

# MDSAA

Master Degree Program in  
**Data Science and Advanced Analytics**

**Business Intelligence**

Duarte Girão, student number: 20220670  
Marina Lashina, student number: 20220728  
Miguel Soares, student number: 20221405  
Natalia Puzina, student number: 20220722

Group 40

**NOVA Information Management School**  
**Instituto Superior de Estatística e Gestão de Informação**

Universidade Nova de Lisboa  
June 2023

## INDEX

1. Introduction	2
2. Presentation of the business organization	2
3. Identification of the business problem	3
4. Identification of the business questions	3
5. Description of the source data and discovery process	4
6. Description of the most important data engineering actions (etl)	6
6.1 Staging Phase	6
6.2 Etl	7
6.2.1 Etl Process - Dimension Product	7
6.2.2 Etl Process - Dimension Date	9
6.2.3 Etl Process - Dimension Location	11
6.2.4 Etl Process - Dimension Point of Sale	13
7. Model Development and Design	16
Data Warehousing importance	16
7.1. Dimensional model design	16
7.2. Dimension and fact tables	17
7.3. Granularity	18
7.4. Design overview	19
7.5. Different datatypes	20
7.6. Measures	22
7.7. Calculated columns	28
8. Model optimization	29
8.1. Hierarchies	29
8.2. Hidden Columns	29
8.3. Formatting	29
9. Visuals used	30
10. Analytical reports prepared to address business needs	31
11. Conclusion	37

## 1. INTRODUCTION

Over the past decades, data has been getting increasing importance in all aspects of the business. Along with the rise of technology and the internet, data has become faster to collect and easier to store.

Today, data plays a crucial role in the modern business world. It provides companies the valuable insights to help them make more informed decisions and improve their financial performance. By analyzing the company's data, we can allow the board to better understand its customer behavior, existing market trends, and the company's overall performance, which can lead to better data-supported decisions and success. Additionally, data can also be useful to identify current inefficiencies and areas for improvement, giving companies a competitive advantage over their direct competitors.

Business Intelligence tools aim to address these issues, by improving operational efficiency and optimizing processes, leading to cost savings and increased profitability.

In summary, data is essential for the business world to keep up competitive and strive in this fast-paced world. With this Business Intelligence project, we aim to help the **Brikint** board with useful data insights that can help them to maintain a competitive edge.

## 2. PRESENTATION OF THE BUSINESS ORGANIZATION

**Brikint** is a retail company specializing in selling a wide variety of products and services from a wide range of brands, mainly in the technology and electronic sectors. Established as a start-up in the USA in 1980, two young and wealthy business analysts came together to raise the business with their capital. During the initial five years, they were able to develop the company, with a focus on serving local communities, by signing supply agreements, establishing partnerships, and gaining customer loyalty. However, the first few years were difficult, and the company struggled to make a name for itself in the market.

With a decade of hard work and dedication, the business began to reap the rewards, and by the end of the twentieth century, the company then decided to expand internationally, along with a digital transformation of all the business processes, with Canada being the first choice.

As of the end of 2021, **Brikint** has become a big-sized company, leading in the market in the retail sector, shaping the current market and among the top preferences of customers. The business summary is the following:

- Sales are made in 5 continents and 7 countries: *Australia, Canada, Germany, Japan, Mexico, Nigeria, and USA*;
- Sales are made via 30 shops: *Australia(6), Canada(4), Germany(3), Japan(2), Mexico(2), Nigeria(2) and USA(10)*;
- Number of manufacturers on sale: 14, including own brand (Maximus)
- Total number of shop staff: 90

On average, the company has three warehouses per country, except the USA, which aggregates five warehouses and concentrates the majority of revenue. In these warehouses, the products are stored and shipped to the rest of the country.

In addition to its physical stores, **Brikint** has a robust online presence. They have a user-friendly website where customers can browse and purchase products from the comfort of their own homes. They also have a strong social media presence, regularly posting updates and promotions to engage with their followers.

Overall, the company has a broad customer base, ranging from 18 to 60 years old, with the biggest demand coming from the 25 to 40-year-old segment, who are always looking for the latest technology trends or want to equip their homes with the latest products, presenting a relevant purchasing power. With a constant eye on innovation and expansion, **Brikint** aims to maintain its position as a leader in the retail sector and continue to meet the ever-changing demands of its customers.

The company's strategic target is to expand its activities further to present in 25 countries by 2035. The expansion plan is still less than 30% completed. The Board of Directors has decided to strengthen both strategic and operational decision-making process by improving analytical functions applied to the company's sales activities.

### 3. IDENTIFICATION OF THE BUSINESS PROBLEM

The main **Brikint** business problem identified as the absence of a flexible analytical system to obtain a regular readily available overview of sales performance in relation to all stores and products sold across all countries of presence. This system should allow data driven decision-making and develop more efficient market strategies based on more robust data insights with overall effect of improving the quality of strategic and tactic decisions. To achieve this objective, we have created the desired virtual solution using **Power BI** software and source data provided by **Brikint** (sales information provided in 7 separate csv files, representing sales from each country between 2014 and 2021, inclusive).

**Power BI** is a business intelligence tool that allows users to analyze data and share insights. The software was chosen due to its ability to incorporate and analyze data from multiple sources, generate visually appealing reports and dashboards, and provide real-time updates. Analysis provided with the specially generated pre-defined or ad hoc created reporting tools incorporated in Power BI could be used at different management level within the organization and respond to business purposes corresponding to each manager's roles and functions. High level performance summaries are especially useful for board members.

### 4. IDENTIFICATION OF THE BUSINESS QUESTIONS

The following business needs summary was provided by Brinkit and addressed by us in the process of model development and design:

1. Sales evolution and change in financial performance is to be presented, aggregated and analyzed by:

- Manufacturer;
- Product;
- Country;
- Store;
- Staff role

Manufacturer analysis should include information about its relevant market share to analyze competition.

2. Sales evolution analysis should cover:

- Historical actual sales data (years from 2014 to 2020)
- Current period actual sales data (year 2021)

3. The model should generate the following additional information:

- Target sales (used as a basis for KPI indicators calculation and evaluation for current period sales)
- Sales forecast (year 2022)
- Discount calculation

Target sales is to be calculated as 15% increase to the actual sales for the corresponding period of previous year. Sales forecast is to be generated automatically and incorporated in sales analysis. Discount criteria was provided to us by Brinkit and will be used for calculation of relevant discount amounts and Sales net of discount, as per Brinkit request.

4. The following additional requirements to be addressed and implemented:

- Max flexible time period for analysis (year, month, week, day)
- Currency of presentation (add EURO)
- Information level restrictions

US Dollar is the currency of operations, EUR is to be used as an option for data presentation.

Security groups for each data domain should be created based on the information level restrictions requested by Brinkit.

The resulting analytical reports are to be grouped and addressed to different users groups within Brinkit based on their functional focus.

## **5. DESCRIPTION OF THE SOURCE DATA AND DISCOVERY PROCESS**

The following source data was provided to us by Brinkit:

1. “*BI\_Dimensions*” Excel file, containing 5 sheets (*product, manufacturer, geo, stores, and staff*) with the following information available:
  - “Product” sheet provides us information on the existing products ID and products names, and their respective manufacturer ID, category, and price;
  - “Manufacturer” sheet provides us with information of the manufacturer ID and their respective name and logo;
  - “Geo” sheet provides us with information of the zip code, the city, the state, the region, the district, and the country;
  - “Stores” sheet provides us with information on the store ID and their respective store name and country;
  - “Staff” sheet provides us with information of the staff ID and their respective role and country;
2. 7 csv files, with the sales detailed information per country where Brikint operates for the period from 2014 to 2021 (Australia, Canada, Germany, Japan, Mexico, Nigeria, USA):
  - The **Product ID** of the corresponding transaction. We can get additional information about this product on the “Product” sheet from the “*BI\_Dimensions*” file;
  - The **Date** of the corresponding transaction. This is in the following format: DD/MM/YYYY.
  - The **Zip Code**, of the corresponding store where the transaction occurred. We can get additional information about this product on the “Geo” sheet from the “*BI\_Dimensions*” file;
  - The **units** of the corresponding transaction. It does not assume decimal numbers;
  - The **revenue**, in dollars, of the corresponding transaction.
  - The **story** of the corresponding store where the transaction occurred. We can get additional information about this product on the “Store” sheet from the “*BI\_Dimensions*” file;
  - The **staffID** of the corresponding employee who attended to the costumer on this specific transaction. We can get additional information about this product on the “Staff” sheet from the “*BI\_Dimensions*” file;

The following external information was used by us in order to address currency conversion request from Brikint:

1. Currency converter source;
2. The Excel file ‘World Currencies’ provided contains information about various countries and their respective currencies:
  - **Country Name:** This column lists the names of various countries around the world. For example, it might include entries such as "United States", "Canada", "Mexico" and so on.
  - **Currency Name:** This column lists the names of the currencies used in each of the countries listed in the first column.
  - **Currency Code:** This column lists the unique three-letter currency codes that are used to identify each currency. These codes are established by the

International Organization for Standardization (ISO) and are commonly used in international finance and commerce.

- **Currency Symbol:** This column lists the symbols that are commonly used to represent each currency.
3. The dataset provided tracks the exchange rate between two currencies over a period of time:
- **Date:** Indicates the date on which the exchange rate was recorded.
  - **Base Currency:** Currency that is being used as the base currency for the exchange rate.
  - **Change Currency:** Currency that is being compared to the base currency to determine the exchange rate.
  - **Rate:** The exchange rate between the Base Currency and the Change Currency on the specified date.

## 6. DESCRIPTION OF THE MOST IMPORTANT DATA ENGINEERING ACTIONS (ETL)

Data Engineering is a broader concept that comprehends the entire process of preparing and transforming all our dataset in order to present more clear and insightful analysis and visualizations. One of the initial subset topics is **Data Warehousing**, which is explained further in 7 of the Report and was initially designed; another one, is **ETL** (Extract, Transform and Load), which we will explore in detail below.

After defined our star schema in a theoretical perspective, we moved on to its implementation of our model in **Power BI Desktop**, using for that purpose, **Power Query Editor**.

### ○ 6.1 STAGING PHASE

We have started by extracting the data from our data server, seven already mentioned “CSV” files. To these files, we have applied the necessary transformations, loaded all the necessary information into the Staging Area and after extracting from it, we have loaded into the Data Warehousing.

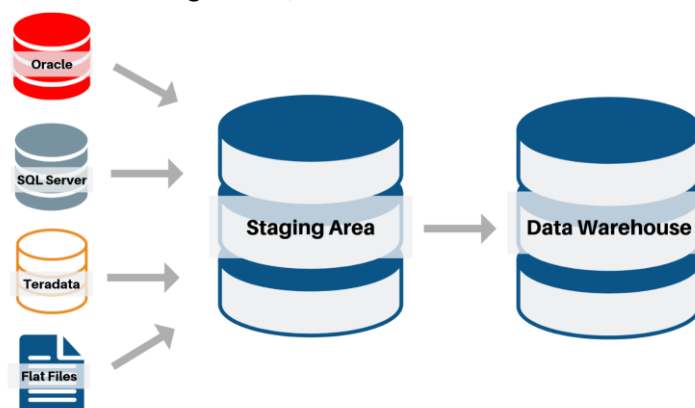
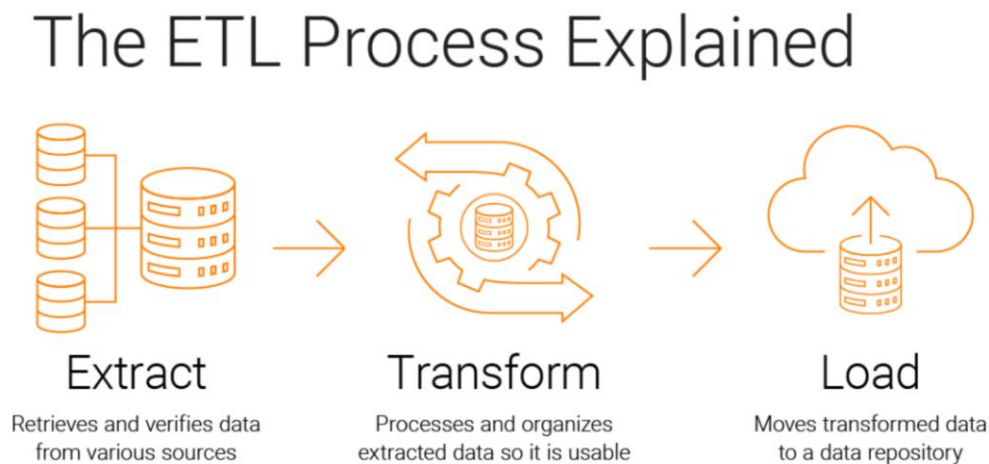


Figure 1 – “Staging Phase”

## ○ 6.2 ETL

ETL is responsible for the extraction, transformation, and loading of data to support effective data analysis, reporting, and decision-making. After designing our Data Warehousing, our main goal is to describe our rationale from the **data extraction** of those original seven “CSV” files into the **data transformation** to enable *Brikint* business decision makers to make final important decisions. In this sense, we will describe the ETL process performed for each one of our five dimensions.



