FINANCIAL INTELLIGENCE ACTIVITY TRACKER (FIAT) SYSTEM

A Proof-of-Concept

Kananelo Lekgetho
lekgethokananelo@gmail.com

Acknowlegements

I would like to extend my deepest gratitude to the University of the Free State, particularly to its remarkable community, for empowering me with the skills and knowledge needed to explore, harness, and innovate with advanced technologies, especially in a context where there is a significant gap in critical skills across South Africa. My heartfelt thanks also go to MerSETA for their generous funding of my postgraduate studies in 2025; your invaluable support has played a crucial role in my academic journey.

This work is dedicated wholeheartedly to my family and friends, whose unwavering encouragement and belief in me have been a constant source of strength throughout the development of this project. Your steadfast support is sincerely cherished and deeply appreciated.

Contents

Table of Figures

Table of Tables

Abstract

This document outlines the design and implementation of the Financial Intelligence Activity Tracker (FIAT) Minimum Viable Product, a data warehouse and business intelligence system aimed at consolidating fragmented personal financial data. The project originated from the need to integrate transaction records across banking platforms, investment accounts, and emails. It details the project lifecycle, starting with requirements analysis and dimensional modelling. The solution features a layered ETL architecture using Python for data extraction and SQL Server Integration Services (SSIS) for transformation and loading into a Microsoft SQL Server data warehouse organised as a fact constellation schema. Metabase is used for self-service data visualisation and analysis. The document includes technology justification, implementation scripts, and example outputs, concluding with reflections on development challenges, the system's proof-of-concept nature, data completeness limitations, and its success in demonstrating a unified DW/BI stack for analytical insights.

Chapter 1: Introduction

Background Information

In 2022, the author initiated an investment portfolio through the EasyEquities platform, generating financial activity across multiple touchpoints, including platform records, email confirmations, and bank statements. While these records were accessible individually, they remained fragmented, preventing a consolidated, longitudinal view of capital movement and performance. By 2025, a central analytical challenge emerged: how to evaluate portfolio performance holistically and derive behavioural insights from underlying activity. Existing platform tools were insufficient, motivating the development of the Financial Intelligence Activity Tracker (FIAT) system as a centralised repository for high-fidelity financial analysis.

Problem Statement

The author often faces fragmented financial data spread across multiple sources, including email confirmations, bank statements, and platform transaction records. This distribution makes it challenging to consolidate, track, and analyse portfolio activity over time. Without a centralised analytical system, it is difficult to quantify total inflows and outflows, identify the entities driving the most transactions or revenue, and detect behavioural patterns in investment activity. Existing platform tools are insufficient due to persistent data silos, inconsistent formats, and limited historical aggregation.

The FIAT system addresses this problem by providing a unified repository for personal financial data, enabling longitudinal analysis of transactions and the extraction of meaningful behavioural insights. The system seeks to answer the following key questions:

1. What are the total investment inflows and outflows over time?
2. Which entities are most frequently transacted with each month?
3. Which entities contribute the most to portfolio gains?
4. How do investment behaviours change over time, and what patterns can be identified from transactional data?

Objectives

The primary objective of this work is to design and implement a data warehouse MVP that consolidates fragmented personal financial data into a unified structure for longitudinal analysis. To achieve this, the system establishes an automated ETL pipeline to ingest semi-structured email confirmations and manually downloaded transaction records, transforming them into a queryable format. The architectural foundation relies

on modelling these financial events using a dimensional schema that captures the specific grain of investments, sales, and withdrawals.

Ultimately, the work produces a minimum viable software product that supports self-service exploration of capital flows while evaluating the capacity of integrated data to support exploratory applications in regulatory analytics simulations. This demonstrates the technical significance of structured data consolidation, offering insights into transactional trends and entity preferences not available through standard consumer interfaces. By transforming fragmented records into a queryable repository, FIAT provides a template for lightweight financial intelligence solutions.

Scope of the Project

The scope of the FIAT MVP is strictly defined by the processing of personal financial data sourced from Gmail API ingestions and manually downloaded transaction statements in PDF or CSV formats. The system does not currently integrate third-party data or implement enterprise-grade multi-user compliance features. Functionality focuses on the analysis of inflows, outflows, and behavioural patterns within the established dataset. This controlled scope allows for testing the reliability of the data pipeline and the validity of the behavioural extraction logic without the overhead of broad-scale commercial deployment.

Solution: FIAT MVP

FIAT functions as a data pipeline that converts fragmented records into actionable intelligence. Data is ingested automatically or manually, validated, and cleaned within a staging environment using SQL Server Integration Services. Curated data is stored in a star schema with a fact constellation design, optimised for historical querying. The Metabase interface provides the end user with tools to explore portfolio performance, capital flows, and strategic patterns.

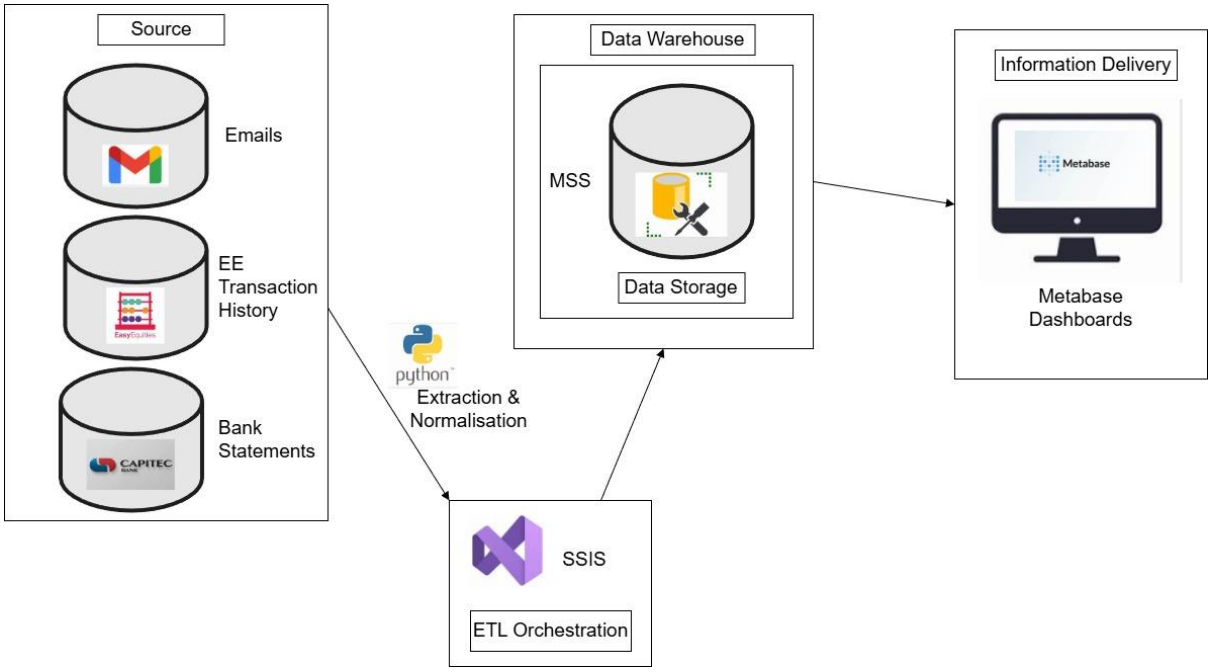


Figure 1: An architectural overview of the proposed DW/BI system

Limitations of the Project

Some data sources that provide a comprehensive transactional overview of the EasyEquities investment account, such as the Transactional History Report, are limited to a maximum date range of one year. As a result, certain transactions, like in-account monthly deductions during periods when the account was inactive, may not be captured. The bank statements also have their own date range limit. Although bank statements have longer date ranges, it becomes a risk if there is still some data that goes beyond the range limit set by the bank.

This limitation is one of the core limitations of the system because it greatly affects the quality of the data, resulting in the incompleteness quality issue. Future work would include using more advanced Python functionalities to extract data from a compiled list of monthly statements to extract some missing data, as it is likely to appear in those records. More aspects of the limitations of the EasyEquities platform are discussed in Chapter 2: Comparison of Existing Systems.

This work made use of personal information of the author as part of the experimentation. Tools used to obtain, process and store these data were all used on a local computer, which is protected by Windows 11 Microsoft Defender. A backup Solid State Drive disk that is password-protected was also used to store backup data. In general, steps were taken to secure the confidential information; however, a strict compliance framework (Financial Sector Regulation Act, 2017 Joint Standard 2 Of 2024 Cybersecurity and Cyber Resilience Requirements) was not adopted, as this work only serves as a proof of concept and not a deployment-ready product.

This work was developed under a limited time frame, and the institutional license privileges used by the author for implementation and debugging are set to expire around the time of graduation in April 2026. Moreover, this work will not have a full chapter dedicated to evaluation; the system will only be evaluated by its ability to deliver to the expectations formulated in the requirements definition statement in Chapter 4. This evaluation will only be discussed in a subsection titled Reflections.

Chapter 2: Existing Systems and Foundational Literature

Introduction

This chapter presents an overview of the systems that effectively aggregate transaction amounts over time for the end-user that are currently being used by the client (the author). It details the limitations of these existing systems and compares them to the proposed FIAAT system. It explores the theoretical foundations of these concepts within the framework of end-to-end data warehousing and business intelligence (DW/BI). Key ideas in data warehousing, business intelligence, and behavioural finance are thoroughly examined, providing a clear and authoritative understanding of how these systems operate and their crucial role in driving informed decision-making.

Comparison of existing systems

Capitec Bank

Capitec is one of the largest commercial banks in South Africa, serving approximately 25 million customers (News24, 2025). Since its establishment in 2001, it has expanded to provide a range of personal and business banking services, including transactional accounts, credit products, and insurance offerings (Capitec, 2026).

Capitec enables customers to deposit funds via ATMs and branches and offers mobile banking services that allow users to monitor their financial activities. The mobile application includes basic data visualisation features, such as bar charts representing monthly spending trends (Figure 2). These visualisations support quick interpretation,

comparison across months, and high-level trend awareness.

Figure 2: The historical data visual presentation from the Capitec Bank Online Banking App

However, several analytical limitations exist:

- The charts emphasise discrete monthly summaries rather than continuous financial behaviour over time. This restricts the ability to trace behavioural patterns
- Visualisations become cluttered over extended time periods, reducing readability.
- Aggregated totals obscure the underlying composition of transactions, limiting drill-down analysis. This limitation is particularly relevant in the context of Business Intelligence, where multidimensional analysis is essential.

Additionally, Capitec's transaction records are internally focused. The system captures only transactions occurring within the banking environment and does not integrate external financial activity. Although users can download historical bank statements (up to six years) in PDF or CSV formats, these files follow bank-specific data structures. This creates a challenge for users seeking to perform cross-platform financial analysis, as multiple incompatible formats must be manually consolidated.

EasyEquities Investment Platform

EasyEquities (EE), operated by First World Trader (Pty) Ltd (a subsidiary of Purple Group Limited, listed on the JSE), was launched in 2014 to broaden access to investment opportunities in South Africa. Through its Fractional Share Rights model, EE allows individuals to purchase portions of shares and invest in both local and international markets (EasyEquities, 2026).

EE provides more advanced visual reporting than Capitec through its monthly statement feature, which includes portfolio performance, asset allocation, and cash flow summaries (Figures 3–5). These reports use multiple visual formats, including percentages, line graphs, and pie charts, each supporting different cognitive strengths in data interpretation (Gulati, 2025).

Another weakness of this system is that it provides too many details in the monthly statements, which end up confusing the author. For example, the Instrument Exposure table (Figure 4) is full of numbers at a glance, which makes the author's brain immediately clog with the single entries of each cell in that table. The weakness of this tabular structure is that it is poor at showing relationships and visual trends. The monthly statements excel at showing relationships (Figure 4), but their time horizon is very limited.

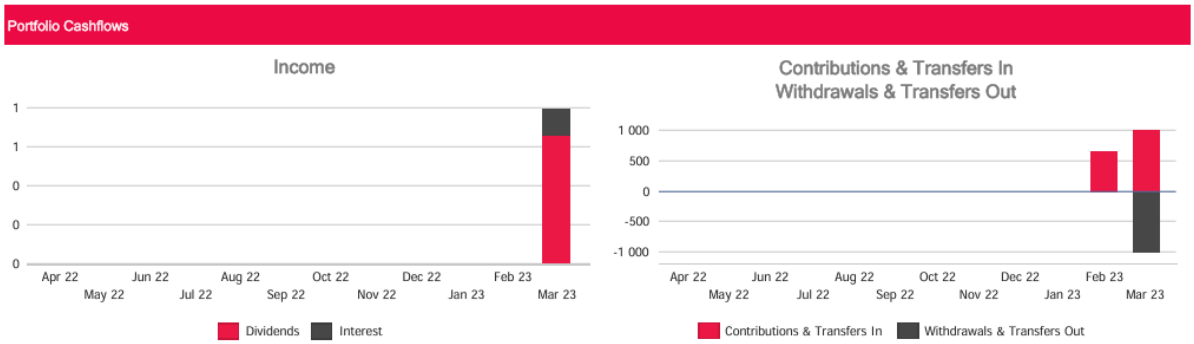


Figure 3: Data visualisation of Portfolio Cashflows from the EE platform

Despite these strengths, the platform has several analytical shortcomings:

- The time horizon is limited. Transaction history reports and statements only provide short- to medium-term views, restricting long-term behavioural analysis.
- Data is pre-curated, meaning users cannot freely perform cross-analysis or multidimensional drill-down beyond what the platform provides.
- Similar to Capitec, EE only records transactions occurring within the platform, excluding related external banking activities such as deposit charges.

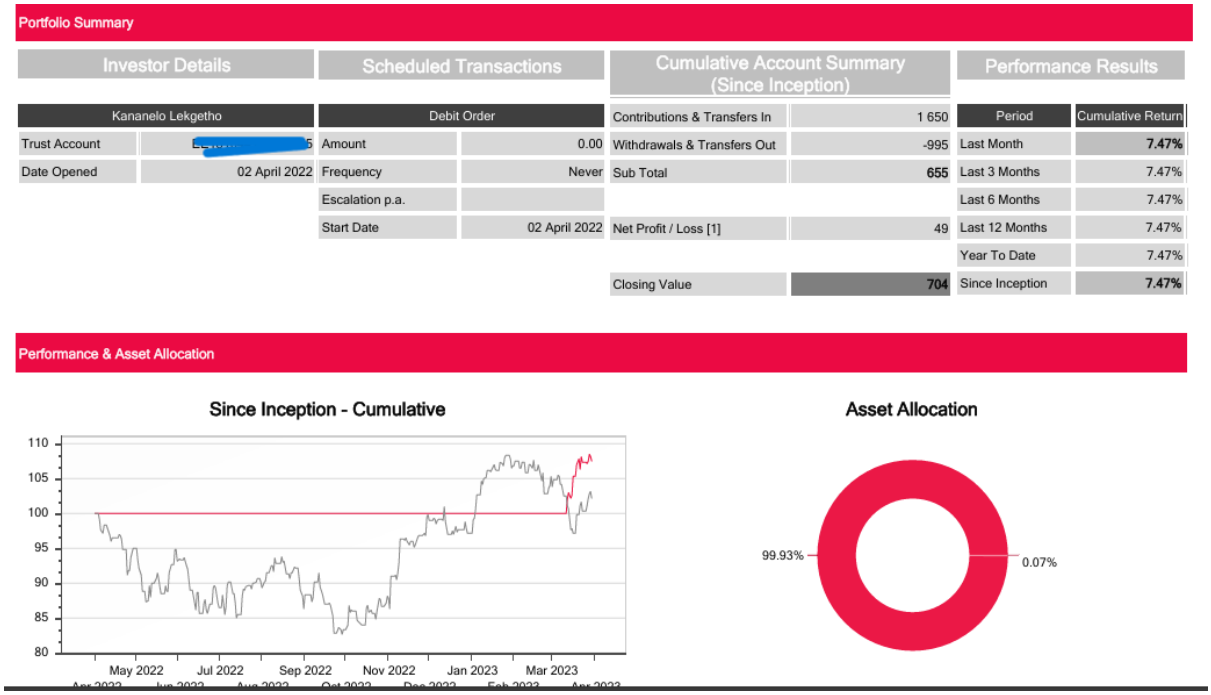


Figure 4: The Portfolio Summary and Performance and Asset Allocation structured data view

Certain tabular views, such as the Instrument Exposure table (Figure 5), present dense numerical data that increases cognitive load and makes trend identification difficult. Tables are effective for precise values but weak for storytelling, pattern recognition, and temporal analysis (Studoco, 2024).

