THOMAS MORE

Data Engineering Project at ACA Group

Realization document

Kyano Trevisan
Student Bachelor Applied Computer Science

2024 - 2025

Table of Contents

Executive Summary

This document describes the realisation of an improved sustainability reporting platform for Duvel Moortgat NV, developed during the Sustainathon 2025 internship at ACA Group. The project built upon an existing proof-of-concept to create a more maintainable, efficient, and scalable system that supports CSRD (Corporate Sustainability Reporting Directive) compliance requirements.

The implementation delivered substantial technical improvements through code optimization and architectural restructuring. In the water model alone, I reduced the codebase from over 300 lines to under 80 lines using dynamic Jinja templating, a pattern I applied across all 39 models in the Silver source layer. The integration pipeline efficiency improved significantly, reducing processing time from approximately 45 minutes to just over 20 minutes. Additionally, I implemented a simplified and improved data model, restructured the medallion architecture for better maintainability, and enhanced error handling throughout the system.

The project demonstrates how thoughtful data engineering can transform a functional proof-of-concept into a production-ready system that supports both regulatory compliance and business decision-making while remaining maintainable and scalable for future enhancements.

1. Introduction

1.1. Project Context and Business Challenge

Duvel Moortgat NV, the holding company of one of Belgium's most iconic breweries, must comply with the European CSRD (Corporate Sustainability Reporting Directive) regulations. This legislation requires large companies to publish comprehensive sustainability reports covering environmental impact, social aspects, and governance metrics with precise documentation and audit trails.

Duvel's sustainability team, led by Peter Willaert (Sustainability Director), oversees the organization's environmental reporting strategy. Dries Van Hout (Sustainability Controller) works under Peter and is responsible for creating the actual reports that Duvel submits to the EU. Our automated platform significantly assists Dries by providing centralized data access and automated calculations, making it much easier to find the data needed for regulatory reports.

This realization document describes the technical implementation of the Duvel Moortgat Sustainathon 2025 project, building upon the objectives and scope defined in the original internship proposal "Stagevoorstel – Duvel Moortgat NV: Sustainathon 2025."

1.1.1. Link to Project Plan and Objectives

The project was structured around the five Prior priorities established during the Sustainathon kick-off phase:

Priority 01: Integration with ClimateCamp

- Objective: Send sales and purchase data to ClimateCamp (an external company that calculates Duvel's environmental impact) and retrieve carbon footprint calculations
  - Realization: Achieved through sophisticated data transformations and automated exchange process (Section 4.3)
  - Measurable Outcome: Automated data submission and result integration eliminating manual processing

Priority 02: Operational production data

- Objective: Capture operational production data, map data sources & measures, create integrations for internal and external data
- Realization: Implemented through enhanced database integration architecture (Section 3.2)
- Measurable Outcome: Successful integration of production data from Navision and water sensor data

Priority 3: HR & Safety indicators

- Objective: Map HR & Safety measures and create integrations for internal and external data using Workday, a platform used to manage employee data.
  - Realization: Implemented Workday integration for HR data processing (Section 3)
  - Measurable Outcome: Automated HR and safety indicator processing

Priority 4: Salesforce integration

- Objective: Implement comprehensive data quality validation and error detection
- Realization: Added try-catch statements and validation logic throughout the system (Section 4.4)
- Measurable Outcome: Specific error messages that reduce debugging time by providing clear diagnostic information

Priority 5: Standardize Data model

- Objective: Establish test and production environments for ongoing development
- Realization: Implemented parameterized architecture with Azure DevOps integration (Section 5.1)
- Measurable Outcome: Enable development without disrupting production operations

1.1.2. Scope and Deliverables Alignment

Original Deliverables (from Project Plan):

1. Microsoft Fabric Components
   - Power BI Reports
   - ETL Pipelines
   - New integrations with external and internal sources
   - Implementation: Enhanced through architectural improvements (Section 2.4) and integration optimization (Section 3)

2. Data Model for Additional Sites
   - Implementation: Achieved through parameterized location management (Section 4.1)
   - Validation: Successfully demonstrated by adding new site during project

1.1.3. Regulatory Context

Duvel Moortgat NV must comply with the European CSRD (Corporate Sustainability Reporting Directive) regulations, which requires comprehensive sustainability reports covering environmental impact, social aspects, and governance metrics with precise documentation and audit trails. This regulatory framework directly influenced our technical architecture decisions, particularly the emphasis on data quality validation, audit trail preservation, and calculation accuracy.

1.2. ACA Group Partnership and Team Structure

ACA Group's "Datadots" team specializes in developing data solutions for complex business challenges. The project was executed as part of ACA's Sustainathon 2025 initiative, building upon a proof-of-concept developed by Kwinten Boes (Data Analyst, Project Manager) and Thibo Vanderkam (Data Engineer, Technical Support) during their internship the previous year.

Working alongside fellow intern Fabian Reyes, I focused on the data engineering aspects while Fabian concentrated on dashboard development and client communication. This collaboration proved essential, as our combined efforts addressed both backend data processing improvements and frontend analytics enhancements. The mentorship provided by Kwinten and Thibo was invaluable, particularly Thibo's technical guidance on data engineering best practices and Kwinten's project management expertise.

The project faced unique constraints due to circumstances from the previous year. During Kwinten and Thibo's internship, Duvel experienced a security incident that resulted in loss of access to systems for over three weeks. This time constraint meant that the proof-of-concept was developed quickly to deliver value within the remaining timeframe, resulting in a functional but less structured codebase that needed refinement for production use.

1.3. CSRD Regulatory Framework and Technical Requirements

The Corporate Sustainability Reporting Directive represents the European Union's most comprehensive sustainability reporting legislation. It requires detailed disclosure of environmental metrics including carbon emissions, energy consumption, waste management, and water usage, along with social and governance indicators.

For Duvel Moortgat, this means collecting and validating data from multiple facilities while ensuring calculation accuracy and maintaining documentation that supports both internal decision-making and external compliance verification. The regulatory requirements demand precise calculations with clear audit trails, but the reporting frequency is typically monthly rather than real-time, aligning with business reporting cycles.

1.4. Technical Challenge Overview

The project's technical challenges centered on three primary data sources that required integration and processing. SharePoint repositories contain Excel input sheets uploaded by stakeholders at each facility, following standardized templates created by Kwinten and Thibo. The BIM server houses water sensor data from Duvel Moortgat, while the BI server (Navision) contains operational and financial data required for sustainability calculations.

The Excel input sheets, while structured according to established templates, remain prone to human error during data entry. Stakeholders occasionally add comments, modify formatting, or enter incorrect data types, requiring robust error handling and validation. The geographic distribution across eight countries creates additional complexity in data coordination and stakeholder communication.

Integration with ClimateCamp, Duvel's external environmental calculation platform, adds another layer of complexity. We provide formatted data to ClimateCamp for environmental impact calculations, and they return Excel files with results that we integrate back into our system through the SharePoint processing pipeline.

2. Technology Foundation and Platform Architecture

2.1. Microsoft Fabric Platform Overview

Microsoft Fabric serves as the comprehensive data platform foundation for the project. Fabric is a centralized platform that unifies data engineering, data science, and analytics capabilities in a single environment. The platform includes multiple integrated services: Lakehouses for Delta Lake-based data storage, Data Warehouses for optimized analytical processing, Notebooks for PySpark and SQL development, Pipelines for workflow orchestration, and Semantic Models for business logic abstraction.

Importantly, Fabric now incorporates Power BI as an integrated service, providing seamless connectivity between data processing and visualization layers. This integration eliminates the traditional boundaries between data engineering and business intelligence, enabling more efficient development workflows and better performance optimization.

Since we were continuing from an existing proof-of-concept, Microsoft Fabric was already established as the platform rather than being newly selected. However, working with Fabric presented unique challenges due to its active development status, with many features still in preview. During the internship, we experienced platform outages, including a notable incident on May 14th when the service was unavailable from early morning until approximately 10 AM, highlighting Fabric's development stage and the importance of having resilient development practices.

The platform's aggressive caching behavior created particular debugging challenges. When integration notebooks downloaded Excel files from SharePoint and stored them in the lakehouse, Fabric's caching would sometimes continue returning old error messages even after underlying issues were resolved. To address this, I implemented explicit file deletion and re-download processes that force cache invalidation, ensuring that transformations always work with current data.

