OBJECTIVE:

For this project, we will build a data lakehouse for a fictitious company, **"AlfaMaq Manufatura S.A."**

A manufacturing company is a type of business that transforms raw materials or components into finished products through the use of industrial processes. This type of company is responsible for designing, producing, assembling, and testing products such as automobiles, electronics, food, clothing, machinery, and equipment.

Manufacturing companies typically have factories or facilities where production processes take place. These facilities may involve the use of machines, equipment, tools, robots, and skilled labor to create products that meet quality and safety specifications.

Manufacturing companies can serve both the end-consumer market and the business market, producing a wide variety of products across different sectors. Some manufacturing companies specialize in a single product, while others produce a broad range of products in various areas.

Manufacturing is one of the main economic activities in many countries, employing millions of people around the world. These companies play a fundamental role in the economy, providing essential products and creating jobs in various areas such as production, administration, sales, and marketing.

AlfaMaq needs to generate reports to better understand its business processes. The company has the following data available:

- **Production data**: Information about production, including quantities produced, production time, defect rates, quality data, and information on raw materials and manufacturing processes.
- **Sales data**: Information about product sales, including prices, quantities sold, sales locations, and customer data.
- **Supplier data**: Information about suppliers of raw materials and other inputs necessary for product manufacturing.
- **Financial data**: Information about the company’s finances, including revenue, expenses, profit, and cash flow.

Below are some of the reports the company would like to have in order to better understand business performance, production efficiency, and market needs:

- **Sales reports**: Reports that provide information on sales by region, product, sales channel, and time period, allowing the company to identify best-selling products, market trends, and the most effective sales channels.

- **Inventory reports**: Reports that show information about inventory levels of raw materials, work-in-progress, and finished products, enabling the company to manage inventory more efficiently and reduce storage costs.
- **Production reports**: Reports that display information on production efficiency, including cycle time, defect rate, machine utilization, and other performance indicators, allowing the company to identify improvement areas and increase production efficiency.
- **Maintenance reports**: Reports that provide information on the company's maintenance activities, including downtime, preventive and corrective maintenance, maintenance costs, and other performance indicators, enabling more efficient asset management.
- **Financial reports**: Reports that show financial information such as revenue, expenses, profit margin, cash flow, and other key financial metrics, allowing the company to assess its financial health and make informed decisions.
- **Quality reports**: Reports that provide information on product quality, including inspection and testing data, defect index, and other quality indicators, enabling the company to identify quality improvement areas and take corrective actions.

Our task now is to implement a data warehouse project that meets the company's needs. Let's get to work.

Conceptual Model:

To create a conceptual model, it is important to detail the entities, their attributes, and the relationships between them. We will build the model using the **Entity-Relationship (ER) notation**, which is common in data warehousing projects. Here's how these entities can be modeled:

Entities and Attributes:

Region

- ID
- City
- State
- Country

Product

- ID
- Name
- Category

Sales Channel

- Name
- Region

Time

- Hour
- Day
- Month
- Year

Customer

- ID
- Name
- Type
- City
- State
- Country

Supplier

- ID
- Name
- Product_Type_Supplied

Facility

- Name
- Region

Sale

- Value
- Quantity
- Region
- Product
- Channel
- Time
- Customer
- Supplier
- Facility

Relationships:

- **Sale and Product**: Each sale is associated with a product.
- **Sale and Customer**: Each sale is associated with a customer.
- **Sales Channel and Region**: Each sales channel is associated with a region.
- **Time and Sale**: Each point in time can record multiple sales (one-to-many relationship).
- **Sale and Supplier**: Suppliers participate in the sales process by providing products or necessary inputs.
- **Sale and Facility**: The facility is relevant to the sales process, indicating where the product was produced or stored.

This model helps visualize how the data is interrelated and facilitates understanding of the key points for analysis, such as sales performance by region, by product, or by sales channel.

Next, we will **refine this model** with **business rules** and **considerations about granularity and reporting**.

Business Rules:

The following business rules are considered for this project:

1. Each sale is associated with a **single product**.
2. Each sale is associated with a **single customer**.
3. We will work with **day-level granularity** as the time unit (this means that sales reports will be generated at a minimum per day).
4. **Multiple sales** can occur within the same time unit (**Day**, in our case).
5. **Suppliers and facilities** are **not relevant** to the sales process at this initial stage.
6. Reports must be generated by **Sales Channel**, and each **Sales Channel is associated with a single Region**.
7. We will need reports by **product category**, in addition to reports by individual products.

Refined Conceptual Model

To create a conceptual model, it is important to detail the entities, their attributes, and the relationships between them. We will build the model using **Entity-Relationship (ER) notation**, which is commonly used in data warehousing projects. Here's how these entities can be modeled:

Entities and Attributes:

Product

- ID
- Name
- Category*

Sales Channel

- Name
- Region

Date

- Day
- Month
- Year
- Full_Date

Customer

- ID
- Name
- Type
- City
- State
- Country

Sale

- Value
- Quantity
- Product
- Channel
- Date
- Customer

Relationships:

- **Product and Sale**: Each sale is associated with one product. A product may appear in multiple sales. (*One-to-Many relationship*).
- **Customer and Sale**: Each sale is associated with one customer. A customer may be associated with multiple sales. (*One-to-Many relationship*).
- **Sales Channel and Sale**: Each sale is associated with one sales channel. A sales channel may be associated with multiple sales. (*One-to-Many relationship*).
- **Date and Sale**: Each point in time can record multiple sales. Each sale is associated with a specific date. (*One-to-Many relationship*).

With this structure, we define **four descriptive entities** (Product, Customer, Date, and Sales Channel) and **one fact entity** (Sale).

