# Sales analytics Project

## Introduction

This data analytics project was developed by M.Eng. Ernesto Cantú, using a Toy Manufacturer Historical Database found on:

<https://www.kaggle.com/datasets/kyanyoga/sample-sales-data?resource=download>

As a Software Engineering Professor at Tec de Monterrey, I face so many times the struggle on how I will teach abstract topics such as Data Base querying without a purpose.

Let’s face it, learning SQL is such a simple thing as learning to do a “Select \* FROM…” but, what do you really need to know is how to give context to the Result Set. What does the business need’s to know about the data you are analyzing.

***This Project’s Purpose***

The project was used to multiple purposes:

1. As a Data Engineering exercise to show my Software Engineering students the difference between good data storing (hardware level) and good data visualizing (business level).
   1. I stored data on a proper database
   2. I transformed some values in order to have a better understanding.
2. As a Data Visualization exercise used on my Decision-Making Course
3. As part of my personal portfolio. Feel free to reach me out if you have any question.

***Contact***

Ernesto Cantú-Valle, Masters on Information Engineering

[ernesto.cantu1989@tec.mx](mailto:ernesto.cantu1989@tec.mx)

Also, reach me out on: [ernesto.cantu@aleph5.com](mailto:ernesto.cantu@aleph5.com)

## Data Processing

I found the data on:

<https://www.kaggle.com/datasets/kyanyoga/sample-sales-data?resource=download>

Kaggle is a free source of datasets used for personal projects, like this one.

***Original data format***

The original dataset comes in a “CSV” file filled with a total of 2,823 records of sales done by the toy manufacturer. The manufacturer share the following data:

|  |  |
| --- | --- |
| Data field | Description |
| Order number | The sale’s unique identifier. As this file gives the whole description of the sale (including the products), this order number is repeated during the file. |
| Product Quantity | Sold Units per product |
| Unit Price | Sales unit price of the product. During the cleaning phase, I found out that this price varies so much from one sale to other. This can be because of many different customer contract conditions. |
| Sales (Product subtotal) | Product quantity \* unit price |
| Order Date | When the order was placed |
| Order Status | I included this to the analysis, but it can be removed. |
| Order Line | Represents the order in which a product was registered on a sale. Also can be removed. |
| Quarter, Month And Year | Metadata of the date |
| Product Line | Description of a group of related products |
| MSRP | Manufacturer’s Suggested Retail Price. |
| Product id | The product’s unique identifier. Give’s no further information. |
| Customer name | Comercial name |
| Customer Phone | Phone number |
| Customer’s Address | Street |
| Customer’s City | City |
| Customer’s State | State (Not all have info.) |
| Customer’s PS Code | Postal Code |
| Customer’s Country | Country |
| Customer’s Territory or Region | Continent |
| Contact | Customer’s contact |
| Deal Size | How big was the sale |

***Data loading***

Loading data into a Database was the first step to analyze it. I chose MySQL, as it is a free RDBMS and also I found a free cloud server that let me host MySQL. So, let’s get started.

*1 – Loading the CSV into MySQL*

The file was a little messed up and I needed a way to load it quickly. That’s where my first tool came up: DBeaver.

DBeaver is a client software to connect to multiple RDBMS and other types of databases. It has useful tools like CSV Loading. Using a local MySQL Server, I started loading the file as I show up next:

On a new database I right clicked on the “Tables” option and selected “Import Data”.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

The only available option to load data is a CSV File, because the database is still empty.

Tabla

Descripción generada automáticamente

You must select then the source file in order to import data:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Keep the process until the file is fully loaded into a table, which will be named after the file.



As you can see, the file data is now into the DB, and we can work on the data engineering:

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

*2 – Transforming data into a proper structure*

*NOTE: Please, feel free to check the steps.sql file to transform the data from the file to a fully Relational Database.*

After analyzing the data and loading it into a table, I started creating some of the catalogs to support the Sales Data.

The catalogs I created were:

1. Order Status
2. Product Line
3. Product\*

In the case of the products, I must say that I tried to reach a product price from the original data but, realized that there where so many prices for the same product across the sales and customers. That’s why, I decided to keep a unique unit price on the Product’s Catalog.

There were a few options:

1. Keep an average price from all the sale prices.
2. Take the smallest price.
3. Take the highest price.

I took the highest price as the unit price. This decision was made to “identify” the difference between the final price and the unit price on the total incomes.

After those 3 catalogs, I proceeded with the Customer’s catalog. In this step, I omitted the Region treatment, for sake of simplicity, but you can also create a Regions catalog and inset them. I decided to use a “Select Case” in order to map those values.

At the end, I divided the Orders info into 2 tables:

1. Orders, which shows order info and customer
2. Order Detail, which shows per order, the quantity of particular products were bought on a specific order.

***Data query***