Bank Loan Report in PowerBI - Finance Domain

DBMS System - **BankLoan**
Table - **financial_loan**
File used - **financial_loan.csv**
Software Used - **MySQL Workbench, Power BI v8.4, DAX Studio**

Objective: To conduct a thorough analysis of bank loan data to compare good loans versus bad loans, gain deep insights into customer behavior and loan purposes, and uncover various key metrics and actionable insights.

Scope:

1. **Data Extraction and Preparation**:
    - Use SQL to extract and preprocess bank loan data from the database.
    - Cleanse and transform data to ensure accuracy and consistency.
2. **Comparative Analysis**:
    - Differentiate between good loans and bad loans based on repayment status, default rates, and other relevant criteria.
    - Analyze the factors contributing to loan performance.
3. **Customer Behavior Insights**:
    - Segment customers based on demographics, credit scores, income levels, and other relevant attributes.
    - Identify patterns and trends in customer behavior related to loan applications and repayments.
4. **Loan Purpose Analysis**:
    - Categorize loans based on their purposes (e.g., personal, home, auto, business).
    - Assess the impact of loan purposes on repayment performance and default rates.
5. **Key Metrics and Insights**:
    - Develop comprehensive dashboards in Power BI to visualize key metrics such as approval rates, average loan amounts, interest rates, and repayment timelines.
    - Provide actionable insights to help improve loan approval processes, reduce default rates, and enhance customer satisfaction.
6. **Reporting and Visualization**:
    - Create interactive and dynamic reports in Power BI to present findings.
    - Ensure reports are user-friendly and can be easily interpreted by stakeholders.

Tools and Technologies:

- SQL for data extraction, cleaning, and transformation.
- Power BI for data visualization and reporting.

Outcome: A comprehensive analysis and reporting system that provides valuable insights into the performance of bank loans, customer behavior, and loan purposes. This will help the bank in making informed decisions to optimize loan approvals, minimize risks, and enhance overall customer satisfaction.

Part 1 - SQL

Best practice is to document the SQL queries along with results when making a Power BI representation. This helps in cross checking the visual data.

1. **Total Applications**

```
SELECT
     COUNT(id) AS total_loan_application
FROM financial_loan;
```

Result Grid | Filter Row

	total_loan_application
▶	38576

Month-To-Date Applications

This metric helps businesses and financial institutions monitor and analyze the trend and volume of loan applications within the current month.

```
SELECT
     COUNT(id) AS MTD_Total_Loan_Applcations
FROM financial_loan
WHERE MONTH(issue_date) = 12;
```

Result Grid | Filter Rows:

	MTD_Total_Loan_Applcations
▶	4313

PMTD Loan Applications

Previous month total loan applications

```
SELECT
     COUNT(id) AS PMTD_Total_Loan_Applications
FROM financial_loan
WHERE
    MONTH(issue_date) = 11
        AND YEAR(issue_date) = 2021;
```

	PMTD_Total_Loan_Applications
▶	4035

2. Total funded amount

```
SELECT
     SUM(loan_amount) AS total_funded_amount
FROM financial_loan;
```

	total_funded_amount
▶	435749075

MTD Total Funded Amount

```
SELECT
     SUM(loan_amount) AS MTD_Total_Funded_Amount
FROM financial_loan
WHERE
    MONTH(issue_date) = 12
        AND YEAR(issue_date) = 2021;
```

	MTD_Total_Funded_Amount
▶	53973425

PMTD Total Funded Amount

```
SELECT
     SUM(loan_amount) AS MTD_Total_Funded_Amount
FROM financial_loan
WHERE
    MONTH(issue_date) = 11
        AND YEAR(issue_date) = 2021;
```

	MTD_Total_Funded_Amount
▶	47754825

3. Total Amount Received

Amount received back from customers

```
SELECT
     SUM(total_payment) AS Total_Amount_Received
FROM financial_loan;
```

	Total_Amount_Received
▶	473059803

MTD Total Amount Received

```
SELECT
     SUM(total_payment) AS MTD_Total_Amount_Received
FROM financial_loan
WHERE
    MONTH(issue_date) = 12
        AND YEAR(issue_date) = 2021;
```

