FACULTY OF SCIENCES OF TUNIS

UNIVERSITY OF TUNIS EL MANAR

Banking transactions and customer data

A COMPARATIVE STUDY : THE IMPACT OF BIG DATA ON THE ANALYSIS OF BANK TRANSACTIONS AND CUSTOMER DATA

Author :

Ben JemiaaMoez

Supervisor :

Ms. Zekri Manel

Novembre 2025

DEDICATION

I dedicate this work to my family for their unconditional support, constant encouragement, and the trust they have always placed in me. To my parents, who instilled in me the values of perseverance and determination and who have always believed in me. To my friends, for their presence, understanding, and encouragement throughout this project. This work is as much theirs as it is mine..

Moez Ben Jemiaa

November 2025

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to all those who contributed, whether directly or indirectly, to the completion of this project. I am especially thankful to Ms. Zekri Manel for her supervision, valuable advice, and constant availability throughout this work. Her guidance and support greatly contributed to the successful progress of this project. I would also like to thank all my teachers for the quality of their instruction and their support during this academic year. Finally, I extend my thanks to my classmates and friends for their discussions, help, and unwavering support.

Moez Ben Jemiaa

November 2025

TABLE OF CONTENTS

TABLE DES FIGURES

LISTE DES TABLEAUX

GENERAL INTRODUCTION

The rapid growth of Big Data is transforming how banks analyze transactions and customer information. This study compares traditional analytics with modern Big Data approaches to evaluate their effectiveness for tasks such as fraud detection and customer segmentation. We combine a concise literature review with empirical tests on representative datasets to assess accuracy, scalability, and practical trade-offs. The findings aim to provide actionable recommendations for applying Big Data methods in banking contexts.

The remainder of this document is organized as follows. Chapter 1 presents the background and theoretical foundations, covering Big Data concepts, common architectures and analytic techniques, and an overview of banking data characteristics. Chapter 2 reviews related work on Big Data analytics applied to banking transactions and customer data. Chapter 3 describes the datasets, preprocessing steps and the experimental setup used for the comparative study. Chapter 4 reports the experimental results, discusses findings and analyses performance trade-offs. Finally, the Conclusion summarizes the main contributions, outlines limitations, and proposes directions for future research.

CHAPITRE 1

BACKGROUND AND THEORETICAL FOUNDATIONS

This short chapter gives the minimal technical background needed for the study : a brief definition of Big Data, common processing patterns, core analytic approaches, and the main challenges specific to banking data.

1.1 Big Data in brief

Big Data denotes datasets that are large, fast, or varied enough to require distributed storage and processing; typical properties are volume, velocity, variety, veracity and value.

1.2 Architectures

Common patterns are batch processing (e.g., Hadoop), stream processing (e.g., Spark Streaming, Flink), and hybrid designs such as the Lambda or Kappa architectures.

1.3 Analytical techniques

Key methods include descriptive analytics, supervised learning (classification/regression), unsupervised methods (clustering, anomaly detection), and sequence/graph analysis for relational patterns.

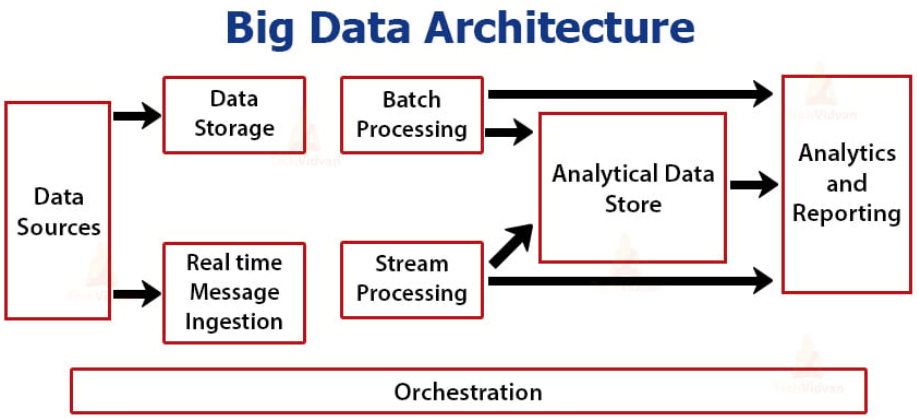

FIGURE 1.1 – Typical Big Data architecture : storage, batch and stream layers, and serving/analytics.

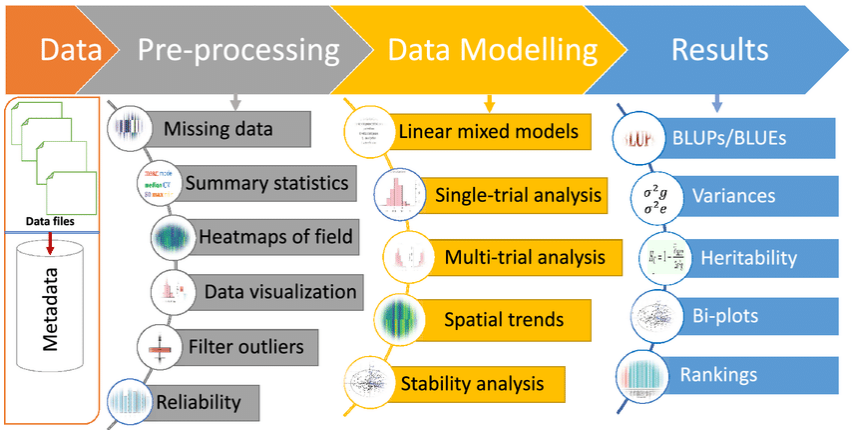

FIGURE 1.2 – Analytic workflow : ETL, feature engineering, model training and deployment.