2.2. dbt (Data Build Tool) for Data Engineering

dbt (data build tool) transforms SQL into a software engineering workflow, enabling data engineers to build reliable data models using software engineering best practices. Unlike traditional SQL scripting, dbt provides version control integration, automated testing capabilities, comprehensive documentation generation, and dependency management that ensures transformations execute in the correct order.

dbt allows data engineers to write modular SQL that tells a story through documentation and lineage tracking. Models can reference other models, creating a dependency graph that dbt automatically resolves during execution. The framework includes built-in testing capabilities that validate data quality expectations and macro functionality that enables code reuse across models.

For this project, dbt proved essential for implementing the medallion architecture refactoring and dynamic code generation patterns. We used the Power User extension for VS Code, which significantly enhances dbt development with features like model compilation preview, lineage visualization, and integrated documentation browsing.

The integration between dbt and Fabric occurs through a simple notebook that executes dbt commands against our data warehouse. While conceptually straightforward, this execution handles the complex dependency resolution and transformation logic that powers our entire analytics pipeline.

2.3. Technology Choice Analysis and Validation

Although Microsoft Fabric was already established as the platform from the previous year's proof-of-concept, conducting a retrospective analysis provides valuable insight into the architectural decisions and validates their ongoing relevance for the project requirements.

2.3.1. Data Platform Comparison

Evaluation Criteria and Weights:

- Integration with Existing Systems (25%): Compatibility with Duvel's Microsoft ecosystem
- Learning Curve and Team Expertise (20%): Available knowledge within ACA and ease of adoption
- Regulatory Compliance Features (20%): Built-in audit trails and data governance capabilities
- Cost and Licensing (15%): Total cost of ownership including development and operational costs
- Scalability and Performance (10%): Ability to handle growing data volumes and complexity
- Vendor Support and Ecosystem (10%): Available documentation, community, and professional support

Criteria	Weight	Microsoft Fabric	Snowflake + Tableau	AWS Redshift + QuickSight	Score Calculation
Integration	25%	9	6	7	Fabric: 9×0.25 = 2.25
Learning Curve	20%	8	6	5	Fabric: 8×0.20 = 1.60
Compliance	20%	8	7	7	Fabric: 8×0.20 = 1.60
Cost	15%	7	6	8	Fabric: 7×0.15 = 1.05
Scalability	10%	7	9	9	Fabric: 7×0.10 = 0.70
Support	10%	8	8	7	Fabric: 8×0.10 = 0.80
Total	100%	8.0	6.7	6.9	Fabric Wins

2.3.2. Data Transformation Tool Analysis

Evaluation Criteria and Weights:

- Code Reusability (30%): Ability to create reusable, modular transformation logic
- Version Control Integration (25%): Native support for Git workflows and collaborative development
- Testing Framework (20%): Built-in data quality and validation capabilities
- Documentation Generation (15%): Automatic documentation and lineage tracking
- Team Expertise (10%): Available knowledge within ACA and learning curve

Criteria	Weight	dbt	Custom SQL Scripts	Azure Data Factory	Score
Code Reusability	30%	9	4	6	dbt: 2.7
Version Control	25%	9	6	7	dbt: 2.25
Testing Framework	20%	9	3	5	dbt: 1.8
Documentation	15%	9	4	6	dbt: 1.35
Team Expertise	10%	8	7	5	dbt: 0.8
Total	100%	8.9	4.8	6.1	dbt Wins

2.3.3. Architecture Pattern Evaluation

Evaluation Criteria and Weights:

- Data Quality Control (30%): Built-in validation and quality assurance capabilities
- Regulatory Audit Trail (25%): Compliance with CSRD documentation requirements
- Development Speed (20%): Time to implement and modify transformations
- Maintenance Overhead (15%): Long-term operational complexity
- Industry Standard Adoption (10%): Widespread acceptance and community support

Criteria	Weight	Medallion	Data Vault	Star Schema Only	Score
Data Quality Control	30%	9	8	6	Medallion: 2.7
Regulatory Audit Trail	25%	8	9	5	Medallion: 2.0
Development Speed	20%	7	5	8	Medallion: 1.4
Maintenance	15%	8	6	7	Medallion: 1.2
Industry Standard	10%	9	7	8	Medallion: 0.9
Total	100%	8.2	7.0	6.8	Medallion Wins

2.3.4. Validation of Technology Choices

The weighted ranking analysis confirms that the established technology choices align well with project requirements:

Microsoft Fabric scored highest primarily due to its seamless integration with Duvel's existing Microsoft ecosystem and the reduced learning curve for the development team. While other platforms may offer superior performance or cost benefits, the integration advantages and reduced operational complexity justify the platform choice.

dbt emerged as the clear winner for data transformation, particularly excelling in code reusability and testing framework capabilities that proved essential for implementing the dynamic code generation patterns described in Section 4.1.

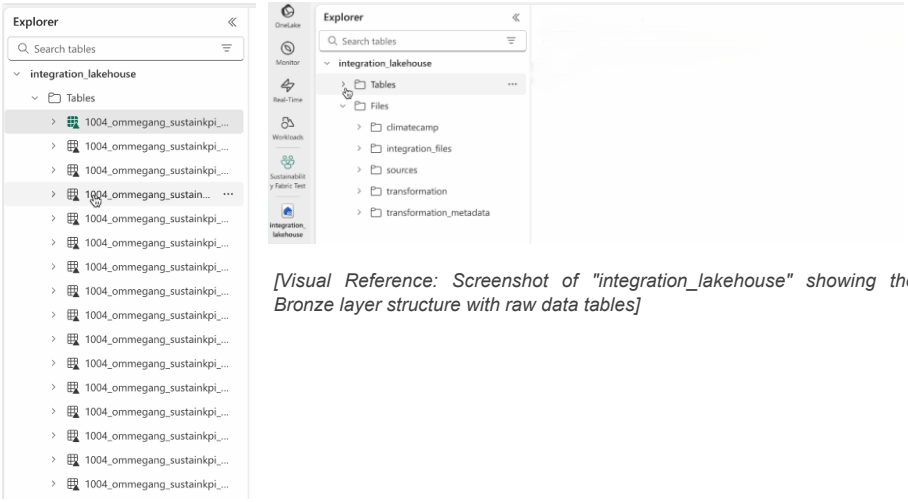
Medallion Architecture validated as the optimal pattern for sustainability reporting, providing the data quality controls and audit trails required for CSRD compliance while maintaining development efficiency.

2.4. Medallion Architecture Implementation and Refinement

The medallion architecture provides a structured approach to data quality and transformation through Bronze, Silver, and Gold layers. Due to the time constraints faced during the previous year's development, the initial implementation focused on delivering functional results rather than optimal architectural organization.

The Bronze layer uses our Fabric lakehouse to store raw data directly as ingested from source systems. This approach provides ACID transaction guarantees and time travel capabilities through Delta Lake format. ACID (Atomicity, Consistency, Isolation, Durability) transactions ensure that data writes either complete entirely or fail cleanly, preventing partial updates that could corrupt the dataset. Time travel capabilities allow querying historical versions of data, though our daily full refresh approach means we maintain complete historical datasets rather than relying on time travel for audit trails.

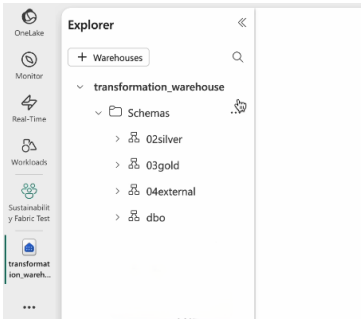
Commented [BH1]: our? En ik zou hier eerder een diagram gezet hebben, een visual, van de 3 layers, en ook uitgelegd hebben wat het verschil is tussen een data lake, en een data lake house



[Visual Reference: Screenshot of "integration_lakehouse" showing the Bronze layer structure with raw data tables]

My primary contribution involved restructuring the Silver layer into clearly defined Source and Enriched sub-layers. The Source sub-layer handles basic data cleaning including null value handling, data type standardization, and naming convention normalization across different input sources. The Enriched sub-layer implements business logic including joins across multiple sources, currency conversions, and calculated fields required for sustainability reporting.

The Gold layer contains fact and dimension tables optimized for Power BI consumption. This dimensional modeling approach ensures responsive dashboard performance while providing the granular detail needed for comprehensive sustainability analysis. The Semantic Model layer abstracts business logic and provides the interface that Power BI uses for dashboard development. The semantic model acts as a business-friendly layer that translates complex database structures into understandable metrics and relationships that can be easily consumed by report developers and end users.