**Figure 2 – “ETL Process Summary”**

### ○ 6.2.1 ETL PROCESS - DIMENSION PRODUCT

Firstly, regarding our dimension Product, we performed several transformations, as described below:

- **Promoted Headers:** We started out with a **Promoted Headers**, since our desired columns headers were not in the first row. This transformation is used in cases where the headers were not correctly identified and allows us to use the values in the first row as the column names, overriding the automatically detected headers. This action was applied to the entire sheet;
- **Changed Type:** Then, we performed some **Change Types** on all columns since they were not in the most adequate data format. We changed both the columns “Product Detail” and “Column2” to any datatype, and the “Column3”, “Column4” and “Column5” to text datatype. Note that in the original data source file, these mentioned 4 columns, were originated from the disaggregation of the “Product Detail” cell that was merging the first 5 cells of the first row;
- **Promoted Headers:** Then, after this change types, we reperformed the first action regarding promoted headers, because our goal was to get the initial second row (and now a first row) as our headers, representing the columns “ProductID”, “ManufacturerID”, “Product”, “Category” and “Price”. After concluding this process, we finally obtained our desired headers.
- **Changed Type1:** Additionally, we performed a second **Change Type** also changed the columns “ProductID” and “ManufacturerID” to any Int64 and the columns “Product”, “Category” and “Price” to text datatype;
- **Filled Down:** Then, we **filled down** all the records in “Category” column. This function aims to fill the empty cells with the value of the cell right above, until finding a new value. When finding



a new value, it automatically fills the cell below with the new value found, following this rationale until the end of the column. This “Category” column was the only one with null values that required this transformation;

- **Replace Value:** Then, we **replaced** all the strings with “USD ” (note that there is a space right after the word “USD”) from “Price” column with blanks, since we want to get the numerical values of each price. For example, from “USD 412.13”, we aimed to keep the “412.13” information on each cell;
- **Split Column by Delimiter:** Then, we perform a necessary action for our future interpretation of the model: **split our columns by our delimited**, since all the records on our “Product” column, had a “|”, dividing the product model name from its category and subcategory. After performing this, we obtained two more columns (at the right of the “Product” column) called “Product.1” and “Product.2”, which, as we will see below, will be our new “Product Category” and “Product SubCategory” columns;
- **Change Type2:** Then, we reperformed the **Change Types** transformation to the columns originated in the previous step (“Product.1” and “Product.2”). We changed both the columns text datatype, since we were dealing with strings;
- **Rename Columns:** Then, we simply **renamed some columns**, aiming to clarify the information presented on which column. We used the prefix “Product” on each column, standardizing the information and facilitating user interpretation:
  - The column “Product.1” was changed into “Product Model”;
  - The column “Product.2” was changed into “Product SubCategory”;
  - The column “Category” was changed into “Product Category”;
  - The column “Price” was changed into “Product Price”;
  - The column “Category” was changed into “Product Category”;
  - The column “Manufacturer ID” was changed into “Product Manufacturer”;
- **Rename Columns1:** Then, we performed one additionally **column rename**, slightly different from the previous one. We transformed the column “ProductID” into “BK Product” along with our Business Key definition, once this is the product that uniquely identifies each product from this dimension table “Product”;
- **Change Type3:** Additionally, we performed one more **Change Types** transformation, similarly to the previous step. We transformed “Product Price” into a currency data type;
- **Merged Queries:** On this step, we performed a queries merge which aims to joins two existing tables together based on matching values from multiple columns. In this specific case, we are joining our current table with “Product” (using “Product Manufacturer” column) with “Manufacturer” table (using “BK Manufacturer” column). Using a left join, we ensured that every record from our “Product” table was kept in the updated table;
- **Expanded Manufacturer:** On this step, we just expanded the table column called “Manufacturer”. This facilities user interpretation, by bringing the nested column to the top level of the table;
- **Rename Columns2:** Once again, we performed one more **column rename**. We transformed the column “Manufacturer.Manufacturer Name” into “Manufacturer Name” and the column “Manufacturer.Manufacturer Logo” into “Manufacturer Logo”;
- **Reordered Columns:** And finally, for a more clear and concise transformation, we **reordered** our existing columns and **removed** unnecessary ones. On this reordering step, we brought the most relevant ones to the left side of the column, organizing the attributes by level of

importance. In this sense, we started by our previously defined Business Key and we established the following order: "BK Product", "Product Manufacturer", "Manufacturer Logo", "Manufacturer Name", "Product Model", "Product SubCategory", "Product Category" and "Product Price".

- **Removed Columns:** Since our "Product Manufacturer" and our "Manufacturer Name" contain the same information, for a clearer interpretation once again, we opted to remove the "Product Manufacturer";
- **Reordered Columns1:** Finally, we performed another columns reordination, by highlighting the "Manufacturer Name" in front of "Manufacturer Logo".

For more detailed information, see below **Figure 3**.

```
let
Source = Excel.Workbook(File.Contents("C:\temp\brikint\Data\BI_Dimensions.xlsx"), null, true),
product_Sheet = Source[Item="product",Kind="Sheet"][Data],
#"Promoted Headers" = Table.PromoteHeaders(product_Sheet, [PromoteAllScalars=true]),
#"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Product Details", type any}, {"Column2", type any}, {"Column3", type text}, {"Column4", type text}, {"Column5", type text}}),
#"Promoted Headers1" = Table.PromoteHeaders(#"Changed Type", [PromoteAllScalars=true]),
#"Changed Type1" = Table.TransformColumnTypes(#"Promoted Headers1",{{"ProductID", Int64.Type}, {"ManufacturerID", Int64.Type}, {"Product", type text}, {"Category", type text}, {"Price", type text}}),
#"Filled Down" = Table.FillDown(#"Changed Type1",{"Category"}),
#"Replaced Value" = Table.ReplaceValue(#"Filled Down", "USD ", "", Replacer.ReplaceText, {"Price"}),
#"Split Column by Delimiter" = Table.SplitColumn(#"Replaced Value", "Product", Splitter.SplitTextByDelimiter("|", QuoteStyle.Csv), {"Product.1", "Product.2"}),
#"Changed Type2" = Table.TransformColumnTypes(#"Split Column by Delimiter",{{"Product.1", type text}, {"Product.2", type text}}),
#"Renamed Columns" = Table.RenameColumns(#"Changed Type2",{{"Product.2", "Product SubCategory"}, {"Product.1", "Product Model"}, {"Category", "Product Category"}, {"Price", "Product Price"}, {"ManufacturerID", "Product Manufacturer"}}),
#"Renamed Columns1" = Table.RenameColumns(#"Renamed Columns",{{"ProductID", "BK Product"}}),
#"Changed Type3" = Table.TransformColumnTypes(#"Renamed Columns1",{{"Product Price", Currency.Type}}),
#"Merged Queries" = Table.NestedJoin(#"Changed Type3", {"Product Manufacturer"}, Manufacturer, {"BK Manufacturer"}, "Manufacturer", JoinKind.LeftOuter),
#"Expanded Manufacturer" = Table.ExpandTableColumn(#"Merged Queries", "Manufacturer", {"Manufacturer Name", "Manufacturer Logo"}, {"Manufacturer.Manufacturer Name", "Manufacturer.Manufacturer Logo"}),
#"Renamed Columns2" = Table.RenameColumns(#"Expanded Manufacturer",{{"Manufacturer.Manufacturer Name", "Manufacturer Name"}, {"Manufacturer.Manufacturer Logo", "Manufacturer Logo"}}),
#"Reordered Columns" = Table.ReorderColumns(#"Renamed Columns2",{"BK Product", "Product Manufacturer", "Manufacturer Logo", "Manufacturer Name", "Product Model", "Product SubCategory", "Product Category", "Product Price"}),
#"Removed Columns" = Table.RemoveColumns(#"Reordered Columns",{"Product Manufacturer"}),
#"Reordered Columns1" = Table.ReorderColumns(#"Removed Columns",{"BK Product", "Manufacturer Name", "Manufacturer Logo", "Product Model", "Product SubCategory", "Product Category", "Product Price"}),
in
#"Reordered Columns1"
```

**Figure 3 – “ETL Dimension Product”**

## ○ 6.2.2 ETL PROCESS - DIMENSION DATE

Second, regarding our dimension Date, we performed several transformations, as described below:

- **Changed Type:** Then, we performed a **Change Types** on the only column from our dimension since it was not in the most adequate data format. We changed from "Column1" to date datatype;
- **Rename Columns:** Then, we simply **renamed the only column** from our dimension, aiming to clarify the information presented on this column:
  - The column "Column1" was changed into "Date";
- **Inserted Day:** Here, we start the disaggregation from our single column "Date" into the several components. We then added a new column called "Day" to the table which will be derived from the previously created column by extracting the day component, through Date.Day function. The resulting table will have an additional column containing the only the day values;
- **Inserted Day of Week:** The second disaggregation we are performing from our column "Date" is into the "Day of Week", through Date.DayOfWeek function. We added a new column called "Day of Week" to the table, containing the day of the week and increased by one, in order to

start by Sunday. The resulting table will have an additional column containing these day of the week values;

- **Inserted Day Name:** The third disaggregation we performed was the addition of a new column called "Day Name" to the dimension table, through Date.DayOfWeekName function. The values in the "Day Name" column come from the corresponding values in the "Day of Week" column by calculating the localized name of the day of the week. Summary, the key match will be the following:
  - **Sunday - 1; Monday - 2; Tuesday - 3; Wednesday - 4; Thursday - 5; Friday - 6; Saturday - 7;**
- **Changed Type1:** Additionally, we performed a second **Change Type** also changed the column "Day of Week" into an Int64 datatype;
- **Added Custom:** A fourth derivation from our initial "Date" column will be the addition of a new column called "Month" to the dimension table. These values in the "Month" column will directly come from the "Date" column by taking the month component, through Date.Month function;
- **Change Type2:** Then, we reperformed the **Change Types** transformation to created column in the previous step ("Month"). We changed both the columns Int64 datatype, since we were dealing with numbers;
- **Inserted Month Name:** The next disaggregation we performed was the addition of a new column called "Month Name" to the dimension table, through Date.MonthName function. The values in the "Month Name" column come from the corresponding values in the "Month" column by calculating the localized name of the month. Summary, the key match will be the following:
  - **January - 1; February - 2; March - 3; April - 4; May - 5; June - 6; July - 7; August - 8; September - 9; October - 10; November - 11; December - 12;**
- **Change Type3:** Additionally, we performed one more **Change Types** transformation, similarly to the previous step. We made sure that the column "Month" was into an Int64 data type;
- **Inserted Year:** The following disaggregation we performed was the addition of a new column called "Year" to the dimension table, through Date.Year function. The values in the "Year" obviously also come from "Date" column.
- **Duplicated Year:** Here, we decided to create a new safety column, which is an exact copy named "Date - Copy" of the existing "Date" column. This was made to keep the original column despite future transformations, to avoid any lost information;
- **Split Column by Delimiter:** Then, we perform a crucial step for our future interpretation of the model: **splited our columns by our delimited**, since all the records on our "Date- Copy" column, had a "/", dividing the different components. After performing this step, we obtained three more columns (at the right of the "Date-Copy" column) called ("Date - Copy.1", "Date - Copy.2", and "Date - Copy.3", which represents the splits from the original column;
- **Added Custom1:** Then, we added a new column called "SK Date" to the table, which results from the concatenation of the values from the three following columns: "Date - Copy.3", "Date - Copy.2", and "Date - Copy.1";
- **Change Type4:** Additionally, we performed one more **Change Types** transformation, following the rationale from previous steps. We transformed the created "SK Date" into an Int64 data type;

