

Case on Banking Loan Risk (Dashboard)

I'll keep **simple English**, clear structure, and **business storytelling**

Project :- Loan Risk & Churn Analysis

Achieved

"Built an end-to-end loan risk and churn analytics solution using Python, MySQL, and Power BI, identifying 13% high-risk customers and delivering actionable insights through a 6-page executive dashboard."

End-to-End Data Analytics Project (Jupyter Notebook → MySQL → Power BI)

1. Problem Statement

Banks face significant financial losses due to **loan defaults and customer churn**. While most customers repay loans responsibly, a **small group of high-risk customers** can create **disproportionate risk** for the organization.

The key challenges addressed in this project are:

- Identifying **high-risk loan customers**
- Understanding **why customers churn or default**
- Segmenting customers into **Low, Medium, and High Risk**
- Presenting insights clearly to **non-technical stakeholders**
- Supporting **data-driven decision making** using dashboards

This project aims to build a **data-driven loan risk and churn monitoring system** using Python, SQL, and Power BI.

2. Objective (WHAT you want to achieve)

To analyze customer financial behavior, build a loan risk model, and identify high-risk churn customers using data analytics and Power BI.

Thought Process and Approach :-

- First of all , after get to know the issue/problem then starts to find the solution/goal.
set the goals (what to achieve ?).
- Find /gather or collecting the data from different sources.
- Once , collected the data then create a new relational database in MySQL ,
- By using connector and SQL alchemy ,import engine to extract the Raw data (tables) from MySQL to Jupyter Notebook.
- Then start normalize the raw data (after data cleaning),
- Next, do all EDA(Exploratory data analysis) like perform all necessary step which is required for this project like creating or adding new columns , feature engineering etc.
- After this all extract the new and cleaned table into the MySQL database for dashboard creation in Power BI.

I'll describe in brief , and step by step

1. About Dataset (Dataset Overview) –

This dataset basically contains information about bank details ,various client details which consists of multiple tables which are interlinked with each other through keys like primary key and foreign key.

The various tables are Banking Relationship, Client-Banking, Gender, Investment Advisor and Period.

The dataset contains **3,000 banking customers** with financial and behavioral attributes.

The data represents a **typical retail banking portfolio**.

The screenshot shows a Jupyter Notebook interface with the following code in cell [3]:

```
import pandas as pd
from sqlalchemy import create_engine

user = 'root'
password = '9832'
port = 3306
host = '127.0.0.1'
database = 'banking_case'

try:
    #create SQLAlchemy engine
    cnx = create_engine(f"mysql+mysqldb://({user}):({password})@({host})/({database})")
    print("Connection Successful!")

    query = "select * from banking_case.customers"
    df = pd.read_sql(query,cnx)
    print(df)
except ExceptionError as e:
    print("Error:",e)
```

Output of the code:

```
Connection Successful!
      ID CustomerID Name Age LocationID Joined Bank \
0   IND12345  Raymond Mills 24  34324 06-05-2019
1   IND65833  Julia Spencer 23  42205 10-12-2001
2   IND47499  Stephen Murray 27  7314 25-01-2010
3   IND72498  Virginia Garza 40  34594 28-03-2019
4   IND60181  Melissa Sanders 46  41269 20-07-2012
...
2995  IND66827  Earl Hall 82  8760 09-10-2014
2996  IND40556  Billy Williamson 44  32837 05-02-2009
2997  IND72414  Victor Black 70  36088 29-12-2009
2998  IND46652  Andrew Ford 56  24871 13-02-2006
2999  IND48216  Amy Nguyen 79  38518 08-12-2005

      Banking Contact Nationality          Occupation \
0   Anthony Torres   American  Safety Technician IV
1   Jonathan Hawkins  African  Safety Technician
2   Anthony Berry  European  Help Desk Operator
```

[3000 rows x 25 columns]																					
df.head(5)																					
[5]:	T=Client ID	Name	Age	Location ID	Joined Bank	Banking Contact	Nationality	Occupation	Fee Structure	Loyalty Classification	—	Bank Deposits	Checking Accounts	Saving Accounts	Foreign Currency Account	Business Lending	Properties Owned	Risk Weighting	BRId	GenderId	IAlid
0	IND81288	Raymond Mills	24	34324	06-05-2019	Anthony Torres	American	Safety Technician IV	High	Jade	—	1485828.64	603617.88	607332.46	12249.96	1134475.30	1	2	1	1	1
1	IND65833	Julia Spencer	23	42205	10-12-2001	Jonathan Hawkins	African	Software Consultant	High	Jade	—	641482.79	229521.37	344635.16	61162.31	2000526.10	1	3	2	1	2
2	IND47499	Stephen Murray	27	7314	25-01-2010	Anthony Berry	European	Help Desk Operator	High	Gold	—	1033401.59	652674.69	203054.35	79071.78	548137.58	1	3	3	2	3
3	IND72498	Virginia Garza	40	34594	28-03-2019	Steve Diaz	American	Geologist II	Mid	Silver	—	1048157.49	1048157.49	234685.02	57513.65	1148402.29	0	4	4	1	4
4	IND60181	Melissa Sanders	46	41269	20-07-2012	Shawn Long	American	Assistant Professor	Mid	Platinum	—	487782.53	446644.25	128351.45	30012.14	1674412.12	0	3	1	2	5

5 rows x 25 columns

2. Project Architecture (End-to-End Flow)

This project follows a **real-world analytics workflow**:

1. **Data Processing & Feature Engineering**
→ Jupyter Notebook (Python)
2. **Data Storage & Aggregation**
→ MySQL Database
3. **Business Intelligence & Visualization**
→ Power BI Dashboard (6 Pages)

This architecture reflects how analytics projects are executed in **actual banking and financial organizations**.