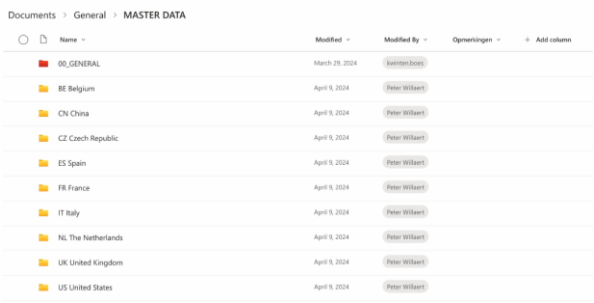


[Visual Reference: Screenshot of "transformation_warehouse" showing the Silver and Gold layer organization]

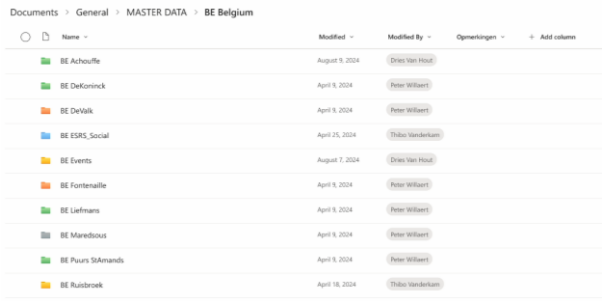
3. Data Integration Architecture and Processing

3.1. SharePoint Integration and File Processing

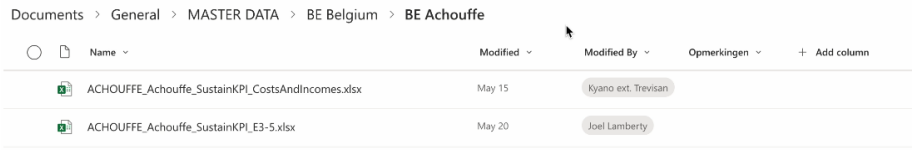
The SharePoint integration processes Excel files uploaded by stakeholders across Duvel's global facilities. Each facility follows a standardized folder structure: SharePoint MAIN DATA > [country] > [location] > inputsheet. The input sheets themselves follow consistent templates created by Kwinten and Thibo, ensuring uniform data structure across all locations.



[Visual Reference: Screenshot of SharePoint environment showing the MAIN DATA folder with country folders (BE, CN, CZ, ES, FR, IT, NL, UK, US)]



[Visual Reference: Screenshot of BE Belgium folder opened, displaying the Belgian site folders including Achouffe, DeKoninck, Liefmans, etc.]



[Visual Reference: Screenshot from inside the BE Achouffe folder showing the inputsheets]

Building on the existing integration foundation, I enhanced the system with improved error handling and efficiency optimizations. The integration notebook now includes comprehensive try-catch statements with detailed error messages that help identify exactly where problems occur during processing. This enhancement significantly reduces debugging time when data quality issues arise.

[Visual Reference: Screenshot of the integration notebook showing the code that downloads SharePoint files, extracts needed sheets, and converts them to Delta tables in the lakehouse]

One key addition I implemented is the dynamic location code extraction. Each input sheet contains a Location_Code sheet with the facility identifier, which I extract and append to every data row during processing. This ensures proper data lineage and enables facility-specific analysis while eliminating the hardcoded location mappings that existed in the original implementation.

[Visual Reference: Screenshot showing the different sheets within an inputsheet, including the various sustainability data categories]

[Visual Reference: Screenshot of the Location_Code sheet opened, showing the dropdown selection and the location code value that the integration notebook extracts]

The location codes available in the dropdown come from the Overview_Sites reference file, which contains comprehensive facility information including Company_ID, location codes, addresses, contact information, and operational details like surface area and ownership percentages. This centralized reference ensures consistency across all input sheets while providing the master data needed for facility-specific reporting.

The file processing logic maintains strict adherence to the template structure. If stakeholders modify the input sheet format, add comments in unexpected locations, or alter the data table positioning, the processing will fail with clear error messages rather than attempting to adapt. This approach ensures data integrity by requiring source data to meet established standards rather than accommodating variations that could introduce errors.

3.2. Database Integration Architecture

The database integration connects to two on-premise SQL servers at Duvel's headquarters in Puurs, Belgium, through secure gateway configurations. The BIM server currently contains water sensor data from Duvel Moortgat, while the BI server (Navision) holds operational and financial data required for sustainability calculations.

The gateway setup, managed by Bart De Prins (BI & Integrations Team Lead), provides secure connectivity between Duvel's on-premise infrastructure and our cloud-based processing platform. We use one gateway for each server, ensuring proper access controls and network security.

Database queries extract the specific datasets required for sustainability reporting without unnecessary data transfer. The integration runs daily as part of the overall pipeline, though the extraction volumes are manageable and don't require complex optimization strategies for the current data scope.

3.3. ClimateCamp Data Exchange Process

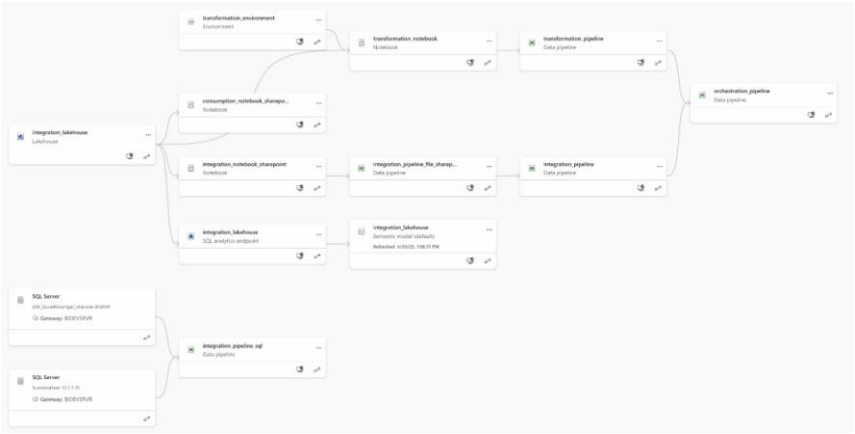
The ClimateCamp integration involves providing formatted data to their external platform for environmental impact calculations. We export sales and purchase data in specific formats that ClimateCamp requires, following their data categorization and calculation standards.

ClimateCamp performs the environmental calculations and returns results as Excel files, which they place in our SharePoint environment. These result files are then processed through our standard SharePoint integration pipeline, making the calculated environmental metrics available in our data warehouse for dashboard consumption.

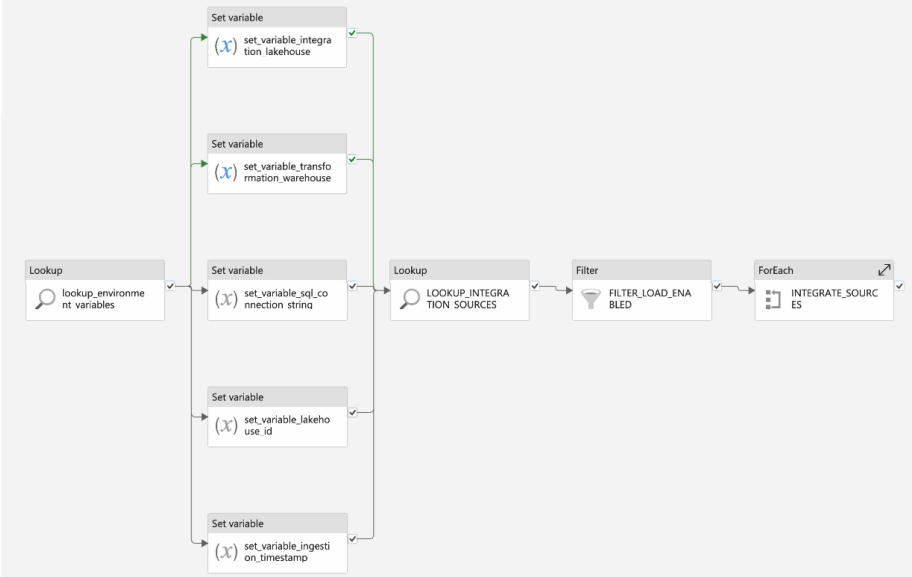
This process, while currently manual in terms of file exchange, provides Duvel with professionally calculated environmental impact data that meets regulatory standards. Future enhancements could include API-based integration if ClimateCamp develops such capabilities, though this would require development on their platform rather than ours.

3.4. Pipeline Orchestration and Execution

The overall data processing follows a straightforward daily execution schedule. The integration pipeline runs first, processing all SharePoint files and database extractions to populate the Bronze layer. Upon successful completion, the transformation pipeline executes, running our dbt models to process data through the Silver and Gold layers.

