Bank Loan Analysis Project

Prepared by: N.H.P.M Nagoda

Tools Used: Python (NumPy, Pandas, Matplotlib, Seaborn), Power BI

Table of content

1. Introduction

Banks handle thousands of loan applications every month and managing these loans effectively is critical for financial stability. To make informed decisions, banks need clear insights into loan performance, repayment behavior, and borrower profiles. However, relying only on traditional reports can make it difficult to track trends, identify risks, and act quickly.

This project focuses on analyzing a bank loan dataset using Python, Power BI. Python was used for data cleaning and analysis and Power BI for building interactive dashboards. The analysis covers key performance indicators such as total loan applications, funded amounts, repayments, interest rates, and borrower debt-to-income ratios.

The purpose of this project is to transform raw data into meaningful insights. By classifying loans into good and bad categories, identifying borrower trends, and presenting the results through visual dashboards, this project helps stakeholders clearly understand the business problems and possible solutions.

2. Objectives

The objectives of this project are:

- Track KPIs such as Total Loan Applications, Funded Amount, Amount Received, Interest Rate, and DTI.
- Classify loans into Good Loans and Bad Loans for performance tracking.
- Analyze borrower profiles by loan terms, employment length, purpose, and home ownership.

Create dashboards to present data clearly to stakeholders.

3. Methodology

3.1 Data Collection

Dataset: Loan records containing fields such as loan amount, funded amount, interest rate, repayment received, employment length, loan purpose, and loan status.

3.2 Tools and Technologies

- Python: Data cleaning and analysis (NumPy, Pandas, Matplotlib, Seaborn).
- Power BI: Dashboard creation.

3.3 Data Preparation (Python Code Example)

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
```

Import Data

```
df = pd.read_excel("C:/Users/Maleesha Prasad/Desktop/Loan Analysis/financial_loan.xlsx")
```

Meta Data

```
print(df)
```

```
           id address_state application_type emp_length  \
0     1077430            GA       INDIVIDUAL   < 1 year
1     1072053            CA       INDIVIDUAL    9 years
2     1069243            CA       INDIVIDUAL    4 years
3     1041756            TX       INDIVIDUAL   < 1 year
4     1068350            IL       INDIVIDUAL  10+ years
...       ...           ...              ...        ...
38571  803452            NJ       INDIVIDUAL   < 1 year
38572  970377            NY       INDIVIDUAL    8 years
38573  875376            CA       INDIVIDUAL    5 years
38574  972997            NY       INDIVIDUAL    5 years
38575  682952            NY       INDIVIDUAL    4 years

                          emp_title grade home_ownership issue_date  \
0                             Ryder     C           RENT 2021-02-11
1                       MKC Accounting     E           RENT 2021-01-01
2                 Chemat Technology Inc     C           RENT 2021-01-05
3                   barnes distribution     B       MORTGAGE 2021-02-25
4                         J&J Steel Inc     A       MORTGAGE 2021-01-01
...                                 ...   ...            ...        ...
38571         Joseph M Sanzari Company     C       MORTGAGE 2021-07-11
38572                       Swat Fame      C           RENT 2021-10-11
38573  Anaheim Regional Medical Center     D           RENT 2021-09-11
38574              Brooklyn Radiology      D           RENT 2021-10-11
38575                   Allen Edmonds      F           RENT 2021-07-11
...
38574       24000          9          33677
38575       18000          7          27679

[38576 rows x 24 columns]
```

Type of Data

```
df.dtypes
```

```
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
```

Summary of all the information

```
df.describe()
```

Python

	id	issue_date	last_credit_pull_date	last_payment_date	next_payment_date	member_id	annual_income	dti	installment	int_rate	loan_amount	total_acc	tot
count	3.857600e+04	38576	38576	38576	38576	3.857600e+04	3.857600e+04	38576.000000	38576.000000	38576.000000	38576.000000	38576.000000	38
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+04	0.133274	326.862965	0.120488	11296.066855	22.132544	12
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+03	0.000000	15.690000	0.054200	500.000000	2.000000	
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+04	0.082100	168.450000	0.093200	5500.000000	14.000000	5
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+04	0.134200	283.045000	0.118600	10000.000000	20.000000	10
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+04	0.185900	434.442500	0.145900	15000.000000	29.000000	16
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+06	0.299900	1305.190000	0.245900	35000.000000	90.000000	58
std	2.113246e+05	NaN	NaN	NaN	NaN	2.668105e+05	6.429368e+04	0.066662	209.092000	0.037164	7460.746022	11.392282	9

First 5 rows and Last 5 rows of the Data Frame