	MTD_Total_Amount_Received
▶	58063250

PMTD Total Amount Received

```
SELECT
     SUM(total_payment) AS PMTD_Total_Amount_Received
FROM financial_loan
WHERE
    MONTH(issue_date) = 11
        AND YEAR(issue_date) = 2021;
```

	PMTD_Total_Amount_Received
▶	50132030

4. Average Interest Rate

```
SELECT
     ROUND(AVG(int_rate) * 100, 2) AS Avg_Interest_Rate
FROM financial_loan;
```

	Avg_Interest_Rate
▶	12.04

MTD Avg Interest Rate

```
SELECT
     ROUND(AVG(int_rate) * 100, 2) AS MTD_Avg_Interest_Rate
FROM financial_loan
WHERE MONTH(issue_date) = 12
      AND YEAR(issue_date) = 2021;
```

	MTD_Avg_Interest_Rate
▶	12.39

PMTD Avg Interest Rate

```
SELECT
      ROUND(AVG(int_rate) * 100, 2)
      AS PMTD_Avg_Interest_Rate
FROM financial_loan
WHERE
      MONTH(issue_date) = 11
      AND YEAR(issue_date) = 2021;
```

	PMTD_Avg_Interest_Rate
▶	11.97

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

Shouldn't be too high or too low the DTI, the better chances the customer of getting loan issued again and it's good for the bank also. Chat Gpt more on this in the final draft.

```
SELECT
    ROUND(AVG(dti) * 100, 2) AS Avg_DTI
FROM financial_loan;
```

	Avg_DTI
▶	13.33

MTD Average DTI

```
SELECT
    ROUND(AVG(dti) * 100, 2) AS MTD_Avg_DTI
FROM financial_loan
WHERE
    MONTH(issue_date) = 12
```

```
 AND YEAR(issue_date) = 2021;
```

	MTD_Avg_DTI
▶	13.66

PMTD Average DTI

```
SELECT
    ROUND(AVG(dti) * 100, 2) AS PMTD_Avg_DTI
FROM financial_loan
WHERE
    MONTH(issue_date) = 11
      AND YEAR(issue_date) = 2021;
```

Result Grid | Filter R

	PMTD_Avg_DTI
▶	13.3

6. Good Loan Percentage

```
      SELECT
      (COUNT(CASE
         WHEN
                 loan_status = 'Fully Paid'
                  OR loan_status = 'Current'
               THEN id END) * 100.0) / COUNT(id) AS
Good_Loan_Percentage
FROM financial_loan;
```

	Good_Loan_Percentage
▶	86.17498

7. Good Loan Applications

```
SELECT
```

```
    COUNT(id) AS Good_Loan_Applications
FROM financial_loan
WHERE
    loan_status in (''Fully Paid', 'Current');
```

	Good_Loan_Applications
▶	33242

8. Good Loan Funded Amount

```
     SELECT
          SUM(loan_amount) AS Good_Loan_Funded_Amount
FROM financial_loan
WHERE
          loan_status IN ('Fully Paid' , 'Current');
```

	Good_Loan_Funded_Amount
▶	370216850

9. Good Loan Received Amount

```
     SELECT
          SUM(total_payment) AS Good_Loan_Received_Amount
FROM financial_loan
WHERE
          loan_status IN ('Fully Paid' , 'Current');
```

	Good_Loan_Received_Amount
▶	435775040

10. Bad Loan Percentage

```
SELECT
   round((COUNT(CASE
        WHEN loan_status = 'Charged Off' THEN id
    END) * 100.0) / COUNT(id),2) AS Good_Loan_Percentage
FROM financial_loan;
```

	Bad_Loan_Percentage
▶	13.83

11. Bad Loan Applications

```
SELECT
     COUNT(id) AS Bad_Loan_Applications
FROM financial_loan
WHERE
     loan_status = 'Charged Off';
```

	Bad_Loan_Applications
▶	5333

12. Bad Loan Funded Amount

```
SELECT
    SUM(loan_amount) AS Bad_Loan_Funded_Amount
FROM financial_loan
WHERE
    loan_status = 'Charged Off';
```

	Bad_Loan_Funded_Amount
▶	65532225

13. Bad Loan Amount Received

```
SELECT
      SUM(total_payment) AS Bad_Loan_Received_Amount
FROM financial_loan
WHERE
      loan_status = 'Charged Off';
```