Instrument Exposure at 31 March 2023

	Opening Balance		Purchases		Sales			Closing Balance					
Instrument	Qty	Cost	Qty	Cost	Qty	Proceeds	Profit/Loss	Qty	Cost	Cost Price	Curr Price	Curr Value	Weight
Anglo American PLC	0.0000	0.00	0.2134	124.72	0.0000	0.00	0.00	0.2134	124.72	584.46	586.00	125.05	17.8%
Anglogold Ashanti Limited	0.0000	0.00	0.0434	17.17	0.0000	0.00	0.00	0.0434	17.17	395.57	431.40	18.72	2.7%
Distell Group Holdings Limited	0.0000	0.00	0.8340	150.86	0.0000	0.00	0.00	0.8340	150.86	180.89	180.50	150.54	21.4%
EC10	0.0000	0.00	5.8982	352.14	-2.8349	184.80	23.32	3.0633	190.67	62.24	67.71	207.42	29.5%
Gold Fields Limited	0.0000	0.00	0.1667	34.23	0.0000	0.00	0.00	0.1667	34.23	205.32	237.00	39.51	5.6%
Impala Platinum Hlgs Limited	0.0000	0.00	0.1993	34.22	0.0000	0.00	0.00	0.1993	34.22	171.70	163.83	32.65	4.6%
Mr Price Group Limited	0.0000	0.00	1.2900	181.05	-0.4974	68.38	-2.02	0.7926	110.64	139.60	144.20	114.29	16.3%
Satrix MSCI Emerging Markets ETF	0.0000	0.00	1.9212	100.33	-1.9212	101.40	1.07	0.0000	0.00	0.00	0.00	0.00	0.0%
Woolworths Holdings Limited	0.0000	0.00	0.2326	15.10	0.0000	0.00	0.00	0.2326	15.10	64.92	63.93	14.87	2.1%
Total		0.00		1 009.82		354.57	22.37		677.61			703.05	100.0%

Figure 5: Table showing Instrument Exposure in the EE Monthly Statement Report

The FIAT system

The proposed FIAT system shares a partial objective with existing platforms in that it aims to provide insight into capital movement between investment and banking environments. However, it extends beyond monitoring by introducing integrated, exploratory, and behaviour-focused analytics.

Unlike existing systems that operate in isolation, FIAT integrates multiple financial data sources, including bank statements, email transaction confirmations, and platform transaction histories. These heterogeneous data sources are standardised and modelled using a dimensional data warehouse, enabling OLAP operations, drill-down analysis, and future data mining capabilities.

FIAT applies sequence linking to analyse chains of financial events over time. This enables deeper behavioural insight rather than single-event detection. Furthermore, FIAT transforms dashboards from passive reporting tools into interactive investigation environments. Through Metabase, users can generate their own queries, ask follow-up questions, apply filters, and explore anomalies dynamically. This supports exploratory analysis rather than static performance monitoring.

Finally, the system establishes a foundation for future enhancements, including prescriptive analytics, improved data quality processes, and Master Data Management (capabilities not emphasised in the existing platforms).

Background and foundational research

This section discusses theoretical concepts that lay the foundation for the work that is to follow. Relevant academic concepts from data warehousing, business intelligence and behavioural finance are discussed in the next subsections. The section concludes by showing the relationship and tension between BI and behavioural finance. This project applies these concepts alongside each other without seeking to resolve this tension

Data Warehousing Fundamentals

Data Warehousing is defined by (Inmon, 2005) as “a subject-oriented, integrated, time-variant, and non-volatile collection of data in support of managements decision making process”. The subject-oriented feature of a data warehouse means that data is organised around specific real-world business subjects, tying together all relevant data sets. Its integrated nature allows data from various sources to be standardised and stored in a single location. Additionally, the time-variant feature enables data warehouses to maintain a much longer time horizon compared to operational systems, allowing for a historical perspective. Lastly, the non-volatile nature of data warehouses means they do not require transaction processing, recovery, or concurrency control mechanisms (Inmon, 2005).

(Kimball & Ross, 2013) put the following aspects as those the Data Warehouse/Business Intelligence systems concern themselves with, among others:

- It should act as a reliable and credible basis for enhanced decision-making.
- It needs to ensure that information is readily accessible.
- The business community must embrace it for it to be considered successful.

Data warehouses can be perceived as a single integrated data repository that is used for analysis of business performance (business intelligence). At the core of data warehousing is a modelling technique called dimensional modelling.

In a dimensional model, fact tables and dimension tables are key components for organising data to simplify analysis. Fact tables hold quantitative data about business events, such as sales amounts and transaction counts, and include foreign keys linking to dimension tables. Dimension tables contain descriptive attributes that provide context to the facts, like customer names, product categories, and dates. Together, these tables enable fast querying and intuitive reporting, with fact tables offering metrics and dimension tables supplying context for analysis.

Although both dimensional and normalised (3NF) models can be shown using entity-relationship diagrams (ERDs), 3NF models are far more complex (on the contrary, star schemas minimise the number of "joins" a database has to perform). That complexity makes them difficult for users to understand and inefficient for business intelligence queries, which are often unpredictable and data-heavy (Kimball & Ross, 2013).

(Golfarelli & Rizzi, 2018) and (Kimball & Ross, 2013) state that this technique is still the most widely used for its simplicity. Although there have been other techniques being introduced for modelling specific situations (e.g. irregular hierarchies and top hierarchies in social BI (Golfarelli & Rizzi, 2018). Given the primary objectives of this project, the Kimball methodology and its star-schema dimensional model are selected as the most suitable foundation. Its proven track record in simplifying complex data for end-user consumption aligns directly to ensure information is 'readily accessible'.

While these foundational principles remain relevant, the architecture of analytical systems has evolved with the advent of cloud computing, scalable storage, and massively parallel processing. Modern data platforms often decouple storage and compute, and the traditional data warehouse now exists alongside concepts like data lakes (for raw, unstructured data) and lakehouses (blending the structure of a warehouse with the flexibility of a lake). The extraction, transformation and loading (ETL) processes associated with data warehousing will be discussed in detail in Chapter 5: Implementation.

The following section will discuss BI as an essential layer of the data warehouse.

Business Intelligence

Negash et al. (2008) define business intelligence systems as those that “combine data gathering, data storage, and knowledge management with analytical tools to present complex internal and competitive information to planners and decision makers”. The authors assert that business intelligence systems deliver actionable information at the right time, in the right place, and in the appropriate format to support decision-makers.

The goal is to enhance the timeliness and quality of the information used in the decision-making process, thereby facilitating managerial tasks.

Coronel et al. (2020) explain the six basic architectural components that make up the BI framework:

1. ETL tools (Extract, Transform, Load) - These tools collect data from different sources, clean and standardise it, and load it into a central repository. They ensure data is accurate, consistent, and ready for analysis.
2. Data store - This is the central place where processed data is kept, usually a data warehouse or data mart. It is structured for fast querying and historical analysis rather than daily transaction processing.
3. Query and reporting - These tools allow users to ask questions of the data and generate structured reports. They turn stored data into organised information such as summaries, tables, and scheduled reports.
4. Data visualisation - This involves presenting data in graphical formats like charts, graphs, and dashboards. Visualisation helps users quickly understand patterns, trends, and comparisons.
5. Data monitoring and alerting - These systems track data continuously and notify users when certain conditions are met, such as unusual activity or threshold breaches. This supports timely decision-making and risk detection.
6. Data analytics - This is the deeper examination of data to discover patterns, relationships, and insights. It can range from descriptive analysis (what happened) to predictive and prescriptive analysis (what might happen and what should be done).

Business Intelligence (BI) systems offer several key advantages (Olszak & Bartuś, 2025). They enhance operational efficiency by automating reporting and providing real-time data access, allowing organisations to respond quickly to changes. BI also improves decision-making with reliable data-driven insights, minimising errors. Furthermore, BI boosts competitiveness through market analysis and customer behaviour insights, helping organisations adapt their strategies. It aids in cost optimisation by identifying areas for expense reduction and resource management. Lastly, BI systems facilitate fraud and anomaly detection by analysing data in real-time, which supports effective risk management and governance.

Recent advances in BI include its integration with artificial intelligence (AI). In their literature review, Olszak and Bartuś (2025) highlighted that analytics 4.0, which emerged after 2020, represents the era of AI-driven Business Intelligence and augmented analytics. In this stage, BI systems have evolved to incorporate Artificial Intelligence and Natural Language Processing (NLP) in order to enhance analytical capabilities. AI is now used to automatically generate business insights, detect anomalies, and provide real-time decision recommendations. Furthermore, data

analysis processes have become increasingly autonomous, with systems learning from user interactions. As a result, the reliance on advanced analytical expertise among managers has been reduced.

One of the most frequently cited concerns regarding artificial intelligence is its limited ability to clearly explain how decisions are made (Liu, 2025). This lack of transparency can create challenges in anti-money laundering (AML) contexts, where regulatory compliance requires decisions to be traceable, justifiable, and legally defensible. When AI systems operate as "black boxes," there is a risk of misinterpretation, regulatory non-compliance, or reduced trust from oversight bodies.

In addition, several barriers continue to slow the adoption of AI and business intelligence technologies. These include technological limitations, high implementation and maintenance costs, a shortage of skilled professionals, integration difficulties with existing systems, and ongoing security and regulatory concerns (Olszak & Bartuś, 2025). Together, these factors highlight that although advanced analytical capabilities are available, access to them is uneven. This imbalance reflects a growing digital divide, where only well-resourced organisations can fully leverage sophisticated technologies while others struggle to participate at the same level.

The autonomous feature that AI systems integrated into BI frameworks have should not be taken to imply that AI systems should replace managers in the executive role. Instead, Coronel et al. (2020) emphasise that, "[...] the manager must initiate the decision support process by asking the appropriate questions. The BI environment exists to support the manager; it does not replace the management function. If the manager fails to ask the appropriate questions, problems will not be identified and solved, and opportunities will be missed. In spite of the very powerful BI presence, the human component is still at the centre of business technology".

As the BI field continues to evolve, (Olszak & Bartuś, 2025) discovered that AI will continue to play a role, particularly in automation, optimisation and data analysis. This section outlined what BI is. It showed that one of the foundational components of BI is data storage, which includes data warehouses as one of the essential storage architectures. This section outlined the advantages and recent developments of BI. The next section outlines the concept of behavioural finance.

Behavioural Finance

Traditional finance theory, particularly Eugene Fama's work on stock price behaviour, argues that price movements are largely independent and unpredictable. This position laid the foundation for the Efficient Market Hypothesis (EMH), which holds that stock prices quickly and fully reflect available information (Fama, 1965). Under this view, investors are assumed to process information rationally, and it becomes extremely

difficult to consistently outperform the market because any informational advantage is rapidly eliminated.

However, this framework assumes that investors interpret and act on information objectively and without systematic error. This is where the limitations of EMH become evident. While markets may be information-rich, EMH does not fully account for how human decision-makers interpret that information. Critics argue that investors do not rely solely on new data rationally and consistently. Instead, their judgments are influenced by psychological biases and emotional responses. This gap between how information is theoretically processed (in EMH) and how it is processed (by real investors) provides the entry point for behavioural finance.

Behavioural finance shifts the focus from markets as purely efficient information processors to investors as psychologically influenced decision-makers. Tversky and Kahneman proposed Prospect Theory in their work (Prospect Theory: An Analysis of Decision Under Risk , 1979). In this paper, the authors discovered that individuals experience twice as much pain from losing a certain amount of money as they feel from gaining the same amount. The authors explain that individuals display different attitudes toward risk depending on whether outcomes are framed as gains or losses. They argue that people tend to be risk-averse when dealing with potential gains, meaning they prefer a certain, smaller reward over a gamble that offers a higher expected value.

In contrast, when facing potential losses, individuals become risk-seeking: they prefer to take a gamble in the hope of avoiding a sure loss, even when the gamble could result in a worse outcome. This pattern of behaviour is referred to as the reflection effect, and it directly challenges the stable, consistent risk preferences assumed under EMH-based rational models. Another important effect is called the isolation effect. The authors observe that people tend to overlook the similarities between options and instead concentrate on their differences. They often disregard how these similarities, when considered alongside other aspects of the options, can influence the outcome of their decisions. (Tversky & Kahneman, 1981) show how the result of this effect leads to the framing effect, which is the inconsistencies that are observed in decisions when the same problem is framed in different ways.

(Etse, Prince, & Linda, 2020) compile some of the present behavioural biases that influence individual financial decision-making. The authors describe the snakebite effect as the situation where an investor is reluctant to invest due to a recent loss. Furthermore, they explain overconfidence bias as a strong overconfidence in an investor's decisions and skills derived from their experience and the information they have access to. This type of bias has been shown to have negative and positive effects on investors.

Another important bias the authors mention is regret bias. It is described as the feeling an investor experiences after suffering a loss on a previous investment. This anticipation of feeling regret again can lead to regret aversion, where the investor becomes hesitant or avoids making new investment decisions altogether because they fear another negative outcome. These biases illustrate practical mechanisms through which information is filtered, distorted, or avoided altogether, further challenging the EMH view of fully rational information use.

In relation to the previous discussion on Business Intelligence, this section highlights a critical insight: access to high-quality data and analytical tools does not guarantee rational decision-making. While Business Intelligence systems are designed to collect, process, and present information for better decisions, behavioural finance shows that human users may still misinterpret, overweight, underweight, or ignore that information due to cognitive and emotional biases. Therefore, understanding investor behaviour requires not only examining the quality of information provided, but also how individuals psychologically process and act on that information.

In conclusion, this section provided relevant foundational research in behavioural finance. It provided different effects that occur when people are given options to choose from that involve risk. Additionally, certain biases that influence financial decision-making were highlighted.

Relevance to the Project

The preceding sections highlight a fundamental tension between the technological precision of data systems and the inherent fallibility of human judgment. While Business Intelligence (BI) systems aim to deliver accurate, complete, and timely information to support rational decision-making, this objective is continually challenged by the cognitive patterns described in Prospect Theory. Human decision-makers rarely act as the “rational agents” assumed in traditional finance; instead, they are influenced by Loss Aversion, in which the psychological impact of a financial loss is roughly twice as strong as an equivalent gain, and by the Reflection Effect, which can drive risky, loss-recovering behaviours.

This project argues that the divergence between system output and human interpretation is not a failure of BI, but a prompt for a more sophisticated architectural approach. The Subject-Oriented nature of Data Warehousing provides a technical remedy for the Isolation Effect, a bias in which individuals focus on isolated, “framed” events while neglecting broader portfolio context. By implementing Dimensional Modelling, specifically the Kimball Star Schema, the system transcends transactional reporting to provide a multidimensional, subject-centric view of an individual’s financial health.

Ultimately, the value of BI in this project is realised through Dimensional Modelling as a structural intervention. The system does not merely display data; it leverages the rigid, subject-oriented logic of the data mart to anchor users to objective patterns, thereby mitigating distortions caused by Overconfidence, Regret Aversion, and Framing. In this framework, the BI system evolves from a passive reporting tool into an active de-biasing engine, ensuring that financial decisions are guided by the longitudinal truth of the "Subject" rather than the emotional noise of the moment.

Conclusion

This chapter compared existing systems with the proposed system. Unique features were highlighted, and the underlying foundational research was also explored. The next section provides details on the requirements analysis, and models illustrating business processes are also included.

Chapter 3: Requirements Analysis

Introduction

Chapter 2 laid out the outline of the proposed system and its comparison with the existing systems and academic theoretical concepts. This section represents the analysis workflow of the as-is system. Activity diagrams were used to model the core investment, selling, and withdrawal processes to identify transactional events, decision points, and system-generated artefacts that inform the design of the data warehouse.

The project followed an Evolutionary Prototyping model. An initial 'As-Is' activity model was used to identify pain points. During the development phase, an Iterative Design approach was taken, allowing the system architecture to evolve as technical constraints were identified. The final 'To-Be' activity diagram (discussed in Chapter 4) serves as the as-built documentation, ensuring 100% alignment between the model and the functional reality of the FIAAT system.

