University of Pisa

Department of Computer Science
MSc in Data Science & Business Informatics

GROUP_ID_778

LABORATORY OF DATA SCIENCE PROJECT

Michele Dicandia 657494

m.dicandia1@studenti.unipi.it

Academic year 2023/2024

CONTENTS

1. INTRODUCTION

This report documents the process and outcomes of the data analysis project based on the provided dataset, primarily composed of the computer_sales.csv file containing computer sales data and the geography.xml file containing geospatial information related to the sales. The main objectives of the project include the construction of a data warehouse, the development of an ETL (Extract, Transform, Load) process, the creation of a Data Cube, the implementation of MDX queries, and the development of a dashboard for data visualization. The project aimed to achieve several key objectives:

1. Construction of a Data Warehouse for efficient data management.
2. Development of an ETL (Extract, Transform, Load) process for data processing.
3. Creation of a Data Cube for multidimensional analysis.
4. Implementation of MDX queries for extracting relevant information.
5. Development of an intuitive dashboard for data visualization.

The primary dataset, represented by the computer_sales.csv file, provides a comprehensive overview of computer sales from March 2013 to April 2018, including details on sales transactions and the hardware specifications of the sold PCs. The addition of the geography.xml file further enriches the dataset with relevant geographic information, which can be linked to the main fact table using their respective primary keys.

2. DATA WAREHOUSE CONSTRUCTION

2.1 ASSIGNMENT 1

To construct the data warehouse following the star schema provided in the reference PDF file, various tables were initially created in SQL Server Management Studio. Each table was defined by specifying the nature of the attributes and establishing appropriate relationships with the primary keys of the dimension tables through foreign keys in the fact table. Subsequently, a diagram was created to clearly visualize the connections of the star schema, as shown in *Figure 1*.

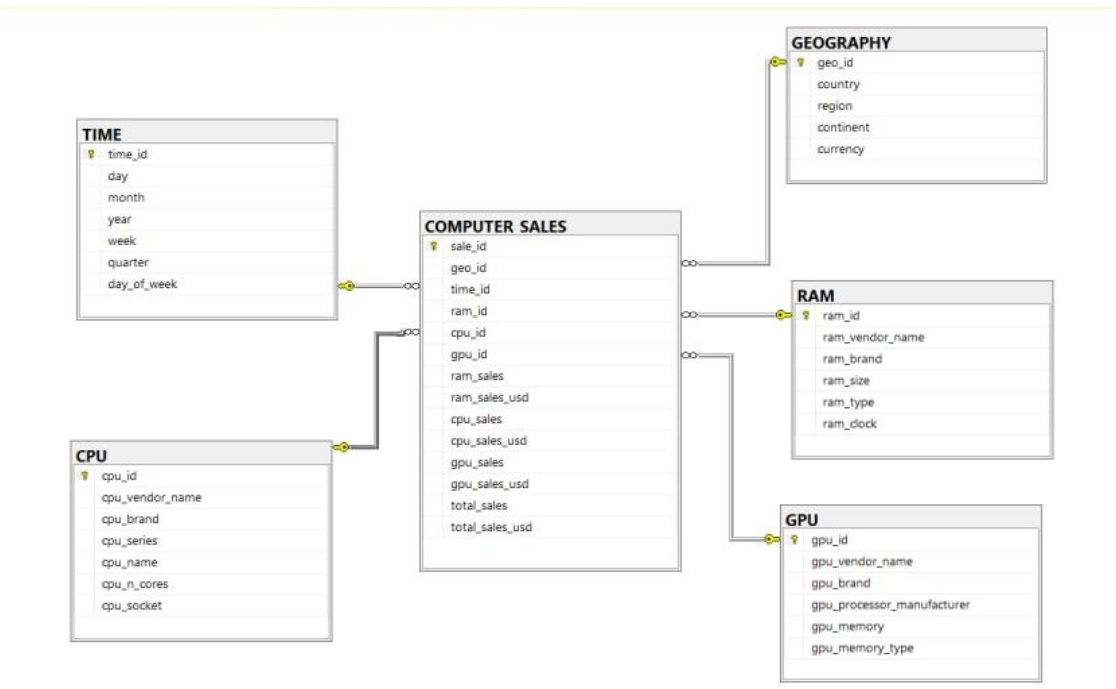


Figure 1

To complete this phase, Python was used to load the two files "computer_sales.csv" and "geography.xml". Several functions were implemented to manage and create the tables of the star schema more efficiently. For example, the "gen_table_distinct" function was used to extract unique values from specified columns and return them as a table of distinct sorted values, the "set_primary_key" function generated primary keys for the tables, and the "add_foreign_key" function added a foreign key to the first dataframe based on corresponding values in the common columns between the first and second dataframes.