```
df.head()
```

	id	address_state	application_type	emp_length	emp_title	grade	home_ownership	issue_date	last_credit_pull_date	last_payment_date	...	sub_grade	tern
0	1077430	GA	INDIVIDUAL	< 1 year	Ryder	C	RENT	2021-02-11	2021-09-13	2021-04-13	...	C4	6(month
1	1072053	CA	INDIVIDUAL	9 years	MKC Accounting	E	RENT	2021-01-01	2021-12-14	2021-01-15	...	E1	3(month
2	1069243	CA	INDIVIDUAL	4 years	Chemat Technology Inc	C	RENT	2021-01-05	2021-12-12	2021-01-09	...	C5	3(month
3	1041756	TX	INDIVIDUAL	< 1 year	barnes distribution	B	MORTGAGE	2021-02-25	2021-12-12	2021-03-12	...	B2	6(month
4	1068350	IL	INDIVIDUAL	10+ years	J&J Steel Inc	A	MORTGAGE	2021-01-01	2021-12-14	2021-01-15	...	A1	3(month

5 rows × 24 columns

```
df.tail()
```

	id	address_state	application_type	emp_length	emp_title	grade	home_ownership	issue_date	last_credit_pull_date	last_payment_date	...	sub_grade	te
38571	803452	NJ	INDIVIDUAL	< 1 year	Joseph M Sanzari Company	C	MORTGAGE	2021-07-11	2021-05-16	2021-05-16	...	C1	mon
38572	970377	NY	INDIVIDUAL	8 years	Swat Fame	C	RENT	2021-10-11	2021-04-16	2021-05-16	...	C1	mon
38573	875376	CA	INDIVIDUAL	5 years	Anaheim Regional Medical Center	D	RENT	2021-09-11	2021-05-16	2021-05-16	...	D5	mon
38574	972997	NY	INDIVIDUAL	5 years	Brooklyn Radiology	D	RENT	2021-10-11	2021-05-16	2021-05-16	...	D5	mon
38575	682952	NY	INDIVIDUAL	4 years	Allen Edmonds	F	RENT	2021-07-11	2021-05-16	2021-05-16	...	F3	mon

5 rows × 24 columns

4. Analysis & Findings

4.1 Total Loan Applications

The total loan applications show how many customers applied for loans overall.

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

```
Total Loan Applications :  38576
```

4.1.1 MTD - Loan Applications

The Month-to-Date (MTD) applications highlight the number of new applications received in the current month, which helps track short-term activity.

```
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(forDecember 2021):4314
```

4.2 Total Funded Amount

This metric shows the total amount of money the bank has disbursed as loans.

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

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

4.2.1 MTD - Funded Amount

The Month-to-Date (MTD) funded amount helps track how much has been given out in the current month.

```
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_funded_amount = mtd_data['loan_amount'].sum()
mtd_funded_amount_millions = mtd_funded_amount/1000000

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

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

4.3 Total Amount Received

This represents the total repayments collected from borrowers.

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

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

4.3.1 MTD - Amount Received

The Month-to-Date (MTD) received amount shows how much has been collected in the ongoing month, which reflects the bank's cash flow.

```
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_received_amount = mtd_data['total_payment'].sum()
mtd_received_amount_millions = mtd_received_amount/1000000

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

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

4.4 Average Interest Rate

The average interest rate measures the typical cost of borrowing for customers. It helps the bank understand its overall lending portfolio and the balance between affordability for customers and profitability for the bank.

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

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

4.5 Average Debt-to-Income Ratio (DTI)

The average DTI shows how much debt borrowers have compared to their income. A lower DTI means borrowers are financially healthier, while a higher DTI indicates higher repayment risks.

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

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

4.6 Good Loan vs Bad Loans

Loans are divided into two groups: good loans (fully paid or current) and bad loans (defaulted or charged off). This classification helps banks measure portfolio quality and credit risk.

4.6.1 Good Loans KPI

```
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 = good_loans['total_payment'].sum()

good_loan_funded_amount_millions = good_loan_funded_amount/1000000
good_loan_received_millions = good_loan_received/1000000

good_loan_percentage = (good_loan_applications/total_loan_applications) * 100

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

```
Good Loan Application :  1098
Good Loan Funded Amount : $18.87M
Good Loan Total Received : $24.20M
Percentage of Good Loan Applications : 2.85%
```