Process Mapping with Activity Diagrams (As-Is)

This section represents the business processes of the existing system using activity diagrams. The diagrams illustrate how the system operates in practice and how process flow changes in response to different conditions and triggering events (Roff, 2003). The following activity diagrams will provide a high-level overview of the investment process as-is.

The Investment Process

Figure 6 models the end-to-end process of making an investment purchase on the EasyEquities investment platform, from the moment the user decides to invest to the point where the system finalises the transaction. It focuses on capital inflow and asset acquisition. As defined (Cambridge University Press and Assessment, 2026),

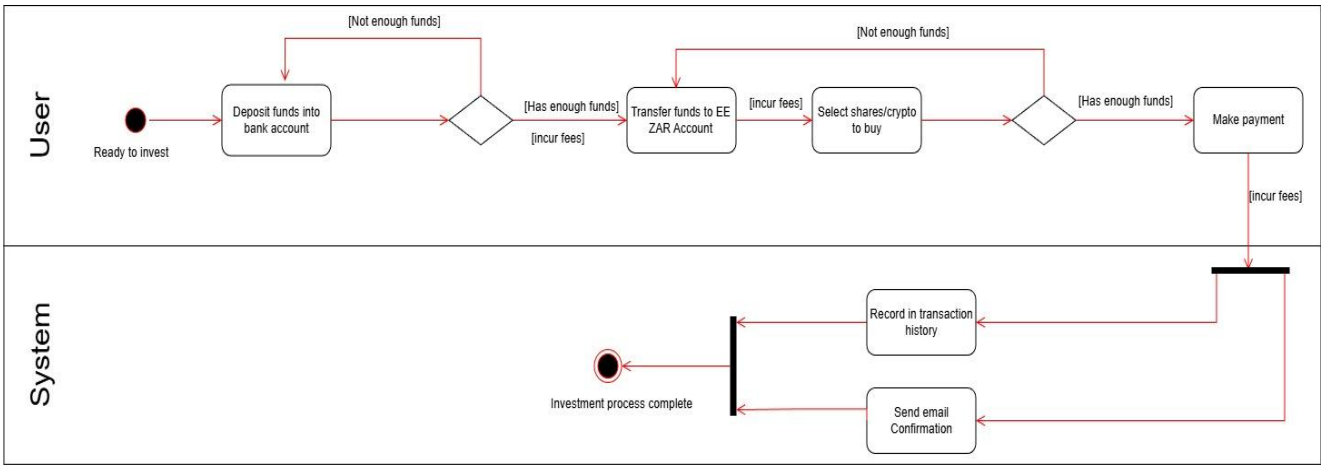


Figure 6: The high-level overview of the investment process

investment means “the act of putting money or effort into something to make a profit or achieve a result”. This definition is particularly relevant here, as it allows the investment process to be interpreted in a behavioural context (considering patterns and dynamics in transactional activity, as discussed in the previous section).

Step-by-step explanation

1. Initial state – Ready to invest (User)

    - This represents the trigger event.
    - The user has decided to invest, but no transaction has yet occurred.

2. Deposit funds into bank account (User)

    - The user ensures liquidity by depositing money into their bank account.
    - This step exists because investment actions depend on available capital.

3. Decision: Has enough funds?

    - The system checks whether the deposited funds are sufficient.
    - If not enough funds, the process loops back to deposit funds again.
    - If enough funds, the process continues.
    - Transfer funds to EasyEquities ZAR account (User)
    - Funds are moved from the bank account to the investment platform’s wallet.
    - This models capital movement, which is important for financial analysis.

5. Select shares/crypto to buy (User)

    - The user selects the financial instruments they want to purchase.
    - This step separates funding from investment choice.

6. Decision: Has enough funds?

    - A second validation occurs.
    - This accounts for:
        - Asset prices
        - Transaction fees
        - Partial balances

7. Make payment (User)

    - This is the atomic transaction where the investment is executed.
    - Fees are incurred at this point.

8. Fork: Parallel system actions (System)

    - After payment, the system performs two actions (which create audit artefacts, which later become data sources) simultaneously:
        - Record in transaction history
        - Send email confirmation

9. Final state – Investment process complete

- The investment has been successfully completed and recorded.

Investment Process Requirements Analysis

Problem Analysis – The main problem in the current investment process is that records of investment events are stored across different platforms, which leads to fragmented and incomplete data. For example, deposit and withdrawal amounts are recorded in bank statements, while transfer fees between EasyEquities accounts are captured in two separate EasyEquities reports: Monthly Statements and the Transaction History Report.

Each data source has limitations. Monthly Statements only cover one month at a time and are password-protected, which slows down access and analysis. The Transaction History Report only covers up to one year, making it difficult to analyse long-term transfer fees. Because each platform charges its own fees and reports them separately, the user cannot easily determine the total cumulative fees paid over time.

Although Capitec and EasyEquities provide basic visual analytics, these views do not show the full picture of all transactions and fees across platforms. As a result, the user cannot perform long-term or holistic analysis of investment performance and/or costs.

To address this problem, data from three main sources, emails, bank statements, and EasyEquities reports, needs to be integrated into a single integrated repository. This integrated solution should include a level of modularisation to reduce the time and effort required to manually collect and consolidate data, while enabling accurate longitudinal analysis of investment activity and fees.

The Selling and Withdrawal Processes

This business process mapping diagram (Figure 77) models capital outflow, covering two parts:

- Selling investments
- Withdrawing funds back to a bank account

The activity diagrams model the core investment, selling, and withdrawal processes in order to identify transactional events, system-generated artefacts, and decision points that inform the design of the data warehouse. They show how value exits the investment platform.

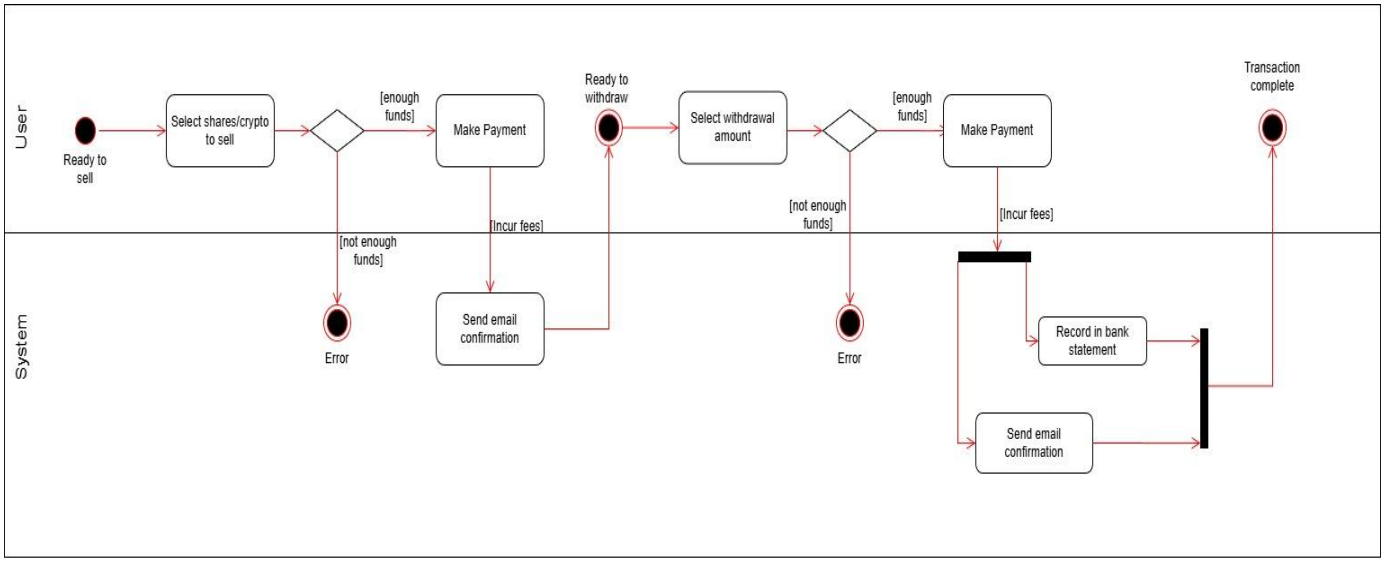


Figure 7: The Investment Selling and Withrawing activity diagrams

Step-by-step explanation

Part A: Selling assets

This process represents the liquidation of investment assets, where a user initiates a sell transaction, the system validates ownership and availability, executes the sale, and confirms the transaction. The outcome of this process is the conversion of investment assets into cash, making funds available for withdrawal.

1. Initial state – Ready to sell (User)
    - The user decides to sell an investment.
2. Select shares/crypto to sell (User)
    - The user chooses which assets to liquidate.
3. Decision: Enough funds?
    - The system checks whether the user owns sufficient quantity of the selected assets.
    - If not enough, the process moves to an error state.
    - If enough, the process continues.
4. Make payment (User)
    - This represents execution of the sell order.
    - Fees are incurred at this stage.
5. System action – Send email confirmation
    - The system notifies the user that the sale has been completed.
6. State transition – Ready to withdraw
    - Selling converts assets into cash, enabling withdrawal.

Part B: Withdrawing funds

This process represents the withdrawal of cash from the investment platform. When a user requests a withdrawal, the system verifies available balances, processes the payment, records the transaction, and issues confirmation. The outcome of this process is the transfer of funds from the platform back to the user's Capitec bank account.

7. Select withdrawal amount (User)
    - The user specifies how much money to withdraw.
8. Decision: Enough funds?
    - The system checks whether sufficient cash is available.
    - If not enough, the process moves to an error state.
    - If enough, the process continues.
9. Make payment (User)
    - The withdrawal transaction is executed.
    - Fees may be incurred.
10. Fork: Parallel system actions (System)
    - The system performs two actions simultaneously:
        - Records the transaction in the bank statement
        - Sends email confirmation to the user
11. Final state – Transaction complete
    - The withdrawal process is completed, and the user receives their funds.

<u>Selling & Withdrawing Requirements Analysis</u>

Problem Analysis- Similar to the previous processes, the main problem encountered in the selling and withdrawal process is the fragmentation of data records.

The activity diagrams documented identify and define all significant financial events captured within the data warehouse by modelling the flow of investment, selling, and withdrawal processes. They explain how transactional data, confirmation emails, and bank statement records are generated as part of normal system operation. By illustrating when these events occur and how they are linked over time, the diagrams provide a clear justification for the adoption of a data-driven dimensional modelling approach.

Building on this understanding of financial events and their flow, the next section develops a comprehensive information package. The activity diagrams help determine the appropriate granularity for the fact tables, whether at the individual transaction, daily summary, or account level, ensuring that each investment, sale, and withdrawal event is captured with the right level of detail. This structured package preserves the temporal and relational context of transactions, enabling accurate reporting, meaningful time-based analysis, and informed financial decision-making

The Information Package

An information package outlines a specific business subject, lists measurable facts related to that subject, and illustrates the various dimensions and their hierarchies, such as year, month, and day within the time dimension (Ponniah, 2010). This approach helps clarify how users intend to view, filter, and summarise the data. By creating information packages, designers can determine the necessary data to be stored, the appropriate level of detail, the potential size of the data warehouse, and the frequency of updates Tables 1-4. Ultimately, it serves as a business-focused blueprint that evolves into fact tables and dimension tables in the data warehouse design.

Key assumptions

- Surrogate keys of integer type with auto-increment values of 1 are used as primary keys for all dimensional tables; therefore, primary keys are not included in the Attributes table.
- The [Date] field in the DimTime entity serves as the primary key, stored as nvarchar(10) in the format “dd/mm/yyyy”. Its values are derived from all transaction dates, with duplicates removed through sorting and filtering.
- The OLTP Field columns in the tables show the origin(s) of the named fields from source files. These source fields originate from the first layer of extraction and are saved in file formats that may include csv, xls, and txt.

Measures Table

- measures are to be categorised into three categories, namely: monetary amounts, counts and time periods.

Table 1: Measures table of the proposed system

Measures Name	Category	OLTP Field	Data Type
Deposit_Amount	Money	[account_statement_31-Dec-2022_to_4-Jan-2026].[Money Out]	Float
Deposit_Fee	Money	[account_statement_31-Dec-2022_to_4-Jan-2026] .[Money Out]	Float
Buy_Amount	Money	[extracted_transactions].[Price]	Float
Buy_Fee	Money	[extracted_transactions].[Fees]	Float
Sell_Amount	Money	[extracted_transactions].[Price]	Float
Sell_Fee	Money	[extracted_transactions].[Fees]	Float

Withdrawal_Amount	Money	[account_statement_31-Dec-2022_to_4-Jan-2026].[Money In] , [Transaction History Report 11_22_2026 7_01_51 AM].[Debit/Credit]	Float
Withdrawal_Fee	Money	[account_statement_31-Dec-2022_to_4-Jan-2026].[Money In] , [Transaction History Report 11_22_2026 7_01_51 AM].[Debit/Credit]	Float

Dimension Table

Table 2: Dimension table of the proposed system

Dimension Name	OLTP Field	Data Type
Account	[account_statement_31-Dec-2022_to_4-Jan-2026].[Account], [Transaction History Report 11_22_2026 7_01_51 AM].[Comment]	Nvarchar
Time	[account_statement_31-Dec-2022_to_4-Jan-2026].[Transaction Date], [Transaction History Report 11_22_2026 7_01_51 AM].[Date], [extracted_transactions].[Date]	Nvarchar, Integer
Entity	[extracted_transactions].[Entity]	Nvarchar

Attributes Table

Table 3: Attributes table of the proposed system

Attributes	OLTP Field	Data Type	Dimension Name
Account_Number	[account_statement_31-Dec-2022_to_4-Jan-2026].[Account], [extracted_transactions].[Acc_No]	Nvarchar	Account
Entity_Name	[extracted_transactions].[Entity]	Nvarchar	Entity
CountryCode	[extracted_transactions].[Currency]	Nvarchar	Entity
Year	[account_statement_31-Dec-2022_to_4-Jan-2026].[Transaction Date], [Transaction History Report 11_22_2026 7_01_51 AM].[Date], [extracted_transactions].[Date]	Integer	Time

Month_Number	[account_statement_31-Dec-2022_to_4-Jan-2026].[Transaction Date], [Transaction History Report 11_22_2026 7_01_51 AM].[Date], [extracted_transactions].[Date]	Integer	Time
Day_Number	[account_statement_31-Dec-2022_to_4-Jan-2026].[Transaction Date], [Transaction History Report 11_22_2026 7_01_51 AM].[Date], [extracted_transactions].[Date]	Integer	Time
Quarter_Number	[account_statement_31-Dec-2022_to_4-Jan-2026].[Transaction Date], [Transaction History Report 11_22_2026 7_01_51 AM].[Date], [extracted_transactions].[Date]	Integer	Time
Month_Name	[account_statement_31-Dec-2022_to_4-Jan-2026].[Transaction Date], [Transaction History Report 11_22_2026 7_01_51 AM].[Date], [extracted_transactions].[Date]	Nvarchar	Time
Day_Name	[account_statement_31-Dec-2022_to_4-Jan-2026].[Transaction Date], [Transaction History Report 11_22_2026 7_01_51 AM].[Date], [extracted_transactions].[Date]	Nvarchar	Time

Information Subject: Investment Account Activity Analysis

Table 4: Information subject of the proposed system

Date	Entity	Account
Year	Name	Acc Number
Month_Number	Country Code	
Day_Number		
Quarter_Number		
Month_Name		
Day_Name		
Facts: Deposit_Amount, Deposit_Fee, Buy_Amount, Buy_Fee, Sell_Amount, Sell_Fee, Withdrawal_Amount, Withdrawal_Fee		

Conclusion

This chapter outlines the requirements workflow by mapping the initial business processes in their as-is state using activity diagrams. This visual representation identified specific processes suitable for automation, integration, and parallelisation to meet the author's data analysis needs.

Each process was further detailed through a step-by-step list, clarifying the actions and decisions at critical points. An information package was developed using an as-built approach to document the current processes. A data-driven methodology was then applied to map existing data and create new data structures. This new structure was decoded to produce a refined information package for effective analysis and decision-making. The next chapter builds on these requirements and defines how the system will achieve its objectives.

Chapter 4: Architectural Design

Introduction

The previous chapter provided insights into how the current system works and what is expected of the FIAT system. This chapter outlines the requirements definition that serves as the foundation for the architectural design. Additionally, it includes the activity diagram of the upcoming system and the fact constellation dimensional model. Furthermore, a dedicated section discusses the justifications for the technologies used to develop the proof of concept.