4. Data Preparation & Feature Engineering (Jupyter Notebook)

4.1 Data Cleaning

- Checked for missing values
- Verified numeric scales
- Normalized ratios (e.g., debt-to-income)
- Ensured consistency across financial variable
- Eliminate outliers

Home X Banking_case X +

localhost:8888/notebooks/Banking_case.ipynb

jupyter Banking_case Last Checkpoint: 4 days ago

File Edit View Run Kernel Settings Help

Start to do the proper Data Cleaning

```
[9]: #standardize column names:-  
#why:- lowercase + underscore avoids spaces/Unicode issues and is easiest to remember.  
#Using single rule avoids typos.  
df.columns = df.columns.str.strip().str.lower().str.replace("[^0-9a-z_]", "", regex=True)  
  
# to clean all white spaces of the value from each and every columns  
  
for i in df.select_dtypes(include = ['object']).columns:  
    df[i] = df[i].astype(str).str.strip()  
df.info()  
df.head(5)
```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 25 columns):
 # Column Non-Null Count Dtype

 0 client_id 3000 non-null object
 1 name 3000 non-null object
 2 age 3000 non-null int64
 3 location_id 3000 non-null int64
 4 joined_bank 3000 non-null object
 5 banking_contact 3000 non-null object
 6 nationality 3000 non-null object
 7 occupation 3000 non-null object
 8 fee_structure 3000 non-null object
 9 loyalty_classification 3000 non-null object
 10 estimated_income 3000 non-null float64
 11 superannuation_savings 3000 non-null float64
 12 amount_of_credit_cards 3000 non-null int64
 13 credit_card_balance 3000 non-null float64
 14 bank_loans 3000 non-null float64
 15 bank_deposits 3000 non-null float64
 16 checking_accounts 3000 non-null float64
 17 saving_accounts 3000 non-null float64
 18 foreign_currency_account 3000 non-null float64
 19 business_lending 3000 non-null float64



```
[4]: df.rename(columns={df.columns[0]:"Client ID"},inplace=True)  
df.head(5) #it is used to rename the first column name it was wrong
```

	Client ID	Name	Age	Location ID	Joined Bank	Banking Contact	Nationality	Occupation	Fee Structure	Loyalty Classification	...	Bank Deposits	Checking Accounts	Sa
0	IND81288	Raymond Mills	24	34324	06-05-2019	Anthony Torres	American	Safety Technician IV	High	Jade	...	1485828.64	603617.88	6073
1	IND65833	Julia Spencer	23	42205	10-12-2001	Jonathan Hawkins	African	Software Consultant	High	Jade	...	641482.79	229521.37	3446
2	IND47499	Stephen Murray	27	7314	25-01-2010	Anthony Berry	European	Help Desk Operator	High	Gold	...	1033401.59	652674.69	2030
3	IND72498	Virginia Garza	40	34594	28-03-2019	Steve Diaz	American	Geologist II	Mid	Silver	...	1048157.49	1048157.49	2346
4	IND60181	Melissa Sanders	46	41269	20-07-2012	Shawn Long	American	Assistant Professor	Mid	Platinum	...	487782.53	446644.25	1283

5 rows × 25 columns

```
[12]: df.describe() #to check the overview of the dataset
```

	Age	Location_ID	Estimated_Income	Superannuation_Savings	Amount_of_Credit_Cards	Credit_Card_Balance	Bank_Loans	Bank_Deposits
count	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3.000000e+03	3.000000e+03
mean	51.039667	21563.323000	171305.034263	25531.599673	1.463667	3176.206943	5.913862e+05	6.715602e+05
std	19.854760	12462.273017	111935.808209	16259.950770	0.676387	2497.094709	4.575570e+05	6.457169e+05
min	17.000000	12.000000	15919.480000	1482.030000	1.000000	1.170000	0.000000e+00	0.000000e+00
25%	34.000000	10803.500000	82906.595000	12513.775000	1.000000	1236.630000	2.396281e+05	2.044004e+05
50%	51.000000	21129.500000	142313.480000	22357.355000	1.000000	2560.805000	4.797934e+05	4.633165e+05
75%	60.000000	22051.500000	212200.205000	25151.710000	2.000000	1522.622500	6.256120e+05	6.187510e+05

```

[11]: df['client_id'] = df['client_id'].astype(str).str.strip()
df['name'] = df['name'].astype(str).str.strip()
print(df['client_id'].head().tolist())
print(df['name'].head().tolist())
['IND81288', 'IND65833', 'IND47499', 'IND72498', 'IND60181']
['Raymond Mills', 'Julia Spencer', 'Stephen Murray', 'Virginia Garza', 'Melissa Sanders']

[12]: # Handle missing values - we have (1190 non-null values out of 3000), it means we have more than 50% null values in joined_data
# create a flag "joined_known"
# Flag missing "joined_bank"
df['joined_known'] = df['joined_bank'].notna().astype(int)

# Impute numeric columns if needed (example: estimated_income)
df['estimated_income'] = df['estimated_income'].fillna(df['estimated_income'].median())

```

Deal or remove outliers

```

: def remove_outliers_iqr(series,k=1.5):
    q1 = series.quantile(0.25)
    q3 = series.quantile(0.75)
    iqr = q3 - q1
    lower = q1 - k*iqr
    upper = q3 + k*iqr
    return series.clip(lower,upper)

# Clip extreme estimated income to reasonable range

df['estimated_income_clipped'] = remove_outliers_iqr(df['estimated_income'])

```

4.2 Loan Risk Churn Definition

Two churn modeling approaches were tested:

Method 1: Rule-Based Risk Paths (Final Choice)

Customers are flagged as high-risk churn if they meet **any of the following**:

- High debt-to-income + high risk score
- Low deposits + high total debt + low income
- New customer with high risk score

This method produced a **realistic churn rate (~13%)**, which aligns with real banking scenarios.

