BUSINESS INTELLIGENCE I COURSE'S PROJECT

REPORT ON DELIVERY 1

Professor Pedro Batista

Group 049
Raimundo Mujica Costa M20221342
Simão Pedro Acciaioli Gouveia Dos Santos M20220561
Tomás Caldeira Cardoso Soares Esteves M20220377
Waleed Bin Tanveer Sabir M20220073

Table of Contents

Table of Illustrations

Introduction

In the advent of the digital age, the data we collect has surpassed the amount we predicted. This change has caused businesses to evolve and adapt to this new era. With the ever-increasing competition between companies, any edge provided is very beneficial.

This advantage can be had in many ways than one. Data has taken its rightful place as one of the most powerful tools at a company's disposal. Gaining an advantage using data can include but is not limited to streamlining logistics to decrease turnover time to the customer to seeing what products generate a greater return in a much more quantifiable way. Managing a sizeable company becomes a lot easier whilst also providing better strategies for the future of the company, employees, and customers.

One of the key challenges amongst many in gaining an advantage using data is to extract valuable information from a sea of unfiltered data. There are tools that are very efficient in collecting data but in most scenarios, this data is not actionable evidence towards a future strategy. There are systems that can be implemented to filter out the necessary information. The next step is storing this and analysing the useful information.

The potential of these steps is being realized by many business owners and there's a natural evolution in using Business Intelligence solutions to gain the maximum advantage against the competition. Storing and analysing data is one of the early steps in creating a system that will aid the business in making better decisions based on information gathered. This is further compounded with identifying relevant markers that provide valuable insight into the market in which the company operates.

This structured approach to managing a business with the help of using and organizing data is referred to as Data Warehousing (DW). All the information is sorted and divided into multiple streams and used by relevant departments.

During Business Intelligence I, we were tasked with organizing and creating a DW solution for Killer Glute Bikes. We intend to streamline their data flow by organizing and sorting their client and company data followed by loading it into an intelligent solution (ETL). This will help in making informed short-term decisions for the company whilst also paving a way for future strategies that will pay dividends in the long run and prove to be the determining factors in the business's long-term growth and success.

The Company

Introduction to Killer Glute Bikes (KGB) company

“At Killer Glute Bikes (KGB) we believe in a healthy lifestyle for you and your family whilst facilitating an unforgettable outdoors experience for all!
We believe that it is in the wild where one finds their true self and helps awaken a satiation for nature and all its gifts.
Since 1905, we have been your international outdoor companion to aid your experience in the true transformational power that nature has to offer. We offer top quality bikes, gear and apparel coupled with our expert advice to help you create inspiring stories and experience special moments alone or with your family and friends. And because we do not have shareholders, your proceeds with KGB will directly benefit the outdoors and support a sustainable business in the fight for life outdoors.
So, whether you are new to the outdoors or a seasoned pro, we hope you will join us on this journey!”

Business Model

Killer Glute Bikes is a worldwide bike and bike apparel seller that operates in three different continents (Europe, Oceania, and North America), five different countries. For that reason, to grant the same quality service globally and therefore maintaining its position as a leading company within its line of business, Killer Glute Bikes structure follows some specific guidelines that ease this mission.
Each product that is available at the KGB stores comes from the same exact supplier. Allying this principle with a meticulous choice in their providers guarantees that Killer Glutes Bikes can deliver the same top tier quality products in all its stores. Even that this model might increase the freight in some products supply chain, with this business plan the company does not have to spend any resources on external warehouses fees.

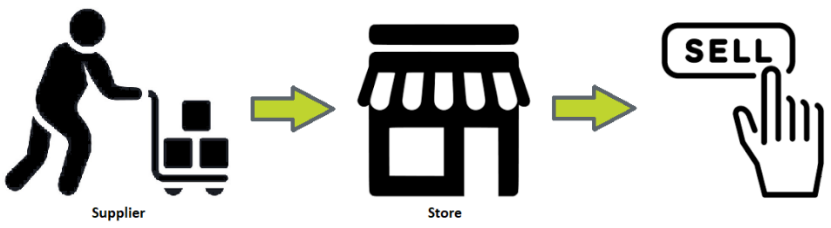


Figure i Supply chain model

Killer Glute Bikes company organization found that the best way to keep its promise of delivering a specialized advice on selling their products to their clients was to fill all its stores with at least one expert in bikes’ related subjects.

Problem Scenario and Business needs

KGB is currently undergoing a restructuration of its organizational management process after a purchase of most of its shares by a new owner. As such, the new CEO needs to understand the current situation of the company.
Having a Data Warehouse with all the information compiled will allow a more stoic approach to solve company’s problems and get a better understanding of the business itself.
For the new board, the primary focus is to optimize the company’s spendings in the multiple markets they are committed. They hope that with the practice of more sustainable procedures, the organization will be more improve their logistical systems to decrease turnover time and increase the understanding of sales to make better decisions for future strategies pertaining to products.

To properly understand the company's reality and figuring out its next move, we identified the following questions:

1. What is the type of products most sold by market and date?
2. What is the average transportation cost, per type of transport, per product, per Market?
3. What is the client segment with the biggest sales by market and date?
4. What is our customer profile by product, product category, and product subcategory?
5. How do product sales vary during the year?
6. What is our average revenue per sale per product type?
7. Which region has higher profits? What is the best investment resolution?

Data Sources

Killer Glute Bikes (OLTP Database)

The data of Killer Glute Bikes company describes their sales, shippers' activity and covers some general facts that, although not being directly involved in the business itself, provide crucial details for the company's operations. For assembling this database, we considered only "meaningful data". This means only the data that we initially perceived as being useful on the resolution of the company's problem scenario was extracted from Killer Glute Bikes' huge dataset.

Data Entities and their Attributes

We selected the entities and their attributes which we considered to give us some good insights on subjects we expected to be relevant on a furtherer analysis.

- **Date table**
  A date table is crucial to carry on this data analysis as it provides an extensive cover on time dimensions, essential to a transactional database.

  String type attributes:
  → full_name_date [nvarchar(100) not null]
  → day_name [nvarchar(50) not null]
  → month_name [nvarchar(50) not null]

  Numeric type attributes:
  🔑 pk_date [int not null] (yyyymmdd code)
  → day_number_of_week [int not null]
  → month_number_of_year [int not null]
  → day_number_of_year [int not null]
  → week_number_of_year [int not null]
  → calendar_quarter [int not null]
  → calendar_year [int not null]
  → day_number_of_month [int not null]

  Date and Time type attributes:
  → bk_period [date not null]

- **ProductCategory table**
  Data about the category of the product.

  String type attributes:
  🔑 pk_productcategory [nvarchar(20)]
  → english_productcategory_name [nvarchar(50)]
  → german_productcategory_name [nvarchar(50)]
  → french_productcategory_name [nvarchar(50)]

- **ProductSubCategory table**
  Data about the subcategory of the product. Provides a more precise partition on products' type.

  <u>String type attributes:</u>
  🔑 pk_productcategory [nvarchar(50)]
  → fk_productcategory [nvarchar(20)] ref *pk_productcategory in Product Category Table*
  → english_productsubcategory_name [nvarchar(200)]
  → german_productsubcategory_name [nvarchar(200)]
  → french_productsubcategory_name [nvarchar(200)]

- **Suppliers table**
  Data regarding the suppliers of the products.

  <u>String type attributes:</u>
  🔑 pk_supplier [nvarchar(20)]
  → supplier_name [nvarchar(50)]
  → contact_details [nvarchar(100)]

  <u>Numeric type attributes:</u>
  → transportation_fees [int not null] (if the fees for transportation are charged)

- **Products table**
  Detailed data regarding products' specifications.

  <u>String type attributes:</u>
  → fk_productcategory [nvarchar(50) not null] ref *pk_productcategory in Product Category Table*
  → fk_productsubcategory [nvarchar(50) not null] ref *pk_productsubcategory in Product SubCategory Table*
  → model_name [nvarchar(50)]
  → details [nvarchar(250)]
  → fk_supplier [nvarchar(50)]

  <u>Numeric type attributes:</u>
  🔑 pk_productkey [nvarchar(20)]
  → standard_cost [decimal(18,10)]
  → colour [nvarchar(50)]
  → list_price [decimal(18,10)]
  → size [int]
  → active [int]

  <u>Date and Time type attributes:</u>
  → product_start_date [date not null]
  → product_end_date [date not null]

- **Territory table**

  General geographical data.

  String type attributes:

  🔑 pk_territorykey [nvarchar(50)]
  → region [nvarchar(50)]
  → country [nvarchar(50)]
  → group_division [nvarchar(50)]

- **SubTerritory table**

  Geographical data that complements Territory table.

  String type attributes:

  🔑 pk_territorysubcategory [nvarchar(50)]
  → fk_territorykey [nvarchar(50)] ref *pk_territorykey in Territory Table*
  → city_name [nvarchar(50)]
  → postal_code nvarchar(50)

  Numeric type attributes:

  → bike_lanes_km [decimal(18,10) not null]
  → protected_bike_lanes_km [decimal(18,10) not null]

- **Stores table**

  Data about the company's stores network.

  String type attributes:

  🔑 pk_storeid [nvarchar(50)]
  → fk_territorysubcategory [nvarchar(50)] ref *pk_territorysubcategroy in SubTerritory Table*
  → city [nvarchar(50)]

  Numeric type attributes:

  → area_m2 [decimal(18,10) not null]
  → rent_per_month [decimal(18,10)]
  → staff_instore [int not null]
  → expert_staff [int not null]
  → average_staff_salary_month [int not null]
  → accessibility_1to5 [int not null] (measures the level of accessibility from category one to category five)
  → showcase_quality_level_1to5 [int not null] (measures the level of the showcase from category one to category five)

- **Customers table**

  Company’s client data.

  <u>String type attributes:</u>

  → customer_name [nvarchar(50)]
  → marital_status [nvarchar(50)]
  → gender [nvarchar(50)]
  → occupation [nvarchar(50)]
  → address_line [nvarchar(50)]
  → phone [nvarchar(50)]

  <u>Numeric type attributes:</u>

  🔑 pk_customerkey [int]
  → yearly_income [int not null]
  → number_children_at_home [int not null]
  → number_cars_owned [int not null]
  → fk_date_first_purchaseid [int] ref *pk_date in Date Table*

  <u>Date and Time type attributes:</u>

  → birth_date [date not null]

- **FactSales table**

  Sales and shipment data.

  <u>String type attributes:</u>

  🔑 fk_territorykey [nvarchar(50)]
  🔑 fk_territorysubcategory [nvarchar(50)]
  🔑 fk_storeid [nvarchar(50)]

  <u>Numeric type attributes:</u>

  🔑 fk_date [int not null]
  🔑 fk_productkey [int not null]
  🔑 fk_customerkey [int not null]
  → sales_order_line_number [int not null]
  → order_quantity [int not null]
  → unit_price [decimal(15,3) not null]
  → product_standard_cost [decimal(10,3) not null]
  → sales_amount [decimal(15,5) not null]
  → tax_amount_unit [decimal(10,3) not null]
  → tax_amount [decimal(15,3) not null]
  → freight [decimal(10,5) not null]
  → total_cost [decimal(15,5) not null]

  <u>Date and Time type attributes:</u>

  → order_date [date]

Data Entities' Relations

We created a diagram in SQL Management Studio that can show the dataset connections with great clarity.

