

Curso SQL CoderHouse



Comisión: 31265

Estudiante: Jerónimo García

Profesor: Juan Ignacio Garicoche

Tutora: Zulma Balalí

Fecha Inicio: 19/04/2022

Fecha Terminado: 01/08/2022

Nombre del Proyecto: Total Gamer

1. Definición de la temática

Total Gamer es una empresa ficticia (creada para el proyecto de CoderHouse) dedicada a la venta de productos relacionada al gaming. No solo de hardware para PC específicamente sino también de periféricos como sillas, accesorios, monitores, etc.

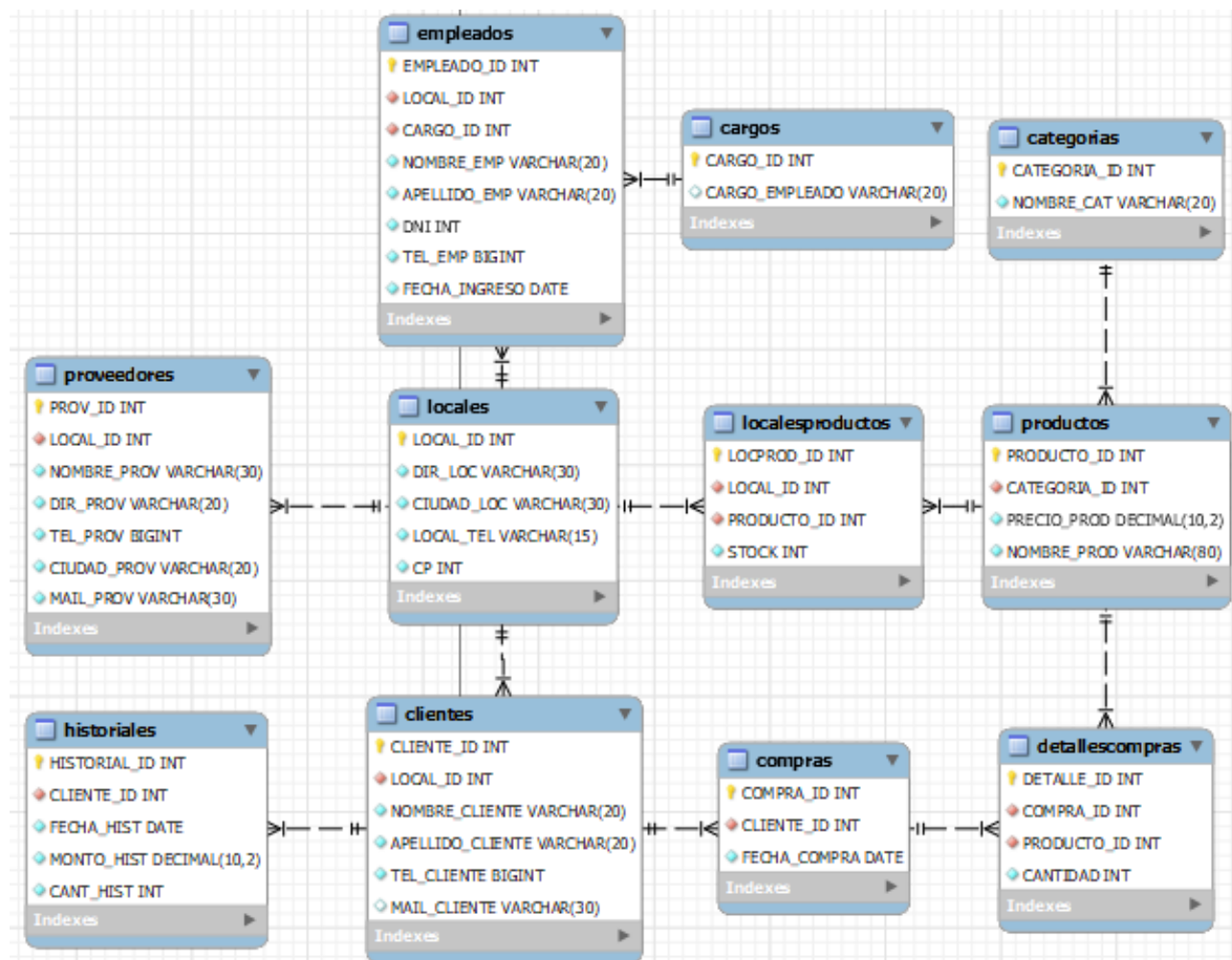
La idea del proyecto es simular la creación de su base de datos permitiendo realizar distintos tipos de consultas, así como también permitiendo ingresar, modificar o eliminar datos de la misma.