Functional Requirements

1. Email Data Extraction

- The system shall extract relevant financial information from emails, including transaction notifications, confirmations, and receipts.
- Extracted data shall be saved in a standardised TXT file format for downstream processing.
- The system shall handle emails in common formats (e.g., HTML, plain text) and ignore irrelevant content.

2. File Handling and Preprocessing

- The system shall accept downloaded files from external sources, including bank statements, platform reports, and email-derived TXT files.
- The system shall validate input files for completeness and format consistency before processing.
- The system shall convert semi-structured data into standardised XLS files suitable for ingestion into the data warehouse.

3. Data Transformation and Normalisation

- The system shall transform and normalise extracted data to conform to predefined schemas for both dimension and fact tables.
- Transformation shall include type conversion, merging, date formatting, field mapping, sorting, and redirecting missing or inconsistent values.

4. Data Loading into the Data Warehouse

- The system shall load dimension tables first, ensuring all reference entities (e.g., Entity, Account, Time) exist before transactional data is processed.
- The system shall then load transactional fact tables, including Deposit, Buy, Sell, and Withdrawal events, in the correct sequence.
- The system shall clear or truncate staging and relevant tables before loading new data to prevent duplicates or inconsistent records.
- The system shall maintain transactional integrity and ensure no partial loads occur.

5. Integration with Visualisation Platform (Metabase)

- The system shall connect to Metabase to allow users to visualise data from the data warehouse.
- The system shall support automated verification that the Metabase service is running after the ETL process completes.
- Users shall be able to create, save, and run custom queries in Metabase.
- Users shall be able to generate visual outputs (charts, graphs, tables) based on query results.

6. User Interaction and Reporting

- The system shall provide users with query-ready, validated analytical data for decision support.
- Users shall be able to filter, sort, and aggregate data in Metabase according to business needs.
- The system shall support scheduled or manual execution of ETL workflows as required by the user.

Activity Diagram (To-Be)

The following activity diagram illustrates the end-to-end processes that occur while the system is running. It depicts the normal workflow and excludes error-handling activities, allowing the focus to remain on the core functionality of the system.

The ETL process is layered to separate fragile extraction logic from structured transformation. Python scripts perform light extraction and normalisation when pulling data from emails, handling semi-structured formats and standardising fields such as dates and amounts. The data is then converted into TXT and CSV files, which serve as stable intermediate formats that decouple extraction from database loading. These structured files are finally ingested into staging tables, where SQL applies validation, typing, and relational transformations before loading the warehouse.

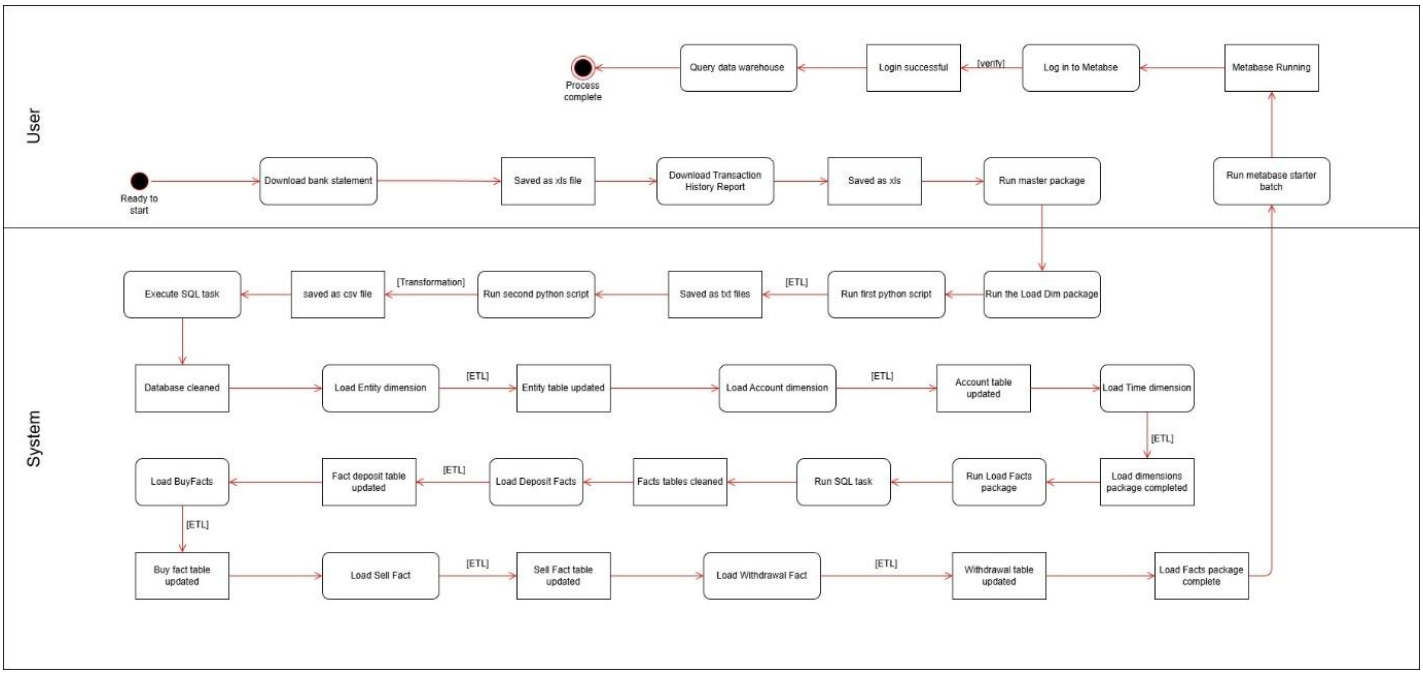


Figure 8: High-level overview of processes that take place in the to-be system

Step-by-step:

1. The process begins with the user manually downloading source data, including bank statements and transaction history reports, from external financial platforms.
2. The downloaded files are saved locally in spreadsheet formats, such as XLS, which act as the raw input sources for the system.
3. The user initiates the FIAAT master ETL package, which coordinates all downstream extraction, transformation, and loading activities.
4. The system executes Python scripts that perform light extraction and normalisation, converting semi-structured financial data into standardised TXT files.
5. The normalised TXT files are further converted into CSV format to create structured, tabular datasets suitable for database ingestion.
6. Before loading new data, the system clears or truncates staging and relevant fact tables to prevent duplicate or inconsistent records.
7. Dimension tables are loaded first, beginning with core entities such as Entity, Account, and Time, ensuring that all reference data exists before transactional data is processed.
8. Once dimensions are in place, the system loads transactional fact tables in sequence, including Deposit, Buy, Sell, and Withdrawal events.

9. After the ETL process completes successfully, the Metabase service is started or verified as running.
10. The user logs into Metabase and queries the data warehouse, enabling analysis, monitoring, and reporting of financial activity.
11. The process concludes with validated, query-ready analytical data available for decision support and intelligence extraction.

Dimensional Model Design

In this project, data-driven dimensional modelling was employed to structure the data mart for intuitive analysis and reporting. Relevant facts and dimensions were identified through analysis of source data, assessment of data quality, and consideration of user query and reporting requirements. A star schema was implemented to align the structure with actual reporting and aggregation patterns, ensuring efficient multidimensional analysis.

A fact constellation schema is an extension of the star schema used in dimensional modelling. In this design, multiple fact tables share common dimension tables, forming a network or "constellation" of interconnected stars. Each fact table represents a different business process or event, but the shared dimensions provide a consistent descriptive context across all processes (Ponniah, 2010).

This approach is particularly useful when an organisation needs to analyse multiple related business activities while maintaining a unified view of key dimensions such as time, customer, product, or location. By allowing fact tables to reuse dimensions, a constellation schema supports complex reporting and cross-functional analysis

without duplicating data, while still preserving the efficiency and intuitiveness of the star schema.

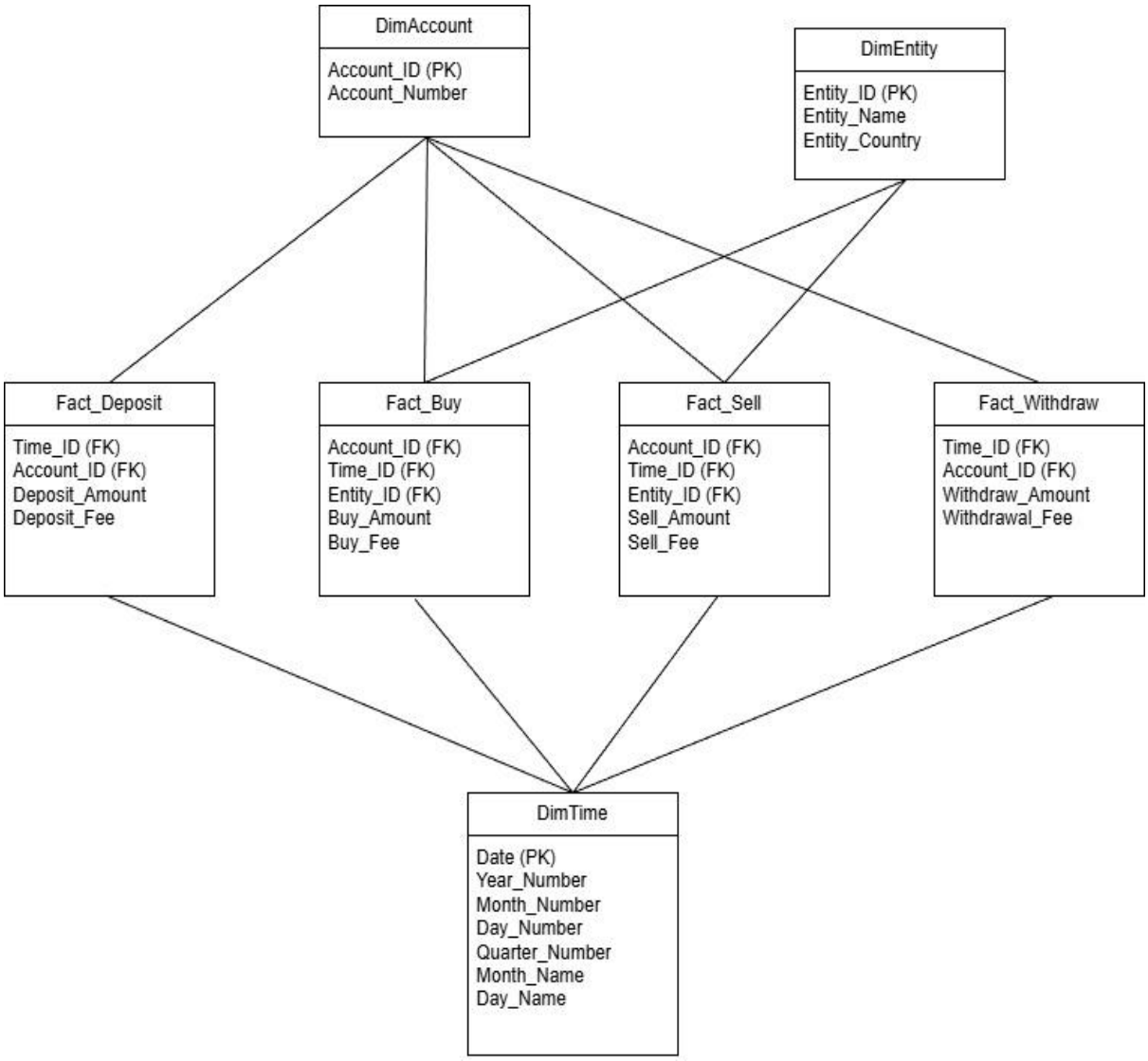


Figure 9: Fact constellation schema

Fact Tables (Quantitative Events)

Fact_Deposit - acts as the central repository for all incoming fund transactions. It captures specific numerical data points, such as the total amount deposited and any associated service fees, which allow analysts to calculate total cash inflows and net deposits across various accounts.

Fact_Buy - tracks the acquisition of assets or services within the system. By storing the purchase amount and the transaction fee, this table enables the business to monitor spending patterns and the total cost of acquisition over time for different entities or account types.

Fact_Sell - documents the liquidation of assets or sales activities. It records the gross sell amount and the fees deducted during the transaction, providing the necessary metrics to analyse revenue generation and the impact of transaction costs on net profitability.

Fact_Withdraw - records every instance where funds are removed from the system. This table focuses on the volume of outflows by tracking the withdrawal amount and the processing fees, which are critical for monitoring liquidity and understanding user behaviour regarding fund removal.

Dimension Tables (Descriptive Context)

DimAccount - provides the identity for the financial containers involved in the transactions. It stores unique identifiers like account numbers, allowing the organisation to group and filter the data from the fact tables to see activity at the individual account holder level.

DimEntity - contains the metadata regarding the individuals performing the transactions. By storing attributes like entity names and countries of origin, it allows the business to perform geographical analysis and evaluate the performance or risk profile of specific market participants.

DimTime - serves as the universal reference for all temporal analysis. It breaks down raw dates into granular levels such as years, quarters, and months, enabling the "drill-down" reporting necessary to identify seasonal trends and compare financial performance across different fiscal periods.

Justification of tools used

Python

Python is a programming language that is easy to read and write (Python.org, 2026). This makes it a good choice for beginners, and it allows developers to create programs quickly. Python runs code line by line and organises data and functions using objects. It comes with many built-in tools and data types, making it convenient for building applications or connecting different software pieces. Its flexible typing system lets programmers write code without having to specify data types.

Python encourages modular programming, which means code can be broken into reusable pieces called modules and packages (Python.org, 2026). Finally, Python and its large standard library are free to use on all major computer systems.

Python’s dominance over its competitors stems primarily from its exceptional readability and high-level abstraction (AppDesk Services, 2026). Unlike languages like C++ or Java, which often require complex memory management and boilerplate code,

Python allows developers to focus on logic and problem-solving through a clean, "English-like" syntax. This simplicity reduces the barrier to entry for beginners and significantly accelerates development cycles for professionals, making it the preferred choice for rapid prototyping and data-driven industries.

While alternatives like Node.js offer superior non-blocking performance for real-time apps, or Go provides faster execution as a compiled language, Python's dynamic semantics and massive standard library offer unmatched versatility. It bridges the gap between simple scripting and complex backend engineering. Whether it's data science, web development, or automation, Python remains the "glue language" of choice because it prioritises the developer's time and mental bandwidth over machine-level micro-optimisations.

The author used Google Gemini AI-generated code to connect to the Gmail account via API to extract data

Table 5: Comparison of Python with alternatives

Feature	Python	Node.js	Java	Go (Golang)
Primary Strength	Readability & AI/ML	Real-time I/O	Enterprise Security	Concurrency/Speed
Execution	Interpreted	Event-driven (JS)	Compiled (JVM)	Compiled
Learning Curve	Very Low	Moderate	Moderate/High	Low/Moderate

SSIS

SQL Server Integration Services (SSIS) is an enterprise-grade platform designed to handle complex data integration and transformation tasks. It is primarily used to automate business processes such as bulk file transfers, data warehouse loading (ETL), data cleansing, and the management of SQL Server objects (Microsoft, 2024). By providing a bridge between disparate systems, it allows organisations to consolidate information from XML files, flat files, and relational databases into a unified destination.

The platform distinguishes itself with a robust suite of built-in tasks and graphical development tools that enable users to build sophisticated data pipelines without writing code. For more advanced requirements, SSIS offers a programmable object model for custom task development. All workflows are managed and executed through the centralised SSIS Catalogue database, ensuring a secure and scalable environment for enterprise data movement.

Opting for SQL Server Integration Services (SSIS) rather than contemporary alternatives is frequently a choice driven by factors such as cost efficiency, control over on-premises data, and strong integration within the existing ecosystem (Tobin, 2026). Although cloud-native solutions like Integrate.io or Fivetran provide quick deployment, they can lead to notable ongoing expenses and security challenges that SSIS avoids by being kept "in-house."

The key advantage of opting for SSIS is its licensing structure; for organisations already utilising SQL Server, SSIS is effectively “complimentary,” as it comes with the current server license. This sharply contrasts with the expensive monthly charges associated with SaaS platforms or the concealed infrastructure and maintenance costs tied to open-source options like Apache Airflow (Tobin, 2026). Additionally, SSIS grants a substantial level of detailed control through custom scripting in C# or VB.NET, enabling developers to manage specific business logic that "no-code" platforms frequently find challenging to address.