The "computer_sales" dataset consists of 3,412,325 rows and 25 columns, and it does not contain attributes with null values. The tables "time_table", "cpu_table", "gpu_table", "ram_table", "geography_table", and "computer_sales_table" were created.

For the "time_table", values of the 'day', 'month', and 'year' variables were extracted from the "time_code" string in the 'yyyymmdd' format. Additionally, the "quarter" variable was derived from the month using the "get_quarter" function, while the "week" and "day_of_week" variables were obtained using the "to_datetime" function of the Pandas library.

Subsequently, an object acting as a dictionary named "dict_sales" was created using the "dict_from_header" function, where the keys represent the column names and the values correspond to the data columns. From this dictionary, relevant columns were extracted, to which the "gen_distinct_value" function was applied, followed by the "set_primary_key" function to generate unique keys for the table tuples.

The "cpu_table", "gpu_table", and "ram_table" were created by simply extracting the variables of interest from the "dict_sales" dictionary and applying the functions to generate distinct rows and insert primary keys.

For the "geography_table", the second file "geography.xml" was imported and joined with the main dataset using the "geo_id" key. The new dataset was then transformed into a dictionary, and the same functions were applied to the variables of interest. Furthermore, the nature of the "geo_id" variable was changed to string to ensure consistency in the key type.

Regarding the creation of the "computer_sales" table, new variables such as "cpu_sales_usd", "gpu_sales_usd", "ram_sales_usd", and "total_sales_usd" were introduced, derived from the respective variables identifying sales in the currency of the country of origin, through exchange rate coefficients. The purpose of these new variables is to enable comparison of sales in the same currency, i.e., with the same unit of measurement. Subsequently, foreign keys were mapped throughout the dataset, and using the same functions, variables of interest were extracted, and the primary key was created.

In conclusion, the tables created are as follows:

- "time_table": 1840 rows and 7 columns
- "cpu_table": 5321 rows and 7 columns
- "gpu_table": 3159 rows and 6 columns
- "ram_table": 11990 rows and 6 columns
- "geography_table": 75 rows and 5 columns
- "computer_sales": 3,381,715 rows and 14 columns

2.2 ASSIGNMENT 2

For the second phase of the assignment, which involved loading the data into Sql Server Management Studio, the necessary Python libraries were imported, including pyodbc, os, re, and tqdm. Additionally, a conversion of the value "Mike’s computer shop" to "Mikes computer shop" was performed in the attributes "sales_vendor_name", "gpu_vendor_name", and "cpu_vendor_name" due to the presence of apostrophes in the SQL string, in order to allow the data to be inserted into the remote table.

Subsequently, the class "Upload_table" was created. Using the credentials to access Sql Server Management Studio, this class was responsible for inserting the data into the tables after removing any pre-existing contents. A mini-batch strategy was adopted, involving periodic commits every 100 rows, which helped reduce the overall execution time of the insertion operation and minimized the risk of data loss in case of errors.

3. ETL PROCESS

In this chapter, the solution to a business question using Sql Server Integration Services (SSIS) with Visual Studio is described. An ETL workflow was developed to conduct a specific analysis.

