****

**Curso SQL: Proyecto Final**

*Datos para todos*

****

**Profesor:** Juan Ignacio Garicoche

**Tutor:** German Rodriguez

**Alumno:** Alexander Billiet

**Comisión:** 31265

**Fecha límite de entrega:** 01/08/2022

INDICE

[Overview: 8](#_Toc133778245)

[Introduction to the subject, Objective, Problem Situation and Business Model 8](#_Toc133778246)

[Entity - Relationship Diagram 10](#_Toc133778247)

[List of tables: 11](#_Toc133778248)

[Table Cliente 11](#_Toc133778249)

[Table Tarjeta 11](#_Toc133778250)

[Table METODO\_DE\_PAGO 12](#_Toc133778251)

[Table Pedido 12](#_Toc133778252)

[Table Descuento 12](#_Toc133778253)

[Bridge Table Descuento – Pedido 13](#_Toc133778254)

[Table Producto 13](#_Toc133778255)

[Bridge table Pedido – Producto 13](#_Toc133778256)

[Table Proveedor 14](#_Toc133778257)

[Data insert 15](#_Toc133778258)

[Considerations: 15](#_Toc133778259)

[Views 16](#_Toc133778260)

[View # 1 – Top 5 Libros más vendidos 16](#_Toc133778261)

[View # 2 – Top 5 Autores más vendidos 17](#_Toc133778262)

[View # 3 – Facturación y Ganancia por Autor 18](#_Toc133778263)

[View # 4 – Descuento Menor Aplicado 18](#_Toc133778264)

[View # 5 – ocurrencias\_descuentos 20](#_Toc133778265)

[Functions 21](#_Toc133778266)

[Function #1 – GANANCIA\_PERIODO 21](#_Toc133778267)

[Function #2 – CONTAR\_TARJETAS 22](#_Toc133778268)

[Stored Procedures 23](#_Toc133778269)

[Stored Procedure #1 – ordenamiento\_producto 23](#_Toc133778270)

[Stored Procedure #2 – nuevo\_proveedor 24](#_Toc133778271)

[Stored Procedure # 3 – nuevo\_pedido 25](#_Toc133778272)

[Triggers 29](#_Toc133778273)

[Aclaración importante: 29](#_Toc133778274)

[Trigger # 1 - log\_insercion\_proveedor: 29](#_Toc133778275)

[Trigger # 2 – chequeo\_vacios\_proveedor: 30](#_Toc133778276)

[Trigger # 3 – trigger\_insercion\_producto: 30](#_Toc133778277)

[Trigger # 4 – chequeo\_producto\_prohibido: 31](#_Toc133778278)

[Implementation of sentences 32](#_Toc133778279)

[Users Creation: 32](#_Toc133778280)

[GRANT permits to users: 32](#_Toc133778281)

[Check how the permits of the DB work: 33](#_Toc133778282)

[Transaction Control Language 37](#_Toc133778283)

[DELETE from TARJETA table using TCL 37](#_Toc133778284)

[INSERT records into DESCUENTO table using TCL 38](#_Toc133778285)

[Backup 42](#_Toc133778286)

[Backup 42](#_Toc133778287)

# Overview:

This is the final Project of a MySQL course I did at Coderhouse academy (coderhouse.com). The general objectives of this project were:

* Create a relational database based on a business model.
* Develop objects that allow the maintenance of the database.
* Implement different queries in SQL that allow the generation of useful information.

The original project was done in Spanish. To upload the project to Github, comments in English have been added to facilitate reading.

Also, this readme file contains several explanations which were part of the final project and that can be useful for understanding the script for people who is initiating their path in MySQL as myself.

Don’t hesitate in reaching me if you have any doubts or comments. Thanks!

# Introduction to the subject, Objective, Problem Situation and Business Model

The theme chosen to develop the database is to simulate an e-commerce company dedicated to the purchase of books from publishers / companies and their subsequent resale, through its website, to end consumers or other reseller companies.