Chosen because it is **interpretable, business-friendly**.

```

[61]: import numpy as np
import pandas as pd

# ----- 1) percentile thresholds (method1) -----
dti_80 = np.percentile(df['debt_to_income'], 80)
risk_80 = np.percentile(df['risk_score'], 80)
deposit_20 = np.percentile(df['bank_deposits'], 20)
debt_80 = np.percentile(df['total_debt'], 80)
income_20 = np.percentile(df['estimated_income'], 20)
years_20 = np.percentile(df['years_with_bank'], 20)

# ----- 2) loan_risk_churn_method1 (binary flag using percentile paths) -----
df['loan_risk_churn_method1'] = np.where(
    # Path 1 : High DTI + High Risk
    (df['debt_to_income'] > dti_80) & (df['risk_score'] > risk_80)) |

    # Path 2 : Low deposits + Low income + high debt
    ((df['bank_deposits'] < deposit_20) & (df['total_debt'] > debt_80) &
     (df['estimated_income'] < income_20)) |

    # Path 3 : New customers with high risk
    ((df['years_with_bank'] < years_20) & (df['risk_score'] > risk_80)),
    1, 0
)

# ----- 3) loan_risk_churn_method2 (alternate scoring approach you had earlier) -----
# Keep the previous metric approach (if you also made method2 earlier).
# If method2 isn't present, you can compute it similarly. Here assume it already exists.
# df['loan_risk_churn_method2'] = ... # (skip if already exists)

# ----- 4) Build a clean, bounded risk_model_score (0-100) -----
# We'll normalize each numeric component safely (avoid divide-by-zero),
# then combine using weights (weights add to 1), and scale to 0-100.

# pick the numeric features we want to use
num_cols = {
    'debt_to_income': 0.30,           # weight 30%
    'risk_score': 0.40,              # weight 40% (risk_score is already 0-100)
    'bank_deposits': 0.15,            # weight 15% (we invert deposits: less deposit => higher risk)
    'total_debt': 0.15               # weight 15%
}

# pack the numeric features we want to use
num_cols = {
    'debt_to_income': 0.30,           # weight 30%
    'risk_score': 0.40,              # weight 40% (risk_score is already 0-100)
    'bank_deposits': 0.15,            # weight 15% (we invert deposits: less deposit => higher risk)
    'total_debt': 0.15               # weight 15%
}

# compute safe max (replace 0 with small number to avoid division by zero)
safe_max = {}
for c in ['debt_to_income', 'risk_score', 'bank_deposits', 'total_debt']:
    m = df[c].max()
    safe_max[c] = m if m and m > 0 else 1.0

# normalized components (0..1)
# - for debt_to_income: bigger => worse => keep as-is
# - for risk_score: it's 0..100 => normalize by 100
# - for bank_deposits: bigger => better => we invert to make bigger => lower risk
# - for total_debt: bigger => worse
df['norm_dti'] = df['debt_to_income'] / safe_max['debt_to_income']
df['norm_risk_score'] = (df['risk_score'] / 100.0) # already 0-100
df['norm_deposits_inv'] = 1.0 - (df['bank_deposits'] / safe_max['bank_deposits'])
df['norm_total_debt'] = df['total_debt'] / safe_max['total_debt']

# ensure no NaN and clip 0..1
for col in ['norm_dti', 'norm_risk_score', 'norm_deposits_inv', 'norm_total_debt']:
    df[col] = df[col].fillna(0).clip(0,1)

# weighted sum (weights sum to 1)
w_dti = num_cols['debt_to_income']
w_risk = num_cols['risk_score']
w_dep = num_cols['bank_deposits']
w_debt = num_cols['total_debt']

```

```

# weighted sum (weights sum to 1)
w_dti = num_cols['debt_to_income']
w_risk = num_cols['risk_score']
w_dep = num_cols['bank_deposits']
w_debt = num_cols['total_debt']

df['risk_model_score'] = (
    df['norm_dti'] * w_dti +
    df['norm_risk_score'] * w_risk +
    df['norm_deposits_inv'] * w_dep +
    df['norm_total_debt'] * w_debt
) * 100.0 # scale to 0-100

# Clip strictly to 0..100 (safety)
df['risk_model_score'] = df['risk_model_score'].clip(0, 100)

# ----- 5) Create categorical segments from score -----
bins = [0, 40, 70, 100]
labels = ['Low Risk', 'Medium Risk', 'High Risk']
df['risk_segment_final'] = pd.cut(df['risk_model_score'], bins=bins, labels=labels, include_lowest=True, right=True)

# ----- 6) Check results -----
print("loan_risk_churn_method1 distribution (%)")
print(df['loan_risk_churn_method1'].value_counts(normalize=True) * 100)

if 'loan_risk_churn_method2' in df.columns:
    print("\nloan_risk_churn_method2 distribution (%)")
    print(df['loan_risk_churn_method2'].value_counts(normalize=True) * 100)

print("\nRisk model score stats:")
print(df['risk_model_score'].describe())

