**LOAN DEFAULT AND CUSTOMER RISK ANALYSIS**





**Sri Harsha Vardhan Sarma Panchagnula Submitted By:**

**Mentor** A Gayathri

**Table of Contents**

Problem Statement................................................................................

Software Requirements Specification....................................................

Architecture...........................................................................................

Dataset Schema…………………………………………………………………………………

Entity Relation Ship Diagram……………………………………………………………..

Project Implementation Steps...............................................................

**Problem Statement**

This project focuses on developing a data pipeline to predict loan defaults and evaluate customer risk profiles using historical banking data. It employs ADF or Five Tran for data ingestion, Python for feature engineering, Snowflake for data storage and transformation, and Power BI for visualization, enabling informed, risk-aware decision-making.

**Software Requirements Specification**

The development of this project involves the integration of multiple cloud-based and analytical tools to enable efficient data ingestion, transformation, storage, and visualization. The following software components were used throughout the pipeline:

**Azure Datalake Storage Gen2**



Figure 1

*Figure 1: Azure Data Lake Storage Gen2 – Hierarchical and high-throughput storage.*

**Azure Data Lake Storage Gen2:**  
Acts as the primary landing zone for raw and semi-structured data. It offers secure and scalable data storage for further processing.

**Azure Data Factory**

Figure 2

*Figure 2: Azure Data Factory – Data orchestration service****.***

**Azure Data Factory (ADF):**  
ADF is used to orchestrate and automate the data ingestion workflows from source systems into Azure Data Lake Storage. It enables scheduled and scalable extraction of raw banking data.

**Snowflake**

Figure 3

*Figure 3: Snowflake – Cloud data warehousing platform*.

Serves as the centralized cloud data warehouse for storing curated datasets. It supports advanced SQL-based transformations and data modeling for downstream analytics.

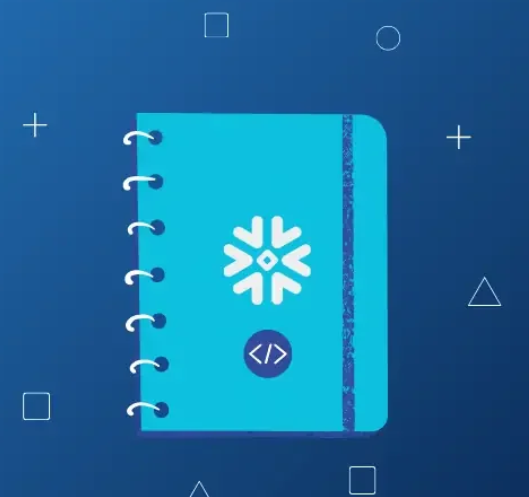
**Snowflake Notebook**

Figure 4

*Figure 4: Snowflake Notebook – Interactive SQL workspace.*

**Snowflake Notebooks:**  
Used to perform SQL scripting and transformation logic interactively. It facilitates efficient development and testing of data workflows within the Snowflake environment.

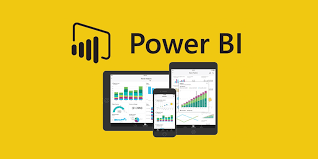
**Power BI**

Figure 5

*Figure 5: Power BI – Business intelligence and reporting tool*.

Employed for building interactive dashboards and visual reports that enable business users to explore customer risk profiles and monitor default trends.

