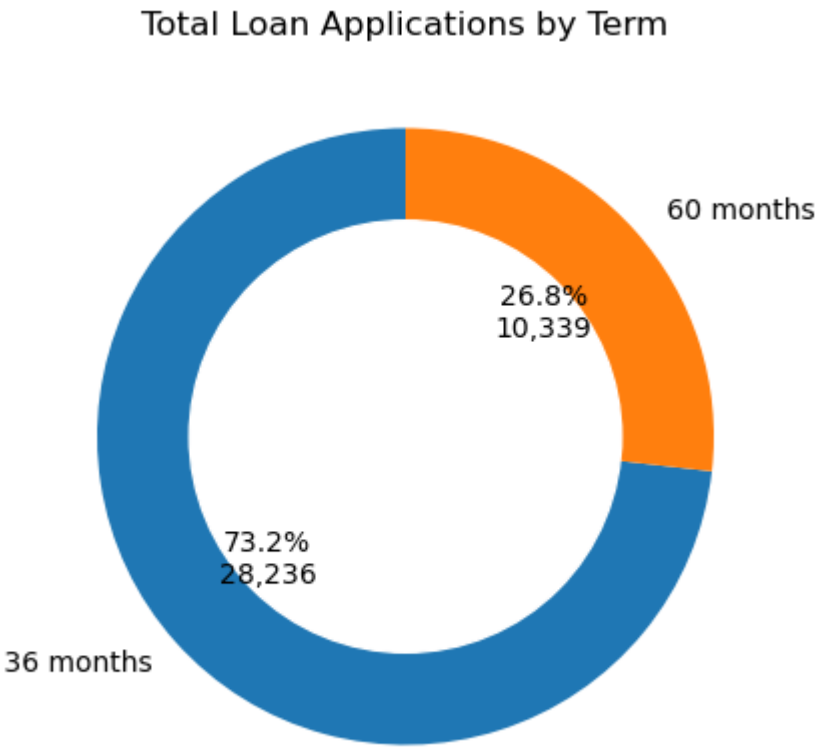
```
# ===== Donut chart =====

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

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': 0.4}
)

plt.gca().add_artist(plt.Circle((0, 0), 0.70, color='white'))

plt.title('Total Received Amount by Term (in ₹ Millions)')
plt.show()
```



In [29]:

```
# ===== Count applications by term =====

term_applications = (
    df.groupby('term')['id']
      .count()
)

total_apps = term_applications.sum()

# ===== Donut chart =====

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

plt.pie(
    term_applications,
    labels=term_applications.index,
```