	Bad_Loan_Received_Amount
▶	37284763

14. Grid View for all types of loan status

```
SELECT
    loan_status,
    COUNT(id) AS Total_Loan_Applications,
    SUM(loan_amount) AS Total_Amount_Funded,
    SUM(total_payment) AS Total_Amount_Received,
    ROUND(AVG(int_rate) * 100, 2) AS Average_Interest_Rate,
    ROUND(AVG(dti) * 100, 2) AS Average_DTI
FROM financial_loan
GROUP BY 1
ORDER BY 2 DESC;
```

	loan_status	Total_Loan_Applications	Total_Amount_Funded	Total_Amount_Received	Average_Interest_Rate	Average_DTI
▶	Fully Paid	32144	351350350	411575126	11.63	13.17
	Charged Off	5333	65532225	37284763	13.86	14
	Current	1098	18866500	24199914	15.05	14.72

15. Monthly Trends by Issue Date

```
SELECT
      MONTH(issue_date) AS Month_Number,
      MONTHNAME(issue_date) AS `Month`,
      COUNT(id) AS Total_Loan_Applications,
      SUM(loan_amount) AS Total_Funded_Amount,
      SUM(total_payment) AS Total_Amount_Received
FROM financial_loan
```

```
GROUP BY 1,2
ORDER BY 1;
```

Month_Number	Month	Total_Loan_Applications	Total_Funded_Amount	Total_Amount_Received
1	January	2332	25031650	27578836
2	February	2279	24647825	27717745
3	March	2627	28875700	32264400
4	April	2755	29800800	32495533
5	May	2911	31738350	33750523
6	June	3184	34161475	36164533
7	July	3366	35813900	38827220
8	August	3441	38149600	42682218
9	September	3536	40907725	43983948
10	October	3796	44893800	49399567
11	November	4035	47754825	50132030
12	December	4313	53973425	58063250

16. Regional Analysis by State

```
SELECT
      address_state AS State,
      COUNT(id) AS Total_Loan_Applications,
      SUM(loan_amount) AS Total_Funded_Amount,
      SUM(total_payment) AS Total_Amount_Received
FROM financial_loan
GROUP BY 1
ORDER BY 1;
```

State	Total_Loan_Applications	Total_Funded_Amount	Total_Amount_Received
AK	78	1031800	1108570
AL	432	4949225	5492272
AR	236	2529700	2777875
AZ	833	9206000	10041986
CA	6893	78476125	83890104
CO	770	8976000	9845810
CT	730	8435575	9357612
DC	214	2652350	2921854
DE	110	1138100	1269136
FL	2773	30046125	31601905
GA	1355	15480325	16728040

State	Total_Loan_Applications	Total_Funded_Amount	Total_Amount_Received
IA	5	56450	64482
ID	6	59750	65329
IL	1486	17124225	18875941
IN	9	86225	85521
KS	260	2872325	3247394
KY	320	3504100	3792530
LA	426	4498900	5001160
MA	1310	15051000	16676279
MD	1027	11911400	12985170
ME	3	9200	10808
MI	685	7829900	8543660
MN	592	6302600	6750746

State	Total_Loan_Applications	Total_Funded_Amount	Total_Amount_Received
MO	660	7151175	7692732
MS	19	139125	149342
MT	79	829525	892047
NC	759	8787575	9534813
NE	5	31700	24542
NH	161	1917900	2101386
NJ	1822	21657475	23425159
NM	183	1916775	2084485
NV	482	5307375	5451443
NY	3701	42077050	46108181
OH	1188	12991375	14330148
OK	293	3365725	3712649

State	Total_Loan_Applications	Total_Funded_Amount	Total_Amount_Received
OR	436	4720150	4966903
PA	1482	15826525	17462908
RI	196	1883025	2001774
SC	464	5080475	5462458
SD	63	606150	656514
TN	17	162175	141522
TX	2664	31236650	34392715
UT	252	2849225	2952412
VA	1375	15982650	17711443
VT	54	504100	534973
WA	805	8855525	9531739
WI	446	5070450	5485161