```

```

if 'loan_risk_churn_method2' in df.columns:
    print("\nloan_risk_churn_method2 distribution (%)")
    print(df['loan_risk_churn_method2'].value_counts(normalize=True) * 100)

print("\nRisk model score stats:")
print(df['risk_model_score'].describe())

print("\nRisk segments counts:")
print(df['risk_segment_final'].value_counts(dropna=False))

# ----- 7) OPTIONAL: delete the old column 'Loan_risk_churn' if you want -----
if 'loan_risk_churn' in df.columns:
    df.drop(columns=['loan_risk_churn'], inplace=True)
    print("\nDropped old column 'loan_risk_churn'.")


loan_risk_churn_method1 distribution (%)
loan_risk_churn_method1
0    86.833333
1    13.166667
Name: proportion, dtype: float64

loan_risk_churn_method2 distribution (%)
loan_risk_churn_method2
0    52.0
1    48.0
Name: proportion, dtype: float64

Risk model score stats:
count    3000.000000
mean     34.514360
std      9.888196

```

The screenshot shows a Jupyter Notebook interface with the title 'Banking_case'. The code cell [40] contains: `df['loan_risk_churn_method2'] = df['loan_risk_churn_method2'].astype(int)`. The code cell [41] contains: `[40]: df.drop(columns=['loan_risk_churn'], inplace=True)`. The code cell [42] contains: `[43]: ## Check both columns tested 1 and tested 2 like method1 and method2` followed by `df[['loan_risk_churn_method1','loan_risk_churn_method2']].head()`. The resulting table output is:

	loan_risk_churn_method1	loan_risk_churn_method2
0	0	0
1	0	0
2	0	1
3	0	0
4	1	1

The code cell [43] also displays the head of the DataFrame. The code cell [44] contains: `[45]: a = df['loan_risk_churn_method2'].value_counts(normalize =True) *100`, `b = df['loan_risk_churn_method1'].value_counts(normalize = True) *100`, and `print(a,b)`. The resulting proportion output is:

Method	Proportion (%)
Method2	52.0
Method1	48.0

The code cell [45] also displays the proportion output for Method1 and Method2.

The code cell [46] contains: `[47]: print("Method1 % :",(df['loan_risk_churn_method1'].value_counts(normalize=True)*100).to_dict())` and `print("Method2 % :",(df['loan_risk_churn_method2'].value_counts(normalize=True)*100).to_dict())`. The resulting percentage output is:

Method	Percentage (%)
Method1	{0: 86.83333333333333, 1: 13.166666666666666}
Method2	{0: 52.0, 1: 48.0}

4.3 Risk Model Score Creation

A **custom weighted risk score (0–100)** was developed using:

- Debt-to-income ratio
- Risk score
- Bank deposits (inverse effect)
- Total debt

This score provides a **continuous measure of customer risk**, rather than a simple yes/no label.

4.4 Risk Segmentation

Customers were classified into:

- **Low Risk** (Score < 40)
- **Medium Risk** (40–70)
- **High Risk** (>70)

This segmentation enables **targeted business actions**.

```
[66]: agg = df.groupby('risk_segment_final', observed=False).agg(
    customers_count=('client_id', 'nunique'),
    avg_risk=('_risk_model_score', 'mean'),
    pct_churn_method1=('_loan_risk_churn_method1', 'mean')
).reset_index()

agg.to_sql('customers_risk_agg', engine, if_exists='replace', index=False)

[66]: 3
```

5. Data Storage (MySQL)

Two tables were created:

a) First Table :- **customers_cleaned**

- Customer-level detailed data
- Used for deep analysis and Power BI visuals

b) Second Table :- **customers_risk_agg**

- Aggregated KPIs by risk segment
- Optimized for dashboard performance

Using MySQL reflects **enterprise-level data pipelines**, where Power BI connects to structured databases.

```
] # save cleaned full table and aggregated table
df.to_sql('customers_cleaned', engine, if_exists='replace', index=False)

] 3000

] agg = df.groupby('risk_segment_final', observed=False).agg(
    customers_count=('client_id', 'nunique'),
    avg_risk=('_risk_model_score', 'mean'),
    pct_churn_method1=('_loan_risk_churn_method1', 'mean')
).reset_index()

agg.to_sql('customers_risk_agg', engine, if_exists='replace', index=False)
```

Safe way to create a pipeline

```

]: from sqlalchemy import create_engine

user = 'root'
password = '9832'
host = '127.0.0.1'
port = 3306
database = 'banking_case'

engine = create_engine(f'mysql+mysqlconnector://{{user}}:{{password}}@{{host}}:{{port}}/{{database}}')

# 1) customer_cleaned
df.to_sql('customer_cleaned', engine, if_exists='replace', index=False)

# 2) customer_agg
agg = df.groupby(['risk_segment', 'location_id']).agg(
    customers_count = ('client_id', 'nunique'),
    avg_risk = ('risk_score', 'mean'),
    avg_income = ('estimated_income', 'mean')
).reset_index()

agg.to_sql('customer_agg', engine, if_exists='replace', index=False)

]: 2941

```

6. Power BI Dashboard Overview (6 Pages)

The dashboard is designed for **both executives and analysts**.

Page 1: Executive Overview

Purpose:

Provide a **high-level snapshot** of Customer loan risk.