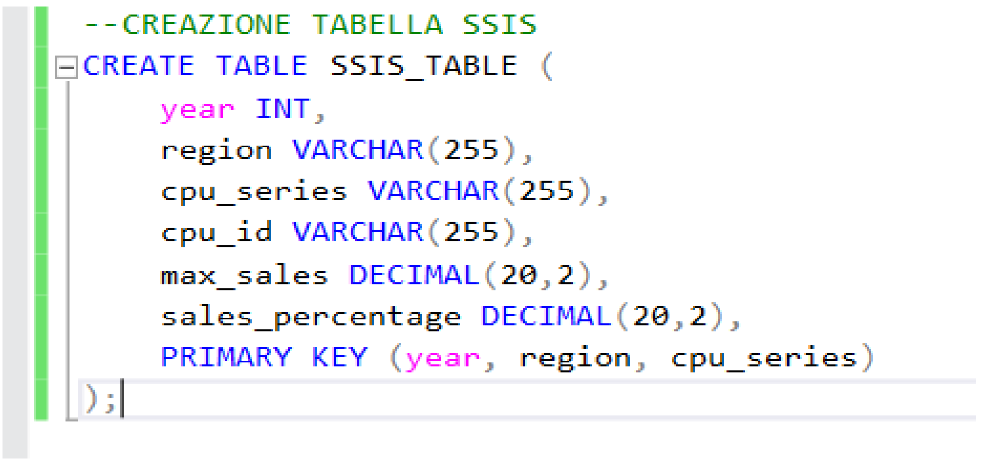
3.1 ASSIGNMENT 3

The business question posed is**: "For each year and region, identify the computer IDs associated with the highest sales of CPUs. Furthermore, augment the result by including the percentage of sales with respect to the total sales of all computers within the same CPU series."**

To address the assigned issue, we proceeded by initiating the Visual Studio development environment and creating an SSIS project. Within the control flow of this project, the "data flow" operation was inserted. Within this operation, data was extracted from the source table using the "OLE DB source" transformation, specifying the server connection and the source table. This transformation was subsequently named "caricamento COMPUTER SALES". Subsequently, a table sorting procedure was applied based on the "time_id" key using the sorting transformation. This operation was also repeated for the "TIME" table. Sorting was essential for applying the merge join transformation between the two tables, using the "time_id" key. During this phase, variables necessary for future analyses, namely "geo_id" and "cpu_id", which would represent the linking keys with other tables, were included. "Year" was added for analysis purposes and "total_sales_usd" for calculating "max_sales" and "percentage_sales", with the exclusion of all other variables not relevant for the analysis. Next, the "GEOGRAPHY" table was extracted using a similar process, sorting both tables according to the "geo_id" variable and merging the two tables using merge join. During this phase, variables such as "year", "cpu_id", "total_sales" were included, and "region" was added for analytical purposes, with the exclusion of "geo_id" as it was no longer needed. The same approach was followed for extracting the "CPU" table, sorting both tables according to "cpu_id", and merging them using merge join, including variables such as "year", "region", "total_sales", "cpu_id", and "cpu_series". Once a table containing all necessary variables was obtained, a grouping transformation was executed for "year", "region", "cpu_series", "cpu_id", and "total_sales_usd" setting it as the maximum value. This allowed obtaining a reduced table without duplicates, going from 3,381,715 to 5,399 rows. Subsequently, a multicast transformation was applied to reuse the same table in various transformations. Then, another grouping transformation was applied, this time only for "year", "region", "cpu_series", and "total_sales_usd" setting it as the maximum value and renaming the column as "max_sales". This way, the "cpu_id" with the highest sales for "year", "region", and "cpu_series" were identified. Next, another grouping transformation was executed for "year", "region", "cpu_series", calculating the sum of "total_sales_usd" and renaming it as "sum_sales". The tables were then sorted according to "year", "region", "cpu_series" and joined, thus obtaining "max_sales" and "sum_sales" in the same table. Using a derived column transformation, the ratio between "max_sales" and "sum_sales" was calculated, creating a new column named "sales_percentage". Then, a merge join of this new table with the complete table from the multicast was performed, sorted by "year", "region", and "cpu_series", thus identifying the "cpu_id" for each combination of "year", "region", and "cpu_series", along with their respective "max_sales" and "sales_percentage". Finally, a final OLE DB destination transformation was executed, loading the resulting table into a new table on Sql Server Management Studio, specifically created to host the data processed by the Data Flow in SSIS.

Figure 2 shows the sql code to create the table where the data is exported from the workflow in visual studio.

Figure 3 shows the data flow of the procedure carried out with visual studio, visible and reproducible by eliminating the data in the SSIS_table.



```sql
--CREAZIONE TABELLA SSIS
CREATE TABLE SSIS_TABLE (
    year INT,
    region VARCHAR(255),
    cpu_series VARCHAR(255),
    cpu_id VARCHAR(255),
    max_sales DECIMAL(20,2),
    sales_percentage DECIMAL(20,2),
    PRIMARY KEY (year, region, cpu_series)
);
```