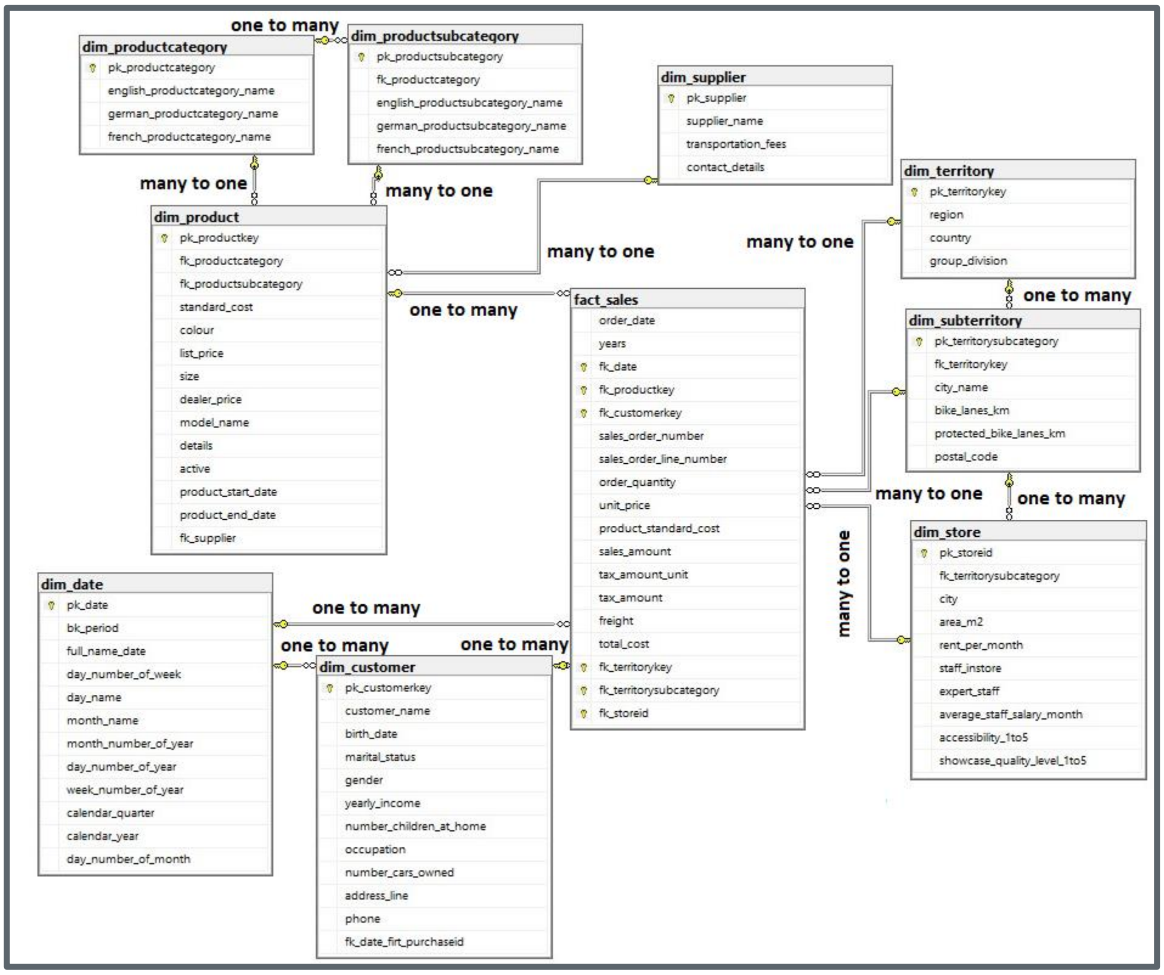


Figure ii Database tables' relation

Developing the Data Warehouse

The main goal of a data warehouse is to store data in an appropriate way so that it is the major support on decision-making process. The principal characteristics that define a data warehouse are the following:

- **Subject-oriented:** as stated before, the main purpose of the data warehouse is to give a support to decision makers by analysing data. It is not designed to assist on daily operations.
- **Integrated:** the assemble of a data warehouse is based upon integrating heterogenous sources.
- **Time-Variant:** the data stored in a data warehouse is not keeping up with the company's live data. Instead, this data provides information from an historical point of view.
- **Non-volatile:** the data warehouse and the data used in the operational environment are always set up individually.

We envisioned to use a Star Schema design for our Data Warehouse as despite using a considerable amount of space, it allows an easier and more concise information split. On the following subchapters we describe our thinking process and appoint the reasons that explain the Killer Glute Bikes' data warehouse design.

Methodology

After assuring every team member had the right mindset and impressions on the importance and requisites for the under development Wonderful Wines of the World Data Warehouse, we developed a working task procedure to ensure we reach our goals.

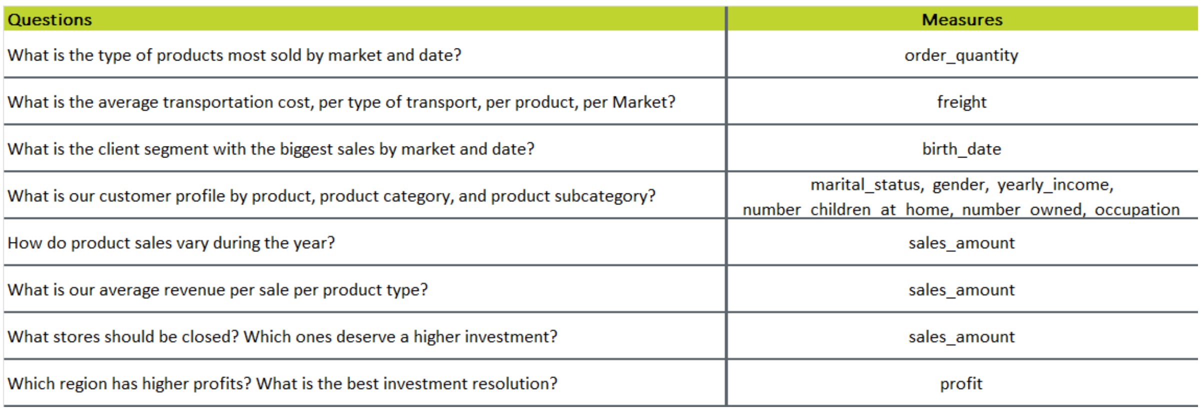
First, bearing in mind the business case questions we need to approach, we would check if we had the necessary measures to evaluate WWW's situation. Probably, after this check-up we would have to define new attributes that will aid the Data Warehouse's users to reach the desired conclusions. We also went through a second look on all of data we had at our disposal: this would give us an indispensable assistance on reaching the Data Warehouse's design that better fits the company's purpose.

This breakthrough scanning will prove to be essential when we finally "decide on the looks" of the data warehouse. At this point, the decision to keep the warehouse as simple and easily readable to its users and future re-developers was already taken.

After structuring what we though as an efficient assembly line for the pre-schematized Data Warehouse, we finally began to give it form.

Getting the right based approach on design the Data Warehouse to answer Business Needs

To get the right overview of the company’s current business reality and achieving a profile on possible resolutions to its future goals, we pointed out the required measures that must figure in the Data Warehouse.



Questions	Measures
What is the type of products most sold by market and date?	order_quantity
What is the average transportation cost, per type of transport, per product, per Market?	freight
What is the client segment with the biggest sales by market and date?	birth_date
What is our customer profile by product, product category, and product subcategory?	marital_status, gender, yearly_income, number_children_at_home, number_owned, occupation
How do product sales vary during the year?	sales_amount
What is our average revenue per sale per product type?	sales_amount
What stores should be closed? Which ones deserve a higher investment?	sales_amount
Which region has higher profits? What is the best investment resolution?	profit

Figure iii Business questions required measures

The *profit* measure does not figure in KGB’s dataset. As it plays a crucial role in further analysis, we will have to create a new field for evaluating this subject.
By using the attributes *sales_amount* and *total_cost*, both from FactSales table we will be able to get this measure.

$$\text{profit} = \text{sales\_amount} - \text{total\_cost}$$

We also will be asked on counselling on closing stores. For that reason, we will establish an attribute referring to the store status: active or unactive.

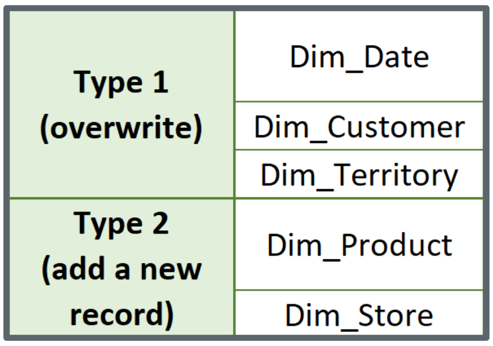
We have sorted out the relevant information broken down into multiple dimensions that would be needed in our DW solution for KGB coupled with the OLTP data source which is going to input a regular stream of new data to the relevant dimensions. These dimensions were carefully chosen based on the attributes that are going to be beneficial in the analyzation phase.

Dimensions	OLTP Entities	Key Data Type
sk_date	Dbo.Date	int
bk_period	Dbo.Date	date
full_name_date	Dbo.Date	nvarchar(50)
day_number_of_week	Dbo.Date	int
day_name	Dbo.Date	nvarchar(50)
month_name	Dbo.Date	nvarchar(50)
month_number_of_year	Dbo.Date	int
day_number_of_year	Dbo.Date	int
week_number_of_year	Dbo.Date	int
calendar_quarter	Dbo.Date	int
calendar_year	Dbo.Date	int
day_number_of_month	Dbo.Date	int
sk_productid	n/a	int
bk_product	Dbo.Product	int
bk_productcategory	Dbo.ProductCategory	nvarchar(50)
bk_productsubcategory	Dbo.ProductSubCategory	nvarchar(50)
productcategory_name	Dbo.ProductCategory	nvarchar(50)
productsubcategory_name	Dbo.ProductSubCategory	nvarchar(50)
standard_cost	Dbo.Product	decimal(18, 10)
colour	Dbo.Product	nvarchar(50)
list_price	Dbo.Product	decimal(18, 10)
size	Dbo.Product	int
dealer_price	Dbo.Product	decimal(18, 10)
model_name	Dbo.Product	nvarchar(50)
details	Dbo.Product	nvarchar(250)
product_start_date	Dbo.Product	date
product_end_date	Dbo.Product	date
bk_supplier	Dbo.Supplier	nvarchar(50)
supplier_name	Dbo.Supplier	nvarchar(50)
transportation_fees	Dbo.Supplier	int
sk_customerid	n/a	int
sk_customer	Dbo.Customer	int
customer_name	Dbo.Customer	nvarchar(50)
birth_date	Dbo.Customer	date
marital_status	Dbo.Customer	nvarchar(50)
gender	Dbo.Customer	nvarchar(50)
yearly_income	Dbo.Customer	int
number_children_at_home	Dbo.Customer	int
occupation	Dbo.Customer	nvarchar(50)
number_cars_owned	Dbo.Customer	int
sk_storeid	n/a	int
bk_store	Dbo.Store	nvarchar(50)
area_m2	Dbo.Store	decimal(18, 10)
rent_per_month	Dbo.Store	decimal(18, 10)
staff_instore	Dbo.Store	int
expert_staff	Dbo.Store	int
average_staff_salary_month	Dbo.Store	int
accesibility_1to5	Dbo.Store	int
showcase_quality_level_1to5	Dbo.Store	int
active	n/a	int
sk_territoryid	n/a	int
bk_territorysubcategory	Dbo.TerritorySubCategory	nvarchar(50)
bk_territorykey	Dbo.Territory	nvarchar(50)
region	Dbo.Territory	nvarchar(50)
group_division	Dbo.Territory	nvarchar(50)
country	Dbo.Territory	nvarchar(50)
city_name	Dbo.TerritorySubCategory	nvarchar(50)
bike_lanes_km	Dbo.TerritorySubCategory	decimal(18, 10)
protected_bike_lanes_km	Dbo.TerritorySubCategory	decimal(18, 10)

Figure iv Entities and their dimensions

Slowly Changing Dimensions

Optimizing the Data Warehouse for better responding to changes that may occur in the future is essential. For this reason, we consider implementing two different types of SCD:



Type 1 (overwrite)	Dim_Date
	Dim_Customer
	Dim_Territory
Type 2 (add a new record)	Dim_Product
	Dim_Store

Figure v SCD types

For Dim_Date, Dim_Customer and Dim_Territory we decided not to implement a new record for future changes, but to rewrite on the previous values. The reason behind this decision comes from the vague modifications that will have an impact on the business decisions over time.

We believe that Dim_Product and Dim_Store have important information about our business needs, and we expect these dimensions to change over time. Implementing a new record such as a SCD will allow us to have complete information from start to finish. In addition, for future analysis, since we have complete information about our business, we will obtain better results. For Dim_Product we added two columns that represent the start date of a product and the date when the same product changed supplier or when the same product stopped producing. For Dim_Store we add a column that shows which are the stores that are active, representing them with a number, one, and which are the stores that are inactive, representing them with a number, zero.

For Dim_Product, as we mentioned before, we consider that some products in the future will suffer changes. One of those changes would be if the product modifies its specification, like supplier. Another of those changes would be if we decide to stop the production of a specific product. Finally, it is probable that in the future we will decide to change the name of a category or subcategory. Adding a new record will help us to have the complete history of our product over time.

In the case of Dim_Store, we do not know what could happened on the KGB's future. For this reason, we must face a situation in which the company does not have the best results and we must close some stores. Add an additional column implementing the SCD type 2 will help us for a situation like this. So, we can have the complete store's status over time.

Assembled Data Warehouse: Dimension Tables and Schema

We decided to establish six dimensions that we found that would manage to answer all Killer Glute Bikes' questions and needs. The dimensions were assembled on a certain order (figure vi), that ensures the Data Warehouse's relations existence. We methodize this step by assuring the dimensions with foreign keys were only coded after the pointed attribute was already established. Briefly, we followed the logical granularity for our Data Warehouse's design.