**Architecture**

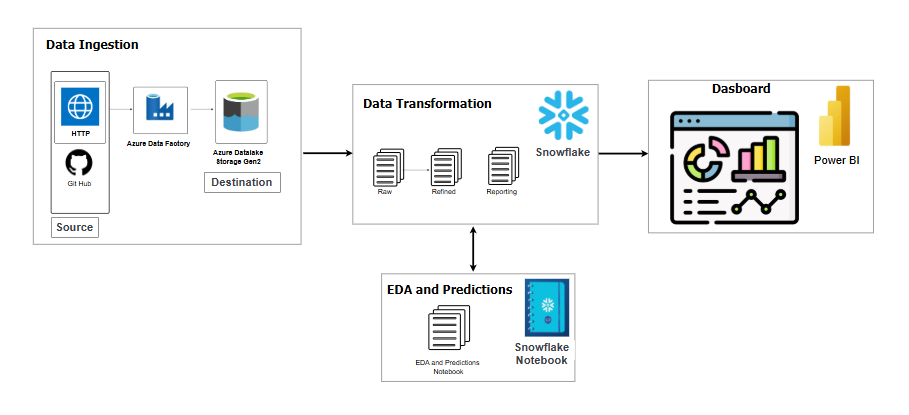


Figure 6

The architecture of the loan default prediction and customer risk analysis system is designed as a modular and scalable pipeline, facilitating seamless integration of data engineering, transformation, analysis, and visualization processes.

The architecture consists of the following core components:

1. **Data Ingestion:**
   * Data is sourced from external systems such as HTTP APIs and GitHub repositories.
   * **Azure Data Factory** orchestrates and automates the extraction of data into **Azure Data Lake Storage Gen2**, which acts as the staging layer for raw data storage.
2. **Data Transformation:**
   * Raw data stored in the data lake is moved into **Snowflake**, where it undergoes structured transformation across three layers:
     + **Raw Layer** – Stores ingested data in its original format.
     + **Refined Layer** – Contains cleaned and standardized datasets.
     + **Reporting Layer** – Includes aggregated data suitable for analytics and reporting.
3. **Exploratory Data Analysis (EDA) and Predictions:**
   * Data from Snowflake is further explored using **Snowflake Notebooks** for EDA and rule-based predictions.
   * Analytical logic is implemented within the notebook to generate insights such as risk scores and default flags.
4. **Dashboard and Reporting:**
   * Transformed data is visualized using **Power BI**, providing interactive dashboards for monitoring customer risk profiles, default trends, and loan performance metrics.

*Figure 6: Project Architecture – End-to-end data pipeline from ingestion to visualization.*

This architecture supports modular development, scalability, and maintainability, ensuring efficient handling of large datasets and timely delivery of insights for business decision-making.

**Dataset Schema**

**Customers.csv**

| Customer\_Id | First\_name | Last\_name | Email | Phone\_Number |Age | Gender | Marital\_Status | Address | Employment\_Status |Income| |Pan\_Number|Account\_Number|Cibil\_Score|Number\_Of\_Dependents |

**Loans.csv**

| Loan\_id | Customer\_id | Loan\_Amount | Loan\_Type |Rate\_Of\_Interest |EMI | Term\_Months | Loan\_Status | Issue\_Date |

