NOVA IMS
Information Management School

Nova Wines S.A.

since 1973

Data Warehouse Solution

Group 10

Catarina Nunes 20230083

Jan Nosorowski 20231552

Amelie Kohl 20230134

Index of contents

Index of Figures

Index of Tables

Nova Information Management School, Lisbon, Portugal

1 Introduction

In contemporary business dynamics, robust data management tools are no longer a mere advantage but essential for enhancing decision-making across all organizational levels. As companies grow, their solutions must grow too, supporting their expansion. Data can give a competitive edge to those who understand its value.

Our project aligns with this idea, utilizing a Business Intelligence solution in the form of a Data Warehouse to optimize decision-making processes and strategically position businesses for heightened success within the market. This approach views data not merely as a resource, but as a powerful tool to transform businesses.

2 The Organization

2.1 Presentation of the organization

Our objective for the Business Intelligence project was to find a company that would allow us to put our knowledge from the classes into practice while also being of interest to us in terms of its business activities. This is why we created the fictional company "Nova Wines S.A.".

Nova Wines S.A. is a family-owned business that aims to showcase the art of Portuguese winemaking to the world. Established fifty years ago in Setúbal, it represents six distinct **wine regions**: Algarve, Bairrada, Dão, Douro, Tejo, and Vinhos Verdes.

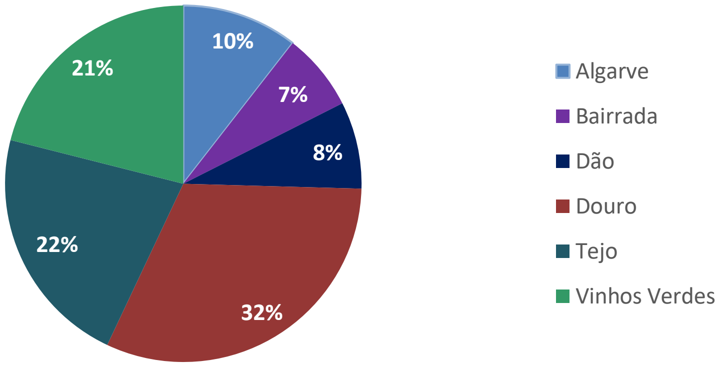


Figure 1. *Distribution of the portfolio across different Wine Regions (%)*

Operating as a **B2B enterprise**, it exports products to **519 clients** across **151 diverse countries**. The client base includes HoReCa establishments, 'enotecas', traditional retail stores, and wholesalers.

As illustrated in **Figure 2 below**, the preponderance of the company's client base is notably concentrated in the **Northern Hemisphere**. Specifically, the regions of Europe and North America collectively contribute significantly, representing almost 70% of the overall clientele.

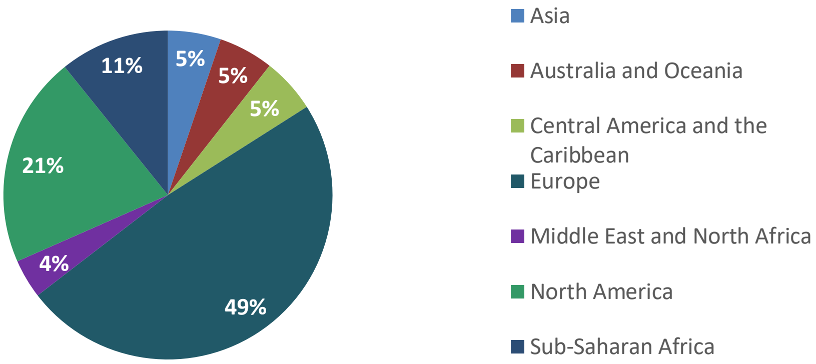


Figure 2. *Distribution of Client Count by Region (%)*

For a more comprehensive perspective, we extend our consideration of the Northern Hemisphere to encompass Europe, North America, Asia, Central America, the Caribbean, the Middle East, and North Africa. This expanded view reveals an incidence of 84%, indicating that a substantial majority of the company's clients experience summer concurrently. This alignment serves as a substantial indicator of seasonality, particularly noteworthy when analyzing alcoholic beverages. Further exploration into this correlation will be conducted in our subsequent analysis.

The company’s product range encompasses various types of alcoholic beverages, including white, red, and rosé wines, brandy, sangria, liqueur, and champagne (Category 1). Additionally, they offer a selection of snacks that pair well with the wines (Category 2). Nova Wines S.A. also provides essential utensils such as corkscrews and coolers to enhance the wine-drinking experience (Category 3). As a trading company, Nova Wines S.A. does not produce these products itself, they are sourced from 10 local producers across the different Portuguese wine regions.

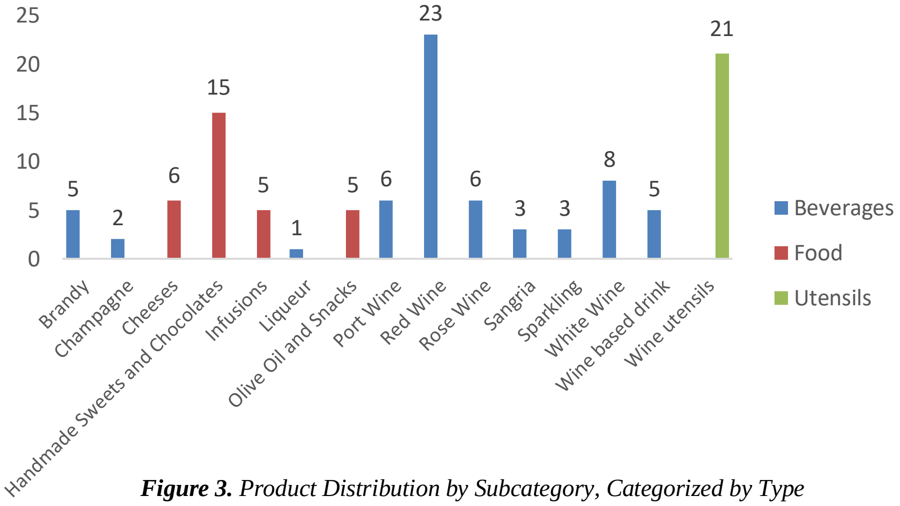


Figure 3. *Product Distribution by Subcategory, Categorized by Type*

2.2 Problem Scenario: Informational problem

Being a family-owned business with traditional roots, Nova Wines S.A. had never seriously considered the integration of Business Intelligence (BI) solutions. However, with the expansion of its operations in the past few years, the company recognized the need to enhance its decision-making processes. This led them to seek assistance from our team to leverage the power of data-driven insights. The company's fifty-year legacy in Portuguese winemaking had relied on time-tested practices, but as global demands evolved, so did Nova Wines S.A.'s approach.

First, one informational problem they seek to address is to analyze the overall **performance of their sales team**, consisting of 29 representatives across two office locations in Lisbon and Porto. Upon every successful sale, the representative gets an individual fixed commission rate. Each representative is assigned to certain clients in a certain region (depending on the segment that is arranged by market relevance). Nova Wines S.A. would like to use the BI solution to re-evaluate the structure of its salesforce. Based on the performance of the representatives, they want to make personnel decisions, such as identifying and potentially laying off underperformers or hiring additional staff. They also consider adjusting the commission rate of the representatives according to their performance.

Another informational challenge the company wishes to address with the BI solution is **forecasting**. Nova Wines S.A. recognizes that certain product types experience increased popularity during specific seasons. For instance, sparkling wines are typically enjoyed during festive occasions, making them particularly popular during holiday times. Leveraging historical sales data, the company aims to gain deeper insights into the seasonality of its products. Their objective is to create accurate forecasts for each season, allowing for adjustments in purchasing volume and potentially refining their product range.

In addition, Nova Wines S.A. wants the BI solution to delve into their **markets and client base**. Currently, boasting 519 clients spanning 151 countries worldwide, the company now aims to improve and pay more attention to the service in key markets and major customers. Specifically, they wish to cultivate strategic, long-lasting relationships. Accordingly, they seek to identify these opportunities through the BI solution.

The company did not provide much information about previous years (before 2020), as a significant portion of its records exist in paper format. Furthermore, recent data is scattered and disorganized throughout the organization, with some lost in a cyber-attack in 2019. However, they now aim to establish a consistent foundation to support future organizational needs.

2.3 Business Questions

Based on the informational problems listed in the previous chapter, we identified the following business questions for the topics of sales team, forecasting, and markets:

Table 1. *Business Questions and Measures*

Informational problem category	Business Question	Measures
Sales Team: Performance evaluation	Who are the **top 3 sales representatives/teams** in the **last 12 months** based on profit?	total_profit
	Which **3 sales representatives/teams exhibited the lowest performance** in the **last 12 months** based on profit?	total_profit
	Which **3 sales representatives/teams** demonstrated the **most significant improvement in performance** from the **previous year to this year**, based on profit?	total_profit
	Which **3 sales representatives/teams** exhibited the **least growth in performance** from the **previous year to this year**, based on profit?	total_profit
	Is the **commission rate positively correlated to the sales generated by each representative** in the **past 12 months**? (High performers should have the highest commission rate)	total_profit
	How can we reorganize the **segments** (Top Level and Mid-Level Regions), considering the **profit per Region over the past three years?**	total_profit
Forecasting	What are the **top 10 and bottom 10 performing products overall**, sold in **each year**, as determined by sales quantity?	quantity_sold
	What are the **top 3 and bottom 3 performing product subtypes** sold in **each quarter, each year**, based on sales quantity?	quantity_sold
	What is the **best and worst performing product** within each product **type**, determined by sales quantity?	quantity_sold
	What is the **top-performing and bottom-performing product** within each **brand**, as determined by sales quantity?	quantity_sold
	What is the **quarterly distribution of annual sales quantity for each product category** (beverages, food, and utensils)? Specifically, what percentage of total sales for each category is attributed to each quarter, based on historical data **(2020-2023)**?	quantity_sold

	What are the **top 10 and bottom 10 performing products overall**, sold each **year**, based on profit?	total_profit
	What is the **annual growth rate in the quantity of products sold** and the corresponding **revenue** (in euros) for NOVA Wines **from 2020 to 2023**?	quantity_sold total_profit
Markets: Global operations	Who are the **top 5 clients** with the **highest sales quantity** from **2020 to 2023**?	quantity_sold
	Who are the **top 5 clients** with the **most significant growth in sales quantity** from **2020 to 2023**?	quantity_sold
	What are the **top 5 countries** with the **highest sales quantit**y from **2020 to 2023**?	quantity_sold
	Which are the **top 5 countries** with the **most significant growth in sales quantity** from **2020 to 2023**?	quantity_sold
	What is the **top-performing product** in each **country**, as determined by sales quantity?	quantity_sold

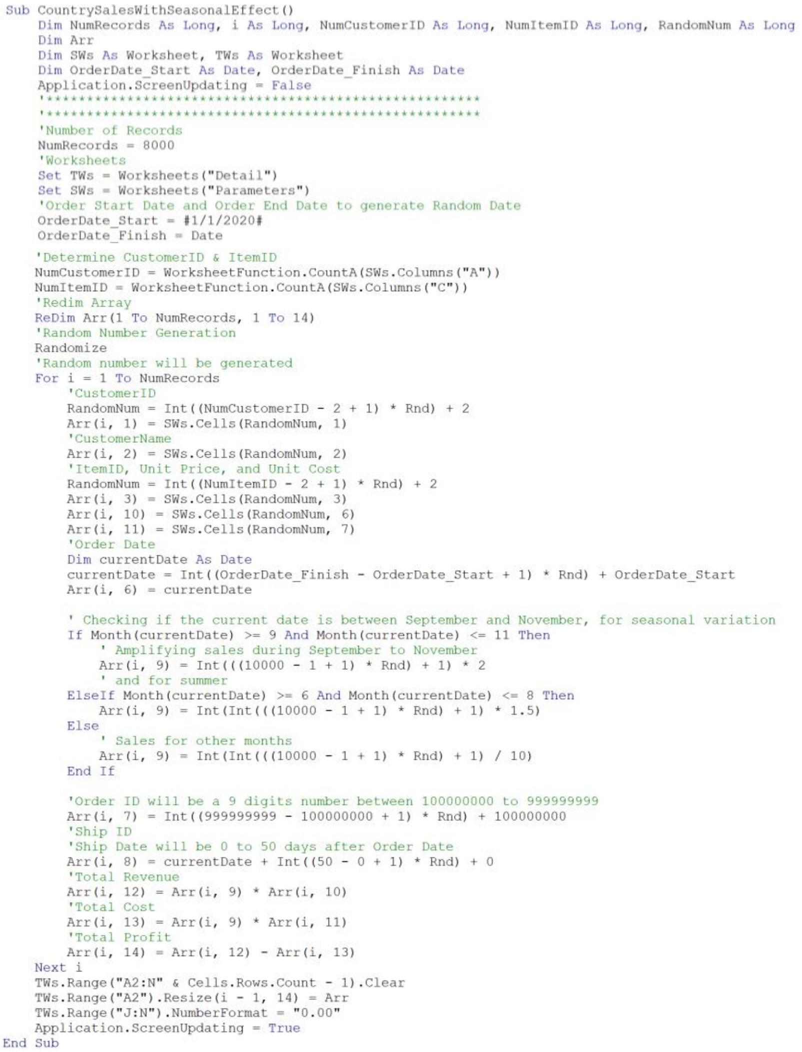
3 Original Data Sources

As mentioned earlier, we crafted our dataset from scratch. After choosing a domain that allowed us flexibility in hierarchical and analytical analysis, we opted for an area closely associated with more traditional methods, yet rich in points where we could draw inspiration for our fictional company, grounded in the reality of a winery, Nova Wines SA.

3.1 Data Source

Our data source is in Excel format. We generated this information with a transactional database in mind, drawing from our experience working in similar companies. It captures various details related to sales, clients, the sales team, suppliers, and product information.