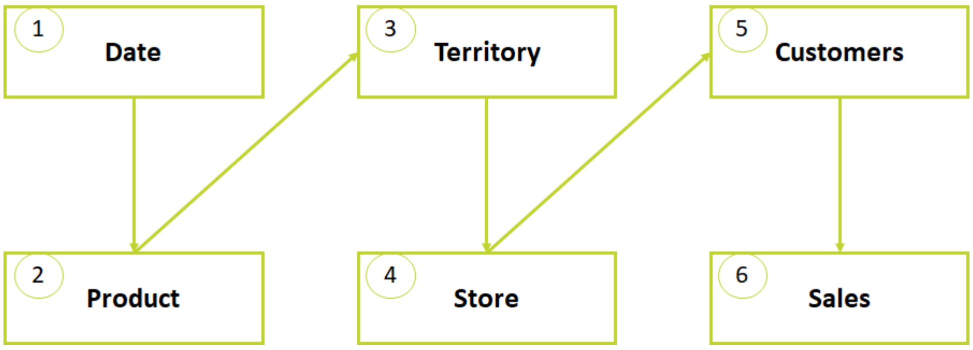
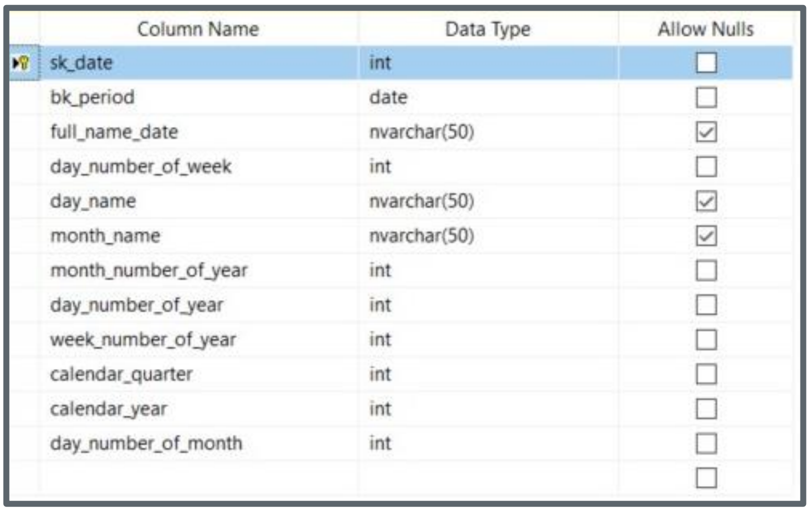


Figure vi Dimensions' order creation

Dimension Date

This table contains all the information pertaining to the calendar. This table serves to be a reference point for the billing department to keep track of day-to-day sales and provides an outlook on aggregate sales (monthly, yearly, etc.).

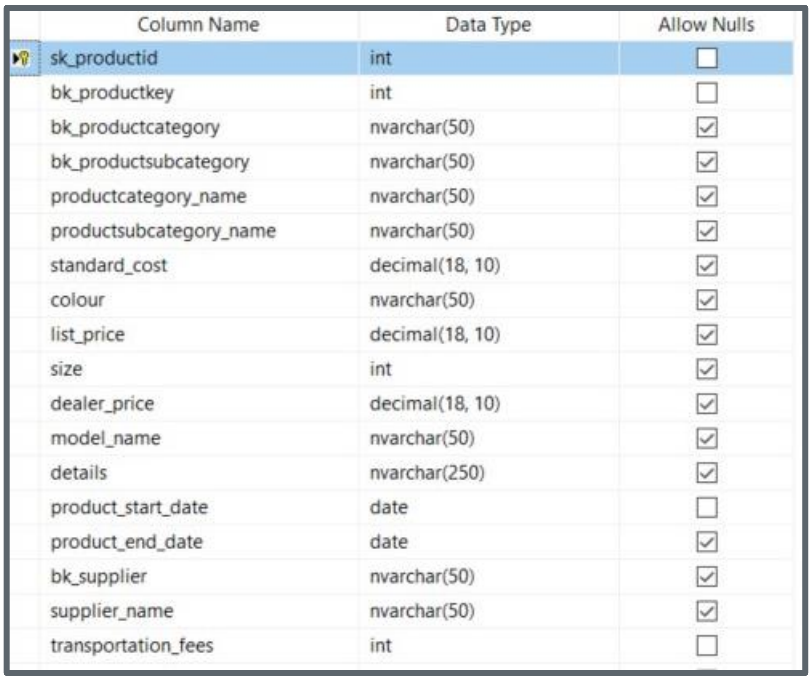


Column Name	Data Type	Allow Nulls
sk_date	int	☐
bk_period	date	☐
full_name_date	nvarchar(50)	☑
day_number_of_week	int	☐
day_name	nvarchar(50)	☑
month_name	nvarchar(50)	☑
month_number_of_year	int	☐
day_number_of_year	int	☐
week_number_of_year	int	☐
calendar_quarter	int	☐
calendar_year	int	☐
day_number_of_month	int	☐
		☐

Figure vii Dimension Date table

Dimension Product

The product table provides the information about all the products the company sells. Including but not limited to the pricing, description, model name etc.

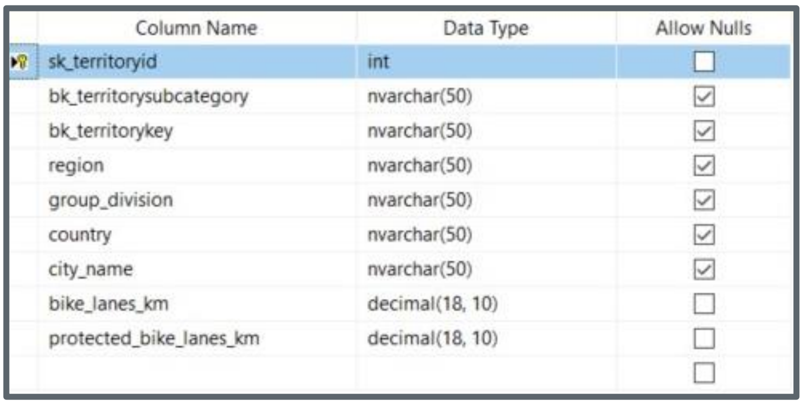


Column Name	Data Type	Allow Nulls
sk_productid	int	☐
bk_productkey	int	☐
bk_productcategory	nvarchar(50)	☑
bk_productsubcategory	nvarchar(50)	☑
productcategory_name	nvarchar(50)	☑
productsubcategory_name	nvarchar(50)	☑
standard_cost	decimal(18, 10)	☑
colour	nvarchar(50)	☑
list_price	decimal(18, 10)	☑
size	int	☑
dealer_price	decimal(18, 10)	☑
model_name	nvarchar(50)	☑
details	nvarchar(250)	☑
product_start_date	date	☐
product_end_date	date	☑
bk_supplier	nvarchar(50)	☑
supplier_name	nvarchar(50)	☑
transportation_fees	int	☐

Figure viii Dimension Product table

Dimension Territory

The territory table portrays the regions that the company actively operates in. This provides with the necessary information pertaining to which region has more sales in the comparison to the rest. This helps with managing inventory based on the sales region. The dimensions in this table like bike lanes and protected bike lanes provide us with precursors to expanding in newer regions based on these indicators.

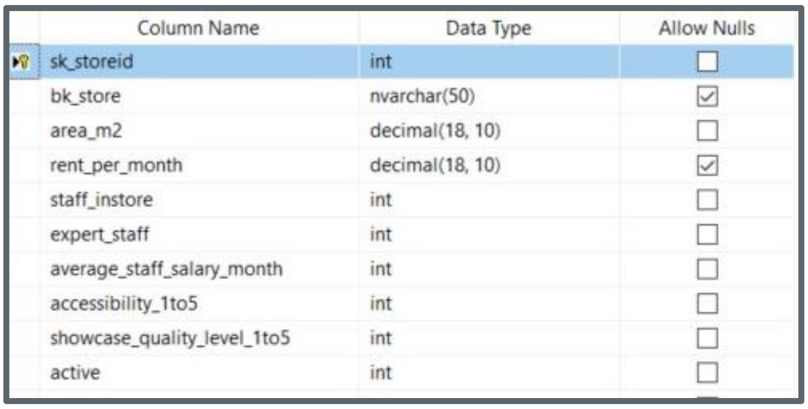


Column Name	Data Type	Allow Nulls
sk_territoryid	int	☐
bk_territorysubcategory	nvarchar(50)	☑
bk_territorykey	nvarchar(50)	☑
region	nvarchar(50)	☑
group_division	nvarchar(50)	☑
country	nvarchar(50)	☑
city_name	nvarchar(50)	☑
bike_lanes_km	decimal(18, 10)	☐
protected_bike_lanes_km	decimal(18, 10)	☐
		☐

Figure ix Dimension Territory table

Dimension Store

The store table functions as a monitoring and managing the individual stores containing the relevant information about the store whilst also showcasing the costs associated with operating store (salaries and rent). It describes the scope and size of the store by rating the showcase quality. These dimensions greatly aid the company in making decisions for individual stores.

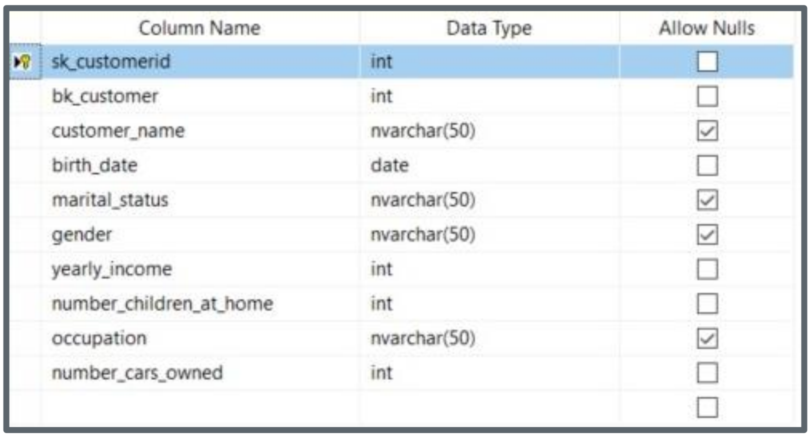


Column Name	Data Type	Allow Nulls
sk_storeid	int	☐
bk_store	nvarchar(50)	✓
area_m2	decimal(18, 10)	☐
rent_per_month	decimal(18, 10)	✓
staff_instore	int	☐
expert_staff	int	☐
average_staff_salary_month	int	☐
accessibility_1to5	int	☐
showcase_quality_level_1to5	int	☐
active	int	☐

Figure x Dimension Store table

Dimension Customer

The customer table describes the client who places the order with the dimensions that are needed to complete the order whilst describing the client itself (numbers of cars owned, number of children) which provides valuable information for future marketing strategies.

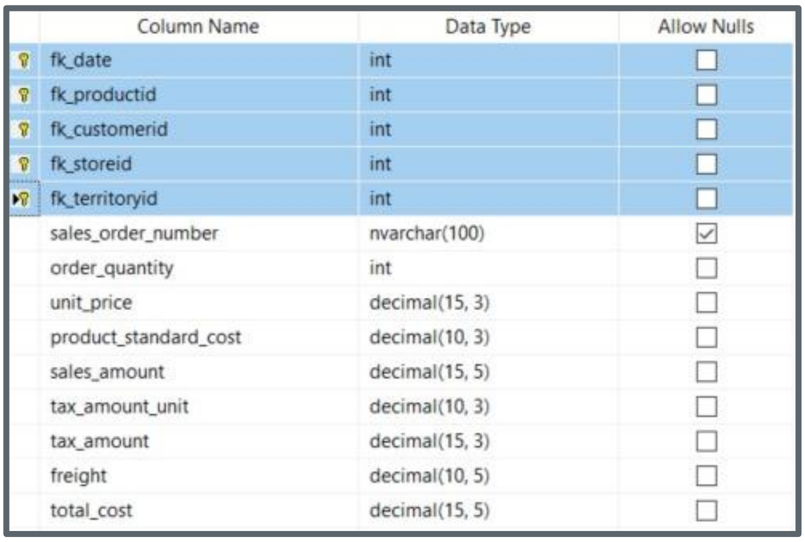


Column Name	Data Type	Allow Nulls
sk_customerid	int	☐
bk_customer	int	☐
customer_name	nvarchar(50)	✓
birth_date	date	☐
marital_status	nvarchar(50)	✓
gender	nvarchar(50)	✓
yearly_income	int	☐
number_children_at_home	int	☐
occupation	nvarchar(50)	✓
number_cars_owned	int	☐
		☐

Figure xi Dimension Customer table

Dimension Sales

The fact sales table is the most comprehensive table in the database as it provides most of the actionable evidence in terms of analysing sales and profit. This table has foreign keys that link it to other tables which paints the full picture of the company's growth/decline.



Column Name	Data Type	Allow Nulls
fk_date	int	☐
fk_productid	int	☐
fk_customerid	int	☐
fk_storeid	int	☐
fk_territoryid	int	☐
sales_order_number	nvarchar(100)	✓
order_quantity	int	☐
unit_price	decimal(15, 3)	☐
product_standard_cost	decimal(10, 3)	☐
sales_amount	decimal(15, 5)	☐
tax_amount_unit	decimal(10, 3)	☐
tax_amount	decimal(15, 3)	☐
freight	decimal(10, 5)	☐
total_cost	decimal(15, 5)	☐

Figure xii Dimension Sales table

Data warehouse schema

We employed the Kimball-methodology for denormalizing data to optimize the steps of queries which results in simplified monitoring of business reports whilst also boasting significant performance improvements. This sort of data modelling is considered the industry standard and is used by many different companies. Simplifying the data results in faster query searches and makes the data much more understandable for non-technical users. A star schema makes the data expansible for the future.