Table 6: Comparison of SSIS with alternatives

Feature	SQL Server (SSIS)	Integrate.io / Fivetran	Apache Airflow	Talend / Informatica
Pricing Model	Bundled (No extra cost)	High SaaS (Usage/Flat)	Free (High Ops cost)	Enterprise (High-tier)
Skill Required	GUI / Basic SQL	No-Code / Drag-and-drop	High (Python / DevOps)	Moderate (Java / SQL)
Deployment	On-Prem / Hybrid	Cloud-Only (SaaS)	Cloud / Self-Hosted	On-Prem / Cloud
Customization	C# / VB.NET Scripting	Limited by Connectors	Unlimited (Pure Python)	High (Java Generation)
Best For	Microsoft-centric Orgs	Rapid Cloud Connectivity	Complex Data Engineering	Multi-Cloud Enterprises

Microsoft SQL Server

Microsoft SQL Server is a powerful relational database management system (RDBMS) that uses T-SQL for data interactions (Microsoft, 2026). It features a Database Engine for high-demand transactions and includes tools like Integration Services (SSIS) for ETL, Analysis Services (SSAS) for business intelligence, and Machine Learning Services to integrate R and Python.

For data management, SQL Server offers Replication for data synchronisation and Reporting Services (SSRS) for report delivery. Although Data Quality Services (DQS) and Master Data Services (MDS) are deprecated in recent versions, SQL Server still ensures high availability, optimised query performance with the Query Store, and robust memory management for reliability and scalability.

Choosing Microsoft SQL Server over its competitors is primarily a decision to favour seamless integration and developer productivity over the modular, "build-it-yourself" nature of open-source alternatives. While engines like PostgreSQL or MySQL are excellent for cost savings, they lack the unified ecosystem of MSSQL, where enterprise-grade ETL (SSIS), analytics (SSAS), and reporting (SSRS) are native, interoperable components rather than disparate third-party tools (DSouza, 2024). This "all-in-one" architecture significantly reduces the time spent on configuration and troubleshooting, allowing teams to focus on data insights rather than system architecture.

Furthermore, MSSQL provides a superior administrative experience through its industry-leading management tools and automated performance tuning. Features like the Query Store and Always On Availability Groups offer a level of out-of-the-box observability and disaster recovery that is often more complex and manual to implement in Oracle or MongoDB (DSouza, 2024). By combining high-performance relational processing with modern support for JSON and Graph data, SQL Server bridges the gap between traditional reliability and modern flexibility, making it the more stable choice for mission-critical enterprise workloads.

Table 7: Comparison of Microsoft SQL Server with alternatives

Feature	Microsoft SQL Server	MySQL / PostgreSQL	Oracle Database	MongoDB / Firebase
Primary Edge	Integrated BI & Tooling	Low Cost / Open Source	Absolute Peak Power	Schema Flexibility
Ease of Use	High (Best-in-class UI)	Moderate (CLI focused)	Low (Very Complex)	High (Developer-friendly)
Data Integrity	Strict ACID Compliance	ACID (with config)	Strict ACID Compliance	Eventual Consistency
Best For	Enterprise Windows/Azure	Web Apps & Startups	Massive Legacy Systems	Real-time / Unstructured

Metabase Data visualisation platform

Metabase is an open-source business intelligence tool designed to make data exploration and visualisation accessible to users with various levels of technical

expertise (Bouchrika, 2026). Its intuitive interface allows users to create interactive dashboards and charts without requiring advanced coding skills, while also accommodating direct SQL queries for more experienced analysts. This flexibility makes Metabase an ideal option for organisations looking for a simple yet effective analytics solution.

A key advantage of Metabase is its capability to connect with various relational databases like MySQL, PostgreSQL, and SQL Server. This wide compatibility enables companies to seamlessly integrate the tool into their current data infrastructure. Users can create data queries visually or opt to write SQL, accommodating different preferences and expertise levels, which promotes accessibility throughout different departments (Bouchrika, 2026).

Choosing Metabase over its alternatives highlights a preference for speed and accessibility over technical complexity. Unlike Tableau and Power BI, which require specialised analysts and create "data bottlenecks," Metabase empowers non-technical users to quickly generate insights with its intuitive "Question Builder." (Thompson, 2025)

Compared to more technical platforms like Looker and Apache Superset, Metabase excels in ease of deployment. Looker demands a "code-first" approach with a steep learning curve, while Metabase allows direct querying of databases without upfront modelling (Thompson, 2025). Even against Superset, Metabase is favoured for its polished, user-friendly interface, making it more appealing to teams that prioritise quick insights over complex customisation.

Table 8: Comparison of Metabase with alternatives

Feature	Metabase	Tableau	Power BI	Looker
Ideal User	Everyone (Self-Serve)	Pro Data Analysts	Microsoft Power Users	Data Engineers
Setup Time	< 5 Minutes	Days / Weeks	Days / Weeks	6+ Weeks (LookML)
Data Strategy	Direct Query	Extract (Hyper)	Extract / Hybrid	Live (SQL-only)
Pricing	Free (OSS) / Low-cost	High (Per user)	Low entry / High scale	Very High / Opaque

Conclusion

This chapter commences by delineating the essential functional requirements, providing a robust foundation for understanding the system's objectives. It then unfolds a thorough as-built design of the key system processes, meticulously detailing each step of the to-be system.

Furthermore, it presents a comparative analysis of the technologies employed relative to their alternatives, elucidating the specific conditions and scenarios that warrant the selection of these particular tools. This comprehensive approach not only clarifies the decision-making process but also highlights the strategic choices made throughout the project. The next chapter will discuss the implementation

Chapter 5: Implementation

Introduction

The previous chapter laid out a blueprint for how the system should achieve its objectives by detailing high-level process mapping via an activity diagram and dimensional model to establish the logical data environment. This chapter starts by outlining the environment under which the FIAT MVP was developed. In addition, the implementation of the key extraction and transformation processes is thoroughly explained, aided by pictures.

Development Environment

The system's proof of concept was developed under the following computer requirements specification:

- Windows 11
- 8 GB RAM
- AMD RYZEN 3 Processor
- 237 GB
- System Type: 64-bit operating system, x64-based processor
- 5G Wi-Fi Internet Access

All the applications, their dependencies and packages required to successfully develop this system are listed in the Appendices: B and C.

Python (Scripts and API)

Python scripts were developed to extract email content via an application programming interface (API) and store the retrieved information as text files. The implementation made use of several Python libraries: requests was used to manage HTTP communication with the API, json was used to parse structured response data, and Python's built-in file input/output functionality was used to persist the extracted content in plain text format. The specifications used to generate Python code with the assistance of Google Gemini AI, as well as the associated safety considerations, are provided in Appendix A.

The project initially intended to use SAS for data mining; however, access was limited to SAS Studio for Academics rather than the full SAS Enterprise environment. This restriction limited the ability to perform the required data extraction and integration tasks. Consequently, Python was adopted as the primary tool for data acquisition and preprocessing due to its flexibility and extensive ecosystem of data-processing libraries.

Key data elements were further extracted and structured using the pandas library, which enabled the transformation of raw data into tabular form. These structured datasets were exported as CSV files to support subsequent stages of analysis and integration.

Layered Approach to Extraction, Transformation and Loading

A layered architecture was adopted to implement the end-to-end extraction, transformation, and loading (ETL) process.

In the first layer, Python was used to handle the extraction and preliminary processing of unstructured and semi-structured data. Email content was retrieved via an API and stored as text files. Supporting libraries included requests for API communication, json for parsing structured responses, BeautifulSoup for processing HTML-based email content, and pandas for converting extracted information into structured tabular datasets. These datasets were exported as CSV files to serve as intermediate data sources for downstream processing.

In the second layer, SQL Server Integration Services (SSIS) within Visual Studio was used to perform structured ETL operations for populating the data mart (next section). Multiple data sources, including CSV, Excel, and text files, were integrated with the target database. SQL scripts and SSIS data flow components were used to clean, transform, and standardise the data according to the schema and quality requirements of the data mart.

A modular ETL design was implemented. Separate SSIS packages were developed for dimension tables and fact tables, allowing transformations specific to each table type to be applied independently (Figures 9 & 10). A master package coordinated the execution of these packages in the appropriate sequence, ensuring that dimensions were loaded before related fact tables. This orchestration improved maintainability, reusability, and process reliability.

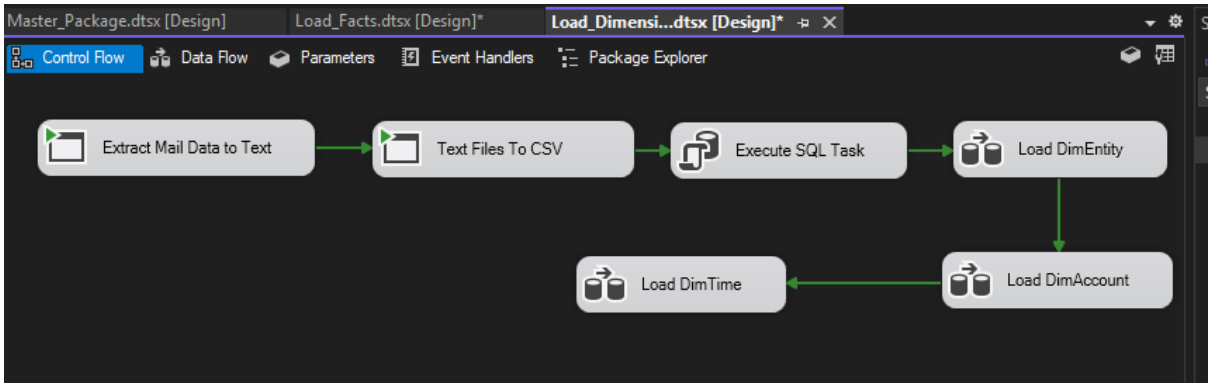


Figure 10: The Load Dimension data pipeline module

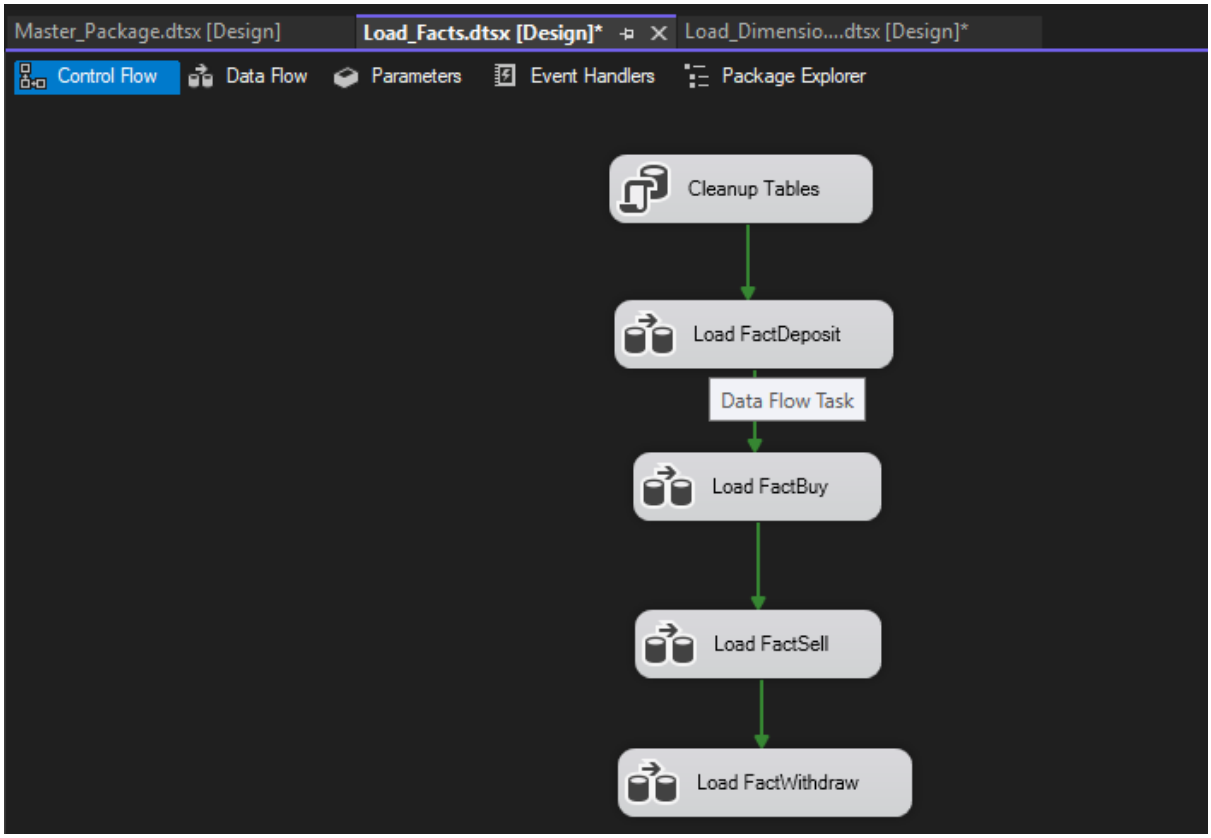


Figure 11: The Load Fact data pipeline module

This layered methodology enabled Python to manage the flexible extraction and preprocessing of raw data, while SSIS provided a controlled and scalable framework for structured data integration. Together, these tools supported a fully automated pipeline, transforming raw source data into standardised, analysis-ready structures within the data mart.

SQL Queries and the Related Process Logic

1. Investment Transaction Date Dimension Derivation

To generate a structured set of calendar attributes from investment-related transactions for use in time-based analysis and reporting.

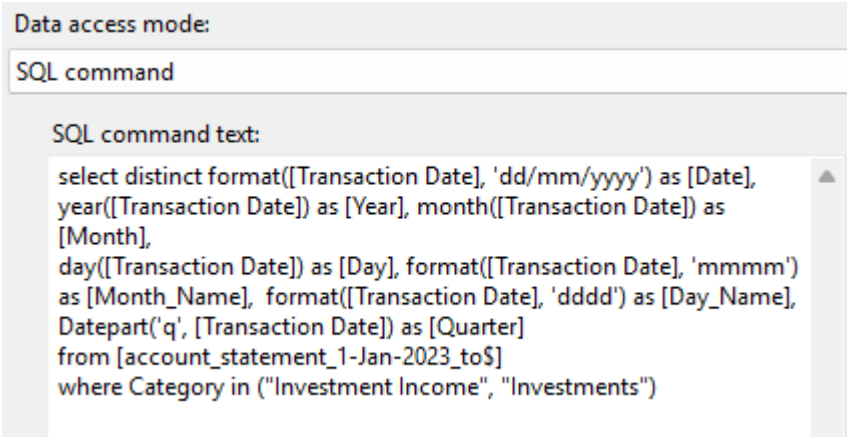


Figure 12: Screenshot of the sql code used to extract one of the dimensions from the bank statement source

Process Logic

The query filters transaction records to include only those categorised as Investment Income or Investments. From each qualifying transaction date, multiple temporal attributes are derived, including the formatted date, year, month, day, month name, day name, and quarter. Duplicate date records are removed using a distinct selection. The result is a normalised set of date-related attributes suitable for populating a Date Dimension table.

2. Fact Table Initialisation (Data Reset)

To prepare the data warehouse fact tables for a fresh data load by removing previously stored transactional summaries.

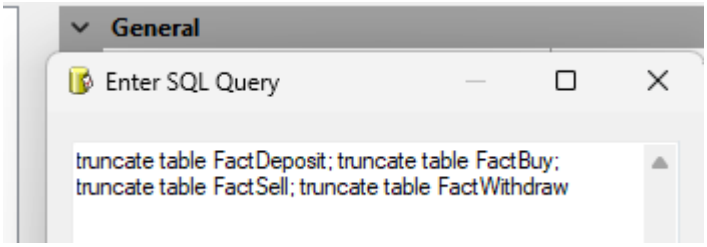


Figure 13: The SQL code cleaning fact tables before the Load Facts module is executed

Process Logic

The process executes a series of truncate operations on the FactDeposit, FactBuy, FactSell, and FactWithdraw tables. Truncation removes all existing records while preserving table structures, ensuring that subsequent ETL loads insert only current, non-duplicated data.

3. Transaction History Normalisation and Event Classification

To transform raw transaction history logs into structured financial events with clearly separated transaction amounts and fees (see Figure 13 below).