1.4 Banking data challenges

- Strong privacy and regulatory constraints (e.g., GDPR) - Class imbalance for rare events (fraud) - Temporal dynamics and concept drift - High-cardinality categorical fields and integration of heterogeneous sources

1.5 Summary

These concise points provide the technical baseline used throughout the report.

CHAPITRE 2

LITERATURE REVIEW

Summary (Sommaire)

This chapter reviews prior work and is organized into these parts :

1. Big Data applications in banking : fraud detection, customer analytics
2. Comparative studies : classical vs. Big Data approaches
3. Datasets, evaluation practices, and benchmarks
4. Open challenges : privacy, concept drift, and operationalization

2.1 Big Data in banking

Recent research and industry reports show that Big Data techniques are increasingly applied to : - Fraud and anomaly detection at scale (streaming classifiers, graph analytics) - Customer segmentation and personalization using high-dimensional features and embeddings - Credit risk modeling enhanced by alternative data sources

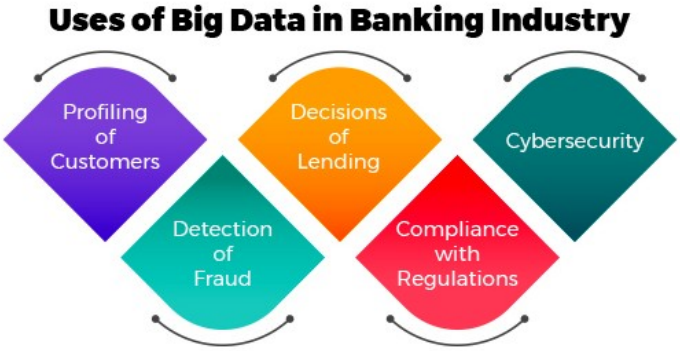

FIGURE 2.1 – Main uses of Big Data in the banking industry : customer profiling, fraud detection, lending decisions, regulatory compliance, and cybersecurity.

2.2 Comparative studies

[H] Several comparative works contrast classical statistical models (logistic regression, tree-based models on single machines) with distributed solutions (Spark MLlib, distributed deep learning). Typical findings : - Big Data implementations scale to larger datasets and support streaming, but may add complexity. - For some problems, optimized classical models remain competitive when data volume is moderate

TABLE 2.1 – Comparative overview : classical approaches vs Big Data approaches

Aspect	**Classical approaches**	**Big Data approaches**
Accuracy	Competitive on moderate, well-curated datasets; often sufficient with careful feature engineering.	Can improve with very large or high-dimensional data; deep models and embeddings may extract additional signal.
Scalability	Limited by single-node resources; horizontal scaling requires re-engineering.	Designed for horizontal scaling across clusters (batch and streaming), handling very large volumes.
Latency	Typically offline/batch; near-real-time is harder to achieve.	Supports low-latency stream processing and online inference for real-time use cases.
Complexity	Lower operational and engineering complexity; quicker prototyping.	Higher engineering overhead (cluster management, distributed pipelines, monitoring).
Cost	Lower infrastructure and maintenance cost for small-to-moderate workloads.	Higher upfront and operational costs for clusters, storage, and engineering.
Interpretability	Often more interpretable (linear models, trees); favorable for regulatory contexts.	Models can be less transparent (deep learning, ensembles); requires extra explainability work.
Deployment time	Faster to deploy proofs-of-concept and small-scale models.	Longer setup time, but supports production-grade, scalable deployments once built.
Suitable use cases	Moderate data volumes, offline analytics, scenarios prioritizing interpretability and low cost.	High-volume, low-latency, heterogeneous data sources, large-scale personalization, and streaming fraud detection.

.

2.3 Datasets and evaluation

Benchmarking is difficult due to limited public banking data. Good practices : - Use time-aware splits to avoid leakage - Report metrics for imbalance (AUC-PR, precision@k) - Provide reproducible code and synthetic datasets when private data cannot be shared

2.4 Open problems and gaps

- Privacy-preserving analytics (secure aggregation, differential privacy) - Robust online learning under concept drift - Clear operational guidelines to reduce engineering and governance overhead

2.5 Summary

This literature review highlights where Big Data adds value and where trade-offs exist, framing the comparative experiments that follow.

CHAPITRE 3

SYSTEM DESIGN, ETL AND VISUALIZATION

Summary (Sommaire)

This chapter presents the practical realization of our Big Data analytics system for the banking context. We begin by describing the ETL (Extract–Transform–Load) process applied to the public *Banking-Dataset*. Then we present the conception of the system through UML diagrams (class diagram, use-case diagrams, and sequence diagrams). Finally, we describe the visualization layer implemented with Power BI.