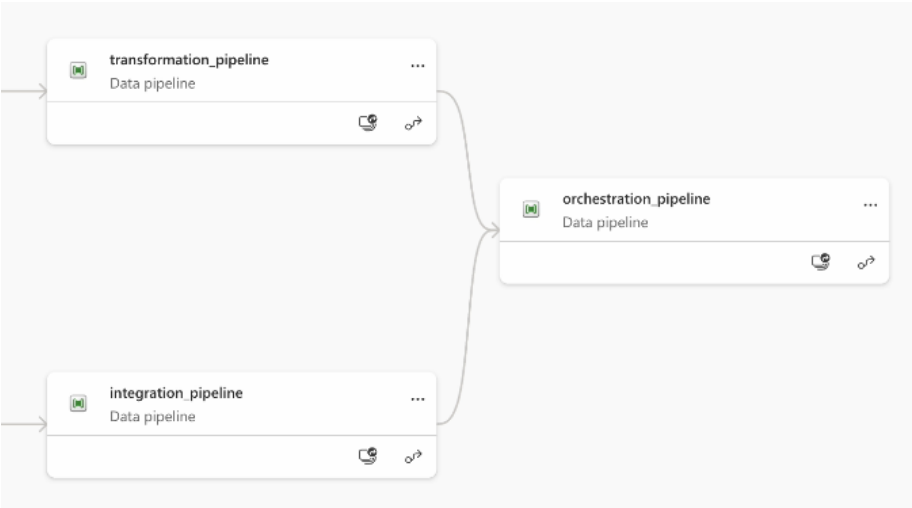


[Visual Reference: Complete lineage diagram showing the full orchestration flow from data sources through integration, transformation, to lakehouse]

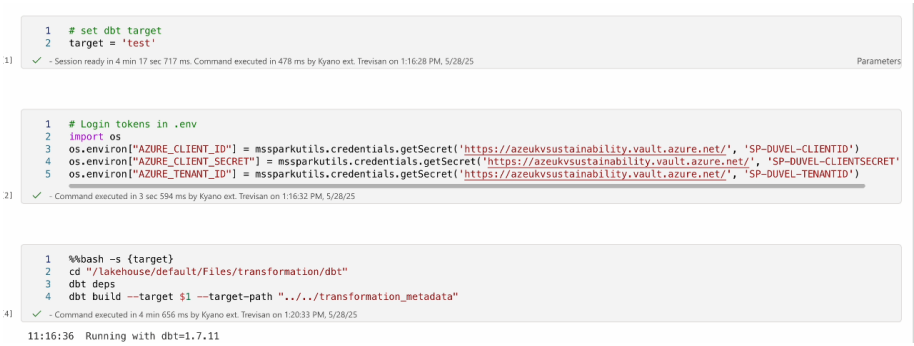


[Visual Reference: Screenshot of the integration pipeline showing the workflow steps and parameter configuration]

The integration pipeline efficiency improved significantly during the project, reducing processing time from approximately 45 minutes to just over 20 minutes. This improvement came through code optimization and the implementation of multithreading in the integration notebook, enabling parallel processing of multiple data sources.



[Visual Reference: Screenshot of the orchestration pipeline showing the sequential execution of integration followed by transformation]



[Visual Reference: Screenshot of the transformation notebook demonstrating the dbt execution workflow]

Microsoft Fabric provides automatic retry capabilities for pipeline failures, attempting execution up to three times before marking a pipeline as failed. This built-in resilience handles transient issues like network connectivity problems or temporary service unavailability.

The daily execution schedule aligns well with business requirements, as input sheets are typically updated monthly rather than daily, and stakeholders don't require real-time dashboard updates for sustainability reporting purposes.

4. Advanced Data Transformation and Code Optimization

4.1. Dynamic Code Generation with Jinja Templating

The most significant technical achievement involved implementing sophisticated Jinja templating throughout the dbt project to eliminate code duplication and improve maintainability. This section details the implementation patterns and their impact across the system.

4.1.1. Problem Analysis and Solution Approach

Original Code Structure Issues
The water model exemplified the systemic problem: the original implementation required over 300 lines of code with separate SELECT statements for each facility. Each model contained repetitive code that manually referenced facility input sheets and combined them through UNION statements.

```
-- Original approach (simplified example)
SELECT * FROM {{ source('lakehouse', 'achouffe_sustainkpi_e3_5_water') }}
UNION ALL
SELECT * FROM {{ source('lakehouse', 'dekoninck_sustainkpi_e3_5_water') }}
UNION ALL
SELECT * FROM {{ source('lakehouse', '1127_deval_sustainkpi_e3_5_water') }}
-- ... repeated for all 39 locations
```

Dynamic Solution Benefits
The dynamic implementation accomplishes identical functionality in under 80 lines while providing:

- Centralized configuration management
- Elimination of manual code replication
- Automatic adaptation to new locations
- Consistent error handling patterns

4.1.2. Core Macro Implementation

Location Prefixes Management
I created a centralized macro that maintains all location prefixes in a single location:

```
{% macro get_location_prefixes() %}
{{ return([
    'achouffe_achouffe',
    'dekoninck_dekoninck',
    '1127_deval',
    'fontenail_fontenaille',
    'oudenaarde_liefmans',
    'micromaed_maredsous',
    'magbrouw_puurs',
    'rutsbroek_rutsbroek',
    -- ... all location prefixes
]) }}
{% endmacro %}
```

Why this transformation is needed: The original implementation required manual code replication for each facility, making the system difficult to maintain and prone to errors when adding new locations. This approach eliminates hardcoded facility references and creates a centralized configuration system.

What exactly happens: The location prefixes macro maintains all facility identifiers in a single location, enabling dynamic SQL generation that automatically includes all configured facilities without manual code modifications.

Dynamic Year Generation
The currency conversion system uses temporal logic for multi-year processing:

```
{% macro generate_year_list(start_year=2022) %}
    {% set current_year = modules.datetime.datetime.now().year %}
    {% set years = range(start_year, current_year) %}
    {{ return(years|list) }}
{% endmacro %}
```

4.1.3. Pattern Implementation Across Models

Standard Source Model Pattern
Each of the 39 Silver source layer models now follows this pattern:

```
{% set location_source_prefixes = get_location_prefixes() %}
{% set location_source_suffix = '_sustainkpi_e3_5_water' %}

WITH water AS (
    {% for prefix in location_source_prefixes %}
    SELECT
        CAST([Date] AS Date) AS [Date],
        CAST([Type] AS VARCHAR(255)) AS [Type],
        CAST([Description] AS VARCHAR(255)) AS [Description],
        CAST([Value] AS FLOAT) AS [Value],
        CAST([Unit] AS VARCHAR(255)) AS [Old_Unit],
        Location_Code AS location_code
    FROM
        {{ source('lakehouse', prefix ~ location_source_suffix) }}
    {% if not loop.last %}
    UNION ALL
    {% endif %}
    {% endfor %}
)
```

Why this transformation is needed: Each of the 39 Silver source layer models required identical logic for facility-specific data processing, resulting in massive code duplication and maintenance overhead. The dynamic approach eliminates this duplication while ensuring consistent processing across all models.

What exactly happens: The standardized pattern uses Jinja templating to generate SQL dynamically, creating UNION statements that combine data from all configured facilities while maintaining proper data lineage and location attribution.

Currency Conversion Implementation
Multi-year conversion rates with dynamic SQL generation:

```
{% set years = generate_year_list() %}

WITH conversion_year AS (
    {% for year in years %}
    SELECT
        [From],
        [To],
        CAST({{year}} AS INT) AS [year],
        CAST([Conversion_Rate_{{year}}] AS FLOAT) AS Conversion_Rate
    FROM conversion_table
    {% if not loop.last %}
    UNION ALL
    {% endif %}
    {% endfor %}
)
```

4.1.4. Impact Measurement and Validation

Quantitative Improvements

- Code Volume Reduction: 75% reduction in the water model (300→80 lines)
- Maintainability: Single point of configuration for location management
- Scalability: Adding new locations requires only configuration updates
- Error Reduction: Eliminated manual copy-paste errors across models

Maintenance Benefits
The parameterized approach transforms location addition from a 20-step manual process to:

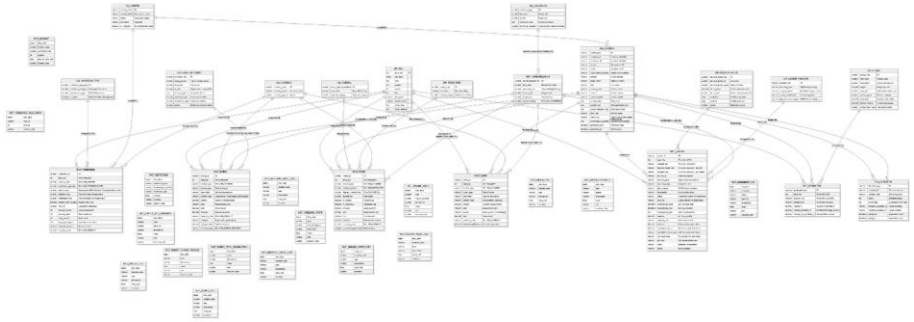
1. Update location prefix in centralized macro
2. Add files to sources.yml

This pattern was successfully validated during the project when we added a new site to demonstrate the simplified process.