The list of products that this company is dedicated to commercialize (books) was taken from the database of the top 50 bestselling books by Amazon from 2009 to 2019: "Amazon Top 50 Bestselling Books 2009 - 2019".[[1]](#footnote-1)

This database has approximately 500 records where there is information about the book, author, user rating, number of reviews, price, year, genre, etc.



Based on this information, examples of e-commerce business model databases were reviewed to get an idea of what the most important entities should be when designing a database of this type.[[2]](#footnote-2).

Based on these examples, it was concluded that, in order to correctly capture the business flow of an e-commerce firm, it was necessary to formulate entities such as: customers or users, orders, suppliers, payment methods, discounts, etc.

Some other entities, such as inventories or shipping methods, were conscientiously left aside to avoid over-complexizing the learning process of the course with issues that did not directly relate to the content of MySQL.

For the creation of these tables, in some cases, the Mockaroo page was used to create, for example, ID numbers, passwords, email addresses, supplier names, etc. In other cases, MS Excel was used directly, both to create random information and to generate coherent relationships between the records in the different tables (example: a CUSTOMER who places an ORDER and decides to use the CARD PAYMENT METHOD, must have a card registered in his name in the CARD table). Many other records were simply invented in a discretionary manner but always keeping a logical relationship with the rest of the information (example: quantity DISCOUNTS are applied only to those ORDERS that exceed 10 items).

Having defined the most important tables that shape the business model, the flow of information that the database is intended to capture is relatively intuitive and can even be seen in the Entity Relationship Diagram in the following section:

Diagram

Description automatically generated

CUSTOMERS of our company register in our web page, from where pertinent information is captured such as: if they are companies or final consumers, what type of unique identification they have (CUIT or DNI), password to log in and operate in our interface and an e-mail as a means of contact.

These CUSTOMERS place ORDERS that in turn contain PRODUCTS that, logically, are purchased by our firm from a list of SUPPLIERS.

On top of this fundamental layer, other dimensions are added such as, for example: credit and debit cards that our CUSTOMERS have registered in our page, PAYMENT METHODS with which our CUSTOMERS will pay for the ORDERS and various DISCOUNTS that our company offers (sometimes based on internal variables such as, for example, quantity discounts or other times discounts on special dates such as Father's or Mother's Day, Book Day, etc.).

This completed a database that correctly captures a simplified business model.

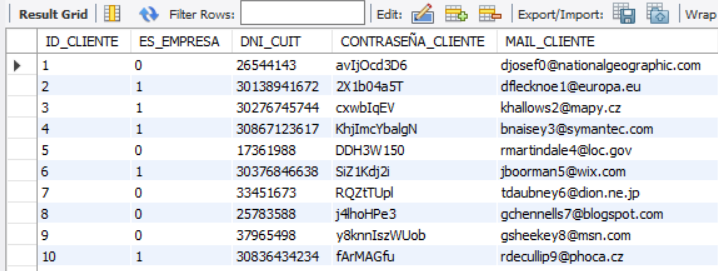
# Entity - Relationship Diagram



# List of tables:

## Table Cliente

These are the customers / users who register on our company's website to buy books. They can be final consumers or reseller companies. It is an exclusive requirement to be registered on our website in order to place an order.



## Table Tarjeta

The different credit and debit cards that our customers have registered when registering on our company's website. Managing this information can be used for sales campaigns, launching promotions, understanding the behavior of our customers, etc.

Graphical user interface, table

Description automatically generated

## Table METODO\_DE\_PAGO

The different payment methods accepted by our company.

Graphical user interface, application, table

Description automatically generated with medium confidence

## Table Pedido

All orders generated by customers through the website. Each order record has, in addition to the identification, a date and time, the customer placing the order and the payment method chosen to complete the transaction.

A screenshot of a computer

Description automatically generated

## Table Descuento

List of the different discounts that the company offers to its customers with their respective names and descriptions.

Graphical user interface, text, application, email

Description automatically generated

## Bridge Table Descuento – Pedido

Since the relationship between DISCOUNT and ORDER is many-to-many, a bridge table is generated where each line has a specific discount applied to a product. An order can have more than one discount applied (example: a quantity discount and in addition a Book Day discount).