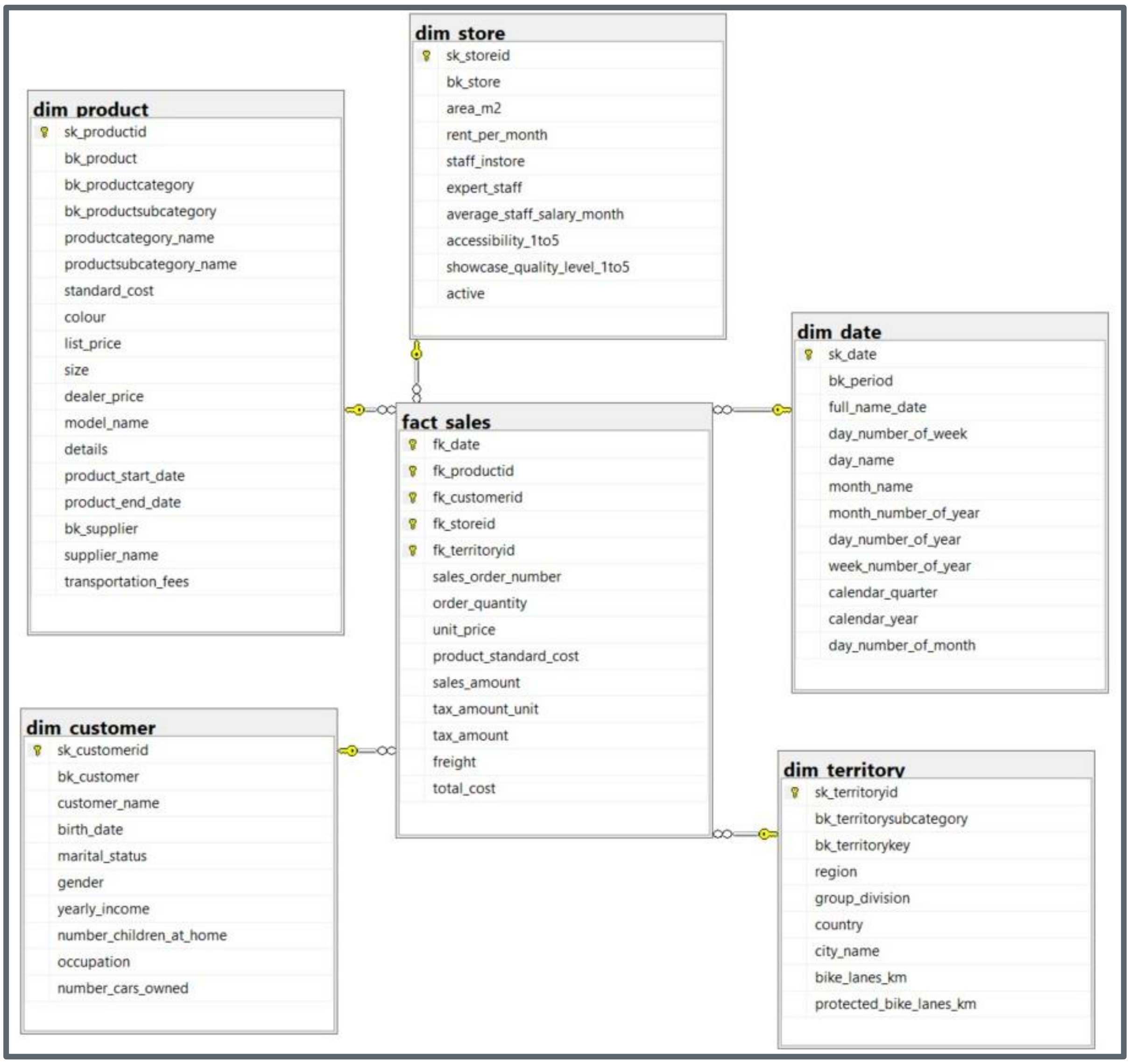


Figure xiii Data Warehouse schema

Hierarchies

Hierarchies are maps for data that go from the lowest level to the highest level and typically range from three to five levels. The creation of these maps helps on searching and analysing the data in a more optimal way. At KGB Data Warehouse we set these hierarchies:

- Date Hierarchy
- Product Hierarchy
- Customer Hierarchy
- Store Hierarchy
- Territory Hierarchy

Date Hierarchy

The Date dimension hierarchy will have six levels of depth. Having the Year at the highest level and the Full name date at the lowest level dimension. If we consider the inconsistent levels of depth between the branches, we could say that this is an unbalanced hierarchy.

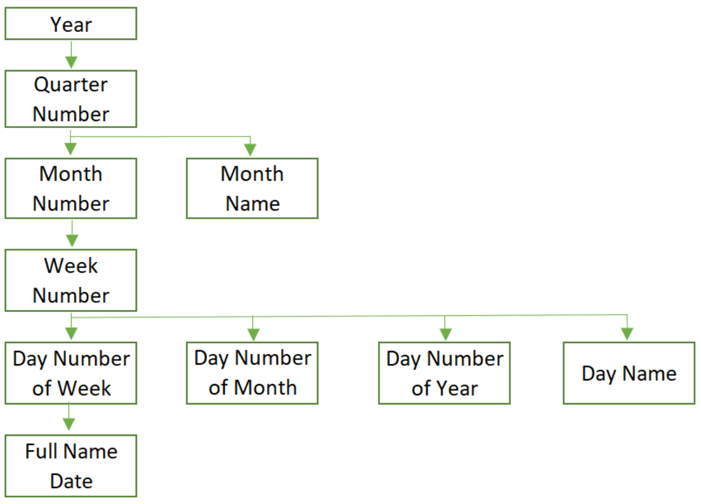


Figure xiv Date hierarchy

Product Hierarchy

For the Product dimension hierarchy will have five levels of depth. Having the Supplier at the highest level and the Model Name at the lowest level, which is divided in Price, Cost, Details, Size and Colour. We can observe in the figure below, that this is a balanced hierarchy, because the branches are descending to the same level.

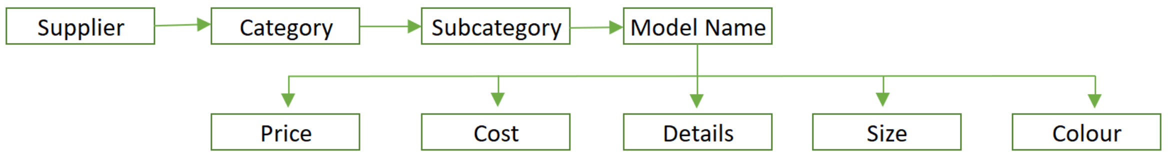


Figure xv Product hierarchy

Customer Hierarchy

The Customer dimension hierarchy will have five levels of depth. At the highest level we have Gender and at the lowest level dimension we have four different attributes that are Occupation, Birth Date, Yearly Income and Customer name. We could say for the figure below that this is a balanced hierarchy.

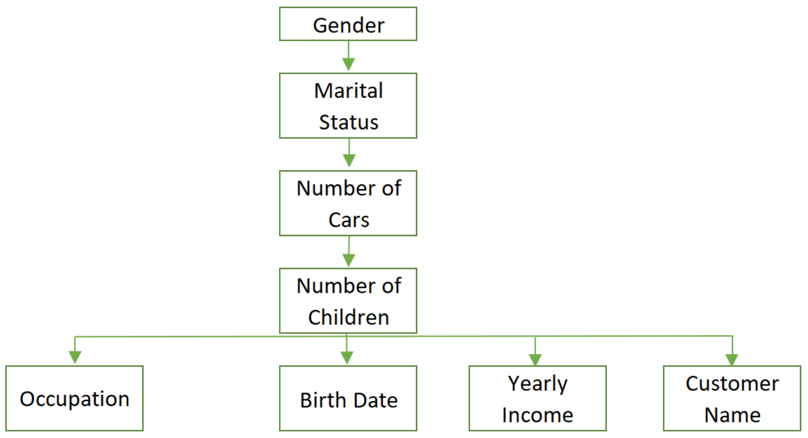


Figure xvi Customer hierarchy

Store Hierarchy

As we can see in the figure below, Store dimension hierarchy will have four levels, the highest level will be Accessibility and the lowest level dimension will be Store Code. At level three we have five different attributes that are Area, rent per Month, Staff in Store, Expert Staff and Average Salary. Because of that this is an unbalanced hierarchy.

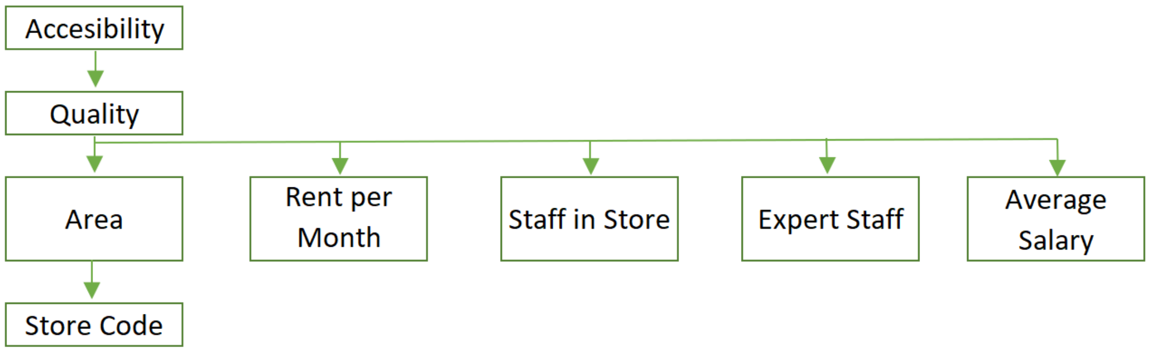


Figure xvii Store hierarchy

Territory Hierarchy

For the Territory dimension hierarchy, we will have a depth of five levels. At the highest level we have the Group and at the lowest level dimension we have City that will be divided into Bike Lanes (km) and Protected Bike Lanes (km). As we can observe in the figure below, this is a balanced hierarchy.

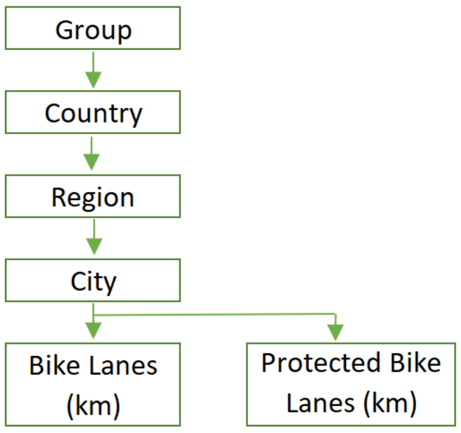


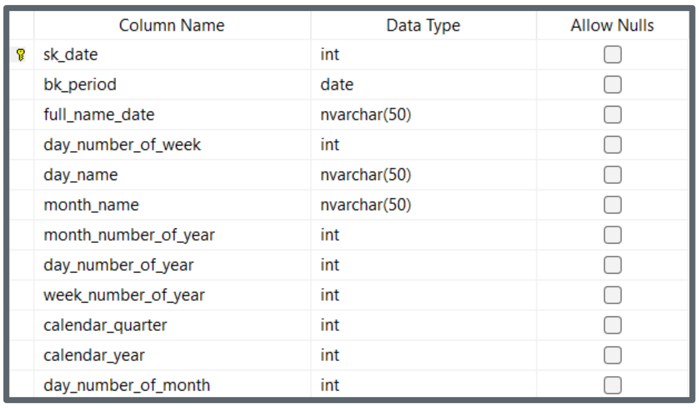
Figure xviii Territory hierarchy

Critical Changes Made After Feedback

We revaluated the code from the first phase of the report and made changes to make the staging area and loading of the data warehouse much efficient and terminologically sound.

The changes we made and why we made them are as follows:

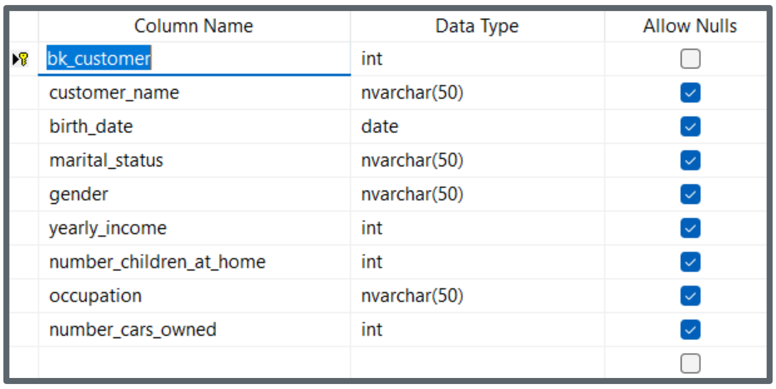
1. There were significant changes made to the code of the Data warehouse
    - We made sure that the dim date table had no attributes that could be null



	Column Name	Data Type	Allow Nulls
🔑	sk_date	int	☐
	bk_period	date	☐
	full_name_date	nvarchar(50)	☐
	day_number_of_week	int	☐
	day_name	nvarchar(50)	☐
	month_name	nvarchar(50)	☐
	month_number_of_year	int	☐
	day_number_of_year	int	☐
	week_number_of_year	int	☐
	calendar_quarter	int	☐
	calendar_year	int	☐
	day_number_of_month	int	☐

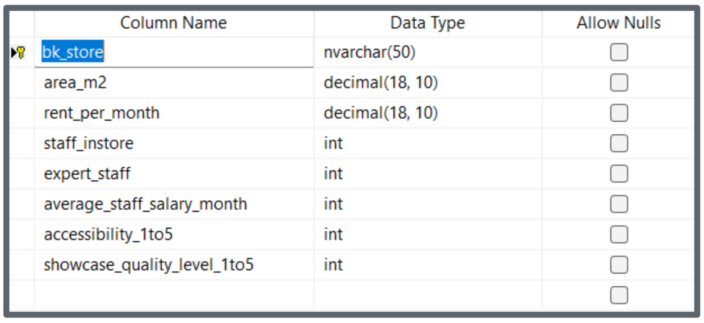
Figure xix Dim_date repolished

    - We looked at the product, store, territory, and customer tables and changed some attributes to null and some to not null depending on the data, and we used identity to generate surrogate keys.