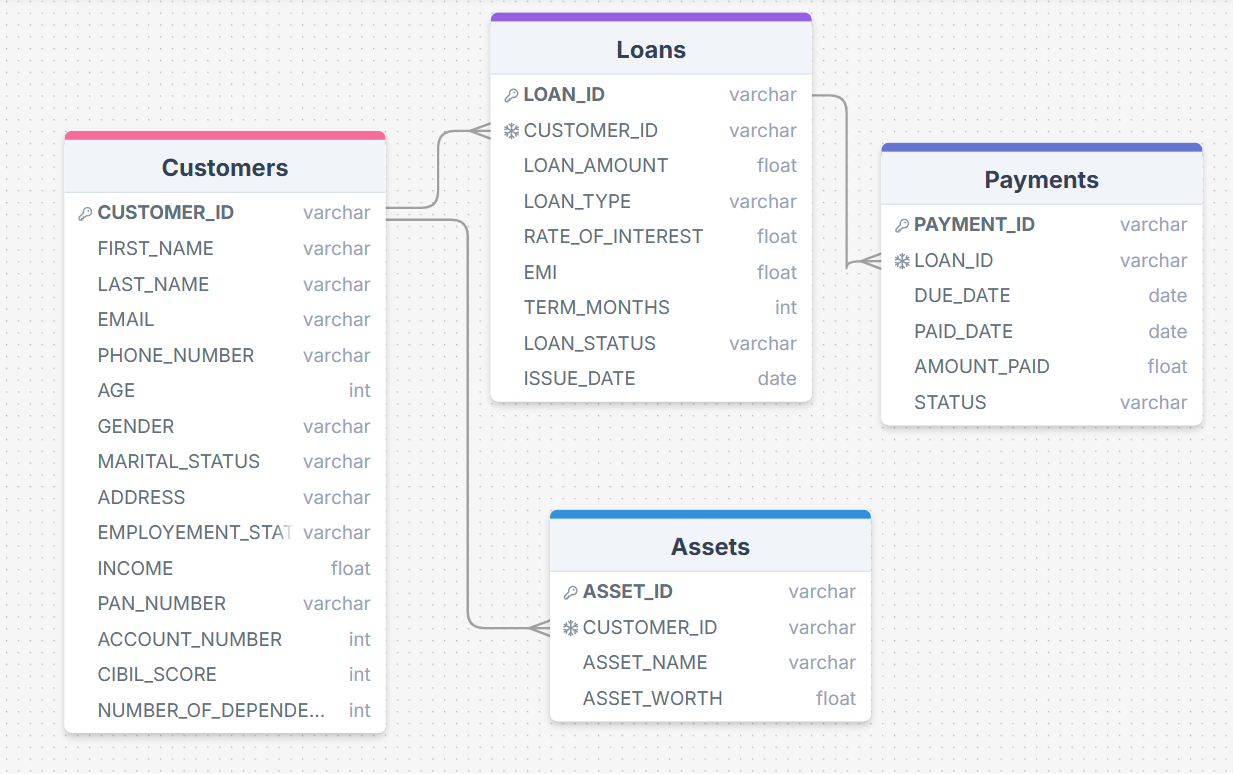
**Payments.csv**

| Payment\_Id | Loan\_Id | Due\_Date | Amount\_Paid | status (OnTime/Late) |

**Assets.csv**

| Asset\_Id | Customer\_Id | Asset\_Name | Asset\_Worth |

**Entity Relationship Diagram**

****

**ER Diagram**

The Entity-Relationship (ER) diagram provides a conceptual representation of the data model used in the loan default prediction system. It illustrates how key entities are related and how the data is structured for storage, transformation, and analysis.