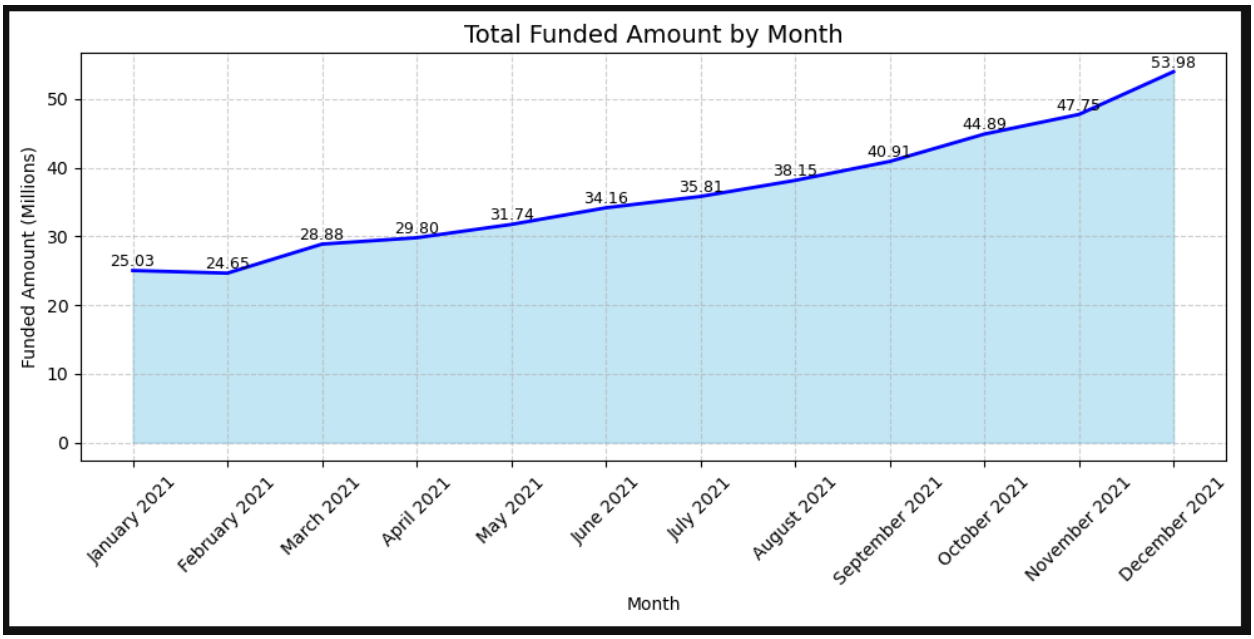
4.6.2 Bad Loans KPI

```
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 = bad_loans['total_payment'].sum()

bad_loan_funded_amount_millions = bad_loan_funded_amount/1000000
bad_loan_received_millions = bad_loan_received/1000000

bad_loan_percentage = (bad_loan_applications/total_loan_applications) * 100

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

```
Bad Loan Application :  5333
Bad Loan Funded Amount : $65.53M
Bad Loan Total Received : $37.28M
Percentage of Bad Loan Applications : 13.82%
```

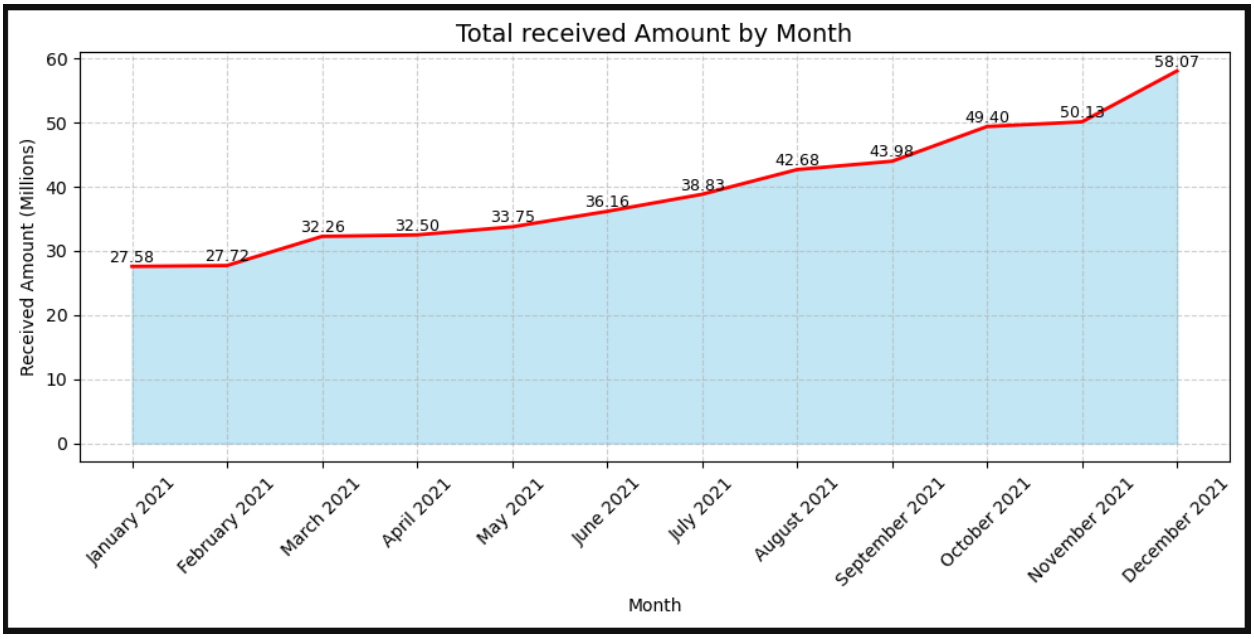
4.7 Monthly Trends by Issue Date

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

plt.figure(figsize=(10, 5))
plt.fill_between(monthly_funded['month_name'],monthly_funded['loan_amount_millions'],color='skyblue',alpha=0.5)
plt.plot(monthly_funded['month_name'],monthly_funded['loan_amount_millions'],color='blue',linewidth=2)

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

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



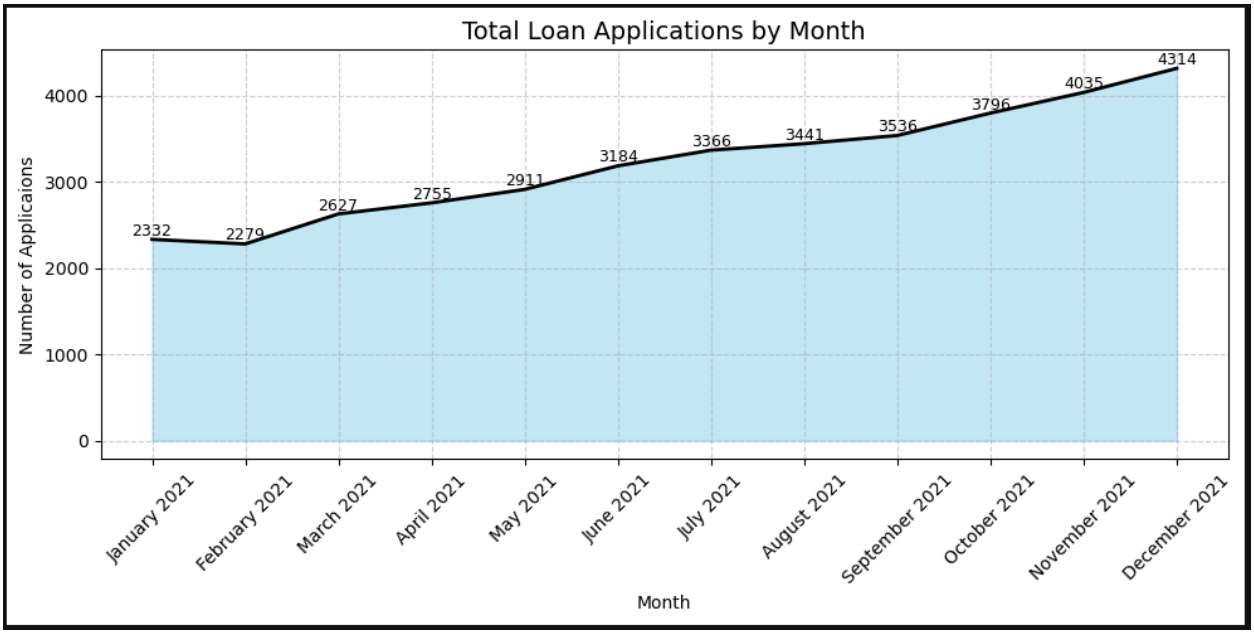
4.8 Monthly Trends by Issue Date for Total Amount Received