State	Total_Loan_Applications	Total_Funded_Amount	Total_Amount_Received
WY	79	890750	1046050
WV	167	1830525	1991936
WI	446	5070450	5485161

17. Loan Data with respect to terms

```
SELECT
      term AS Term,
      COUNT(id) AS Total_Loan_Applications,
      SUM(loan_amount) AS Total_Funded_Amount,
      SUM(total_payment) AS Total_Amount_Received
FROM financial_loan
GROUP BY 1
ORDER BY 1;
```

Term	Total_Loan_Applications	Total_Funded_Amount	Total_Amount_Received
36 months	28237	273041225	294709458
60 months	10338	162707850	178350345

18. Employee Length

```
SELECT
```

```
      emp_length AS Employee_Length,
      COUNT(id) AS Total_Loan_Applications,
      SUM(loan_amount) AS Total_Funded_Amount,
      SUM(total_payment) AS Total_Amount_Received
FROM financial_loan
GROUP BY 1
ORDER BY 1;
```

	Employee_Length	Total_Loan_Applications	Total_Funded_Amount	Total_Amount_Received
▶	< 1 year	4575	44210625	47545011
	1 year	3229	32883125	35498348
	10+ years	8870	116115950	125871616
	2 years	4382	44967975	49206961
	3 years	4088	43937850	47551832
	4 years	3428	37600375	40964850
	5 years	3273	36973625	40397571
	6 years	2227	25604650	27897528
	7 years	1772	20811725	22584136
	8 years	1476	17558950	19025777
	9 years	1255	15084225	16516173

19. Purpose of loan

```
SELECT
      purpose AS PURPOSE,
      COUNT(id) AS Total_Loan_Applications,
      SUM(loan_amount) AS Total_Funded_Amount,
      SUM(total_payment) AS Total_Amount_Received
FROM financial_loan
GROUP BY 1
ORDER BY 1;
```

PURPOSE	Total_Loan_Applications	Total_Funded_Amount	Total_Amount_Received
car	1496	10215575	11313784
credit card	4998	58885175	65214084
Debt consolidation	18214	232459675	253801871
educational	315	2161650	2248380
home improvement	2876	33350775	36380930
house	366	4824925	5185538
major purchase	2110	17251600	18676927
medical	667	5533225	5851372
moving	559	3748125	3999899
other	3824	31155750	33289676
renewable_energy	94	845750	898931
small business	1776	24123100	23814817
vacation	352	1967950	2116738
wedding	928	9225800	10266856

20. Home Ownership spread of loan data

```
SELECT
     home_ownership AS Home_Ownership,
     COUNT(id) AS Total_Loan_Applications,
     SUM(loan_amount) AS Total_Funded_Amount,
     SUM(total_payment) AS Total_Amount_Received
FROM financial_loan
GROUP BY 1
ORDER BY 1;
```

Home_Ownership	Total_Loan_Applications	Total_Funded_Amount	Total_Amount_Received
MORTGAGE	17197	219321150	238463308
NONE	3	16800	19053
OTHER	98	1044975	1025257
OWN	2838	29597675	31729129
RENT	18439	185768475	201823056

Part 2 - Power BI

- Check data quality in Power BI Query Editor, an in-built functionality of power BI.

- Go to View in Transform data and do a column quality check.

- Emp_title has null values but this column is okay to have null values.