**Key Entities and Relationships:**

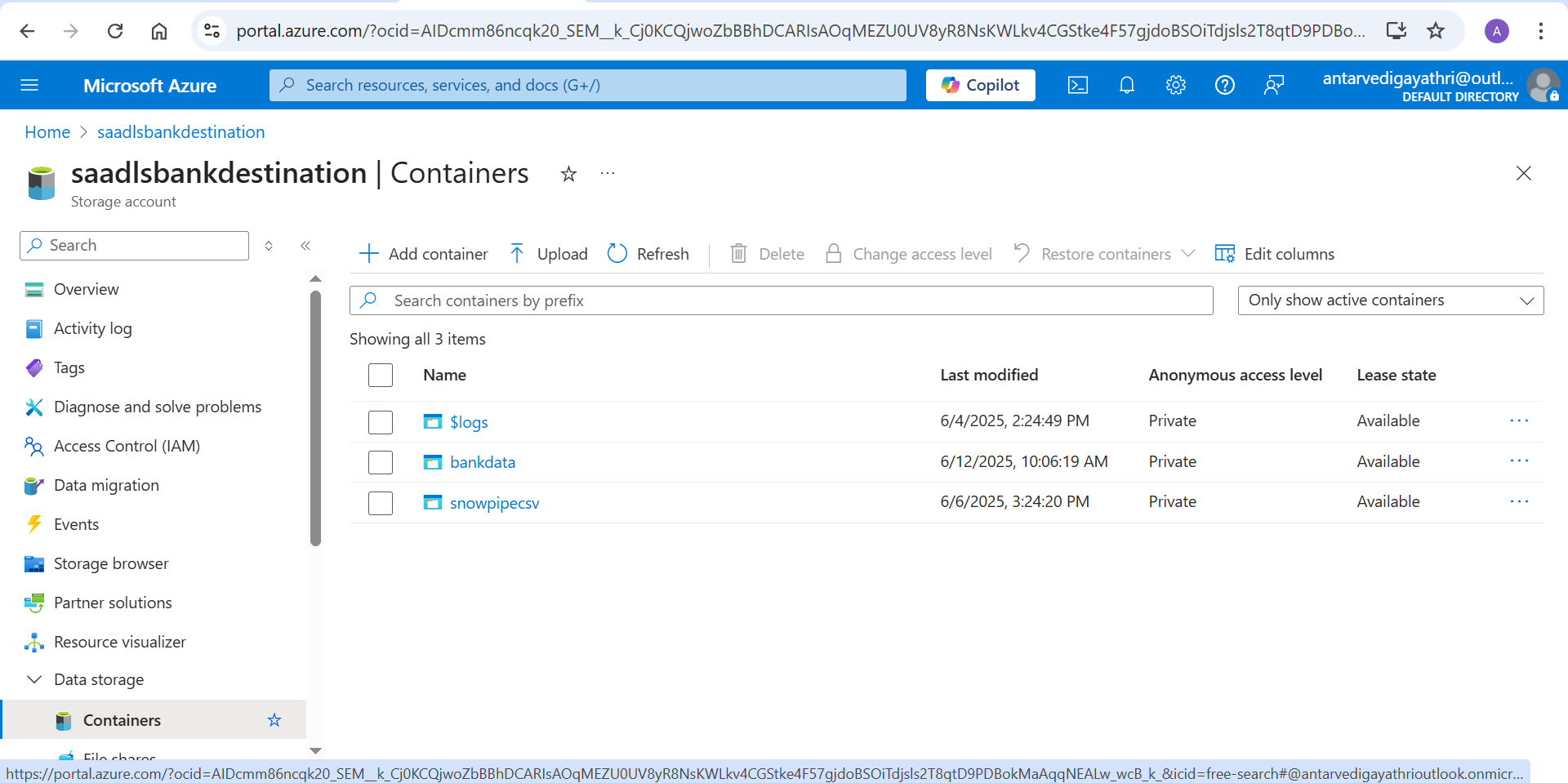
1. **Customers**
   * Contains demographic and financial details of bank customers.
   * CUSTOMER\_ID serves as the primary key and is referenced in both the *Loans* and *Assets* tables.
   * Attributes include name, contact details, employment status, income, PAN number, CIBIL score, etc.
2. **Loans**
   * Represents the loans availed by customers.
   * Linked to *Customers* via CUSTOMER\_ID.
   * Includes attributes such as LOAN\_ID, LOAN\_AMOUNT, LOAN\_TYPE, RATE\_OF\_INTEREST, EMI, TERM\_MONTHS, ISSUE\_DATE, and loan STATUS.
3. **Payments**
   * Tracks payment history for each loan.
   * Connected to *Loans* through LOAN\_ID.
   * Key attributes include DUE\_DATE, PAID\_DATE, AMOUNT\_PAID, and STATUS, which are critical for identifying payment defaults.
4. **Assets**
   * Captures details about customer-owned assets.
   * Related to *Customers* using CUSTOMER\_ID.
   * Attributes include ASSET\_NAME and ASSET\_WORTH.

*Figure 7: ER Diagram – Depiction of relationships between customers, loans, payments, and assets.*

This data model supports effective normalization and referential integrity, ensuring a structured foundation for analytics, predictions, and reporting within the project.