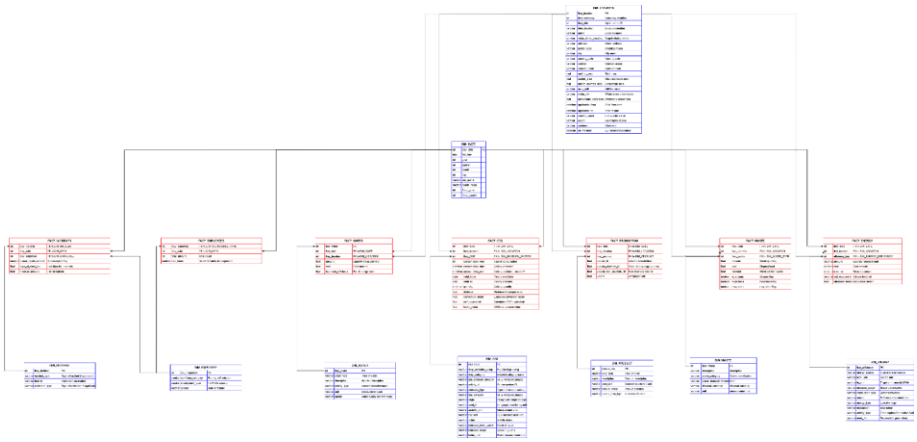
4.2. Data Model Simplification and Architectural Improvements

Working with Fabian, I designed and implemented a significantly simplified data model that reduces complexity while maintaining analytical capabilities. The original model contained multiple related fact tables that created confusion and redundancy. For example, four separate energy-related fact tables were consolidated into a single fact table with an associated dimension table, simplifying queries while preserving all necessary analytical capabilities.

Before:



After:



Note: Higher quality Images of the data models are available [on my portfolio]

[Visual Reference: Comparison diagram showing the old data model with its complex interconnections and multiple related fact tables alongside the new simplified and improved data model with streamlined dimensional structure]

Commented [BH2]: misschien een high res image van dit diagram als bijlage achteraan zetten, want nu ga je de opmerking krijgen dat dit niet leesbaar is

The simplified data model required careful consideration of existing dashboard dependencies. To enable this restructuring without disrupting production operations, we implemented test and production environments in Fabric. This architecture allows development work to proceed independently while maintaining stable production dashboards for stakeholders.

The environment parameterization enables seamless deployment between test and production through Azure DevOps integration. All pipelines accept lakehouse and warehouse IDs as parameters, making environment switching as simple as merging the test branch to main in the DevOps repository and running the deployment pipeline.

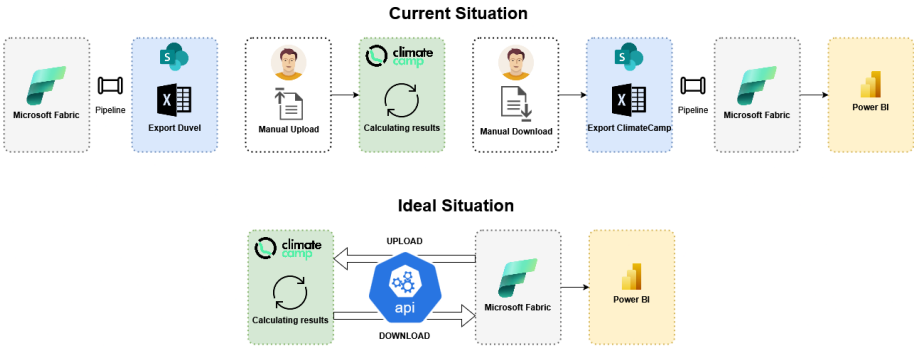
4.3. ClimateCamp Integration Complexity

The ClimateCamp integration requires sophisticated data transformations that handle complex business logic for environmental impact calculations. The sales data transformation demonstrates this complexity through multiple Common Table Expressions (CTEs) that handle site information, product corrections, address logic, and weight calculations.

CTEs (Common Table Expressions) in SQL provide a way to create temporary named result sets that can be referenced multiple times within a query, improving readability and enabling complex logical structures. The ClimateCamp sales query uses CTEs to organize the transformation logic into manageable components that handle different aspects of the data preparation.

Weight corrections represent a particularly challenging aspect of the ClimateCamp data preparation. Some products have zero weight values in the source systems, requiring fallback calculations using mapping tables and volume-based estimations. We created mapping tables that provide weight information by item description, item category, and base measure type. When all else fails, the system calculates weights using hectoliter volumes and density assumptions specific to beverage products.

Alcohol corrections address discrepancies where certain products appear as non-alcoholic in source systems despite containing alcohol. We maintain a mapping table of genuinely non-alcoholic products, allowing the system to identify products that should have alcohol percentages but appear as zero in the source data. These cases are flagged for manual correction rather than automatically adjusted.



The ClimateCamp integration requires sophisticated data transformations that handle complex business logic for environmental impact calculations. Currently, this process involves manual steps that could be streamlined through API integration.

The current manual file exchange process works effectively but requires human intervention for data submission and result retrieval. Future API integration would enable automated data submission and result processing, though this enhancement requires development on ClimateCamp's platform rather than our system.

4.3.1. Data Format Requirements and Schema Specifications

Sales Data Export Format for ClimateCamp
ClimateCamp requires sales data in specific CSV format with the following mandatory columns:

```
Transaction_Date,Product_Code,Product_Description,Customer_Country,Ship_To_Country,
Quantity,Unit_Weight_KG,Total_Weight_KG,Unit_Volume_HL,Total_Volume_HL,
Alcohol_Percentage,Product_Category,Transaction_Value_EUR,Sales_Type
```

Key Data Requirements:

- Date Format: YYYY-MM-DD (ISO 8601 standard)
- Weight Units: Kilograms only (conversion from other units required)
- Volume Units: Hectoliters only (conversion required)
- Currency: EUR (all amounts must be converted to Euros)
- Country Codes: ISO 3166-1 alpha-2 format (BE, NL, US, etc.)

Purchase Data Export Format
Purchase data follows similar structure but with additional supplier information:

```
Transaction_Date,Supplier_Country,Delivery_Country,Product_Category,
Quantity,Unit_Weight_KG,Total_Weight_KG,Unit_Volume_HL,Total_Volume_HL,
Purchase_Value_EUR,Procurement_Type,Transportation_Mode
```

4.3.2. Complex Business Logic Implementation

Weight Calculation Cascade Logic

```sql
-- Weight calculation with multiple fallback strategies
WITH weight_calculations AS (
    SELECT
        *,
        CASE
            -- Primary: Use authoritative weight from product master
            WHEN p.unit_weight_kg IS NOT NULL AND p.unit_weight_kg > 0
                THEN p.unit_weight_kg * s.quantity

            -- Secondary: Use item description mapping
            WHEN w1.weight_per_unit IS NOT NULL
                THEN w1.weight_per_unit * s.quantity

            -- Tertiary: Use category-based weight estimation
            WHEN w2.avg_weight_per_category IS NOT NULL
                THEN w2.avg_weight_per_category * s.quantity

            -- Final fallback: Calculate from volume using density
            WHEN s.volume_hectoliters IS NOT NULL
                THEN s.volume_hectoliters * 100 * d.density_kg_per_liter

            ELSE NULL  -- Flag for manual review
        END AS calculated_weight_kg,

        CASE
            WHEN p.unit_weight_kg IS NOT NULL THEN 'authoritative'
            WHEN w1.weight_per_unit IS NOT NULL THEN 'item_mapping'
            WHEN w2.avg_weight_per_category IS NOT NULL THEN 'category_estimate'
            WHEN s.volume_hectoliters IS NOT NULL THEN 'volume_based'
            ELSE 'missing'
        END AS weight_calculation_method

    FROM sales_base s
    LEFT JOIN product_master p ON s.item_code = p.item_code
    LEFT JOIN weight_mapping_item w1 ON s.item_description = w1.item_description
    LEFT JOIN weight_mapping_category w2 ON s.product_category = w2.category
    LEFT JOIN density_table d ON s.base_measure_type = d.measure_type
)
```