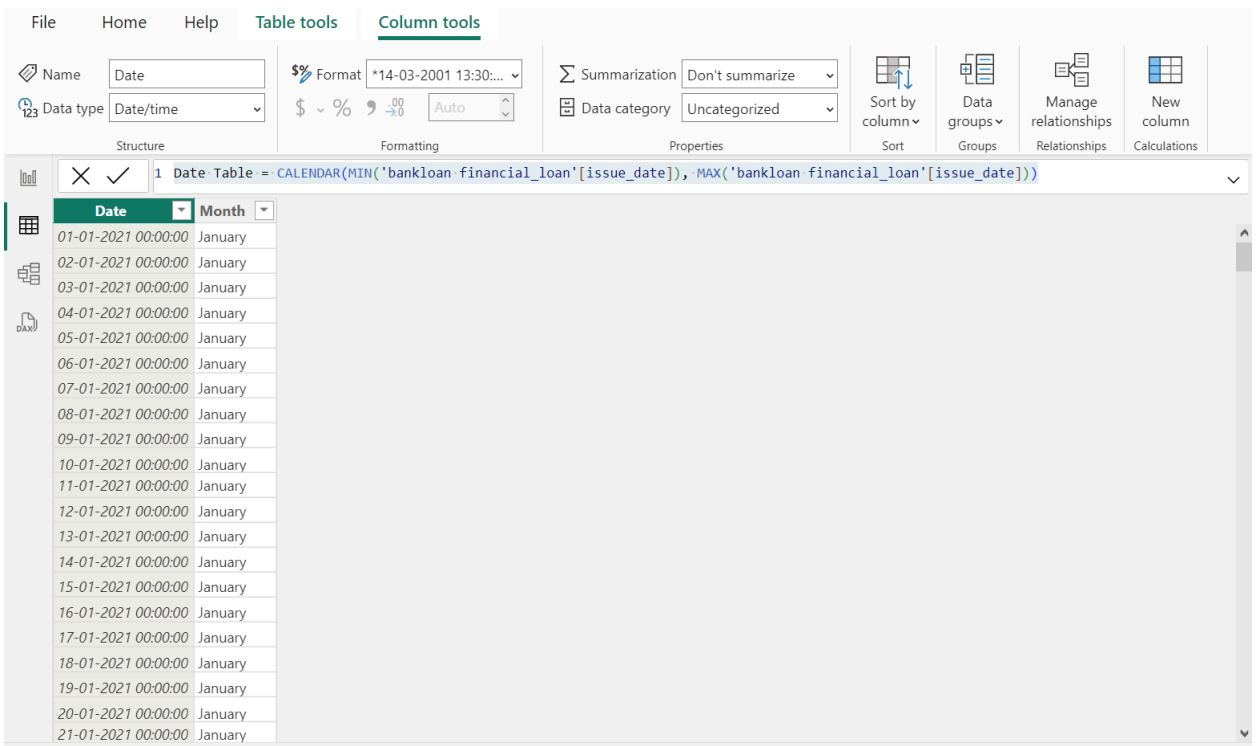
- We need to use time intelligence functions for MTDs, so we create a new ‘Date table’ using DAX query’s calendar function on issue_date.

```
Date Table = CALENDAR(MIN('bankloan financial_loan'[issue_date]),
MAX('bankloan financial_loan'[issue_date]))
```

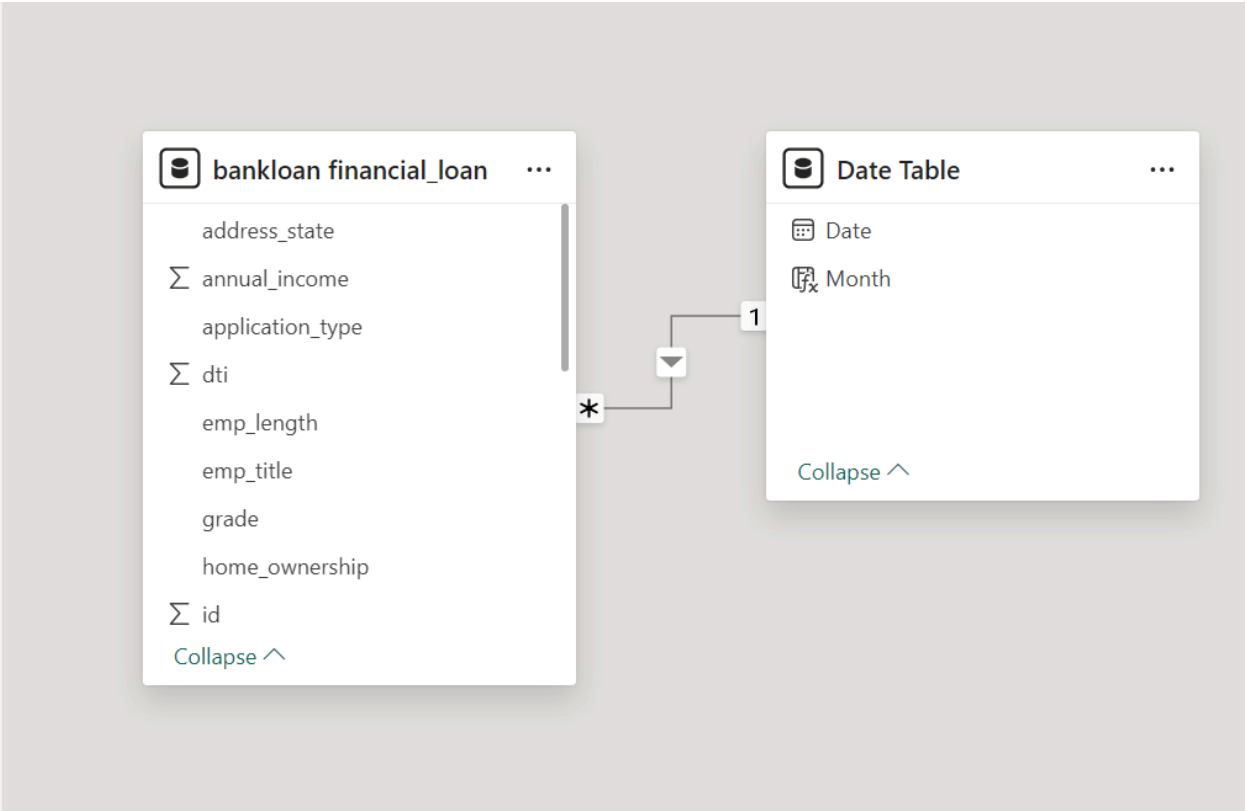
- issue_date will become the primary key of this table, as all the rows have unique values.