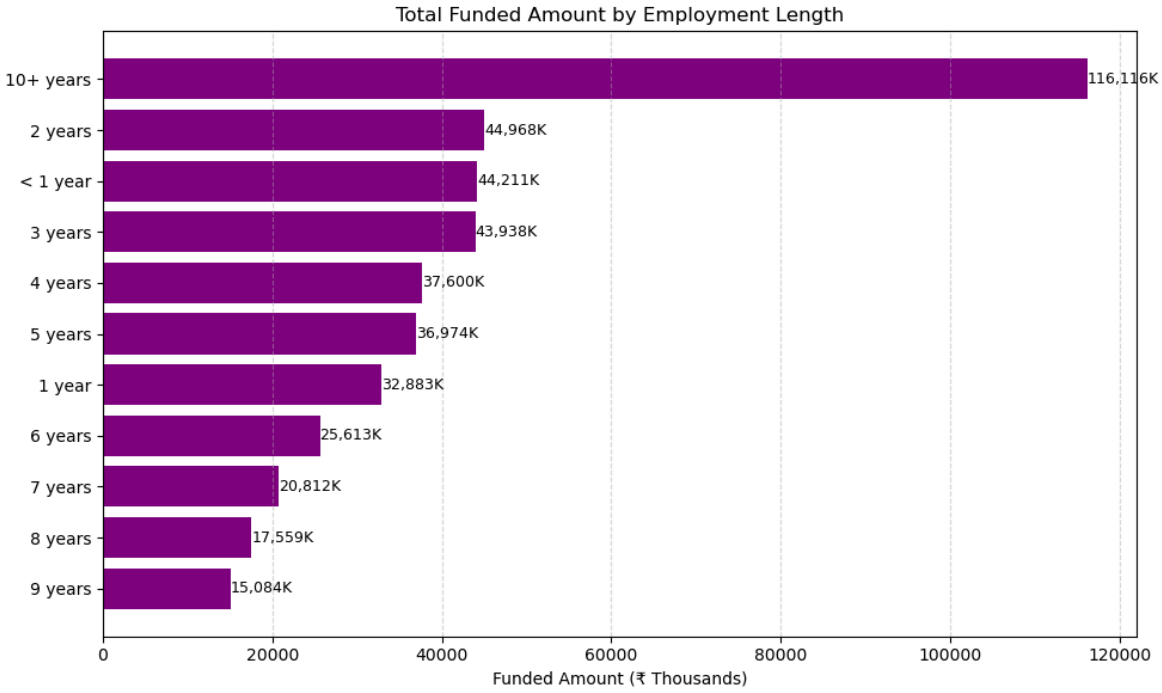
```
    autopct=lambda p: f"{p:.1f}%\n{int(p * total_apps / 100):,}",
    startangle=90,
    wedgeprops={'width': 0.4}
)

# White center for donut
plt.gca().add_artist(plt.Circle((0, 0), 0.70, color='white'))

plt.title('Total Loan Applications by Term')
plt.show()
```



In [30]:

```
# ===== Aggregate funded amount by employment length =====

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

# ===== Plot =====

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

bars = plt.barh(
    emp_funding_thousands.index,
    emp_funding_thousands.values,
    color='purple'
)

# Add value labels
for bar in bars:
    width = bar.get_width()
    plt.text(
```

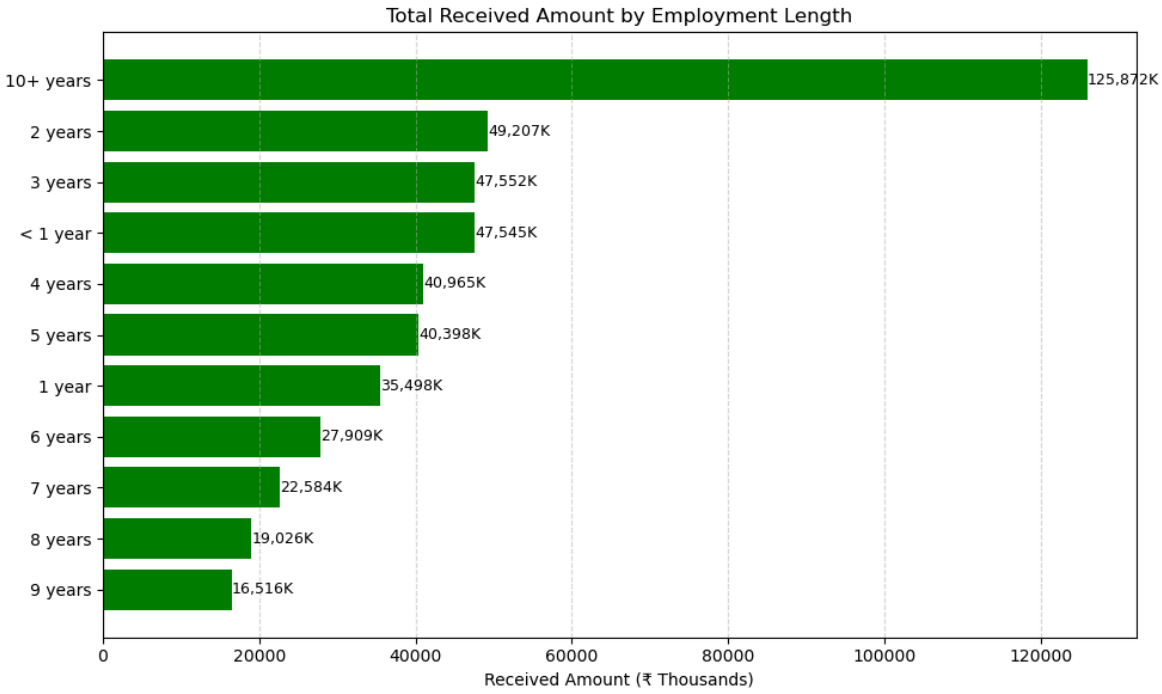
```
        width + 5,
        bar.get_y() + bar.get_height() / 2,
        f"{width:,.0f}K",
        va='center',
        fontsize=9
    )

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()
```