**Project Implementation Steps**

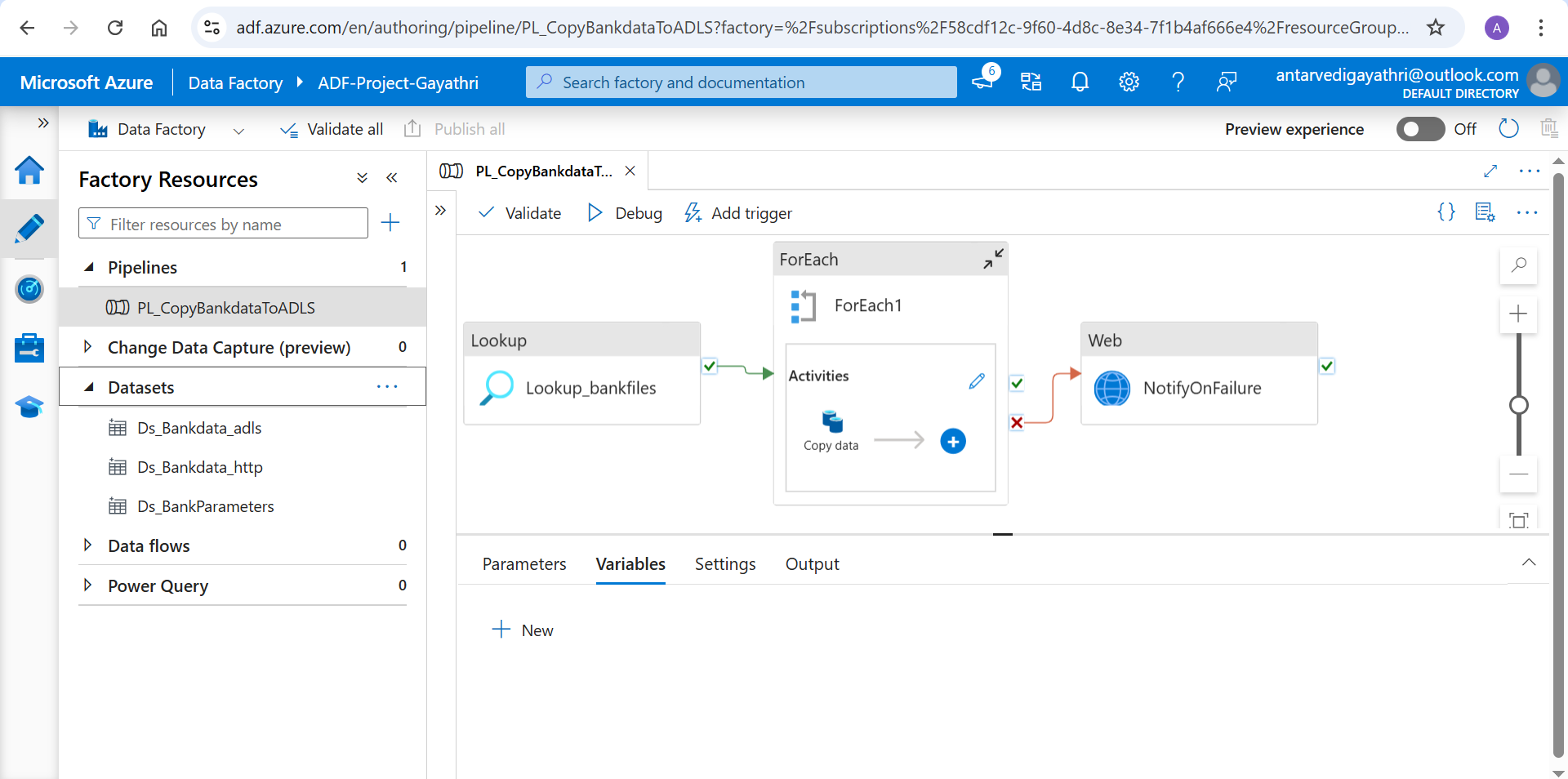
**Step 1: - Set Up Azure Data Lake Storage (ADLS)**



**Fig 1: - Creation of Storage Account**

1. **Login to Azure Portal:**  
   Navigate to [https://portal.azure.com](https://portal.azure.com" \t "_new) and sign in with your credentials.
2. **Create a Storage Account:**
   * Click **“Create a Resource”** from the left-hand menu.
   * Search for **“Storage Account”** and select the result named *Storage account – blob, file, table, queue*.
   * Click **“Create”** and fill in the following:
     + **Subscription**: Azure for Students
     + **Resource Group**: demo
     + **Storage Account Name**: saadlsbankdestination
     + **Region**: East US
     + Choose standard performance settings and defaults.
3. **Configure Advanced Settings (Optional):**
   * Enable hierarchical namespace to use ADLS Gen2.
   * Configure network rules and replication settings as needed.
4. **Create a Container:**  
   Once the storage account is deployed:
   * Go to the account → Containers → Click **+ Container**
   * Name the container **bankdata**.
   * Set public access level to *Private (no anonymous access)*.

