BANK LOAN ANALYSIS - FINANCE DOMAIN

In [2]:
```python
#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]:
```python
#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_ownership	iss
0	1077430	GA	INDIVIDUAL	< 1 year	Ryder	C	RENT	2
1	1072053	CA	INDIVIDUAL	9 years	MKC Accounting	E	RENT	2
2	1069243	CA	INDIVIDUAL	4 years	Chemat Technology Inc	C	RENT	2
3	1041756	TX	INDIVIDUAL	< 1 year	barnes distribution	B	MORTGAGE	2
4	1068350	IL	INDIVIDUAL	10+ years	J&J Steel Inc	A	MORTGAGE	2
...	...	...	...	...	...	...	...	
38571	803452	NJ	INDIVIDUAL	< 1 year	Joseph M Sanzari Company	C	MORTGAGE	2
38572	970377	NY	INDIVIDUAL	8 years	Swat Fame	C	RENT	2
38573	875376	CA	INDIVIDUAL	5 years	Anaheim Regional Medical Center	D	RENT	2
38574	972997	NY	INDIVIDUAL	5 years	Brooklyn Radiology	D	RENT	2
38575	682952	NY	INDIVIDUAL	4 years	Allen Edmonds	F	RENT	2

38576 rows × 24 columns

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

df.head()
```

Out[4]:

	id	address_state	application_type	emp_length	emp_title	grade	home_ownership	issue_da
0	1077430	GA	INDIVIDUAL	< 1 year	Ryder	C	RENT	2021-0
1	1072053	CA	INDIVIDUAL	9 years	MKC Accounting	E	RENT	2021-0
2	1069243	CA	INDIVIDUAL	4 years	Chemat Technology Inc	C	RENT	2021-0
3	1041756	TX	INDIVIDUAL	< 1 year	barnes distribution	B	MORTGAGE	2021-0
4	1068350	IL	INDIVIDUAL	10+ years	J&J Steel Inc	A	MORTGAGE	2021-0

5 rows × 24 columns

In [5]: *#last five records*

```python
df.tail()
```

Out[5]:

	id	address_state	application_type	emp_length	emp_title	grade	home_ownership	issue_
38571	803452	NJ	INDIVIDUAL	< 1 year	Joseph M Sanzari Company	C	MORTGAGE	202
38572	970377	NY	INDIVIDUAL	8 years	Swat Fame	C	RENT	202
38573	875376	CA	INDIVIDUAL	5 years	Anaheim Regional Medical Center	D	RENT	202
38574	972997	NY	INDIVIDUAL	5 years	Brooklyn Radiology	D	RENT	202
38575	682952	NY	INDIVIDUAL	4 years	Allen Edmonds	F	RENT	202

5 rows × 24 columns

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

```python
df.shape
```

Out[6]: (38576, 24)

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

```python
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_payment_date
count	3.857600e+04	38576	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	2021-07-26 20:42:20.605557760
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
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
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
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
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
std	2.113246e+05	NaN	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 == latest_montl
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 == latest_montl
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_millions))
```

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 == latest_montl
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%

```python
In [16]:  #average dti

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

Avg DTI: 13.33%

```python
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_amount_in_mil
          print("Good Loan Received Amount (in Millions): ${:.2f}M".format(good_loan_received_amount_in
          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 Received Amount (in Millions): $435.79M
Percentage of Good Loan Applications: 86.18%

```python
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_amount_in_milli
          print("Bad Loan Received Amount (in Millions): ${:.2f}M".format(bad_loan_received_amount_in_m
          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%

```python
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()
```
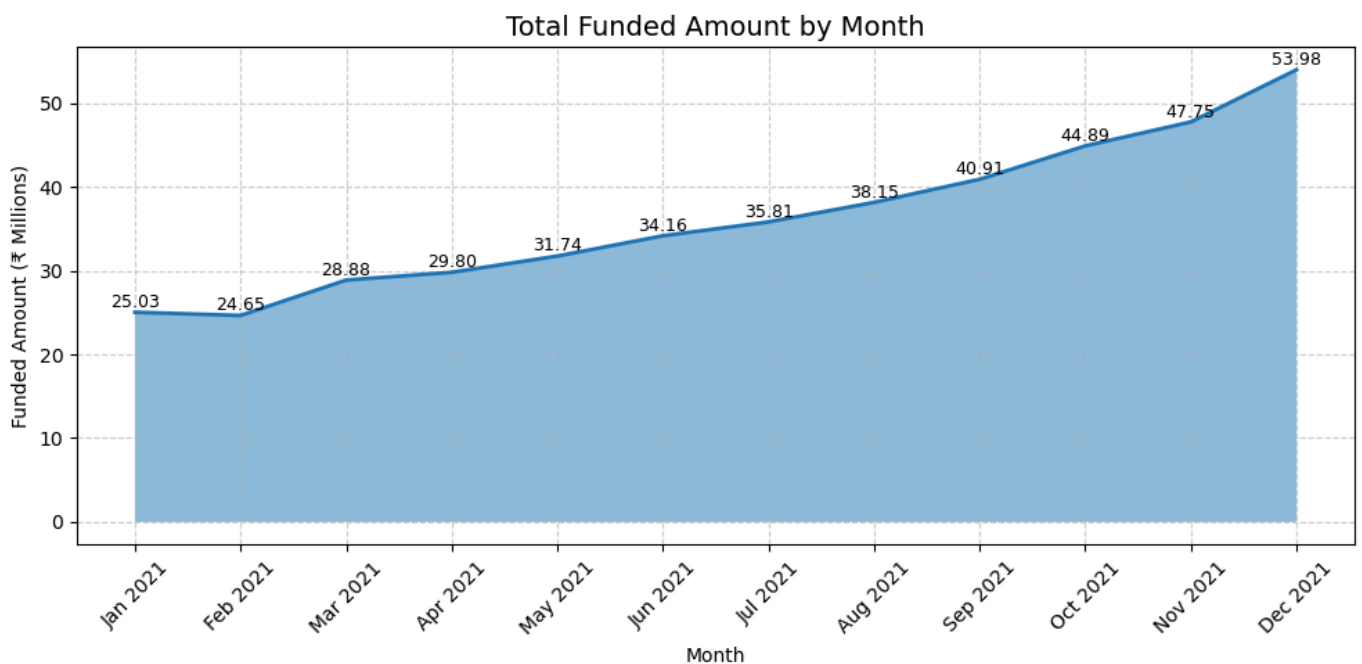