```
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='skyblue',alpha=0.5)
plt.plot(monthly_received['month_name'],monthly_received['received_amount_millions'],color='red',linewidth=2)

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

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



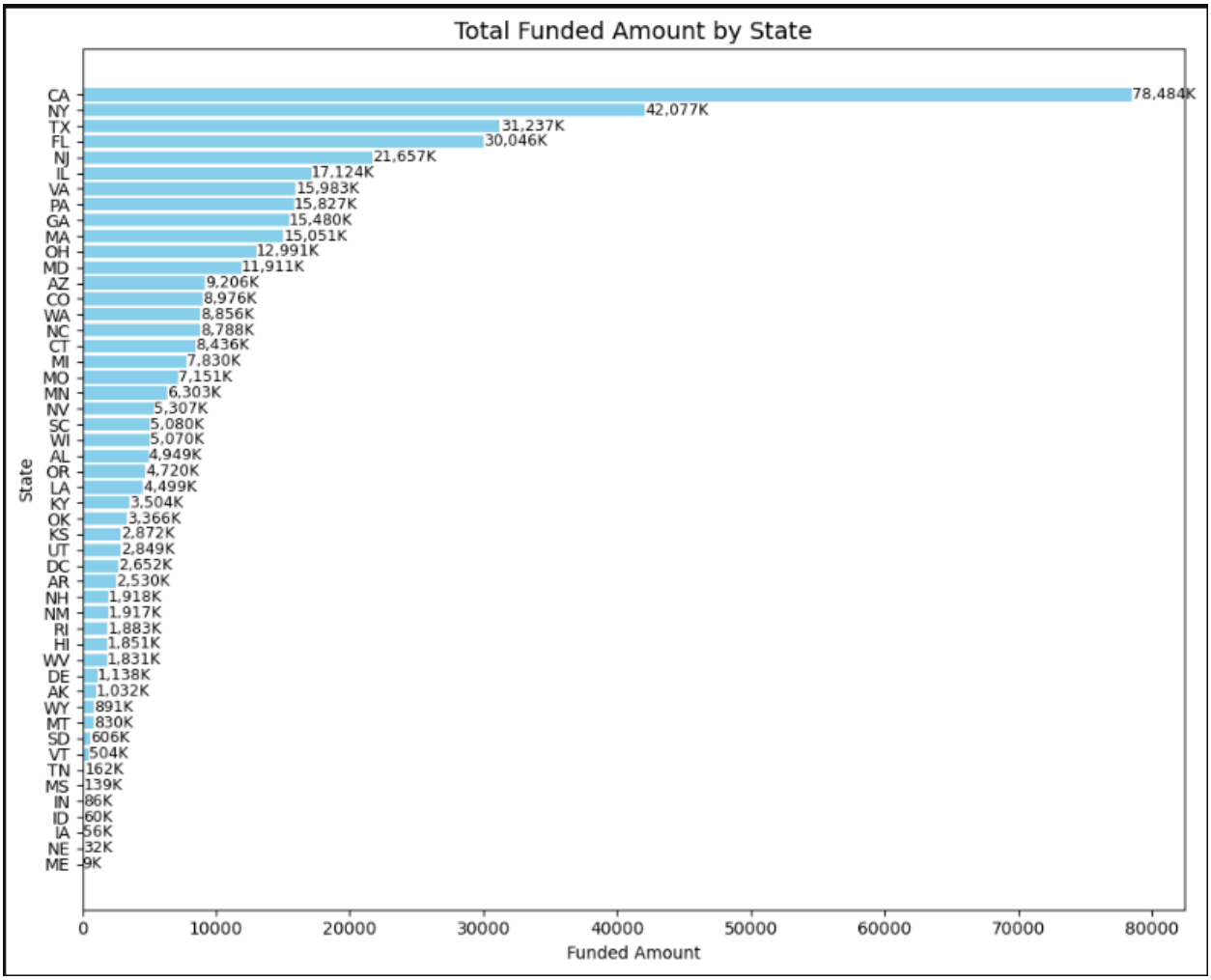
4.9 Monthly Trends by Issue Date for Total Loan Applications