To generate a substantial volume of sales records, we developed the following VBA code as an Excel macro:



```
Sub CountrySalesWithSeasonalEffect()
    Dim NumRecords As Long, i As Long, NumCustomerID As Long, NumItemID As Long, RandomNum As Long
    Dim Arr
    Dim SWs As Worksheet, TWs As Worksheet
    Dim OrderDate_Start As Date, OrderDate_Finish As Date
    Application.ScreenUpdating = False
    '*******************************************************
    '*******************************************************
    'Number of Records
    NumRecords = 8000
    'Worksheets
    Set TWs = Worksheets("Detail")
    Set SWs = Worksheets("Parameters")
    'Order Start Date and Order End Date to generate Random Date
    OrderDate_Start = #1/1/2020#
    OrderDate_Finish = Date

    'Determine CustomerID & ItemID
    NumCustomerID = WorksheetFunction.CountA(SWs.Columns("A"))
    NumItemID = WorksheetFunction.CountA(SWs.Columns("C"))
    'Redim Array
    ReDim Arr(1 To NumRecords, 1 To 14)
    'Random Number Generation
    Randomize
    'Random number will be generated
    For i = 1 To NumRecords
        'CustomerID
        RandomNum = Int((NumCustomerID - 2 + 1) * Rnd) + 2
        Arr(i, 1) = SWs.Cells(RandomNum, 1)
        'CustomerName
        Arr(i, 2) = SWs.Cells(RandomNum, 2)
        'ItemID, Unit Price, and Unit Cost
        RandomNum = Int((NumItemID - 2 + 1) * Rnd) + 2
        Arr(i, 3) = SWs.Cells(RandomNum, 3)
        Arr(i, 10) = SWs.Cells(RandomNum, 6)
        Arr(i, 11) = SWs.Cells(RandomNum, 7)
        'Order Date
        Dim currentDate As Date
        currentDate = Int((OrderDate_Finish - OrderDate_Start + 1) * Rnd) + OrderDate_Start
        Arr(i, 6) = currentDate

        ' Checking if the current date is between September and November, for seasonal variation
        If Month(currentDate) >= 9 And Month(currentDate) <= 11 Then
            ' Amplifying sales during September to November
            Arr(i, 9) = Int(((10000 - 1 + 1) * Rnd) + 1) * 2
            ' and for summer
        ElseIf Month(currentDate) >= 6 And Month(currentDate) <= 8 Then
            Arr(i, 9) = Int(Int(((10000 - 1 + 1) * Rnd) + 1) * 1.5)
        Else
            ' Sales for other months
            Arr(i, 9) = Int(Int(((10000 - 1 + 1) * Rnd) + 1) / 10)
        End If

        'Order ID will be a 9 digits number between 100000000 to 999999999
        Arr(i, 7) = Int((999999999 - 100000000 + 1) * Rnd) + 100000000
        'Ship ID
        'Ship Date will be 0 to 50 days after Order Date
        Arr(i, 8) = currentDate + Int((50 - 0 + 1) * Rnd) + 0
        'Total Revenue
        Arr(i, 12) = Arr(i, 9) * Arr(i, 10)
        'Total Cost
        Arr(i, 13) = Arr(i, 9) * Arr(i, 11)
        'Total Profit
        Arr(i, 14) = Arr(i, 12) - Arr(i, 13)
    Next i
    TWs.Range("A2:N" & Cells.Rows.Count - 1).Clear
    TWs.Range("A2").Resize(i - 1, 14) = Arr
    TWs.Range("J:N").NumberFormat = "0.00"
    Application.ScreenUpdating = True
End Sub
```

Figure 4. *VBA code that generates the Sales Orders Data*

This macro was specifically designed to allow the specification of the desired quantity of sales, the chosen years, and, notably, to have analytical interest. The dataset spans transactional sales records from 2020 to 2023, totaling 8000 entries. The currency of every monetary value is Euro (EUR).

As a simplification, for the purpose of generating the data, we assume that clients place orders for only one product per order per day.

While acknowledging that this involves random number generation, part of the code introduces **seasonality -** which reflects the consideration of the wine/beverages industry's reality, where seasonal patterns are common **-** that will be utilized in subsequent analyses and can serve as a more **tangible connection to a real company**.

To explain better, the multiplier element influences the increase or decrease in sales. This narrative divides the year into three periods:

- **Pre-Christmas Season** (September to November): Occurring after the harvests in September, this period focuses on exporting to countries where orders may take several weeks to arrive, anticipating the demand leading up to Christmas.
- **Low Season** (Including December): Encompassing December, this period aligns with the conclusion of most Christmas sales to export markets, representing a slower phase in the sales cycle.
- **Summer Months** (After June): Associating with increased beverage consumption and environments conducive to consumption. Taking into consideration that a significant portion of our customers are in the North Hemisphere, aligning with the summer months in that region.

It's crucial to emphasize that **this narrative was crafted for data generation**, bearing in mind that **random data is being generated in the first place**. The multipliers used may not compensate for the generated numbers for other periods, and **exceptions may arise**. This nuanced approach maintains the analytical interest in the subsequent analysis we are going to conduct.

3.2 Dataset Structure

Our data set contains 6 main tables, divided by sheets in our Excel file 'BII_NOVAWINESSOURCE_G10': Products, Clients, Suppliers, Sales team, Orders, and Date table. We utilized ChatGPT to generate customized data in specific domains, such as names, addresses, and phone numbers, meeting our desired criteria.

- **Products sheet:** This table lays the groundwork for defining our product offerings. We list and categorize our company's products. It includes details like product categories, subcategories, cost (what we pay to the supplier), and retail price per unit.
- **Client's sheet:** This is where we establish the foundation for our global company. It holds details about our clients, including their names, shipping addresses, and other important information.
- **Suppliers sheet:** Tailored for our B2B business model, provides insights into the entities supplying our products. It includes essential information about supplier names and other relevant details.
- **Sales Team sheet:** To complete the base in illuminating Nova Wines' global and expansive nature, we let this guide us through the formation of a sales team that harmonizes with this vision. It offers essential details such as names, emails, and phone numbers, serving as a fundamental component in shaping a sales team that integrates with our international aspirations.
- **Sales Orders sheet:** The Orders information, as mentioned earlier, was generated using VBA code. It provides details about all orders placed between 2020 and 2023. It specifically captures information on order placement and shipment dates, product price and cost, and of course, the quantity ordered.

Additionally, we create a support file called 'DateDimension_SupportFile':

- **Date table:** The Date table serves as an external component, added to populate the Date dimension table. It plays a crucial role in providing temporal context and enhancing the analytical depth of our dataset.

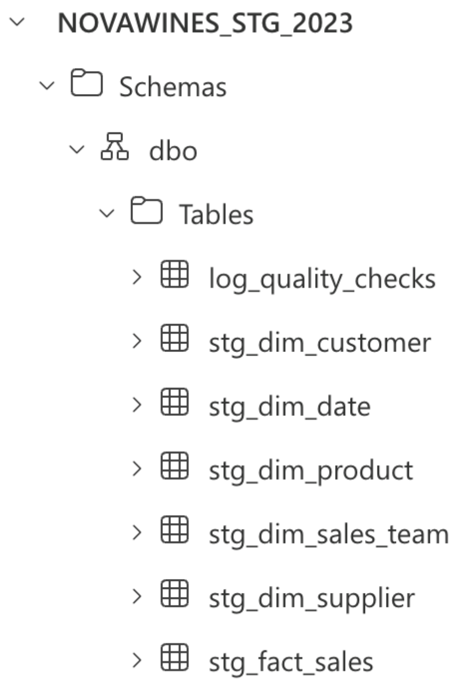
It's crucial to emphasize that despite the artificial origin of our generated data, we treat it with the same regard as real-world information. We recognize the inherent fluctuations and specific nuances embedded within, providing us the opportunity to effectively apply and optimize our Data Warehouse solution. This approach ensures that our analyses and insights remain grounded and relevant, even in a simulated context.

4 Staging Area

The Staging Area is a vital component of our data processing system, and we integrated it as a **Staging Warehouse**. It acts as a **temporary repository** for our multi-source data, allowing us to run the **ETL** (Extract, Transform, Load) processes before transferring it to the Data Warehouse. The structure of the Staging Area is similar to the one of the Data Warehouse, containing the dimension tables **stg_dim_customer**, **stg_dim_date**, **stg_dim_product**, **stg_dim_supplier**, **stg_dim_sales_team,** and the fact table **stg_fact_sales**.

However, unlike the Data Warehouse, the tables in the Staging Area are **not connected**, which enables us to make changes to the data without affecting other information. Our Staging Area design prioritizes flexibility by not connecting tables, ensuring easy integration. It doesn't enforce primary keys, promoting an agile and rule-driven approach.

Furthermore, we equip our Staging Area with an extra table named "log_quality_checks," which helps us to maintain a consistent record of ETL operations and stores the results of **data quality checks (more details on data quality checks in the ETL section)**. This setup ensures that we process data efficiently.



- NOVAWINES_STG_2023
  - Schemas
    - dbo
      - Tables
        - log_quality_checks
        - stg_dim_customer
        - stg_dim_date
        - stg_dim_product
        - stg_dim_sales_team
        - stg_dim_supplier
        - stg_fact_sales

Figure 5. *Staging Area Structure*

5 Data Warehouse

The Data Warehouse acts as a structured repository designed for analytics processes, facilitates the building of reports, and enables effective organizational decision-making.

In our Data Warehouse design process, we adopted **Kimball's** modeling methodology, also known as the bottom-up approach to denormalize the data into a **Star Schema**. This schema involves a central fact table (or more) connected to dimension tables, making query writing fast and straightforward, and the organization and definition of data are initially focused on a specific department or business process. Our choice of the Star Schema aligns with our objective of creating an efficient and adaptable platform to gain insights and expedite information analysis and reports within the Data Warehouse.

So, the first step was to **identify the underlying business process**. Upon reviewing the informational challenges and business inquiries outlined in sections 2.2 and 2.3, it became evident that **sales is the pivotal business process** to be represented in the Data Warehouse.

Subsequently, **the next step is to define the grain** at which to store and analyze the data. Each row in the dataset represents a **unique sale transaction**:

- Involving a **specific customer,**
- Involving a **specific product**,
- By **day** (order placed date),
- Sold by a **specific salesperson**,
- Purchased from a **specific supplier**,
- With a specified quantity sold,
- At a given unit price in euros,
- Incurring a unit cost in euros.

Meaning that, if a customer buys two different products on the same day, there will be two separate records in the fact table. If the same customer buys the same product on two different days, there will also be two different records in the fact table, and so on. By capturing unique sales transactions, our Data Warehouse provides the flexibility to aggregate data at higher levels, such as daily, weekly, or monthly summaries. This enables a comprehensive view of sales performance across various dimensions while preserving the ability to drill down to unique transactions when necessary.

In the two final phases, we will **identify the dimensions** (provide descriptive context) and **the facts** (contain numerical measures). We'll provide more details on this process in the following sections.

5.1 Dimension Tables of the Data Warehouse

Regarding the dimensions in our one-fact Star Schema, we've used the customer, date, product, sales team, and supplier dimensions due to their relevance to the business process, offering context to the sales transactions.

To ensure our Data Warehouse's reliability, performance, and usability, we are using Surrogate Keys, Business Keys, and Foreign Keys. These keys play a fundamental role in maintaining the integrity of the data, facilitating efficient data retrieval, and supporting relationships between tables.

A Business Key (BK) is a unique identifier chosen for its relevance in a specific business context. Unlike Surrogate Keys, which are more technical, Business Keys align with real-world semantics, representing natural identifiers like product codes or order IDs. We incorporate a BK into our design to ensure a more user-friendly representation of information that reflects actual business processes.

- **dim_customer**

Table 2. *Customer Dimension*

Column Name	Data Type	Description
sk_customer	INT	Surrogate Key: Used as a unique identifier for dim_customer
bk_customer	INT	Business key: Customer ID
company_name	VARCHAR (60)	Name of the customer company
region	VARCHAR(50)	Name of the region
country	VARCHAR (50)	Name of the country
city	VARCHAR (50)	Name of the city
street_number	VARCHAR(100)	House number and street name
customer_phone_number	VARCHAR(30)[1]	Phone number of the representative

Considering that each customer is associated with a single unique address, and in our context, this address precisely reflects the location where orders are successfully shipped, we've opted for not creating a 'dim_location' dimension. The rationale behind this decision lies in the fact that all our analyses related to location are inherently linked to the customer.

[1] Generating phone numbers led to unusually large country indicators, which, while not crucial for our analysis, prompted adjustments in the Data Warehouse to accommodate them.

- **dim_date**

***Table 3**. Date Dimension*

Column Name	Data Type	Description
sk_date	INT	Surrogate Key: Used as a unique identifier for dim_date
full_date	DATE	For example: 19/11/2023
year	INT	2023
semester	VARCHAR(5)	S2
quarter	VARCHAR(5)	Q4
month	INT	11
week_number	INT	46
day	INT	19
day_week	INT	7
day_week_short	VARCHAR(10)	Sun
is_weekend	VARCHAR(5)	Yes

- **dim_product**

***Table 4.** Product Dimension*

Column Name	Data Type	Description
sk_product	INT	Surrogate Key: Used as a unique identifier for dim_product
bk_product	INT	Business Key: Product ID
type	VARCHAR(50)	Product type (beverage, utensil, food)
sub_type	VARCHAR(50)	Product subtype (red wine, white wine, etc.)

wine_region	VARCHAR(30)	The wine region the product originates from
brand	VARCHAR(255)	Brand of the product
product_name	VARCHAR(255)	Name of the product
voltage	VARCHAR(15)	Alcohol content in % for the beverages – descriptive attribute (15) to capture Non Applicable
capacity	VARCHAR(10)	Capacity of the product
unit_price	DECIMAL(10,2)	Price of a product unit
unit_cost	DECIMAL(10,2)	Cost of a product unit

In this scenario, the unit price corresponds to the total retail price paid by the client, in our case, discounts are not applicable. However, in real-life scenarios where discounts are common, an additional measure for discount price could be added to the fact table.

- **dim_sales_team**

Table 5. *Sales Team Dimension*

Column Name	Data Type	Description
sk_salesperson	INT	Surrogate Key: Used as a unique identifier for dim_salesteam
bk_salesperson	INT	Business Key: Salesperson ID
first_name	VARCHAR(30)	First name of the salesperson
last_name	VARCHAR(70)	Last name of the salesperson
office	VARCHAR(50)	Office location the salesperson is assigned to
segment	VARCHAR(50)	Name of the segment
team	VARCHAR(5)	Name of the team
Sales_person_phone_number	VARCHAR(20)	Phone number of the salesperson
commission_rate	DECIMAL(10,2)	Commission rate that the salesperson gets (stored as %)

The commission rate is a fixed attribute and does not correspond to a measure since it represents a constant value for each salesperson and does not involve calculations or aggregations. Unlike measures that quantify performance or numerical results, the commission rate serves as a static property defining the rate at which salespersons earn commissions, remaining consistent across the dataset.