Figure 2

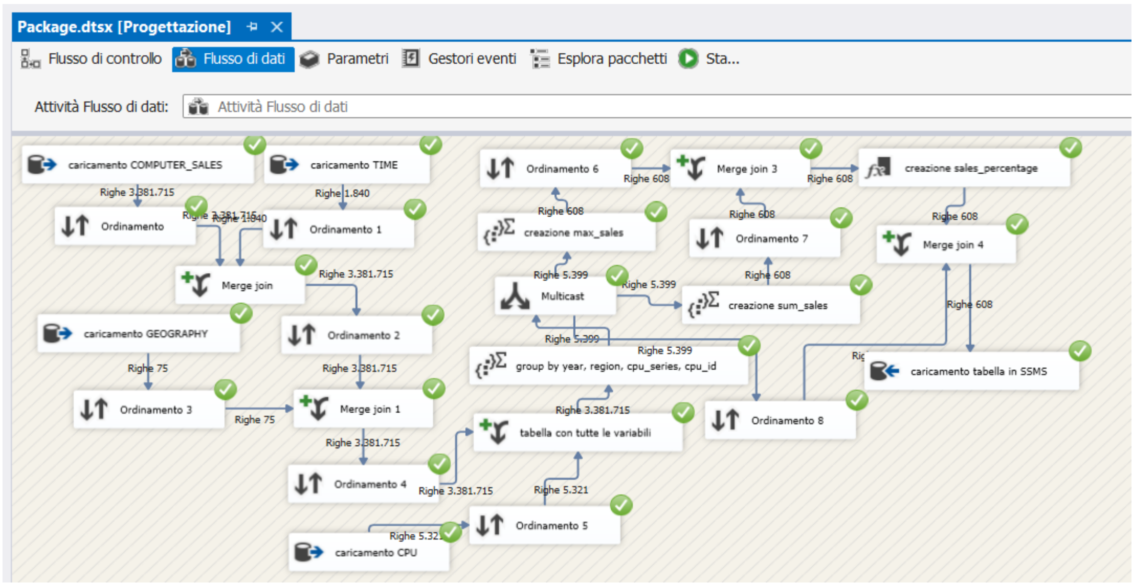


Figure 3

4. MULTIDIMENSIONAL DATA ANALYSIS

This section focuses on solving specific business questions using a datacube that will be generated from the existing database. To address these issues, we will make use of MultiDimensional eXpression (MDX) within SQL Server Management Studio.

4.1 ASSIGNMENT 4

The fourth task involves building a data cube from the data tables and integrating appropriate hierarchies. The process begins by creating a new Analysis Services project in Visual Studio, where a data source is configured with tables imported from SQL Server Management Studio. Subsequently, the cube is created, defining the necessary dimensions with their respective hierarchies to address business questions.

After data import, a data source view is created for "Group_ID_778_DB," followed by defining dimensions with their respective hierarchies. The CPU dimension includes (cpu_id → cpu_brand) [*figure 4*], the GPU dimension includes (gpu_id → gpu_brand) [*figure 5*], the RAM dimension includes (ram_id → ram_brand) [*figure 6*], while the GEOGRAPHY dimension comprises (geo_id → region → country → continent) [*figure 7*] . Additionally, the TIME dimension is enriched with new variables like "month_str," combining the month number with its name (e.g., 01-January, 02-February, ...) and "day_str," obtained by concatenating the current month in character format with the day (e.g., January-16, March-24, ...). These variables will be used to create a new hierarchy useful for subsequent analysis via the MDX SQL cube. An additional hierarchy is created with month_hier → year using the month_hier attribute created from the union of the year variable and the month variable (e.g., 2016-08, ...) which will serve for future analysis in assignment 6.

The TIME dimension will then have the following structure: (time_id → day_str → month_str); (month_hier → year) [figure 8]. No further hierarchies such as time_id → day → week → month → quarter → year were created as they were considered unnecessary for analytical purposes.

After defining dimensions, the cube creation process proceeds, including measures such as "ram_sales_usd," "cpu_sales_usd," "gpu_sales_usd," and the dimensions TIME, GEOGRAPHY, CPU, GPU, RAM.

Finally, the project is processed and deployed to the server, completing the operation with the deployment of the destination, specifying the server connection string and the database name "GROUP_ID_778_DB," through the process command.