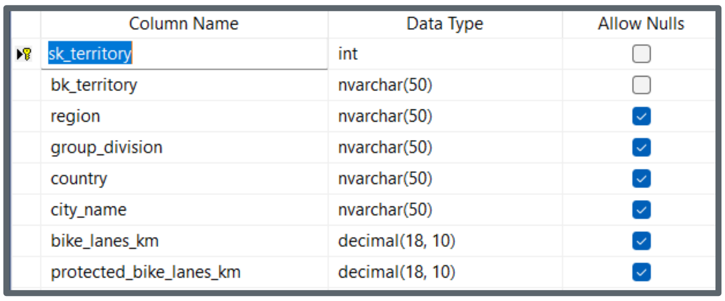
	Column Name	Data Type	Allow Nulls
▶🔑	bk_customer	int	☐
	customer_name	nvarchar(50)	☑
	birth_date	date	☑
	marital_status	nvarchar(50)	☑
	gender	nvarchar(50)	☑
	yearly_income	int	☑
	number_children_at_home	int	☑
	occupation	nvarchar(50)	☑
	number_cars_owned	int	☑
			☐

Figure xx Dim_customer repolished



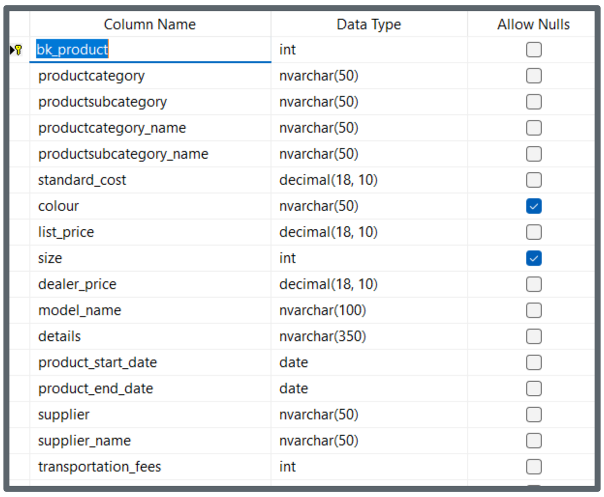
Column Name	Data Type	Allow Nulls
bk_store	nvarchar(50)	☐
area_m2	decimal(18, 10)	☐
rent_per_month	decimal(18, 10)	☐
staff_instore	int	☐
expert_staff	int	☐
average_staff_salary_month	int	☐
accessibility_1to5	int	☐
showcase_quality_level_1to5	int	☐
		☐

Figure xxi Dim_store repolished



Column Name	Data Type	Allow Nulls
sk_territory	int	☐
bk_territory	nvarchar(50)	☐
region	nvarchar(50)	☑
group_division	nvarchar(50)	☑
country	nvarchar(50)	☑
city_name	nvarchar(50)	☑
bike_lanes_km	decimal(18, 10)	☑
protected_bike_lanes_km	decimal(18, 10)	☑

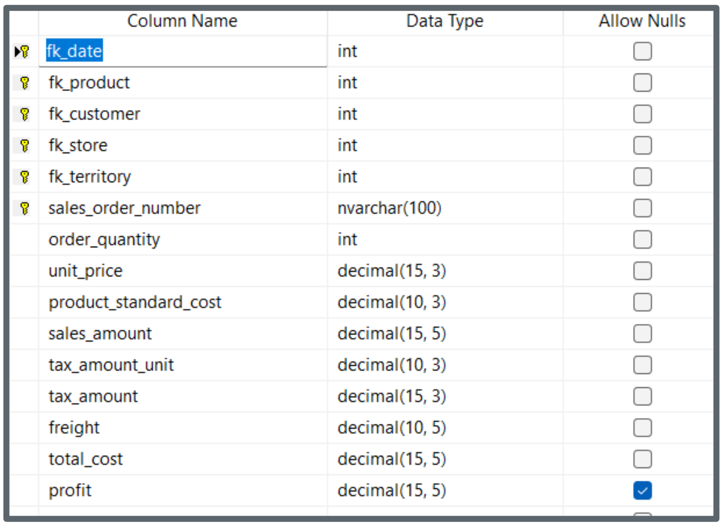
Figure xxii Dim_territory repolished



Column Name	Data Type	Allow Nulls
bk_product	int	☐
productcategory	nvarchar(50)	☐
productsubcategory	nvarchar(50)	☐
productcategory_name	nvarchar(50)	☐
productsubcategory_name	nvarchar(50)	☐
standard_cost	decimal(18, 10)	☐
colour	nvarchar(50)	☑
list_price	decimal(18, 10)	☐
size	int	☑
dealer_price	decimal(18, 10)	☐
model_name	nvarchar(100)	☐
details	nvarchar(350)	☐
product_start_date	date	☐
product_end_date	date	☐
supplier	nvarchar(50)	☐
supplier_name	nvarchar(50)	☐
transportation_fees	int	☐

Figure xxiii Dim_product repolished

- Corrected the specified constraints and added sales_order_number to the constraints.



Column Name	Data Type	Allow Nulls
fk_date	int	☐
fk_product	int	☐
fk_customer	int	☐
fk_store	int	☐
fk_territory	int	☐
sales_order_number	nvarchar(100)	☐
order_quantity	int	☐
unit_price	decimal(15, 3)	☐
product_standard_cost	decimal(10, 3)	☐
sales_amount	decimal(15, 5)	☐
tax_amount_unit	decimal(10, 3)	☐
tax_amount	decimal(15, 3)	☐
freight	decimal(10, 5)	☐
total_cost	decimal(15, 5)	☐
profit	decimal(15, 5)	✓

Figure xxiv Repolished fact_sales

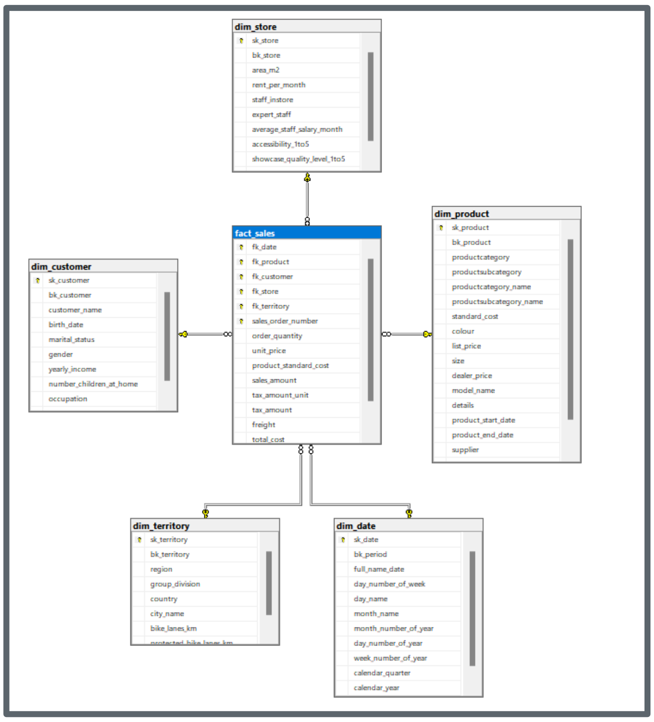


Figure xxv Repolished Data Warehouse

2. We improved the documentation of the methodology of the established data warehouse design.

Extract, Transform and Load (ETL) Process

Our primary injection of data pertaining sales, customers, region, etc. came from SQL management studio and all information was imported as a CSV comma delimited flat file.

CSV files are commonly used in large databases because of their efficiency when it comes to read and write speeds since they are only plain text files whilst also being easy to edit. This allows for faster load times, but they also have some problems when it comes to complex data but according to our judgement, we decided to use CSV comma delimited files because our dataset felt like it could benefit from it.

ETL process was divided into 3 steps:

1. Load the data into the Staging Area.
2. Load the Staging Area to the Data Warehouse.
3. Execution of Staging Area and Data Warehouse.

First step involves the extraction of data followed by transformation. Data is then loaded into the Data Warehouse as the last step.

ETL - Staging Area

In the architecture of Data Warehouse, the staging area is where data is copied from the operating systems. It is a transformational process where data is cleansed and merged before it could be loaded into the Data Warehouse. Staging Area provides flexibility during the process of extraction and loading of data which increases the efficiency and ensures the data integrity. The design of Staging Area and Data Warehouse structure must match to achieve maximum efficiency which reduces the need for any future modifications.

To sum up the staging area is a intermediate repository where data gets itself in the best possible shape. It also:

1. Serves as a point for data correction
2. Acts as a separation from the DW because it presents the possibility of uploading it in a more controlled timeline.

We had several data types such as floats, data, currency, date, integers, strings, etc. The number of these data types had to be reduced and converted into data types like nvarchar, date, integer which are more SQL friendly. (Condensed multiple data types to 3 to improve efficiency and speed).

Methodology

The ETL staging area is a critical component of the ETL process because it allows for the separation of the extracting, transforming, and loading of the data. This separation ensures that the data is clean and accurate before it is loaded into the destination system. In addition, the ETL staging area provides a temporary storage location for the data, which can be useful when dealing with large volumes of data or when the data needs to be accessed by multiple users or processes.

In general, the ETL staging area should be designed to be scalable, secure, and efficient to ensure that it can support the needs of the organization and the ETL process.
The methodology for using a staging area in the ETL process typically involves the following steps:

1. Extract: The data is extracted from its source and moved to the staging area. This step may involve using extraction tools and techniques, such as SQL queries or APIs, to retrieve the data from the source system.

2. Transform: The data in the staging area is then transformed to fit the requirements of the destination system. This step may involve cleaning and formatting the data, as well as applying any necessary transformations or calculations.

3. Load: The transformed data is then loaded into the destination system. This step may involve using load tools and techniques, such as SQL INSERT statements or bulk load operations, to transfer the data from the staging area to the destination system.

4. Verify: After the data has been loaded into the destination system, it is typically verified to ensure that it has been loaded correctly and that it meets the requirements of the destination system. This step may involve comparing the data in the destination system to the original data in the source system or running tests to ensure that the data is accurate and complete.

By following this methodology, we can use a staging area to manage the ETL process effectively and efficiently and ensure the accuracy and integrity of the data as it is transferred from its source to the destination system.

Loading the ETL Package

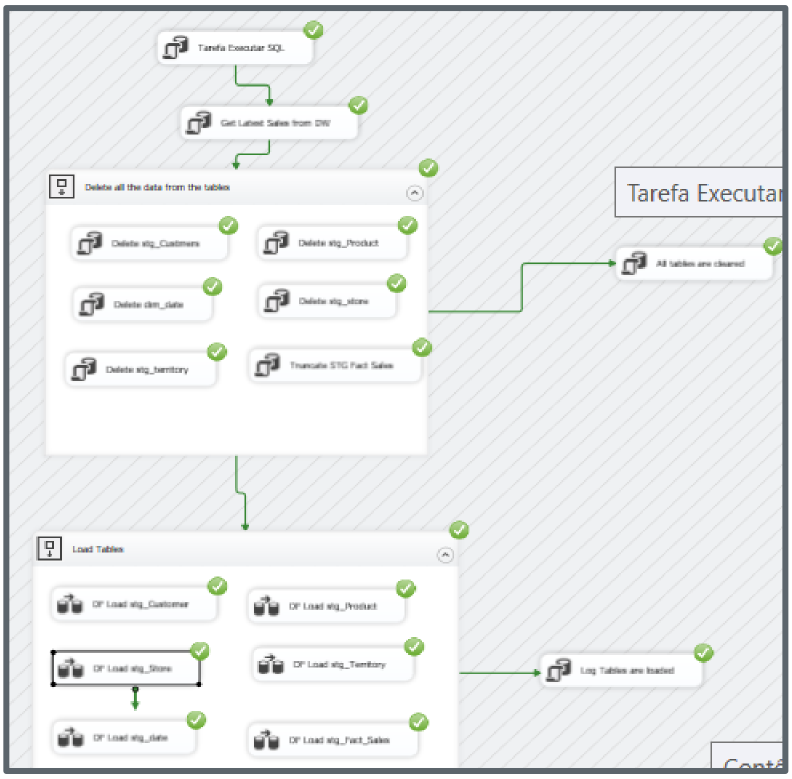
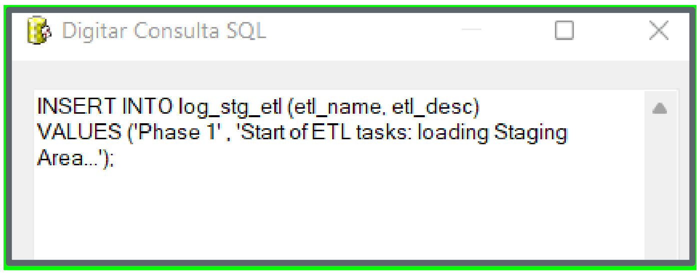


Figure xxvi Staging Area Loading_ Full process

We start the ETL package by identifying the data that we wanted to extract from its source – an Excel spreadsheet. After this step, we decided to convert the data type to a CSV file.