- **Reordered Columns:** And finally, as we did in the previous dimension, for a more clear and concise transformation, we **reordered** our columns. On this reordering step, we brought the most relevant ones to the left side of the column, organizing the attributes by level of importance. In this sense, we started by our previously defined Surrogated Key and we established the following order: *SK Date*, *Date*, *Day*, *Day of Week*, *Day Name*, *Month*, *Month Name*, *Year*, *Date - Copy.1*, *Date - Copy.2*, *Date - Copy.3*.
- **Removed Columns:** Getting to the end of our process, we are certain that we did not loose any important information, we can delete our duplicated columns, keeping only the needed information for a clearer interpretation once again. So, we opted to remove the following columns: *Date - Copy.1*, *Date - Copy.2*, *Date - Copy.3*;

For more detailed information, see below **Figure 4**.

```

let
    Source = List.Dates(#date(2014,1,1), Number.From(#date(2021,12,31)) + 1 - Number.From(#date(2014,1,1)), #duration(1,0,0,0)),
    #"Converted to Table" = Table.FromList(Source, Splitter.SplitByNothing(), null, null, ExtraValues.Error),
    #"Changed Type" = Table.TransformColumnTypes(#"Converted to Table",{{"Column1", type date}}),
    #"Renamed Columns" = Table.RenameColumns(#"Changed Type",{{"Column1", "Date"}}),
    #"Inserted Day" = Table.AddColumn(#"Renamed Columns", "Day", each Date.Day([Date]), Int64.Type),
    #"Inserted Day of Week" = Table.AddColumn(#"Inserted Day", "Day of Week", each Date.DayOfWeek([Date]) + 1),
    #"Inserted Day Name" = Table.AddColumn(#"Inserted Day of Week", "Day Name", each Date.DayOfWeekName([Date]), type text),
    #"Changed Type1" = Table.TransformColumnTypes(#"Inserted Day Name",{{"Day of Week", Int64.Type}}),
    #"Added Custom" = Table.AddColumn(#"Changed Type1", "Month", each Date.Month([Date])),
    #"Changed Type2" = Table.TransformColumnTypes(#"Added Custom",{{"Month", Int64.Type}}),
    #"Insert Month Name" = Table.AddColumn(#"Added Custom", "Month Name", each Date.MonthName([Date]), type text),
    #"Changed Type3" = Table.TransformColumnTypes(#"Insert Month Name",{{"Month", Int64.Type}}),
    #"Inserted Year" = Table.AddColumn(#"Changed Type3", "Year", each Date.Year([Date]), Int64.Type),
    #"Duplicated Column" = Table.DuplicateColumn(#"Inserted Year", "Date", "Date - Copy"),
    #"Split Column by Delimiter" = Table.SplitColumn(Table.TransformColumnTypes(#"Duplicated Column", {{"Date - Copy", type text}}, "pt-PT"), "Date - Copy",
        Splitter.SplitTextByDelimiter("/", QuoteStyle.Csv), {"Date - Copy.1", "Date - Copy.2", "Date - Copy.3"}),
    #"Added Custom1" = Table.AddColumn(#"Split Column by Delimiter", "SK Date", each [#"Date - Copy.3"] & [#"Date - Copy.2"] & [#"Date - Copy.1"]),
    #"Changed Type4" = Table.TransformColumnTypes(#"Added Custom1",{{"SK Date", Int64.Type}}),
    #"Reordered Columns" = Table.ReorderColumns(#"Changed Type4",{"SK Date", "Date", "Day", "Day of Week", "Day Name", "Month", "Month Name", "Year", "Date - Copy.1", "Date - Copy.2", "Date - Copy.3"}),
    #"Removed Columns" = Table.RemoveColumns(#"Reordered Columns",{"Date - Copy.1", "Date - Copy.2", "Date - Copy.3"})
in
    #"Removed Columns"
  
```

**Figure 4 – “ETL Dimension Date”**

### ○ 6.2.3 ETL PROCESS - DIMENSION LOCATION

Third, regarding our dimension Location, we performed few transformations, when compared to the previous dimensions, as described below:

- **Promoted Headers:** We started out with **Promoted Headers**, as we had already done in previous dimensions, since our desired columns headers were not in the first row. This transformation replaced the automatically detected headers for the correct ones. This step was applied to the entire sheet;
- **Changed Type:** Then, we performed some **Change Types** on all columns since they were not in the most adequate data format. We changed both the columns "Source", "Column3", "Column4", "Column5" and "Column6" to text datatype, while "Public Database" we have changed into any datatype;
- **Remove Top Rows:** Then, we deleted the first two rows from the original, which does not bring useful information, referring only to the place and data of the withdrawal place;

- **Promoted Headers:** Then, we reperformed the first step regarding promoted headers, because our goal was to get the initial third row (and now a first row) as our headers, representing the columns "Zip", "City", "State", "Region", "District" and "Country". After concluding this process, we finally obtained our desired headers.
- **Changed Type1:** Additionally, we performed a second **Change Type**, changing all the columns "Zip", "City", "State", "Region", "District" and "Country" into text datatype;
- **Split Column by Delimiter:** Once again, we were obligated to **split our columns by our delimited**, since all the records on our "City" column, had a ",", since only the first part before comma corresponds to the city name;
- **Change Type2:** Then, we reperformed the **Change Types** transformation to the columns originated in the previous step ("City.1" and "City.2"). We changed both the columns text datatype, since we were dealing with strings;
- **Removed Columns:** Since only the "City.1" is relevant for us, we deleted "City.2";
- **Rename Columns:** Then, we **renamed** the "City.1" **column** for "City", making it more clear;
- **Replace Value:** Then, we **replaced** all the null values on "Country" column with "Nigeria" value, since there is only blank values from "Nigeria" dataset;
- **Added Index:** Then, we include an index into the table, to make it our surrogate key;
- **Rename Columns1:** After including the index, we simply **renamed that column**, and defined it as the surrogate key, which uniquely defines each record in this dimension:
  - o The column "Index" was changed into "SK Location";
- **Reordered Columns:** And finally, we **reordered** our existing columns, keeping the most relevant ones to the left side of the column, organizing the attributes by level of importance. In this sense, we started by our previously defined Surrogate Key and we established the following order: "SK Location", "Zip", "City", "State", "Region", "District", "Country".
- **Removed Columns:** Since our "District" contained multiple null values and we do not consider it much relevant for our analysis, we opted to remove it;
- **Removed Columns:** On this step, we aimed to remove the rows with the same "Zip" and "Country" values. The resulting table will only contain unique combinations of different "Zip" and "Country" values;
- **Filtered Rows:** On the last step, we filter the rows in the table and select only the ones where the value in the "State" column is either null or an empty string (""). At the end, we will only keep the rows where the "State" column is either null or empty.

For more detailed information, see below **Figures 5 and 6**.

```

let
    Source = Excel.Workbook(File.Contents("C:\temp\Brikint\Data\81_Dimensions.xlsx"), null, true),
    geo_Sheet = Source[[Item="geo", Kind="Sheet"]][Data],
    #Promoted Headers = Table.PromoteHeaders(geo_Sheet, [PromoteAllScalars=true]),
    #Changed Type = Table.TransformColumnTypes(#Promoted Headers,{{"Source:", type text}, {"Public Database ", type any}, {"Column3", type text}, {"Column4", type text}, {"Column5", type text}, {"Column6", type text}}),
    #Removed Top Rows = Table.Skip(#Changed Type, 2),
    #Promoted Headers1 = Table.PromoteHeaders(#Removed Top Rows, [PromoteAllScalars=true]),
    #Changed Type1 = Table.TransformColumnTypes(#Promoted Headers1,{{"Zip", type text}, {"City", type text}, {"State", type text}, {"Region", type text}, {"District", type text}, {"Country", type text}}),
    #Split Column by Delimiter = Table.SplitColumn(#Changed Type1, "City", Splitter.SplitTextByEachDelimiter({","}, QuoteStyle.Csv, false), {"City.1", "City.2"}),
    #Changed Type2 = Table.TransformColumnTypes(#Split Column by Delimiter,{{"City.1", type text}, {"City.2", type text}}),
    #Removed Columns = Table.RemoveColumns(#Changed Type2, {"City.2"}),
    #Renamed Columns = Table.RenameColumns(#Removed Columns,{{"City.1", "City"}}),
    #Replaced Value = Table.ReplaceValue(#Renamed Columns, null, "Nigeria", Replacer.ReplaceValue, {"Country"}),
    #Added Index = Table.AddIndexColumn(#Replaced Value, "Index", 1, 1, Int64.Type),
    #Renamed Columns1 = Table.RenameColumns(#Added Index,{{"Index", "SK Location"}}),
    #Reordered Columns = Table.ReorderColumns(#Renamed Columns1, {"SK Location", "Zip", "City", "State", "Region", "District", "Country"}),
    #Filtered Rows = Table.SelectRows(#Reordered Columns, each true),
    #Removed Columns1 = Table.RemoveColumns(#Filtered Rows, {"District"})
in
    #Removed Columns1

```

Figure 5 – “ETL Dimension Location”

```

let
    Source = #Geo Location,
    #Removed Duplicates = Table.Distinct(Source, {"Zip", "Country"}),
    #Filtered Rows = Table.SelectRows(#Removed Duplicates, each ([State] = null or [State] = ""))
in
    #Filtered Rows

```

Figure 6 – “ETL Dimension Location”

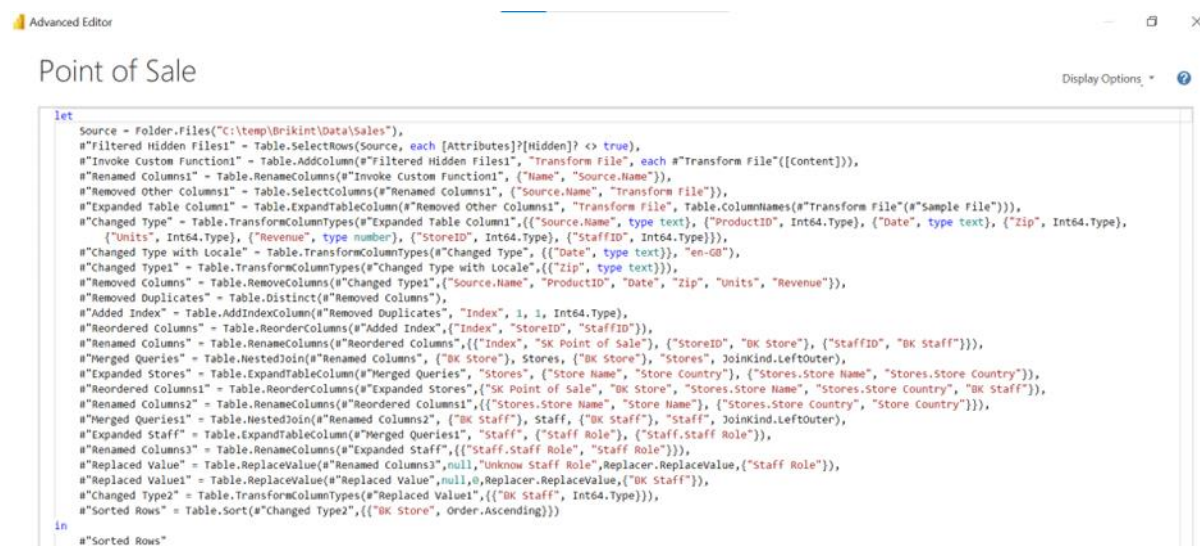
#### ○ 6.2.4 ETL PROCESS - DIMENSION POINT OF SALE