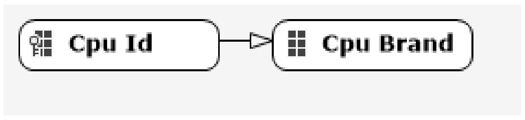


Figure 4

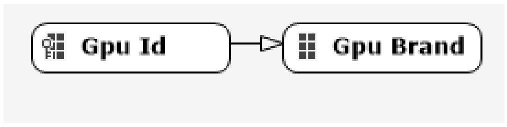


Figure 5

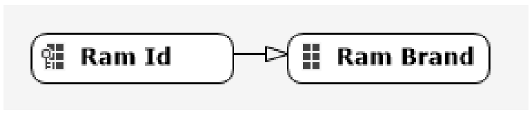


Figure 6

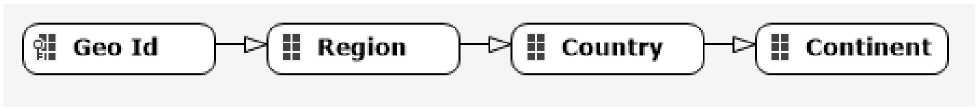


Figure 7

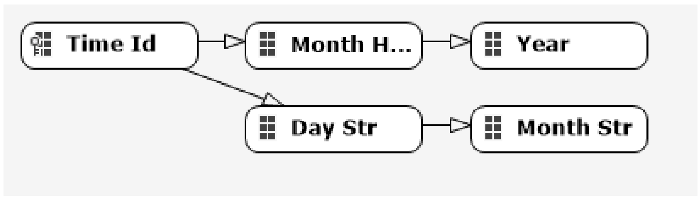


Figure 8

4.2 ASSIGNMENT 5

Once the cube design was completed, we were able to address some business questions using SQL Server Analysis Services (SSAS) and MultiDimensional eXpressions (MDX) in SQL Server Management Studio.

The business question was as follows: **"Produce the following analysis using an MDX query: Show the top 5 CPU, RAM, and GPU brands with respect to the monthly average sales for each region in Europe."**

To tackle this challenge, three different MDX queries were developed:

1. Find the monthly average sales of the top 5 CPU brands for each region in Europe.
2. Find the monthly average sales of the top 5 GPU brands for each region in Europe.
3. Find the monthly average sales of the top 5 RAM brands for each region in Europe.

The first problem was addressed with the following query:

```
WITH
  MEMBER [Measures].[AvgMonthlyCPUSales] AS
    Avg(
      Descendants(
         [TIME].[Gerarchia].CurrentMember,
         [TIME].[Gerarchia].[Month Str]
      ),
       [Measures].[Cpu Sales Usd]
   )
SELECT
   [Measures].[AvgMonthlyCPUSales] ON COLUMNS,
   NONEMPTY(
      GENERATE(
         ([TIME].[Month Str].[Month Str], [GEOGRAPHY].[Region].[Region]),
         TOPCOUNT(
            ([TIME].[Month Str].CURRENTMEMBER,
                     [GEOGRAPHY].[Region].CURRENTMEMBER,
                     [CPU].[Cpu Brand].[Cpu Brand]),
            5,
            [Measures].[AvgMonthlyCPUSales]
         )
      )
   )ON ROWS
FROM [Group ID 778 DB]
WHERE [GEOGRAPHY].[Gerarchia].[Continent].&[Europe]
```

Initially, the measure [AvgMonthlyCPUSales] was calculated in the "WITH" block using the AVG function to determine the monthly average CPU sales ([Cpu Sales Usd]) across all months.

Subsequently, the calculated measure [AvgMonthlyCPUSales] was selected on the column axis.

Using the GENERATE function, a set of tuples was created by combining elements from the [TIME].[Month Str] and [GEOGRAPHY].[Region].[Region] hierarchies.