**Step 2: Build the Azure Data Factory Pipeline**

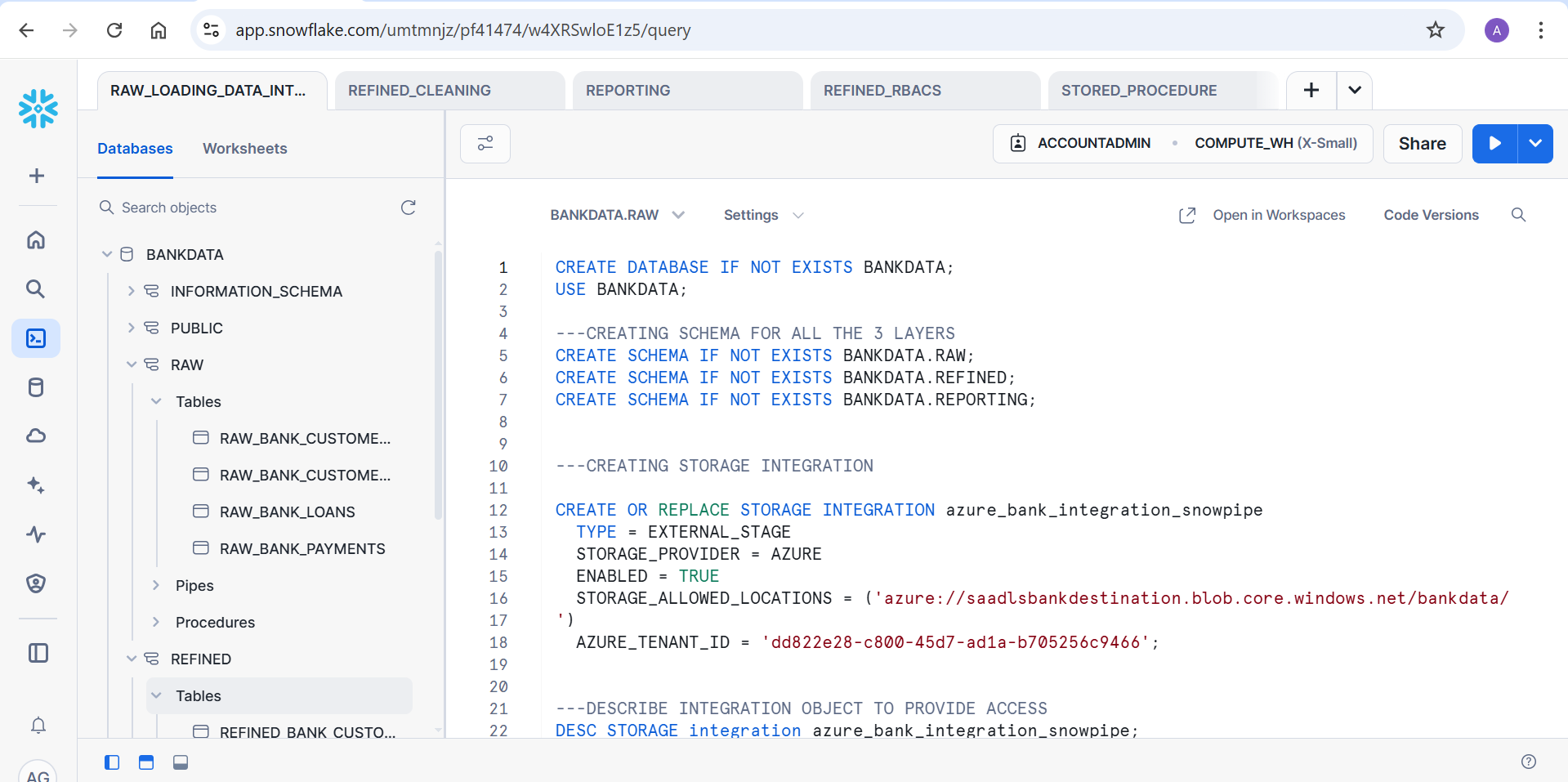


**Fig 2: - Creation of Pipeline**

1. **Create an Azure Data Factory Instance:**
   * In the portal, search for **“Data Factory”** and click **“Create”**.
   * Use the name **Azure-Project-Gayathri** and link it to the existing resource group and region.
2. **Create Linked Services:**  
   These define source and destination connections.
   * **ls\_adls\_bankdata**: Connects to the ADLS container bankdata.
   * **ls\_http\_bankapi**: Connects to external data source (GitHub in this case) via HTTP.
3. **Create Datasets:**  
   Define the structure of incoming and outgoing data:
   * **Ds\_bankdata\_http** – for reading data from HTTP source.
   * **Ds\_bankdata\_adls** – for writing data to ADLS.
   * **Ds\_bankdata\_parameters** – parameterized dataset used in lookup.
4. **Create Parameter JSON File:**  
   Prepare a JSON file named bankdata\_parameters.json with keys like:
   * p\_rel\_url
   * p\_sink\_folder
   * p\_sink\_file\_name  
     Upload this file to the bankdata container in ADLS.
5. **Configure Lookup Activity:**
   * Use the Lookup activity to read bankdata\_parameters.json.
   * This enables the pipeline to loop through multiple file URLs dynamically.
6. **Add ForEach Activity:**
   * Inside the ForEach loop, pass each set of parameters.
   * Place a **Copy Data** activity within the loop to copy from HTTP to ADLS.
7. **Add Web Activity (for Failure Notification):**