3.1 Source Dataset and ETL Process

3.1.1 Source dataset

The raw data used in this project comes from the public repository *Banking-Dataset* available on GitHub [1] and more specifically from the file `Comprehensive_Banking_Database.csv`. This file is a wide, denormalized table where each row contains information about :

— the customer (socio-demographic profile and contact data),

— one or more accounts and their balances,

— transactions performed on these accounts,

1. `https://github.com/ahsan084/Banking-Dataset`

— loans and their characteristics,

— credit cards and their limits/balances,

— customer feedback and resolution status,

— anomalies (potential fraud or abnormal behavior),

— the branch associated to the operations.

While convenient for storage, this wide format is not suitable for analytical queries or dashboard tools. We therefore implemented an ETL process to normalize the CSV into several relational tables.

3.1.2 Objectives of the ETL

The ETL process has the following objectives :

— **Extract** data from the original CSV file.

— **Transform** the data into a clean, consistent relational schema : data type coercion, standardization of identifiers, and derivation of a unique `AccountID`.

— **Load** the different entities into an Excel workbook with one sheet per table, ready to be imported into Power BI.

The target schema contains the following tables :

Customers master data on bank customers.

Accounts bank accounts associated with customers.

Transactions financial transactions performed on accounts.

Loans loan contracts.

Cards credit cards.

Feedback customer feedback and complaint resolution.

Branches bank branches.

Anomalies records flagged as anomalous.

DataDictionary documentation of primary keys and relationships.

3.1.3 Implementation of the ETL (Python)

The ETL is implemented in Python in the script `ETL.py`. The script relies on the libraries `pandas` and `openpyxl` and can be executed as follows :

```
python ETL.py \
    Comprehensive_Banking_Database.csv \
    Comprehensive_Banking_Database.xlsx
```

The script performs the following main steps :

3.1.3.0.1 Column detection and typing. Column names are matched in a case-insensitive way, which allows some flexibility if the CSV headers slightly differ. Common date fields (e.g., `Transaction Date`, `Date Of Account Opening`, `Approval/Rejection Date`) are parsed using `pandas.to_datetime`, while numeric fields (e.g., balances, loan amounts, interest rate, credit limit) are cleaned and converted to numeric types.

3.1.3.0.2 Generation of a deterministic `AccountID`. Because the original dataset does not always provide a stable account identifier, the script synthesizes an `AccountID` by hashing the triplet `(CustomerID, AccountType, DateOfAccountOpening)` :

— the values are concatenated into a string;
— a SHA-1 hash is computed and truncated;
— the result is prefixed by `"ACC_"`.

This approach guarantees that the same logical account always receives the same identifier when the ETL is re-run.

3.1.3.0.3 Construction of relational tables. The script then builds each table :

— **Customers** : one row per `CustomerID` with demographic attributes and contact information.
— **Accounts** : one row per `AccountID`, linked to `CustomerID`, with account type, balance, opening date, last transaction date, and branch.
— **Transactions** : one row per `TransactionID`, linked to `AccountID` and optionally to a branch, with date, type, amount and resulting balance.
— **Loans** : one row per `LoanID`, linked to `CustomerID` and a branch, with amount, type, interest rate, term, approval/rejection date and status.
— **Cards** : one row per `CardID`, linked to `CustomerID`, with card type, credit limit, balance, minimum payment, due dates and reward points.
— **Feedback** : one row per `FeedbackID`, linked to `CustomerID`, describing the feedback type, date, resolution status and resolution date.
— **Branches** : list of distinct branch identifiers (`BranchID`).

— **Anomalies** : subset of rows where the `Anomaly` field is present, linked to `CustomerID`.

3.1.3.0.4 Export to Excel and data dictionary. Finally, all tables are written into a single Excel workbook (`.xlsx`), each table corresponding to a separate sheet. A `DataDictionary` sheet summarizes primary keys, foreign keys and comments about each table. For each sheet the script prints the number of rows created, which facilitates validation of the ETL process.

3.2 Conceptual and Logical Design

In this section we present the conceptual and logical design of the analytical system. We first describe the data model (class diagram), then we detail the functional requirements through use-case diagrams, and finally we illustrate the dynamic behavior using sequence diagrams.

3.2.1 Data Model / Class Diagram

The relational schema generated by the ETL is imported into Power BI and used as the central analytical data model. Figure 3.1 shows the class (or entity-relationship) diagram corresponding to the tables.

The main entities are :

— **DimCustomer** : stores the master data of customers (identifier, name, age, gender, address, city, contact number, email).

— **DimAccount** : describes accounts with their type, balance, opening date, last transaction date and associated branch.

— **FactTransactions** : fact table containing all account transactions with their date, type, amount and resulting balance.

— **Loans** : list of loans granted to customers with conditions and status.

— **Cards** : credit cards with their limits, balances and payment deadlines.

— **FactFeedback** : feedback or complaints expressed by customers, including resolution status.

— **FactAnomalies** : anomalies or suspicious behaviors linked to customers.

