Credit Risk Analysis for Loan Applicants — Full Project Report

Project: Credit Risk Analysis for Loan Applicants
Author: HARINATH MAKKA
Date: (add date of export)
Keywords: Credit risk, loan approval, SQL, Power BI, EDA, feature engineering, DTI, Income Band, BFSI

Part 1 — Executive Summary, Background & Data

1. Executive Summary

This project analyzes loan application records to identify the drivers of loan approval and to build an explainable workflow for credit risk insights using SQL and Power BI. The pipeline: raw CSV → SQL cleaning & transformation → cleaned CSV export → Power BI dashboards (visualization and stakeholder-ready metrics). Key outcomes:

- Clear influence of credit history, income bands and debt-to-income on approvals.
- A reproducible SQL pipeline for cleaning and feature engineering.
- Interactive Power BI dashboards summarizing KPIs and segment-level approval performance.
- Actionable recommendations for loan decision rules and monitoring.

---

2. Background & Motivation

Banks need robust, interpretable tools to decide on loan approvals and to quantify risk. This project shows how standard BFSI datasets can be cleansed and analyzed with SQL and presented to decision makers using Power BI. The objective is not necessarily production ML scoring but demonstrable data-driven insights that help credit officers make better decisions.

---

3. Project Objectives

- Clean and preprocess loan application data using SQL (MySQL Workbench).
- Create derived features relevant to credit risk (Total_Income, DTI, Income_Band).
- Produce dashboards showing approval rates across segments (income, credit history, property area, dependents).
- Provide reproducible SQL scripts and a Power BI exported PDF for documentation.

---

4. Data Description

Files in repository

- `Data/loan.csv` — raw dataset of loan applicants
- `Data/loan_cleaned_data.csv` — cleaned & engineered dataset used for visualization
- `SQL_Queries/project_credit_risk_analysis.sql` — SQL script with cleaning & feature creation queries
- `PowerBI/credit_risk_analysis_HM.pdf` — exported dashboard
- `Documentation/Credit_Risk_Analysis_Project_Documentation.doc` — longer project doc

Common fields (typical)

- `Loan_ID` — unique application id
- `Gender` — Male/Female
- `Married` — Yes/No
- `Dependents` — number of dependents (0,1,2,3+)
- `Education` — Graduate / Not Graduate
- `Self_Employed` — Yes/No
- `ApplicantIncome` — primary income
- `CoapplicantIncome` — co-applicant income
- `LoanAmount` — loan amount (in thousands)
- `Loan_Amount_Term` — term in months
- `Credit_History` — 1 (meets guidelines) or 0 (not)
- `Property_Area` — Urban / Semiurban / Rural
- `Loan_Status` — Y (Approved) / N (Rejected)

(If your column names differ, use those names in SQL; below I use the typical names above.)

---

5. Data Quality Issues Observed (before cleaning)

- Missing values in `Credit_History`, `LoanAmount`, `Loan_Amount_Term`, `Gender`, `Dependents`.
- `Dependents` sometimes contains string `'3+'`.
- Numeric columns stored as text or with inconsistent formatting.
- Need to calculate `Total_Income` and normalize `LoanAmount` scale.

- Outliers in `ApplicantIncome` and `LoanAmount`.

---

6. SQL Cleaning & Preprocessing Approach (summary)

All SQL transformations were performed in MySQL Workbench using the `project_credit_risk_analysis.sql` script. General steps:

1. Load CSV into staging table `loan_raw`.
2. Trim and standardize categorical fields (`TRIM`, `UPPER`).
3. Replace `Dependents = '3+'` with integer `3`.
4. Fill missing `Credit_History` with `0` or `NULL` depending on analysis choice (we kept NULL for missing indicator and created binary flag).
5. Fill missing `LoanAmount` using median per `Education` or global median.
6. Create derived features:
    - `Total_Income = ApplicantIncome + CoapplicantIncome`
    - `LoanAmount_log = LOG(LoanAmount)` (for distribution)
    - `DTI = (LoanAmount*1000) / Total_Income` or another consistent measure
    - `Income_Band` buckets (Low/Medium/High) based on quantiles
    - `DTI_Band` buckets (Low/Medium/High)
7. Output cleaned table `loan_cleaned`.