In [31]:

```
# ===== Aggregate received amount by employment length =====

emp_received_thousands = (
    df.groupby('emp_length')['total_payment']
      .sum()
      .sort_values()
      / 1000
)

# ===== Plot =====

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

bars = plt.barh(
    emp_received_thousands.index,
    emp_received_thousands.values,
    color='green'
)

# Add value labels
for bar in bars:
    width = bar.get_width()
    plt.text(
        width + 5,
```

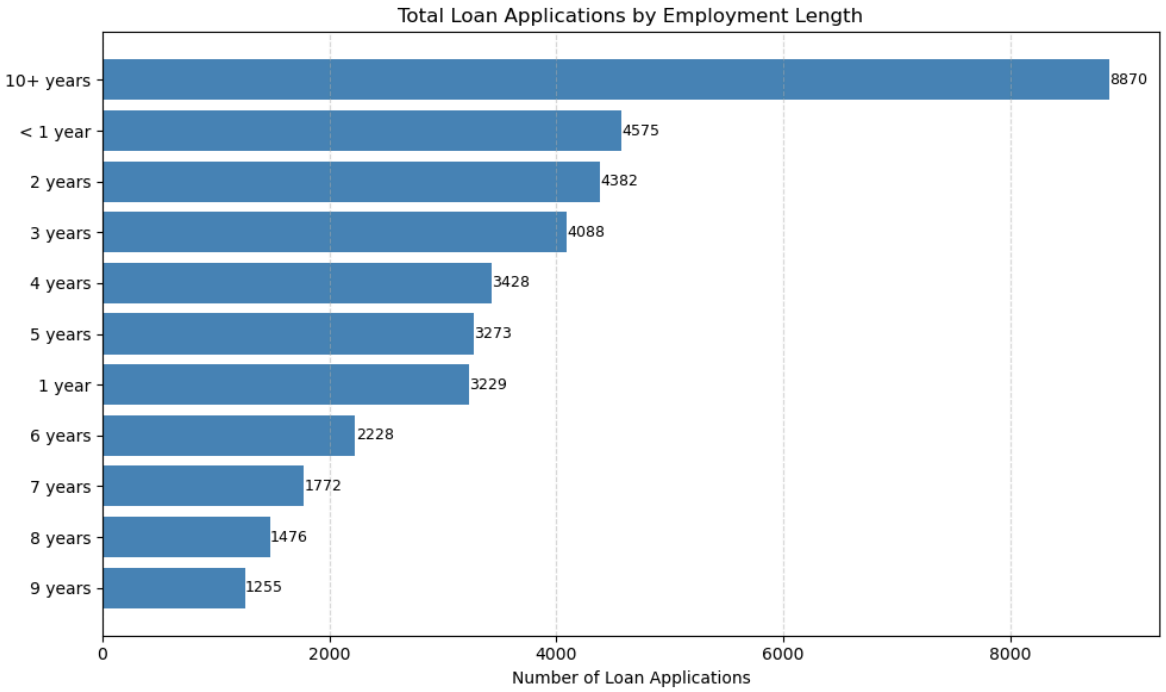
```
        bar.get_y() + bar.get_height() / 2,
        f"{width:,.0f}K",
        va='center',
        fontsize=9
    )

plt.xlabel("Received Amount (₹ Thousands)")
plt.title("Total Received Amount by Employment Length")
plt.grid(axis='x', linestyle='--', alpha=0.5)

plt.tight_layout()
plt.show()
```