Total Funded Amount by Month

```
In [20]:  # ===== 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()
```
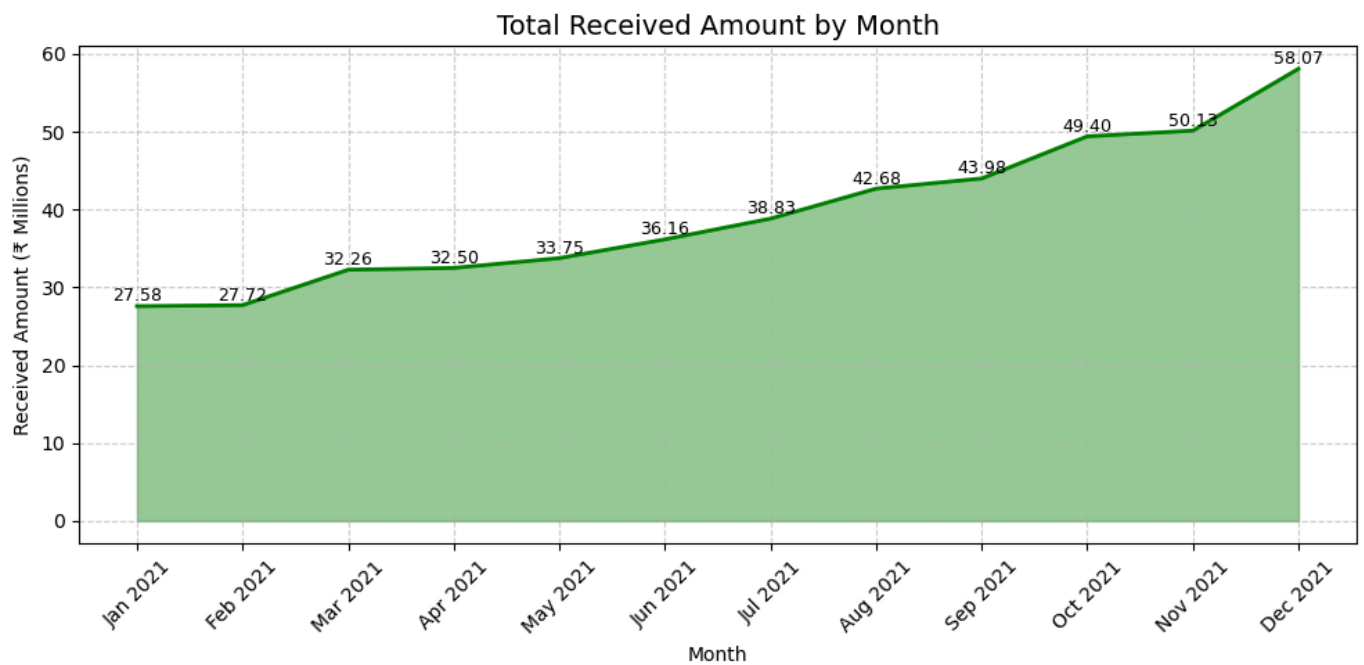
Total Received Amount by Month

```python
# 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)),
```
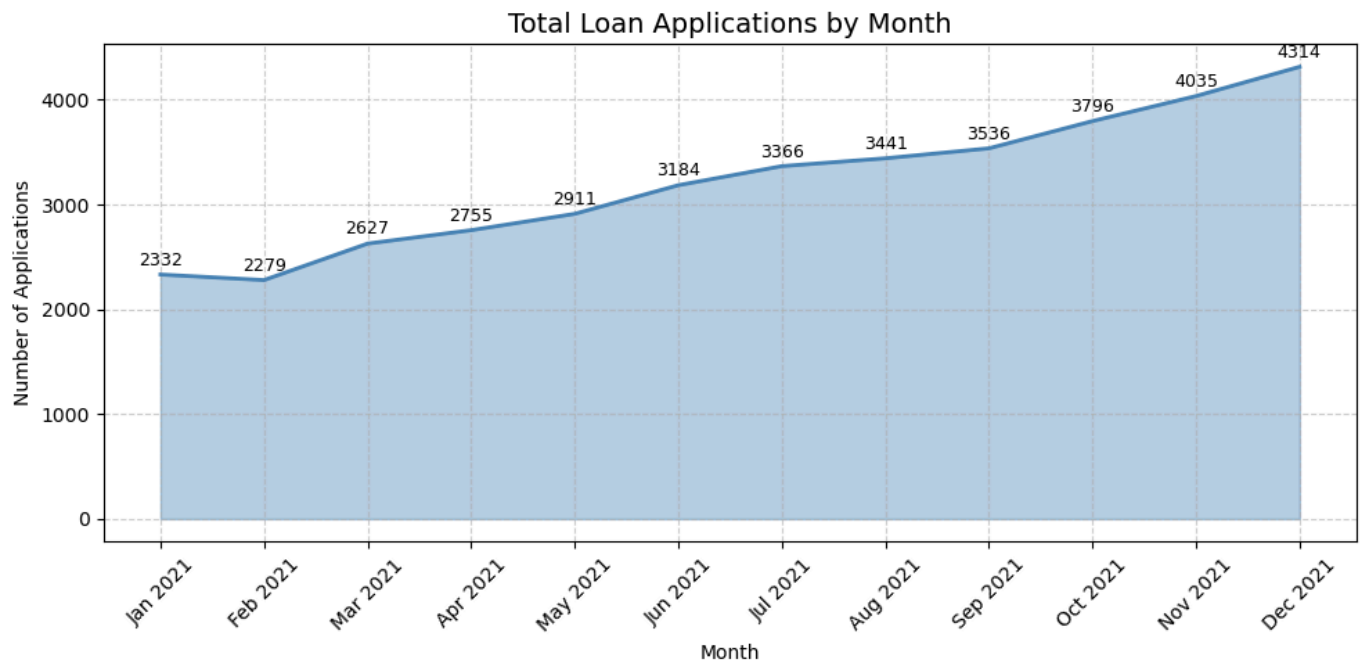
```
        labels=monthly_applications['month_name'],
        rotation=45
    )

plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```


Total Loan Applications by Month