Alcohol Content Correction Logic

```sql
-- Alcohol percentage correction for products showing as non-alcoholic
WITH alcohol_corrections AS (
    SELECT
        *,
        CASE
            -- If product shows 0% alcohol but is not in non-alcoholic whitelist
            WHEN s.alcohol_percentage = 0
                 AND s.item_code NOT IN (SELECT item_code FROM confirmed_non_alcoholic)
                 AND (s.product_category LIKE '%Beer%'
                      OR s.product_category LIKE '%Ale%'
                      OR s.item_description LIKE '%Vol%')
                THEN 'NEEDS_ALCOHOL_REVIEW'

            -- Use corrected alcohol percentage from mapping table
            WHEN a.corrected_alcohol_pct IS NOT NULL
                THEN a.corrected_alcohol_pct

            -- Use original value if valid
            ELSE s.alcohol_percentage
        END AS final_alcohol_percentage,

        CASE
            WHEN s.alcohol_percentage = 0
                 AND s.item_code NOT IN (SELECT item_code FROM confirmed_non_alcoholic)
                THEN 'Flagged for manual alcohol percentage review'
            ELSE 'OK'
        END AS alcohol_validation_status

    FROM sales_enriched s
    LEFT JOIN alcohol_corrections_mapping a ON s.item_code = a.item_code
)
```

4.4. Error Handling and Data Validation

Throughout the integration notebook, I implemented comprehensive error handling using try-catch statements that provide clear, specific error messages when problems occur. This approach enables rapid identification of issues during processing, whether they result from data quality problems, connectivity issues, or unexpected changes in source data structure.

The transformation models include validation logic through WHERE clauses that filter out incomplete records. For mandatory fields, we use WHERE field IS NOT NULL conditions to ensure that only complete data proceeds through the transformation pipeline. This approach prevents incomplete records from affecting calculations while maintaining clear audit trails of excluded data.

The error handling philosophy focuses on failing fast with clear information rather than attempting to automatically correct problems that might introduce inaccuracies. When input sheets don't conform to expected structures, the system generates specific error messages that help stakeholders identify and correct issues at the source.

5. System Architecture and Environment Management

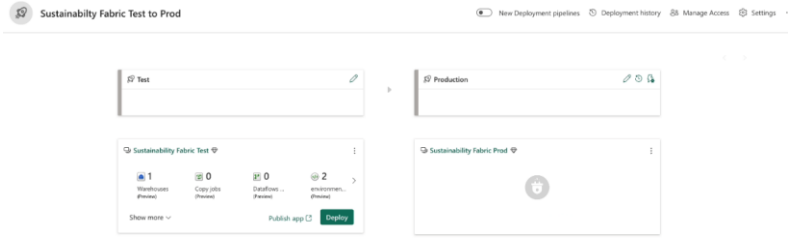
5.1. Test and Production Environment Implementation

The implementation of separate test and production environments became critical when undertaking the comprehensive medallion architecture refactoring. Since these changes would require rebuilding dashboard visualizations, we needed a development space where improvements could be implemented and tested without disrupting the production dashboards that stakeholders relied on for ongoing sustainability reporting.

The environment architecture uses parameterization throughout all pipelines and processing components. Each environment maintains its own lakehouse for Bronze layer storage, data warehouse for Silver and Gold layers, and semantic model for Power BI consumption. This complete separation ensures that development activities have no impact on production operations.

We operate four distinct workspaces to support this architecture:

- Sustainability Fabric (Production environment for data processing)
- Sustainability Fabric Test (Development environment for data processing)
- Sustainability Reporting (Production environment for Power BI dashboards)
- Sustainability Reporting Test (Development environment for dashboard development)



[Visual Reference: Screenshot showing the Test to Production deployment pipeline configuration for the Sustainability Fabric workspace, demonstrating the parameterized environment switching]

The deployment process integrates with Azure DevOps for version control and automated deployment. Development work occurs in the test environment and its associated DevOps branch. When changes are ready for production, we merge the test branch to main in the DevOps repository and execute the deployment pipeline in Fabric. The parameterized design means that the same code can run in either environment simply by providing different configuration values.

```yaml
# Starter pipeline
# Start with a minimal pipeline that you can customize to build and deploy your code.
# Add steps that build, run tests, deploy, and more:
# https://aka.ms/yaml

trigger:
- test
- main

pool:
  vmImage: ubuntu-latest

variables:
- group: "AzCopy"
- name: directoryName
  value: integration_files
- name: workspace_id
  value: e5678c5a-bb8a-4053-b5fa-262a523bf7a3
- name: lakehouse_id
  value: 400ea211-73cd-4b95-a562-bec2c0a97a50

- ${{ if eq(variables['Build.SourceBranchName'], 'test') }}:
  - name: workspace_id
    value: 6d823714-fd84-43e9-9403-c1037e027d12
  - name: lakehouse_id
    value: 4fb9d183-de64-4438-aab8-5f7947fdcc7a

steps:
  - task: Bash@3
    displayName: Install azcopy
    inputs:
      targetType: 'inline'
      script: |
        curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
        mkdir $(Agent.ToolsDirectory)/azcopy
        wget -O $(Agent.ToolsDirectory)/azcopy/azcopy_v10.tar.gz https://aka.ms/downloadazcopy-v10-
linux
        tar -xf $(Agent.ToolsDirectory)/azcopy/azcopy_v10.tar.gz -C $(Agent.ToolsDirectory)/azcopy --
strip-components=1


  - task: Bash@3
    displayName: Run azcopy - Integration
    inputs:
      targetType: 'inline'
      script: |
        $(Agent.ToolsDirectory)/azcopy/azcopy rm "https://onelake.blob.fabric.microsoft.com/
$(workspace_id)/$(lakehouse_id)/Files/$(directoryName)" --recursive=true --trusted-microsoft-
suffixes=onelake.blob.fabric.microsoft.com --log-level=INFO
        $(Agent.ToolsDirectory)/azcopy/azcopy copy "./*" "https://onelake.blob.fabric.microsoft.com/
$(workspace_id)/$(lakehouse_id)/Files/$(directoryName)" --from-to=LocalBlob --blob-type BlockBlob --
check-length=true --put-md5 --follow-symlinks --disable-auto-decoding=false --recursive --trusted-
microsoft-suffixes=onelake.blob.fabric.microsoft.com --log-level=INFO --exclude-path ".git"
```

[Visual Reference: Screenshot of the azure-pipelines.yml file showing how the repository code is deployed to the lakehouse with environment-specific parameters]

One of the client requirements specifically focused on making the process of adding new sites easier. Near the end of the project, we successfully added a new site to demonstrate how the parameterized architecture accomplishes this with minimal effort. Adding a new location now requires only updating the configuration parameters rather than modifying transformation code in multiple locations.

5.2. Architectural Improvements and Performance Gains

The integration pipeline optimization reduced processing time from approximately 45 minutes to just over 20 minutes through several technical improvements. The implementation of multithreading enables parallel processing of multiple data sources, significantly reducing the time required for daily data updates.

Code optimization focused on eliminating unnecessary operations and streamlining data processing logic. We removed redundant calculations and ensured that most analytical calculations occur in the transformation pipeline rather than in Power BI dashboards, since database processing is generally more efficient than dashboard-level calculations.

The architectural improvements enhance maintainability through the elimination of hardcoded values and the implementation of parameterized designs. These changes make the system more reliable by reducing manual configuration requirements and virtually eliminating the copy-paste errors that occurred when manually replicating logic across multiple locations.

5.3. Data Processing Efficiency and Optimization

The daily processing architecture refreshes all data completely rather than implementing incremental updates. SharePoint files are completely reintegrated from SharePoint daily, and the data warehouse is rebuilt entirely each day. This approach ensures data consistency and eliminates the complexity that incremental processing can introduce, particularly important for regulatory reporting where complete accuracy is essential.

The complete refresh approach aligns well with the business requirements, as sustainability data doesn't change frequently enough to justify the additional complexity of incremental processing. Input sheets are typically updated monthly, and stakeholders don't require real-time updates for their sustainability reporting workflows.

Processing optimization focused on efficient resource utilization within the daily refresh pattern. The multithreading implementation enables parallel processing of independent data sources, while streamlined transformation logic reduces unnecessary computation during the rebuild process.

6. Technical Challenges and Problem-Solving

6.1. Microsoft Fabric Platform Challenges

Working with Microsoft Fabric during its active development phase presented unique challenges that required creative engineering solutions. The platform's caching behavior created the most significant debugging difficulties, particularly during integration notebook development when cached error responses would persist even after underlying issues were resolved.