— **Branches** : bank branches identified by `BranchID`.

Foreign key relationships follow a star-schema pattern : several fact tables (`FactTransactions`, `FactFeedback`, `FactAnomalies`, `Loans`, `Cards`) reference descriptive dimensions (`DimCustomer`, `DimAccount`, `Branches`).

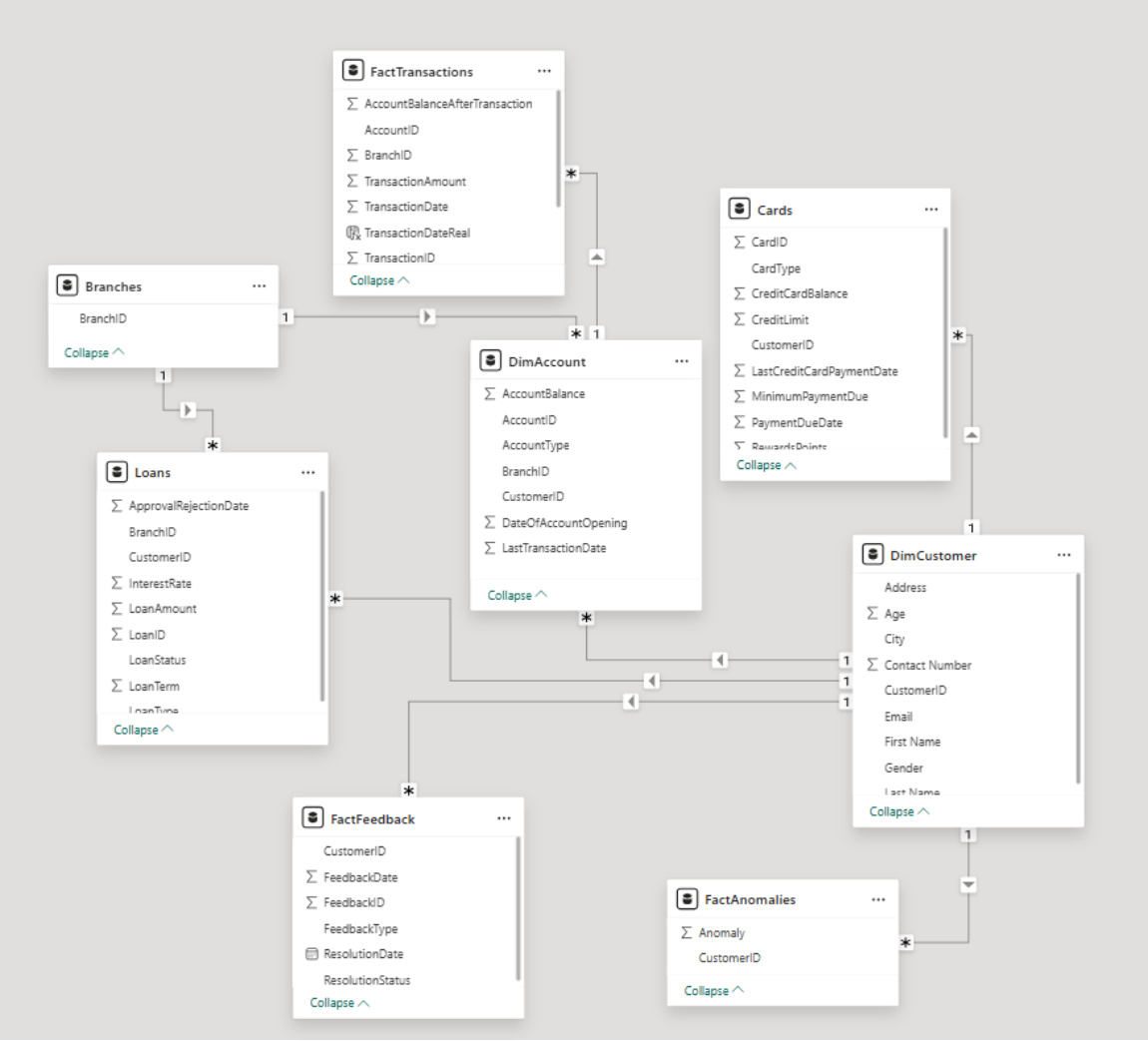

FIGURE 3.1 – Relational/class diagram of the analytical schema : customers, accounts, transactions, loans, cards, branches, anomalies and feedback.

3.2.2 Use-Case Diagram

The system is used by several types of actors :

— **Fraud Analyst** : monitors anomalies and investigates suspicious customer behavior.

— **Risk Manager** : analyzes credit risk and the performance of the loan portfolio.

— **Marketing Analyst** : studies customer segments and their profitability.

— **Top Management** : consults aggregated strategic dashboards.

Figure 3.2 shows the general use-case diagram, generated with PlantUML from the English specification given below.