Table

Description automatically generated

## Table Producto

It is the list of the products that our company sells. In this case, books. The name of the book, the author, and the purchase price were taken directly from the Kaggle database (mentioned above). To calculate the selling price, a 50% markup was added. In addition, each of these products was assigned to a supplier.

Graphical user interface

Description automatically generated

## Bridge table Pedido – Producto

Since the relationship between ORDER and PRODUCT is many to many, a bridge table is generated where each line is a product (in a certain quantity) contained within an order. To better understand this table, you can look at the screen below where you can see that order 4 has 7 items from book 32 and also 6 items from book 98.

Graphical user interface, table

Description automatically generated

## Table Proveedor

List of suppliers that sell products (books) to our company.

Graphical user interface, text, application

Description automatically generated

# Data insert

Records in this project were inserted into the database using the MySQL Workbench "Table Data Import Wizard".

CSV flat files from which the records were imported are uploaded into this repository.

Given the relationship between primary and foreign keys, the order of import should be as follows:

Text

Description automatically generated with low confidence

# Views

## View # 1 – Top 5 Libros más vendidos

This view offers the 5 best-selling items (books), as well as their respective authors and number of items sold.

It is composed of the tables PEDIDO\_PRODUCTO and PRODUCTO.

The intuition of this view is to be able to have a query to understand which are the most popular books in sales and to make business decisions, advertising campaigns, etc.

This is what the view returns when called:

Graphical user interface, text, application

Description automatically generated

## View # 2 – Top 5 Autores más vendidos

This view offers the top 5 best-selling authors of books. It is interesting because an author can have more than one book for sale. Therefore, this view brings different information from the previous one.

It is composed of the tables PEDIDO\_PRODUCTO and PRODUCTO.

The intuition of this view is to be able to have a query to understand which are the most popular authors in sales and to make business decisions, advertising campaigns, etc.

This is what the view returns when called:

Graphical user interface, text, table

Description automatically generated with medium confidence

## View # 3 – Facturación y Ganancia por Autor

This view shows the revenue for each author, the cost of goods sold and the gross profit and its respective %.

In addition, it is decided to show only those authors whose name contains the letter "J" in order to use the LIKE operator.

It is composed of the tables PEDIDO\_PRODUCTO and PRODUCTO.

The intuition of this view is to be able to have a query to understand which authors are the most popular in sales and most profitable and make business decisions accordingly.

This returns the view when called with the loaded data:

Table

Description automatically generated

## View # 4 – Descuento Menor Aplicado

This view brings a join between two tables to know which orders received the lowest possible discount currently offered by the company.

The grace of this view is to be able to filter the information with a WHERE without knowing the ID or the name of the lowest discount of the company through a subquery.

It is composed of the tables DESCUENTO\_PEDIDO, PEDIDO and makes the subquery to the DESCUENTO table.

This returns the view when called with the loaded data:

Graphical user interface, table

Description automatically generated

## View # 5 – ocurrencias\_descuentos

This view counts the occurrences of discounts (DESCUENTO table) on orders (DESCUENTO\_PEDIDO table).

The intuition of this view is to be able to understand how much the DISCOUNTS offered by our company are being applied and based on this to evaluate their effectiveness.

This returns the view when called with the data loaded:

Graphical user interface, application

Description automatically generated

# Functions

## Function #1 – GANANCIA\_PERIODO

The idea of this function is that it receives two input parameters (start date and end date) and returns the sales profit for that period.

What it does is to calculate the sales and subtract the costs of that sold merchandise to show, precisely, the profit of the period that is passed as parameter.

It is important to mention that the dates passed as parameter must be "valid".

To call the function, the start and end dates are passed as parameters as shown below:

Graphical user interface

Description automatically generated with medium confidence

Which returns:

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

## Function #2 – CONTAR\_TARJETAS

The function count\_cards receive an input parameter "param\_card\_type" that can be "credit" or "debit" and will return the number of cards that are registered in our database.