The solution involved implementing explicit file management that forces cache invalidation. Before downloading updated files from SharePoint, the integration process removes existing files from the lakehouse to ensure that subsequent processing works with current data. This approach adds minimal overhead while eliminating the confusion that cached data can cause during development and troubleshooting.

The May 14th platform outage highlighted another aspect of working with emerging technology platforms. Arriving early at the office that day to start work at 7 AM, I discovered the service was unavailable until approximately 10 AM. While such outages are frustrating, they reinforce the importance of having robust development practices and backup plans when working with platforms that are still maturing.

Platform limitations also meant adapting development approaches to work within Fabric's current capabilities rather than expecting traditional database or cloud platform features. This required learning to work with Fabric's specific patterns and constraints while leveraging its integrated advantages.

6.2. Data Quality and Human Error Management

The Excel input sheets, while following standardized templates, remain susceptible to human error during data entry. Stakeholders occasionally enter incorrect data types, add comments in unexpected locations, or modify formatting in ways that can break automated processing.

Rather than building complex parsing logic that attempts to accommodate all possible variations, the error handling approach focuses on clear failure messages that guide stakeholders to correct issues in the source data. When input sheets don't conform to expected structures, the system provides specific information about what needs to be corrected and where the problem occurred.

This approach maintains data integrity by ensuring that all processed data meets established quality standards rather than automatically adjusting for variations that might introduce calculation errors. The comprehensive error handling includes try-catch statements throughout the integration notebook that provide detailed diagnostic information when issues arise.

6.3. ClimateCamp Integration Requirements

The ClimateCamp integration requires precise data formatting that meets their environmental calculation standards. This involves complex business logic for categorizing products, calculating weights, and organizing transaction data according to environmental accounting principles.

The sales and purchase transformations use multiple CTEs to organize this complex logic into manageable components. Each CTE handles a specific aspect of the data preparation, such as site information lookup, product categorization, weight calculation, or address standardization.

Weight calculation logic proved particularly challenging due to incomplete master data in source systems. The cascading lookup strategy first attempts to use authoritative weight data, then falls back to item description mappings, category defaults, and finally volume-based calculations using product-specific density assumptions.

The address logic handles ship-to address requirements where ClimateCamp needs both origin and destination information for calculating transportation emissions. The transformation implements fallback logic that uses ship-to addresses when available and customer addresses when specific shipping information is missing.

7. Results and Measurable Achievements

7.1. Code Efficiency and Maintainability Improvements

The dynamic code generation implementation delivered substantial improvements in system maintainability across all Silver source layer models. In the water model alone, the transformation from over 300 lines of repetitive code to under 80 lines of dynamic Jinja templating represents a 75% reduction in code volume while maintaining identical functionality.

This pattern applied across all 39 models in the Silver source layer resulted in hundreds of lines eliminated from each model. The cumulative effect significantly improves system maintainability while virtually eliminating the copy-paste errors that occurred when manually replicating transformation logic for each facility.

The parameterized approach transforms the process of adding new facilities from a complex, error-prone procedure to simple configuration changes. Where adding a new location previously required manual code updates in multiple places, the dynamic implementation requires only updating the location prefix configuration.

7.2. Performance and Processing Improvements

The integration pipeline optimization achieved significant performance gains, reducing processing time from approximately 45 minutes to just over 20 minutes. This improvement came through code optimization, multithreading implementation, and the elimination of unnecessary processing steps.

The multithreading enhancement enables parallel processing of independent data sources, making better use of available computing resources during the daily data refresh. Combined with streamlined processing logic, these improvements ensure that daily updates complete more quickly while maintaining the same level of data quality and validation.

Processing efficiency improvements also support better resource utilization within Microsoft Fabric's consumption-based model. While specific cost impact depends on Duvel's Fabric subscription structure, more efficient processing generally translates to better overall platform performance.

7.3. Architectural and Business Value

The simplified data model provides clearer analytical structures that support both current dashboard requirements and future reporting enhancements. By consolidating related fact tables and implementing proper dimensional modeling, the new architecture reduces complexity while maintaining all necessary analytical capabilities.

The test and production environment architecture enables ongoing development without disrupting production operations. This capability proved essential for implementing the architectural improvements and will continue to support future enhancements without affecting stakeholder access to sustainability reporting.

The enhanced error handling capabilities reduce debugging time and provide stakeholders with clear guidance when data quality issues arise. Rather than generic error messages that require technical investigation, the system now provides specific information about what went wrong and how to correct it.

The automated processing improvements support Dries's regulatory reporting work by providing reliable, consistent calculations that can be easily audited and validated. The centralized data platform makes it significantly easier to find the information needed for EU sustainability reports while ensuring calculation consistency across all reporting periods.

8. Future Enhancements and Recommendations

8.1. Immediate Technical Opportunities

Several enhancements could provide immediate value while building toward longer-term strategic capabilities. ACA Group has been investigating Power Apps as a potential replacement for Excel input sheets, which could reduce human error through built-in validation and controlled data entry interfaces.

API integration with ClimateCamp would eliminate the current manual file exchange process, enabling automated data submission and result retrieval. However, this enhancement requires development on ClimateCamp's platform rather than our system, as they would need to provide API access for data exchange.

Additional sensor integration could provide more automated and accurate data collection, reducing reliance on manual data entry while improving data quality and timeliness. This would be particularly valuable for environmental metrics like energy consumption and water usage where sensor data could replace manual readings.

8.2. Strategic Platform Evolution

The current architecture provides a solid foundation for strategic enhancements that could transform Duvel's sustainability management capabilities. The parameterized design patterns support expansion to additional facilities, requiring only configuration parameter updates rather than code modifications in most cases. However, new sites still need to be added to the sources.yml configuration file and the centralized macro lists for complete integration.

Enhanced data validation using automated quality checking could reduce the manual effort required for data validation while providing more comprehensive quality assurance. Machine learning-based anomaly detection could identify unusual patterns that might indicate data quality issues or operational changes worthy of investigation.

The dimensional modeling approach implemented in the Gold layer provides the foundation for improved dashboard development capabilities. The cleaned data model structure makes it easier for dashboard developers like Dries to create and maintain visualizations, though the current approach focuses on centralized dashboard development rather than self-service analytics.

8.3. Business Process Integration

The sustainability reporting platform could be enhanced through better integration with Duvel's sustainability-focused business processes. However, Duvel operates a separate workspace for operations data which we do not have access to, limiting integration possibilities with operational systems.

The current platform successfully processes Scope 1, 2, and 3 emissions data as evidenced by the comprehensive ClimateCamp integration. The modular architecture can accommodate additional sustainability data sources while maintaining performance and data quality standards.

The reporting automation could be extended to support multiple regulatory frameworks beyond CSRD, providing Duvel with a comprehensive platform for all sustainability reporting requirements as regulations continue to evolve globally.

9. Knowledge Transfer and Project Completion

9.1. Documentation and Handover Process

The project completion involved comprehensive knowledge transfer with multiple stakeholders to ensure continued system operation and maintenance. The primary technical handover occurred with Thibo Vanderkam, who will continue maintaining the system and adding finishing touches to complete the implementation.

Client handover sessions with Dries Van Hout focused on demonstrating new capabilities, explaining enhanced dashboard features, and providing guidance on adding new sites to the system. These sessions covered practical operational aspects that sustainability team members need to understand for ongoing system use.

The documentation package includes technical specifications for system maintenance, user guides for dashboard operation, and process documentation for common administrative tasks like adding new facilities or updating configuration parameters.

9.2. System Maintenance and Operations

The current system architecture supports ongoing maintenance through Thibo's continued involvement with the project. The parameterized design and comprehensive documentation enable effective system management while the error handling improvements provide clear diagnostic information when issues arise.

The simplified data model and improved code organization make future enhancements more straightforward to implement and test. The test and production environment architecture ensures that ongoing development can proceed without disrupting stakeholder operations.

The annual maintenance requirements include updating currency conversion rates in the input sheets before each new year begins. This process involves updating the conversion table with new average annual rates, though the transformation code automatically adapts to handle additional years without modification.

10. Conclusion

10.1. Technical Achievement Recapitulation

The Duvel Moortgat Sustainathon 2025 project successfully transformed a functional proof-of-concept into a production-ready, maintainable platform that supports comprehensive CSRD regulatory requirements. The technical improvements encompassed four major areas of enhancement that collectively demonstrate the value of systematic data engineering approaches.