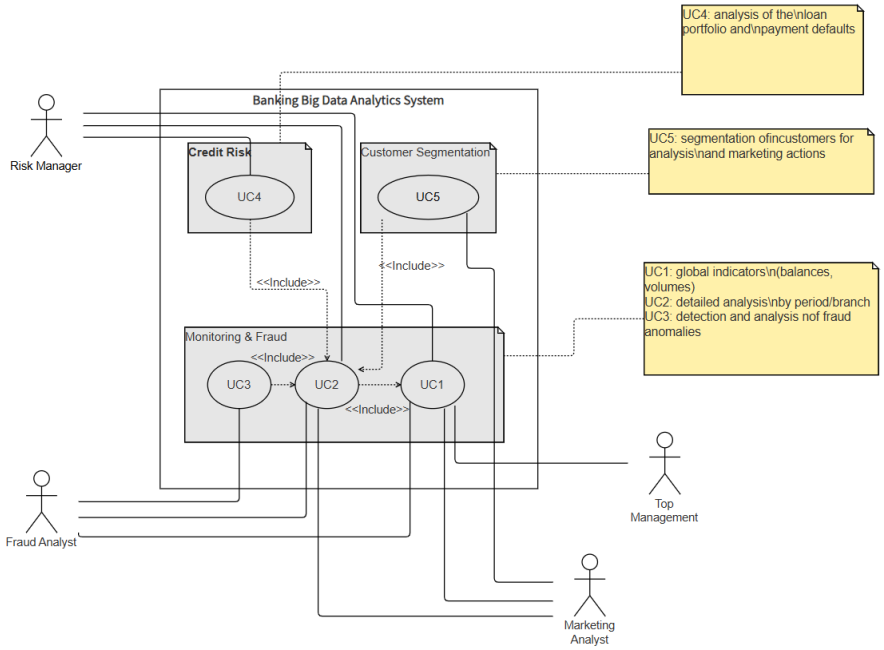

FIGURE 3.2 – General use-case diagram for the banking Big Data analytics system.

3.2.2.1 List of main use cases (UC1–UC5)

The simplified diagram is based on the following five main use cases, grouped by functional area :

3.2.2.1.1 Monitoring and fraud

UC1 – View global KPIs (balances, volumes) The user views global indicators of activity (number of active accounts, total transaction volume, average balances) over a selected period.

UC2 – Analyze transactions (by period / branch) The user explores transactions in more detail by period, branch, product type or customer segment, with filtering and comparison capabilities.

UC3 – Detect and analyze fraud anomalies The system highlights anomalous patterns in transactions and presents alerts that can be investigated in detail by the fraud analyst.

3.2.2.1.2 Credit risk and customer segmentation

UC4 – Analyze loan portfolio and defaults The risk manager monitors loan exposures, payment delays and default rates by product, region or customer segment.

UC5 – Segment customers The marketing analyst builds customer segments based on profile and behavior (transactions, cards, loans, feedback) to support analytical studies and targeted actions.

3.2.3 Refined Use Cases

Some critical use cases are refined below.

UC3 – Detect and analyze fraud anomalies

Main actor : Fraud Analyst.

Preconditions :

— the tables `FactTransactions` and `FactAnomalies` are populated by the ETL;

— the Power BI dashboards are deployed and accessible.

Main scenario :

1. The analyst opens the fraud anomalies dashboard.
2. The system displays a list of anomalies aggregated by customer, anomaly type and severity.
3. The analyst filters by period, region or product type.
4. The analyst selects a customer to view the detailed transactions associated with the anomaly.
5. The analyst classifies each case (confirmed fraud, false positive, under investigation).
6. The anomaly status is updated in the system.

UC4 – Analyze loan portfolio and defaults

Main actor : Risk Manager.

Main scenario :

1. The risk manager opens the loan portfolio report.
2. The system aggregates loan exposures by type, region, customer segment and internal rating.
3. The user views indicators such as delay in payment, default rates and recovery rates.
4. The user identifies the riskiest portfolio segments and exports detailed results for further analysis.

UC1 – View global KPIs (balances, volumes)

Actors : Fraud Analyst, Marketing Analyst, Risk Manager, Top Management.

Main scenario :

1. The user connects to Power BI and selects the “Transactions and Balances” dashboard.
2. The system displays global indicators (number of active accounts, transaction volume, average balance).
3. The user applies filters (period, branch, account type, customer segment).
4. The charts and KPIs are updated dynamically.

3.2.4 Sequence Diagrams

To illustrate the dynamic behavior of the system, we present two sequence diagrams : one for fraud investigation and one for dashboard consultation.

Sequence Diagram – UC3 : Fraud investigation

Figure 3.3 shows the interactions between the Fraud Analyst, the Power BI dashboard and the Data Warehouse (result of the ETL).