- **dim_supplier**

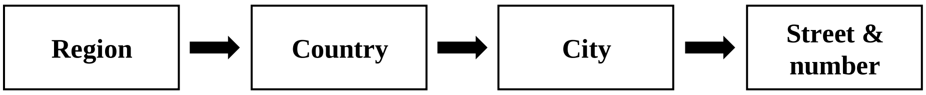
Table 6. *Supplier Dimension*

Column Name	Data Type	Description
sk_supplier	INT	Surrogate Key: Used as a unique identifier for dim_supplier
bk_supplier	INT	Business Key: Supplier ID
production_type	VARCHAR(20)	Type of Production of the Supplier (organic vs non-organic)
supplier_name	VARCHAR(60)	Name of the supplier company
supplier_phone_number	VARCHAR(20)	Phone number of the supplier contact person

5.2 Hierarchies

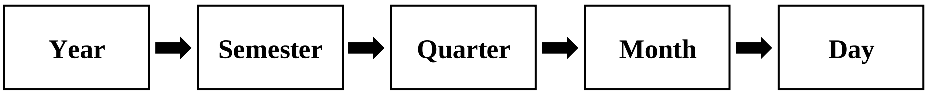
- **dim_customer**

The dim_custumer incorporates region, country, city, and street & house number (combined). Serves as a representation of client addresses for shipped products, enabling analysis from broad to specific geographical details.



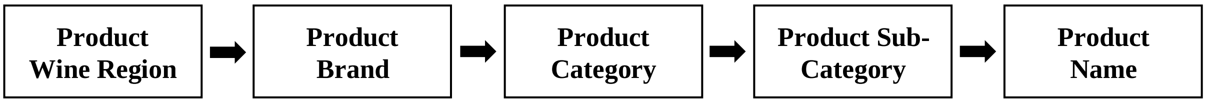
- **dim_date**

The dim_date includes year, semester, quarter, month, and day for a flexible view of time, from high-level seasonal trends to more detailed insights.



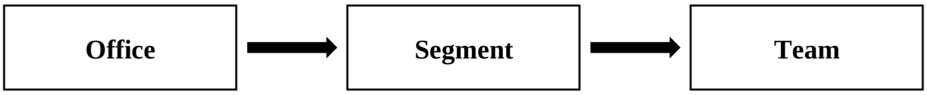
- **dim_product**

The dim_product includes the hierarchies below, establishing a structure to navigate through our product portfolio. Relevant categories also have additional unique attributes like capacity and alcohol % associated with each product.



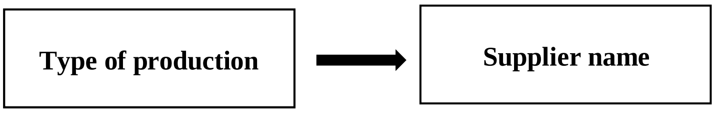
- **dim_sales_team**

This dimension is structured based on office location, segment, and teams. Salespersons are categorized by whether they work in Porto or Lisbon, and within each office, they are further grouped by segment (based on market importance). Additionally, each segment includes various teams, responsible for specific clients.



- **dim_supplier**

In dim_supplier, we have a simple hierarchy where we can categorize each supplier into organic and non-organic production types.



5.3 Fact Table of the Data Warehouse

As we identified sales as the fact in our Data Warehouse design, fact_sales represents the central table in our Star Schema. The chosen measures in the fact table — quantity_sold; unit_price; unit_cost; order_value; and total_profit — directly align with Nova Wines S.A.'s goals as defined in 2.2. They enable a concise evaluation of sales team performance, accurate forecasting based on product trends, and targeted analysis of key markets and clients.

Table 7. *Sales Fact Table*

Column Name	Data Type	Description
fk_product	INT	Foreign key of dim_product
fk_customer	INT	Foreign key of dim_customer
fk_salesperson	INT	Foreign key of dim_sales_person
fk_supplier	INT	Foreign key of dim_supplier
fk_date	INT	Foreign key of dim_date
quantity_sold	INT	Number of units of the product sold
unit_price	DECIMAL(10,2)	Price paid by the Customer for each product unit
unit_cost	DECIMAL(10,2)	How much each product unit costs to the company
order_value	DECIMAL(10,2)	quantity_sold x unit_price: total paid by the customer
total_profit	DECIMAL(10,2)	order_value – (quantity_sold x unit_cost): financial gain or loss associated with a specific order

5.4 Star Schema Model

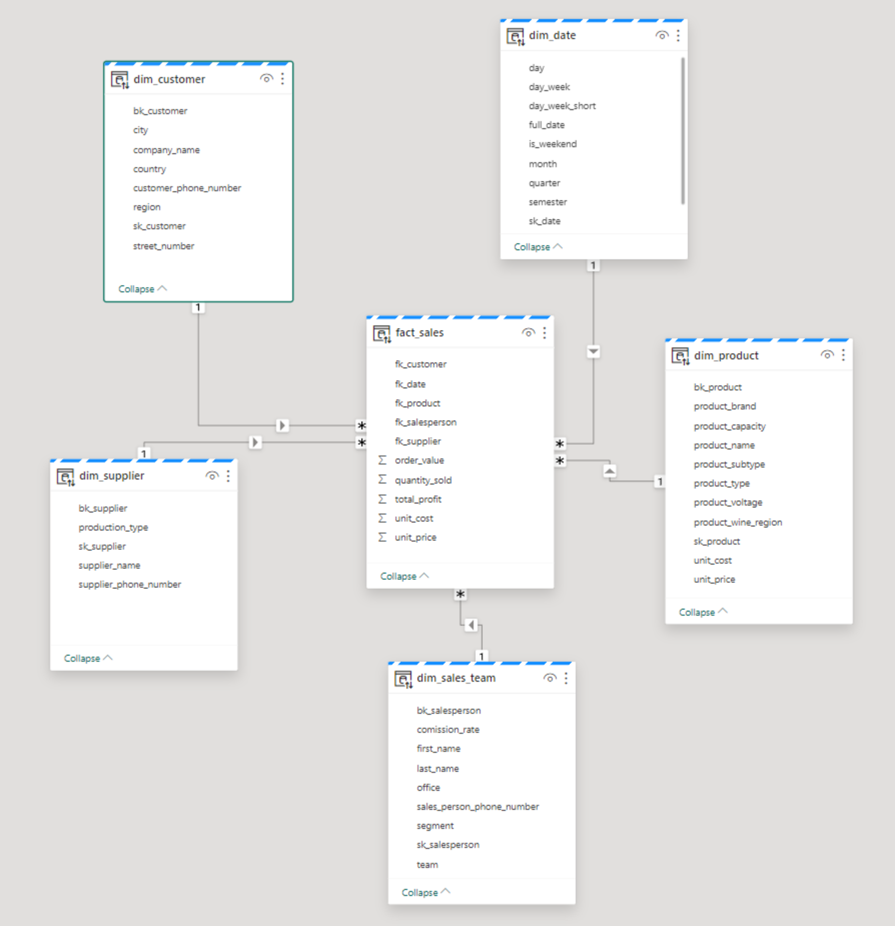


Figure 6. *Star Schema Model*

The outcome of our Data Warehouse design is the Star Schema above. Every table includes a surrogate key (SK) and forms active one-to-many relationships with the foreign key (FK) in the fact table.

Some difficulties arose in this phase. Some connections in our original design idea would lead to a Snowflake Schema, and some changes in perspective had to be made to maintain the Star Schema. Above all, we always had to keep in mind the readability of the project, considering that we are serving a company that has never had many advanced data storage solutions. Names had to be intuitive, and connections had to be easily understandable. We always have to consider who will be reading our project on the other end, aiming to make the analysis and conclusions as simple as possible.

6 ETL Process

After establishing the Staging Area and designing the Data Warehouse to effectively address our business inquiries, the subsequent phase involves initiating the ETL (Extract, Transform, Load) process. This multi-step procedure encompasses selecting data sources for extraction, applying transformations to the data, and loading the converted (transformed) data into the Data Warehouse.

For the ETL process, we used the following distinct sources:

- 5 CSV files, each corresponding to a dimension and fact table (we opted to convert each sheet from the original Excel file into a separate CSV file).
- 2 CSV support files, one containing date-related information and the other containing order-date information.

To streamline the process, we created a Source Lakehouse named **"LH_NOVAWINES_SOURCES".** We uploaded our data in the form of separate CSV files for each dimension and fact table.

Our base ETL process is organized into three fundamental pipelines:

- **PL_NOVAWINES_LOAD_STG Pipeline:** Extracts data from source files, performs necessary transformations, and Loads data into the Staging Area.
- **PL_NOVAWINES_VALIDATE_STG Pipeline**: Contains the structure enabling Data Quality checks, and Stores results in our "log_quality_checks" table in the Staging Area.
- **PL_NOVAWINES_LOAD_DW Pipeline**: Loads data from the Staging Area into the final Data Warehouse and creates surrogate keys for the dimension tables in the fact table.

Additionally, we developed conditional loading using two additional pipelines:

- **PL_NOVAWINES_CHECK_VALIDATE Pipeline**: Checks if any results from the second pipeline return FAIL.
- **PL_NOVAWINES_COND_LOAD_DW Pipeline**: Performs full loading if the pipeline above is successful.

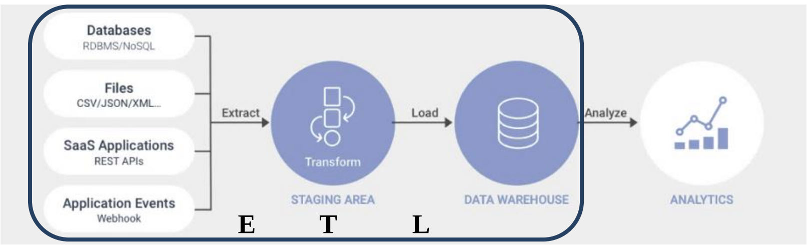


Figure 7. *Architecture of our solution, we will not perform the analysis in this project.*[2]

[2] Source: https://www.analyticsvidhya.com/blog/2021/11/the-ultimate-guide-to-setting-up-an-etl-extract-transform-and-load-process-pipeline/

6.1 Loading the Data into the Staging Area

PL_NOVAWINES_LOAD_STG Pipeline

As mentioned before, this first pipeline is used to load the data from the Source Lakehouse into the Staging Area. Within the pipeline, we are using the following activities:

1. **Multiple SQL script activities** to clear **stg_fact_sales** and all **stg dimension tables:**

Figure 8. *Script Activity*

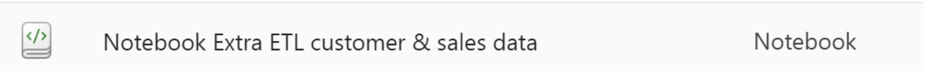
It is crucial to go through this phase to avoid having duplicate data, especially since we are using the "append" option to load data into specific tables and this is a complete (full load) pipeline. During this phase, we execute the pipeline a fair number of times, and we cannot afford to have any duplicate entries, so we use this to ensure data integrity and prevent duplicates in subsequent loads. We decided to use a separate SQL script activity for each of the dimension tables, as it is the safer and easier option.

2. **Multiple Wait Activities, in between the data clearing processes and data flows:**

Figure 9*. Wait Activity*

These serve as 'dummy' activities, serving the purpose of synchronizing the preceding data-clearing processes and data flows. The inclusion of these 'dummy' activities is essential for ensuring proper coordination in the workflows, allowing time for the completion of each step in the pipeline.

3. **A Notebook Activity to perform complex ETL processes on customer and sales data:**



This part is explained in detail in the section **Extra (original) ETL Work.** We are using this activity to resolve any issues we have with duplicate customers and customer IDs. We then transform the sales data accordingly.

We start by extracting the CSV files “customers_novawines_oltp.csv” and “orders_novawines_oltp.csv”. Finally, we load the two resulting data frames into the delta tables of the Lakehouse, which are then ready for the second round of ETL.

4. A Dataflow to load the stg_dim_customer table:

DF load stg_dim_customer
Dataflow Gen2

After the first ETL using the Notebook Activity, this Dataflow ensures that the processing to the delta was done correctly and makes a few adjustments, such as dropping columns and replacing text in the phone data to ensure a consistent format. We chose to use a Dataflow for these modifications as it allowed us to visualize our table while we worked on it. This was extremely helpful in controlling the outcome.

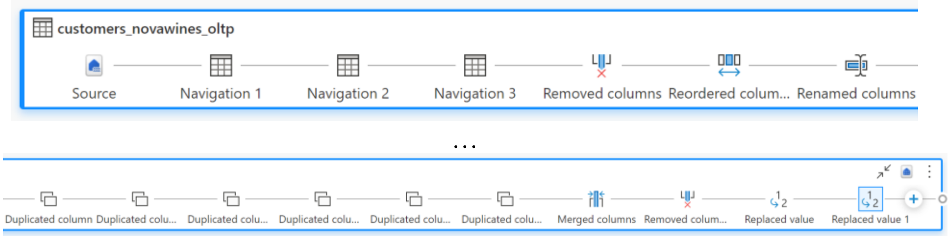
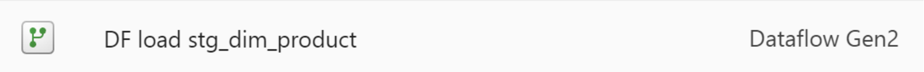


Figure 10. *DF load stg_dim_customer*

5. A Dataflow to load the stg_dim_product:


DF load stg_dim_product
Dataflow Gen2

In this Dataflow, some changes were implemented. Firstly, we established the desired presentation format for the alcohol voltage percentage and the capacity of each product. Since these values are purely descriptive, we opted to convert them into text format. Additionally, we appended a suffix to signify the respective units (% for voltage and _ml or _g for the capacity column). Furthermore, we merged the two columns related to capacity—one containing only gram information and the other exclusively milliliter information—into a single capacity column.

In the process of encoding, as explained in detail in the section ***Special Characters Visualization*** below, we encountered an issue while publishing the Dataflow. The problem arose due to the presence of a white space in one of the column names. In order to resolve the issue, we tried to load the CSV file into a table in the Lakehouse, an error occurred which indicated that the table containing the space was the "Alcohol Voltage" table and we proceeded to edit this in the Dataflow.