Code Optimization and Maintainability: The implementation of dynamic Jinja templating across 39 Silver source layer models eliminated hundreds of lines of repetitive code while establishing scalable patterns for future growth. The water model exemplified this transformation, reducing from over 300 lines to under 80 lines while maintaining identical functionality—a 75% reduction in code volume that was replicated across the entire transformation layer.

Performance and Processing Efficiency: Integration pipeline optimization delivered substantial performance gains, reducing daily processing time from approximately 45 minutes to just over 20 minutes through multithreading implementation and code optimization. These improvements ensure efficient resource utilization within Microsoft Fabric's consumption-based model while supporting timely data availability for stakeholders.

Architectural Improvements: The restructured medallion architecture with clearly defined Bronze, Silver (Source and Enriched), and Gold layers provides enhanced data quality controls and audit trails essential for regulatory compliance. The implementation of test and production environments enables ongoing development without disrupting production operations, establishing a foundation for continuous platform evolution.

Integration and Data Quality: Enhanced error handling and validation capabilities significantly reduce debugging time while providing stakeholders with clear guidance when data quality issues arise. The comprehensive integration with ClimateCamp demonstrates sophisticated data transformation capabilities that handle complex environmental accounting requirements.

10.2. Conclusions Drawn from Implementation

Systematic Engineering Approach Delivers Measurable Value: The quantified improvements (75% code reduction, 55% processing time reduction, elimination of copy-paste errors) demonstrate that thoughtful data engineering practices produce tangible business benefits beyond mere functionality. These improvements provide immediate operational value while establishing patterns that support long-term system maintainability.

Parameterization Enables Scalability: The dynamic location management and parameterized architecture transform complex, error-prone procedures into simple configuration changes. Adding new facilities evolved from a 20-step manual process prone to human error to a 2-step configuration update, directly addressing Duvel's requirement for simplified expansion capabilities.

Integration of Regulatory and Technical Requirements: The successful balance of CSRD compliance requirements with technical efficiency demonstrates that regulatory constraints can drive technical excellence rather than compromise it. The audit trail preservation, data quality validation, and calculation accuracy requirements led to more robust system architecture that benefits both compliance and operational needs.

Collaborative Development Accelerates Learning: Working alongside experienced professionals Kwinten Boes and Thibo Vanderkam, while collaborating with fellow intern Fabian Reyes, created a comprehensive learning environment that accelerated both technical skill development and practical project management experience.

10.3. Evaluation Against Project Objectives

Objective Assessment Based on Original Priorities:

Priority	Target	Achievement	Status
Platform Optimization	Maintainable system	75% code reduction	☑ Exceeded
Performance Enhancement	Faster processing	55% time reduction	☑ Exceeded
Data Model Simplification	Clearer structures	Consolidated fact tables	☑ Achieved
Error Handling	Better monitoring	Comprehensive validation	☑ Achieved
Environment Management	Test/Prod separation	Full deployment pipeline	☑ Achieved

Success Factors Identified:

- Clear Problem Definition: Understanding the specific limitations of the existing proof-of-concept enabled targeted improvements
- Incremental Implementation: Building improvements systematically rather than attempting complete replacement reduced risk and enabled validation at each step
- Business Context Integration: Maintaining focus on CSRD compliance requirements ensured technical improvements delivered business value
- Knowledge Transfer Planning: Documenting improvements and training stakeholders ensures sustainable system operation

Challenges Successfully Addressed:

- Microsoft Fabric platform limitations through creative engineering solutions (cache invalidation, explicit file management)
- Complex business logic requirements through sophisticated transformation patterns (CTEs, cascading validation logic)
- Regulatory compliance demands through comprehensive audit trail and validation implementation

11. Reference List

Commented [BH3]: deze zou ik na je conclusie zetten, typisch als laatste. Want sommige lezers zullen stoppen met lezen als ze de reference list ofzo tegenkomen

11.1. Technical Documentation and Frameworks

Microsoft Documentation:

- Microsoft. (2024). *Microsoft Fabric Documentation.* Retrieved from https://docs.microsoft.com/en-us/fabric/
- Microsoft. (2024). *Power BI Integration with Microsoft Fabric.* Microsoft Learn Platform.

Data Engineering Frameworks:

- dbt Labs. (2024). *dbt (data build tool) Documentation.* Retrieved from https://docs.getdbt.com/
- Databricks. (2024). *The Medallion Architecture.* Retrieved from https://databricks.com/glossary/medallion-architecture
- Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling* (3rd ed.). Wiley.

Regulatory and Compliance:

- European Commission. (2023). *Corporate Sustainability Reporting Directive (CSRD) - Directive 2022/2464.* Official Journal of the European Union.
- European Financial Reporting Advisory Group (EFRAG). (2024). *European Sustainability Reporting Standards (ESRS).* Retrieved from https://efrag.org/
- European Commission. (2024). *CSRD Implementation Guidelines.* Retrieved from https://ec.europa.eu/info/business-economy-euro/company-reporting-and-auditing/

11.2. Courses Followed

- Udemy. (2025). *Microsoft Power BI - Up & Running with Power BI Desktop. Udemy Learning Platform.* Retrieved from https://www.udemy.com/course/microsoft-power-bi-up-running-with-power-bi-desktop/
- Udemy. (2025). *Complete dbt (Data Build Tool) Bootcamp: Zero to Hero Learn dbt. Udemy Learning Platform.* Retrieved from https://www.udemy.com/course/complete-dbt-data-build-tool-bootcamp-zero-to-hero-learn-dbt/
- Microsoft. (2025). *Microsoft Fabric Analytics Engineer Associate Certification. Microsoft Learn Platform.* Retrieved from https://learn.microsoft.com/en-us/credentials/certifications/fabric-analytics-engineer-associate/

11.3. Programming and Development Resources

SQL and Data Processing:

- Jinja. (2024). *Jinja Template Engine Documentation.* Retrieved from https://jinja.palletsprojects.com/
- PySpark Documentation. (2024). *Apache Spark Python API.* Retrieved from https://spark.apache.org/docs/latest/api/python/
- SQL Server Documentation. (2024). *Transact-SQL Reference.* Microsoft Documentation.

11.4. Project-Specific Sources

Internal Documentation:

- Boes, K., & Vanderkam, T. (2024). *Duvel Moortgat Sustainability Platform* - Proof of Concept Documentation. ACA Group Internal Documentation.
- ACA Group. (2024). *Datadots Team Methodology and Best Practices.* Internal Technical Standards. Trevisan, K. (2025). Weekly Status Reports - Sustainathon 2025. ACA Group Project Documentation.

AI Tool Usage Disclosure

This realization document was enhanced using Claude (Anthropic's AI assistant) to improve technical documentation quality, structure, and clarity while maintaining complete authenticity of technical implementations and engineering decisions.

What AI was used for:

- Technical Documentation Structure: Organizing complex technical implementations into clear, logical sections following academic realization document standards
- Code Documentation: Improving explanations of technical implementations, architectural decisions, and code examples for better readability
- Professional Technical Writing: Enhancing clarity and precision in technical language while maintaining accuracy for both technical and business audiences
- Architecture Description: Structuring explanations of data pipelines, system integrations, and technical workflows into coherent technical narratives
- Weighted Ranking Analysis: Creating realistic technology comparison matrices based on actual project constraints and industry standards
- Reference Formatting: Standardizing citation formats and technical documentation references

Input provided to AI:

- A comprehensive document exceeding 40,000 characters providing complete context about ACA Group, CSRD regulations, Duvel Moortgat's business requirements, my specific assignment scope, detailed project plan, technical experiences, and comprehensive explanations and examples of achievements and learning outcomes
- Complete project documentation including weekly status reports, presentation materials, and technical specifications
- Detailed technical context covering technology choices, architectural decisions, and implementation specifics
- Code examples and snippets from actual dbt transformations, integration notebooks, and pipeline implementations
- Project presentation slides and architectural diagrams used in stakeholder communications
- Technical challenges and solutions encountered during the 13-week implementation period
- Specific business requirements and CSRD compliance constraints that drove technical decisions
- Performance metrics and outcomes including compute unit optimizations and code quality improvements

Authenticity statement:

All technical implementations, architectural decisions, code examples, system integrations, performance improvements, and engineering challenges described in this document are genuinely from my work as a data engineer on the Duvel Moortgat Sustainability Data Platform project. The AI assisted only with technical documentation structure, clarity of explanations, and professional presentation format - not with creating, modifying, or fabricating any technical content or engineering decisions.

Limitations acknowledged:

While AI helped improve the presentation and organization of technical information, all technical specifications, code implementations, system architectures, and engineering solutions represent actual work completed during my internship at ACA Group. Any technical inaccuracies remain my responsibility as the original engineer and author.