- Create a new column for this ‘Date Table’ called ‘Month’, to fetch the month names, using DAX query.

```
Month = FORMAT('Date Table'[Date], "mmmm")
```



So now we have two independent tables, who have no relationship with each other. We establish a one to many relationship in Data Modelling View by connecting the ‘date’ column with ‘issue_date’ in the main ‘bankloan financial_loan’ table.



Following are the new measures created, using DAX query to build the report.

1. Total Loan Applications

```
Total Loan Applications = COUNT('bankloan financial_loan'[id])
```

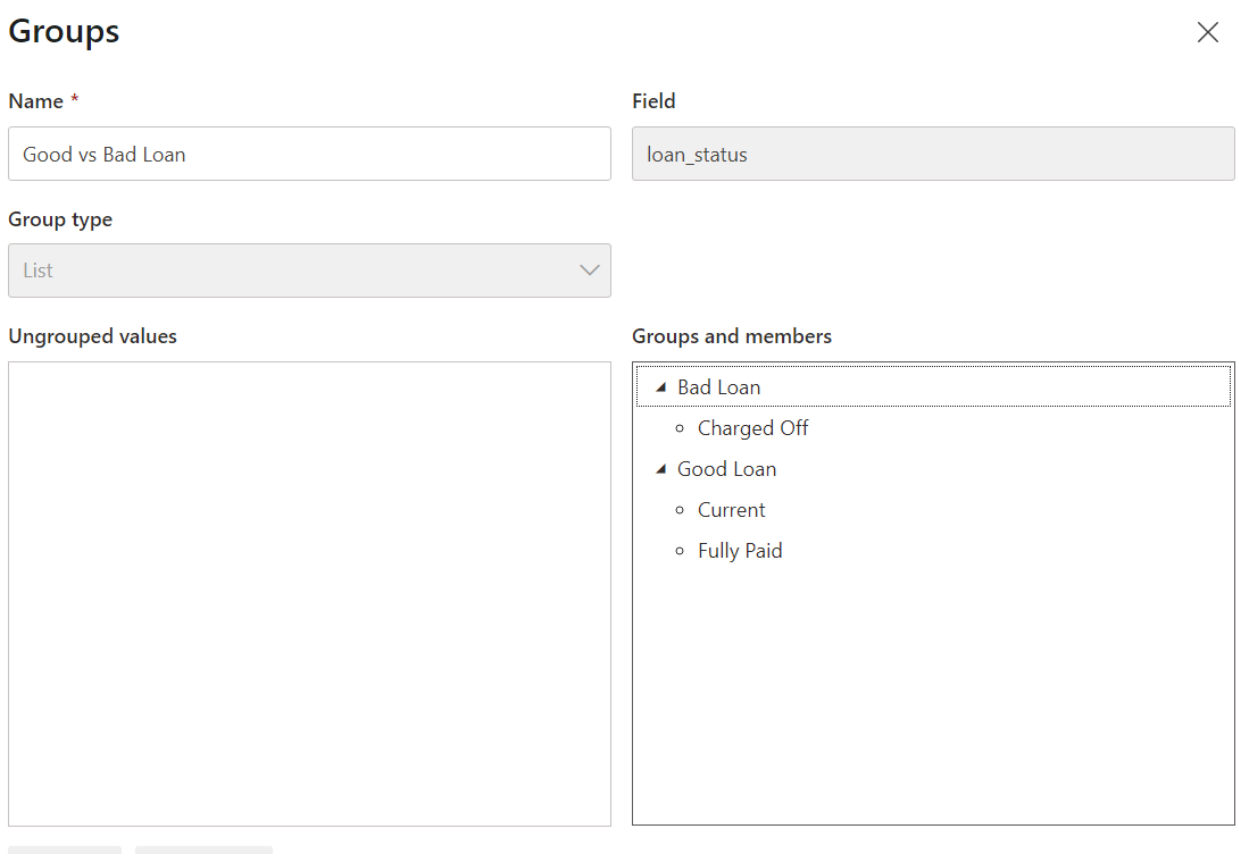
2. Total Funded Amount

```
Total Funded Amount = SUM('bankloan
financial_loan'[loan_amount])
```