- **Filtered Hidden Files1:** We started out by filtering out hidden files from the original table;
- **Invoke Custom Function1:** Then, we added a new column called "Transform File" to our table;
- **Renamed Columns1:** Then, we simply **renamed** the column “Name” into “Stores.Name”;
- **Removed Other Columns1:** On this step, we removed all the countries from our columns, besides “Tranform File” and “Stores.Name” from our table, created in the two previous points;
- **Expanded Table Column1:** On this step, we just expanded the table column "Transform File", which will be added separately in the resulting table. This will facilitate user interpretation and bringing the nested columns into the top;
- **Changed Type:** Then, we performed some **Change Types** on all columns since they were not in the most adequate data format. We changed both the columns "Source.Name" and "Date" text datatype, the columns "ProductID", "Zip", "Units", "StoreID" and "StaffID" to Int64 datatype and the column “Revenue" to number datatype;

- **Changed Type with Locale and Changed Type1:** Similarly to the previous step, we aimed to transform "Zip" column into text datatype, using the locale "en-GB" (English one);
- **Removed Columns:** We removed the columns "Source.Name", "ProductID", "Date", "Zip", "Units", "Revenue", since they do not contain needed information;
- **Removed Duplicates:** Then, we removed all the totally equal rows (with all column values equal) from our table, keeping only the unique records;
- **Added Index:** Then, we include an index into the table, to make it our surrogate key in the following steps;
- **Reordered Columns:** After adding the index column, we reorganized our existing table by level of relevancy, in the following order: "Index", "StoreID", "StaffID";
- **Renamed Columns:** Then, we gave the right name for each of the three columns in the table: "SK Point of Sale" for "Index", "BK Store" for "StoreID" and "BK Staff" for "StaffID";
- **Merged Queries:** On this step, we performed a queries merge which aims to joins two existing tables together based on matching values from multiple columns. In this specific case, we are joining our current table with "Stores" (using "BK Store" column from both columns). Using a left join, we ensured that every records from our currently table was kept in the new table;
- **Expanded Stores:** Here, we expanded the nested table column "Stores", creating two separate columns: one for "Store Name" and one for "Store Country";
- **Reordered Columns1:** Once again, for clearer and concise transformation, we **reordered** our existing columns. As previously, we brought the most relevant ones to the left side of the column, starting by our Surrogate Key, establishing the following order: "SK Point of Sale", "BK Store", "Stores.Store Name", "Stores.Store Country", "BK Staff".
- **Renamed Columns2:** After having our columns in the right order, we performed some name changes: "Stores.Store Name" for "Store Name" and "Stores.Store Country" for "Store Country";
- **Merged Queries1:** Once again, we performed a new join, by joining our current table with "Staff" (using "BK Staff" column from both columns). Using a left join, we ensured that every records from our currently table was kept in the new table;
- **Expanded Staff:** Here, we expanded the nested table column "Staff", creating one separate column called "Staff.Staff Role";
- **Renamed Columns3:** Having expanded "Staff Role", we changed its name from "Staff.Staff Role" into "Staff Role";
- **Replaced Value:** On this step we aim to replace every possible null values in the "Staff Role" column, created, with the following text: "Unknown Staff Role";

- **Replaced Value1:** Then, similarly to the previous step, we aim to replace every possible null values in the "BK Staff" column with the value 0;
- **Changed Type2:** Once again, we aimed to transform the aimed to transform "BK Staff" into a Int64;
- **Sorted Rows:** Now, sorting out our records in the table based on the values in the "BK Store" column in an ascending order:
- **Removed Column:** On this step, we removed the columns "BK Store" and "BK Staff";

For more detailed information, see below **Figure 7 and 8**.



The screenshot shows the 'Advanced Editor' window with the title 'Point of Sale'. The script is written in MQL and performs various data transformation steps. It starts by loading data from a folder, then applies a series of transformations including renaming columns, expanding table columns, changing data types, removing columns, and finally sorting the data by 'BK Store' in ascending order. The script concludes with a 'Sorted Rows' step.

```
let
Source = Folder.Files("C:\temp\Brikint\Data\Sales"),
#"Filtered Hidden Files1" = Table.SelectRows(Source, each [Attributes]?[Hidden]? <> true),
#"Invoke Custom Function1" = Table.AddColumn(#"Filtered Hidden Files1", "Transform File", each #"Transform File"([Content])),
#"Renamed Columns1" = Table.RenameColumns(#"Invoke Custom Function1", {"Name", "Source.Name"}),
#"Removed Other Columns1" = Table.SelectColumns(#"Renamed Columns1", {"Source.Name", "Transform File"}),
#"Expanded Table Column1" = Table.ExpandTableColumn(#"Removed Other Columns1", "Transform File", Table.ColumnNames(#"Transform File"("Sample File"))),
#"Changed Type" = Table.TransformColumnTypes(#"Expanded Table Column1",{{"Source.Name", type text}, {"ProductID", Int64.Type}, {"Date", type text}, {"Zip", Int64.Type}, {"Units", Int64.Type}, {"Revenue", type number}, {"StoreID", Int64.Type}, {"StaffID", Int64.Type}}),
#"Changed Type with Locale" = Table.TransformColumnTypes(#"Changed Type", {{("Date", type text)}, {"en-GB"}),
#"Changed Type1" = Table.TransformColumnTypes(#"Changed Type with Locale",{{("Zip", type text)}),
#"Removed Columns" = Table.RemoveColumns(#"Changed Type1",{"Source.Name", "ProductID", "Date", "Zip", "Units", "Revenue"}),
#"Removed Duplicates" = Table.Distinct(#"Removed Columns"),
#"Added Index" = Table.AddIndexColumn(#"Removed Duplicates", "Index", 1, 1, Int64.Type),
#"Reordered Columns" = Table.ReorderColumns(#"Added Index",{"Index", "StoreID", "StaffID"}),
#"Renamed Columns" = Table.RenameColumns(#"Reordered Columns",{{("Index", "SK Point of Sale"), {"StoreID", "BK Store"}, {"StaffID", "BK Staff"}}),
#"Merged Queries" = Table.NestedJoin(#"Renamed Columns", {"BK Store"}, Stores, {"BK Store"}, "Stores", JoinKind.LeftOuter),
#"Expanded Stores" = Table.ExpandTableColumn(#"Merged Queries", "Stores", {"Store Name", "Store Country"}, {"Stores.Store Name", "Stores.Store Country"}),
#"Reordered Columns1" = Table.ReorderColumns(#"Expanded Stores",{"SK Point of Sale", "BK Store", "Stores.Store Name", "Stores.Store Country", "BK Staff"}),
#"Renamed Columns2" = Table.RenameColumns(#"Reordered Columns1",{{("Stores.Store Name", "Store Name"), {"Stores.Store Country", "Store Country"}),
#"Merged Queries1" = Table.NestedJoin(#"Renamed Columns2", {"BK Staff"}, Staff, {"BK Staff"}, "Staff", JoinKind.LeftOuter),
#"Expanded Staff" = Table.ExpandTableColumn(#"Merged Queries1", "Staff", {"Staff Role"}, {"Staff.Staff Role"}),
#"Renamed Columns3" = Table.RenameColumns(#"Expanded Staff",{{("Staff.Staff Role", "Staff Role"}),
#"Replaced Value" = Table.ReplaceValue(#"Renamed Columns3",null,"Unknown Staff Role",Replacer.ReplaceValue,{"Staff Role"}),
#"Replaced Value1" = Table.ReplaceValue(#"Replaced Value",null,0,Replacer.ReplaceValue,{"BK Staff"}),
#"Changed Type2" = Table.TransformColumnTypes(#"Replaced Value1",{{("BK Staff", Int64.Type)}),
#"Sorted Rows" = Table.Sort(#"Changed Type2",{{("BK Store", Order.Ascending)}})
in
#"Sorted Rows"
```

**Figure 7 – “ETL Dimension Point of Sale”**



The screenshot shows the 'Advanced Editor' window with the title 'Dim Point of Sale'. The script is a simple MQL query that selects data from a table named 'Point of Sale' and removes the columns 'BK Store' and 'BK Staff'.

```
let
Source = #"Point of Sale",
#"Removed Columns" = Table.RemoveColumns(Source,{"BK Store", "BK Staff"})
in
#"Removed Columns"
```

**Figure 8 – “ETL Dimension Point of Sale”**



## 7. MODEL DEVELOPMENT AND DESIGN

### ○ DATA WAREHOUSING IMPORTANCE

Firstly, we would like to clarify Data Warehousing importance in a Business Intelligence context. It evolves organizing and storing big volumes of either structured or semi-structured historical data from several data sources in a central single repository. This provides a unique view of the data and is designed to support efficient analysis and Power BI is an example of a tool that can use data warehouses as a source of data for analysis and reporting.

Typically, a robust and efficient Data Warehousing present the following characteristics:

- **Scalability**, since is constructed to deal with very large volumes of data. This includes both a horizontally and a vertically scalability to tackle increasing user demands;
- **Subject-Oriented**, once is made to support reporting on specific topics to *Brikint*, such as sales, product types, points of sale and date, providing more insightful and optimized for subsequent analysis;
- **Time-Variant**, normally storing historical data, as already said above. This allows the users to comprehend trends and patterns throughout time and capture historical images of data, facilitating comparisons;
- **Optimization**, in order to obtain faster query performance. We applied several techniques like aggregations to improve query speed and analytical performance, as described in the previous section 6 of the report;
- **Quality and Cleanliness**, passing through a data quality management process to guarantee a high level of accuracy, consistency and completeness of the data. This includes data cleaning, and data validation techniques, which described in the previous report section;
- **Non-Volatile**, since once loaded into the warehouse, it cannot lately be updated or modified directly. These updates are made through ETL processes that refresh the data warehouse with the new information;
- **Consolidated**, since it captures data from diverse sources. Provides an uniform view of the data by solving inconsistencies and standardizing formats from all data sources.

Data Warehousing is a step of the highest importance and its accuracy is key in order to correctly address our business needs defined in sections 3 and 4 of the Report.

### 7.1. DIMENSIONAL MODEL DESIGN

The dimensional model designed follows a **Ralph Kimball's Methodology** star schema, which is one of its core concepts. This methodology:

- follows a **Business-Driven Approach**, focusing on aligning data warehouse with the business needs and objectives of the company (*Brinkit*, in this case). It highlights the business requirements and the key performance indicators (KPIs) to drive the design of the data warehouse;
- presents a very intuitive and user-friendly **Dimensional Modeling**, called Start Schema (the one we are following). It defines relationships, primary and foreign keys and is highly

optimized to perform query performance, ensuring data quality and accuracy in our database;

- supports the usage of multiple **Business Intelligence Tools** as Power BI and Tableau to enable data visualization, reporting, and analysis, which makes it easier for business users to access data.

A star schema is a database modeling technique commonly used in data warehousing and business intelligence projects. It is a schema design used to organize and store data in a way that simplifies query processing and reporting in a relational data model. Its basic form consists of a single fact table surrounded by multiple dimension tables, all connected by joins between the primary key and foreign keys (usually called surrogate keys). Dimension tables describe business entities—the things you model. Entities can include products, people, places, and concepts including time itself. Fact tables store observations or events, and can be sales orders, stock balances, exchange rates, temperatures, etc. A fact table contains dimension key columns that relate to dimension tables, and numeric measure columns. The dimension key columns determine the dimensionality of a fact table, while the dimension key values determine the granularity of a fact table. The star schema adopts a denormalized structure, which means that redundant data is intentionally introduced to improve query performance. This denormalization simplifies the data model and eliminates the need for complex joins during querying. The principles of a star schema can be defined as follows: 1) Dimension tables facilitate filtering and grouping of data. 2) Fact tables enable the summarization of data. In summary, a **star schema** is known for its simplicity, as it requires fewer joins compared to other database models. This makes it easier for users to create queries that are both quick and accurate.

## 7.2. DIMENSION AND FACT TABLES

**Figure 9** represents the dimensional model adopted. There are, in total, 6 dimension tables and 2 fact tables are represented.