In [32]:

```
# ===== Count loan applications =====

emp_applications = (
    df.groupby('emp_length')['id']
      .count()
      .sort_values()
)

# ===== Plot =====

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

bars = plt.barh(
    emp_applications.index,
    emp_applications.values,
    color='steelblue'
)

# Add value labels
for bar in bars:
    width = bar.get_width()
    plt.text(
        width + 5,
        bar.get_y() + bar.get_height() / 2,
        f"{int(width)}",
```

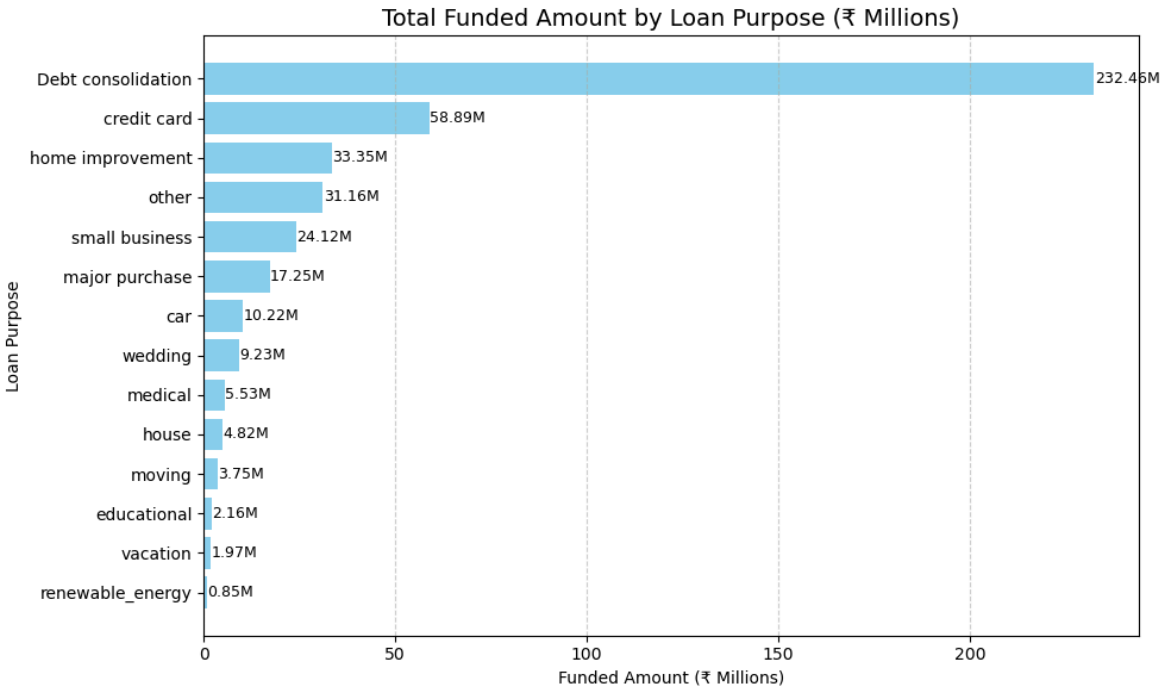
```
        va='center',
        fontsize=9
    )

plt.xlabel("Number of Loan Applications")
plt.title("Total Loan Applications by Employment Length")
plt.grid(axis='x', linestyle='--', alpha=0.5)

plt.tight_layout()
plt.show()
```