3. Total Received Amount

```
Total Received Amount = SUM('bankloan
financial_loan'[total_payment])
```

We group by loan_status field. ‘Fully Paid’ and ‘Current’ come under the label ‘Good Loan’ and ‘Charged Off’ is labeled as ‘Bad Loan’.



Then we calculate the percentage using total loan applications.

4. Good Loan Percentage

```
Good Loan % = (CALCULATE([Total Loan Applications], 'bankloan
financial_loan'[Good vs Bad Loan] = "Good Loan" )) / [Total
Loan Applications]
```

5. Good Loan Applications

```
Good Loan Applications = CALCULATE([Total Loan Applications],
'bankloan financial_loan'[Good vs Bad Loan]= "Good Loan")
```

6. Good Loan Funded Amount

```
Good Loan Funded Amount = CALCULATE([Total Funded Amount],
'bankloan financial_loan'[Good vs Bad Loan]= "Good Loan")
```

7. Good Loan Received Amount

```
Good Loan Received Amount = CALCULATE([Total Received Amount],
'bankloan financial_loan'[Good vs Bad Loan]= "Good Loan")
```

8. Bad Loan Percentage

```
Bad Loan % = (CALCULATE([Total Loan Applications], 'bankloan
financial_loan'[Good vs Bad Loan] = "Bad Loan" )) / [Total Loan
Applications]
```

9. Bad Loan Applications

```
Bad Loan Applications = CALCULATE([Total Loan Applications],
'bankloan financial_loan'[Good vs Bad Loan]= "Bad Loan")
```

10. Bad Loan Funded Amount

```
Bad Loan Funded Amount = CALCULATE([Total Funded Amount],
'bankloan financial_loan'[Good vs Bad Loan]= "Bad Loan")
```

11. Bad Loan Received Amount

```
Bad Loan Received Amount = CALCULATE([Total Received Amount],
'bankloan financial_loan'[Good vs Bad Loan]= "Bad Loan")
```

12. Average DTI

```
Average DTI = AVERAGE('bankloan financial_loan'[dti])
```

13. Average Interest Rate

```
Average Interest Rate = AVERAGE('bankloan
financial_loan'[int_rate])
```

Create a new parameter to filter data according to 'Total Loan Applications', 'Total Funded Amount' and 'Total Received Amount'.

14. Select Measure

```
Select Measure = {
```

```
("Total Funded Amount", NAMEOF('bankloan financial_loan'[Total
Funded Amount]), 0),
("Total Loan Applications", NAMEOF('bankloan
financial_loan'[Total Loan Applications]), 1),
("Total Received Amount", NAMEOF('bankloan
financial_loan'[Total Received Amount]), 2)}
```