We have logged the start runtime in the table log_stg_etl using an Execute SQL Task. We have done this by first creating two new variables, 'ETL_Name' to get the current system time, and 'ETL_Desc' to get the description of the task. These values are then stored on the Log Table to keep records of all the times the process started as shown below.



```
INSERT INTO log_stg_etl (etl_name, etl_desc)
VALUES ('Phase 1' , 'Start of ETL tasks: loading Staging
Area...');
```

Figure xxvii One of log_stg_etl

Staging Area Table Data Clearing

The following sequence container deletes all previous data in the staging area. This is a crucial step as it helps to maintain the integrity, performance, and simplicity of the process. To clear the previous data in the staging area, we employ two methods: DELETE for the dimension tables and TRUNCATE for the fact table. As there are no relationships between the tables at this point in the project, they can all be erased simultaneously.

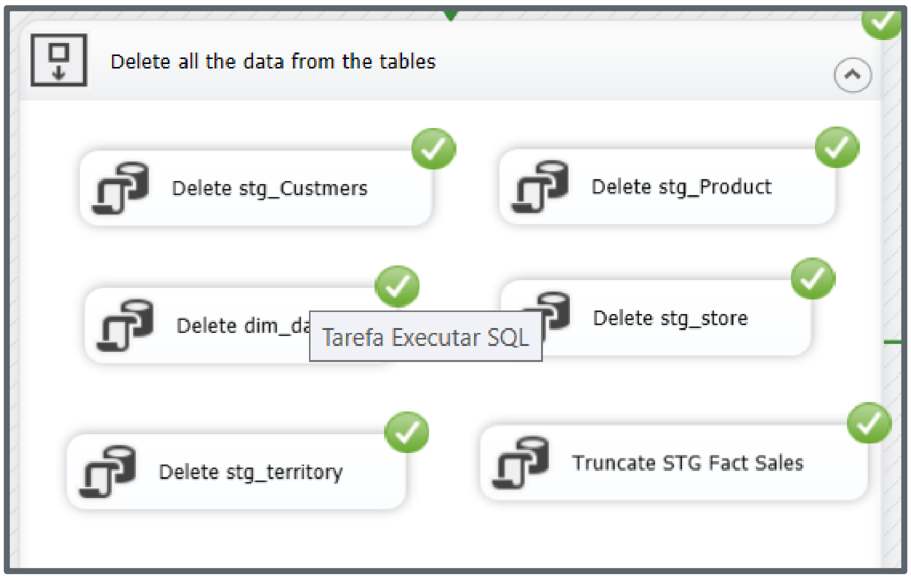


Figure xxviii Deleting Container for Staging Area

To delete the information in the dimension tables, we use an Execute SQL Task with the code "DELETE FROM 'Dimension'". For the fact tables, we utilize the TRUNCATE function, which is a faster DDL (Data Definition Language) statement that cannot be undone, in contrast to the DELETE function which is a DML (Data Modification Language) statement that logs each row that is deleted and can be reversed. We chose to use the TRUNCATE function for the fact tables due to its ability to improve performance when deleting many rows, as was the case with the stg_fact_sales table.

We did not delete any information from the LOG_Stg_ETL table as we desire to keep a record of all the steps taken in our workflow. These records will not be transferred to the data warehouse.

Staging Area Dimensions - Data Loading

We created a new sequence container to organize all the completed loading processes for the tables in the staging area. At this stage of the project, there were no relationships between the tables, allowing us to load them simultaneously, as shown below. However, it was still important to ensure that the data was properly transformed and cleaned according to the specific requirements of each table. This required careful attention to detail and adherence to best practices. By fulfilling these requirements, we completed the loading processes and progressed to the next phase of the project.

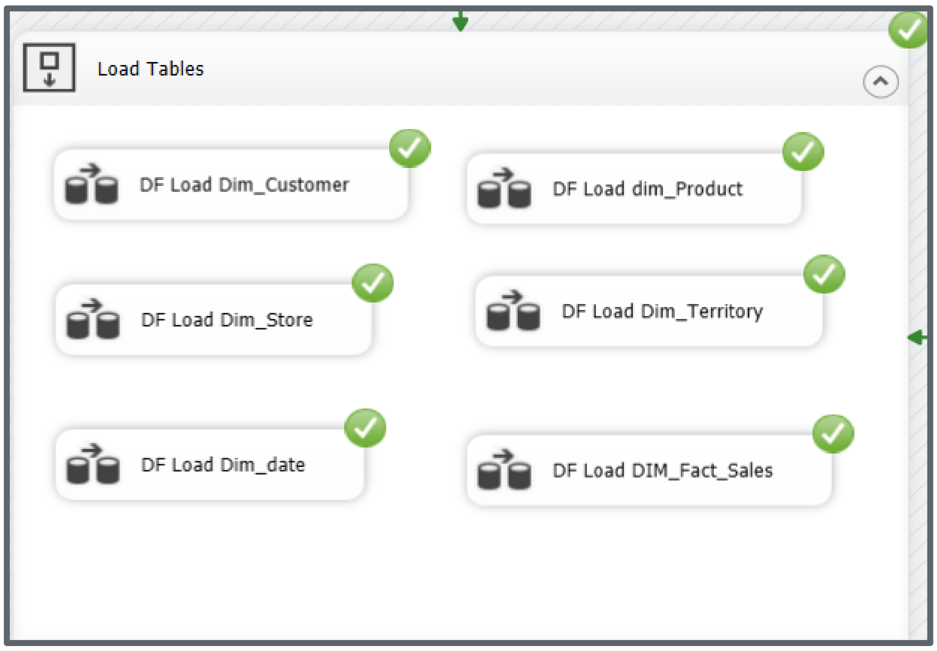


Figure xxix Loading Container for Staging Area

Loading stg_Customer

The process of importing the KGB's customer dimension was made easier by the shared structure and naming conventions between the database and the KGB's table. Mapping the input columns, as shown below, with the destination columns is simple as they all share the same name.

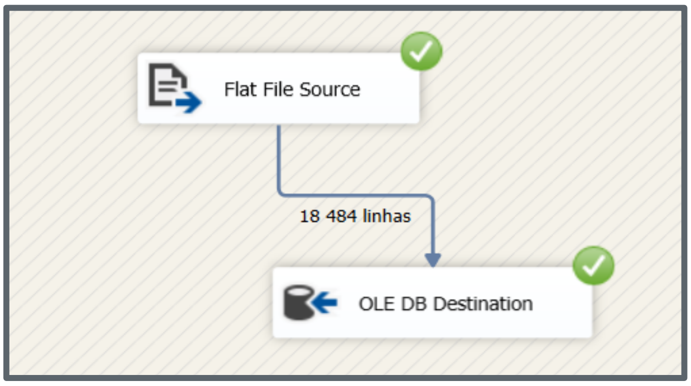


Figure xxx loading stg_Customer

Loading stg_Product

The process of loading the KGB's product dimension into the system was a Type 0 process. One of the challenges we faced during this process was mapping the input columns, as shown below. There were some flaws in the data type conversions that needed to be addressed, which required careful attention to detail and a thorough understanding of the data to ensure that the columns were properly aligned, and any errors were corrected. Despite these challenges, we were able to successfully load the product dimension and move on to the next phase of the project.

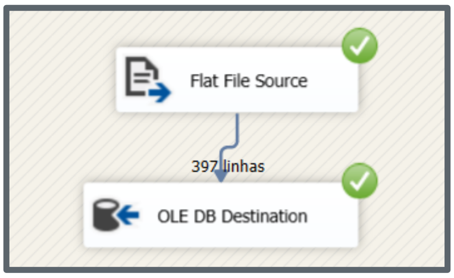


Figure xxxi Loading stg_Product

Loading stg_Store

Loading the KGB's Store dimension was also Type 0 process. Mapping the input columns, as shown below, was challenging once there were some flaws in the data types conversions.

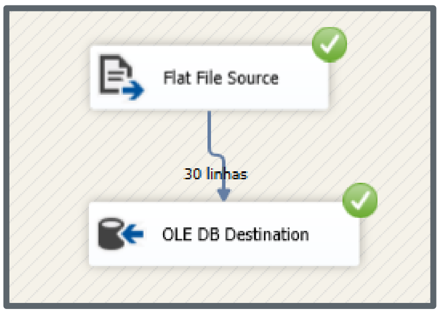


Figure xxxii Loading stg_Store

Loading stg_Territory

Loading the KGB's Territory dimension was also a Type 0 process.

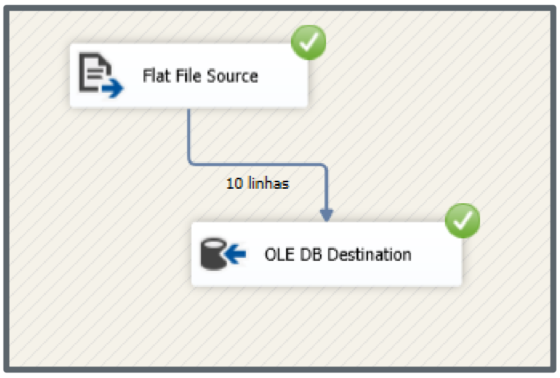


Figure xxxiii Loading stg_Territory

Loading stg_Date

Loading the KGB's Date dimension was also a Type 0 process.

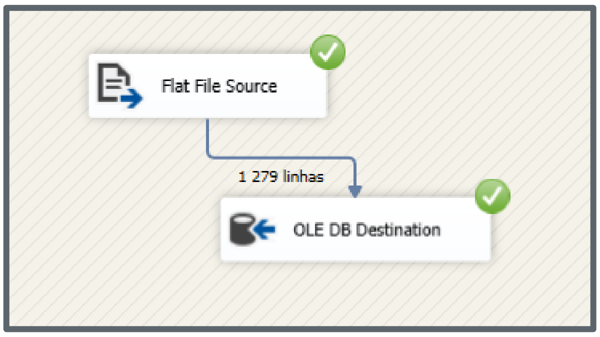


Figure xxxiv Loading stg_Date

Loading Dim_Fact_Sales

Loading the KGB's Fact Sales staging dimension was a different process: we had to ensure ways of doing the desired incremental loading. We achieved this by the usage of a *Conditional Split* after applying a *Data Conversion.*

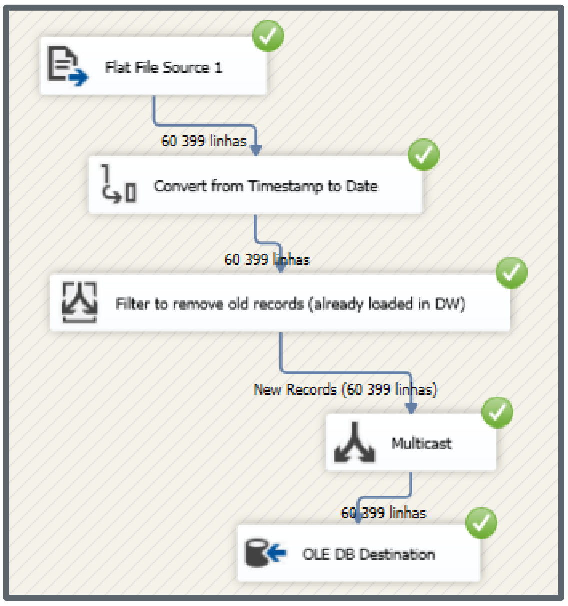

Flat File Source 1
60 399 linhas
Convert from Timestamp to Date
60 399 linhas
Filter to remove old records (already loaded in DW)
New Records (60 399 linhas)
Multicast
60 399 linhas
OLE DB Destination

Figure xxxv stg_fact_sales loading

ETL – Loading the Data Warehouse

At the time the data is polished and reshaped within the Staging Area, the process of loading it into the Data Warehouse may go ahead. This stage of the assembly of the Business Intelligence project is where data truly becomes information. Although it had already gone through the procedures described in the previous chapter, only now the knowledge will be properly stored so it can be a resourceful helping tool to KGB: the Data Warehouse would now be able to start working. Undesirably, we were not able to properly load the *fact_table*. We will later cover this topic in the conclusion.

In this segment we describe all the operations that took place to get the data loaded to the rest of the data warehouse tables.