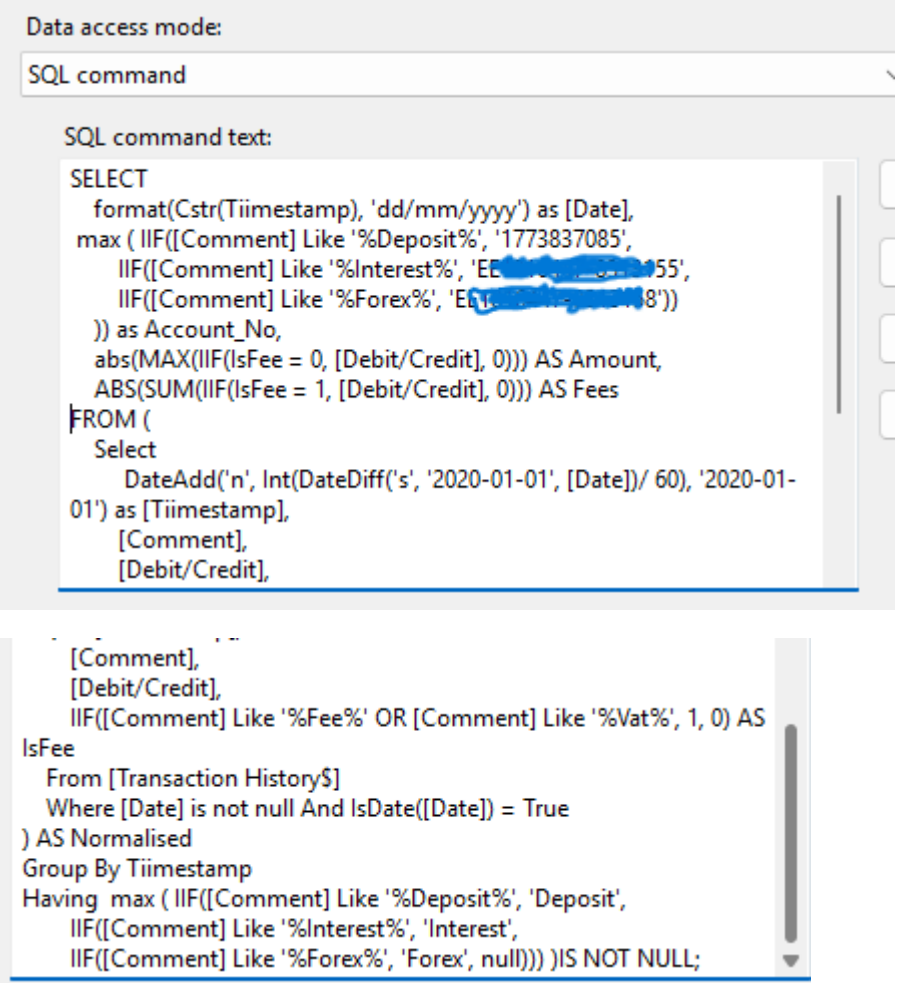


Figure 14: Transaction History Normalisation and Event Classification

Process Logic

The inner query first validates transaction dates and standardises timestamps by rounding them to the nearest minute. It also flags fee-related entries based on keywords in the transaction comments.

The outer query groups records by the normalised timestamp, effectively consolidating related entries into single financial events. Transaction types are inferred from comment text (e.g., Deposit, Interest, Forex), and each group is assigned a corresponding account classification code. The principal transaction amount is calculated from non-fee entries, while total fees are calculated from fee-flagged entries. Only transactions with identifiable classifications are retained.

4. Buy Transaction Standardisation with Currency Conversion

To prepare purchase transactions for analysis by standardising monetary values into a single reporting currency.

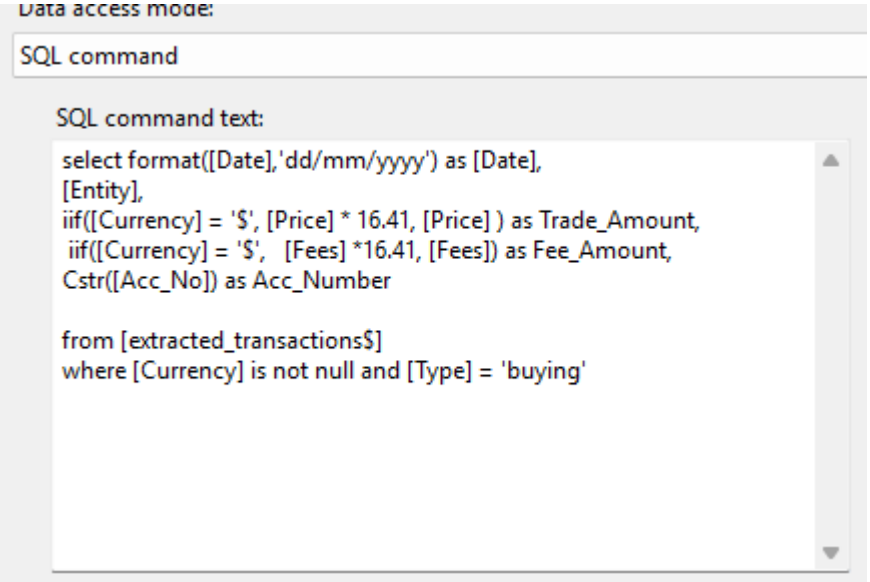


Figure 15: Buy Transaction Standardisation with Currency Conversion

Process Logic

The query filters for transactions labelled as buying and with a valid currency value. Trade prices and associated fees are conditionally converted into local currency using a fixed exchange rate when the original currency is USD. Account numbers are standardised as text values. The output provides normalised buy transaction records with consistent monetary units.

5. Currency-Based Market Classification

To derive a geographic or market segment indicator from transaction currency.

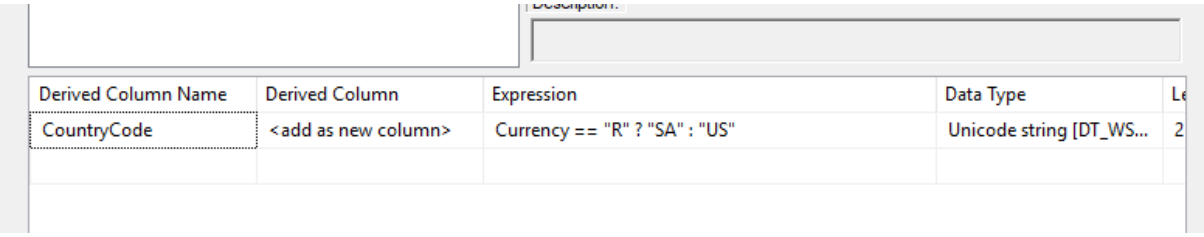


Figure 16: Currency-Based Market Classification

Process Logic

A conditional rule evaluates the currency code of each transaction. Transactions conducted in Rand are classified as South African (SA), while all others are classified as United States (US). This derived attribute supports regional segmentation in reporting and analysis.

6. Sell Transaction Standardisation with Currency Conversion

To standardise selling transactions into a consistent reporting currency for comparative financial analysis.

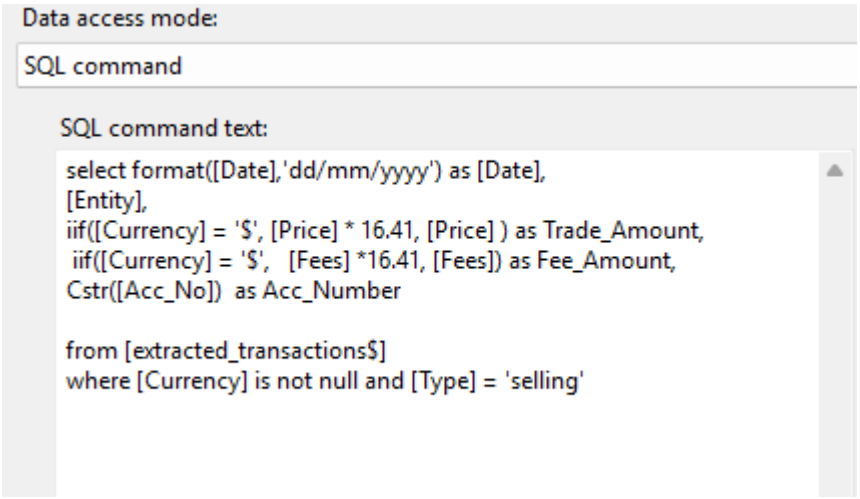


Figure 17: Sell Transaction Standardisation with Currency Conversion

Process Logic

The query selects transactions labelled as selling and ensures that monetary values are expressed in a single currency. Transactions in USD are converted using a fixed exchange rate, while local currency transactions remain unchanged. Fees undergo the same conversion logic. Account identifiers are standardised as text fields to maintain consistency across fact records.

7. Investment Income Extraction

To isolate and structure records representing income generated from investments.

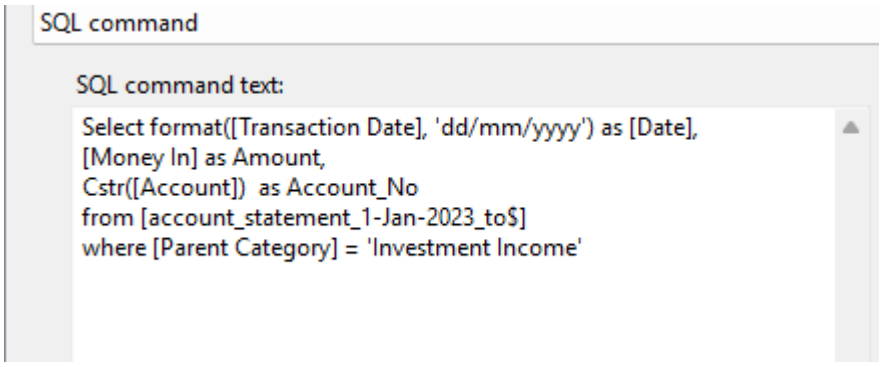


Figure 18: Investment Income Extraction

Process Logic

The query filters transactions where the parent category is explicitly Investment Income. For each qualifying record, the transaction date is formatted, the incoming monetary

value is recorded as the amount, and the account number is standardised. The resulting dataset represents structured investment income events suitable for loading into an income-related fact table.

Information Delivery

A batch script with commands to initiate the Metabase platform was created. This script is executed every time the user wants to use the data visualisation platform. The following sections discuss the data visualisation and the insights from the data source.

The user is required to have an account that has permissions to access the database through the platform. More details about how to connect to the DBMS are in Appendix C.

Top 5 most valuable assets based on their average exit price

The Top 5 most valuable assets based on their average exit price chart compares the average buying price versus the average selling price for five entities. It shows how much the assets appreciated between entry and exit. Large gaps between the buy and sell bars indicate strong price growth and good trade timing, while smaller gaps suggest limited upside. This graph helps you see which assets tend to sell at much higher prices than they were bought, highlighting strong performers in terms of price appreciation rather than total profit.

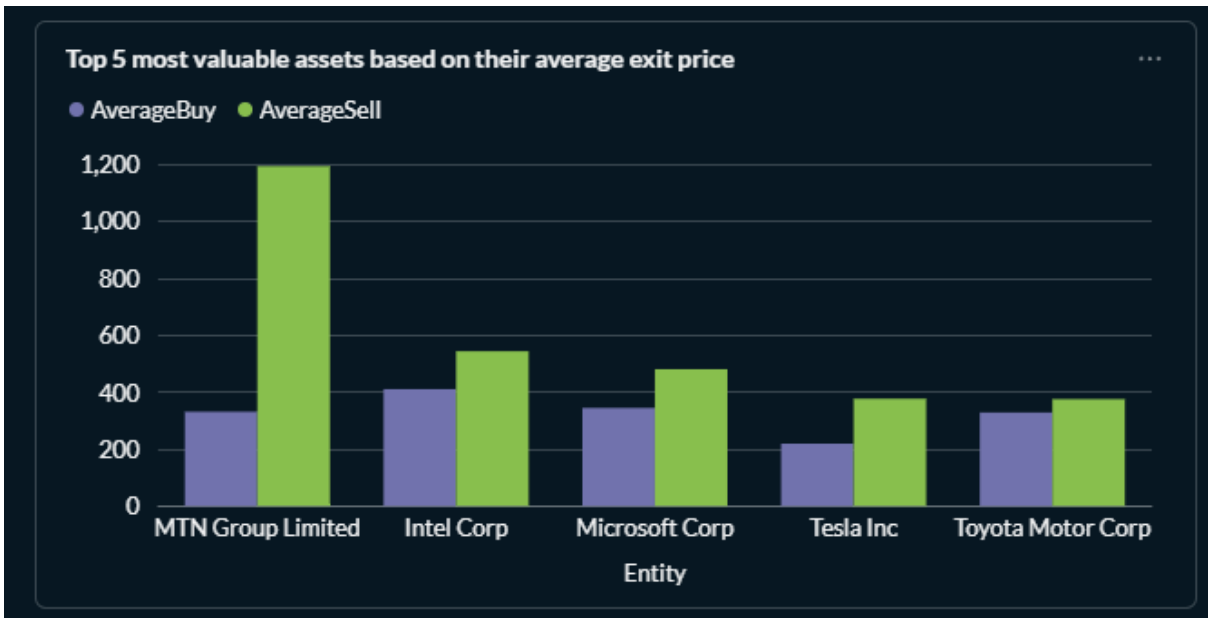


Figure 19: Top 5 most valuable assets based on their average exit price

Top 10 most profitable entities

The Top 10 most profitable entities chart ranks companies based on trading profitability. It displays gross profit, total transaction fees, net profit after fees, and the percentage impact of fees. This allows you to see not just where you made money, but where costs reduced your gains. Some entities may show high gross profit but also high fees, lowering their net contribution. This graph is key for identifying which investments truly add value after accounting for trading costs.

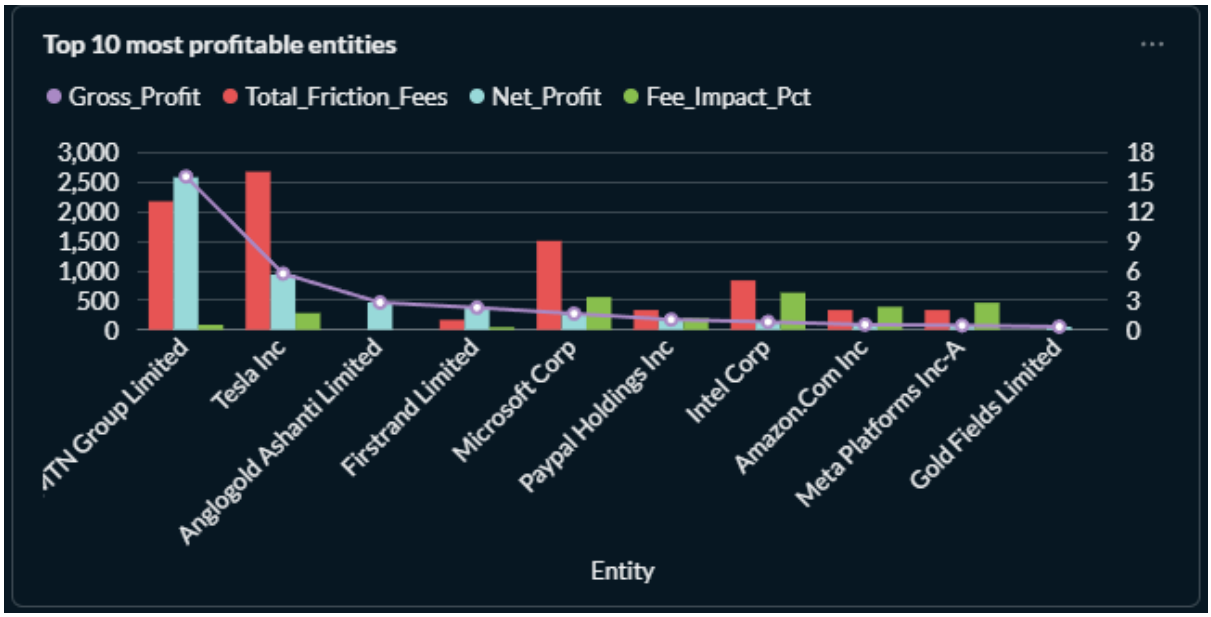


Figure 20: Top 10 most profitable entities

The Top 5 assets that were bought but never sold

The Top 5 assets that were bought but never sold donut chart highlights capital that is currently tied up in open positions. It shows which assets you invested in but have not exited, along with the total value locked in those holdings. A large concentration in one or two assets may indicate reduced liquidity or long-term holding strategies. This visual helps you understand opportunity cost and portfolio exposure in unsold investments.