What the function does is to count the number of rows of the CARD table with a count(\*) filtering with a WHERE by the column "debit\_credit" depending on whether "debit" or "credit" has been entered as parameter.

As shown below, there are 56 credit cards registered and 44 debit cards:

Graphical user interface, text

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

# Stored Procedures

## Stored Procedure #1 – ordenamiento\_producto

Sorts the "product" table by the column you want to use for sorting. It has two input parameters:

- "field\_to\_sort" is the column by which you want to sort the "product" table.

- "order" is a boolean that accepts 0 for ascending order and 1 for descending order.

The stored procedure on the column "autor\_libro" is called:

Text

Description automatically generated with medium confidence

And this is what it returns respectively:

Table "product" sorted in ascending order by column "author\_book".

Graphical user interface

Description automatically generated with medium confidence

Table "product" sorted in descending order by column "author\_book":Graphical user interface, website

Description automatically generated

## Stored Procedure #2 – nuevo\_proveedor

Stored Procedure that inserts a new supplier in the PROVEEDOR table. It accepts 2 input parameters:

- "param\_supplier\_name" which will be the name of the supplier to add.

- param\_supplier\_type" which will be the type of supplier (SRL, SA, SAS, etc).

An IF is added to make sure that the supplier’s name does not already exist in the PROVEEDOR table. In case it already exists, the SP does not add the record and warns by means of a message.

The SP is called as follows:

Text

Description automatically generated with medium confidence

And we get that the first 3 records have been added while the last one is not added and a message is raised warning that there is already a supplier with that name in the table:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

## Stored Procedure # 3 – nuevo\_pedido

Stored Procedure that inserts a new ORDER in the tables "PEDIDO", "PEDIDO\_PRODUCTO" and "DESCUENTO\_PEDIDO"

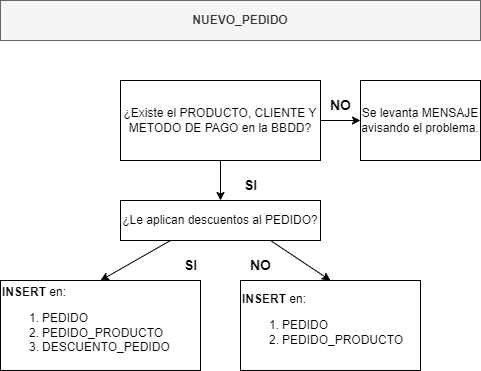
It does this through a TRANSACTION since 3 different tables are being impacted. It has 5 parameters:

1. "id\_libro\_param" product being ordered
2. "cant\_prod\_param" quantity of product being ordered
3. "fecha\_pedido\_param" date of the order
4. "id\_cliente\_param" client ID who is placing the order
5. "metodo\_de\_pago\_param" payment method of the order

In case the product, customer ID or payment method does not exist in the database, the SP will display a message indicating the problem.

In addition, in case the ORDER date is a special date, the corresponding discount will be automatically applied (Ex: Discount for purchase on "Book Day").

In case the ORDER exceeds 10 items, discount # 1 (Quantity discount) will be applied automatically.



In this first part of the SP, we first check with an IF that the parameters entered (ID\_LIBRO\_PARAM, ID\_CLIENTE\_PARAM y METODO\_DE\_PAGO\_PARAM) previously exist in the database.

If they do not exist, a warning message is raised and the insertion of records is suspended. If they do exist, the script continues:

A picture containing diagram

Description automatically generated

Then, it checks if the FECHA\_PEDIDO\_PARAM matches the date of any of the discounts and stores that information in the variable @ID\_DESCUENTO\_FECHA. If no date discount is applied then @ID\_DESCUENTO\_FECHA is set to False:

Text

Description automatically generated with medium confidence

The last check is to understand if the quantity discount applies: if CANT\_PROD\_PARAM is greater than 10 items, then the quantity discount applies and @ID\_DESCUENTO\_CANTIDAD is stored as "1" (quantity discount ID). If, on the other hand, CANT\_PROD\_PARAM is less than 10 items, then the quantity discount is not applied and @ID\_DESCUENTO\_CANTIDAD is stored as False.