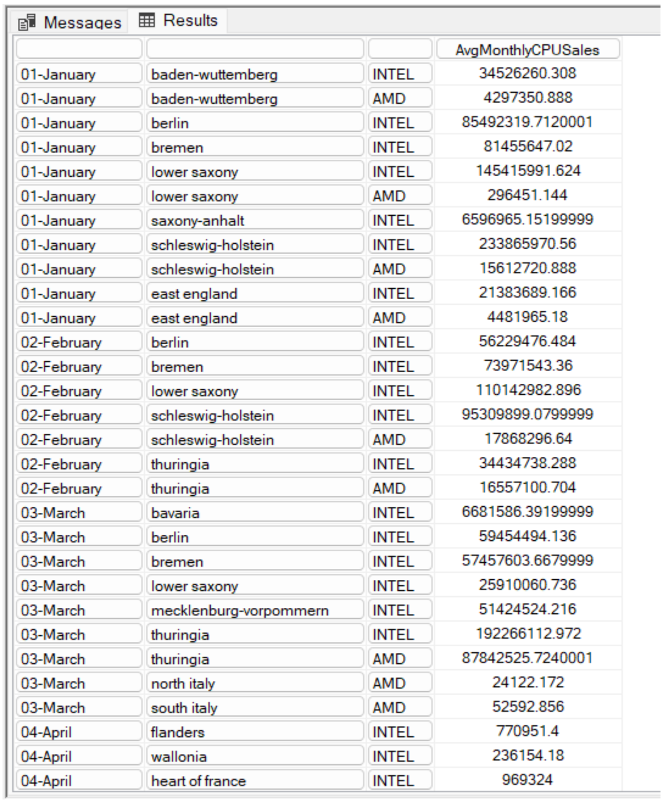
The TOPCOUNT function returned the top 5 members of the [CPU].[Cpu Brand].[Cpu Brand] hierarchy based on the [AvgMonthlyCPUSales] measure.

To ensure only valid tuples, i.e., those with a non-null measure, were filtered, the NONEMPTY function was used.

In summary, the second block of code returned the top 5 CPU brands for each month and for each region on the rows. The 'FROM [Group ID 778 DB]' block indicates the source of the measures and dimensions.

Finally, the 'WHERE [GEOGRAPHY].[Hierarchy].[Continent].&[Europe]' code block limits the selection of the hierarchy to only values belonging to the Europe continent.

Figure 9 shows the results of the first query.



Messages | Results

			AvgMonthlyCPUSales
01-January	baden-wuttemberg	INTEL	34526260.308
01-January	baden-wuttemberg	AMD	4297350.888
01-January	berlin	INTEL	85492319.7120001
01-January	bremen	INTEL	81455647.02
01-January	lower saxony	INTEL	145415991.624
01-January	lower saxony	AMD	296451.144
01-January	saxony-anhalt	INTEL	6596965.15199999
01-January	schleswig-holstein	INTEL	233865970.56
01-January	schleswig-holstein	AMD	15612720.888
01-January	east england	INTEL	21383689.166
01-January	east england	AMD	4481965.18
02-February	berlin	INTEL	56229476.484
02-February	bremen	INTEL	73971543.36
02-February	lower saxony	INTEL	110142982.896
02-February	schleswig-holstein	INTEL	95309899.0799999
02-February	schleswig-holstein	AMD	17868296.64
02-February	thuringia	INTEL	34434738.288
02-February	thuringia	AMD	16557100.704
03-March	bavaria	INTEL	6681586.39199999
03-March	berlin	INTEL	59454494.136
03-March	bremen	INTEL	57457603.6679999
03-March	lower saxony	INTEL	25910060.736
03-March	mecklenburg-vorpommern	INTEL	51424524.216
03-March	thuringia	INTEL	192266112.972
03-March	thuringia	AMD	87842525.7240001
03-March	north italy	AMD	24122.172
03-March	south italy	AMD	52592.856
04-April	flanders	INTEL	770951.4
04-April	wallonia	INTEL	236154.18
04-April	heart of france	INTEL	969324

Figure 9

The table in *Figure 9* presents the result of the previously described query, showing the top 5 cpu_brand with their respective average sales for each combination of month and European regions. Since the cpu_brand variable had only "Intel" and "AMD" categories, the table displays only these two categories.

For example, in January, in the Baden-Württemberg region, the Intel category recorded an average sale of 34,526,260.38 USD. Similarly, in March, in the Northern Italy region, the AMD category recorded an average sale of 24,122.72 USD.

For selecting the top 5 GPU and RAM brands, the same approach used for CPUs was utilized. The measures gpu_sales_usd and ram_sales_usd were used, calculating the monthly average sales, and renamed as AvgMonthlyGPUSales and AvgMonthlyRAMSales respectively. In the code, the hierarchy [CPU].[Cpu Brand].[Cpu Brand] was replaced with [GPU].[Gpu Brand].[Gpu Brand] for the top 5 GPU brands and [RAM].[Ram Brand].[Ram Brand] for the top 5 RAM brands.

4.3 ASSIGNMENT 6

To analyze the data available in the cube, Power BI was used to create a meaningful plot/dashboard. The data was retrieved from the cube using the server connection string. An important analysis focused on the total sales per continent, revealing that 93.58% of the overall sales (across CPU, GPU, and RAM) comes from Europe. Subsequently, the hierarchy was expanded to display the total sales for each country worldwide. It was found that 77.85% of the total sales come from Germany.