```python
# ===== 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()
```


Total Funded Amount by State (in ₹ Thousands)

State	Funded Amount (₹ '000)
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
MS	139K
IN	86K
ID	60K
IA	56K
NE	32K
ME	9K

In [23]:
```python
# ===== 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
```
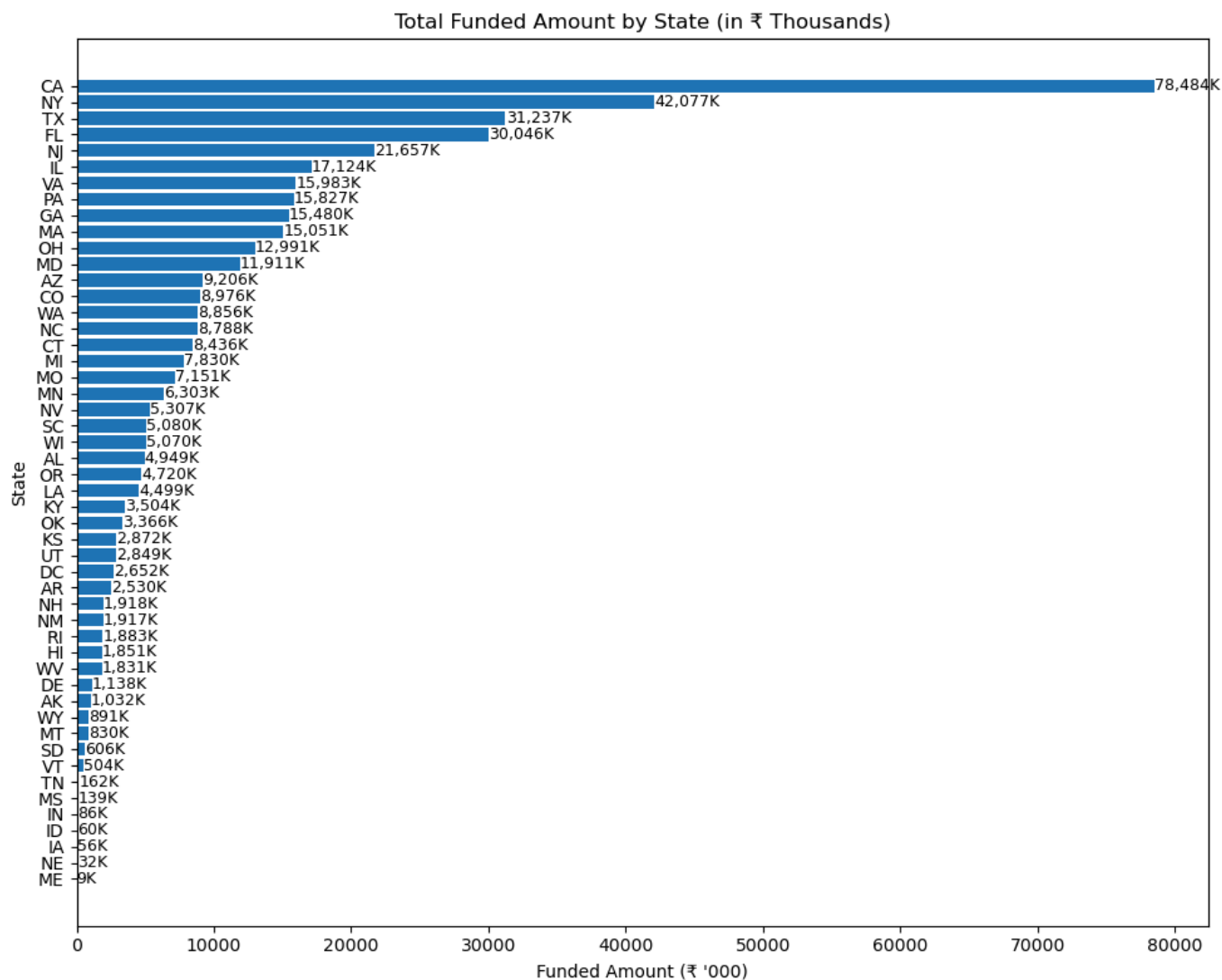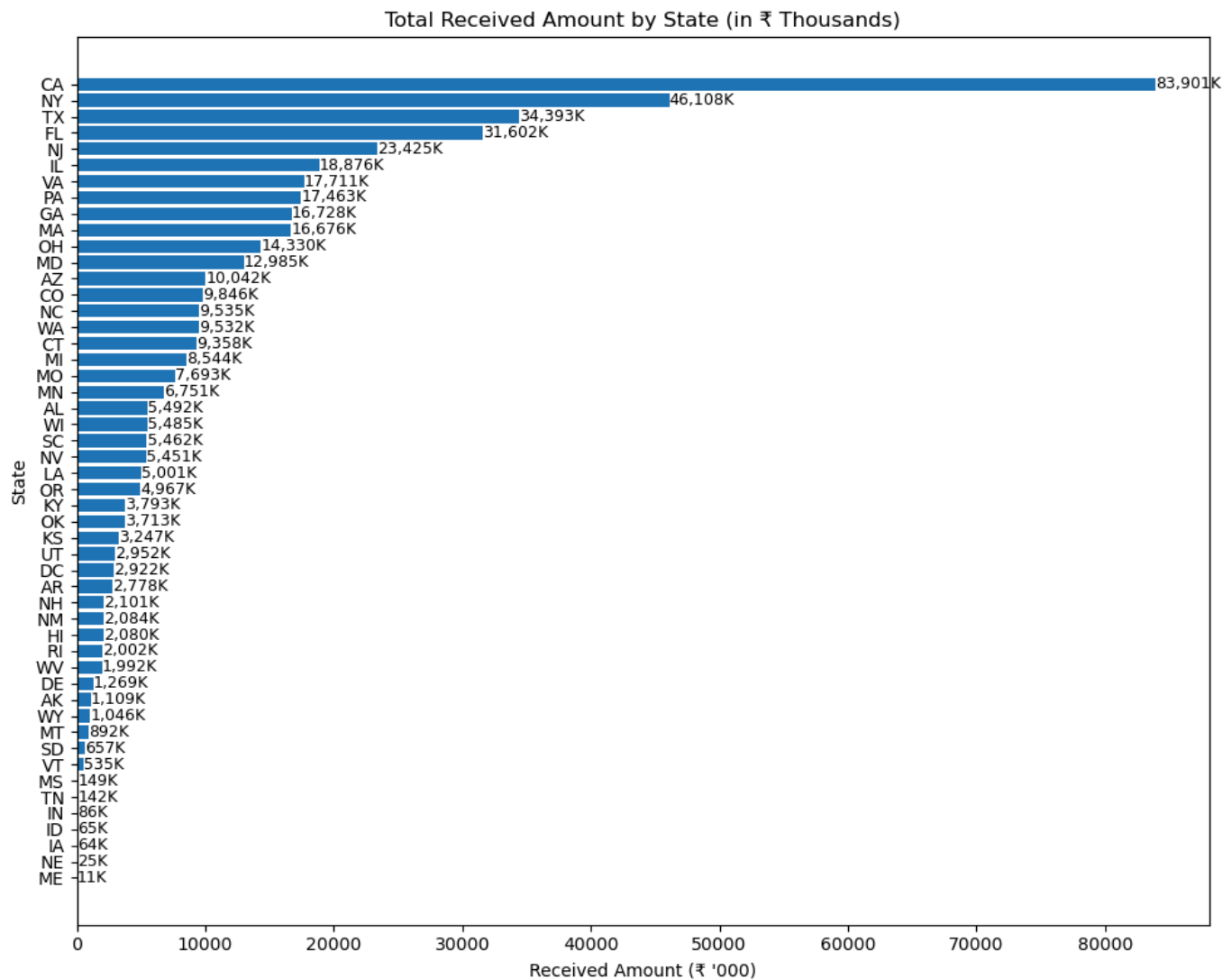
```
    )

plt.title('Total Received Amount by State (in ₹ Thousands)')
plt.xlabel('Received Amount (₹ \'000)')
plt.ylabel('State')