A picture containing text

Description automatically generated

Then we proceed to the INSERT of the PEDIDO/ORDER. A transaction is opened and the first INSERT is given in the PEDIDO table and the second in the PEDIDO\_PRODUCTO table:

Text

Description automatically generated

Finally, in case there are discounts, these records are inserted in the table DESCUENTO\_PEDIDO, as follows:

Graphical user interface, text, application

Description automatically generated

We test the SP to verify that it works correctly.

We see that, prior to using the SP, the PEDIDO table only goes up to ID\_PEDIDO = 500:

Graphical user interface, text, application

Description automatically generated

We proceed to call the SP passing it as parameters:

* id\_libro\_param = 1
* cant\_prod\_param = 11
* fecha\_pedido\_param = '2022-10-16'
* id\_cliente\_param = 1
* metodo\_de\_pago = 1



We check the tables to see if the procedure ran correctly.

We see that the record appears in the PEDIDO table with ID\_PEDIDO = 511 with the parameters we passed above:

Graphical user interface, table

Description automatically generated

The same happens in the PEDIDO\_PRODUCTO table where the 11 items of the ID\_LIBRO = 1 are reflected:

Graphical user interface, application

Description automatically generated

Finally, it is noted that, since the quantity discount (ID\_DTO = 1) and the Mother's Day discount (ID\_DTO = 5) apply to this order, both discounts are recorded in the DESCUENTO\_PEDIDO table:

Graphical user interface, application

Description automatically generated

# Triggers

## Trigger # 1 - log\_insercion\_proveedor:

It is a trigger that after a new record is inserted in the "PROVEEDOR" table, it logs information in the " log\_insercion\_proveedor" table. The information stored in this last table is the id\_proveedor”, “nombre\_proveedor”, “tipo\_proveedor”, the "user" who added the record, the date and the time.

This TRIGGER is "AFTER" an "INSERT" event in the "PROVEEDOR" table and basically captures the data already mentioned and saves it as a new record in the " log\_insercion\_proveedor " table.

To test trigger “log\_insercion\_proveedor”, we insert 3 new suppliers into the table “PROVEEDOR”:

Graphical user interface, text, application

Description automatically generated

Then we check the table “log\_insercion\_proveedor” to confirm the trigger has worked ok and added the required registers to the table:



Graphical user interface, application

Description automatically generated

## Trigger # 2 – chequeo\_vacios\_proveedor:

What this trigger does is to validate BEFORE the INSERT of a record to the "PROVEEDOR" table that the record you are trying to insert meets a certain condition.

In this case, if the name of the supplier you are trying to INSERT is empty (new.nombre\_proveedor = ‘’) then it raises a signal that interrupts the INSERT and sends a message to the user explaining why the INSERT was interrupted.

When an attempt is made to insert an empty supplier name as shown below, Workbench returns the following message:

A picture containing table

Description automatically generated



## Trigger # 3 – trigger\_insercion\_producto:

This trigger basically stores data of the INSERTs that have been made to the table PRODUCTO in the table " tabla\_log\_insercion\_producto" with some particularities.

Besides storing information about the "user" that performs the INSERT, "the date", "the time", the "book\_id" and the "book\_name" it also stores the amount of books in the catalog after the INSERT and the new average cost.