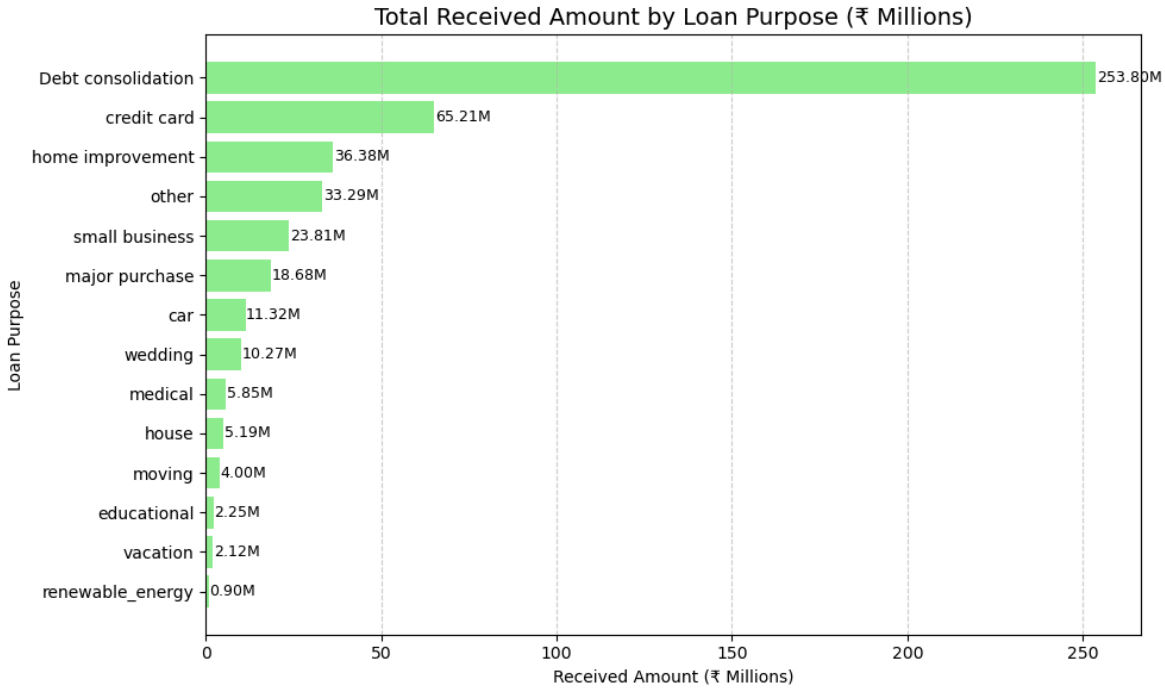
In [33]:

```
purpose_funding_millions = (
    df.groupby('purpose')['loan_amount']
      .sum()
      .sort_values()
      / 1_000_000
)

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 + 0.1,
        bar.get_y() + bar.get_height() / 2,
        f"{width:.2f}M",
        va='center',
        fontsize=9
    )

plt.title('Total Funded Amount by Loan Purpose (₹ 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()
```