---

7. Example SQL: Key cleaning & feature queries

Below are representative SQL snippets you can paste into `project_credit_risk_analysis.sql` or run step-by-step.

```
-- 1. Create staging table (adjust types as needed)
CREATE TABLE IF NOT EXISTS loan_raw (
  Loan_ID VARCHAR(20) PRIMARY KEY,
  Gender VARCHAR(10),
  Married VARCHAR(5),
  Dependents VARCHAR(5),
  Education VARCHAR(20),
  Self_Employed VARCHAR(5),
  ApplicantIncome INT,
  CoapplicantIncome INT,
  LoanAmount FLOAT,
  Loan_Amount_Term FLOAT,
  Credit_History INT,
  Property_Area VARCHAR(20),
  Loan_Status CHAR(1)
);

-- LOAD DATA INFILE ... (or import via Workbench GUI)
```

```
-- Example: Import CSV into loan_raw using Workbench Data Import wizard.

-- 2. Standardize Dependents and cast to INT
CREATE TABLE IF NOT EXISTS loan_step1 AS
SELECT
  Loan_ID,
  TRIM(Gender) AS Gender,
  TRIM(Married) AS Married,
  CASE WHEN TRIM(Dependents) = '3+' THEN 3
       WHEN TRIM(Dependents) = '' THEN NULL
       ELSE CAST(TRIM(Dependents) AS UNSIGNED) END AS Dependents,
  TRIM(Education) AS Education,
  TRIM(Self_Employed) AS Self_Employed,
  NULLIF(ApplicantIncome, '')+0 AS ApplicantIncome,
  NULLIF(CoapplicantIncome, '')+0 AS CoapplicantIncome,
  NULLIF(LoanAmount, '')+0 AS LoanAmount,
  NULLIF(Loan_Amount_Term, '')+0 AS Loan_Amount_Term,
  NULLIF(Credit_History, '')+0 AS Credit_History,
  TRIM(Property_Area) AS Property_Area,
  Loan_Status
FROM loan_raw;

-- 3. Impute LoanAmount with median by Education
-- Compute medians (MySQL doesn't have a median built-in; use approximate
method)
CREATE TABLE IF NOT EXISTS loan_medians AS
SELECT Education,
       (SELECT AVG(x.LoanAmount) FROM
           (SELECT LoanAmount FROM loan_step1 ls WHERE ls.Education =
l.Education AND LoanAmount IS NOT NULL
            ORDER BY LoanAmount LIMIT 2 OFFSET (SELECT FLOOR(COUNT(*)/2) FROM
loan_step1 ls2 WHERE ls2.Education = l.Education)
           ) x) AS median_loan
FROM (SELECT DISTINCT Education FROM loan_step1) l;

-- For simplicity, we'll use global median if education-based median is null
SET @global_median := (SELECT AVG(x.LoanAmount) FROM
                        (SELECT LoanAmount FROM loan_step1 WHERE LoanAmount IS
NOT NULL
                         ORDER BY LoanAmount LIMIT 2 OFFSET (SELECT
FLOOR(COUNT(*)/2) FROM loan_step1 WHERE LoanAmount IS NOT NULL)
                        ) x);

-- Create cleaned table with derived features
CREATE TABLE IF NOT EXISTS loan_cleaned AS
SELECT
  ls.Loan_ID,
  ls.Gender,
  ls.Married,
  ls.Dependents,
  ls.Education,
  ls.Self_Employed,
  ls.ApplicantIncome,
  ls.CoapplicantIncome,
  ls.ApplicantIncome + ls.CoapplicantIncome AS Total_Income,
  COALESCE(ls.LoanAmount, lm.median_loan, @global_median) AS LoanAmount,
  ls.Loan_Amount_Term,
  ls.Credit_History,
  ls.Property_Area,
  ls.Loan_Status,
  -- derived columns
  CASE
    WHEN ls.ApplicantIncome + ls.CoapplicantIncome < 2500 THEN 'Low'
```

```
    WHEN ls.ApplicantIncome + ls.CoapplicantIncome BETWEEN 2500 AND 6000 THEN
'Medium'
    ELSE 'High' END AS Income_Band,
  ROUND( (COALESCE(ls.LoanAmount, lm.median_loan, @global_median)*1000) /
NULLIF(ls.ApplicantIncome + ls.CoapplicantIncome,0), 2) AS DTI
FROM loan_step1 ls
LEFT JOIN loan_medians lm ON lm.Education = ls.Education;
```

> Note: The median computations above are illustrative. In practice use a simpler imputation (global median or group median computed in a small helper query) or compute median using stored procedures or export to Python/pandas for robust medians.