Figure 10 shows the two graphs: the first ring chart presents the total sales per continent, while the second pie chart presents the total sales per country.

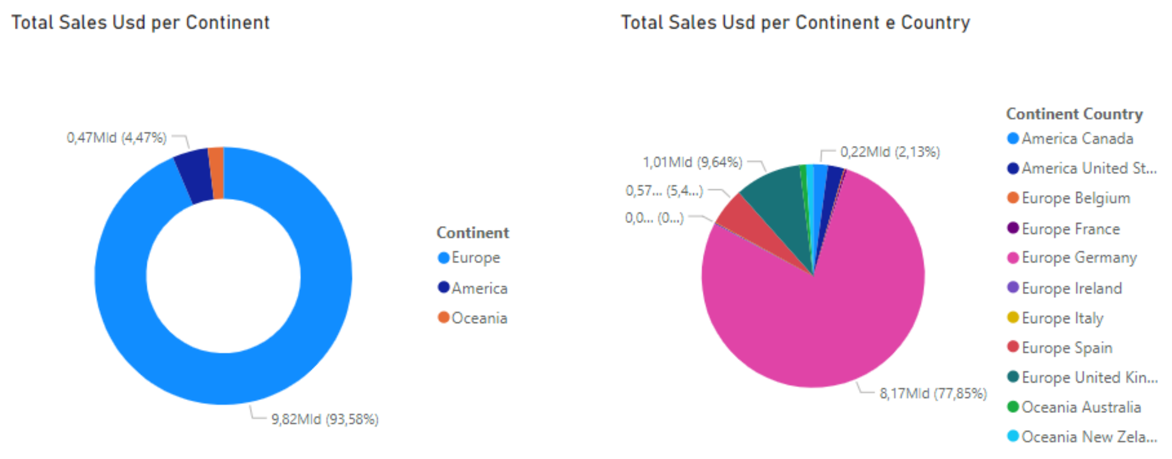


Figure 10

Another important analysis was to evaluate the proportion of total sales attributed to CPUs, GPUs, and RAM. Through a pie chart, depicted in *Figure 11*, the total percentage of sales for each category was highlighted. It emerged that CPU sales prevail, accounting for 64.49% of total sales, followed by GPUs with 29.52% of total sales.

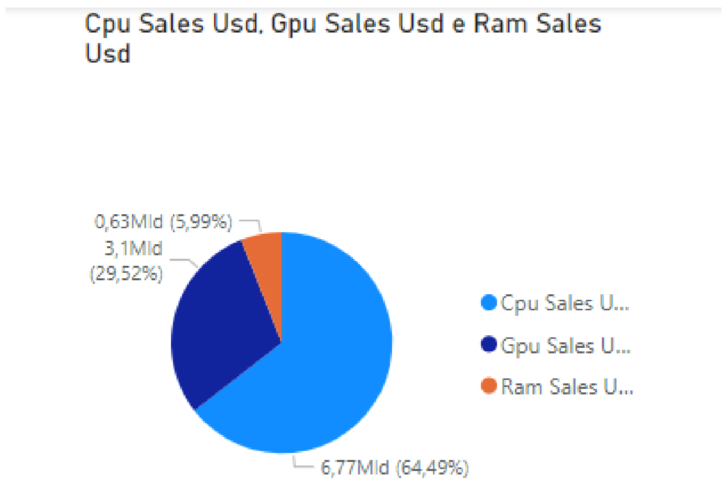


Figure 11

Subsequently, CPU, GPU, and RAM sales for each brand are presented through three histograms in *Figure 12*.

The first histogram shows that approximately 6.41 billion USD is allocated to the sale of Intel CPUs, while only 0.35 billion USD is allocated to the sale of AMD CPUs.

The second histogram reveals that most of the GPU sales come from the PNY GPU, with exactly 1.63 billion USD.

The third histogram illustrates RAM sales by brand, where approximately 158 million USD of RAM sales come from the sale of G.Skill RAM, followed by Kingston with 145 million USD.