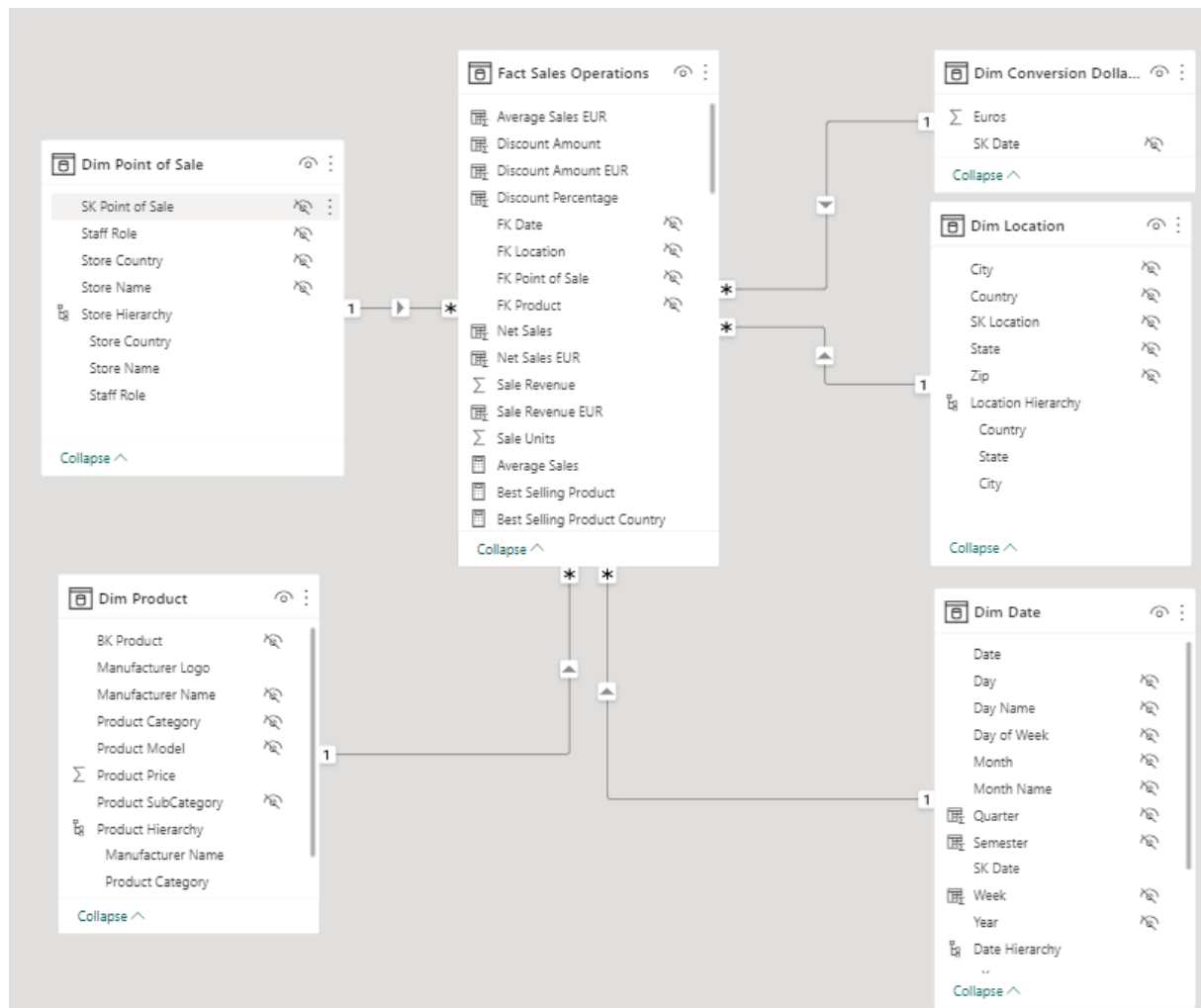


Figure 9 – “Brikint dimensional model”

### 7.3. GRANULARITY

Taking into consideration the business needs previously identified and that we aim to provide an answer, is now important to define granularity, which basically refers to the level of detail from our data. In order to find the right granularity for each dimension we had to find the right balance between efficiency and detail, because too much detail does not mean necessarily a better report, since it may be very time consuming and most not easily provide us the patterns and behaviors we are looking for.

The list, per dimension, of our granularity levels (organized with an increasing level of detail) is the following:

- **Dimension Date** – Year, Semester, Quarter, Month, Week and Days;
- **Dimension Location** – Country, Region, State, City and Zip;
- **Dimension Point of Sale** – Country, Name and Role;
- **Dimension Product** – Manufacturer Name, Product Category, Product Sub Category and Product Model;

## 7.4. DESIGN OVERVIEW

The below table represents the main components of the data model:

Activity	Description
1. Business processes identified (1)	Sales transactions
2. Key measures identifies (2)	Sales revenue and sales quantity
3. Dimensions Identified (6)	Date, Customer Location, Product, Point of Sale, and Currency
4. Hierarchies Identified (3)	<b>Store Country --&gt; Store Name--&gt; Staff Role</b> ( <i>in Point of Sale Dimension</i> ) <b>Year --&gt; Semester → Quarter--&gt;Month --&gt; Week</b> ( <i>in Date Dimension</i> ) <b>Category --&gt; Subcategory --&gt; Model</b> ( <i>in Product Dimension</i> )
5. Fact tables created (2)	<b>Sales Operations</b> ( <i>date key, product key, point of sale key, manufacturer key, customer location key, date, sales revenue, and sales quantity</i> )
6. Dimension tables created (6)	<b>Customer Location</b> ( <i>Country, Region, City</i> ) <b>Product</b> ( <i>Model, Subcategory, and Category</i> ) <b>Date</b> ( <i>Year, Semester, Quarter, Month, Week, Day</i> ) <b>Point of Sale</b> ( <i>Store Country, Store Name, and Role</i> ) <b>Currency</b> ( <i>Code, Symbol and Country and Currency Name</i> )
7. Relationships defined	o A <b>one-to-many relationship</b> between the fact table “Sales Operations” and the dimension “Product”, since one product different can be sold multiple times
	o A <b>one-to-many relationship</b> between the fact table “Sales Operations” and the dimension “Point of Sales”, since on point of sale (store/role) can have multiple sales
	o A <b>one-to-many relationship</b> between the fact table “Sales Operations” and the dimension “Manufacture”, since one manufacturer can supply in multiple sales
	o A <b>one-to-many relationship</b> between the fact table “Sales Operations” and the dimension “Date”, since one date can have multiple sales
	o A <b>one-to-many relationship</b> between the fact table “Sales Operations” and the dimension “Customer Location”, since one location can be a destination of multiple sales

Activity	Description
	o A <b>one-to-one relationship</b> between the fact table “Exchange Rate” and the dimension “Date”, since one date can have multiple exchanges rates for different currencies.
	o A <b>one-to-one relationship</b> between the fact table “Exchange Rate” and the dimension “Currency”, since one currency has a different rate depending on the transaction date.

**Table 1** – “Description of the data model”

## 7.5. DIFFERENT DATATYPES

The data types are represented in the table below:

Table Reference	Column Reference	Data Type
Conversion Dollars to Euros	Euros	Whole Number
	FK Data	Date
Dim PointOfSale	SK Point of Sale	Whole Number
	Store Name	Text
	Staff Role	Text
	Store Country	Text
Dim Date	SK Date	Whole Number
	Date	Date
	Day	Whole Number
	Day of Week	Whole Number
	Day Name	Text
	Month	Whole Number
	Month Name	Text
	Year	Whole Number
Dim Location	SK Location	Whole Number
	Country	Text

Table Reference	Column Reference	Data Type
	City	Text
	Region	Text
	State	Text
	District	Text
Dim Product	BK Product	Whole Number
	Product Model	Text
	Product Subcategory	Text
	Product Category	Text
	Manufacturer Name	Text
	Manufacturer Logo	Text
	Product Price	Fixed Decimal Number
Dim Currency	SK Currency	Whole Number
	Currency Code	Text
	Symbol	Text
	Country and Currency	Text
Fact SalesOperation	FK Product	Whole Number
	FK Date	Whole Number
	Date	Date
	FK Manufacturer	Whole Number
	FK Point of Sale	Whole Number
	FK Customer Location	Whole Number
	Units	Whole Number
	Revenue	Fixed Decimal Number

**Table 2 – “Data Types”**

## 7.6. MEASURES

After having **Star Schema model** designed, we have created and incorporated into 'Fact Sales Operations' 7 'functionality based' groups of new measures to widen the analytical capabilities of our data model and obtain deeper insights into our business data, allowing to respond to the existing business needs. Measures were built mainly to address and keep monitoring specific key performance indicators (KPIs):

### 1. Target Sales, basis for KPI calculation and evaluation criteria

1	Target Sales	<code>[PY Sales] * 1.15</code>
2	Target Performance	<code>DIVIDE([CY Sales] , [Target Sales])</code>
3	Target Annual Sales Variance	<code>[CY Sales] - [Target Sales]</code>

**Table 3 – “Measures - Target Sales”**

### 2. Market share, competition, contribution share KPI

1	Share Market	<code>VAR Revenue = SUM('Fact Sales Operations'[Sale Revenue])</code>  <code>VAR TotalRevenue = CALCULATE(SUM('Fact Sales Operations'[Sale Revenue]), ALLSELECTED('Fact Sales Operations'))</code>  <code>RETURN</code>  <code>DIVIDE(Revenue, TotalRevenue, 0)</code>
2	Top Manufacturer Share	<code>[Top Manufacturer Revenue]/[Total Net Sales]</code>
3	Top Market Share	<code>[Top Market Revenue]/[Total Net Sales]</code>
4	Top Product Share	<code>[Top Product Revenue]/[Total Net Sales]</code>
5	Top StaffRole Share	<code>[Top StaffRole Revenue]/[Total Net Sales]</code>
6	Top Store Share	<code>[Top Store Revenue]/[Total Net Sales]</code>

**Table 4 – “Measures - Market Share and Competition”**

### 3. Best / Top performers within relevant analytical perspective

1	Best Selling Product	<code>FIRSTNONBLANK (TOPN(1,VALUES('Dim Product'[Product Model]), SUM('Fact Sales Operations'[Net Sale])),1)</code>
2	Best Selling Product Country	<code>VAR MaxSale = CALCULATE (     SUM('Fact Sales Operations'[Net Sales]),     ALLSELECTED('Dim Location'[Country])) RETURN     FIRSTNONBLANK (         TOPN (             1,             VALUES('Dim Product'[Product Model]),             CALCULATE(SUM('Fact Sales Operations'[Net Sales]), ALL('Dim Location'[Country])),             DESC         ),         1     )</code>
3	Top 5 Stores	<code>IF([Rank Store]&lt;=5, [Total Net Sales], BLANK())</code>
4	Bottom 5 Store	<code>IF(RANKX(ALL('Dim Point of Sale'[Store Name]), [Total Net Sales],, ASC)&lt;=5, [Total Net Sales],BLANK())</code>
5	Top Manufacturer Revenue	<code>MAXX(     TOPN(1,     SUMMARIZE('Dim Product','Dim Product'[Manufacturer Name], "Rev", [Total Net Sales]), [Rev],DESC),[Rev])</code>
6	Top Manufacturer Units	<code>VAR TopManufacturer = [Top Ranked Manufacturer Name] RETURN     CALCULATE(     SUM('Fact Sales Operations'[Sale Units]),     'Dim Product'[Manufacturer Name] = TopManufacturer)</code>