Key Metrics:

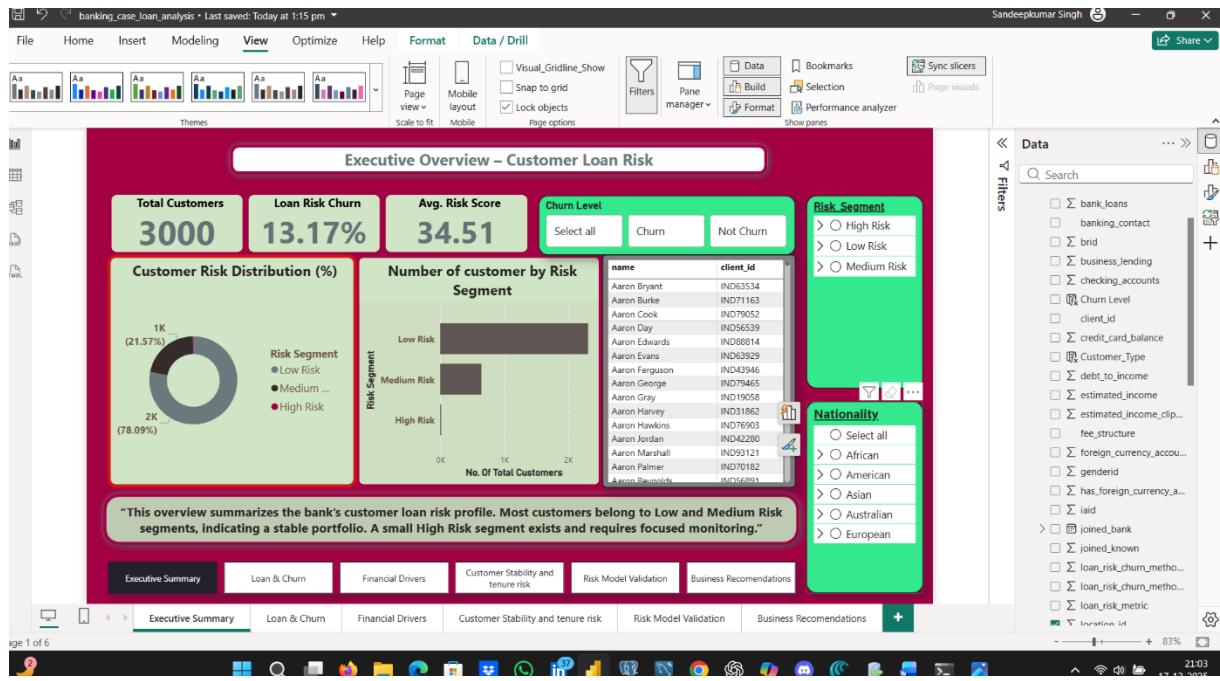
- Total customers
- Loan risk churn %
- Average risk score

Visuals:

- Donut chart: Customer distribution by risk segment
- Bar chart: Customer count by risk segment

Business Insight:

Most customers are low to medium risk, but a **small high-risk segment requires attention.**



Page 2: Loan Risk & Churn Analysis

Purpose:

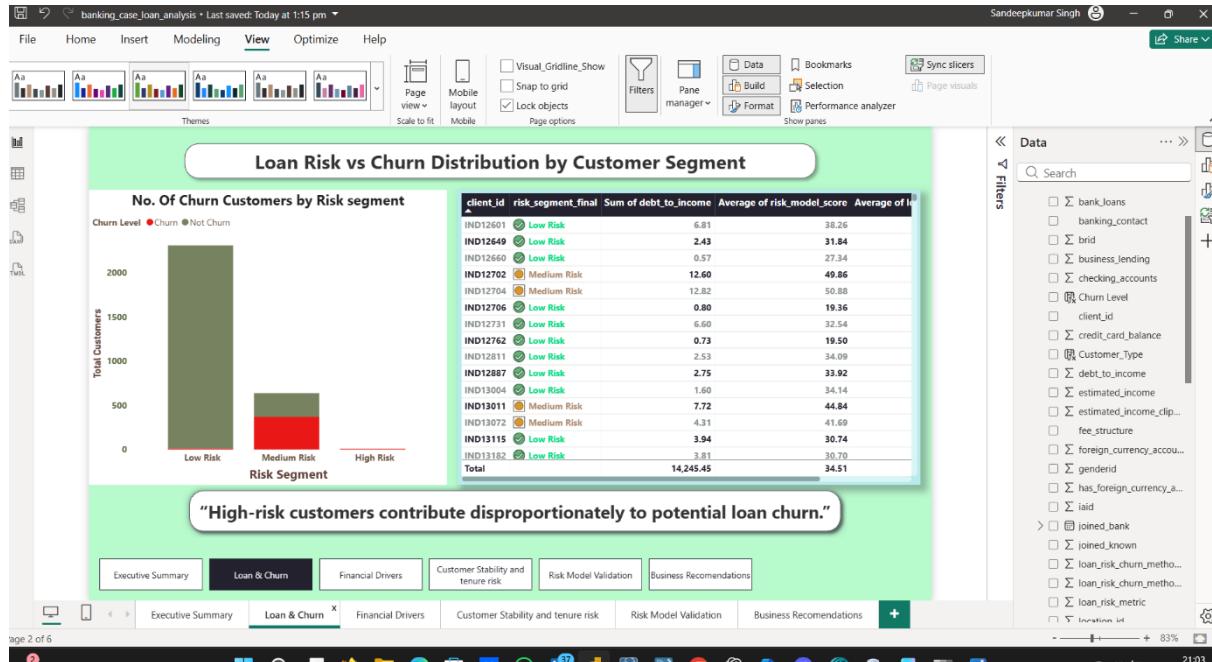
Understand how churn is distributed across risk segments.

Visuals:

- Stacked column chart:
Risk segment vs churned / non-churned customers

Insight:

Medium-risk customers show higher churn volume due to **larger population size**, while high-risk customers show **higher churn intensity**.