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

plt.figure(figsize=(10, 5))
plt.fill_between(monthly_applications['month_name'],monthly_applications['loan_applications_count'],
                 color='skyblue',alpha=0.5)
plt.plot(monthly_applications['month_name'],monthly_applications['loan_applications_count'],
         color='black',linewidth=2)

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

plt.title('Total Loan Applications by Month',fontsize=14)
plt.xlabel('Month')
plt.ylabel('Number of Applicaions')
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()
```



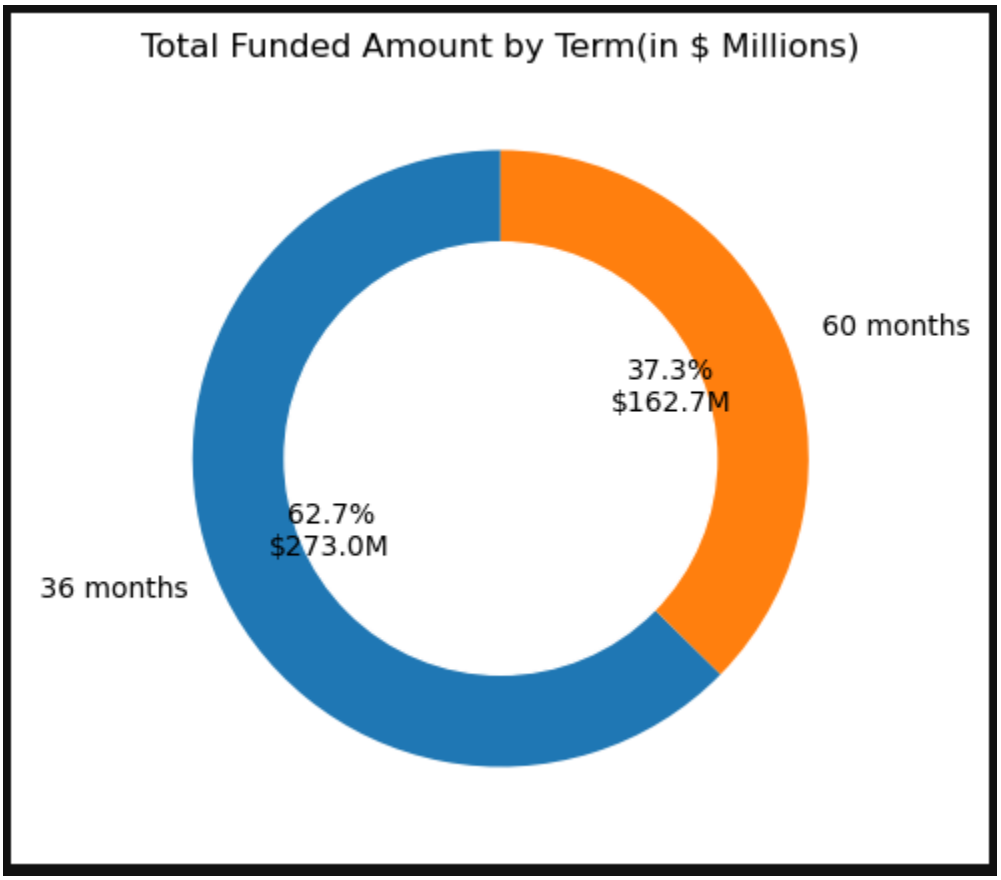
4.10 Regional Analysis by State for Total Funded Amount

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

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

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

plt.title('Total Funded Amount by State',fontsize=14)
plt.xlabel('Funded Amount')
plt.ylabel('State')
plt.tight_layout()
plt.show()
```