Let's see how it is executed:

First, the number of different products in the PRODUCTO table and the average costo\_compra\_libro are looked up, to capture these values BEFORE INSERTING the new record:



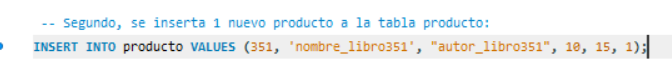
This results in a book count of 350 whose average purchase value is $13.02:

Graphical user interface, website

Description automatically generatedGraphical user interface, text, application, chat or text message, website

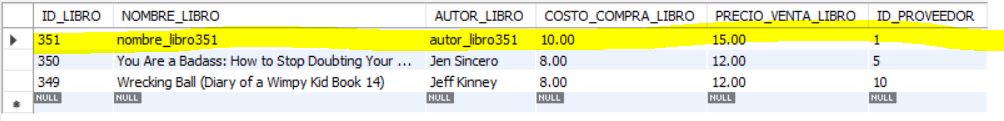
Description automatically generated

Then proceed to insert a new record in the PRODUCTO table:



And it is checked that it has been inserted correctly:





Now we check that the TRIGGER has worked correctly:

Text

Description automatically generated

When calling the " tabla\_log\_insercion\_producto " we get:



This shows that all fields have been stored after the INSERT. It is also observed how the number of books has risen to 351 and the average has changed to $13.01 as these two calculations consider the last INSERT.

Then the record entered into the PRODUCTO table is deleted just to keep the database clean:

Graphical user interface, text, application

Description automatically generated

## Trigger # 4 – chequeo\_producto\_prohibido:

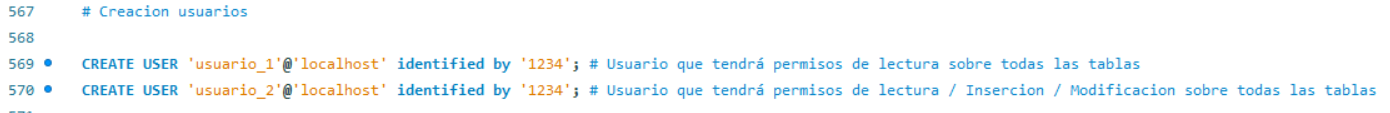
This last TRIGGER is very similar to #2. The TRIGGER is called before an INSERT event in the PRODUCT table and, being of type BEFORE, what it does is to prevent the INSERT in case the author\_book of the record is equal to VOLDEMORT (by company policy the incorporation of books written by VOLDEMORT to the catalog is forbidden).

Similarly, the TRIGGER raises a console message to alert the user why the INSERT cannot be carried out:

# Implementation of sentences

## Users Creation:

The two users (user\_1 and user\_2) are created, both with the domain "localhost" as local domain and with a password equal to 1234:



Check users were created ok:

Graphical user interface, text, application

Description automatically generated

We see they have no permissions granted yet:

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface, text, application

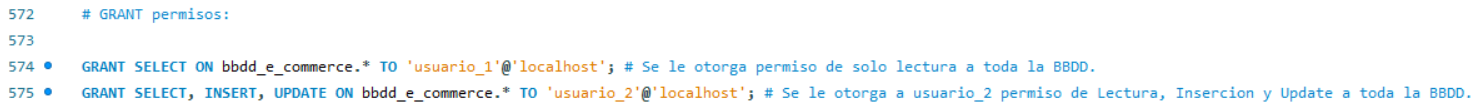
Description automatically generated

A picture containing graphical user interface

Description automatically generated

## GRANT permits to users:

user\_1 is granted read permissions over the entire database while user\_2 is granted read, insert and modify permissions:



Check granted permits:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

## Check how the permits of the DB work:

First, two new connections are created in MySQL Workbench for each of the users:

Graphical user interface, text, application, chat or text message

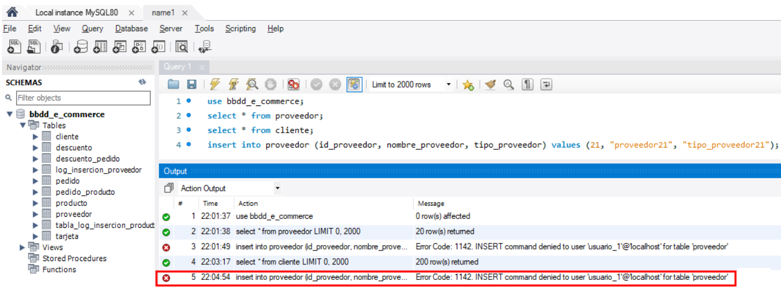
Description automatically generated

In the case of user\_1 we see that it can access in read mode (i.e., call information through a SELECT) in all the tables of the DB:

Graphical user interface

Description automatically generated

But when trying to insert a record we get the following error:



In the case of user\_2 we can see that he can access in read mode to all tables without problems:

Graphical user interface, text, application

Description automatically generated

In addition, it can perform an INSERT without any problems:

Graphical user interface, text, application, email

Description automatically generated

And an UPDATE:

Graphical user interface, text, application, email

Description automatically generated

But when trying to delete the record we get the permission restriction:

Graphical user interface, text, application, email

Description automatically generated

# Transaction Control Language

## DELETE from TARJETA table using TCL

A DELETE of records from the TARJETA table will be performed using TCL. First, the TARJETA table is called to see what it contains and we see that it has 100 cards registered:

Graphical user interface, text, application

Description automatically generated

For the DELETE with TCL the transaction starts with START TRANSACTION. In the next line the last 3 records are deleted (id = 100, 99, 98), which are not effective until the transaction is closed with a COMMIT.

Finally, the changes are made effective with the COMMIT statement:

A picture containing chart

Description automatically generated

The TARJETA table is checked and it is observed that the last 3 records were deleted correctly:

Graphical user interface, text, application

Description automatically generated

## INSERT records into DESCUENTO table using TCL

We proceed to make an INSERT of 8 records to the DESCUENTO table with TCL and implementing SAVEPOINTs.

First, the table is checked before the INSERT and it is observed that there are only 5 records in the table:

Graphical user interface, text, application, email

Description automatically generated

The INSERT with TCL of the 8 new records is assembled. In addition, a SAVEPOINT (primeros\_cuatro\_registros) is added after the insertion of the 4th record and another SAVEPOINT (segundos\_cuatro\_registros) after the insertion of the last 4 records.

A picture containing calendar

Description automatically generated

To demonstrate the operation, still within the TRANSACTION, we call a SELECT and you can see that the 8 records were inserted into the table, but have not yet impacted the database since the TRANSACTION is still open:

Graphical user interface, text, application, email

Description automatically generated

When doing ROLLBACK TO segundos\_cuatro\_registros, we return to the same place since this ROLLBACK was added after the INSERT of the 8 records:

Graphical user interface, text, application

Description automatically generated

But when doing the ROLLBACK TO primeros\_cuatro\_registros and then a SELECT we see that it goes back to the time of the TRANSACTION where only 4 records had been added:

Graphical user interface, text, application

Description automatically generated

Finally, the created SAVEPOINTs are deleted, and the TRANSACTION is closed with a ROLLBACK to avoid impacting the database as seen in the last SELECT:

Graphical user interface, text, application, email

Description automatically generated

# Backup

## Backup

To generate the backup of our database through MySQL Workbench, we go to the "Administration" tab and then to "Data Export":

Graphical user interface, text, application

Description automatically generated

This takes us to the next screen where we choose the database to be exported as well as its tables and views:

Graphical user interface, application

Description automatically generated

Select the database of the final project (bbdd\_e\_commerce), all its tables and in the selector at the bottom right choose "Dump Structure and Data".

In the "Objects to export" select Store Procedures, functions and Triggers:

Graphical user interface, application

Description automatically generated

Finally, and according to the instructions, we export everything to a single file, choose the destination folder where the .sql will be saved and check the "Include Create Schema" option before clicking on "Start Export":

Graphical user interface, text, application, email

Description automatically generated

The export is checked and it is noted that everything went well:

A picture containing shape

Description automatically generated

You Will find this file within the repository.

1. <https://www.kaggle.com/datasets/sootersaalu/amazon-top-50-bestselling-books-2009-2019> [↑](#footnote-ref-1)
2. <https://fabric.inc/blog/ecommerce-database-design-example/>, <http://www.webassist.com/tutorials/Free-eCommerce-MySQL-Database>, <https://github.com/runninguru/MySQL-eCommerce>. [↑](#footnote-ref-2)