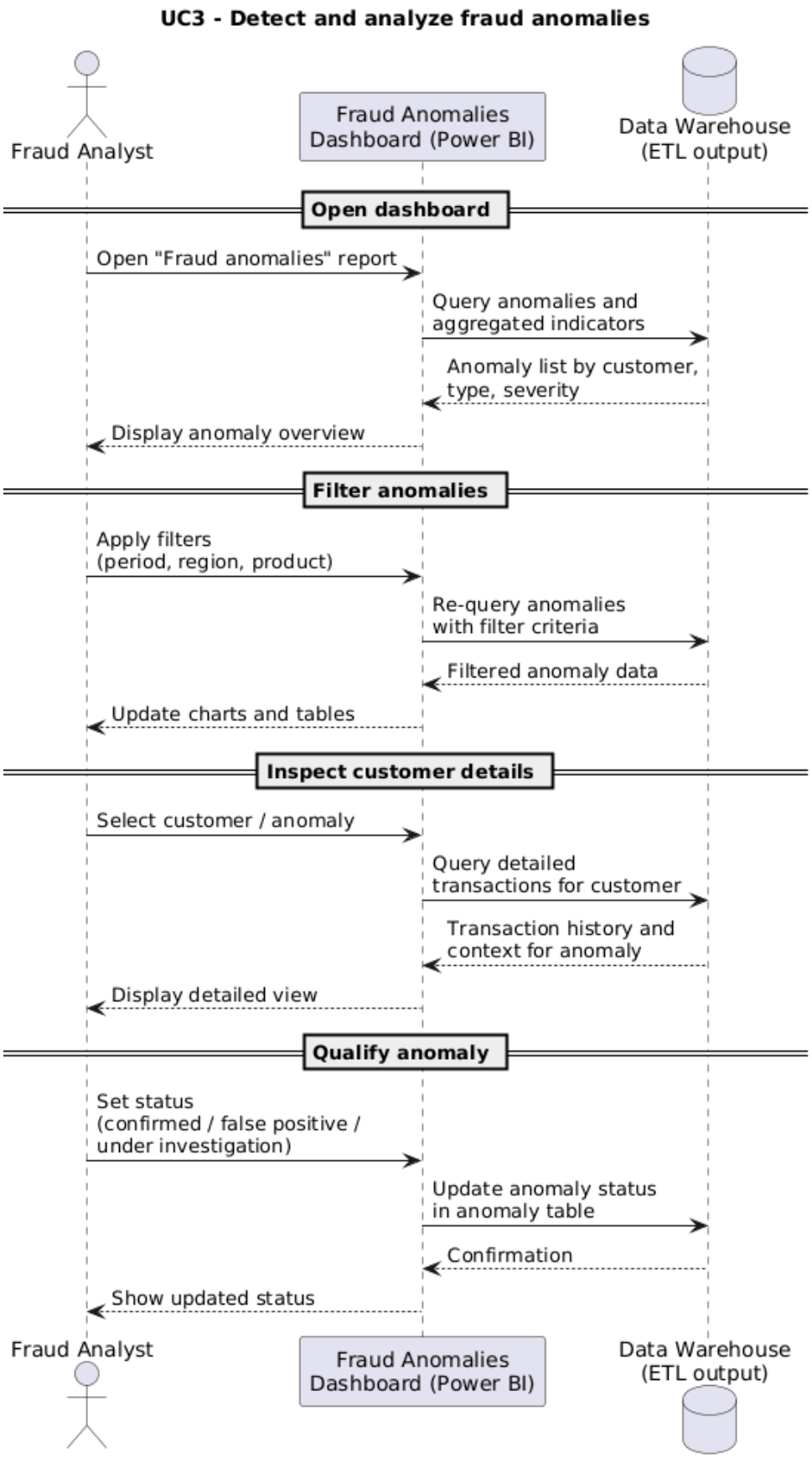

FIGURE 3.3 – Sequence diagram for the “Detect and analyze fraud anomalies” use case (UC3).

Sequence Diagram – UC1 : Dashboard consultation

Figure 3.4 describes the steps when an analyst consults the transaction and balance dashboard.