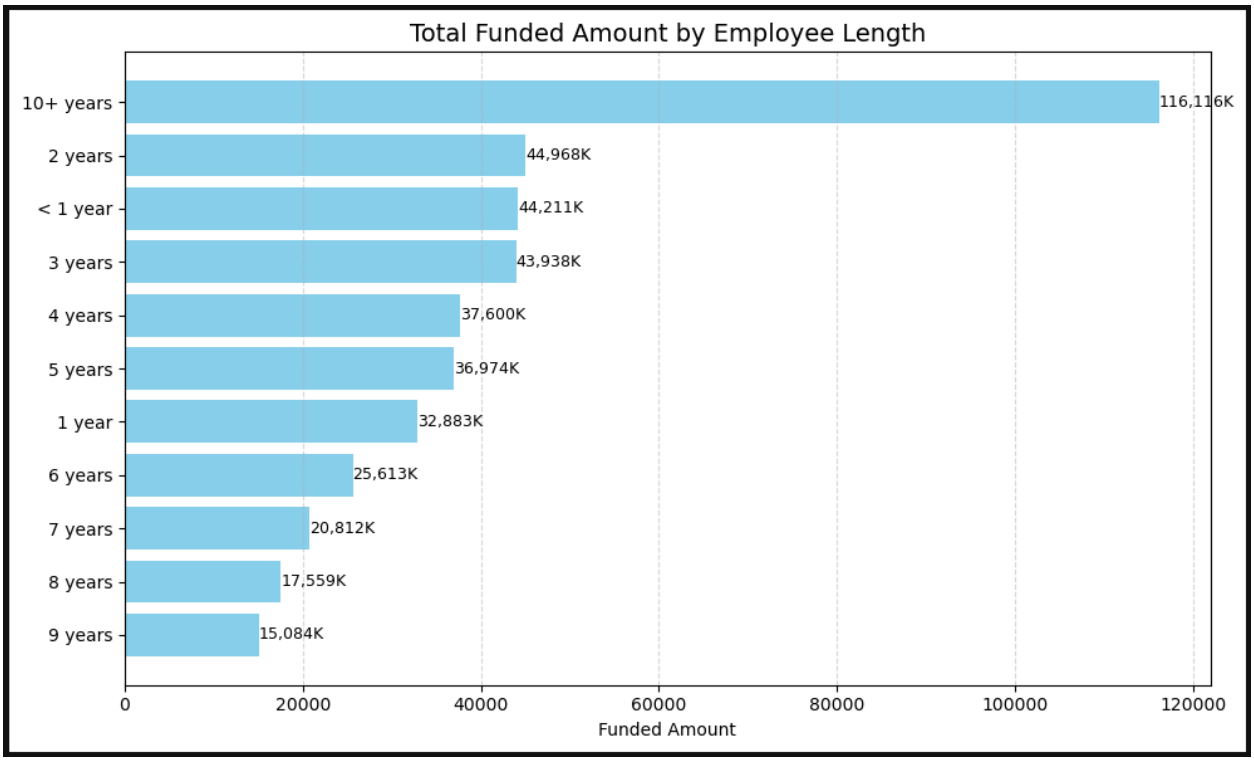
4.11 Loan Term Analysis by Total Funded Amount

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

plt.figure(figsize=(5,5))
plt.pie(
    term_funding_millions,
    labels = term_funding_millions.index,
    autopct = lambda p: f"{p:.1f}%\n${p*sum(term_funding_millions)/100:.1f}M",
    startangle = 90,
    wedgeprops = {'width':0.4}
)