7	Top Market Revenue	<pre> MAXX(      TOPN(1,          SUMMARIZE('Dim Location','Dim Location'[Country], "Rev", [Total Net Sales]),         [Rev],DESC),[Rev]) </pre>
8	Top Market Units	<pre> VAR TopMarket = [Top Ranked Market]  RETURN  CALCULATE(      SUM('Fact Sales Operations'[Sale Units]),      'Dim Location'[Country] = TopMarket) </pre>
9	Top Product Revenue	<pre> MAXX(      TOPN(1,          SUMMARIZE('Dim Product','Dim Product'[Product Model], "Rev", [Total Net Sales]),         [Rev],DESC),[Rev]) </pre>
10	Top Product Units	<pre> VAR TopProduct = [Top Ranked Product]  RETURN  CALCULATE(      SUM('Fact Sales Operations'[Sale Units]),      'Dim Product'[Product Model] = TopProduct) </pre>
11	Top Ranked Manufacturer Logo	<pre> CALCULATE(SELECTEDVALUE('Dim Product'[Manufacturer Logo] ),  TOPN( 1, 'Dim Product', [Total Net Sales], DESC ) ) </pre>
12	Top Ranked Manufacturer Name	<pre> CALCULATE(SELECTEDVALUE('Dim Product'[Manufacturer Name] ),  TOPN( 1, 'Dim Product', [Total Net Sales], DESC ) ) </pre>
13	Top Ranked Market	<pre> CALCULATE(SELECTEDVALUE('Dim Location'[Country] ),  TOPN( 1, 'Dim Location', [Total Net Sales], DESC ) ) </pre>

14	Top Ranked Product	<code>CALCULATE(SELECTEDVALUE('Dim Product'[Product Model] ),  TOPN( 1, 'Dim Product', [Total Net Sales], DESC ) )</code>
15	Top Ranked StaffRole	<code>CALCULATE(SELECTEDVALUE('Dim Point of Sale'[Staff Role] ),  TOPN( 1, 'Dim Point of Sale', [Total Net Sales], DESC ) )</code>
16	Top Ranked Store	<code>CALCULATE(SELECTEDVALUE('Dim Point of Sale'[Store Name] ),  TOPN( 1, 'Dim Point of Sale', [Total Net Sales], DESC ) )</code>
17	Top StaffRole Revenue	<code>MAXX(  TOPN(1,  SUMMARIZE('Dim Point of Sale','Dim Point of Sale'[Staff Role], "Rev", [Total Net Sales]), [Rev],DESC),[Rev])</code>
18	Top StaffRole Units	<code>VAR TopStaff = [Top Ranked StaffRole]  RETURN  CALCULATE(  SUM('Fact Sales Operations'[Sale Units]),  'Dim Point of Sale'[Staff Role] = TopStaff)</code>
19	Top Store Revenue	<code>MAXX(  TOPN(1,  SUMMARIZE('Dim Point of Sale','Dim Point of Sale'[Store Name], "Rev", [Total Net Sales]), [Rev],DESC),[Rev])</code>
20	Top Store Units	<code>VAR TopStore = [Top Ranked Store]  RETURN  CALCULATE(  SUM('Fact Sales Operations'[Sale Units]),  'Dim Point of Sale'[Store Name] = TopStore)</code>

**Table 5 – “Measures - Top”**

#### 4. Sales growth evaluation

Abbreviations used : CM (current month), CY (current year), MoM (month-on-month), PM (previous month), PY (previous year), YoY (year-on-year)

1	Average Sales	<code>AVERAGE('Fact Sales Operations'[Net Sale])</code>
2	CM Sales	<code>Var CM = MAX('Dim Date'[Month])</code>  <code>RETURN</code>  <code>CALCULATE([Total Net Sales], 'Dim Date'[Month]=CM)</code>
3	CY Sales	<code>VAR CY = MAX('Dim Date'[Year])</code>  <code>RETURN</code>  <code>IF(</code> <code>    CALCULATE(SUM('Fact Sales Operations'[Net Sale]), 'Dim Date'[Year] = CY) =</code> <code>    BLANK(),</code>  <code>    0,</code>  <code>    CALCULATE(SUM('Fact Sales Operations'[Net Sale]), 'Dim Date'[Year] = CY))</code>
4	CY Units	<code>VAR CY = MAX('Dim Date'[Year])</code>  <code>RETURN</code>  <code>IF(</code> <code>    CALCULATE(SUM('Fact Sales Operations'[Sale Units]), 'Dim Date'[Year] = CY) =</code> <code>    BLANK(),</code>  <code>    0,</code>  <code>    CALCULATE(SUM('Fact Sales Operations'[Sale Units]), 'Dim Date'[Year] = CY))</code>
5	PM Sales	<code>CALCULATE([Total Net Sales], PARALLELPERIOD('Dim Date'[Date],-1,MONTH))</code>

6	PY Sales	<pre> VAR MaxYear = MAX('Dim Date'[Year])  RETURN  CALCULATE(      SUM('Fact Sales Operations'[Net Sale]),      FILTER(ALL('Dim Date'), 'Dim Date'[Year] = MaxYear - 1)) </pre>
7	YoY Growth (year on year)	<pre> DIVIDE([CY Sales] - [PY Sales], [PY Sales], BLANK()) </pre>
8	YoY Sales Variance (	<pre> [CY Sales] - [PY Sales] </pre>

**Table 6 – “Measures - Abbreviations”**

#### 6. Count of unique categories within relevant analytical perspective

1	Unique Countries	DISTINCTCOUNT('Dim Location'[Country])
2	Unique Manufacturers	DISTINCTCOUNT('Dim Product'[Manufacturer Name])
3	Unique Products	DISTINCTCOUNT('Fact Sales Operations'[FK Product])
4	Unique Stores	DISTINCTCOUNT('Dim Point of Sale'[Store Name])

**Table 7 – “Measures - Uniques”**

#### 7. Totals and aggregation

1	Max Year Sales (All Manufacturers)	<pre> VAR AllManufacturers =     ALL('Dim Product'[BK Product])  VAR MaxSales =     CALCULATE(         MAXX(             SUMMARIZE(                 'Fact Sales Operations',                 'Dim Date'[Year],                 'Dim Product'[Manufacturer Name],                 "Total Sales", SUM('Fact Sales Operations'[Sale Revenue])             ),             [Total Sales]         ),         AllManufacturers )  RETURN     MaxSales </pre>
2	Total Discount	<pre> IF(ISBLANK(SUM('Fact Sales Operations'[Discount Amount])), 0, SUM('Fact Sales Operations'[Discount Amount])) </pre>
3	Total Net Sales	<pre> IF(ISBLANK(SUM('Fact Sales Operations'[Net Sale])), 0, SUM('Fact Sales Operations'[Net Sale])) </pre>
4	Total Orders	<pre> COUNTROWS('Fact Sales Operations') </pre>
5	Total Sales	<pre> IF(ISBLANK(SUM('Fact Sales Operations'[Sale Revenue])), 0, SUM('Fact Sales Operations'[Sale Revenue])) </pre>
6	Total Units	<pre> IF(ISBLANK(SUM('Fact Sales Operations'[Sale Units])), 0, SUM('Fact Sales Operations'[Sale Units])) </pre>

**Table 8 – “Measures - Totals”**

## 8. Technical measures used to implement the above functions

1	Previous Year	[Year Max]-1
2	Year Max	Max('Dim Date'[Year])
3	Year Min	Min('Dim Date'[Year])

**Table 9 – “Measures - Years”**

To address Brikint’s request to perform currency conversion from US Dollars into Euro - all relevant calculated fields used for financial performance calculations were also created in EURO.

### 7.7. CALCULATED COLUMNS

3 calculated columns was created to address Brikint’s requirement to calculate Sales Discount and Sales net of discounts :

1. Discount Percentage (%),
2. Discount Value
3. Net Sales Amount ( being Sale amount minus Discount provided to customer in accordance with Discount policy in place with Brikint).

Table below represents the formulas used for calculation.

1	Discount Amount	'Fact Sales Operations'[Sale Revenue]*'Fact Sales Operations'[Discount Percentage]
2	Discount Percentage	SWITCH(  TRUE(),  'Fact Sales Operations'[Sale Units] <= 10, 0,  'Fact Sales Operations'[Sale Units] <= 20, 0.05,  'Fact Sales Operations'[Sale Units] <= 50, 0.10,  'Fact Sales Operations'[Sale Units] > 50, 0.15)
3	Net Sale	'Fact Sales Operations'[Sale Revenue]-'Fact Sales Operations'[Discount Amount]

**Table 10 – “Calculated Columns”**

## 8. MODEL OPTIMIZATION

### 8.1. HIERARCHIES

Below are 3 hierarchies we have established, which we will develop into more detail:

- **Store Country --> Store Name --> Staff Role** (*in Point of Sale Dimension*)
  - Within each country, there are multiple stores and within each store there are multiple staff roles, although these may be the same for the various countries. Our Staff Role represents the last layer;
- **Year --> Semester → Quarter → Month --> Week → Day** (*in Date Dimension*)
  - Naturally, we have a division of years in semester (divided by 2); a subsequent division in quarters (divided by 2); a subsequent division in months (divided by 3); a subsequent division in weeks (divided by 4) and finally, a subsequent division in days, which represents the highest level of detail on this hierarchy;
- **Manufacturer Name → Product Category --> Product Subcategory --> Product Model** (*in Product Dimension*)
  - For each manufacturer, there are multiple product categories; within each there are multiple Subcategories and finally, subdividing into product models. Our Model represents the last layer;

## 8.2. HIDDEN COLUMNS

In our final report, we decided to hide some columns, for a clearer and easier reporting analyzes, avoiding duplication, since most of the hidden columns, were already reflected as a Business Key in another column, or represented either by some new measure or an additional column. Our hidden columns are distributed for the following dimensions:

- **Dimension Conversion Dollars to Euros**
  - SK Date;
- **Dimension Date**
  - Day Name, Day of Week and Month Name;
- **Dimension Location**
  - SK Location and ZIP;
- **Dimension Point of Sale**
  - SK Point of Sale;
- **Dimension Product**
  - BK Product and Manufacturer Name;
- **Fact Sales Operations**
  - FK Date, FK Location, FK Point of Sale and FK Product.

## 8.3. FORMATTING

Constant and uniform format was used throughout our dashboards, facilitating the user comprehension and interpretation and speed up the time to obtain the main conclusions out of it:

- Included a **four different box options** on each slide, allowing the user to easily move either to the next previous slide or the next one, to comeback to the Overview page, and to apply a visualization filter over date and another one (depending on the slide in question);
- Applied a **blue color** tone to all the slide headlines, column titles and the space involving the majority of the cards;

- Have used **letter type Segoe UI** for all graphs titles, varying the letter sizes between 14 and 16, according the relevancy;
- Applied a **letter size of 36** to each slide main title, and a **letter size of 16** to highlight the “tops” from each category (represented by card visuals);
- **Standartized our slides, always including six visuals (counting with both graphs and tables)** per slide (which was set-up as our initial maximum limit), in order to avoid busy slides. We defend that we must be able to do more with few, carefully selected information. More graphs does not mean more information. During the process of designing this visuals, we carefully made sure to:
  - create headers centered, both vertically and horizontally;
  - include our data with, at maximum, two decimal places, avoiding very busy tables. When dealing with big amounts (in millions, mostly), we did not even created decimal places, for a matter of clarity and objectivity;
- Included the **seven different country flags** where the company operates, which works as a filter, and automatically filters our data analysis according with our selection;

We believe that these points above mentioned, represents an interactive and dynamic presentation, being at the same time, very goal-orienting to address our initial questions.

## 9. VISUALS USED

We have focus our analysis on few visuals, which we believe were more appropriate to our needs:

- **Cards.** This visual type allows us to display key metrics and summary information in a concise and visually appealing manner, providing a quick overview of important data points.
- **Line Charts.** This visual type allows us to visualize trends and patterns over time, making it easy to identify changes, fluctuations, and seasonality in our data.
- **Bar charts (clustered, stacked, column).** These visual types allow us to compare values across different categories or groups, making it simple to identify variations and rankings within our data.
- **Basic area chart.** This visual type allows us to visualize the distribution of values over time, highlighting the magnitude and changes in data trends.
- **Tables.** This visual type allows us to present detailed data in a structured and organized format, facilitating easy exploration, sorting, and filtering of information.
- **Enlighten World Flags.** This visual type allows us to represent countries using their respective national flags, adding a visually appealing element to our reports and enabling country-based analysis.
- **Map.** This visual type allows us to geographically represent data and explore regional patterns, providing insights into location-based trends and spatial relationships.
- **Pie chart and donut chart.** These visual types allow us to showcase the composition and proportion of different categories within a dataset, making it easy to understand relative proportions and identify major contributors.
- **Interactive filters with the logo of manufacturers.** This visual type allows us to use custom images or logos of manufacturers as interactive filters, enabling users to filter and analyze data based on specific manufacturers or brands.