Índice

• DER.....	5
• Descripción de tablas.....	6
1. Tabla Locales	
2. Tabla Empleados	
3. Tabla Cargos	
4. Tabla Proveedores	
5. Tabla Clientes	
6. Tabla Historiales de Compras	
7. Tabla Productos	
8. Tabla LocalesProductos	
9. Tabla Compras	
10.Tabla DetallesDeCompras	
11.Tabla Categorías	
• Inserción de datos.....	10
• Vistas.....	12
1. Vista de productos con precio menor al promedio total	
2. Vista producto más caro	
3. Vista empleado más viejo	
4. Vista de cantidad de productos distintos por categoría	
5. Vista de cantidad de compras por cliente	
6. Vista de gasto total por compra	
• Funciones.....	17
1. Función de stock del producto en el local asignado	
2. Función de monto de dinero total por local	
• Stored Procedures.....	19
1. Stored Procedure productos más caros o más baratos	
2. Stored Procedure inserción de empleado	
3. Stored Procedure cantidad de productos por categoría	
4. Stored Procedure cantidad de productos disponibles	
5. Stored Procedure cantidad de empleados por cargo.	

• Triggers	23
1. Trigger inserción producto AFTER	
2. Trigger inserción empleado AFTER	
3. Trigger de espacios vacíos BEFORE	
4. Trigger precio menor a 1000 BEFORE	
• Sublenguaje DCL	27
• Sublenguaje TCL	28
• BackUp	29
• Herramientas y sitios usados	29

2. Diagrama Entidad Relación



Este es el DER autogenerado por MySQL Workbench. En la carpeta de archivos se encuentra el DER realizado mediante Draw.io.

3. Descripción de Tablas

A continuación se presentara una breve descripción de cada una de las tablas y la relación entre las mismas visualizadas en el DER.

Tabla Locales

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN	NOTES
TRUE	LOCAL_ID	INT	TRUE	TRUE		ID de los Locales
	DIR_LOC	VARCHAR	TRUE		30	Dirección
	CIUDAD_LOC	VARCHAR	TRUE		30	Ciudad
	LOCAL_TEL	BIGINT	TRUE	TRUE	15	Numero Telefónico
	CP	INT	TRUE		8	Código Postal

Relación 1-N con tabla Clientes. Si bien el cliente podrá comprar en todos los locales, solo se registrara en un local específico.

Relación 1-N con tabla proveedores. Los productos serán entregados al local central.

Relación 1-N con empleados. Cada empleado estará asignado a un local en específico.

Relación N-N con productos. Ver más adelante en la tabla puente.

Tabla Empleados

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN	NOTES
TRUE	EMPLEADO_ID	INT	TRUE	TRUE		ID de los Empleados
FK	LOCAL_ID	INT	TRUE	TRUE		Local en el que trabaja
	NOMBRE_EMP	VARCHAR	TRUE		20	Nombre empleado
	APELLIDO_EMP	VARCHAR	TRUE		20	Apellido empleado
	DNI	INT	TRUE	TRUE		DNI Empleado
	TEL_EMP	BIGINT	TRUE	TRUE	15	Teléfono Empleado
	FECHA_INGRESO	DATE	TRUE			Fecha de Ingreso
FK	CARGO_ID	INT	TRUE	TRUE		Cargo que realiza

Relación 1-N con tabla cargos. Cada Empleado tendrá un cargo específico dentro de la empresa.

Tabla Cargos

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN	NOTES
TRUE	CARGO_ID	INT	TRUE	TRUE		ID Cargo
	CARGO_EMP	VARCHAR	TRUE	TRUE	20	Nombre de Cargo

Aclara el nombre de cargo que corresponde a cada ID.