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



4.12 Employee Length by Total Funded Amount

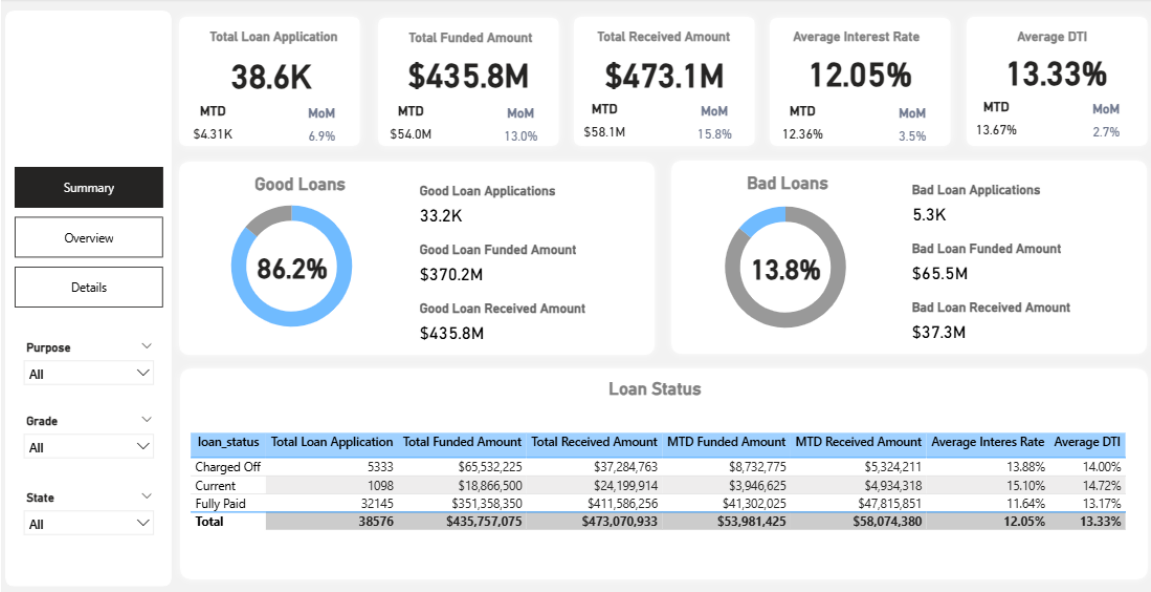
```
emp_funding = df.groupby('emp_length')['loan_amount'].sum().sort_values()
emp_funding_thousands = emp_funding/1000

plt.figure(figsize=(10, 6))
bars = plt.barh(emp_funding_thousands.index, emp_funding_thousands, 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 Employee Length',fontsize=14)
plt.xlabel('Funded Amount')
plt.grid(axis='x', linestyle='--',alpha=0.5)
plt.tight_layout()
plt.show()
```



5. Dashboards & Visualizations

5.1 Dashboard 1 – Summary

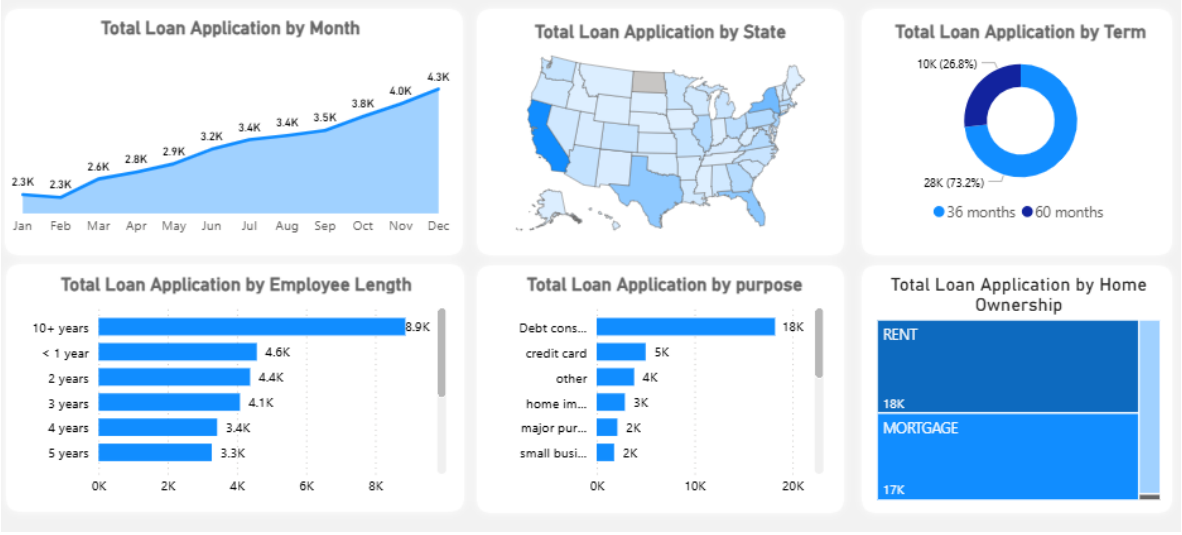
- KPIs: Total Loan Applications, Funded Amount, Amount Received.
- Good vs Bad Loan comparison.



loan_status	Total Loan Application	Total Funded Amount	Total Received Amount	MTD Funded Amount	MTD Received Amount	Average Interes Rate	Average DTI
Charged Off	5333	$65,532,225	$37,284,763	$8,732,775	$5,324,211	13.88%	14.00%
Current	1098	$18,866,500	$24,199,914	$3,946,625	$4,934,318	15.10%	14.72%
Fully Paid	32145	$351,358,350	$411,586,256	$41,302,025	$47,815,851	11.64%	13.17%
Total	**38576**	**$435,757,075**	**$473,070,933**	**$53,981,425**	**$58,074,380**	**12.05%**	**13.33%**