Page 3: Financial Drivers

Purpose:

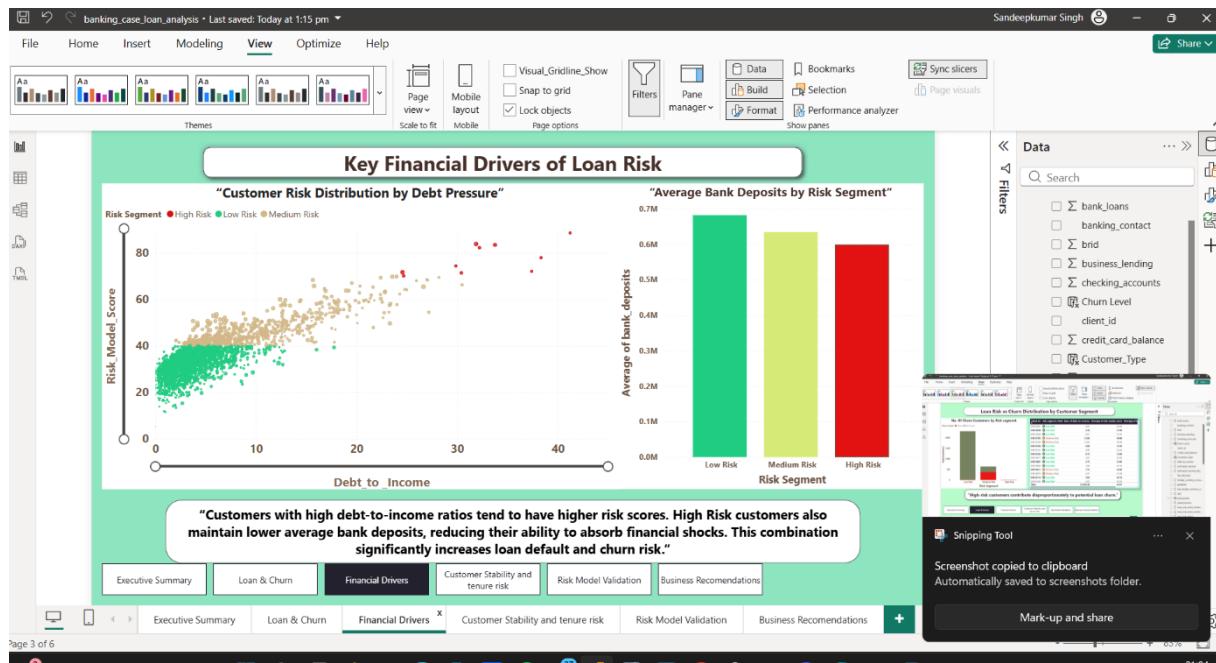
Identify financial variables driving risk.

Visuals:

1. Scatter plot
 - X: Debt-to-income
 - Y: Risk model score
 - Size: Total debt
 - Color: Risk segment
2. Bar chart
 - Average bank deposits by risk segment

Insight:

Customers with **high debt burden and low savings** consistently appear in higher risk segments.



Page 4: Customer Stability

Purpose:

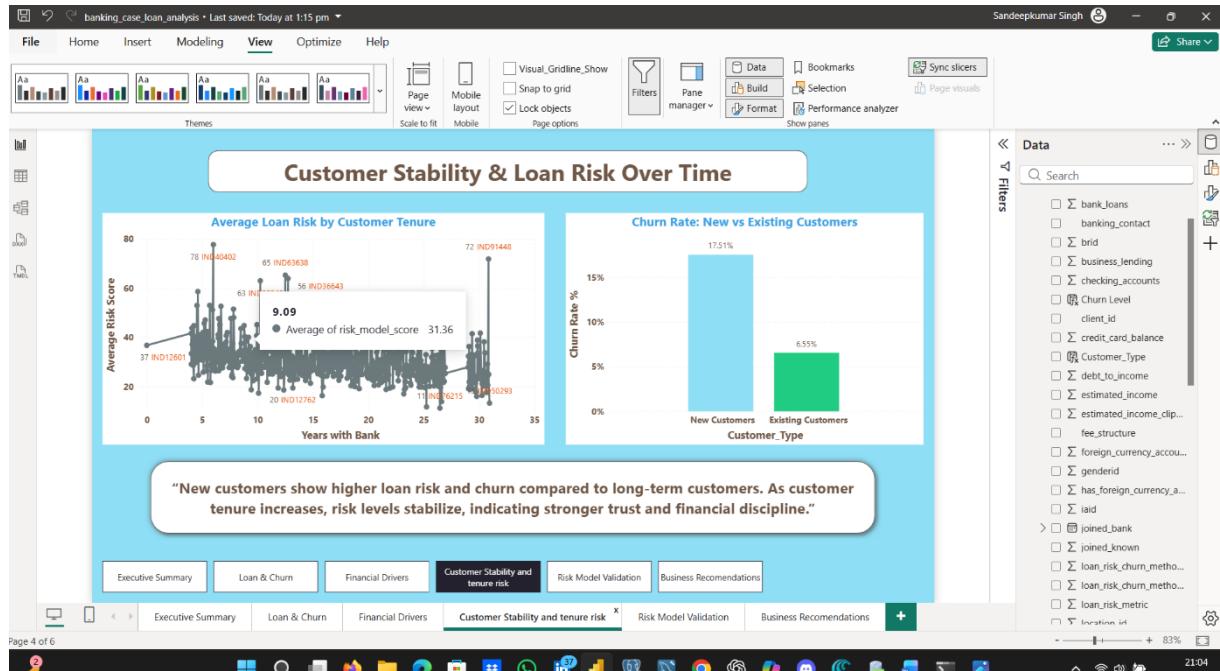
Analyze risk behavior over customer tenure.

Visuals:

1. Line chart
 - o Years with bank vs average risk score
2. Bar chart
 - o Churn %: New vs long-term customers

Insight:

New customers tend to have **higher risk and churn**, indicating the need for stronger early engagement.