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


Total Received Amount by State (in ₹ Thousands)

```
# ===== 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,
```
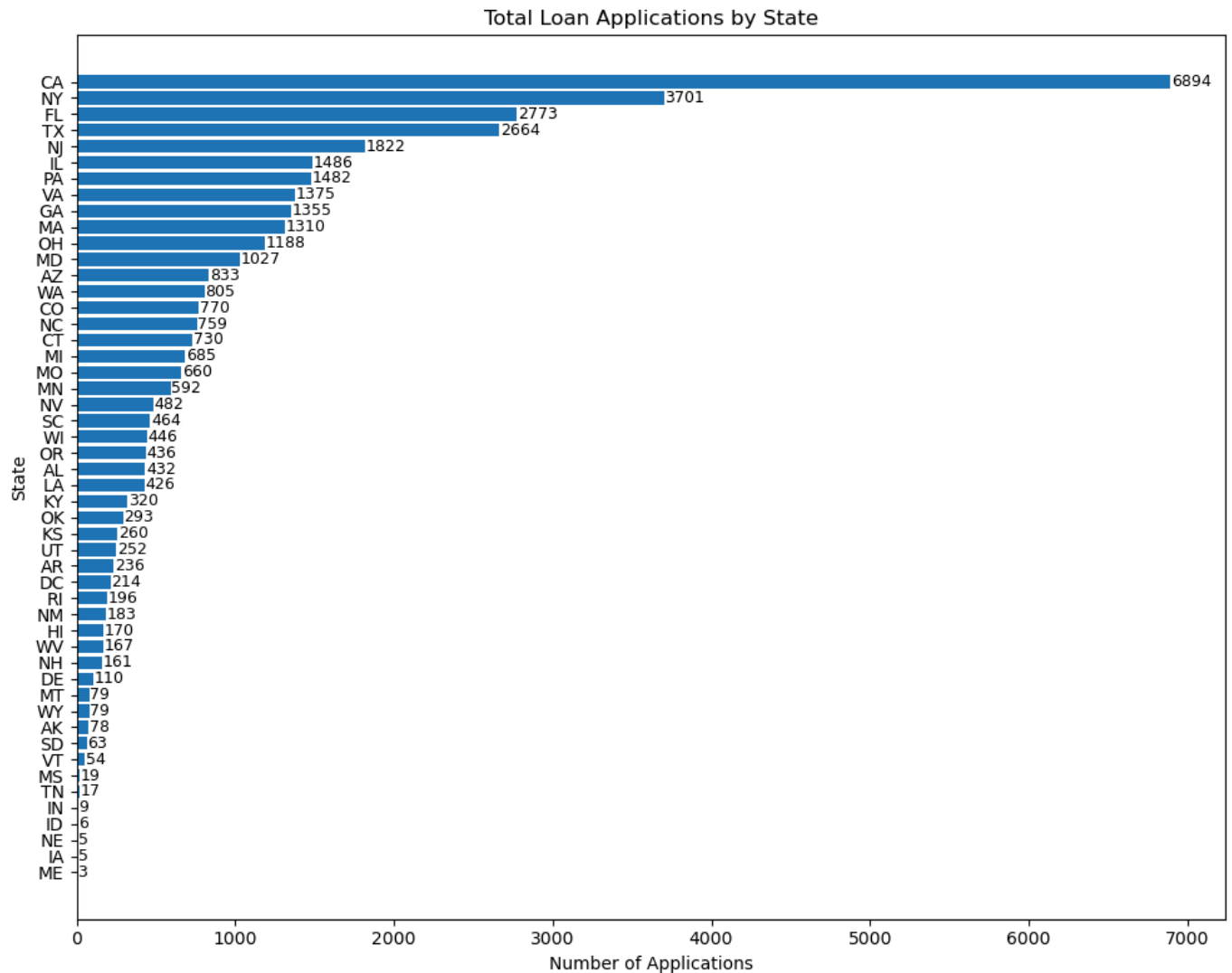
```
            f'{int(width)}',
            va='center',
            fontsize=9
        )

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

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


Total Loan Applications by State

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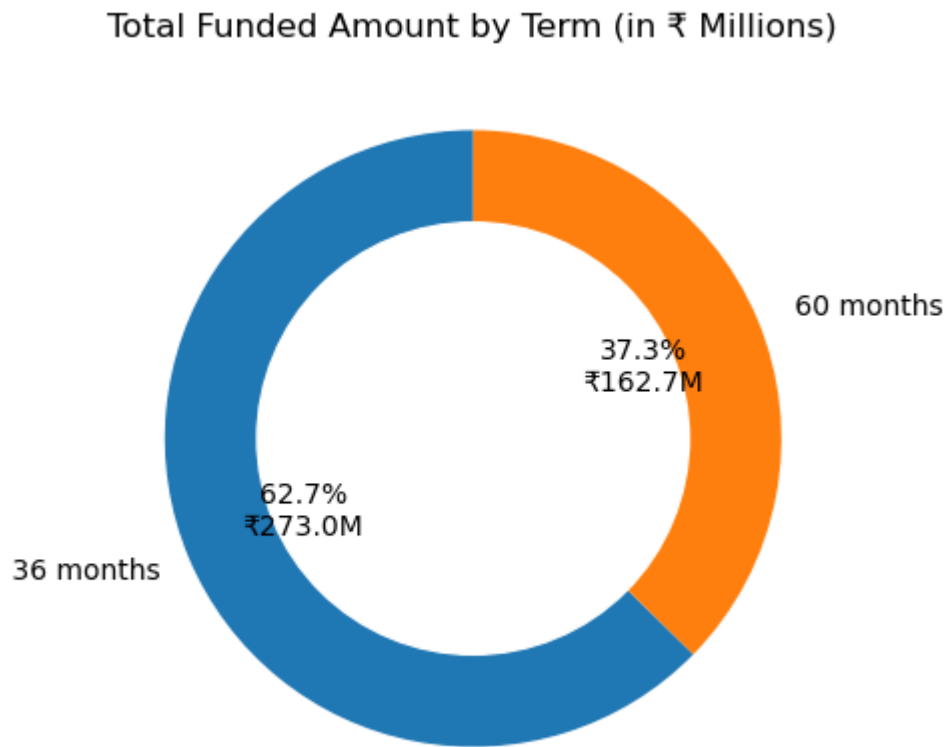
```
plt.gca().add_artist(plt.Circle((0, 0), 0.70, color='white'))

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

Total Funded Amount by Term (in ₹ Millions)