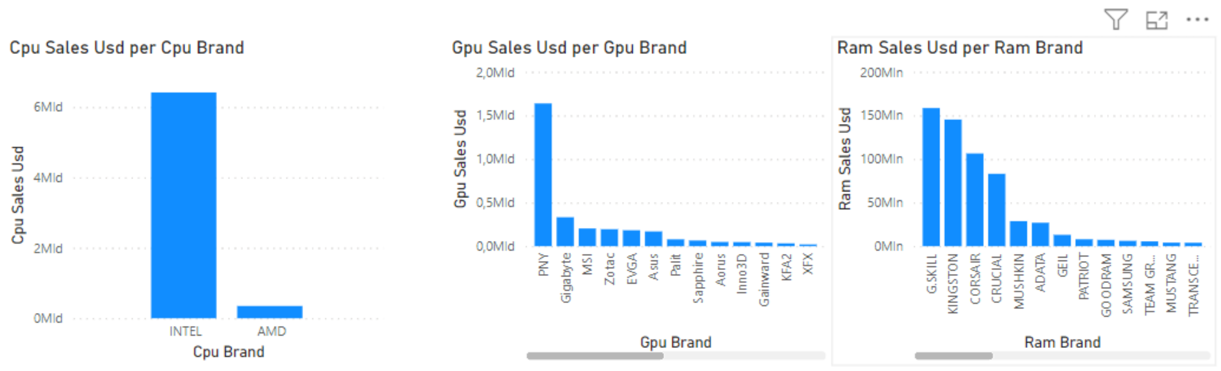


Figure 12

Finally, a last analysis concerns the total sales over time, using a time plot where the x-axis represents the new variable created, month_hier, and the y-axis represents the total sales, as shown in *Figure 13*.

It is noted that from March 2013 to April 2018, there has been a positive trend in total sales, especially with a surge in sales from March 2016 to April 2017, where the average increases from 150 million USD to 320 million USD for this time interval. Subsequently, there is a drastic decline in sales starting from May 2017 onwards, with an average of 170 million USD for the period from May 2017 to April 2018.

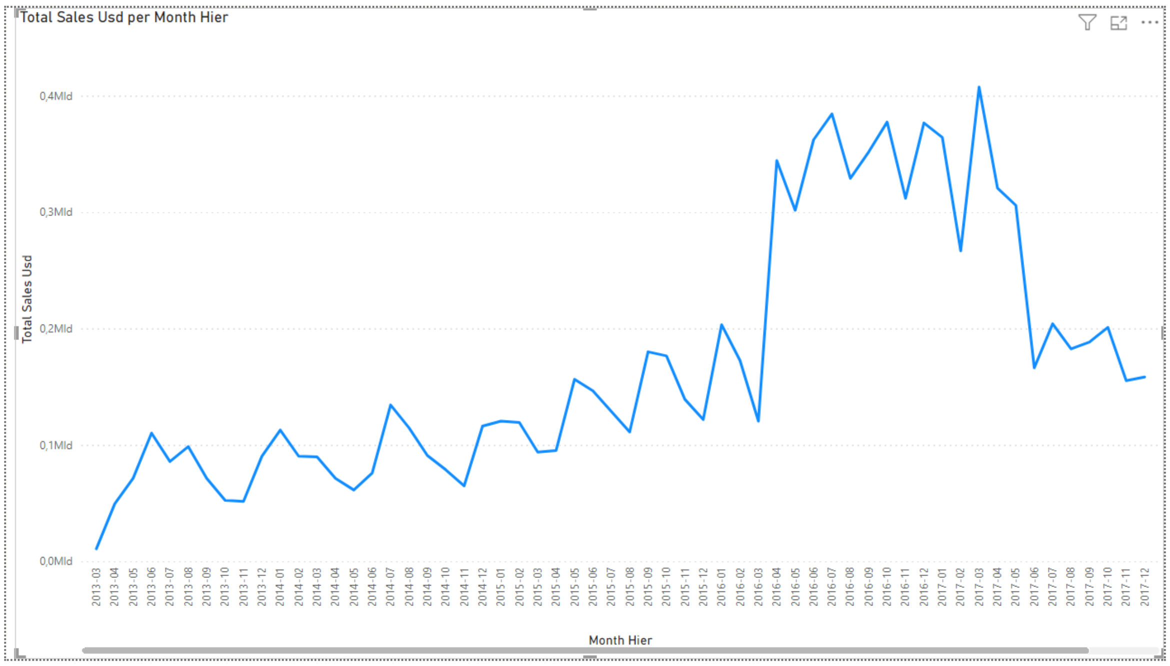


Figure 13