Your conceptual model should be refined to the point where all relationships are **one-to-many**.

Next, we will build the **dimensional model**.

Dimensional Model

The **Star Schema** and **Snowflake Schema** are two common approaches for organizing Data Warehouses, each with its own characteristics and advantages. Below is a description of each:

In the **Star Schema**, the structure is characterized by a central table called the **fact table**, which stores quantitative transactional data, such as sales or expenses. This fact table is surrounded by several **dimension tables**, which contain descriptive data related to the entries in the fact table. The dimensions are linked to the fact table via foreign keys.

The **Snowflake Schema** is a more normalized variation of the Star Schema. The fact table is also at the center, but the dimension tables are **normalized** into multiple hierarchically related tables. This means that the information in each dimension can be split into additional tables, creating a structure that resembles a **snowflake**.

For Project 1, we can work with either model, as follows:

If we choose the Star Schema, the product table would look like this (for example):

ID	**Name**	**Category**
1001	Refrigerator	Home Appliance
1002	Stove	Home Appliance
1003	Washing Machine	Home Appliance
1004	4K TV	Electronics

If we choose the Snowflake Schema, it would look like this (for example):

Product Table

ID	**Name**	**Category_ID**
1001	Refrigerator	9991
1002	Stove	9991
1003	Washing Machine	9991
1004	4K TV	9992

Category Table

Category_ID	**Category_Name**	**Parent_Category**
9991	Home Appliance	Imported
9992	Electronics	Domestic

To simplify the **ETL process** in **Project 1**, we will work with the **Star Schema model**. Feel free to adjust the project and work with the **Snowflake Schema** if desired.

Logical Model:

The logical model translates a conceptual model into a structure that can be implemented in a database management system (DBMS). It details the tables, the data types for each column, and the relationships between these tables through primary and foreign keys. The logical model is technology-independent; it's like a template that can be used in any DBMS.

Let's transform the conceptual model of Project 1 into a relational database logical model. This model will include the definition of tables, primary keys (PKs) and foreign keys (FKs), data types for each attribute, and some index suggestions to optimize queries.

Table: Dim_Produto

- Produto_ID (PK) - INT
- Nome - VARCHAR(255)
- Categoria - VARCHAR(255)

Table: Dim_Canal

- Canal_ID (PK) - INT
- Nome - VARCHAR(255)
- Região - VARCHAR(255)

Table: Dim_Data

- Data_ID (PK) - INT
- Dia - INT
- Mês - INT
- Ano - INT
- Data_Completa - DATE

Table: Dim_Cliente

- Cliente_ID (PK) - INT
- Nome - VARCHAR(255)
- Tipo - VARCHAR(255)
- Cidade - VARCHAR(255)
- Estado - VARCHAR(50)
- País - VARCHAR(255)

Table: Fato_Venda

- Produto_ID (FK) - INT
- Canal_ID (FK) - INT
- Data_ID (FK) - INT
- Cliente_ID (FK) - INT
- Valor - DECIMAL(10, 2)
- Quantidade - INT

Each sale is associated with a product, a sales channel, a date, and a client through the foreign keys Produto_ID, Canal_ID, Data_ID, and Cliente_ID, respectively.

Data Type Suggestions

- INT for identifiers and quantities.
- VARCHAR(255) for names, categories, cities, states, countries, and types—text fields of variable length but not excessively long.
- DECIMAL(10,2) for monetary values, allowing large values with two decimal places.
- DATE for complete dates, allowing efficient storage and easy date comparisons.

Index Suggestions

- Index on the column Produto_ID in the sales table to speed up queries by product.
- Index on Cliente_ID in the sales table to speed up queries by client.
- Index on Data_Completa in the date table to optimize queries that filter by specific dates.
- Composite index on (Canal_ID, Data_ID) in the sales table may be useful for queries that filter simultaneously by channel and date.

Now let’s convert this logical model into a physical model.

Fisical Model:

```
-- Create schema
CREATE SCHEMA dw AUTHORIZATION useradmin;

-- Dimension: Product
CREATE TABLE dw.dim_product (
   sk_product SERIAL PRIMARY KEY,
   id_product INT,
   name VARCHAR(255),
   category VARCHAR(255),
   insertion_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP);

-- Dimension: Channel
CREATE TABLE dw.dim_channel (
   sk_channel SERIAL PRIMARY KEY,
   id_channel INT,
   name VARCHAR(255),
   region VARCHAR(255),
   insertion_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP);

-- Dimension: Customer
CREATE TABLE dw.dim_customer (
   sk_customer SERIAL PRIMARY KEY,
   id_customer INT,
   name VARCHAR(255),
   type VARCHAR(255),
   city VARCHAR(255),
   state VARCHAR(50),
   country VARCHAR(255),
   insertion_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP);
```

```
-- Dimension: Date
CREATE TABLE dw.dim_date (
    sk_date SERIAL PRIMARY KEY,
    full_date DATE,
    day INT,
    month INT,
    year INT);

-- Fact Table: Sales
CREATE TABLE dw.fact_sales (
    sk_product INT NOT NULL,
    sk_customer INT NOT NULL,
    sk_channel INT NOT NULL,
    sk_date INT NOT NULL,
    quantity INT NOT NULL,
    sales_value DECIMAL(10, 2) NOT NULL,
    insertion_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (sk_product, sk_customer, sk_channel, sk_date),
    FOREIGN KEY (sk_product) REFERENCES dw.dim_product (sk_product),
    FOREIGN KEY (sk_customer) REFERENCES dw.dim_customer (sk_customer),
    FOREIGN KEY (sk_channel) REFERENCES dw.dim_channel (sk_channel),
    FOREIGN KEY (sk_date) REFERENCES dw.dim_date (sk_date));
```