---

8. Export cleaned data

After cleaning, export `loan_cleaned` to CSV:

```
SELECT * FROM loan_cleaned;
-- Use Workbench export wizard: right-click result -> Export Resultset -> CSV
```

Name exported file `Data/loan_cleaned_data.csv`.

Part 2 — Exploratory Analysis, Visualization, Insights & Recommendations

9. Exploratory Data Analysis (SQL queries & findings)

Use SQL to compute summary statistics and segment-wise approval rates.

A. Overall approval rate

```
SELECT
  COUNT(*) as total_applications,
  SUM(CASE WHEN Loan_Status = 'Y' THEN 1 ELSE 0 END) as approved,
  ROUND(100.0 * SUM(Loan_Status = 'Y') / COUNT(*), 2) as approval_rate_pct
FROM loan_cleaned;
```

B. Approval by Credit History

```
SELECT Credit_History,
       COUNT(*) AS num,
       SUM(Loan_Status = 'Y') AS approved,
       ROUND(100.0 * SUM(Loan_Status = 'Y') / COUNT(*), 2) AS approval_pct
FROM loan_cleaned
GROUP BY Credit_History
ORDER BY Credit_History DESC;
```

Finding: Applicants with `Credit_History = 1` typically have much higher approval rates.

C. Approval by Income Band

```
SELECT Income_Band,
       COUNT(*) AS num,
       SUM(Loan_Status = 'Y') AS approved,
```

```
       ROUND(100.0 * SUM(Loan_Status = 'Y') / COUNT(*), 2) AS approval_pct
FROM loan_cleaned
GROUP BY Income_Band
ORDER BY FIELD(Income_Band,'Low','Medium','High');
```

Finding: Higher income bands show higher approval percentages.

D. Approval by DTI (chunked into bands)

```
SELECT CASE
         WHEN DTI < 20 THEN 'Low'
         WHEN DTI BETWEEN 20 AND 40 THEN 'Medium'
         ELSE 'High' END AS DTI_BAND,
       COUNT(*) AS num,
       SUM(Loan_Status = 'Y') AS approved,
       ROUND(100.0 * SUM(Loan_Status = 'Y') / COUNT(*), 2) AS approval_pct
FROM loan_cleaned
GROUP BY DTI_BAND;
```

Finding: Lower DTI bands correspond with better approval rates.

E. Cross-segmentation (Credit History + Income Band)

```
SELECT Credit_History, Income_Band,
       COUNT(*) AS num,
       SUM(Loan_Status = 'Y') AS approved,
       ROUND(100.0 * SUM(Loan_Status = 'Y') / COUNT(*), 2) AS approval_pct
FROM loan_cleaned
GROUP BY Credit_History, Income_Band
ORDER BY Credit_History DESC, FIELD(Income_Band,'Low','Medium','High');
```

This highlights that credit history plus income is highly predictive.

---

10. Power BI Dashboard Design (Suggested Pages)

The `credit_risk_analysis_HM.pdf` exported in your repo follows this layout. To reproduce or edit in Power BI, create the following pages:

Page 1 — Executive Summary (KPI dashboard)

- KPIs: Total Applications, Approval Rate (%), Average Loan Amount, Median DTI
- Small trend line of approvals over time (if time data exists)
- Top-line conclusion

Page 2 — Approval Drivers

- Bar chart: Approval rate by Credit History
- Stacked bar: Approval by Income_Band and Property_Area
- Heatmap: Approval rate by Dependents vs Income_Band

Page 3 — Credit Risk Segmentation

- Boxplot or violin (or aggregated bars) of LoanAmount by Loan_Status
- DTI distribution histogram split by Loan_Status

- Scatter: Total_Income vs LoanAmount sized by approval probability

Page 4 — Operational Tables

- Searchable table of applicants and fields (Loan_ID, Income, Credit_History, DTI, Status)
- Filters for quick segmentation (Gender, Education, Property_Area)

Page 5 — Recommendations

- Rules-based suggestions (see below)
- Monitoring metrics & alerts suggestions

---

11. Models & Scoring (optional)

This project focuses on SQL & BI, but if you want to move to predictive modeling:

Suggested models (explainable, suitable for BFSI):

- Logistic Regression (baseline, interpretable)
- Random Forest (robust, handles nonlinearity)
- XGBoost (best performance often; add SHAP for explainability)

Suggested features

- Total_Income (numeric)
- DTI (numeric)
- Credit_History (binary)
- Income_Band (categorical)
- Dependents (numeric)
- LoanAmount (numeric or log-transformed)
- Property_Area (categorical)
- Education / Self_Employed (categorical)
- Interaction terms (e.g., Credit_History * Income_Band)

Evaluation metrics

- F1-score (for class imbalance)
- ROC-AUC (for ranking)
- Precision@k for top decision cutoffs
- Confusion matrix with business-based cost matrix

Explainability

- Use coefficient signs & p-values (LogReg)
- Use SHAP values for tree models to explain each prediction

---

12. Key Insights & Findings (summary)

- **Credit history is the strongest single driver** — applicants with positive credit history `=1` have significantly higher approval rates.
- **Income matters** — higher total household income strongly correlates with approvals.
- **DTI is critical** — higher debt to income is associated with rejections.
- **Property area & dependents** affect but are secondary to credit and income.
- **Simple rules** (Credit_History=1 and DTI < threshold) capture a large share of approvals with high precision.

---

13. Business Recommendations

1. **Use Credit History as primary screening**: auto-accept candidates with `Credit_History=1`, `DTI < 25%`, and `Income_Band = High` (subject to manual verification).
2. **Manual review rules**: applicants with missing `Credit_History` but otherwise strong income and low DTI should go for manual underwriting.
3. **Monitoring dashboard**: track monthly approval rate, average DTI of approved loans, and rate of approval declines by credit officer.
4. **Data collection improvement**: enforce mandatory credit-history checks, standardize dependents entry, and store time fields to enable trend analysis.
5. **Pilot predictive scoring**: run a logistic regression model in production for a pilot (A/B test) before full automation.

---

14. Limitations & Future Work

- **Modeling not part of current deliverable**: this report focuses on SQL & BI insights; predictive models should be trained on a hold-out set and backtested.
- **Data limitations**: missing values and small sample sizes in subgroups can bias results.
- **Temporal validation**: if dataset spans multiple years, time-based validation should be used.
- **Data leakage**: ensure no feature used in modeling is only available post-approval.
- **Regulatory considerations**: fairness and explainability need to be tested for bias.

---

15. Reproducibility & Files in Repo

- `Data/loan.csv` — raw data (source)

- `Data/loan_cleaned_data.csv` — cleaned dataset used for Power BI
- `SQL_Queries/project_credit_risk_analysis.sql` — SQL script used for cleaning and feature engineering (place queries from section 7 inside)
- `PowerBI/credit_risk_analysis_HM.pdf` — exported dashboard for review
- `Documentation/Credit_Risk_Analysis_Project_Documentation.doc` — supporting document (if present)

---

16. Appendix A — Useful SQL snippets (select & aggregation)

Top 10 highest DTI rejected applicants

```
SELECT Loan_ID, Total_Income, LoanAmount, DTI, Loan_Status
FROM loan_cleaned
WHERE Loan_Status = 'N'
ORDER BY DTI DESC
LIMIT 10;
```

Pivot-style approvals by Property Area

```
SELECT Property_Area,
       SUM(Loan_Status='Y') AS approved,
       SUM(Loan_Status='N') AS rejected,
       ROUND(100.0 * SUM(Loan_Status='Y') / COUNT(*), 2) AS approval_pct
FROM loan_cleaned
GROUP BY Property_Area;
```

---

17. Appendix B — Power BI Tips (quick)

- Use slicers for `Credit_History`, `Income_Band`, `Property_Area`.
- Create calculated columns: `Total_Income`, `DTI`, `Income_Band`.
- Use conditional formatting on approval_pct to highlight risk segments.
- Export to PDF: File → Export → PDF. Place file in `/PowerBI/credit_risk_analysis_HM.pdf`.

---

18. Conclusion

This Credit Risk Analysis project demonstrates how SQL and Power BI can be used together to clean, analyze and communicate meaningful credit decisions for loan applicants. The code, cleaned dataset, and the dashboard provide a reproducible pipeline and actionable business insights that can support underwriting decisions and monitoring.

End of Report