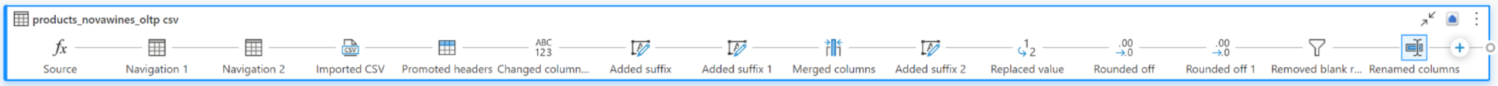
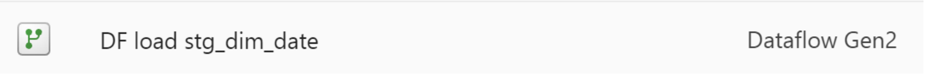


Figure 11. *DF load stg_dim_product*

6. A Dataflow to load the stg_dim_date:


DF load stg_dim_date Dataflow Gen2

The date dimension is crucial for time-driven analysis and data normalization. Due to incorrect formatting in the full date column, a decision was made to rebuild it. The transformation involved combining year, month, and day, merging them, and converting the format to a proper date. PowerQuery's 'Date and time column' function was essential for obtaining the day of the week, the week number, and the quarter. The subsequent steps included adjusting day of the week values, appending 'Q' to quarter values, assigning quarters to semesters using 'if' statements, and replacing binary values in is_weekend with 'Yes' or 'No'.

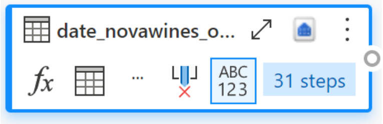
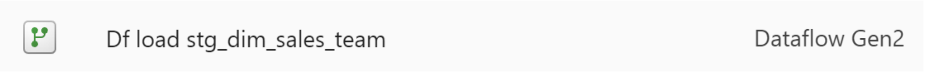


Figure 12. *DF load stg_dim_date*

7. A Dataflow to load the stg_dim_sales_team:


Df load stg_dim_sales_team Dataflow Gen2

Despite the organization of the data intended for loading into this dimension, a challenge arose during this phase. The Dataflow for this dimension, along with the customer and supplier dataflows, contained phone numbers. This became problematic as these dataflows were impeding the refresh process. The issue stemmed from our initial use of the INT (integer) data type for the phone numbers columns. Subsequently, we realized that this column did not represent a numerical value but rather a descriptive property.

To address this, we made a necessary adjustment to our schema by changing the data type of the phone numbers columns to VARCHAR. This modification allows for the representation of non-numeric, descriptive data and ensures flexibility to accommodate international indicators and various formats, resolving the obstruction in the data refresh process.

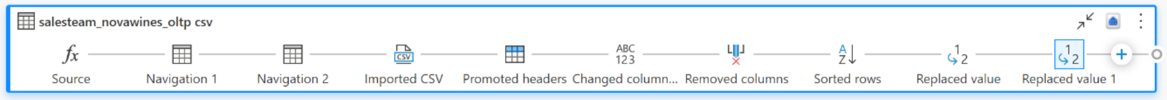
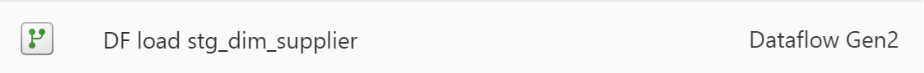


Figure 13. *DF load stg_dim_sales_team*

8. A Dataflow to load the stg_dim_supplier:



DF load stg_dim_supplier Dataflow Gen2

This dimension was also straightforward to handle. We utilized a dataflow to visualize the data and dropped a column that didn't match our schema.
We face the same issue with the phone number type mentioned earlier.

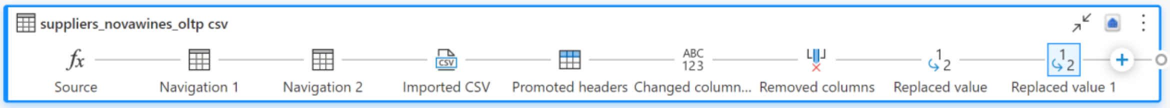
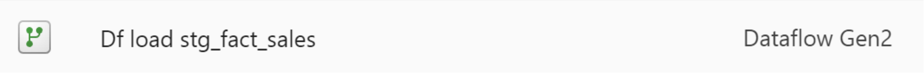


Figure 14. *DF load stg_dim_supplier*

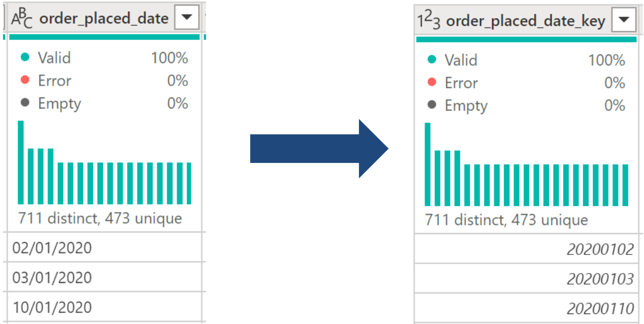
9. A Dataflow to load the stg_fact_sales:



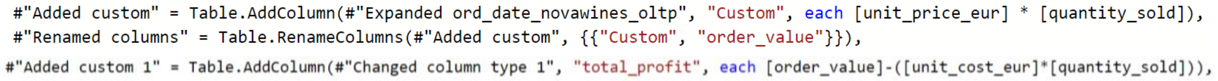
Df load stg_fact_sales Dataflow Gen2

As mentioned earlier, the customer data (specifically the customer ID which will serve as the foreign key in the final design) has already undergone some preparation in the notebook. However, further transformations are required in this Dataflow to achieve the desired table format for this phase.

To begin with, we merge the "ord_date_novawines_oltp" support table, which has the date key (order placed date) for each order ID, with our "orders_novawines_oltp" table, which contains the remaining sales order information. We merge these two sources using the "order_placed_date" column:



Next, we incorporate the necessary custom columns, namely, 'order_value' and 'total_profit,' by applying the following calculation:



```
#"Added custom" = Table.AddColumn(#"Expanded ord_date_novawines_oltp", "Custom", each [unit_price_eur] * [quantity_sold]),
#"Renamed columns" = Table.RenameColumns(#"Added custom", {{"Custom", "order_value"}}),
#"Added custom 1" = Table.AddColumn(#"Changed column type 1", "total_profit", each [order_value]-([unit_cost_eur]*[quantity_sold])),
```

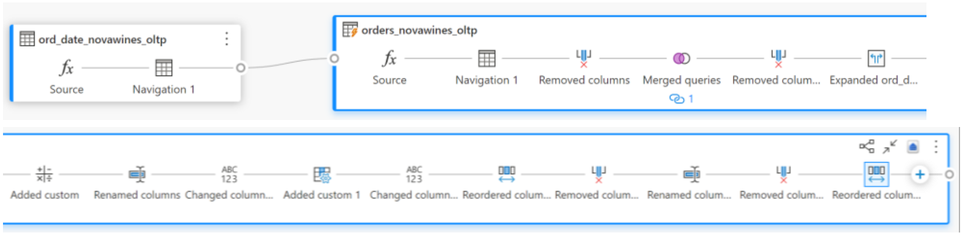


Figure 15. *DF load stg_fact_sales*

10. Final Design:

Although it is possible to load all the tables in the Staging Area simultaneously, we have decided to load the fact table only after completing the loading of all dimension tables. This approach aims to simplify the data loading process, and it helps to organize and streamline the pipeline's efficiency by allowing parallel processing of the dimension tables. Once all dimension tables have been loaded, we then proceed to the loading of the fact table.

In this initial phase of the project, we deliberately refrain from using Surrogate Keys. This decision allows us to focus on loading individual tables without the need for surrogate keys or complex relationships at the staging area level. The simplicity gained from this approach not only contributes to better performance during the initial data loading phase but also ensures that we have the necessary setup for performing data quality checks in the next phase of the project.

We have connected all the activities mentioned above into a continuous running pipeline, which will proceed only when the previous activity has been completed successfully.

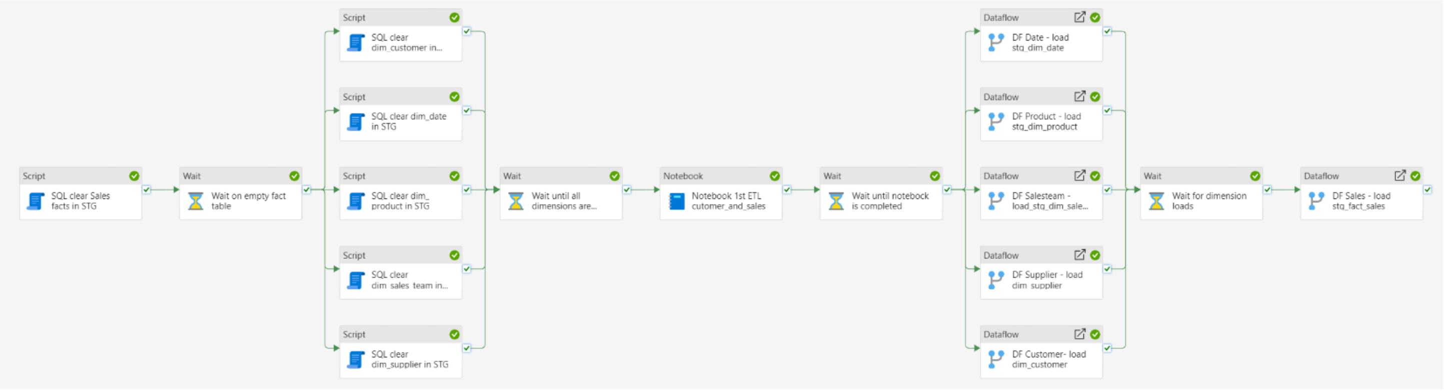

Script
SQL clear Sales facts in STG
Wait
Wait on empty fact table
Script
SQL clear dim_customer in...
Script
SQL clear dim_date in STG
Script
SQL clear dim_ product in STG
Script
SQL clear dim_sales_team in...
Script
SQL clear dim_supplier in STG
Wait
Wait until all dimensions are...
Notebook
Notebook 1st ETL cutomer_and_sales
Wait
Wait until notebock is completed
Dataflow
DF Date - load stg_dim_date
Dataflow
DF Product - load stg_dim_product
Dataflow
DF Salesteam - load_stg_dim_sale...
Dataflow
DF Supplier - load dim_supplier
Dataflow
DF Customer- load dim_customer
Wait
Wait for dimension loads
Dataflow
DF Sales - load stg_fact_sales

Figure 16*. PL_NOVAWINES_LOAD_STG pipeline design*

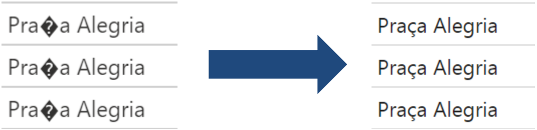
6.2 Loading the Data into the Staging Area: Special considerations

- **Special characters visualization:**

In handling the files containing supplier, sales team, and product data, we encountered an issue with displaying special characters. Initially, we employed preload tables of the Lakehouse and utilized them in the first dataflow (ETL + load to the Staging Area). However, we later identified the need for an adjustment in our approach.

We attempted to use CSV files and experimented with changing the encoding (including trying encoding 1252, and 1200), but this approach proved unsuccessful. Upon further investigation, we hypothesized that the problem might be associated with how we saved the CSV files, specifically not using encoding utf-8.

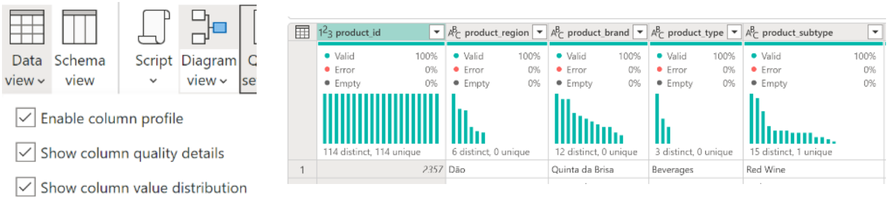
To address this, we saved the files from the original Excel sheets in CSV format with utf-8 encoding. After reloading these files into the Lakehouse and incorporating them into the Dataflows, we observed that the issue with special characters was successfully resolved:



Before	After
Pra�a Alegria	Praça Alegria
Pra�a Alegria	Praça Alegria
Pra�a Alegria	Praça Alegria

The data from clients and sales CSV files undergoes initial processing in a notebook, where we explicitly specify the UTF-8 encoding when opening them. It's worth noting that for date data, we do not utilize special characters. This approach ensures that the files are treated with the appropriate encoding, allowing for the correct representation of characters in the data.

- **Column profile view:**



☑ Enable column profile

☑ Show column quality details

☑ Show column value distribution

	product_id	product_region	product_brand	product_type	product_subtype
	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%
	114 distinct, 114 unique	6 distinct, 0 unique	12 distinct, 0 unique	3 distinct, 0 unique	15 distinct, 1 unique
1	2357	Dão	Quinta da Brisa	Beverages	Red Wine

Figure 17. Column profile view

While not inherently part of the ETL process itself, enabling the column profile view proved to be very useful for our understanding and readability of the data. This played a crucial role in identifying anomalies such as non-unique/ missing values where they weren't expected. A notable example was the discovery of repeated IDs in the customers' data, prompting the development of a dedicated notebook with specialized procedures to address and resolve this particular issue.