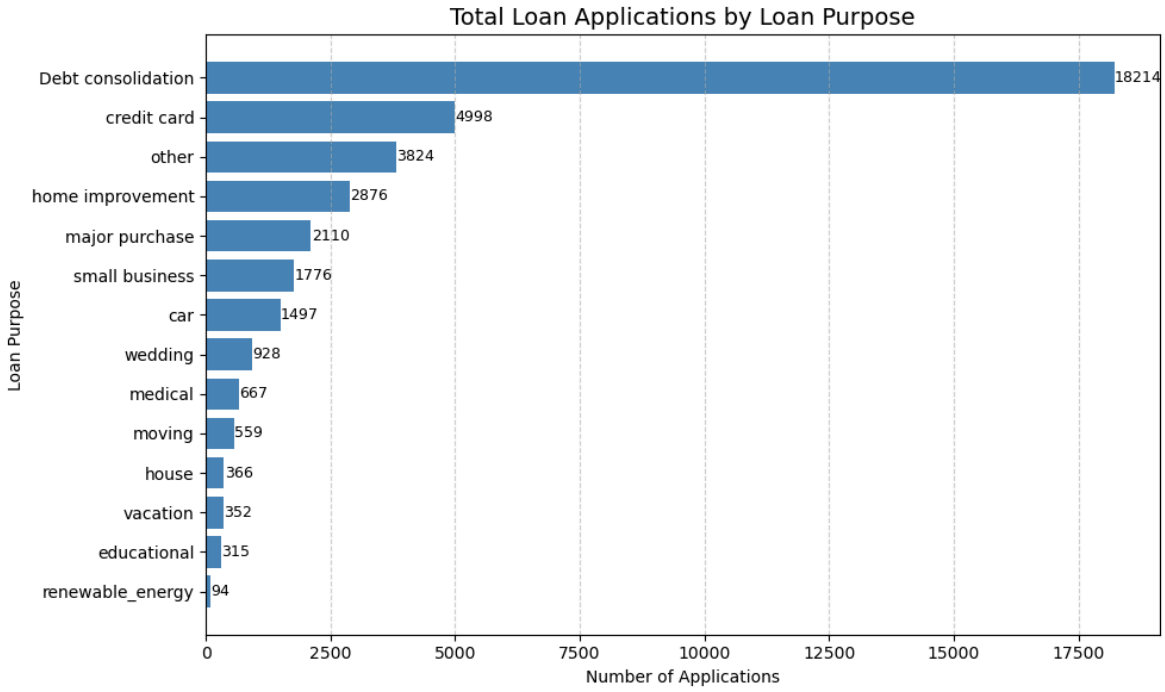


In [34]:
```
purpose_received_millions = (
    df.groupby('purpose')['total_payment']
      .sum()
      .sort_values()
      / 1_000_000
)

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

bars = plt.barh(
    purpose_received_millions.index,
    purpose_received_millions.values,
    color='lightgreen'
)

for bar in bars:
    width = bar.get_width()
    plt.text(
        width + 0.1,
        bar.get_y() + bar.get_height() / 2,
        f"{width:.2f}M",
        va='center',
        fontsize=9
    )

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



In [35]:

```
purpose_applications = (
    df.groupby('purpose')['id']
      .count()
      .sort_values()
)

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

bars = plt.barh(
    purpose_applications.index,
    purpose_applications.values,
    color='steelblue'
)

for bar in bars:
    width = bar.get_width()
    plt.text(
        width + 5,
        bar.get_y() + bar.get_height() / 2,
        f"{int(width)}",
        va='center',
        fontsize=9
    )

plt.title('Total Loan Applications by Loan Purpose', fontsize=14)
plt.xlabel('Number of Applications')
plt.ylabel('Loan Purpose')
plt.grid(axis='x', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```



In [36]:
```python
# ===== Aggregate funded amount by home ownership =====

home_funding = (
    df.groupby('home_ownership')['loan_amount']
      .sum()
      .reset_index()
)

# Convert to millions
home_funding['loan_amount_millions'] = home_funding['loan_amount'] / 1_000_000

# ===== 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()
```

Total Funded Amount by Home Ownership (₹ Millions)

In [37]:

```
# ===== Aggregate received amount =====

home_received = (
    df.groupby('home_ownership')['total_payment']
      .sum()
      .reset_index()
)

# Convert to millions
home_received['received_millions'] = home_received['total_payment'] / 1_000_000

# ===== Treemap =====

fig = px.treemap(
    home_received,
    path=['home_ownership'],
    values='received_millions',
    color='received_millions',
    color_continuous_scale='Greens',
    title='Total Received Amount by Home Ownership (₹ Millions)'
)

fig.show()
```

Total Received Amount by Home Ownership (₹ Millions)

In [38]:
```
# ===== Count applications =====

home_applications = (
    df.groupby('home_ownership')['id']
      .count()
      .reset_index(name='total_applications')
)

# ===== Treemap =====

fig = px.treemap(
    home_applications,
    path=['home_ownership'],
    values='total_applications',
    color='total_applications',
    color_continuous_scale='Blues',
    title='Total Loan Applications by Home Ownership'
)

fig.show()
```

Total Loan Applications by Home Ownership