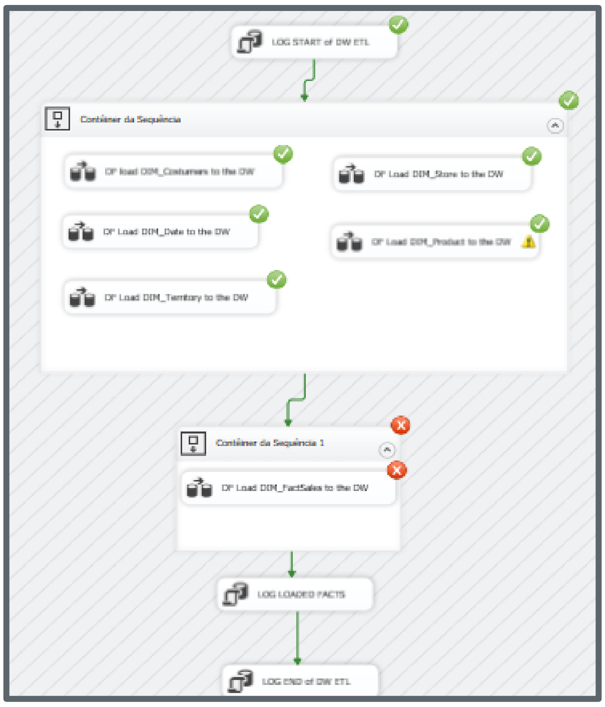

LOG START of DW ETL
Contêiner da Sequência
DF load DIM_Costumers to the DW
DF Load DIM_Store to the DW
DF Load DIM_Date to the DW
DF Load DIM_Product to the DW
DF Load DIM_Territory to the DW
Contêiner da Sequência 1
DF Load DIM_FactSales to the DW
LOG LOADED FACTS
LOG END of DW ETL

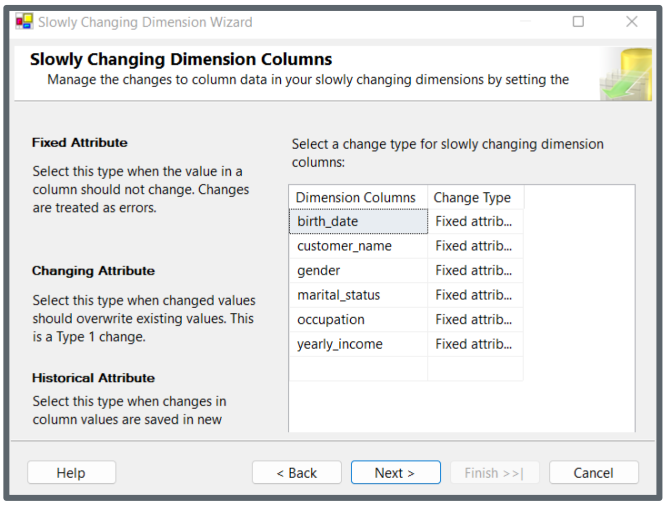
Figure xxxvi Data Warehouse Load

Approach on the Slowly Changing Dimensions (SCD)

To begin, we gathered information from the Staging Area and linked each Data Warehouse dimension with its corresponding dimension in the SA for all dimension tables. After this information was collected, we set up Slowly Changing Dimensions (SCD) for each dimension and specified the SCD types for each attribute. If we are performing an incremental data load, it is necessary to ensure that any existing data is updated without causing an error (type 1 SCD) while also keeping track of these changes (type 2 SCD).

Loading Dim_Customer

Regarding the Dim_Customer data flow, we've configured the birth_date, customer_name, gender, maritial_status, occupation and yearly_income attributes as SCD type 0 (Fixed Attribute), because we believe that new data on these attributes should overwrite the existing data in the dimension table with new data from the source system. We have so on considered that the source system is the primary source of truth for the data and any changes made to the data in the dimension table should be overwritten by the source system. We also believe that the data in the source system is clean and does not require any further transformation or cleansing before being loaded into the dimension table.


Slowly Changing Dimension Wizard

Slowly Changing Dimension Columns

Manage the changes to column data in your slowly changing dimensions by setting the

Fixed Attribute

Select this type when the value in a column should not change. Changes are treated as errors.

Changing Attribute

Select this type when changed values should overwrite existing values. This is a Type 1 change.

Historical Attribute

Select this type when changes in column values are saved in new

Select a change type for slowly changing dimension columns:

Dimension Columns	Change Type
birth_date	Fixed attrib...
customer_name	Fixed attrib...
gender	Fixed attrib...
marital_status	Fixed attrib...
occupation	Fixed attrib...
yearly_income	Fixed attrib...

Figure xxxvii Loading Customer

Loading Dim_Product

Looking at the Products Dimension, our decision was to overwrite existing values in supplier_name (making it a type 1 SCD) and to save all the values regarding dealer_price, standard_cost and transportation_fees by defining the change type as SCD 2 (Historical Attribute). We consider that it's relevant to keep historical track of changes in these dimensions as shown below.

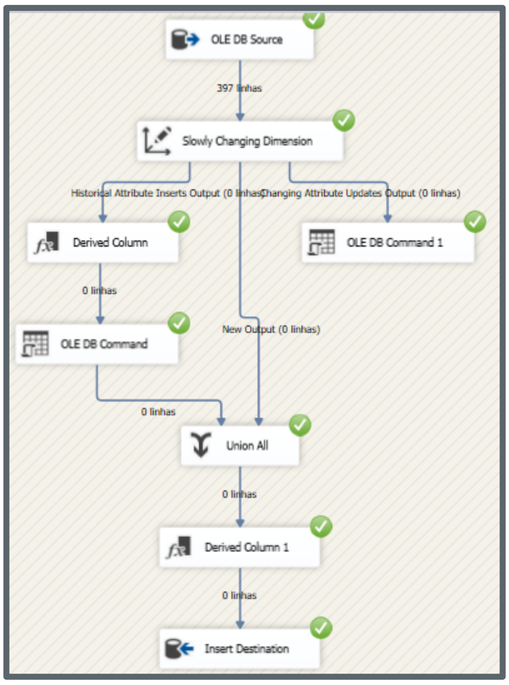


Figure xxxviii Loading Dim_Product

Loading Dim_Store

Regarding the Dim_Store data flow, we've configured the accessibility_1to5, active, area_m2, average_staff_salary, expert_staff, rent_per_month and staff_instore attributes as SCD type 0 (Fixed Attribute), because we believe that the data in the source system is clean and does not require any further transformation or cleansing before being loaded into the dimension table.

Loading Dim_Date

Regarding the Dim_Date data flow, we've configured all attributes as SCD type 0 (Fixed Attribute).

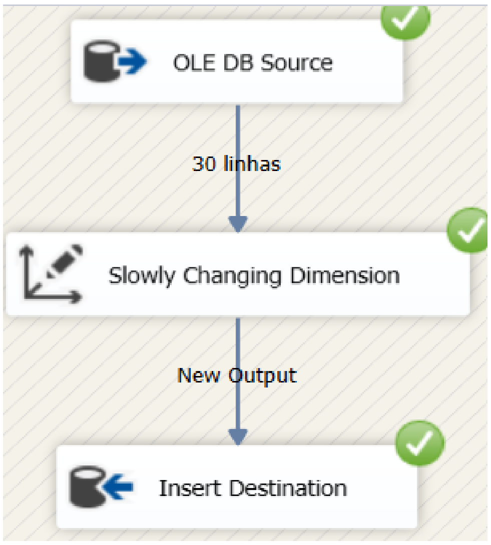
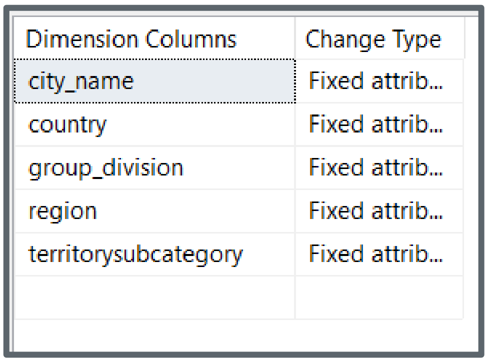


Figure xxxix Loading Dim_Store

Loading Dim_Territory

Regarding the Dim_Territory data flow, we've configured all attributes as SCD type 0 (Fixed Attribute).



Dimension Columns	Change Type
city_name	Fixed attrib...
country	Fixed attrib...
group_division	Fixed attrib...
region	Fixed attrib...
territorysubcategory	Fixed attrib...

Figure xl Loading Dim_Territory

Loading Fact Sales

In the process of loading Fact Sales, we could not achieve the desired final outcome: the full load of the Fact Sales dimension.

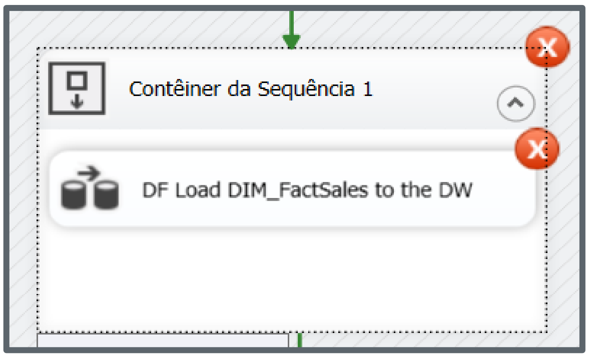


Figure xli Loading Fact Sales

Conclusion

KGB had some challenges with the way their data was stored and had severe shortcomings in optimization. After analysing the data and assessing the needs in the context of the business, we designed a solution for a Data Warehouse.

We extensively went over the literature provided to us during our classes to question the validity of the data warehouses employed and after significant amount of troubleshooting, we reached a significant point where we were satisfied by our design implementation of our Data warehouse and ETL processes.

As there are multiple ways to solve a problem when it comes to the transformational side of data handling, we had to rely on a trial-and-error methodology to assess which solution was the most fitting for our design. The staging area proved to be very beneficial and showcased its true importance before the data was loaded into the warehouse because of the abundance of data types and formats.

Concerning to a business point of view:

In business intelligence the dimensional model is way of organizing data that makes it easy to understand and analyze. One key aspect of the dimensional model is the use of dimensions, which are attributes or categories that can be used to slice and dice the data in different ways.

The inflection points of most queries in a business-related system are related to sales and we specifically worked on getting that streamlined. We opted for 4 dimensions that all lead to shining a light on what we consider to be the most valuable insight according to our aforementioned business questions. Customers, store, territory, and product accentuate the sales data.

For instance, we recalled the business questions related to the biggest sales by market and date; the fact table should be able to access the customers info to retrieve the birth date. The same stands for “getting to know our client profile by product, product category and product sub-category”.

We must also remember that the data’s natural granularity obeys a certain order of thought, and we must not stray from it too much, to keep the model’s interpretation to the user.

Lastly, date dimension in a business intelligence dimensional model is fixed and can be populated with a set of valued dates, allowing for an easier pre-aggregation and faster querying within the data warehouse. This also emanates the need to check the validity of dates during processing. For that reason, we decided to keep that separate from the rest of the dimensions.

Analytic value proposition for the business

1. Improve data access – By combining all our different source files (flat files), we managed to centralize and integrate numerous aspects of KGB’s business, aiding decision makers in streamlining the process of knowledge acquirement.

2. Increase Data Quality – Through the implementation of the staging area, KGB analysts will be much more certain of dealing with meaningful data whilst also reducing the risk of errors in their analysis.

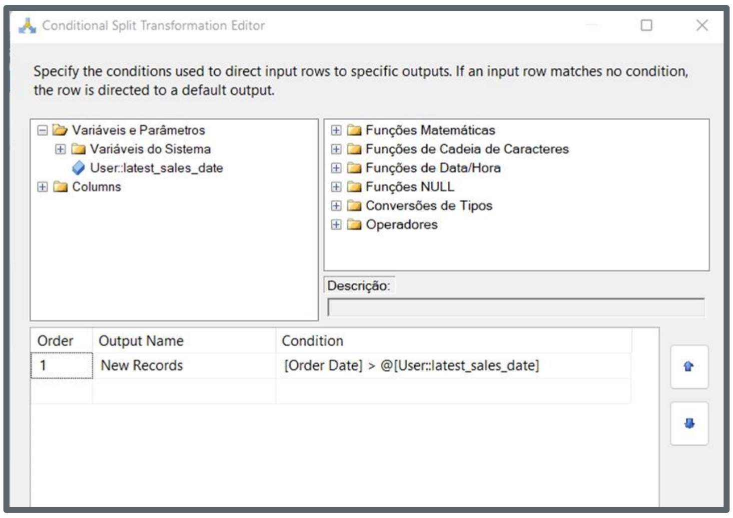
3. Better Performance – The star schema optimizes the coverage of all business aspects. For this reason, the data warehouse is optimized for reporting and analysis which means that queries will run faster, and assistant can handle larger volumes of data. Merging different dimensions to provide a bigger overall picture whilst optimizing the speed as well.

4. Improve analysis – By bringing up all the different business perspective and wrapping them in the data warehouse, KGB will be able to expand its scope on business behaviour.

5. Historical analysis – Through continuous data storage, KGB will be able to save the history of its sales. This can be useful for trend analysis, forecasting and identifying patterns in the data.
6. Data Governance – In the advent of bid data, cybersecurity plays a vital role in safeguarding the business' digital presence. Data warehousing provides a centralized system for managing and enforcing data security, compliance, and government policies, which is particularly important when dealing with sensitive data. In respects to KGB, there is a lot of data pertaining to customers that needs protective measures which is greatly improved through the implementation of data warehouses.

