# 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>

***Purpose***

As a Software Engineering Professor at Tec de Monterrey University, 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 needs to know about the data you are analyzing.

The project was developed for 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 properly structured database
   2. I cleaned values in order to have a better dataset.
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)

## Original raw dataset

I found the dataset 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 shared 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 times unit price.  It is important to notice that this field had almost a 46% of incorrect data according to the expected value. |
| Order Date | When the order was placed |
| Order Status | I included this to the analysis in order to apply a real time analytics business case, 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 | Commercial name of the customer |
| 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 |

NOTE: You can manipulate the CSV file on a tool like Excel.

## Data Preparation

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 – Creating a Relational Database to hold the clean information***

*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 restructure 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.

At the end, I had the following Database design:

Diagrama

Descripción generada automáticamente

***3 – Cleaning data from the original dataset***

The last step, before analyzing data was to populate my new Database with clean data from the original dataset.

For the first three catalogs, I used simple queries to extract information from the original dataset, as shown next:

*Status Catalog*

To extract the status data from the original dataset, I used the following query:

**insert** **into** status (status)

**select** **distinct** status **from** `sales\_data\_sample.csv`

*Product-Detail Catalog*

To extract the product detail from the original dataset, I used the following query:

**insert** **into** product\_line (product\_line)

**select** **distinct** PRODUCTLINE **from** `sales\_data\_sample.csv`

*Product Catalog*

First, to populate the Product’s catalog (as mentioned before) I had to choose the product’s Unit Price. This decision was made to complement the analysis with two prices:

1. The Sales price (final price) per unit
2. The Catalog’s price (a base unit price)

The Catalog’s unit price was obtained by looking for the highest product’s price in the original data.

**INSERT** **INTO** product (product\_id, pl\_id, unit\_price, mrsp)

**select**

o.PRODUCTCODE,

pl.pl\_id,

**MAX**(o.PRICEEACH), #took the highest price

o.MSRP

**from** `sales\_data\_sample.csv` o

**join** product\_line pl **on** pl.product\_line = o.PRODUCTLINE

**group** **by** o.PRODUCTCODE, pl.pl\_id, o.PRODUCTLINE, o.MSRP

*Customers Catalog*

The customer’s catalog was easy to extract from the original data, just by looking for the different customers who have purchased.

**INSERT** **INTO** customer

(name, phone, address1, address2, city, state, pscode, country, territory, contact)

**select**

CUSTOMERNAME,

PHONE,

ADDRESSLINE1,

ADDRESSLINE2,

CITY,

STATE,

POSTALCODE,

COUNTRY,

TERRITORY,

**concat**(CONTACTFIRSTNAME,' ',CONTACTLASTNAME)

**from** `sales\_data\_sample.csv` o

**group** **by** CUSTOMERNAME,PHONE, ADDRESSLINE1, ADDRESSLINE2,CITY,STATE,POSTALCODE,COUNTRY, TERRITORY, **concat**(CONTACTFIRSTNAME,' ',CONTACTLASTNAME)

*Orders*

As mentioned before, the orders was divided into 2 tables:

1. The orders table.
2. The product detail.

The orders table was populated with the following query:

**insert** **IGNORE** **INTO** order`

(`order`, order\_date, **quarter**, month\_id, `year`, customer, status, deal\_size)

**select**

o.ORDERNUMBER,

str\_to\_date(o.ORDERDATE, '%m/%d/%Y'),

o.QTR\_ID,

o.MONTH\_ID,

o.YEAR\_ID,

c.customer\_id,

s.status\_id,

o.DEALSIZE

**from** `sales\_data\_sample.csv` o

**join** customer c **on** c.name = o.CUSTOMERNAME

**join** status s **on** s.status = o.STATUS

**group** **by** o.ORDERNUMBER ,o.ORDERDATE,o.QTR\_ID,o.MONTH\_ID,o.YEAR\_ID,c.customer\_id, s.status\_id, o.DEALSIZE

The order detail (products ordered) was populated with the following query:

**INSERT** **INTO** detail\_order

(`order`, product\_id, quantity, sell\_price, order\_line)

**select**

o.ORDERNUMBER,

o.PRODUCTCODE,

o.QUANTITYORDERED,

o.PRICEEACH,

o.ORDERLINENUMBER

**from** `sales\_data\_sample.csv` o

Just as mentioned before, the final sales value was ignored, in order to obtain the final sale via the PRICEEACH times the QUANTITYORDERED.

***4 – Moving the database to an Online Server***

Just to be able to query the database from Looker Studio, I migrated the database to a hosting service called: “Aiven.io”.

In order to set up your account, please check my tutorial shared on the repo:

url\_file

*Generating the dump file*

*Loading it into Aiven*