Tabla Proveedores

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN	NOTES
TRUE	PROV_ID	INT	TRUE	TRUE		ID de proveedores
FK	LOCAL_ID	INT	TRUE	TRUE		Local al que provee
	NOMBRE_PROV	VARCHAR	TRUE	TRUE	20	Nombre empresa proveedora
	DIR_PROV	VARCHAR	TRUE		20	Dirección proveedor
	TEL_PROV	BIGINT	TRUE	TRUE		Teléfono proveedor
	CIUDAD_PROV	VARCHAR	TRUE		20	Ciudad del proveedor
	MAIL_PROV	VARCHAR	TRUE	TRUE	30	Email del proveedor

Otorga información sobre los distintos proveedores.

Tabla Clientes

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN	NOTES
TRUE	CLIENTE_ID	INT	TRUE	TRUE		ID del cliente
FK	LOCAL_ID	INT	TRUE			Local registrado
	NOMBRE_CLIENTE	VARCHAR	TRUE		20	Nombre del cliente
	APELLIDO_CLIENTE	VARCHAR	TRUE		20	Apellido del cliente
	TEL_CLIENTE	BIGINT	TRUE	TRUE		Teléfono del cliente
	MAIL_CLIENTE	VARCHAR	FALSE	TRUE	30	Mail del cliente

Se relaciona 1-1 con historiales. A cada cliente le corresponderá un solo historial de compras.

Se relación 1-N con compras. Cada cliente podrá realizar múltiples compras.

Tabla Historiales de Compras

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN	NOTES
TRUE	HISTORIAL_ID	INT	TRUE	TRUE		ID del historial
FK	CLIENTE_ID	INT	TRUE			Cliente que compro
	FECHA_HIST	DATE	TRUE			Fecha de la compra
	MONTO_HIST	DECIMAL(10,2)	TRUE			Monto de la compra
	CANT_HIST	INT	TRUE			Cantidad de productos

Brinda información sobre el historial de compras de los clientes registrados.

Tabla Productos

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN	NOTES
TRUE	PRODUCTO_ID	INT	TRUE	TRUE		ID del historial
FK	CATEGORIA_ID	INT	TRUE			Conexión categoría
	NOMBRE_PROD	VARCHAR	TRUE	TRUE	50	Nombre producto
	PRECIO_PROD	DECIMAL(10,2)	TRUE			Precio producto

Información sobre el producto. Se relaciona 1-1 con categorías y 1-1 con detalle de compra. Además de la ya mencionada N-N con locales.

Tabla LocalesProductos

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN	NOTES
TRUE	LOCPROD_ID	INT	TRUE	TRUE		ID del LocalProducto
FK	LOCAL_ID	INT	TRUE			Conexión Local
FK	PRODUCTO_ID	INT	TRUE			Conexión Producto
	STOCK	INT	TRUE			Cantidad de Stock

Tabla puente que relaciona productos con locales. Además presenta el stock del producto que se encuentra en dicho local.

Tabla Compras

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN	NOTES
TRUE	DETALLE_ID	INT	TRUE	TRUE		ID del Detalle
FK	COMPRA_ID	INT	TRUE			ID de la Compra
FK	PRODUCTO_ID	DATE	TRUE			ID del Producto
	CANTIDAD	INT	TRUE			Cantidad de productos

Se relaciona 1-N con DetallesCompras

Tabla DetallesCompras

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN	NOTES
TRUE	DETALLE_ID	INT	TRUE	TRUE		ID del Detalle
FK	COMPRA_ID	INT	TRUE			ID de la Compra
FK	PRODUCTO_ID	DATE	TRUE			ID del Producto
	CANTIDAD	INT	TRUE			Cantidad de productos

Se relaciona 1-1 con productos. Por cada compra de un producto del cliente se genera un detalle de compra. Una compra del cliente puede contener múltiples detalles de compra.

Tabla Categorías

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN	NOTES
TRUE	CATEGORIA_ID	INT	TRUE	TRUE		ID categoría
	NOMBRE_CAT	VARCHAR	TRUE	TRUE	20	Nombre de categoría

Aclara la categoría del producto.

Anexado a los demás archivos se subirá el archivo Excel con la descripción de todas las tablas. No se repitió relación entre tablas. Por ejemplo: si en tabla clientes se aclara que se relaciona 1-N con compras en la tabla compras no se vuelve a aclarar dicha relación.