6.3 Data Quality checks

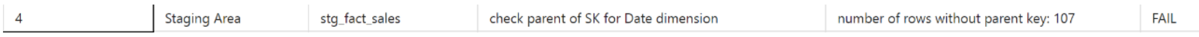
PL_NOVAWINES_VALIDATE_STG Pipeline

For this part, we used a separate pipeline to perform the data quality checks employing 4 different rules and the result was stored in the log_quality_checks as FAIL or OK:

Table 8. *Data Quality Check Rules*

Rule	Tables tested	Description
Rule 1	stg_dim_customer	We maintain a precautionary rule despite generating our index in the Notebook to anticipate potential issues like inconsistencies or duplicate loadings, especially during clear script failures. This extra layer of precaution safeguards data integrity and prevents complications in subsequent processes.
	stg_dim_product	Using a script activity this rule Checks the Integrity of Business Key (in the role of PK), meaning the number of rows with repeated BK using a count for each stg_dim table.
	stg_dim_sales_team	
	stg_dim_supplier	
	stg_dim_date	We depend on an external data support file, not generated in Microsoft Fabric. Rules 1 and 2 are applied to maintain consistency in the date dimension.
Rule 2	stg_dim_customer	Using an SQL script, check if there are any duplicated rows by verifying the uniqueness of attributes across the entire dimension.
	stg_dim_product	
	stg_dim_sales_team	
	stg_dim_supplier	
	stg_dim_date	
Rule 3	stg_fact_sales	Using a SQL script, check for a unique combination of Foreign Keys on the sales fact table to ensure it is a proper composite PK.
Rule 4	stg_dim_customer + stg_fact_sales	Using a Dataflow, this rule checks if each Fact table Foreign Key is associated with a unique Business Key in the parent Dimension, without any missing dimension values (no NULL BKs coming from the Dimension table). For Rule 4, we used Dataflows since it requires joining two tables. The visual aid during setup was helpful, especially for the fk_date.
	stg_dim_product + stg_fact_sales	
	stg_dim_sales_team + stg_fact_sales	
	stg_dim_supplier + stg_fact_sales	
	stg_dim_date + stg_fact_sales	

- **Special considerations for rule 4 applied stg_dim_date:** When conducting a left join in this Dataflow, only 7876 out of the 7983 values from the date key column align with the sales fact date requirements, causing the rule to fail. To address this, we extended the temporal range of our date file.



4	Staging Area	stg_fact_sales	check parent of SK for Date dimension	number of rows without parent key: 107	FAIL

Figure 18. *Rule 4 Fail for stg_dim_date*

- **Other Fails:** Occasionally, the rules return a FAIL status because we independently adjusted and published the loading Dataflows, which leads them to append the data as defined. Initially, we were concerned that the clear scripts for the loading staging were not working correctly. We reached this conclusion, but as a precaution, we decided to implement separate clear scripts for each dimension and fact table in the first pipeline, instead of clearing all dimensions simultaneously, every time it's needed.

- **Final Design:**

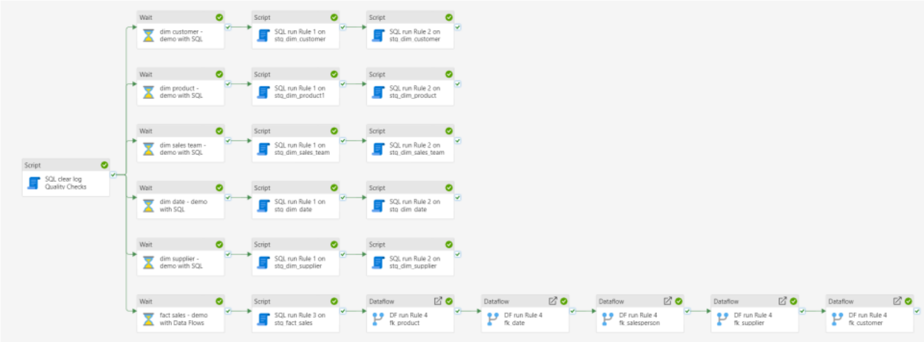
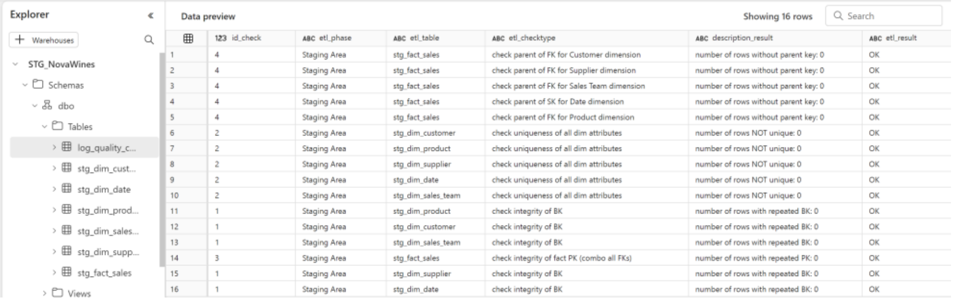

Script
SQL clear log Quality Checks
Wait
dim customer - demo with SQL
Script
SQL run Rule 1 on stg_dim_customer
Script
SQL run Rule 2 on stg_dim_customer
Wait
dim product - demo with SQL
Script
SQL run Rule 1 on stg_dim_product1
Script
SQL run Rule 2 on stg_dim_product
Wait
dim sales team - demo with SQL
Script
SQL run Rule 1 on stg_dim_sales_team
Script
SQL run Rule 2 on stg_dim_sales_team
Wait
dim date - demo with SQL
Script
SQL run Rule 1 on stg_dim_date
Script
SQL run Rule 2 on stg_dim_date
Wait
dim supplier - demo with SQL
Script
SQL run Rule 1 on stg_dim_supplier
Script
SQL run Rule 2 on stg_dim_supplier
Wait
fact sales - demo with Data Flows
Script
SQL run Rule 3 on stg_fact_sales
Dataflow
DF run Rule 4 fk_product
Dataflow
DF run Rule 4 fk_date
Dataflow
DF run Rule 4 fk_salesperson
Dataflow
DF run Rule 4 fk_supplier
Dataflow
DF run Rule 4 fk_customer

Figure 19*. PL_NOVAWINES_VALIDATE_STG Pipeline*

- **Final Results:**



	id_check	etl_phase	etl_table	etl_checktype	description_result	etl_result
1	4	Staging Area	stg_fact_sales	check parent of FK for Customer dimension	number of rows without parent key: 0	OK
2	4	Staging Area	stg_fact_sales	check parent of FK for Supplier dimension	number of rows without parent key: 0	OK
3	4	Staging Area	stg_fact_sales	check parent of FK for Sales Team dimension	number of rows without parent key: 0	OK
4	4	Staging Area	stg_fact_sales	check parent of SK for Date dimension	number of rows without parent key: 0	OK
5	4	Staging Area	stg_fact_sales	check parent of FK for Product dimension	number of rows without parent key: 0	OK
6	2	Staging Area	stg_dim_customer	check uniqueness of all dim attributes	number of rows NOT unique: 0	OK
7	2	Staging Area	stg_dim_product	check uniqueness of all dim attributes	number of rows NOT unique: 0	OK
8	2	Staging Area	stg_dim_supplier	check uniqueness of all dim attributes	number of rows NOT unique: 0	OK
9	2	Staging Area	stg_dim_date	check uniqueness of all dim attributes	number of rows NOT unique: 0	OK
10	2	Staging Area	stg_dim_sales_team	check uniqueness of all dim attributes	number of rows NOT unique: 0	OK
11	1	Staging Area	stg_dim_product	check integrity of BK	number of rows with repeated BK: 0	OK
12	1	Staging Area	stg_dim_customer	check integrity of BK	number of rows with repeated BK: 0	OK
13	1	Staging Area	stg_dim_sales_team	check integrity of BK	number of rows with repeated BK: 0	OK
14	3	Staging Area	stg_fact_sales	check integrity of fact PK (combo all FKs)	number of rows with repeated PK: 0	OK
15	1	Staging Area	stg_dim_supplier	check integrity of BK	number of rows with repeated BK: 0	OK
16	1	Staging Area	stg_dim_date	check integrity of BK	number of rows with repeated BK: 0	OK

Figure 20. *log_quality_check table results*

This validation is used as a condition to load our Data Warehouse, if any of the rule application result return "FAIL", the full loading will not be completed.

6.4 Loading the Data Warehouse

PL_NOVAWINES_LOAD_DW Pipeline

The third data pipeline serves the purpose of loading the data from the Staging Area into the final Data Warehouse.

Like in the first pipeline, we start off by using multiple SQL script activities, to clear our fact_sales in the first step and all our dimension tables in the second step. Again, we do this to prevent duplicate data in subsequent loads in the future.

After all the warehouse tables are cleared, we start to load the dimension tables. For this, we decided to create a copy data activity for each dimension load. We decided to use copy data activities, as they are optimized to move data without extensive transformations, which we already did in the first pipeline.

Figure 21. *Copy Data activity.*

- **Copy Data for dim_date:** The copy data activity for the dim_date is simply transferring the cleaned and transformed data from the Staging Area table into the designated Data Warehouse table. Due to the previous transformation, the columns from the source table exactly match the columns of the destination table, which simplified the column mapping.

- **Copy Data for dim_customer:** In addition to the transfer of clean and transformed data, this copy data activity serves another purpose: to create the surrogate key of the table. We do this by using SQL code when retrieving the data from the source stg table. The code we used retrieves all columns from the stg_dim_customer and for each row, it generates a unique sequential surrogate key (sk_customer) based on the order of the bk_customer column in ascending order. The sk_customer column is now part of the output and mapped along all other columns of the source table into the designated columns of the destination table.

- We perform **the same procedure for the dim_product, dim_sales_team and dim_supplier**. Initially, challenges emerged during the copy data operation for these dimensions due to an invalid column mapping in the 'type' column. Fortunately, we swiftly resolved this by implementing the same necessary modifications we implement during the Staging Area loading, adjusting data types for descriptive attributes (ex. Phone numbers, alcohol %) and some Varchar() capacities.

- **Copy Data for fact_sales:** In addition to transferring the fact_sales data from the staging table to the final warehouse table, the copy data activity is used to substitute the **Old Foreign Keys** of the dimension tables with the **New Foreign Keys**, which we created in the previous copy data activities of the pipeline. The SQL code we used retrieves columns from both the staging fact table and the dimension tables by joining them based on their Foreign Key (FK) and Business Key (BK) relationships. This creates a cohesive dataset, which is ready for insertion into the final fact table of the data warehouse.

- **Final Design:**

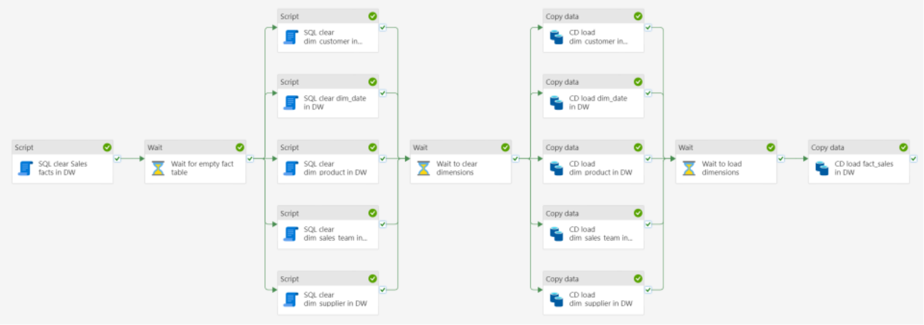

Script
SQL clear Sales facts in DW
Wait
Wait for empty fact table
Script
SQL clear dim_customer in...
Script
SQL clear dim_date in DW
Script
SQL clear dim_product in DW
Script
SQL clear dim_sales_team in...
Script
SQL clear dim_supplier in DW
Wait
Wait to clear dimensions
Copy data
CD load dim_customer in...
Copy data
CD load dim_date in DW
Copy data
CD load dim_product in DW
Copy data
CD load dim_sales_team in...
Copy data
CD load dim_supplier in DW
Wait
Wait to load dimensions
Copy data
CD load fact_sales in DW

***Figure 22**. PL_NOVAWINES_LOAD_DW Pipeline Design*

7 Extra (original) ETL Work

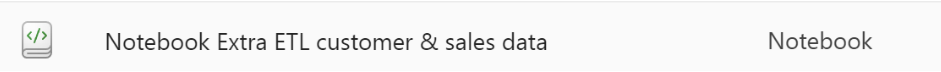
7.1 Use of Notebook for complex ETL work

As mentioned earlier, we encountered some issues with the customer data. In this section, we will elaborate on how we utilized the Notebook Activity to address these problems. We have also detailed the entire process in the markdowns of the notebook to make it easier to comprehend the decisions taken while reading the code. To summarize, we start by conducting a pre-exploratory analysis, also in Excel, using a pivot to understand the problems we are facing and reach three cases for detailed analysis:

Table 9*. Case problems for Customer data*

Cases	Description	Illustration	Approach
1	Customers with Various Locations in the US.	The company uses client IDs to discern the specific location of an order. However, due to clients having multiple locations, there's a need to assign distinct IDs to orders from each location for tracking purposes	**No immediate Action.** **Special Attention:** These clients require special attention during client performance analysis to ensure sales are not counted separately. **Consideration for Future:** For improved organization, the company may contemplate creating a separate dimension for location in the future.
2	Same Customer, Different IDs	The company faces instances where a single customer is associated with different IDs, due to differences in phone numbers or minor variations in addresses like house numbers.	**Addressed in the Notebook:** The approach entails retaining the latest ID as it reflects the most recent information. Sales data from older IDs is consolidated and transferred to the most recent one.
3	Same ID, Different Customer	The system erroneously assigns identical IDs to two distinct clients, leading to confusion in associating sales with each respective client.	**Addressed in the Notebook:** The approach involves an analysis of the impact caused by these two clients. Subsequently, sales associated with these clients are removed to maintain data consistency. **Add New Index:** To rectify the situation, establish an association with the previous ID in the sales data to ensure accurate tracking and reporting.