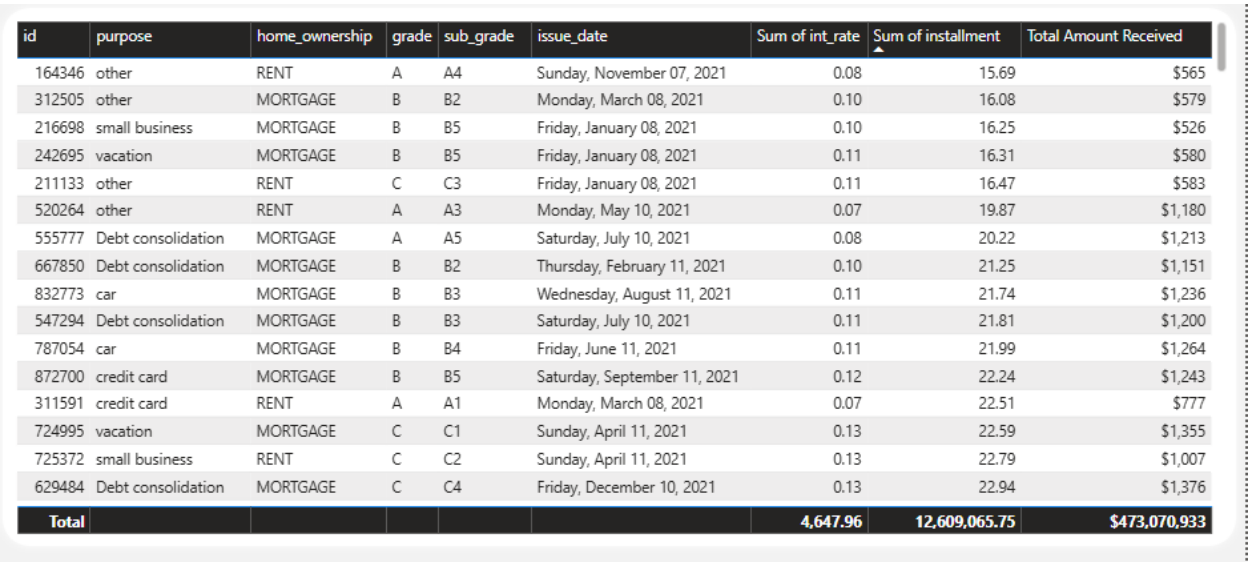
5.2 Dashboard 2 – Overview

- Monthly loan trends (line chart).
- Regional distribution (map).
- Loan purpose breakdown (bar chart).
- Employment length and home ownership analysis.



5.3 Dashboard 3 – Details

- Grid view with all loan-level details for deeper investigation.



id	purpose	home_ownership	grade	sub_grade	issue_date	Sum of int_rate	Sum of installment	Total Amount Received
164346	other	RENT	A	A4	Sunday, November 07, 2021	0.08	15.69	$565
312505	other	MORTGAGE	B	B2	Monday, March 08, 2021	0.10	16.08	$579
216698	small business	MORTGAGE	B	B5	Friday, January 08, 2021	0.10	16.25	$526
242695	vacation	MORTGAGE	B	B5	Friday, January 08, 2021	0.11	16.31	$580
211133	other	RENT	C	C3	Friday, January 08, 2021	0.11	16.47	$583
520264	other	RENT	A	A3	Monday, May 10, 2021	0.07	19.87	$1,180
555777	Debt consolidation	MORTGAGE	A	A5	Saturday, July 10, 2021	0.08	20.22	$1,213
667850	Debt consolidation	MORTGAGE	B	B2	Thursday, February 11, 2021	0.10	21.25	$1,151
832773	car	MORTGAGE	B	B3	Wednesday, August 11, 2021	0.11	21.74	$1,236
547294	Debt consolidation	MORTGAGE	B	B3	Saturday, July 10, 2021	0.11	21.81	$1,200
787054	car	MORTGAGE	B	B4	Friday, June 11, 2021	0.11	21.99	$1,264
872700	credit card	MORTGAGE	B	B5	Saturday, September 11, 2021	0.12	22.24	$1,243
311591	credit card	RENT	A	A1	Monday, March 08, 2021	0.07	22.51	$777
724995	vacation	MORTGAGE	C	C1	Sunday, April 11, 2021	0.13	22.59	$1,355
725372	small business	RENT	C	C2	Sunday, April 11, 2021	0.13	22.79	$1,007
629484	Debt consolidation	MORTGAGE	C	C4	Friday, December 10, 2021	0.13	22.94	$1,376
Total						**4,647.96**	**12,609,065.75**	**$473,070,933**

6. Business Insights & Recommendations

- Regions with high bad loan percentages should undergo stricter credit evaluation.
- Banks should focus on borrowers with low DTI ratios and longer employment history.
- Loan products can be designed around debt consolidation and credit repayment, as they are the most common purposes.
- Monitoring MTD and MoM KPIs helps identify repayment risks earlier.

7. Future Work

- Automate dashboards with real-time loan data.
- Apply machine learning models to predict defaults.
- Expand to customer segmentation and credit scoring models.

8. Conclusion

This project demonstrates how data analytics can transform raw loan data into meaningful insights. Using Python, and Power BI, I analyzed 38,000+ loan records, identified key performance indicators, and built dashboards that give decision-makers a clear view of loan performance.

The findings show actionable strategies for reducing risks and improving loan management. This project highlights my ability to use data analysis tools to solve real business problems.