4. Inserción de Datos

Para la inserción de datos se usó primeramente la consola. Una vez insertados los datos con la asistencia de Navicat se exportaron los datos, se modificaron con Notepad++ y se creó el script de inserción de datos. A continuación se mostraran los comandos utilizados por consola. Los archivos Excel se guardaron en formato para lograr la inserción. Dichos archivos serán adjuntados en el drive.

```
Microsoft Windows [Versión 10.0.19044.1826]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\jero_>cd C:\Program Files\MYSQL\MYSQL Workbench 8.0\

C:\Program Files\MySQL\MySQL Workbench 8.0>mysql --local-infile=1 -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 20
Server version: 8.0.28 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use total_gamer
Database changed
mysql> show tables;
+-----+
| Tables_in_total_gamer |
+-----+
| cargos                 |
| categorias             |
| clientes               |
| compras               |
| detallescompras        |
| empleados              |
| historiales            |
| locales                |
| localesproductos       |
| productos              |
| proveedores            |
+-----+
11 rows in set (0.01 sec)
```

Una vez que se checkeo que las tablas estuvieran creadas en la base de datos, se prosiguió a insertar los datos.

```
mysql> SET GLOBAL local_infile=1;
Query OK, 0 rows affected (0.00 sec)

mysql> load data local infile 'C:/Users/jero/Desktop/Datos/Locales.csv' into table locales fields terminated by ',' ignore 1 lines;
Query OK, 5 rows affected (0.01 sec)
Records: 5 Deleted: 0 Skipped: 0 Warnings: 0

mysql> load data local infile 'C:/Users/jero/Desktop/Datos/Proveedores.csv' into table proveedores fields terminated by ',' ignore 1 lines;
Query OK, 6 rows affected (0.02 sec)
Records: 6 Deleted: 0 Skipped: 0 Warnings: 0

mysql> load data local infile 'C:/Users/jero/Desktop/Datos/Cargos.csv' into table cargos fields terminated by ',' ignore 1 lines;
Query OK, 6 rows affected (0.01 sec)
Records: 6 Deleted: 0 Skipped: 0 Warnings: 0

mysql> load data local infile 'C:/Users/jero/Desktop/Datos/Empleados.csv' into table empleados fields terminated by ',' ignore 1 lines;
Query OK, 20 rows affected (0.01 sec)
Records: 20 Deleted: 0 Skipped: 0 Warnings: 0

mysql> load data local infile 'C:/Users/jero/Desktop/Datos/Clientes.csv' into table clientes fields terminated by ',' ignore 1 lines;
Query OK, 10 rows affected (0.01 sec)
Records: 10 Deleted: 0 Skipped: 0 Warnings: 0

mysql> load data local infile 'C:/Users/jero/Desktop/Datos/HistorialesDeCompras.csv' into table historiales fields terminated by ',' ignore 1 lines;
Query OK, 25 rows affected (0.02 sec)
Records: 25 Deleted: 0 Skipped: 0 Warnings: 0

mysql> load data local infile 'C:/Users/jero/Desktop/Datos/Categorias.csv' into table categorias fields terminated by ',' ignore 1 lines;
Query OK, 10 rows affected (0.01 sec)
Records: 10 Deleted: 0 Skipped: 0 Warnings: 0

mysql> load data local infile 'C:/Users/jero/Desktop/Datos/Productos.csv' into table productos fields terminated by ',' ignore 1 lines;
Query OK, 30 rows affected (0.01 sec)
Records: 30 Deleted: 0 Skipped: 0 Warnings: 0

mysql> load data local infile 'C:/Users/jero/Desktop/Datos/Compras.csv' into table compras fields terminated by ',' ignore 1 lines;
Query OK, 30 rows affected (0.01 sec)
Records: 30 Deleted: 0 Skipped: 0 Warnings: 0

mysql> load data local infile 'C:/Users/jero/Desktop/Datos/DetallesDeCompras.csv' into table detallescompras fields terminated by ',' ignore 1 lines;
Query OK, 35 rows affected (0.01 sec)
Records: 35 Deleted: 0 Skipped: 0 Warnings: 0

mysql> load data local infile 'C:/Users/jero/Desktop/Datos/LocalesProductos.csv' into table localesproductos fields terminated by ',' ignore 1 lines;
Query OK, 79 rows affected (0.01 sec)
Records: 79 Deleted: 0 Skipped: 0 Warnings: 0

mysql> _
```

No se registró ningún skipped ni warnings. Los datos fueron ingresados correctamente.