   * Connect the pipeline failure path to a **Web Activity**.
   * Configure it to send a WhatsApp notification using a webhook (e.g., via Twilio or Logic Apps) when the pipeline fails.

Now the Files will be copied to the ADLS bankdata container.

**Step 3: - Set Up Snowflake and Perform Transformations**



**Fig 3: - Uploading dataset into Container**

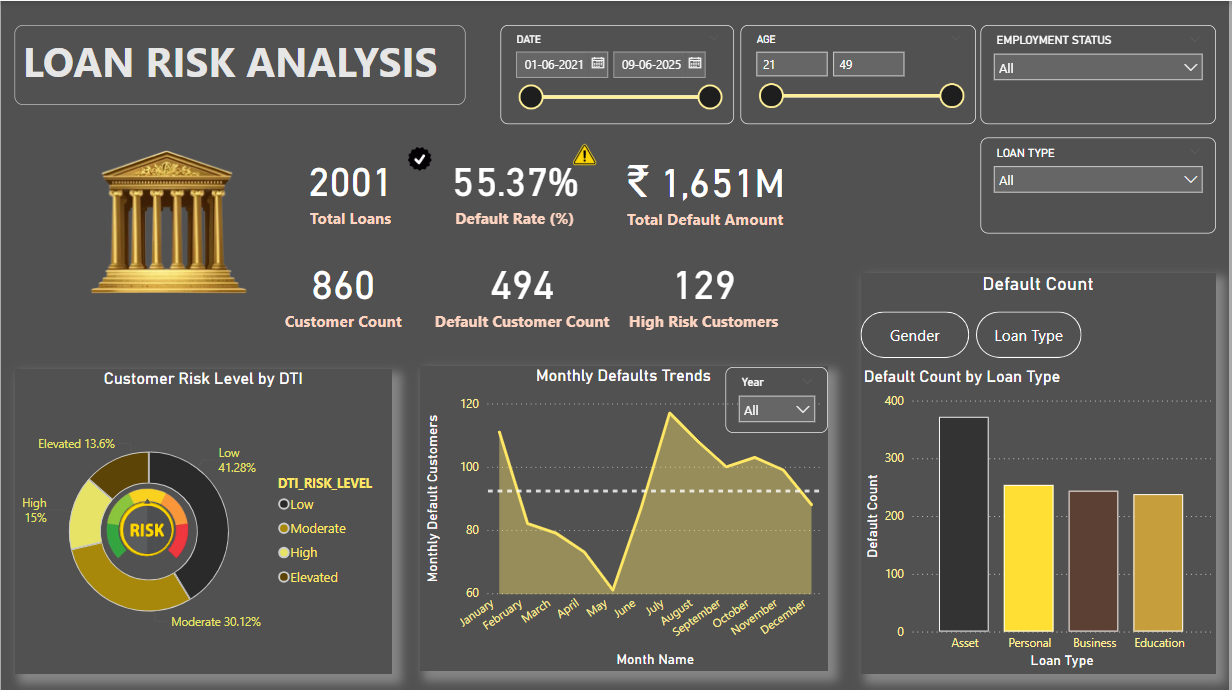
1. **Create a Snowflake Account:**
   * Access Snowflake via https://app.snowflake.com.
   * Create a new **Warehouse** and **Database** for the project.
2. **Define Schemas:**
   * **RAW**: Ingested data from ADLS using Snowpipe.
   * **REFINED**: Cleaned and transformed data.
   * **REPORTING**: Final data used in dashboards.
3. **In the RAW Schema:**
   * Set up **Storage Integration** with Azure.
   * Create **File Format** objects to interpret CSV format.
   * Define **Notification Integration** to trigger ingestion via events.
   * Create base tables:
     + Raw\_Bank\_Customers
     + Raw\_Bank\_Customer\_Assets
     + Raw\_Bank\_Payments
     + Raw\_Bank\_Loans
4. **Create and Configure Snowpipe:**
   * Use Snowpipe to automate ingestion from the ADLS bankdata container.
   * Ensure event notifications are enabled and mapped to the Snowflake pipe.
5. **Transform Data in the REFINED Layer:**
   * Handle:
     + Data type casting
     + Missing/null values
     + Role-Based Access Control (RBAC)
     + Dynamic Data Masking for sensitive fields
   * Store transformed data in cleaned tables.
6. **Analysis & Predictions:**