60 months

37.3%
₹162.7M

62.7%
₹273.0M

36 months

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()
```
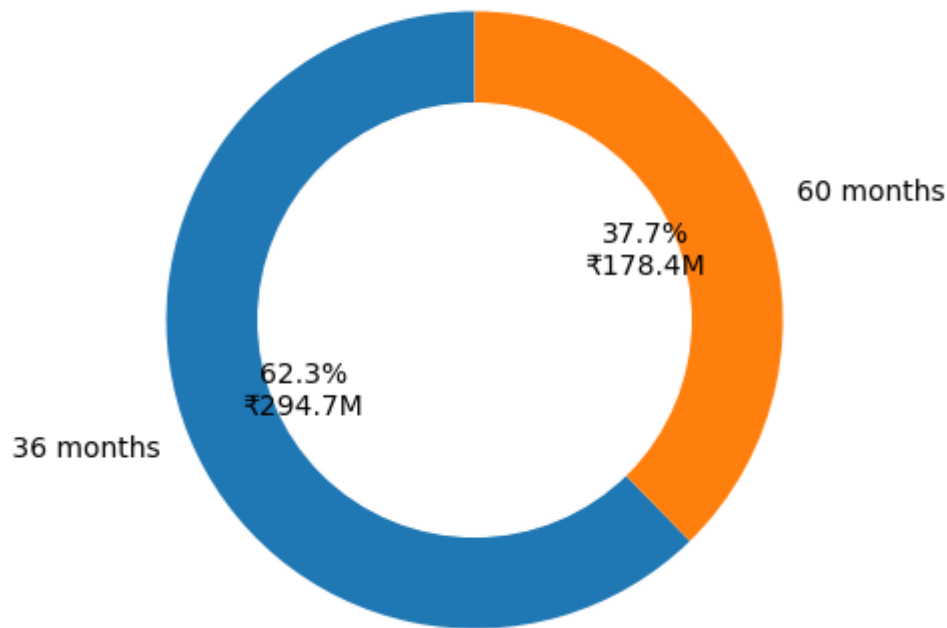
Total Received Amount by Term (in ₹ Millions)

60 months

37.7%
₹178.4M

62.3%
₹294.7M

36 months

In [29]:
```python
# ===== 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()
```
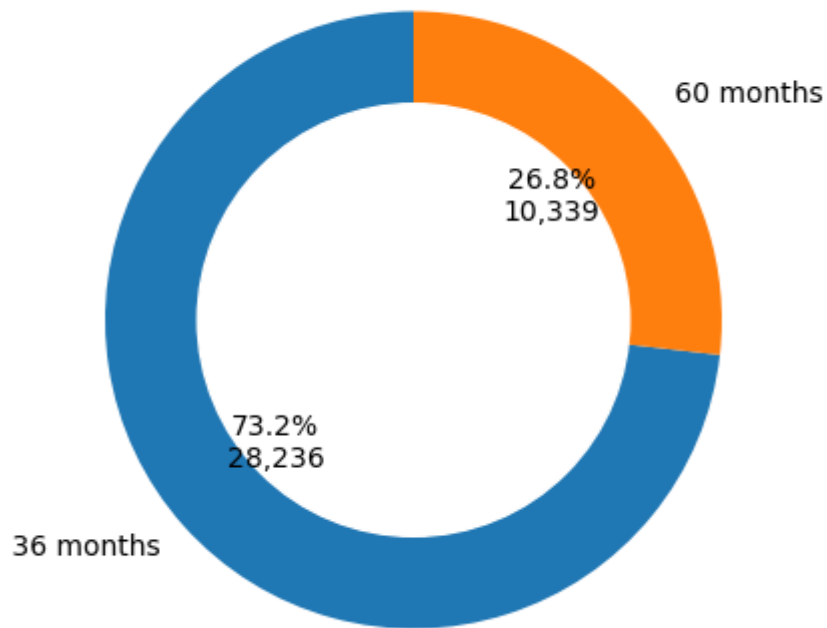
Total Loan Applications by Term

60 months

26.8%
10,339

73.2%
28,236

36 months

In [30]:
```python
# ===== 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()
```

Total Funded Amount by Employment Length
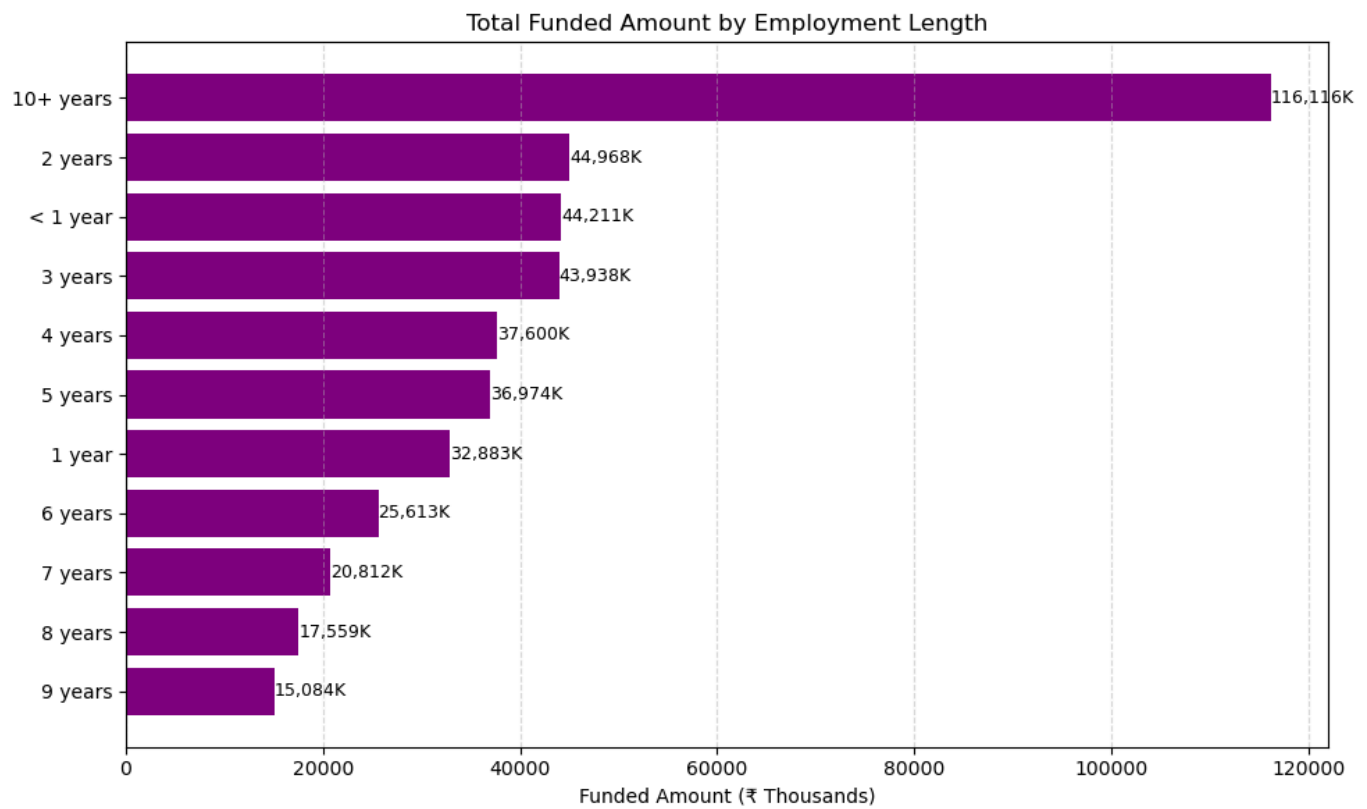