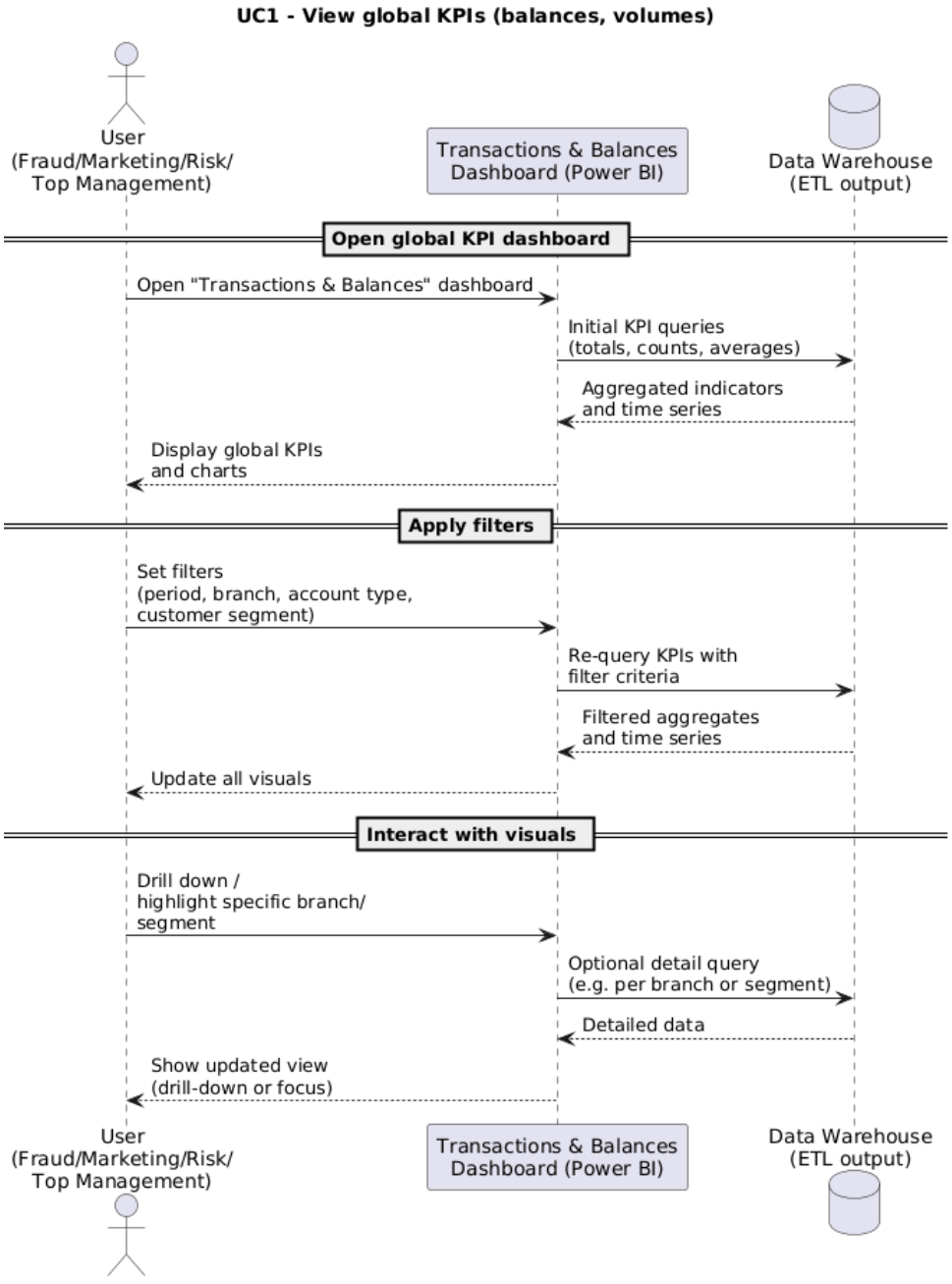

FIGURE 3.4 – Sequence diagram for the “View global KPIs” use case (UC1).

3.3 Visualization with Power BI

3.3.1 Objectives of the Visualization Layer

The visualization layer aims to transform the relational data produced by the ETL into an interactive dashboard that supports analysis of banking transactions and customer data. The main objectives are :

— provide synthetic indicators on transactions, account balances, loans, customer activity and anomalies ;

— allow interactive exploration of the data by month, branch, city and product type ;

— offer a consolidated view of customer and branch performance in order to support data-driven decisions.

3.3.2 Power BI Data Model

The Power BI data model reuses the schema described in Figure 3.1. The tables `DimCustomer`, `DimAccount`, `FactTransactions`, `Loans`, `Cards`, `FactFeedback`, `FactAnomalies` and `Branches` are loaded from the Excel file produced by the Python ETL. Relationships are configured according to the primary and foreign keys so that filters can propagate from customers and branches to accounts, transactions, loans, feedback and anomalies.