## 10. ANALYTICAL REPORTS PREPARED TO ADDRESS BUSINESS NEEDS

To address business needs described in section 4, we have prepared 7 analytical reports (including the “Overview” one, representing the cover and main business achievements):

1. Overview;
2. Product Analysis;
3. Manufacturer Analysis;
4. Stores Analysis;
5. Market Analysis;
6. Staff Analysis
7. Sales Analysis

**Product Analysis** contains different visualizations (charts, tables etc):

1. Filters by manufacturers and countries with logos and flags allows us to select specific country or manufacturer for analysis.
2. On the top right corner of the dashboard we have filters by date and store name.
3. On the top left side, there are three cards displaying information about the top Product Model, including its name, sales value, units, and share. The content of these cards changes dynamically based on the selected filters mentioned above.
4. In the center of the dashboard, there is stacked bar chart that visualizes the names and sales of the top Product Models.
5. On the right side line chart which represents annual dynamic of total sales by subcategory of products.
6. Below the table with information by top 5 Product Models about selected current year sales value, sales in units, growth of sales value to previous year in percentage and target performance in percentage as well.

All graphs and tables in the dashboard change dynamically depending on selected filters by countries, manufacturers, stores and time period.

### **Goal:**

Product Analysis report is addressed to Product management team and allows them to monitor target performance and sales dynamics by products, look for development potential of products across countries, stores and manufacturers, identify weaknesses and potential threats by products and look for opportunities of implementing product KPI for sales team.

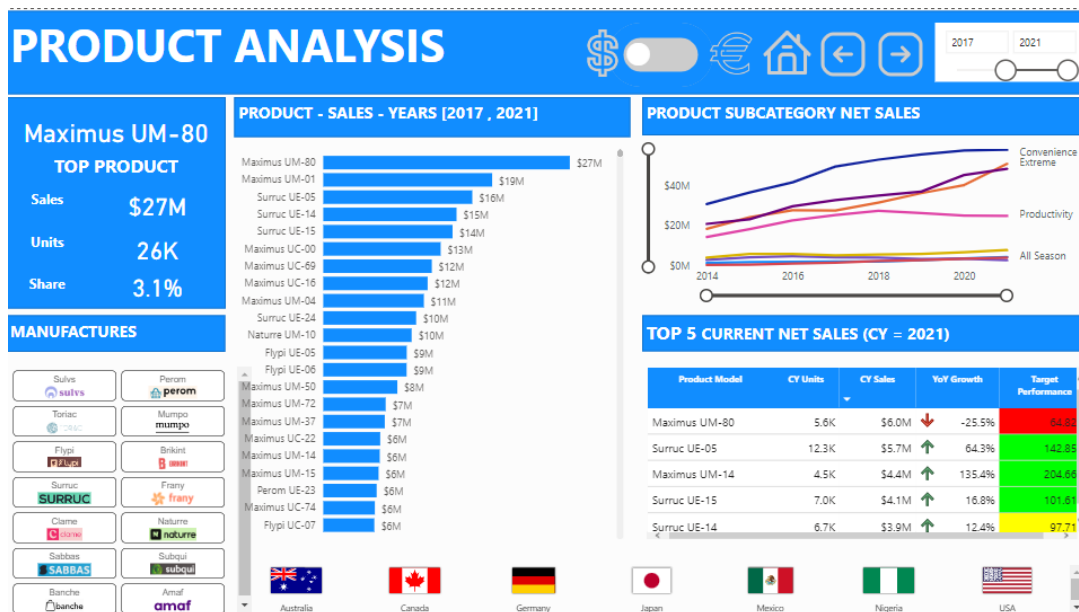


Figure 10 – "Product Analysis Dashboard"

Manufacturer Analysis contains following visualizations:

1. Filters by manufacturers and countries with logos and flags allows us to select specific countries or manufacturers for analysis.
2. On the top right corner of the dashboard we have filters by date and store name.
3. On the top left side, there are three cards displaying information about the top Manufacturer and its logo, including sales value, units and share. The content of these cards changes dynamically based on the selected filters mentioned above.
4. In the center of the dashboard, there is a table with information containing Manufacturer name, Total Net Sales, Share Market, Total Units, number of Unique Products and Best Selling Product by manufacturers.
5. Below the line chart which represents Total Net Sales dynamics by years and by manufacturers.
6. On the right side of the dashboard there is a bar chart representing Total Net Sales volume and number of orders by Product subcategories.
7. On the right side below there is a table with top 5 Manufacturers for selected current year. This table contains Manufacturer Name, current year sales value and sales units, sales growth of current year to previous year and target performance, the both are in percent.

All graphs and tables in the dashboard change dynamically depending on selected filters by countries, manufacturers, stores and time period.

#### Goal:

Manufacturer Analysis report is addressed to Sales management team and allows them to monitor target performance and sales dynamics and trends by manufacturers, to analyse share of our manufacturer and competitors, to look for potential of sales growth in different countries and product categories, to analyze the top products by different manufacturers and countries.

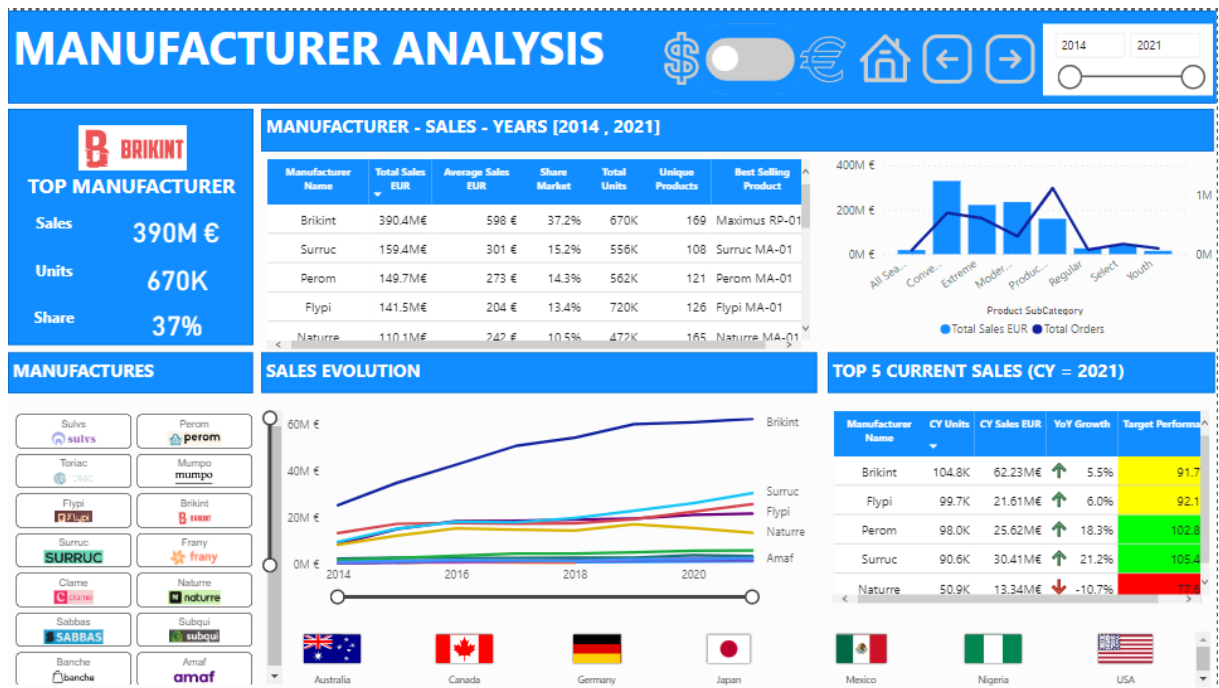


Figure 11 – “Manufacturer Analysis Dashboard”

**Stores Analysis**, which contains 6 different visualizations and the 7 different countries flags, that allows the user to select the country of analysis. On the top right corner of the slide, we have the possibility to filter our analysis by date.

1. The **first** visualization contains three cards informing us about the top main indicators from the top store of the selected country: one for the total sales value (in M Euros), one for the total sales units (in K units) and one for the total share of this store in the entire country market for the specific selected data (in %).
2. The **second** visualization we used a table with 5 different columns with information per store of the selected country: total net sales, share market, total units, total orders and best selling products. This table was sorted by the highest value in total net sales and the values are cumulative from the period of analysis (from the start to the end year inclusively).
3. The **third** visualization represents a histogram (line and stacked column chart) displaying the total orders per Product Category over our selected filters.
4. The **fourth** visualization represents a lateral histogram (clustered bar chart) displaying the total net sales value also per Product Category over our selected filters, similarly to the previous visual.
6. And finally, the **fifth** and last visualization is a table that displays (information exclusively about the current year of analysis), per store, the current year units, sales amounts, growth since last period and evaluation in comparison with target. Note that we incorporated an up and a down arrow on, representing positives and negatives growth value; and incorporated a red, yellow or green background on target performance, representing values in the range of 0-50%, 50%-85% and 85% - 100%, respectively.

## Goal:

Stores Analysis report is designed to provide business decision-makers (board members) and local management team and allows them to closely monitor sales evolution at a store level. It provides information about all stores of any specific country with a wide variety of metrics, namely the Market Share and the respective Best Selling Product. Additionally, for the top sales of the selected year, it provides comparison between the current performance with the target and growth information.

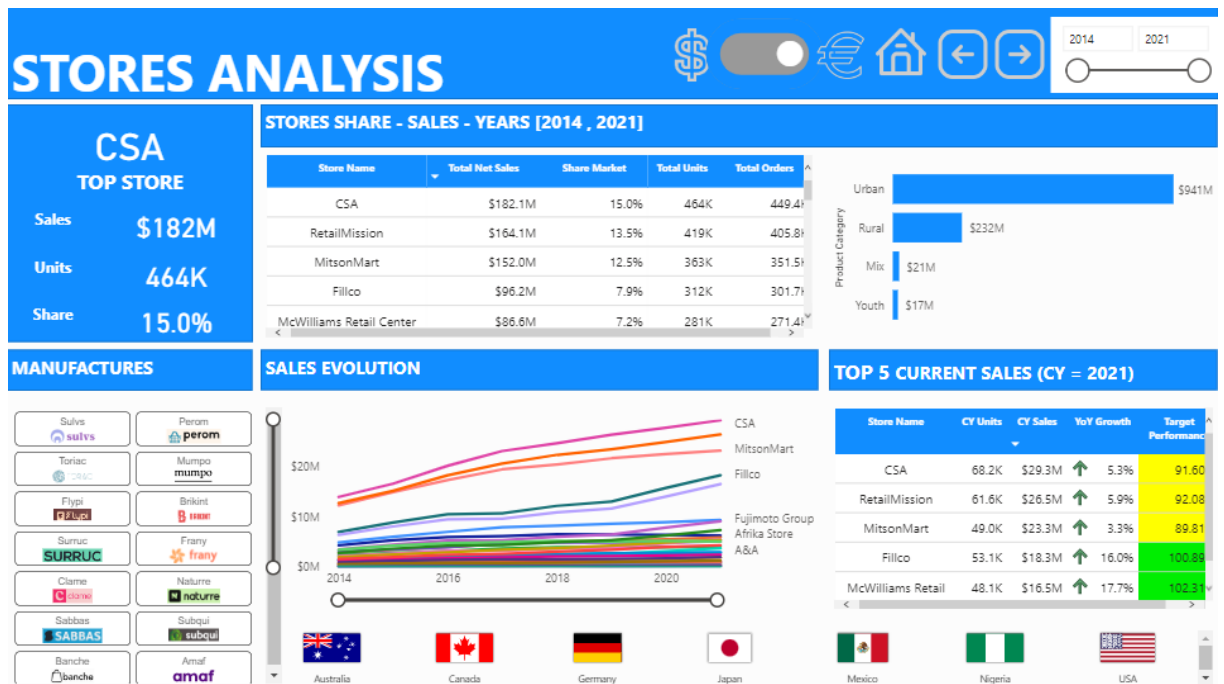


Figure 12 – “Stores Analysis Dashboard”

**Staff Analysis**, which contains 5 different visualizations and the 7 different countries flags, that allows the user to select the country and the manufacturer of analysis. On the top right corner of the slide, we have the possibility to filter our analysis by date.