All analytical operations are conducted within **Snowflake Notebooks**, which provide an interactive environment to write and execute SQL and Python for data exploration, modelling, and transformation.

* + **Exploratory Data Analysis (EDA):**
* Perform initial data exploration on cleaned tables from the REFINED schema using **SQL cells**.
* Analyze key metrics such as:
  + Analyzed key metrics such as data shape, size, and schema using .shape, .info(), and .dtypes. Summary statistics were derived with .describe(), and missing values were identified using .isnull().sum(). These insights guided data cleaning and risk prediction logic.
  + Null or outlier values in key features.
  + Correlations between variables (e.g., income vs. loan amount, missed payments vs. age).
* **Loan Default Prediction (Rule-Based Logic):**
* Implement rule-based logic in **Python** or **SQL cells** to flag high-risk loans.
* Sample logic:
  + If missed\_payments ≥ 3 → mark as **Defaulter**.
  + If EMI / monthly\_income > 0.4 → flag as **High DTI (Debt-to-Income)**.
  + Combine criteria into a DEFAULT\_STATUS column.
* **Customer Risk Profiling:**

Assign a **DTI Ratio** and **DTI Risk Level** (Low, Moderate, Elevated, High) to each customer based on:

* + DTI Ratio
  + Create a final profiling table summarizing Loan ID, Customer ID, Income, Default Status, DTI ratio, and DTI Risk Level
* **Output to Reporting Schema:**
* Final processed and enriched data is written to **tables in the REPORTING schema** such as:
  + Performance\_Metrics
  + Loan\_Summary

**Step 4: - Power BI Dashboard and Visualization**

**Fig 4: - Power BI Dashboard**

The final reporting layer is built in **Power BI**, leveraging a **Direct Query** connection to the REPORTING schema in Snowflake. This ensures real-time data access and up-to-date insights.

* **Connecting Power BI to Snowflake**
* Establish a **Direct Query** connection using Snowflake credentials.
* Select the relevant tables such as CUSTOMER\_RISK\_PROFILES and LOAN\_DEFAULT\_SUMMARY from the REPORTING schema.
* Ensure gateway and region configurations align with your Snowflake deployment.
* **Dashboard Layout Overview**

**Top Section – KPI Cards**

* + - **Default Rate (%)**
    - **Number of High-Risk Customers**
    - **Total Default Amount**
    - **Total Loans Issued**

**Middle Section – Risk Visuals**

* **Left Panel:** Heatmap illustrating default rates by income level and age group.
* **Centre Panel:** Bar chart comparing loan types against their default rates.
* **Right Panel:** Line chart displaying monthly trends in loan defaults.

**Bottom Section – Detailed Reports**

* **Left Panel:** Table showing customer risk profiles, including credit scores and risk categories.
* **Right Panel:** Interactive slicers for filtering by:
  + Age Group
  + Date
  + Employment Status
  + Loan Type