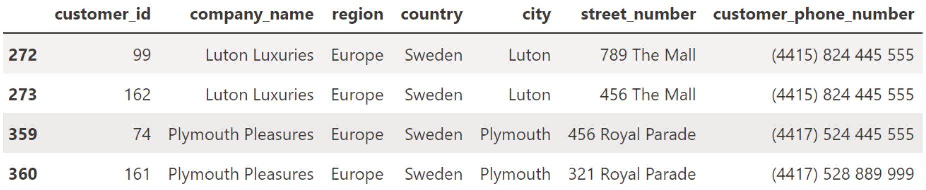
Now, exploring the Notebook deeper:



Notebook Extra ETL customer & sales data Notebook

- **Case 2: Same Customer, Different IDs**

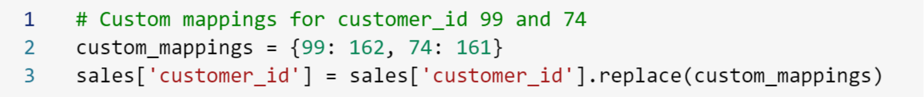
Following the approach that the same client can indeed have different IDs in our scenario if they have locations in different cities or countries, we begin by grouping by ['company_name', 'country', 'city']:



	customer_id	company_name	region	country	city	street_number	customer_phone_number
272	99	Luton Luxuries	Europe	Sweden	Luton	789 The Mall	(4415) 824 445 555
273	162	Luton Luxuries	Europe	Sweden	Luton	456 The Mall	(4415) 824 445 555
359	74	Plymouth Pleasures	Europe	Sweden	Plymouth	456 Royal Parade	(4417) 524 445 555
360	161	Plymouth Pleasures	Europe	Sweden	Plymouth	321 Royal Parade	(4417) 528 889 999

***Figure 23**. Same Customer with Different IDs*

We verify that these differences seem like incremental changes for the same client (we are not able to perform a double check, but there's a possibility of doing that in a 'real' company). Therefore, we only retain the IDs 162 and 162, assuming that the IDs were attributed sequentially, and the higher ID corresponds to the most recent entry. Subsequently, we substitute the sales in the sales order data in case there are some sales associated with IDs 99 and 74:

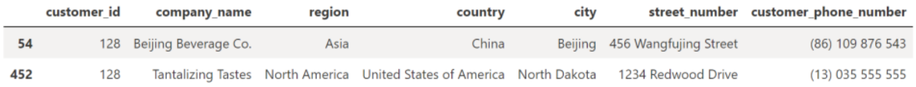


```
# Custom mappings for customer_id 99 and 74
custom_mappings = {99: 162, 74: 161}
sales['customer_id'] = sales['customer_id'].replace(custom_mappings)
```

***Figure 24.** Custom mapping for Customers with multiple IDs*

- **Case 3: Same ID, Different Customer**

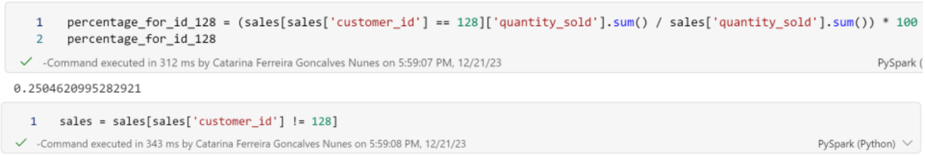
The next step involves some analysis and data filtering. Since we have these two customers with the ID 128:



	customer_id	company_name	region	country	city	street_number	customer_phone_number
54	128	Beijing Beverage Co.	Asia	China	Beijing	456 Wangfujing Street	(86) 109 876 543
452	128	Tantalizing Tastes	North America	United States of America	North Dakota	1234 Redwood Drive	(13) 035 555 555

***Figure 25.** Same ID for different customers*

Having no other way to associate the sales with the clients (as we only have the ID), we decided to analyze the impact of the sales with ID 128, and it corresponds to 0.25% of the total quantity sold for our period. To ensure the consistency of our analysis in the dataset and considering that this number is not meaningful, we decided to remove this ID from the sales order table:



```
percentage_for_id_128 = (sales[sales['customer_id'] == 128]['quantity_sold'].sum() / sales['quantity_sold'].sum()) * 100
percentage_for_id_128
```

✓ -Command executed in 312 ms by Catarina Ferreira Goncalves Nunes on 5:59:07 PM, 12/21/23 PySpark (

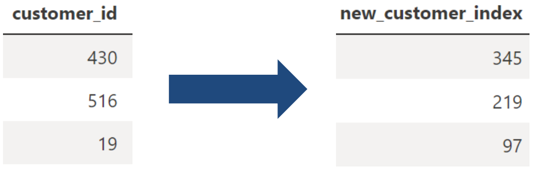
```
0.2504620995282921
```

```
sales = sales[sales['customer_id'] != 128]
```

✓ -Command executed in 343 ms by Catarina Ferreira Goncalves Nunes on 5:59:08 PM, 12/21/23 PySpark (Python)

Figure 26*. Approach for case 3*

After the removal of ID 128 from the sales order table, the next step involves restructuring the customer data. To accomplish this, we create a new index for the customer data. This new index serves as a mechanism for general mapping, allowing us to associate the old customer ID with the most recent one. In essence, this process ensures that the customer data remains coherent and aligned with the changes made in the sales order table.



customer_id	new_customer_index
430	345
516	219
19	97

In summary, the methodologies employed were driven by limitations in available data for error mapping. To ensure data integrity, strategic compromises were considered and implemented throughout the analytical process.

7.2 Extra Pipeline to make a conditional loading based on the Validate Pipeline result with email notification.

To ensure a seamless progression through the entire loading process, from the first pipeline to the last, and to prevent potential errors, we implemented a conditional loading process.

Building upon the foundation of the PL_NOVAWINES_VALIDATE_STG Pipeline, we introduced an additional pipeline named **PL_NOVAWINES_CHECK-_VALIDATE**. This new pipeline is designed to tally the number of 'FAIL' values inserted into the 'etl_result' column of the log_quality_checks table in the Staging Area. If this count is greater than zero, the pipeline is marked as unsuccessful.

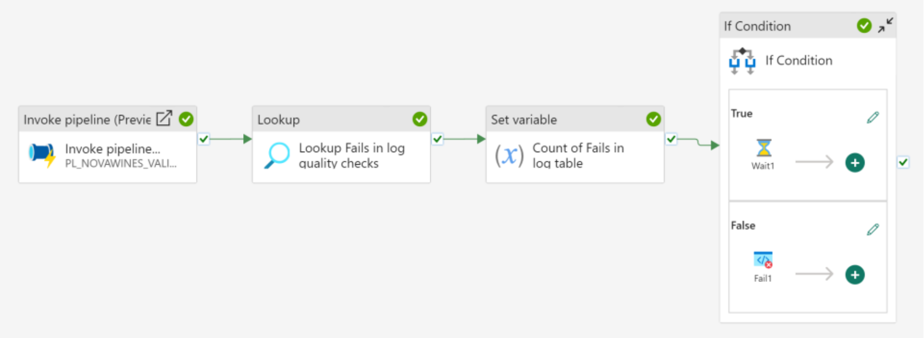


Figure 27. *PL_NOVAWINES_CHECK_VALIDATE Pipeline*

To complete the full pipeline sequence, the final component, **PL_NOVAWINES_COND_LOAD_DW** integrates the following sub-pipelines:

- PL_NOVAWINES_LOAD_STG Pipeline;
- PL_NOVAWINES_CHECK_VALIDATE Pipeline (emphasis on not including the 2nd pipeline as it provides the base condition);
- PL_NOVAWINES_LOAD_DW Pipeline.

Additionally, an email notification activity is incorporated. Upon a successful pipeline run and data loading, an email notifies us of the operation's success. If, for any reason, the load fails, we receive an email indicating a failure in the validation check of the data warehouse loading. This inclusion facilitates monitoring the pipeline's status, particularly when scheduling loadings. This proactive measure enables swift reaction and issue resolution in the event of any errors during the loading process.

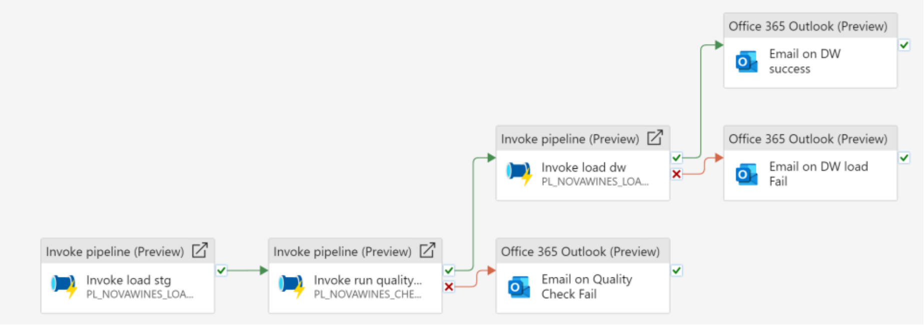

Office 365 Outlook (Preview)
Email on DW success
Invoke pipeline (Preview)
Invoke load dw
PL_NOVAWINES_LOA...
Office 365 Outlook (Preview)
Email on DW load Fail
Invoke pipeline (Preview)
Invoke load stg
PL_NOVAWINES_LOA...
Invoke pipeline (Preview)
Invoke run quality...
PL_NOVAWINES_CHE...
Office 365 Outlook (Preview)
Email on Quality Check Fail

Figure 28. *PL_NOVAWINES_COND_LOAD_DW Pipeline*

8 Critical Review

In this project, despite our results appearing quite straightforward, we chose an unconventional path. By creating our own database, we ended up creating a *'double-edged sword'* situation. On one hand, we are always aware that real company data tends to hold much greater complexity than what we generate; in many situations, the process may not be as straightforward. More ETL treatments will be necessary, and as the complexity grows, they will demand more critical thinking and agility in reading and analysis.

However, we introduced a level of complexity by building a business from scratch. Developing a solution tailored to a company requires an understanding of how that company operates and the specific data we need. This necessitates not only technical knowledge but also a deeper comprehension of the business itself. It underscores the importance of adaptability and a nuanced approach in creating a solution that truly aligns with real-world scenarios.

Furthermore, when contemplating solutions, we must have problems. In our case, in addition to solutions, we also had to identify problems that were aligned with reality and could be applied in a real-world scenario.

We've also faced issues maintaining data consistency, especially when using AI and code to generate data. Notably, we encountered challenges with phone numbers, as several clients shared the same number due to a generating error. Despite these issues, we've diligently examined the data, using mistakes like repeated customer IDs and names to practice different cleaning techniques.

Focusing on a technical perspective, certain errors surfaced during the workflow. While these discrepancies are addressed in this report, we can emphasize that the primary challenges we encountered were tied to highly specific, nuanced issues such as format variations and script inconsistencies, which can be tricky to spot. Additionally, it is worth noting the learning curve associated with Microsoft Fabric. Despite being a relatively new tool in the market with limited available resources, a substantial effort was invested to delve deep into the best practices for utilizing this tool.

In conclusion, it's important to note that synthetic data may not perfectly mirror real-world situations. Nonetheless, our commitment to understanding data generation methods, determining necessary information, and refining data cleaning processes remains strong. This approach significantly contributes to our learning journey, providing a comprehensive understanding of the entire process.

9 Lessons Learned

9.1 Star-Schema vs. Snowflake-Schema

Our initial approaches in designing the warehouse often led us to a Snowflake Schema due to the connections or paths we wanted to follow. The major difference lies in the level of normalization, where the Star Schema involves denormalization for simplicity and performance, while the Snowflake Schema is more normalized with additional subtables. Understanding the differences between the two schemas was instrumental in achieving our final desirable result: a Star Schema.

9.2 dim_order removal

Originally, we introduced a 'dim order' dimension in our Data Warehouse design intending to align it with common industry practices. Despite each order ID being inherently unique, this dimension was conceived to mitigate potential issues related to order ID repetition and to ensure adaptability to the dynamic needs of companies over time.

However, after receiving feedback, we decided to streamline our approach. Given that each column in the order ID corresponds to our grain in our simplified universe, we have chosen to represent our Fact Table directly with each of these individual transactions. This adjustment allows for a more straightforward and efficient representation of our data, aligning with the nature of our simplified business model.

9.3 Descriptive attributes

Descriptive attributes refer to non-numeric values, such as phone numbers or descriptive measures.

9.4 Learning journey with Microsoft Fabric

As a side note, we would like to emphasize that Microsoft Fabric is an exceedingly user-friendly platform, particularly when working collaboratively. Nevertheless, we have encountered some challenges in modifying or storing data flows. Nonetheless, we have been able to mitigate the risk of losing all our progress by adopting a strategy to refresh data flows immediately upon opening them. This has proven to be quite beneficial, given the frequency with which we encounter the issue.

10 Conclusion

In this project, we developed a Data Warehouse solution based on the business needs of Nova Wines S.A.

With the successful implementation of a robust data warehouse solution, this traditional wine company is now equipped with powerful tools to gain insights and drive informed decision-making. The data warehouse enables us to identify and recognize top-performing sales representatives and teams based on profit, optimize commission rates to align with sales performance, and strategically reorganize segments for enhanced profitability. Additionally, we can pinpoint the top and bottom-performing products across various dimensions, including sales quantity and profit, facilitating effective inventory management and product strategy. The temporal distribution analysis provides a nuanced understanding of product category sales trends, while client and country-specific analyses empower us to cultivate key relationships and target high-growth markets. Overall, this solution positions our company at the forefront of data-driven decision-making, fostering continuous improvement, and ensuring sustained success in the dynamic global wine export market (as posed in chapter 2.3 of the report).

Adopting a star schema in our data warehouse has streamlined data retrieval, boosted query performance, and enhanced analytical capabilities. This structured design, which includes a central fact table and surrounding dimension tables, simplifies interactions with extensive sales data and, customer, supplier, sales team, date and product data, enabling quick and efficient analysis. The star schema's scalability, flexibility, and support for dimension hierarchies ensures ease of maintenance and positioning our company for agile adaptation to evolving business requirements (the relatively few joins needed make the queries fast and easy to use).

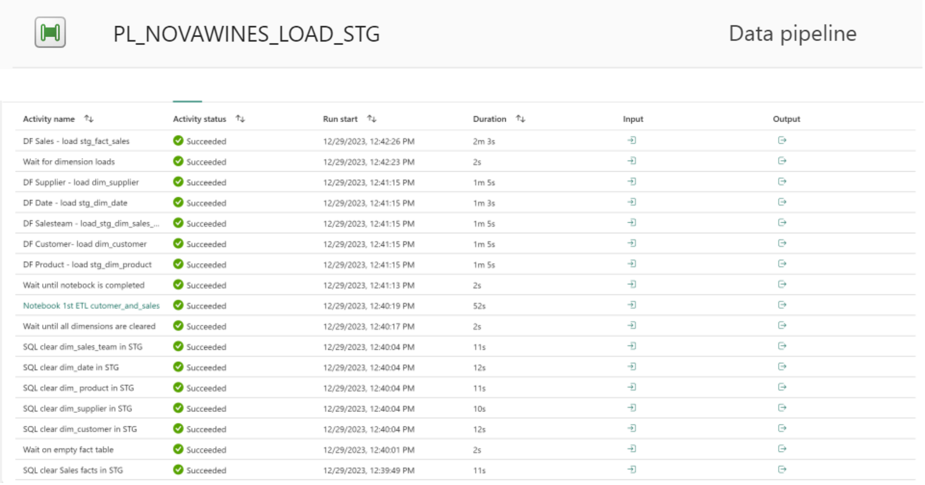
Our unconventional database creation had benefits and challenges. Recognizing the company´s data profile, we prioritized detailed ETL treatments, critical thinking, and maintainability. Despite challenges, our commitment to refining processes stands strong, resulting in a robust Data Warehouse Solution that can be conveniently updated and scheduled.