1. The **first** visualization contains three cards informing us about the top main indicators from the top staff role of the selected country: one for the total sales value (in M Euros), one for the total sales units (in K units) and one for the total share of this staff role in the entire market (in %).
2. The **second** visualization we used a table with 5 different columns with information per staff role of the selected country and manufacturer: total net sales (in M €), share market (in %), total sales units, the amount of unique products and the identification of the best selling products. This table was sorted by the highest value in total net sales and the values are cumulative from the period of analysis (from the start to the end year inclusively).
3. The **third** visualization represents a histogram (line and stacked column chart) displaying the total net sales per Product SubCategory, using our selected filters.
4. The **fourth** visualization represents a line chart, displaying the Total Net Sales value throughout the different years.

5. And finally, the **fifth** and last visualization is a table that displays (information exclusively about the current year of analysis), per staff role, the current year units, sales amounts, growth since last period and evaluation in comparison with target. Note that we incorporated an up and a down arrow on, representing positives and negatives growth value; and incorporated a red, yellow or green background on target performance, representing values in the range of 0-50%, 50%-85% and 85%-100%, respectively.

### Goal:

Staff Analysis report is designed to provide business decision-makers (board members) and human resources team and allows them to closely monitor employees' performance evolution at a role level. It provides information filtered by manufacturer and by country and detailed by about all stores of any specific country. Per role, we provide information for the selected year about the current year sales amounts and units, and a comparison between the current performance with the target and growth information.

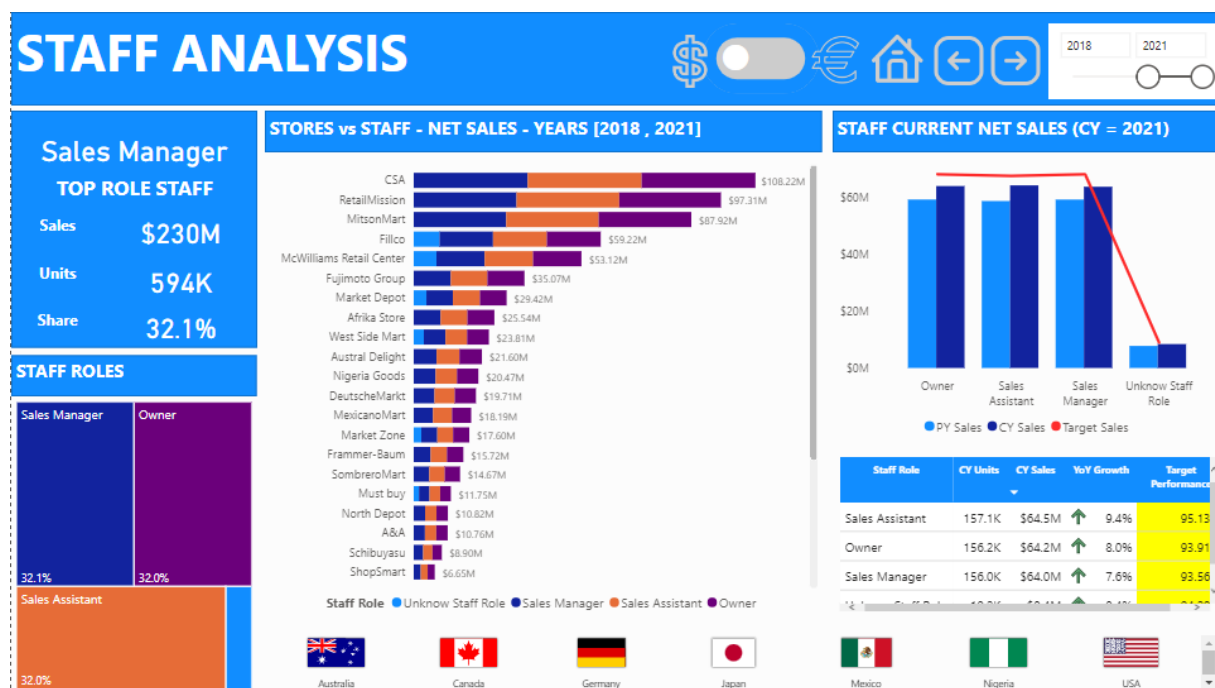


Figure 13 – "Staff Analysis Dashboard"

Market Analysis report contains six visualisations and filters:

1. Filters by manufacturers and countries with logos and flags allows us to select specific countries or manufacturers for analysis across all visualisations.
2. On the top right corner of the dashboard we have filters by date and store name.
3. There are five cards displaying information about specific Markets(countries) including sales value, share of this market, numbers of regions and cities, number of stores. The content of these cards changes dynamically based on the selected filters mentioned above.
4. Visualisation map represents geographical view of Net Sales by countries. There are pie charts by Product category for each country.

5. The next visualisation - donut chart represents Total Net Sales by products category, by shares in percent.
6. The table represents current year sales value and units , sales growth of current year to previous year and target performance , the both are in percent by countries.
7. Below the line chart which represents Total Sales dynamics by years and by countries.
8. The last two charts represent Top-5 and Bottom-5 Stores by Total Sales for selected country or across several/all countries for a selected period of time.

All graphs and tables in the dashboard change dynamically depending on selected filters by countries, manufacturers and time period.

### Goal:

Market Analysis report is addressed to Sales management team and allows them to monitor target performance, sales dynamics and trends by countries, to analyse market share of country, to have a geographical view on sales regarding our sales and competitors sales, product categories and stores.

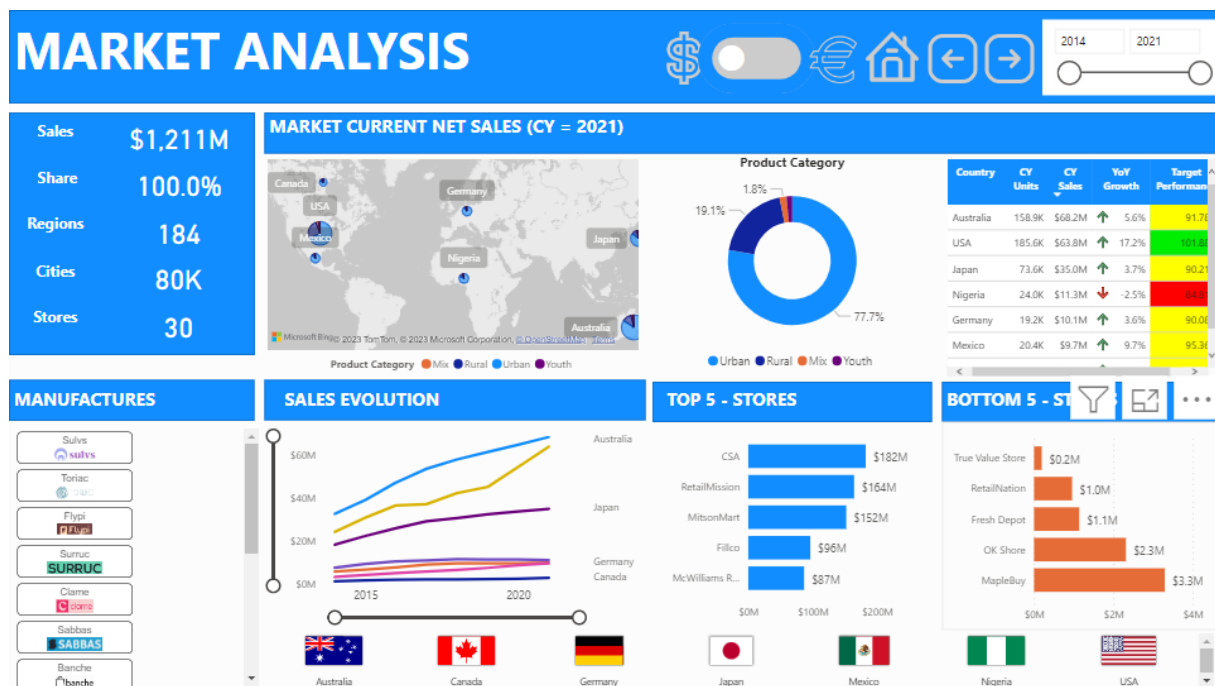


Figure 14 – “Market Analysis Dashboard”

Sales Analysis report contains six visualizations and filters:

1. Filters by manufacturers and countries with logos and flags allows us to select specific countries or manufacturers for analysis across all visualizations.
2. On the top right corner of the dashboard we have filters by date and store name.
3. There are three cards displaying information about Top store by sales including sales value, sales units, share of this store. The content of these cards changes dynamically based on the selected filters mentioned above.
4. Area chart "Sales evolution by selected time period" represents sales volume and dynamics of sales.

5. The table represents current year sales value and units , sales growth of current year to previous year and target performance , the both are in percent, discount amount by year..
6. The line chart "Sales Forecast" shows sales volume dynamics for selected periods of time and predicted sales values for future periods based on historical data.

#### Goal:

Sales Analysis report is addressed to Sales management team and allows them to have sales forecast based on historical data by countries, manufacturers and stores, to monitor target performance, sales dynamics and trends by years.

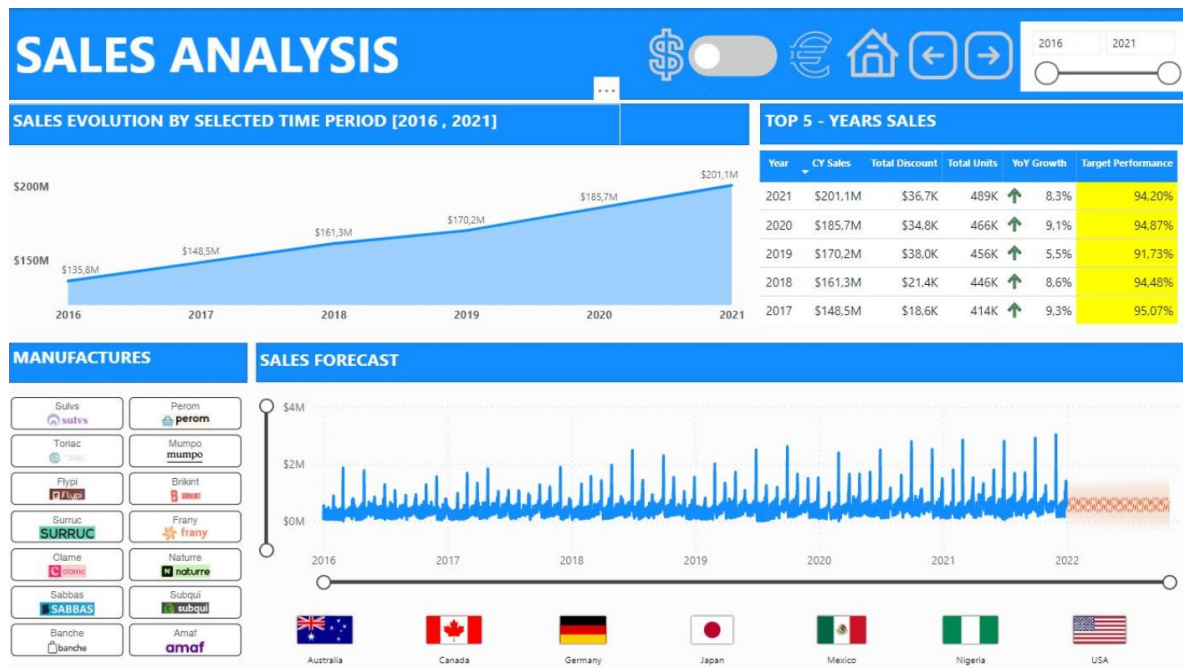


Figure 15 – “Sales Analysis Dashboard”

## 11. CONCLUSION

We believe that functionality and analytical opportunities of our model together with analytical reports we have prepared adequately responds to the business needs of Brikint and could be used as a starting point of further development of internally generated data analysis to improve the speed and quality of data driven decision making process in **Brikint**. Increased volume and quality of analytical information will assist **Brikint** in achievement of its tactical and strategic targets.