3.3.3 Dashboard – Banking transactions and customer data

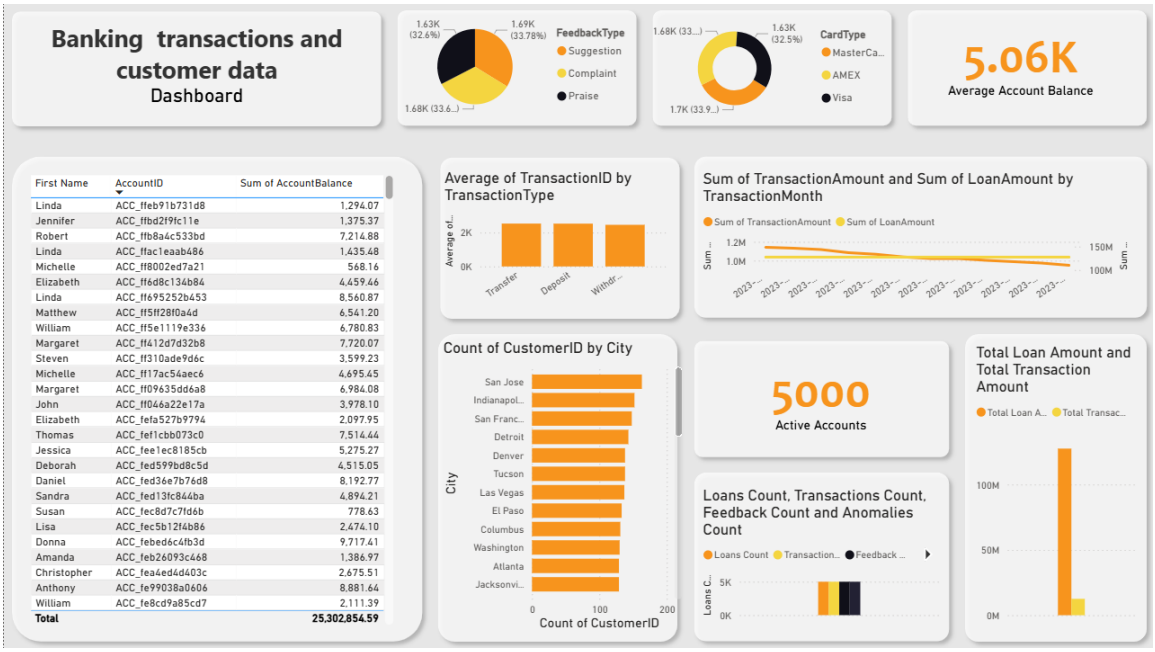

FIGURE 3.5 – Power BI dashboard for banking transactions and customer data.

A single synthesis dashboard was designed to summarize the main aspects of the dataset. The layout follows a top–down logic, from global indicators to more detailed views.

The main components of the dashboard are :

— **KPI cards** at the top of the page show the average account balance, the number of active accounts and other global measures such as total loan amount or total transaction volume.

— A set of **pie and donut charts** summarizes the composition of the portfolio by feedback type (suggestion, complaint, praise) and by card type (MasterCard, AMEX, Visa), giving a quick overview of product usage and customer satisfaction signals.

— **Bar charts** display the distribution of activity by transaction type (transfer, deposit, withdrawal) and the number of customers per city, which highlights the most active customer locations.

— A **time-series chart** presents the sum of transaction amounts and loan amounts by transaction month, allowing the user to observe seasonal patterns and the joint evolution of lending and transaction activity.

— A large **table of accounts** lists customers, account identifiers and the sum of account balances, with totals at the bottom. This table acts as a detailed view that can be filtered from the other visuals.

— Additional **comparison charts** show, for example, the total loan amount versus total transaction amount, as well as the counts of loans, transactions, feedbacks and anomalies, providing a compact summary of the main volumes in the dataset.

All visuals share the same report filters, so that selecting a given city, branch, period or product type updates the entire dashboard consistently. This interactive Power BI page demonstrates how the integrated data model can be used to obtain both a high-level and a detailed view of banking transactions and customer data.

3.4 Summary

In this chapter we have presented the complete pipeline from raw data to decision-support dashboards. Starting from the public *Banking-Dataset*, a Python ETL script normalizes the CSV file into a relational schema stored in an Excel workbook. This schema, imported into Power BI, serves as the basis for our analytical model. We then described the conceptual design through a class diagram, a general use-case diagram with refined use cases, and sequence diagrams illustrating key scenarios. Finally, we presented the main

dashboards developed for transactions, fraud, credit risk and customer analysis, which will be used in the next chapter to evaluate the contributions of Big Data techniques in the banking context.

GENERAL CONCLUSION

This work has examined how Big Data techniques transform the collection and use of banking transactions and customer data. The study shows that scalable storage and distributed processing make it possible to exploit detailed, multi-channel information for fraud detection, customer analytics and credit risk assessment, while raising new challenges in terms of complexity, governance and privacy.

On a practical level, a Python-based ETL pipeline was implemented to normalize a public banking dataset into a clear relational schema, covering customers, accounts, transactions, loans, cards, branches, feedback and anomalies. This schema supported a set of key analytical use cases and interactive dashboards in Power BI, enabling the monitoring of global KPIs, detailed transaction analysis, anomaly investigation, portfolio risk assessment and customer segmentation.

Overall, the results illustrate that the real contribution of Big Data in banking lies in the coherent integration of data engineering, modeling and visualization around concrete decision-making needs.

NETOGRAPHIE & BIBLIOGRAPHIE

[1] Moez Ben Jemiaa. Power bi dashboard for banking transactions and customer data. Power BI project file (.pbix) provided as part of this thesis, 2025. Available from the author upon request.

[2] Ahmed Ahsan. Banking-dataset : Comprehensive banking database. https://github.com/ahsan084/Banking-Dataset, 2024. Accessed 2025-05-01.

[3] Moez Ben Jemiaa. Etl.py : Csv-to-relational excel script for the banking dataset. GitHub repository (personal project), 2025. Python ETL used in this thesis.

[4] Python Software Foundation. Python language reference. https://www.python.org/, 2024. Accessed 2025-05-01.

[5] pandas development team. pandas : Python data analysis library. https://pandas.pydata.org/, 2024. Accessed 2025-05-01.

[6] Eric Gazoni and contributors. openpyxl : A python library to read/write excel 2010 xlsx/xlsm files. https://openpyxl.readthedocs.io/, 2024. Accessed 2025-05-01.

[7] Microsoft Corporation. Microsoft power bi. https://powerbi.microsoft.com/, 2024. Accessed 2025-05-01.

[8] PlantUML Team. Plantuml : Open-source uml tool. https://plantuml.com/, 2024. Used to generate diagrams.

All artefacts of the project, including the Python ETL script [3], the relational Excel file, and the Power BI `.pbix` dashboard file [1], are available from the author upon request.