Even though the subject of our project was fictitious, we treated it like a real-life business problem. Accordingly, we paid attention to aspects such as user-friendliness, to ensure that all stakeholders could effectively use and derive insights from the data.

11 Appendix: Executed procedures

This appendix showcases our most recently completed tasks. First individually, then the full conditional loading pipeline. Also includes a failure procedure to show how the conditional statement works.

1. ETL & Loading the Staging Area



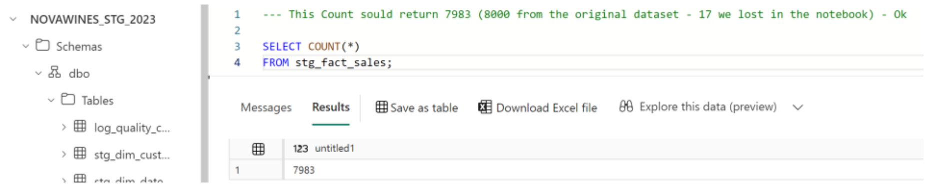
PL_NOVAWINES_LOAD_STG

Data pipeline

Activity name	Activity status	Run start	Duration	Input	Output
DF Sales - load stg_fact_sales	✓ Succeeded	12/29/2023, 12:42:26 PM	2m 3s	⇥	⇨
Wait for dimension loads	✓ Succeeded	12/29/2023, 12:42:23 PM	2s	⇥	⇨
DF Supplier - load dim_supplier	✓ Succeeded	12/29/2023, 12:41:15 PM	1m 5s	⇥	⇨
DF Date - load stg_dim_date	✓ Succeeded	12/29/2023, 12:41:15 PM	1m 3s	⇥	⇨
DF Salesteam - load_stg_dim_sales_...	✓ Succeeded	12/29/2023, 12:41:15 PM	1m 5s	⇥	⇨
DF Customer- load dim_customer	✓ Succeeded	12/29/2023, 12:41:15 PM	1m 5s	⇥	⇨
DF Product - load stg_dim_product	✓ Succeeded	12/29/2023, 12:41:15 PM	1m 5s	⇥	⇨
Wait until notebock is completed	✓ Succeeded	12/29/2023, 12:41:13 PM	2s	⇥	⇨
Notebook 1st ETL cutomer_and_sales	✓ Succeeded	12/29/2023, 12:40:19 PM	52s	⇥	⇨
Wait until all dimensions are cleared	✓ Succeeded	12/29/2023, 12:40:17 PM	2s	⇥	⇨
SQL clear dim_sales_team in STG	✓ Succeeded	12/29/2023, 12:40:04 PM	11s	⇥	⇨
SQL clear dim_date in STG	✓ Succeeded	12/29/2023, 12:40:04 PM	12s	⇥	⇨
SQL clear dim_ product in STG	✓ Succeeded	12/29/2023, 12:40:04 PM	11s	⇥	⇨
SQL clear dim_supplier in STG	✓ Succeeded	12/29/2023, 12:40:04 PM	10s	⇥	⇨
SQL clear dim_customer in STG	✓ Succeeded	12/29/2023, 12:40:04 PM	12s	⇥	⇨
Wait on empty fact table	✓ Succeeded	12/29/2023, 12:40:01 PM	2s	⇥	⇨
SQL clear Sales facts in STG	✓ Succeeded	12/29/2023, 12:39:49 PM	11s	⇥	⇨

Figure 29. *Successful pipeline running 1*

stg_fact_sales row count, 7983 (8000 – 17 from the Notebook cleaning)



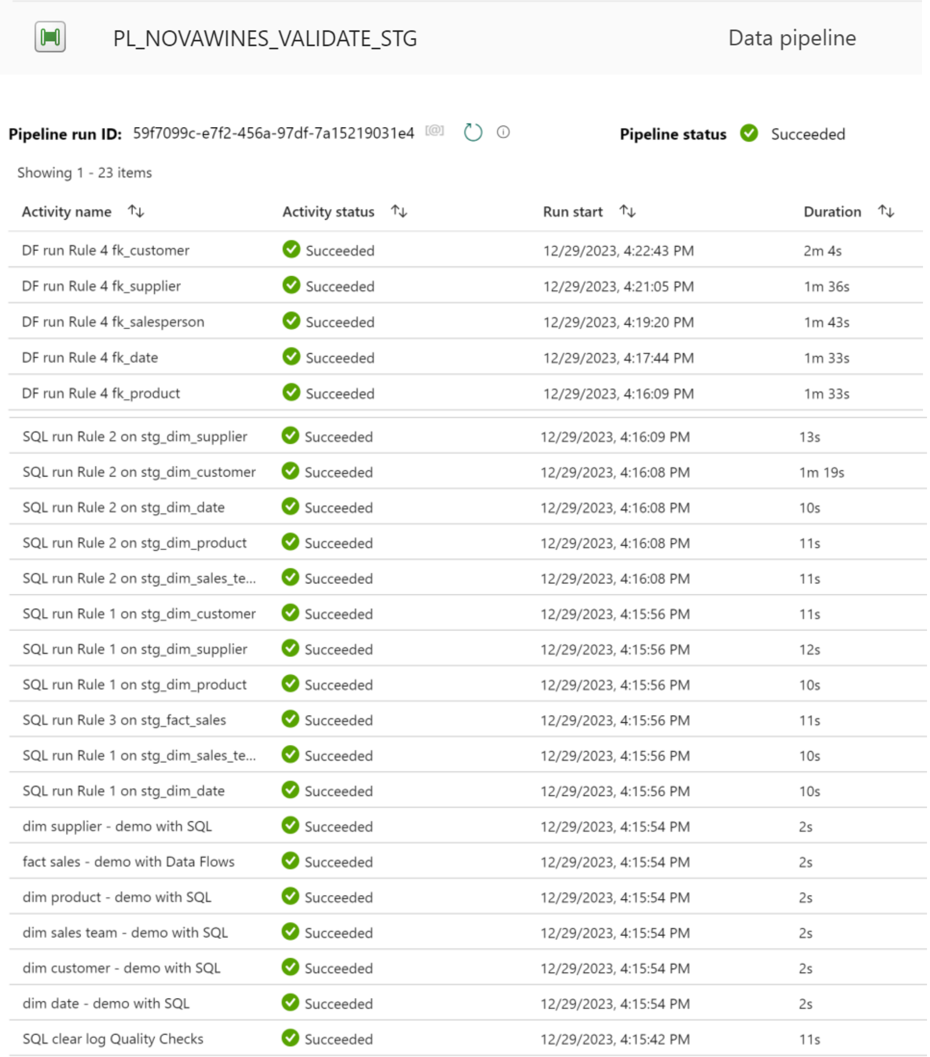
```
--- This Count sould return 7983 (8000 from the original dataset - 17 we lost in the notebook) - Ok

SELECT COUNT(*)
FROM stg_fact_sales;
```

	untitled1
1	7983

Figure 30. *stg_fact_sales row count*

2. Validation Rules



PL_NOVAWINES_VALIDATE_STG — Data pipeline

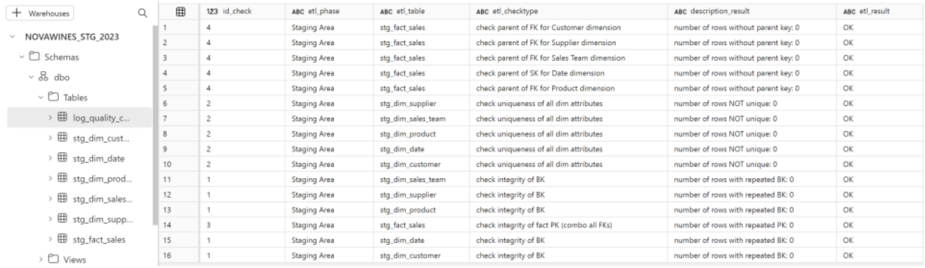
Pipeline run ID: 59f7099c-e7f2-456a-97df-7a15219031e4 — **Pipeline status** ✓ Succeeded

Showing 1 - 23 items

Activity name	Activity status	Run start	Duration
DF run Rule 4 fk_customer	✓ Succeeded	12/29/2023, 4:22:43 PM	2m 4s
DF run Rule 4 fk_supplier	✓ Succeeded	12/29/2023, 4:21:05 PM	1m 36s
DF run Rule 4 fk_salesperson	✓ Succeeded	12/29/2023, 4:19:20 PM	1m 43s
DF run Rule 4 fk_date	✓ Succeeded	12/29/2023, 4:17:44 PM	1m 33s
DF run Rule 4 fk_product	✓ Succeeded	12/29/2023, 4:16:09 PM	1m 33s
SQL run Rule 2 on stg_dim_supplier	✓ Succeeded	12/29/2023, 4:16:09 PM	13s
SQL run Rule 2 on stg_dim_customer	✓ Succeeded	12/29/2023, 4:16:08 PM	1m 19s
SQL run Rule 2 on stg_dim_date	✓ Succeeded	12/29/2023, 4:16:08 PM	10s
SQL run Rule 2 on stg_dim_product	✓ Succeeded	12/29/2023, 4:16:08 PM	11s
SQL run Rule 2 on stg_dim_sales_te...	✓ Succeeded	12/29/2023, 4:16:08 PM	11s
SQL run Rule 1 on stg_dim_customer	✓ Succeeded	12/29/2023, 4:15:56 PM	11s
SQL run Rule 1 on stg_dim_supplier	✓ Succeeded	12/29/2023, 4:15:56 PM	12s
SQL run Rule 1 on stg_dim_product	✓ Succeeded	12/29/2023, 4:15:56 PM	10s
SQL run Rule 3 on stg_fact_sales	✓ Succeeded	12/29/2023, 4:15:56 PM	11s
SQL run Rule 1 on stg_dim_sales_te...	✓ Succeeded	12/29/2023, 4:15:56 PM	10s
SQL run Rule 1 on stg_dim_date	✓ Succeeded	12/29/2023, 4:15:56 PM	10s
dim supplier - demo with SQL	✓ Succeeded	12/29/2023, 4:15:54 PM	2s
fact sales - demo with Data Flows	✓ Succeeded	12/29/2023, 4:15:54 PM	2s
dim product - demo with SQL	✓ Succeeded	12/29/2023, 4:15:54 PM	2s
dim sales team - demo with SQL	✓ Succeeded	12/29/2023, 4:15:54 PM	2s
dim customer - demo with SQL	✓ Succeeded	12/29/2023, 4:15:54 PM	2s
dim date - demo with SQL	✓ Succeeded	12/29/2023, 4:15:54 PM	2s
SQL clear log Quality Checks	✓ Succeeded	12/29/2023, 4:15:42 PM	11s

Figure 31. Successful pipeline running 2

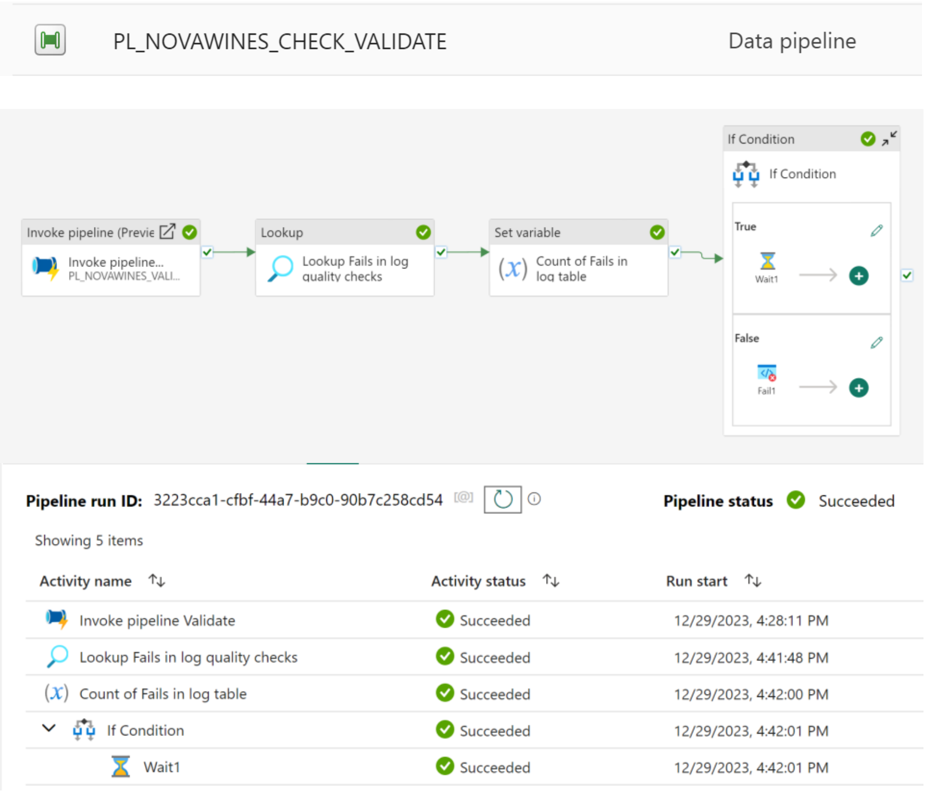
log_quality_checks:



	id_check	etl_phase	etl_table	etl_checktype	description_result	etl_result
1	4	Staging Area	stg_fact_sales	check parent of FK for Customer dimension	number of rows without parent key: 0	OK
2	4	Staging Area	stg_fact_sales	check parent of FK for Supplier dimension	number of rows without parent key: 0	OK
3	4	Staging Area	stg_fact_sales	check parent of FK for Sales Team dimension	number of rows without parent key: 0	OK
4	4	Staging Area	stg_fact_sales	check parent of SK for Date dimension	number of rows without parent key: 0	OK
5	4	Staging Area	stg_fact_sales	check parent of FK for Product dimension	number of rows without parent key: 0	OK
6	2	Staging Area	stg_dim_supplier	check uniqueness of all dim attributes	number of rows NOT unique: 0	OK
7	2	Staging Area	stg_dim_sales_team	check uniqueness of all dim attributes	number of rows NOT unique: 0	OK
8	2	Staging Area	stg_dim_product	check uniqueness of all dim attributes	number of rows NOT unique: 0	OK
9	2	Staging Area	stg_dim_date	check uniqueness of all dim attributes	number of rows NOT unique: 0	OK
10	2	Staging Area	stg_dim_customer	check uniqueness of all dim attributes	number of rows NOT unique: 0	OK
11	1	Staging Area	stg_dim_sales_team	check integrity of BK	number of rows with repeated BK: 0	OK
12	1	Staging Area	stg_dim_supplier	check integrity of BK	number of rows with repeated BK: 0	OK
13	1	Staging Area	stg_dim_product	check integrity of BK	number of rows with repeated BK: 0	OK
14	3	Staging Area	stg_fact_sales	check integrity of fact PK (combo all FKs)	number of rows with repeated PK: 0	OK
15	1	Staging Area	stg_dim_date	check integrity of BK	number of rows with repeated BK: 0	OK
16	1	Staging Area	stg_dim_customer	check integrity of BK	number of rows with repeated BK: 0	OK