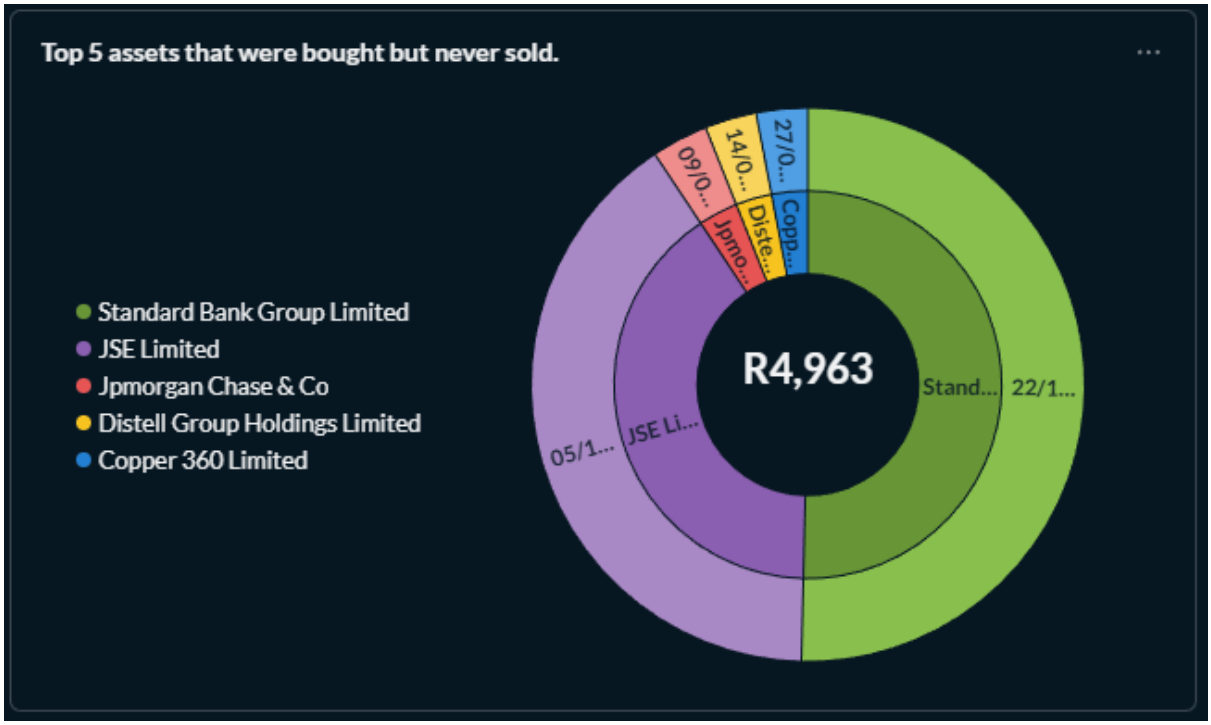


Figure 21: The Top 5 assets that were bought but never sold

Which periods did my trades make the most money?

The Which periods did my trades make the most money line chart analyzes performance over time. It tracks total realized gains alongside average percentage return by month. Peaks show periods where your trades were highly profitable, while dips or negative areas reveal weaker performance months. Comparing gain and return together helps distinguish between months where you made a lot due to larger trade sizes versus months where trades were more efficient in percentage terms. This chart is useful for spotting seasonal trends and improving trade timing.

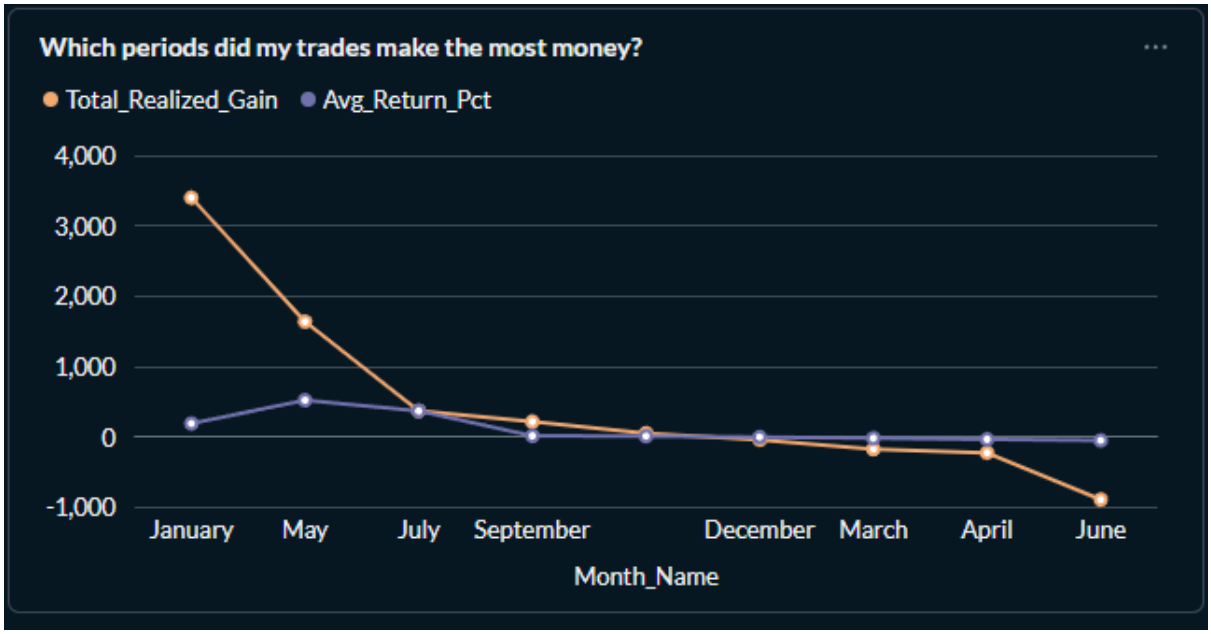


Figure 22: Which periods did my trades make the most money?

Reflections

Much of the initial work was conducted with minimal documentation, which presented its own set of challenges. The author prioritised practical outcomes over formal recording, focusing primarily on identifying effective strategies and noting what did not yield results. For instance, the main visualisation tool initially intended for use was Grafana dashboards. However, after several unsuccessful attempts to achieve the desired outcomes with Grafana, the author took the initiative to research and evaluate alternative platforms. Through this exploration, Metabase was discovered to be an outstanding option, as it facilitated the democratisation of data access and analysis from a firm's perspective, empowering users across various levels of the organisation to make informed decisions.

The author adopted an Agile methodology for this project, as it promotes flexibility and allows for the exploration of unfamiliar technologies while consistently delivering incremental value. Serving the dual role of both client and developer posed specific challenges, particularly in accurately deriving detailed and precise requirements. This duality led to limited documentation in the project's early stages, prompting the author to focus on rapid prototyping to develop workable solutions instead.

One of the most significant hurdles faced during the transformation process was consolidating transactions from multiline journal entries into a single, coherent fact

row. This consolidation involved aggregating associated fees and taxes, which proved to be technically demanding. The complexities of ensuring data integrity and accuracy during this process were pivotal to the overall success of the project, making it the most challenging aspect of the transformation journey.

From the visualisations, the fact that some missing data are unaccounted for raises doubts about the accuracy and completeness of the data. Nevertheless, the outcomes have proven that the system is technically feasible, with the core functionalities performing as expected. While the justifications for selecting the chosen technologies were analysed thoroughly, many were ultimately influenced by the author's familiarity and prior experience with them. For instance, SSIS, a robust tool designed for enterprise-grade systems, stands out as an example of a technology leveraged due to its established presence in larger organisations. In contrast, Metabase, which proves to be particularly effective for smaller firms, highlights the diversity in technology choices based not only on capability but also on the specific needs and scale of the business, alongside practicality.

Future improvements envisioned for the system include:

1. Advanced analytics
2. More enhanced data quality by accurately capturing from sources
3. SARS ITR12 auto-fil feature
4. Improving and upgrading data privacy and protection through safer data storage, processing and manipulation.
5. More detailed sequence diagrams and other related modelling diagrams
6. More descriptive statistical analytics based on the theoretical concepts of Behavioural Finance

References

AppDesk Services. (2026, January). *Top 8 Python Alternatives*. Retrieved from https://appdeskservices.com/python-alternatives/

Bouchrika, I. (2026, January 29). *Metabase Review 2026: Pricing, Features, Pros & Cons, Ratings & More*. Retrieved from Research.com: https://research.com/software/reviews/metabase

Cambridge University Press and Assessment. (2026). *Cambridge Dictionary*. Retrieved January 21, 2026, from Meaning of investment in English: https://dictionary.cambridge.org/dictionary/english/investment

Capitec. (2026). *About us*. Retrieved January 26, 2026, from Capitec.com: https://www.capitecbank.co.za/about-us/

Coronel, C., Morris, S., Crockett, K., & Blewett, C. (2020). Business Intelligence Architecture. In C. Coronel, S. Morris, K. Crockett, & C. Blewett, *Database Principles: Fundamentals of Design, Implementation and Management* (pp. 753-758). Hampshire,: Marinda Louw. Retrieved January 06, 2026

DSouza, D. (2024). *Top 10 Microsoft SQL Server Alternatives in 2026*. Retrieved from Hevodata: https://hevodata.com/learn/top-sql-server-alternatives/

Easy Equities. (2026). *Platform Features*. Retrieved January 30, 2026, from EasyEquities.co.za: https://www.easyequities.co.za/platform-features

Etse, N., Prince, G., & Linda, S. (2020, October). Behavioural Finance and Investment Decisions: Does Behavioral Bias Matter? *International Business Research*. Retrieved January 31, 2026, from https://doi.org/10.5539/ibr.v13n11p65

Fama, E. F. (1965, January). Behaviour of Stock Market Prices. *The Journal of Business*, 34-105. Retrieved January 31, 2026, from https://www.academia.edu/1232064/The_behavior_of_stock_market_prices

Golfarelli, M., & Rizzi, S. (2018, May). From Star Schemas to Big Data:20+ Years of Data Warehouse Research. doi:10.1007/978-3-319-61893-7_6

Gulati, J. (2025, May 08). *Understanding Effective Data Presentation Techniques*. Retrieved January 30, 2026, from Statology.org: https://www.statology.org/understanding-effective-data-presentation-techniques/

Inmon, W. H. (2005). *Building the Data Warehouse.* John Wiley & Sons. Retrieved January 26, 2026, from

https://books.google.co.za/books/about/Building_the_Data_Warehouse.html?id=QFKTmh5IFS4C&redir_esc=y

Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling.* John Wiley & Sons, Inc. Retrieved January 26, 2026

Microsoft. (2024, September 26). *SQL Server Integration Services*. Retrieved from Microsoft: https://learn.microsoft.com/en-us/sql/integration-services/sql-server-integration-services?view=sql-server-ver17

Microsoft. (2026, January 28). *What is SQL Server?* Retrieved from Microsoft.com: https://learn.microsoft.com/en-us/sql/sql-server/what-is-sql-server?view=sql-server-ver17

Negash, S., Gray, P., Burstein, F., & Holsapple, C. W. (2008). Business Intelligence. In S. Negash, P. Gray, F. Burstein, & C. W. Holsapple, *Business Intelligence. Handbook on Decision Support Systems 2:* . Communications of the Association for Information Systems. doi:10.1007/978-3-540-48716-6_9

News24. (2025, October). *Capitec hits 25 million clients as profits surge 26%*. Retrieved January 26, 2026, from News24: https://www.news24.com/brandstory/partner-content/capitec-hits-25-million-clients-as-profits-surge-26-20251002-0509

Olszak, C. M., & Bartuś, K. (2025). AI-enhanced Business Intelligence for decision-making. *Procedia Computer Science* . doi:https://doi.org/10.1016/j.procs.2025.09.160

Olszak, C. M., & Bartuś, K. (2025). AI-enhanced Business Intelligence for decision-making. *Procedia Computer Science* . doi:10.1016/j.procs.2025.09.160

Ponniah, P. (2010). *Data Warehousing Fundamentals for IT Professionals.* New Jersey: John Wiley & Sons, Inc.

Python.org. (2026). *What is Python? Executive Summary*. Retrieved from Python.org: https://www.python.org/doc/essays/blurb/

Ren, F. (2024). A Comprehensive Analysis of Behavioral Finance and its Impact on Investment Decisions . *Highlights in Business, Economics and Management , 32*. Retrieved January 30, 2026

Roff, J. T. (2003). *UML™: A Beginner's Guide.* New York: McGraw-Hill Companies. Retrieved January 21, 2026

Studoco. (2024). *What are the advantages disadvantages to presenting tabular data*. Retrieved January 30, 2026, from Studoco.com: https://www.studocu.com/en-us/messages/question/3118883/what-are-the-advantagesdisadvantages-to-presenting-tabular-data-as-opposed-to-visual-data

Thompson, J. (2025, January 10). *Metabase alternatives: peeking at other Business Intelligence tools*. Retrieved from Metabase: https://www.metabase.com/blog/metabase-alternatives

Tobin, D. (2026, January 11). *Best SSIS Alternatives - 2026*. Retrieved from Integrate.io: https://www.integrate.io/blog/best-ssis-alternatives/

Tversky, A., & Kahneman, D. (1979, March). Prospect Theory: An Analysis of Decision Under Risk . *Econometrica*, 263-292. Retrieved January 31, 2026, from https://scispace.com/pdf/prospect-theory-an-analysis-of-decision-under-risk-4z6oysdahj.pdf

Tversky, A., & Kahneman, D. (1981, January). The Framing of Decisions and the Psychology of Choice. *Science*, 453-458. Retrieved January 31, 2026, from https://psych.hanover.edu/classes/Cognition/Papers/tversky81.pdf

Vora, A. S., & Kulkarni, A. A. (2024, July 24). Shannon meets Myerson: Information extraction from a strategic sender. *Mathematical Social Sciences*, 19. Retrieved January 20, 2026, from https://doi.org/10.1016/j.mathsocsci.2024.07.002

APPENDIX A: Prompts used to generate Python scripts for the first layer of extraction from Google Gemini AI

The following prompts were used to generate Python code that does the following using Google Gemini:

1. Extracts email content from my Gmail account using Google Cloud API
2. Extracts specific data from email messages and saves it in a csv file format

The code for both tasks aligns with industry standards and is considered safe to use.

Prompt 1: To generate Script.py (Gmail Extraction)

Write a Python script to download specific EasyEquities emails from Gmail using the Google API. The script must:

- Authentication: Use OAuth2 with google-auth-oauthlib. Store credentials in token.pickle and credentials.json. Handle invalid_grant errors by clearing the token and forcing a re-login.
- Search Query: Search for emails from info@easyequities.co.za with subjects matching exactly: "confirmation of open order" OR "Confirmation of your transaction".
- Content Extraction: Recursively search the email payload for text/plain or text/html. If HTML is found, use BeautifulSoup to extract clean text.
- File Management: Create an output directory named emails_text. Save each email as a .txt file with a filename format of YYYYMMDD_HHMMSS_Subject_MessageID.txt. Ensure special characters in filenames are replaced with underscores.
- Formatting: Each saved .txt file must include a header containing the "Subject" and "Date" followed by a separator line of 40 dashes.

Prompt 2: To generate ExportData.py (Text Mining & CSV)

Write a Python script to mine local text files for financial data and export them to a CSV. The script must:

- Setup: Process all .txt files in a folder named emails_text and output to extracted_transactions.csv.
- Cleaning: Replace non-breaking spaces and standardise whitespace while maintaining line structure.
- Common Regex Extraction:
    - Date: Extract text between Date: and +0000.

Figure 23: The first section of the prompts to Google Gemini AI

- Entity: Match text after "Trade Invoice for", "Tax Invoice for", or "EasyEquities Order", stopping at "SHARES" or "Traded".
- Account: Find text between Acc. number: and Trader. Map E############5 to R and E############8 to $ in a separate currency column.