```
In [31]:  # ===== Aggregate received amount by employment length =====

          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()
```

Total Received Amount by Employment Length

Employment Length	Amount
10+ years	125,872K
2 years	49,207K
3 years	47,552K
< 1 year	47,545K
4 years	40,965K
5 years	40,398K
1 year	35,498K
6 years	27,909K
7 years	22,584K
8 years	19,026K
9 years	16,516K

Received Amount (₹ Thousands)
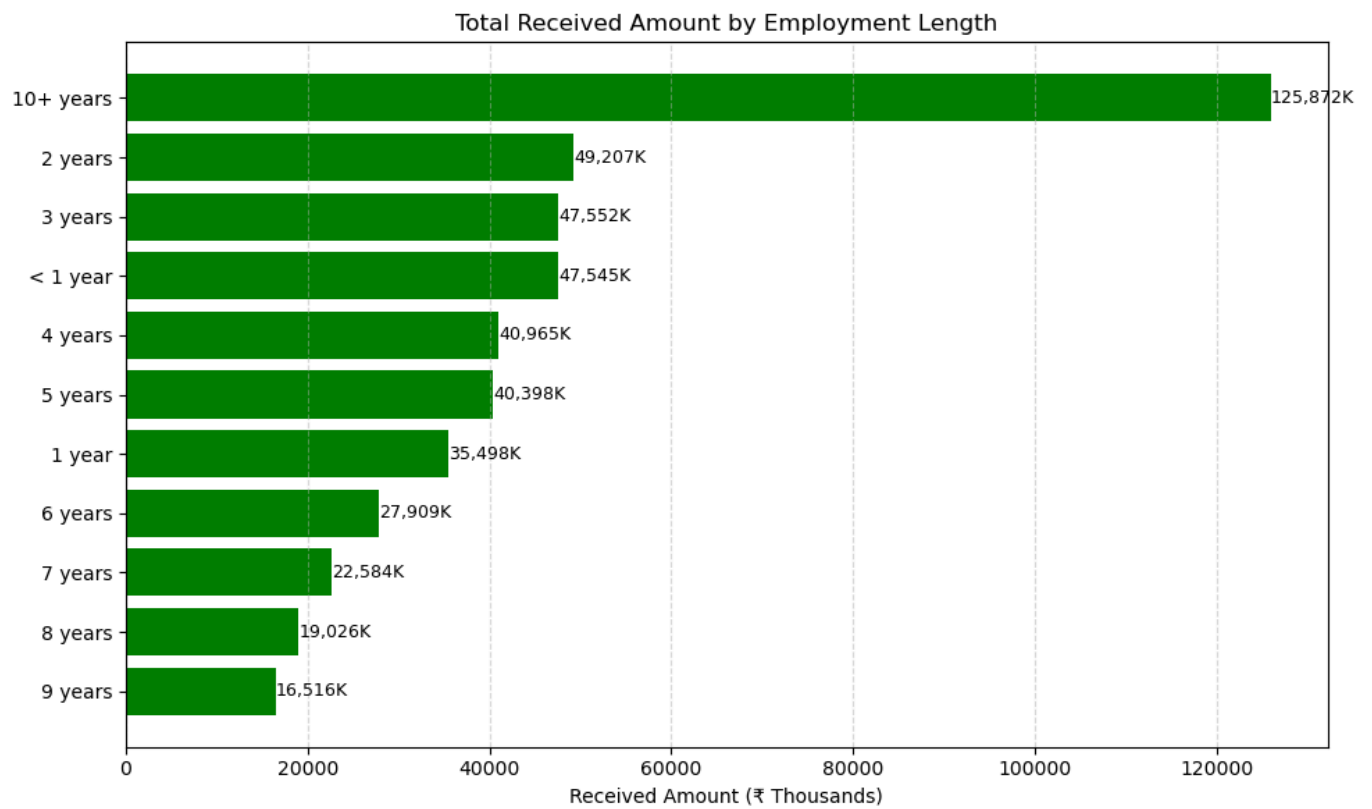
In [32]:
```python
# ===== Count loan applications =====

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()
```

Total Loan Applications by Employment Length

Employment Length	Number of Loan Applications
10+ years	8870
< 1 year	4575
2 years	4382
3 years	4088
4 years	3428
5 years	3273
1 year	3229
6 years	2228
7 years	1772
8 years	1476
9 years	1255
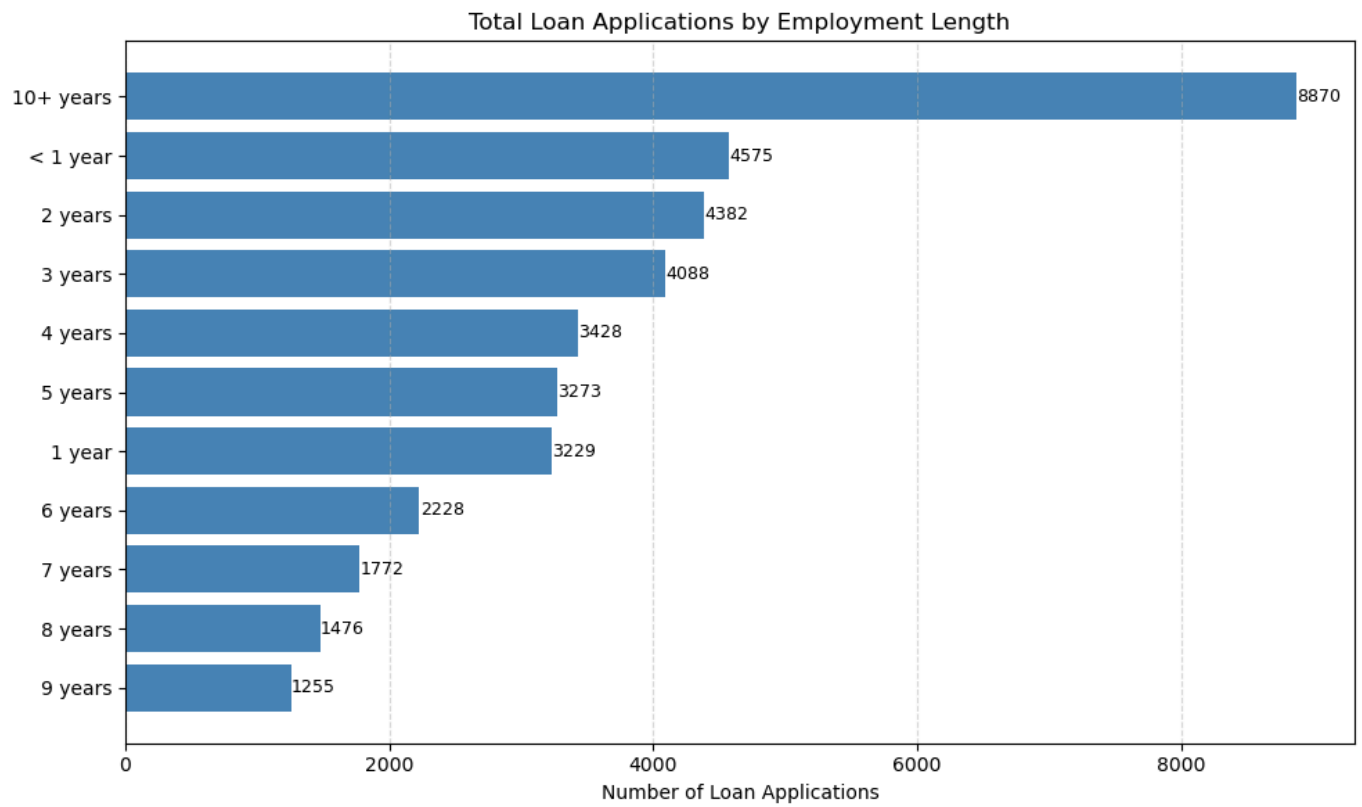
In [33]:
```python
purpose_funding_millions = (
    df.groupby('purpose')['loan_amount']
    .sum()
    .sort_values()
    / 1_000_000
)