Figure 32. *Results from the rules*

3. Test the Validation results with Lookup and If Condition



Pipeline run ID: 3223cca1-cfbf-44a7-b9c0-90b7c258cd54

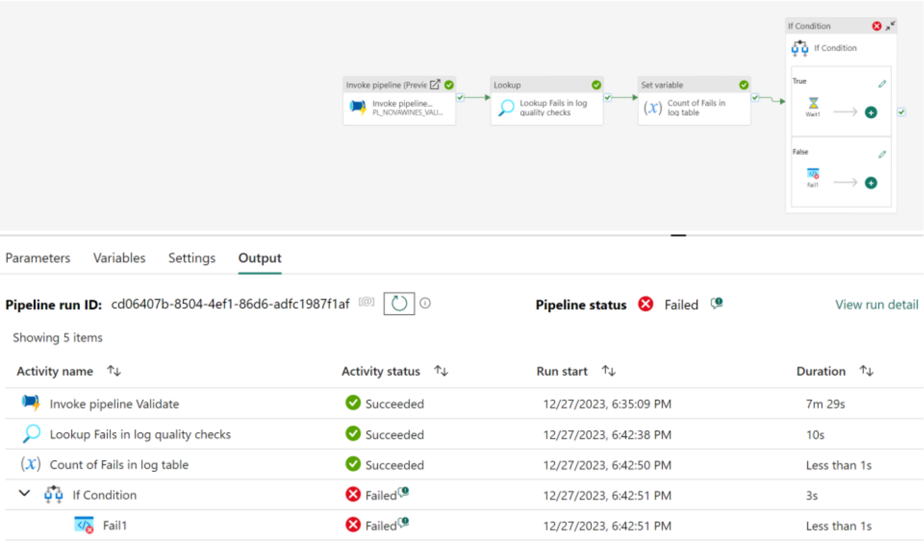
Pipeline status ✓ Succeeded

Showing 5 items

Activity name	Activity status	Run start
Invoke pipeline Validate	✓ Succeeded	12/29/2023, 4:28:11 PM
Lookup Fails in log quality checks	✓ Succeeded	12/29/2023, 4:41:48 PM
Count of Fails in log table	✓ Succeeded	12/29/2023, 4:42:00 PM
If Condition	✓ Succeeded	12/29/2023, 4:42:01 PM
Wait1	✓ Succeeded	12/29/2023, 4:42:01 PM

Figure 33. *Successful pipeline running 3*

As a reference, when the date information did not fully match the sales orders fact table, we obtained the following result in this pipeline:


Invoke pipeline (Previe
Invoke pipeline... PL_NOVAWINES_VALI...
Lookup
Lookup Fails in log quality checks
Set variable
Count of Fails in log table
If Condition
If Condition
True
Wait1
False
Fail1

Parameters Variables Settings **Output**

Pipeline run ID: cd06407b-8504-4ef1-86d6-adfc1987f1af

Pipeline status Failed

View run detail

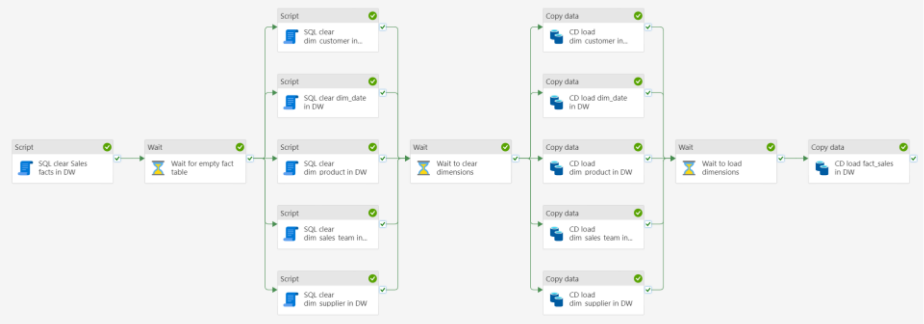
Showing 5 items

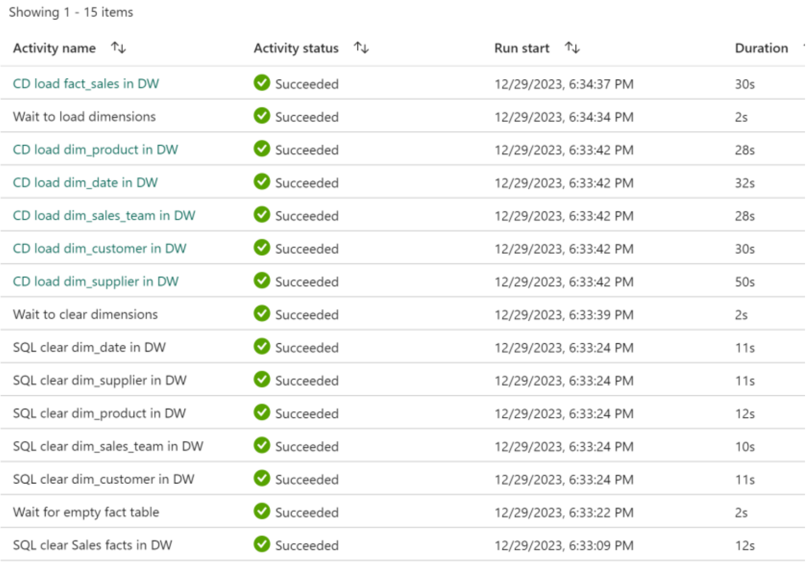
Activity name	Activity status	Run start	Duration
Invoke pipeline Validate	Succeeded	12/27/2023, 6:35:09 PM	7m 29s
Lookup Fails in log quality checks	Succeeded	12/27/2023, 6:42:38 PM	10s
Count of Fails in log table	Succeeded	12/27/2023, 6:42:50 PM	Less than 1s
If Condition	Failed	12/27/2023, 6:42:51 PM	3s
Fail1	Failed	12/27/2023, 6:42:51 PM	Less than 1s

Figure 34. Unsuccessful pipeline running

4. Load the Data Warehouse & SK configuration

PL_NOVAWINES_LOAD_DW Data pipeline


Script
SQL clear Sales facts in DW
Wait
Wait for empty fact table
Script
SQL clear dim customer in...
Script
SQL clear dim_date in DW
Script
SQL clear dim product in DW
Script
SQL clear dim sales team in...
Script
SQL clear dim supplier in DW
Wait
Wait to clear dimensions
Copy data
CD load dim customer in...
Copy data
CD load dim_date in DW
Copy data
CD load dim product in DW
Copy data
CD load dim sales team in...
Copy data
CD load dim supplier in DW
Wait
Wait to load dimensions
Copy data
CD load fact_sales in DW

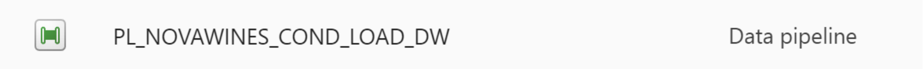


Showing 1 - 15 items

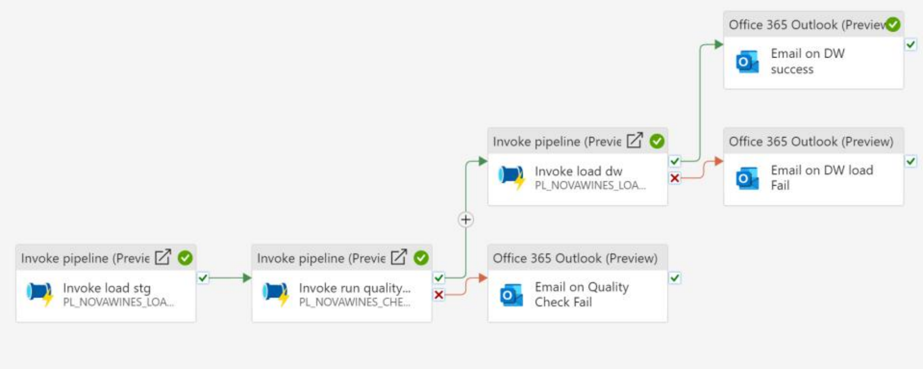
Activity name	Activity status	Run start	Duration
CD load fact_sales in DW	✓ Succeeded	12/29/2023, 6:34:37 PM	30s
Wait to load dimensions	✓ Succeeded	12/29/2023, 6:34:34 PM	2s
CD load dim_product in DW	✓ Succeeded	12/29/2023, 6:33:42 PM	28s
CD load dim_date in DW	✓ Succeeded	12/29/2023, 6:33:42 PM	32s
CD load dim_sales_team in DW	✓ Succeeded	12/29/2023, 6:33:42 PM	28s
CD load dim_customer in DW	✓ Succeeded	12/29/2023, 6:33:42 PM	30s
CD load dim_supplier in DW	✓ Succeeded	12/29/2023, 6:33:42 PM	50s
Wait to clear dimensions	✓ Succeeded	12/29/2023, 6:33:39 PM	2s
SQL clear dim_date in DW	✓ Succeeded	12/29/2023, 6:33:24 PM	11s
SQL clear dim_supplier in DW	✓ Succeeded	12/29/2023, 6:33:24 PM	11s
SQL clear dim_product in DW	✓ Succeeded	12/29/2023, 6:33:24 PM	12s
SQL clear dim_sales_team in DW	✓ Succeeded	12/29/2023, 6:33:24 PM	10s
SQL clear dim_customer in DW	✓ Succeeded	12/29/2023, 6:33:24 PM	11s
Wait for empty fact table	✓ Succeeded	12/29/2023, 6:33:22 PM	2s
SQL clear Sales facts in DW	✓ Succeeded	12/29/2023, 6:33:09 PM	12s

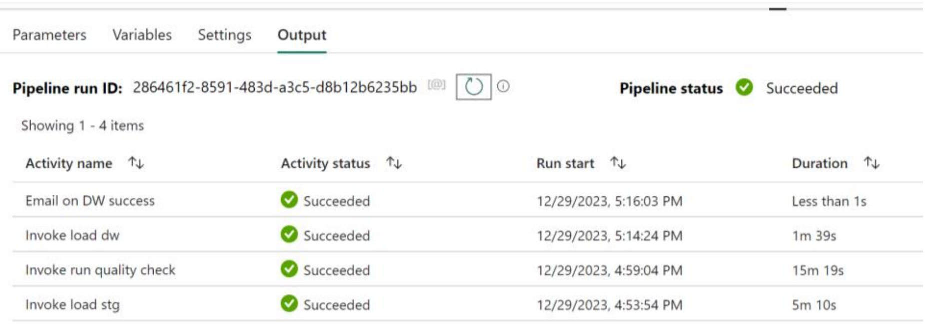
Figure 35. Successful pipeline running 4

5. Full conditional Loading



PL_NOVAWINES_COND_LOAD_DW | Data pipeline





Pipeline run ID: 286461f2-8591-483d-a3c5-d8b12b6235bb

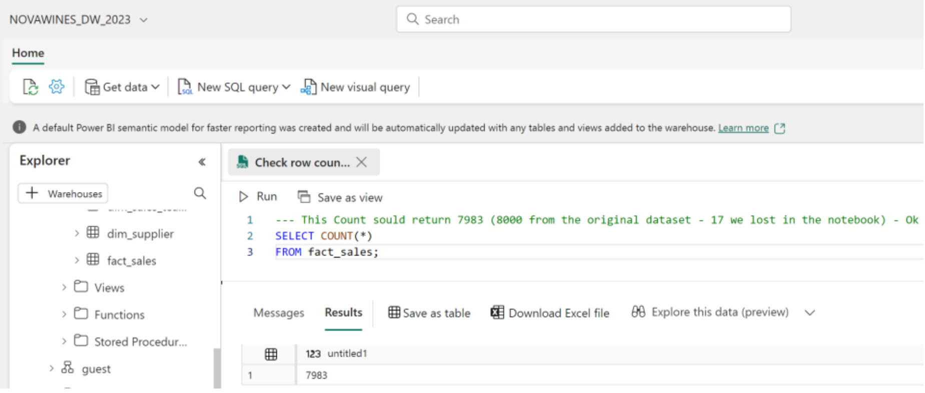
Pipeline status ✓ Succeeded

Showing 1 - 4 items

Activity name	Activity status	Run start	Duration
Email on DW success	✓ Succeeded	12/29/2023, 5:16:03 PM	Less than 1s
Invoke load dw	✓ Succeeded	12/29/2023, 5:14:24 PM	1m 39s
Invoke run quality check	✓ Succeeded	12/29/2023, 4:59:04 PM	15m 19s
Invoke load stg	✓ Succeeded	12/29/2023, 4:53:54 PM	5m 10s

Figure 36. Successful pipeline running 5

To check the number of rows in the `fact_sales` table, we execute a query (available in shared queries):



A default Power BI semantic model for faster reporting was created and will be automatically updated with any tables and views added to the warehouse. Learn more

```sql
--- This Count sould return 7983 (8000 from the original dataset - 17 we lost in the notebook) - Ok
SELECT COUNT(*)
FROM fact_sales;
```

	123 untitled1
1	7983

Figure 37. fact_sales row count