# Banking/Financial Transaction System

## 1. Introduction and Data Generation

This project involved creating a database that models banking and financial transactions to illustrate advanced skills in relational schema design, appropriate data type application, and realistic synthetic data generation. The final implementation significantly exceeded the minimum threshold, incorporating more than 50,000 transaction records.

The dataset was generated entirely through Python 3 scripts, ensuring full compliance with the restriction on using external data sources.

- **Faker Library:** To enhance realism, the Faker library was employed for generating authentic-looking randomized data, including customer names, email addresses, and geographic locations, ensuring practical authenticity within the database.
- **Random Library:** Python's built-in random library was used to introduce numerical variability throughout the dataset. This included generating unique identifiers, randomly selecting categorical values, and assigning realistic monetary figures such as account balances, transaction amounts, and interest rates within sensible ranges.
- **Dimensionality:** The database schema consists of four core tables, each with different record counts. This design reflects a realistic operational environment where transactional data significantly outweighs static entities, ensuring scalability and authenticity.
    - Branches: 10 records
    - Customers: 2,000 records
    - Accounts: 3,000 records
    - Transactions: Over 50,000 records (serving as the required large-volume fact table).

## 2. Database Schema

### 2.1. Textual Schema and Data Types

| Table | Column Name | Data Type | Constraint/Key | Data Type Scale | Justification |
|---|---|---|---|---|---|
| Branches | BranchID | INTEGER | Primary Key | Ratio | Unique identifier. |
| | Location | TEXT | UNIQUE, NOT NULL | Nominal | Categorical name of the branch. |
| | Branch_Tier | TEXT | CHECK | Ordinal | Ordered status (HQ > Regional...). |
| | Assets | REAL | CHECK | Ratio | Total assets held (meaningful zero). |
| Customers | CustomerID | INTEGER | Primary Key | Ratio | Unique identifier. |
| | BranchID | INTEGER | Foreign Key | Ratio | Links to the servicing branch. |
| | Account_Type | TEXT | CHECK | Nominal | Categorical (Personal/Business/Joint). |
| | Risk_Category | TEXT | CHECK | Ordinal | Ordered risk assessment (Low < High). |
| | Name | TEXT | NOT NULL | Nominal | PII, used as a categorical label. |

| | | | | | |
|---|---|---|---|---|---|
| Accounts | AccountID | INTEGER | Primary Key | Ratio | Unique account number. |
| | CustomerID | INTEGER | Foreign Key | Ratio | Links to the customer. |
| | Currency | TEXT | CHECK | Nominal | Categorical type of money (GBP/EUR/USD). |
| | Interest_Rate | REAL | CHECK | Ratio | Percentage rate (meaningful zero). |
| | Balance | REAL | NOT NULL | Ratio | Current amount (meaningful zero). |
| Transactions | TransactionID | INTEGER | Primary Key | Ratio | Unique identifier. |
| | AccountID | INTEGER | Foreign Key | Ratio | Links to the affected account. |
| | Transaction_Date | TEXT | NOT NULL | Interval | Time stamp (arbitrary zero in time). |
| | Transaction_Type | TEXT | CHECK | Nominal | Categorical (Deposit, Withdrawal, etc.). |
| | Amount | REAL | CHECK | Ratio | Monetary value (meaningful zero). |
| | Description | TEXT | NULL allowed | Nominal | Transaction details (allows missing data). |

### 2.2. Inter-Table Relationships and Key Constraints

The integrity of the database rests on a robust network of linked tables:

- Referential Integrity (Foreign Keys): The schema incorporates three essential foreign key dependencies:
    - Customers.BranchID refers to Branches.BranchID
    - Accounts.CustomerID refers to Customers.CustomerID
    - Transactions.AccountID refers to Accounts.AccountID
- Multivariate Key Logic (Composite Key): The Accounts table uses a composite key made up of both the CustomerID and Currency. This setup ensures that each customer can only have one account per currency type. For example, a customer can have one account in GBP and another in EUR, but they can't have two separate GBP accounts.

## 3. Justification for Schema Design and Ethical Oversight

### 3.1. Schema Rationale and Data Realism

This four-table schema is, in fact, designed to meet the criteria of Third Normal Form (3NF) for the ultimate integrity and efficiency of data representation.

- **Normalization Strategy:** This partitioning methodology eliminates data redundancy or proliferation of data. For example, in the large Transactions log, storing client PII, such as names, would create unnecessary data bloat and introduce complexity in case a correction on a client's information is needed.
- **Realism via Imperfection (Missing/Inaccurate Data):**

- Data Voids: The Transactions.Description attribute was coded to allow NULL entries, which happen in about 3% of records. This models real-world gaps in the data, such as incomplete logging or missing merchant information.
- Data Inconsistencies: To simulate real-world system errors, about 1% of the transactions are deliberately duplicated with slight changes to their primary keys. This mimics issues like accidental double-posting of a transaction, which can happen due to glitches or logging errors. These kinds of inconsistencies are common in real datasets and highlight the importance of data cleaning.

## 3.2. Ethical Considerations and Data Privacy

- **Financial Data Protection (Security Posture):**
    - **PII Handling:** Attributes like Customers.Name and Customers.Email are considered Personally Identifiable Information. On a live banking system, this type of information should be encrypted at rest and anonymized or tokenized for personnel who do not need direct access.
    - **Scope Limitation:** In line with the data minimization principle, as provided by various legislations such as the GDPR, among others, this schema has intentionally avoided the capture of information which is considered not essential, such as detailed physical addresses or the numbers of payment cards.
- **Fraud Detection and Fair Practice:**
    - **Analytical Use:** Data points such as Customers.Risk_Category and the historic Transactions activity have been used by the bank for fraud detection.
    - **Ethical Modeling:** Any analytical model developed based on this data should be tested rigorously to minimize algorithmic bias. They must not unfairly penalize groups of customers or characterize them as high-risk simply because of statistical patterns that in themselves do not point to fraudulent activities.
    - **Usage Constraints:** Use of the data shall be strictly limited to its stated purpose, such as security, compliance, and regulatory reporting; use for unauthorized purposes, such as targeted marketing, is strictly prohibited without explicit consent.

## 4. Code Appendix

GitHub Repository link:
https://github.com/Gouthamnaik189/Financial-DB-Generator

GoogleColab Notebook Link:

https://colab.research.google.com/drive/1qmmOxZEbT4n_V0HUEm28iqnKCBwxm482?usp=sharing