Critical Assessment

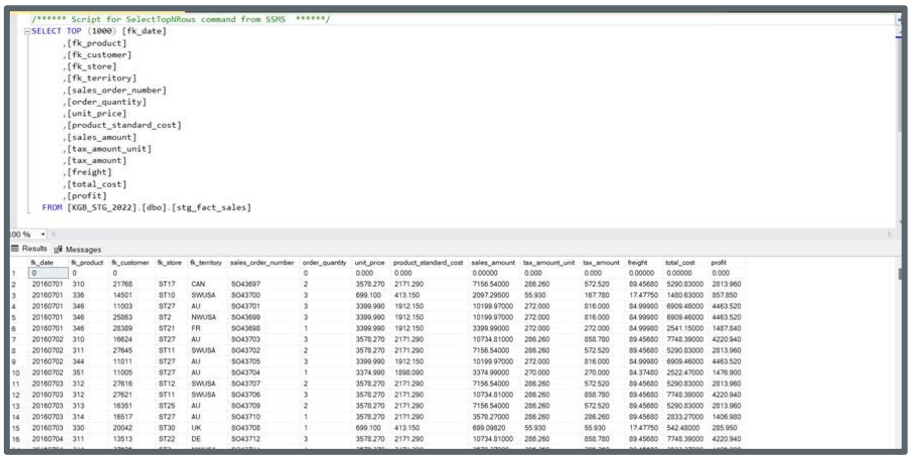
We were having trouble setting up the variable called "latest_sales_date" into the system variables. We searched online as to how we could solve this problem. As it is an important step in the incremental load automation part of the ETL process. We were then able to troubleshoot the problem and fixed the error.



Conditional Split Transformation Editor

Specify the conditions used to direct input rows to specific outputs. If an input row matches no condition, the row is directed to a default output.

Order	Output Name	Condition
1	New Records	[Order Date] > @[User::latest_sales_date]



```
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP (1000) [fk_date]
      ,[fk_product]
      ,[fk_customer]
      ,[fk_store]
      ,[fk_territory]
      ,[sales_order_number]
      ,[order_quantity]
      ,[unit_price]
      ,[product_standard_cost]
      ,[sales_amount]
      ,[tax_amount_unit]
      ,[tax_amount]
      ,[freight]
      ,[total_cost]
      ,[profit]
  FROM [KGB_STG_2022].[dbo].[stg_fact_sales]
```

	fk_date	fk_product	fk_customer	fk_store	fk_territory	sales_order_number	order_quantity	unit_price	product_standard_cost	sales_amount	tax_amount_unit	tax_amount	freight	total_cost	profit
1	0	0	0				0	0.000	0.000	0.00000	0.000	0.000	0.00000	0.00000	0.000
2	20160701	310	21768	ST17	CAN	SO43697	2	3578.270	2171.290	7156.54000	286.260	572.520	89.45680	5290.83000	2813.960
3	20160701	336	14501	ST10	SWUSA	SO43700	3	699.100	413.150	2097.29500	55.930	167.780	17.47750	1480.63000	857.850
4	20160701	346	11003	ST27	AU	SO43701	3	3399.990	1912.150	10199.97000	272.000	816.000	84.99980	6909.46000	4463.520
5	20160701	346	25863	ST2	NWUSA	SO43699	3	3399.990	1912.150	10199.97000	272.000	816.000	84.99980	6909.46000	4463.520
6	20160701	346	28389	ST21	FR	SO43698	1	3399.990	1912.150	3399.99000	272.000	272.000	84.99980	2541.15000	1487.840
7	20160702	310	16624	ST27	AU	SO43703	3	3578.270	2171.290	10734.81000	286.260	858.780	89.45680	7748.39000	4220.940
8	20160702	311	27645	ST11	SWUSA	SO43702	2	3578.270	2171.290	7156.54000	286.260	572.520	89.45680	5290.83000	2813.960
9	20160702	344	11011	ST27	AU	SO43705	3	3399.990	1912.150	10199.97000	272.000	816.000	84.99980	6909.46000	4463.520
10	20160702	351	11005	ST27	AU	SO43704	1	3374.990	1898.090	3374.99000	270.000	270.000	84.37480	2522.47000	1476.900
11	20160703	312	27616	ST12	SWUSA	SO43707	2	3578.270	2171.290	7156.54000	286.260	572.520	89.45680	5290.83000	2813.960
12	20160703	312	27621	ST11	SWUSA	SO43706	3	3578.270	2171.290	10734.81000	286.260	858.780	89.45680	7748.39000	4220.940
13	20160703	313	16351	ST25	AU	SO43709	2	3578.270	2171.290	7156.54000	286.260	572.520	89.45680	5290.83000	2813.960
14	20160703	314	16517	ST27	AU	SO43710	1	3578.270	2171.290	3578.27000	286.260	286.260	89.45680	2833.27000	1406.980
15	20160703	330	20042	ST30	UK	SO43708	1	699.100	413.150	699.09820	55.930	55.930	17.47750	542.48000	285.950
16	20160704	311	13513	ST22	DE	SO43712	3	3578.270	2171.290	10734.81000	286.260	858.780	89.45680	7748.39000	4220.940

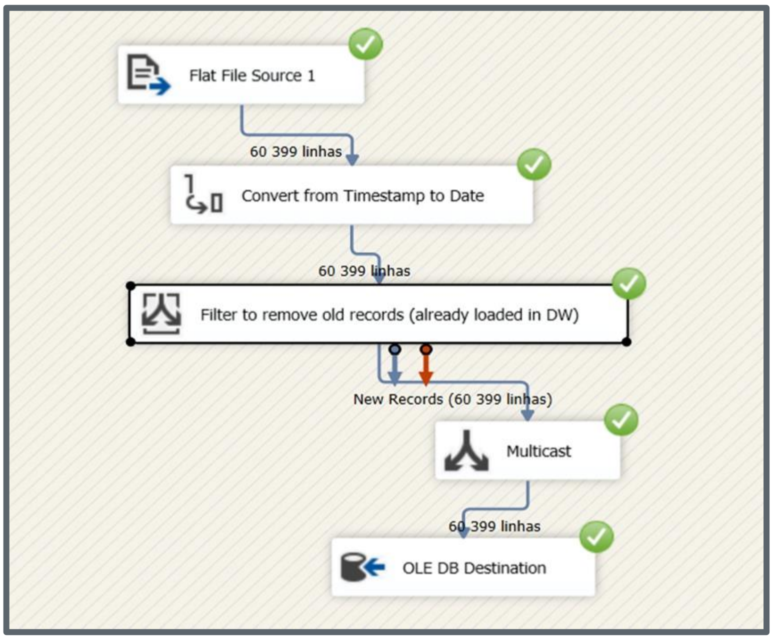
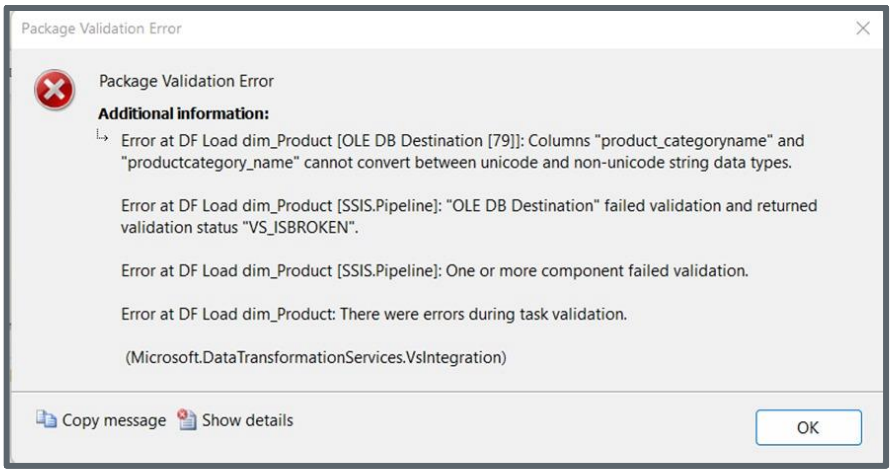

Flat File Source 1
60 399 linhas
Convert from Timestamp to Date
60 399 linhas
Filter to remove old records (already loaded in DW)
New Records (60 399 linhas)
Multicast
60 399 linhas
OLE DB Destination

Figure xlii Error 1 Figures

We got this error when loading the product dimension from the fact sales table .



Package Validation Error

Package Validation Error

Additional information:

Error at DF Load dim_Product [OLE DB Destination [79]]: Columns "product_categoryname" and "productcategory_name" cannot convert between unicode and non-unicode string data types.

Error at DF Load dim_Product [SSIS.Pipeline]: "OLE DB Destination" failed validation and returned validation status "VS_ISBROKEN".

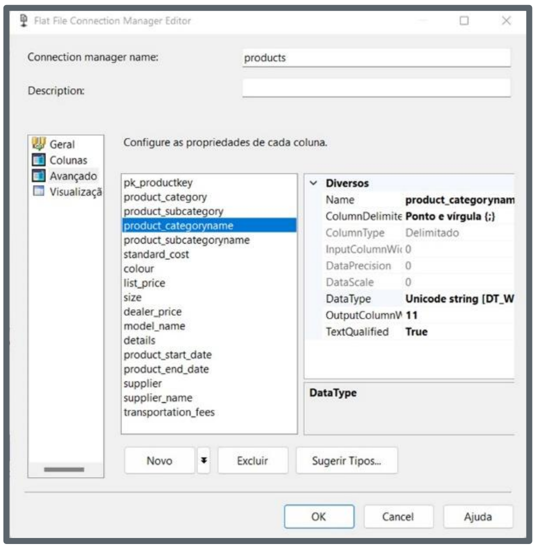
Error at DF Load dim_Product [SSIS.Pipeline]: One or more component failed validation.

Error at DF Load dim_Product: There were errors during task validation.

(Microsoft.DataTransformationServices.VsIntegration)

Figure xliii Error 2 figure 1

We were then forced to change the data type by going to the connection manager tab > Flat File Connection Manager Editor > Advanced and then changed the data type to Unicode String as shown below:


Flat File Connection Manager Editor

Connection manager name: products

Description:

Configure as propriedades de cada coluna.

```
pk_productkey
product_category
product_subcategory
product_categoryname
product_subcategoryname
standard_cost
colour
list_price
size
dealer_price
model_name
details
product_start_date
product_end_date
supplier
supplier_name
transportation_fees
```

Diversos	
Name	product_categorynam
ColumnDelimite	Ponto e vírgula {;}
ColumnType	Delimitado
InputColumnWi	0
DataPrecision	0
DataScale	0
DataType	Unicode string [DT_W
OutputColumnW	11
TextQualified	True

DataType

Figure xliv Error 2 figure 2

We faced the following error when we tried loading the fact sales table from the Staging Area to the Datawarehouse.

“[Lookup Costumer [2]] Error: Row yielded no match during lookup.”

“[LookupCostumer[2]]Error:SSISErrorCode DTS_E_INDUCEDTRANSFORMFAILUREONERROR.

The "Lookup Costumer" failed because error code 0xC020901E occurred, and the error row disposition on "Lookup Costumer.Outputs[Lookup Match Output]" specifies failure on error. An error occurred on the specified object of the specified component. There may be error messages posted before this with more information about the failure.”

We tried different approaches to solve it but with no avail. We have realized that this error message typically occurs when a SQL Server Integration Services (SSIS) package is using a Lookup transformation and the data being looked up does not have a match in the reference table. With that being said, we went through an analysis of our data, and it seemed that every key condition was set properly but the error persisted. One possible reason that we thought of was one row is not “picking” the right data and because of that, we keep getting this error repeatedly. We then tried to "Redirect rows to no match output" option in the lookup transformation to capture the rows that are not getting matched, and it gave us the following result:

With this option marked, there was yet no data loaded to the Data Warehouse. We noticed that in the end of the process only 9935 lines were having a match.

Additionally, we made sure that our input data was clean and didn’t have any null or empty value in join columns. Even after trying all the remedies we could think of, the error was not fixed.

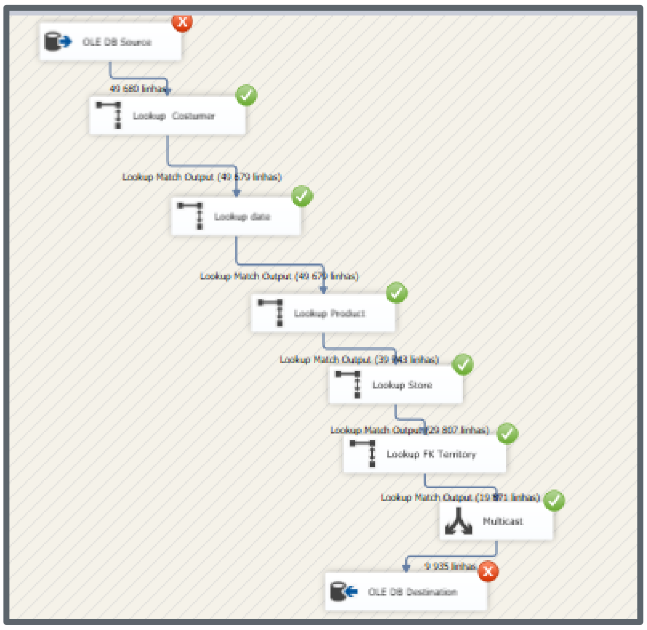

OLE DB Source
49 680 linhas
Lookup Costumer
Lookup Match Output (49 679 linhas)
Lookup date
Lookup Match Output (49 679 linhas)
Lookup Product
Lookup Match Output (39 843 linhas)
Lookup Store
Lookup Match Output (29 807 linhas)
Lookup FK Territory
Lookup Match Output (19 871 linhas)
Multicast
9 935 linhas
OLE DB Destination

Figure xlv Error 3