- Reference Numbers: Extract Order and Invoice numbers by finding the word "Order" or "Invoice" and capturing the first succeeding string that starts with a # and has 5-15 alphanumeric characters.
- Transaction Logic: Classify into four types based on keywords:
  1. Open Order Buy: Triggered by "total estimated transaction cost" (no "less costs"). Capture values for "estimated cost" and "trade value" (stopping at "AMOUNT").
  2. Open Order Sell: Triggered by "net est. amount". Capture "less costs" and use a strict regex [0-9., R$]+ to extract the amount due.
  3. Buying: Triggered by "total transaction cost". Capture "total transaction cost" and "trade value".
  4. Selling: Triggered by "net amount" and "less costs". Capture "less costs" and "net amount due".
- Robustness: Use re.IGNORECASE and re.DOTALL. Ensure that if an extracted value exceeds 50 characters, only the first line is kept to prevent capturing body text.

Safety concerns regarding the first script:

The security of the provided script depends on the implementation of local authentication and the restriction of data access through specific API permissions.

Authentication and Session Management

The script employs the OAuth 2.0 framework to facilitate access to the Gmail API. It requires a credentials.json file to identify the application and generate a token.pickle file to maintain the authorisation session. This architecture ensures that the user's Google password is never accessed or stored by the code. By utilizing the gmail.readonly scope, the script is technically restricted to viewing message data and cannot perform actions such as sending, deleting, or modifying emails.

Data Privacy and Local Processing

Figure 24:: The second section of the prompts to Google Gemini AI

All email retrieval and text extraction occur within the local environment of the user's machine. The script connects to Google's servers to fetch message content and then processes it locally using the BeautifulSoup library to convert HTML into plain text. No data is transmitted to external third-party servers during this process. The output is stored directly in a local directory named emails_text.

Risks and User Responsibilities

The account's safety remains linked to the physical and digital protection of the local machine. If the token.pickle, or credentials.json files are compromised or shared publicly, an unauthorised party could gain read-only access to the inbox. The script includes logic to refresh expired sessions, but if re-authentication fails, it requires the user to manually trigger the login flow again to generate a new secure token.

Safety concerns regarding the second script:

1. This script runs entirely offline. It simply reads text files that are already sitting in a folder on the computer. It does not (and cannot) send financial data to anyone else. Your data remains strictly under your sovereignty.

2. Read-Only on Your Source Files: The script opens email text files in 'r' (read) mode. It extracts information but cannot delete, change, or corrupt the original email text files. If the script crashes or fails, the original data remains untouched.

3. No Credentials Needed: Unlike the Gmail script, this code doesn't touch token.pickle or credentials.json. It doesn't know who you are; it only sees text patterns.

4. The Only Risk: Overwriting the Output: The only "destructive" action this script takes is writing the extracted_transactions.csv file.

Caution: If you already have a file named extracted_transactions.csv in that folder, this script will overwrite it with the new data every time you run it.

Fix: If you want to keep old versions, just rename the CSV file after it is generated.

Summary

Feature Status Note Connectivity: No internet connection used.

Data Privacy: Data never leaves your folder.

File Integrity: Original text files are never modified.

Dependencies: Uses only standard Python tools (os, csv, re).

Figure 25: : The third section of the prompts to Google Gemini AI

APPENDIX B: Resources required to run the Metabase platform

Metabase Installation Guides

- Metabase Official Installation (all methods) - https://www.metabase.com/docs/latest/installation-and-operation/installing-metabase

- Run Metabase using Docker (recommended) - https://www.metabase.com/docs/latest/operations-guide/running-metabase-on-docker

Metabase Troubleshooting (connections & configs) - https://www.metabase.com/docs/latest/troubleshooting-guide
Required Downloads (Dependencies)
1. Java (Required for Metabase JAR)
    - Java 17+ (LTS version recommended) — Official Oracle/OpenJDK builds - https://adoptium.net/?variant=openjdk17
    - Java Install & Configure Guide (Linux / Windows) - https://adoptium.net/installation.html

2. Metabase Application (Download) -
Metabase JAR (standalone app)
https://www.metabase.com/start

Database Connectivity Drivers

Microsoft SQL Server JDBC Driver (for Metabase connections)

- Microsoft JDBC Driver for SQL Server (Download) - https://learn.microsoft.com/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server
- JDBC Setup Guide for Metabase - https://www.metabase.com/docs/latest/databases/connections/sql-server

Metabase Configuration & Connect Guides

- Connect Metabase to SQL Server (How-To) - https://www.metabase.com/docs/latest/databases/connections/sql-server

- Metabase Admin Guide (Database Connections & Security) - https://www.metabase.com/docs/latest/admin-guide
- Metabase User Permissions & Access Control - https://www.metabase.com/docs/latest/users-guide/permissions

Getting Started with Queries & Dashboards

- Metabase Basics (Querying & Dashboards) - https://www.metabase.com/docs/latest/users-guide
- SQL Editor & Querying with Metabase - https://www.metabase.com/docs/latest/users-guide/04-sql-queries

APPENDIX C: Resources for installing Python, SQL Server, Visual Studio and SSIS tools

Below are links to official download and documentation pages for all tools referenced in the system design:

A. Python (Core Language)

- Official site (download installers & setup): https://www.python.org/downloads/
- Documentation (installation & configuration): https://docs.python.org/3/

B. Python Package Manager (pip)

- pip comes bundled with Python 3.x installations.
- Documentation: https://pip.pypa.io/en/stable/

C. Python Libraries

1. pandas (Data manipulation & tabular processing)

- Package site & documentation: https://pandas.pydata.org/
- Installation: pip install pandas

2. BeautifulSoup (HTML parsing)

- Library homepage: https://www.crummy.com/software/BeautifulSoup/
- Installation (with HTML parser support):
    - pip install beautifulsoup4
    - pip install lxml

3. requests (HTTP client)

- Documentation & guide: https://docs.python-requests.org/
- Installation: pip install requests

4. json (Built-in Module)

- Python's standard library for JSON: https://docs.python.org/3/library/json.html
- (No installation required; part of Python core.)

D. API Client Configuration

If connecting to a specific API (e.g., Gmail API, Microsoft Graph, Google Workspace), download the official SDK and documentation:

- Google API (e.g., Gmail API)
- Developer console (credentials, APIs): https://console.developers.google.com/

- Python client library & docs: https://developers.google.com/api-client-library/python
- Gmail API documentation: https://developers.google.com/gmail/api

E. SQL Server

- SQL Server Developer Edition (Free for development) - https://learn.microsoft.com/sql/sql-server/download-sql-server
- Licensing & edition comparison: https://learn.microsoft.com/sql/sql-server/editions-and-components

F. Visual Studio and SSIS Tools

- Visual Studio Community (Free): https://visualstudio.microsoft.com/vs/community/
- Setup guide: https://learn.microsoft.com/visualstudio/install/install-visual-studio

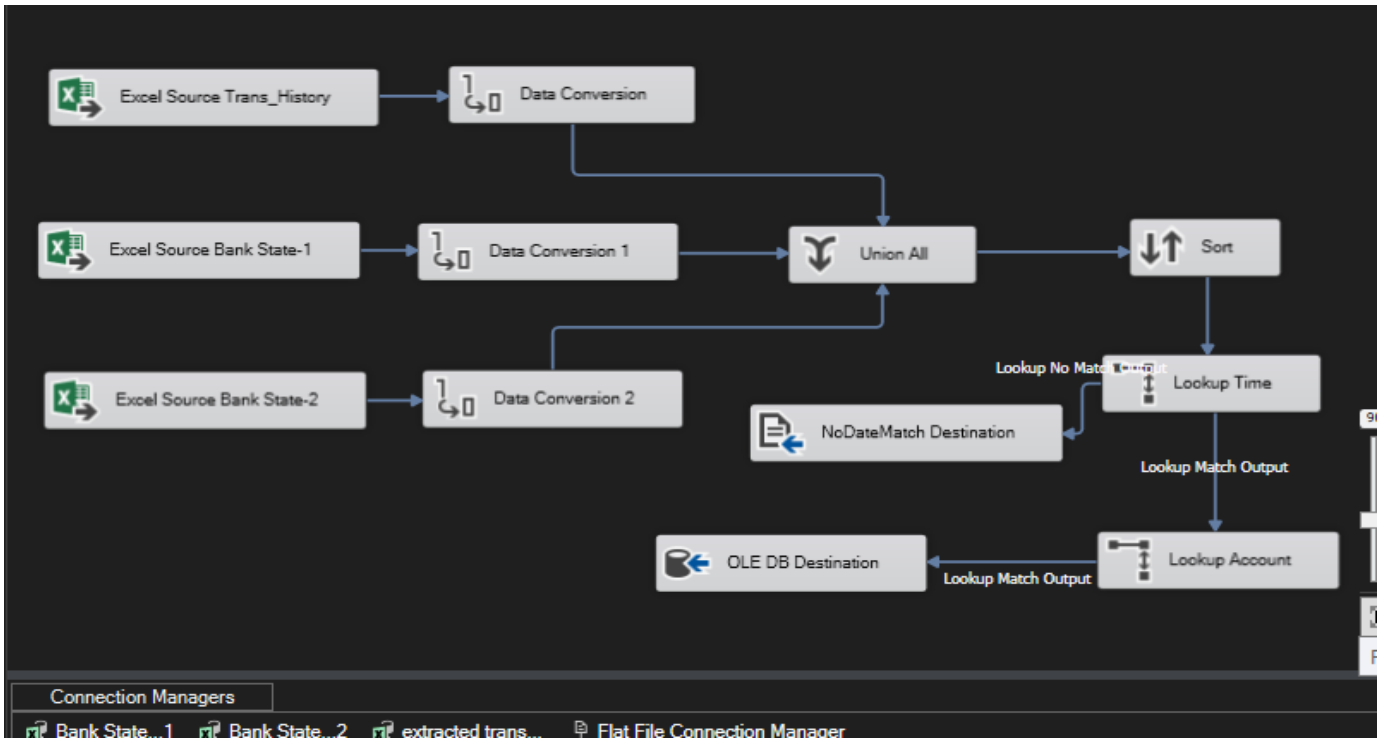
SQL Server Data Tools (SSDT)

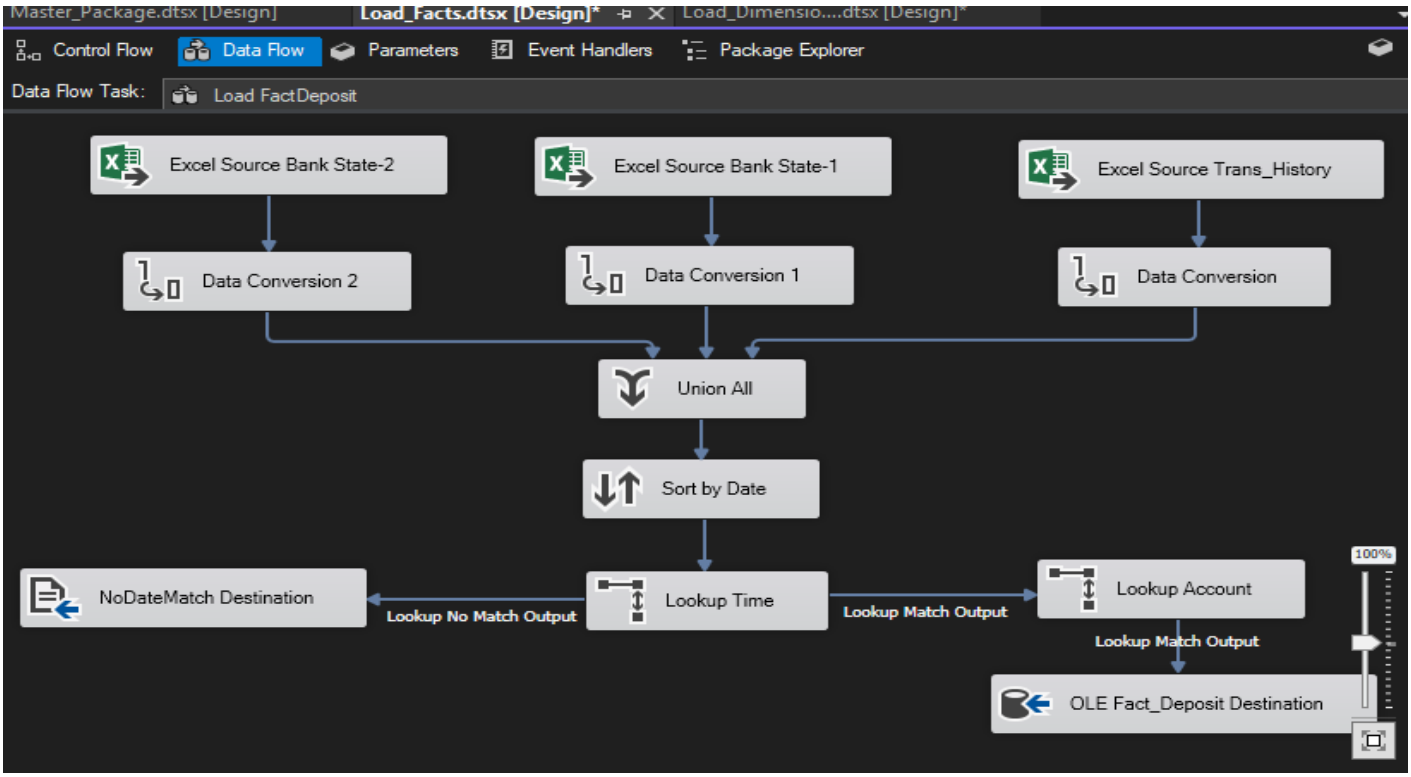
- Required for SSIS package development inside Visual Studio - https://learn.microsoft.com/sql/ssdt/download-sql-server-data-tools-ssdt

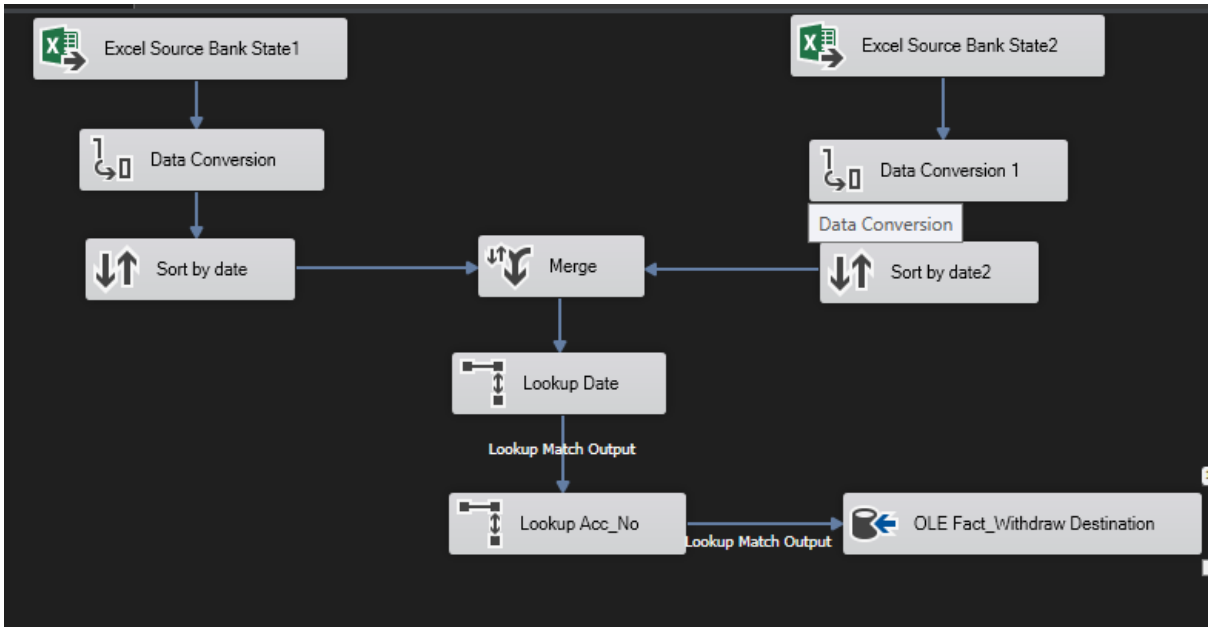
SSIS (SQL Server Integration Services)

- Overview and usage: https://learn.microsoft.com/sql/integration-services/

APPENDIX D: Additional visuals of the data pipelines






Excel Source Bank State1
Excel Source Bank State2
Data Conversion
Data Conversion 1
Data Conversion
Sort by date
Merge
Sort by date2
Lookup Date
Lookup Match Output
Lookup Acc_No
Lookup Match Output
OLE Fact_Withdraw Destination