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()
```
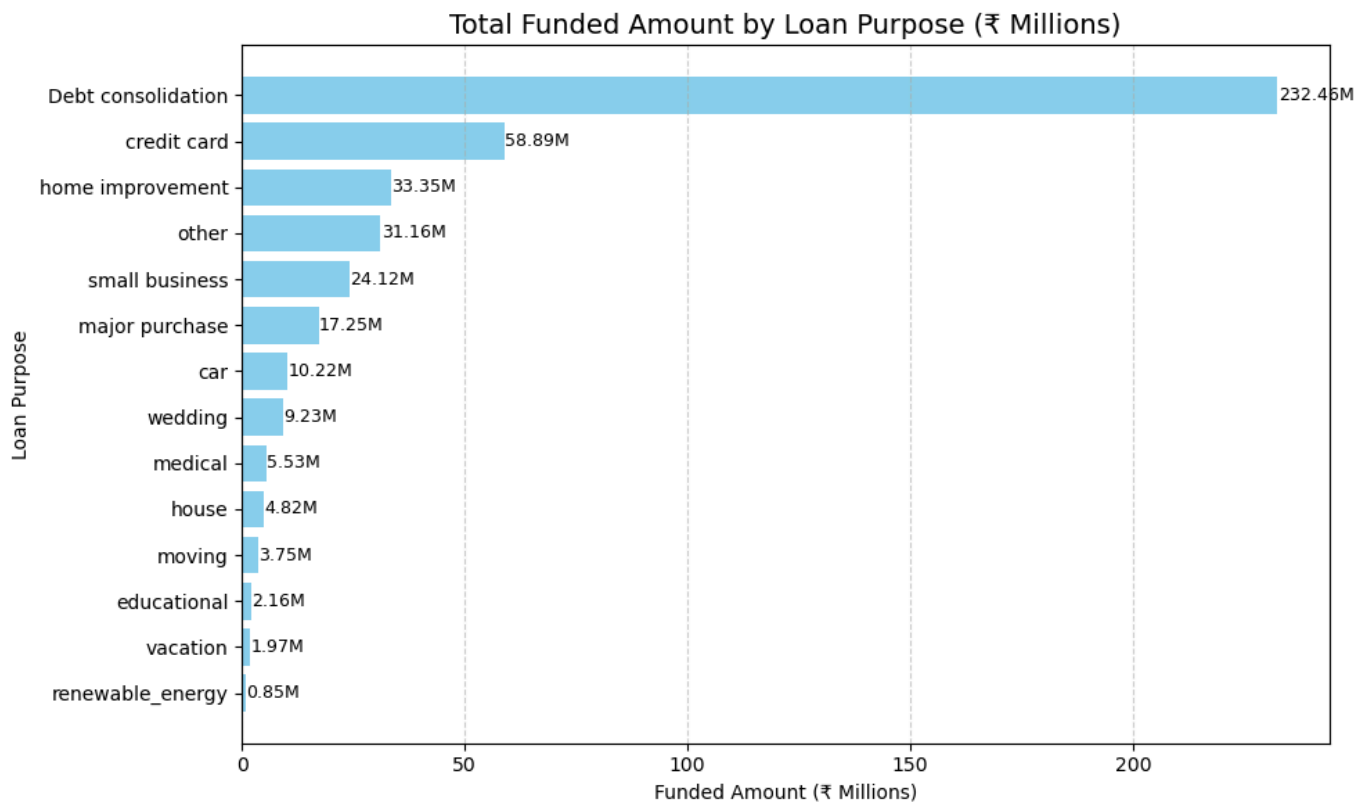
Total Funded Amount by Loan Purpose (₹ Millions)

Total Funded Amount by Loan Purpose (₹ Millions)

Loan Purpose	Funded Amount (₹ Millions)
Debt consolidation	232.46M
credit card	58.89M
home improvement	33.35M
other	31.16M
small business	24.12M
major purchase	17.25M
car	10.22M
wedding	9.23M
medical	5.53M
house	4.82M
moving	3.75M
educational	2.16M
vacation	1.97M
renewable_energy	0.85M

In [34]:
```python
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()
```
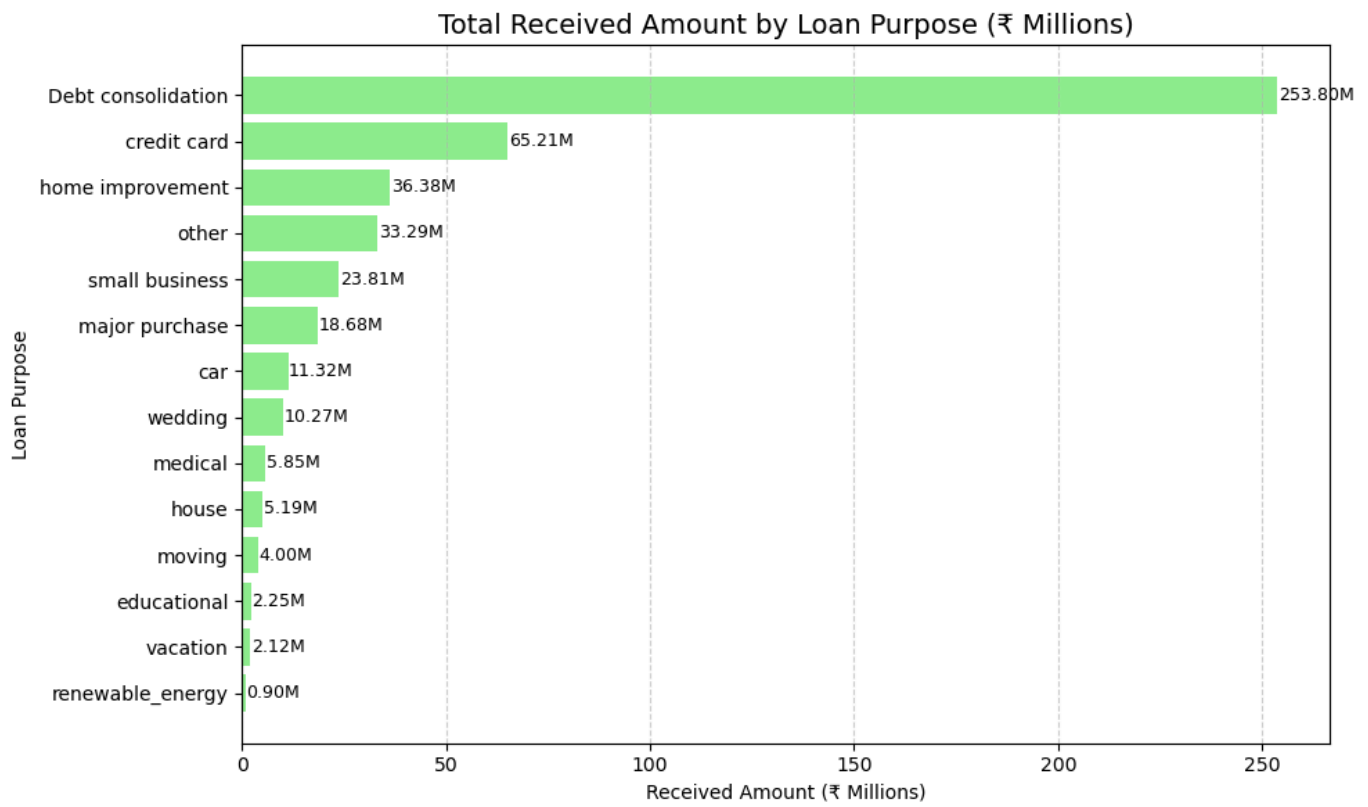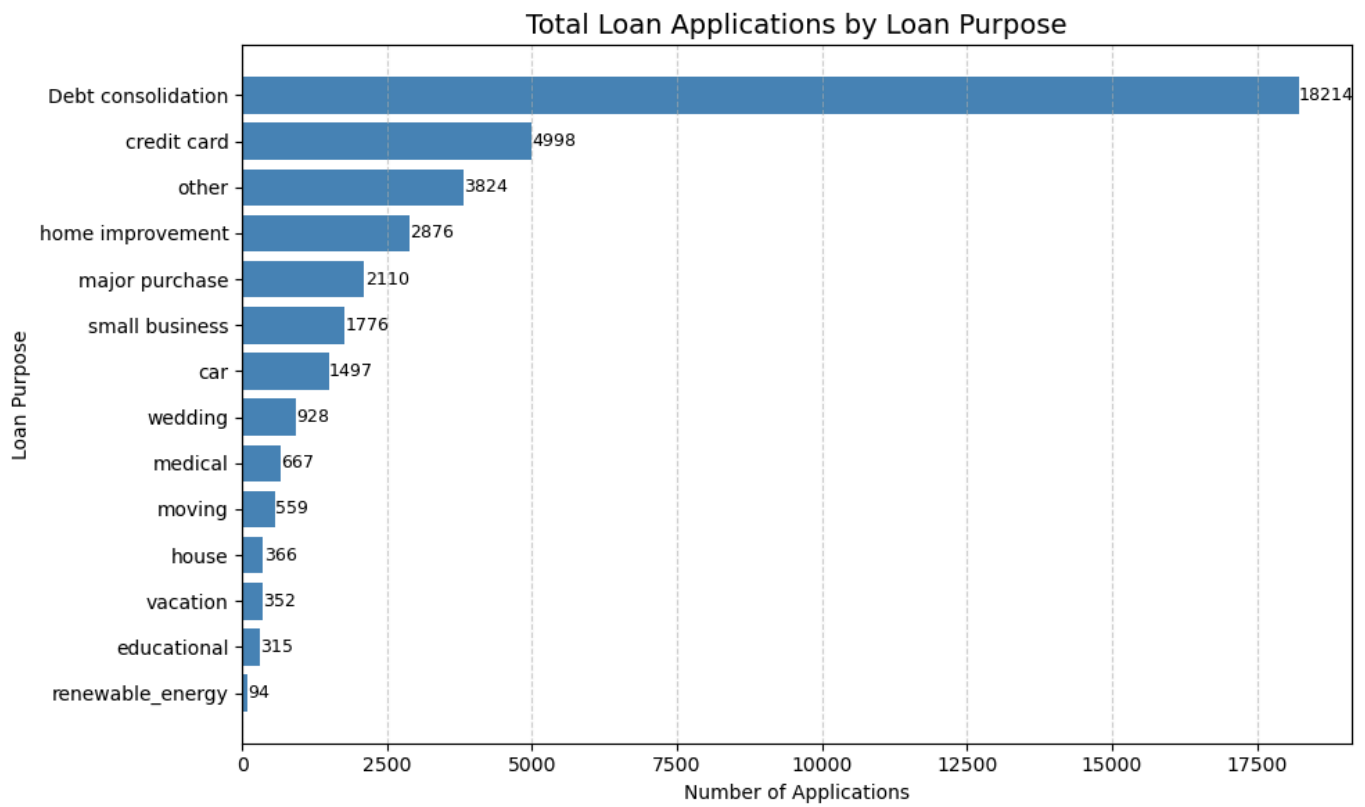
Total Received Amount by Loan Purpose (₹ Millions)

Loan Purpose	Received Amount (₹ Millions)
Debt consolidation	253.80M
credit card	65.21M
home improvement	36.38M
other	33.29M
small business	23.81M
major purchase	18.68M
car	11.32M
wedding	10.27M
medical	5.85M
house	5.19M
moving	4.00M
educational	2.25M
vacation	2.12M
renewable_energy	0.90M

In [35]:

```python
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()
```

Total Loan Applications by Loan Purpose

Loan Purpose	Number of Applications
Debt consolidation	18214
credit card	4998
other	3824
home improvement	2876
major purchase	2110
small business	1776
car	1497
wedding	928
medical	667
moving	559
house	366
vacation	352
educational	315
renewable_energy	94

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()
```
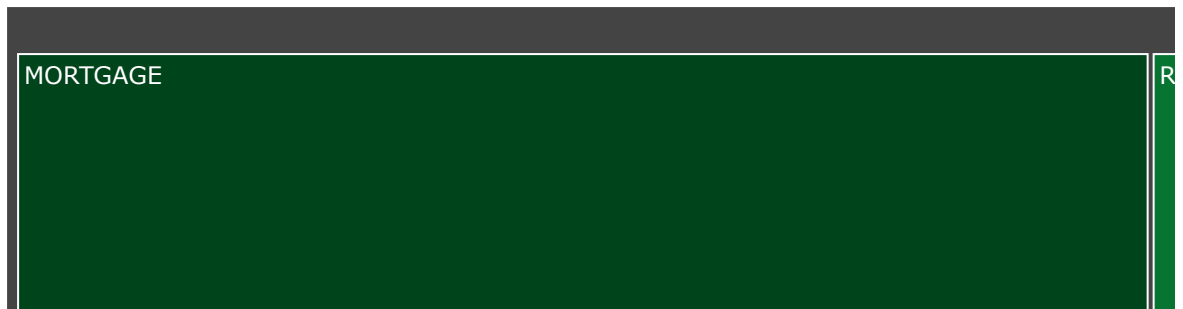
Total Received Amount by Home Ownership (₹ Millions)


```
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

RENT

MORTG