Page 5: Risk Model Validation

Purpose:

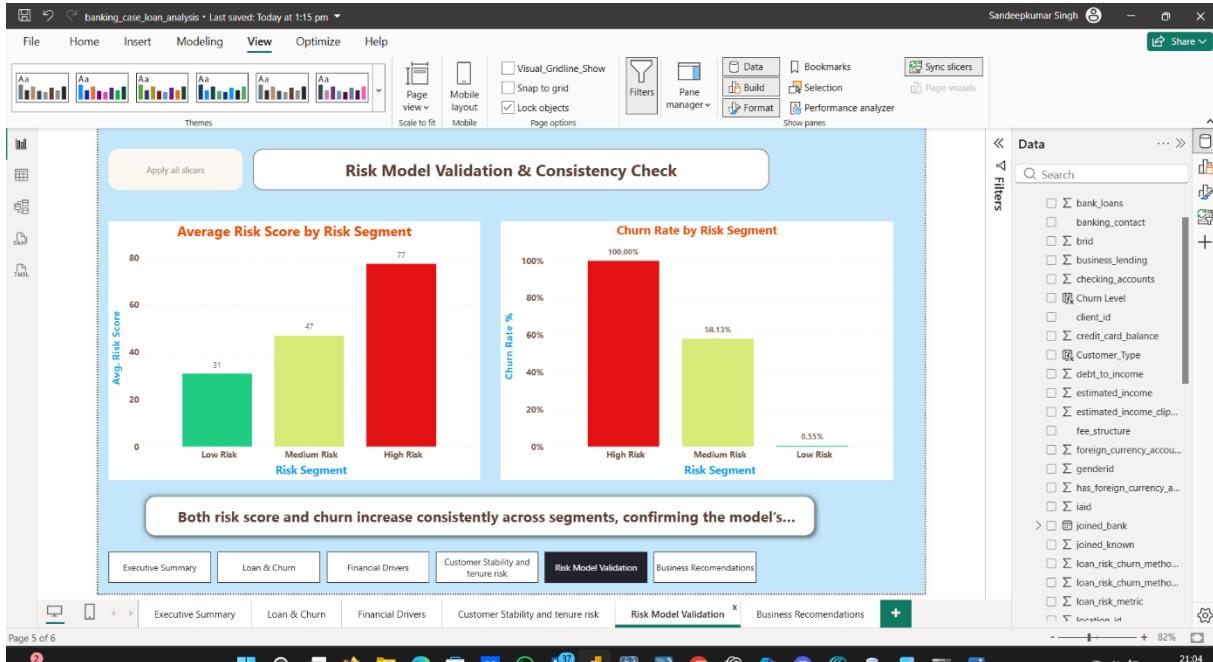
Validate whether the model segments customers meaningfully.

Visuals:

- Risk score distribution by segment
- Churn rate by risk segment
- Key Influencers visual (loan risk churn)

Insight:

Risk increases progressively from Low → Medium → High, confirming the model behaves logically.



Page 6: Business Recommendations

Purpose:

Convert insights into **actionable strategies**.

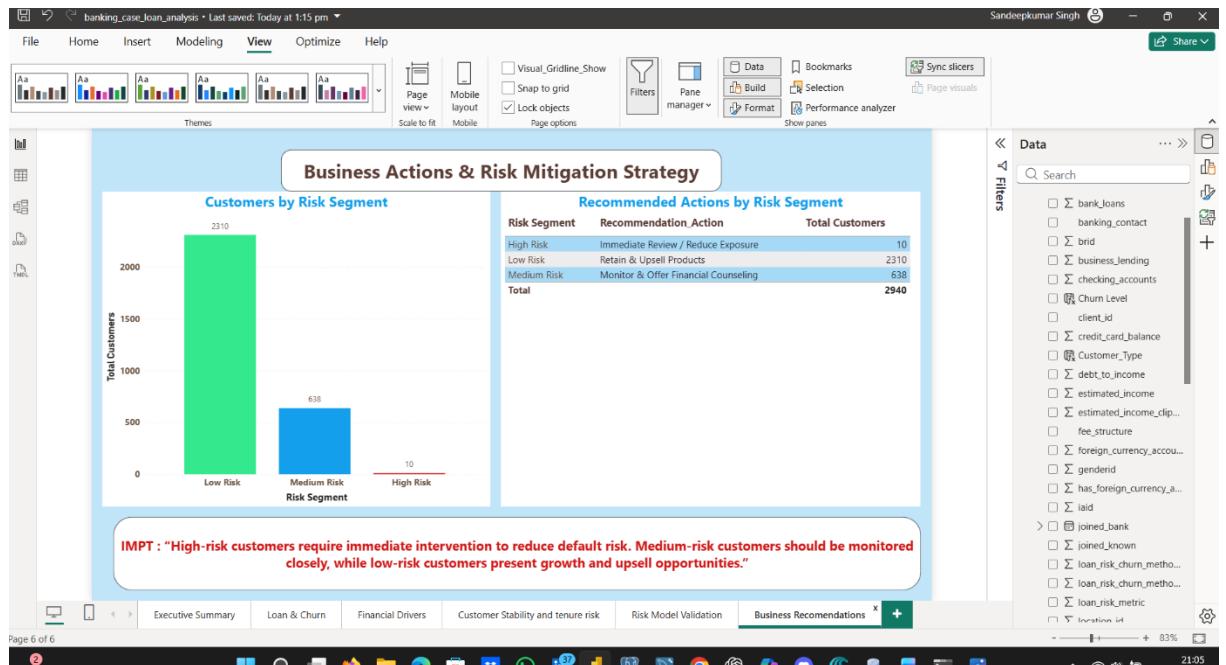
Example Actions:

- High Risk:** Reduce credit exposure, proactive monitoring
- Medium Risk:** Retention offers, financial guidance
- Low Risk:** Upsell opportunities

Visuals:

- Risk segment vs recommended action

- KPI summary



Screenshots of some Calculate Columns and Measures are created on this project using DAX functions

banking_case_loan_analysis • Last saved: Today at 1:15 pm

Sandeepkumar Singh

Table tools

Column tools

Customer_Type

IF(

Summarization: Don't summarize

Data category: Uncategorized

Sort by column

Data groups

Manage relationships

New column

Customer Type

Average Risk Score = AVERAGE('banking_case customers_cleaned'[risk_model_score])

Home table

Measures (2)

High Risk Customers

Home table

Measures (2)

High Risk Customers = CALCULATE([Total Customers], 'banking_case customers_cleaned'[risk_segment_final] = "High Risk")

File **Home** **Help** **Table tools** **Measure tools**

Customer_Type

Loan Risk Churn %

Home table

Measures (2)

Loan Risk Churn % = AVERAGE('banking_case customers_cleaned'[loan_risk_churn_method1])

banking_case_loan_analysis • Last saved: Today at 1:15 pm

Sandeepkumar Singh

Table tools

Customer_Type

Relationships

Calculations

banking_case customers_cleaned

client_id	name	age	location_id	joined_bank	banking_contact	nationality	occupation	fee_structure	loyalty_classification	estimated_income	state
IND26283	Joshua Hughes	58	95	Shawn Cook	European	Human Resources Assistant I	High	Jade	Platinum	98369.29	Florida
IND86703	Lillian Bell	41	11172	Stephen Payne	European	Biostatistician I	High	Platinum	71422.04	Illinois	
IND80929	Mary Austin	35	12694	Joshua Ryan	European	VP Marketing	High	Silver	263452.26	Illinois	
IND48405	Joe Lawrence	23	5800	Dennis Morris	European	Civil Engineer	High	Gold	122608.89	Illinois	
IND36608	Kimberly Schmidt	34	5278	Anthony Torre	European	Account Representative I	High	Silver	209330.69	Illinois	
IND90860	Anna Welch	51	43235	Shawn Wallace	European	Business Systems Development Analyst	High	Gold	131121.99	Illinois	
IND97689	Paula Ray	57	39015	Roger Alexander	European	Structural Engineer	High	Jade	107655.52	Illinois	
IND88778	Albert Bryant	17	785	James Castillo	European	Structural Analysis Engineer	High	Jade	309468.8	Illinois	
IND81583	Judith Matthews	64	23770	Nicholas Simmons	European	Media Manager I	High	Gold	258577.76	Illinois	
IND41611	Terry Bowman	34	6773	George Lewis	European	VP Accounting	High	Gold	216551.65	Illinois	
IND38441	Stephen Stewart	73	26699	Anthony Berry	European	Business Systems Development Analyst	High	Jade	204136.69	Illinois	
IND79955	Beverly Arnold	28	22689	Roger Alexander	European	Office Assistant II	High	Jade	132987.89	Illinois	
IND41067	Louis Ramirez	45	2764	Shawn Cook	European	Office Assistant I	High	Jade	242843.25	Illinois	
IND88784	Joshua Webb	34	501	Douglas Tucker	European	Web Developer I	High	Silver	280981.59	Illinois	
IND45638	Timothy Johnston	26	20753	Todd Roberts	European	Geologist II	High	Silver	131425.91	Illinois	
IND78162	Nicole Sanchez	75	17420	Ernest Rivera	European	Media Manager III	High	Gold	236297.5	Illinois	
IND17984	Gary Bell	18	31814	Nicholas Cunningham	European	Office Assistant IV	High	Jade	158424.74	Illinois	
IND34859	David Fernandez	38	10258	Nicholas Cunningham	European	Recruiter	High	Jade	171944.38	Illinois	
IND87992	Walter Matthews	36	31048	Anthony Torres	European	Geologist I	High	Jade	276516.44	Illinois	
IND39173	Eugene Austin	40	37837	Adam Hernandez	European	Data Coordinator	High	Silver	354296.33	Illinois	
IND35302	Jessica Black	78	35348	Joe Hanson	European	Product Engineer	High	Jade	209541	Illinois	
IND34318	Nancy Black	40	18785	Bruce Butler	European	Engineer IV	High	Jade	185421.5	Illinois	
IND79633	Samuel Gilbert	54	32020	Jesse Evans	European	Geologist I	High	Jade	154578.81	Illinois	
IND25477	Anthony Gardner	69	39052	Joshua Ryan	European	Health Coach IV	High	Jade	137063.36	Illinois	
IND44333	David Law	29	7401	Shawn Cook	European	Statistician I	High	Platinum	179214.85	Illinois	
IND76186	Russell Gutierrez	34	3748	Stephen Payne	European	Systems Administrator III	High	Silver	217568.28	Illinois	
IND89128	Nicholas Barnes	57	42936	Roger Alexander	European	Data Coordinator	High	Jade	33068.7	Illinois	

Table: banking_case customers_cleaned (3000 rows)

Created by :- Sandeep Kumar Singh

7. Business Impact

This solution enables the bank to:

- Proactively identify risky customers
 - Reduce loan default losses
 - Improve customer retention
 - Allocate resources efficiently
 - Support data-driven executive decisions
-

8. Tools Used

- Python (Pandas, NumPy, Scikit-learn)
 - Jupyter Notebook
 - MySQL
 - Power BI
 - SQLAlchemy
-

9. Conclusion

This project demonstrates strong skills in:

- Data cleaning and feature engineering
- Risk modeling and segmentation
- SQL-based data pipelines
- Executive-level dashboard design
- Business storytelling with data