Cabe aclarar que la inserción tuvo que ser realizada en correcto orden.

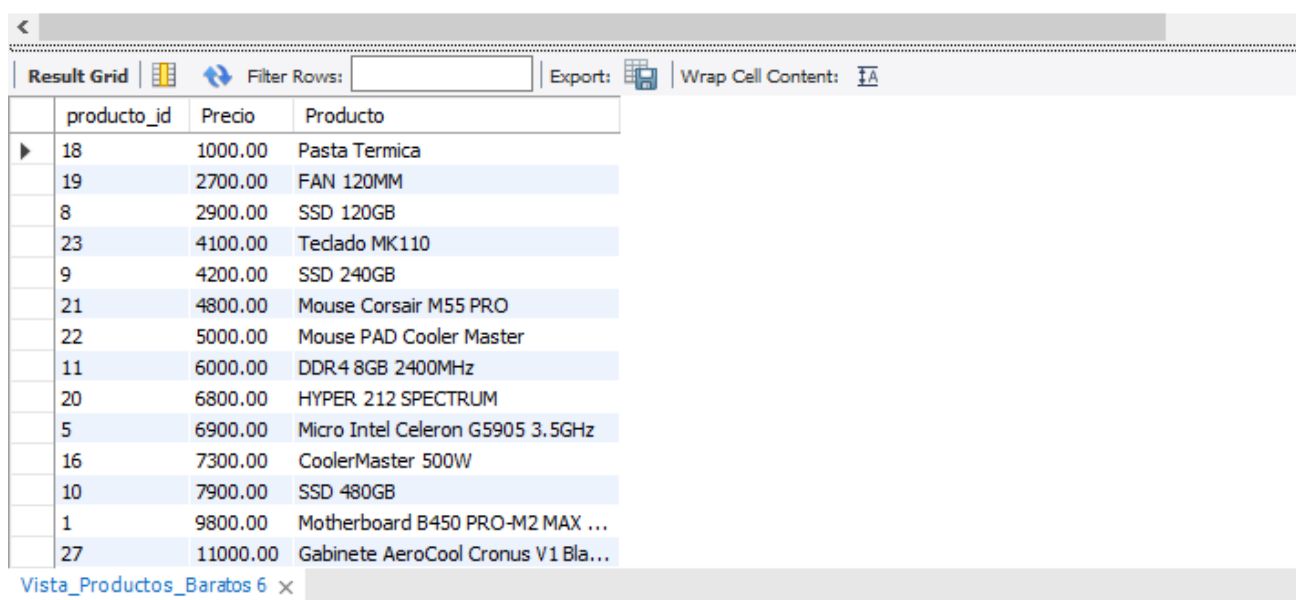
1. Locales
2. Proveedores
3. Cargos
4. Empleados
5. Clientes
6. HistorialesDeCompra
7. Categorias
8. Productos
9. Compras
10. DetallesDeCompras
11. LocalesProductos

5. Vistas

Se crearon un total de 6 vistas.

Vista de productos con precio menor al promedio total

```
207 -- Vista de productos con precio menor al promedio ordenados por precio de forma ascendente
208 • DROP VIEW IF EXISTS Vista_Productos_Baratos;
209 • CREATE VIEW
210 Vista_Productos_Baratos AS
211 SELECT producto_id, precio_prod AS Precio, nombre_prod AS Producto
212 FROM Productos
213 WHERE precio_prod < (SELECT AVG(precio_prod) FROM productos)
214 ORDER BY precio_prod ASC;
```



The screenshot shows a database interface with a toolbar at the top containing icons for 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below the toolbar is a table with the following data:



	producto_id	Precio	Producto
▶	18	1000.00	Pasta Termica
	19	2700.00	FAN 120MM
	8	2900.00	SSD 120GB
	23	4100.00	Teclado MK110
	9	4200.00	SSD 240GB
	21	4800.00	Mouse Corsair M55 PRO
	22	5000.00	Mouse PAD Cooler Master
	11	6000.00	DDR4 8GB 2400MHz
	20	6800.00	HYPER 212 SPECTRUM
	5	6900.00	Micro Intel Celeron G5905 3.5GHz
	16	7300.00	CoolerMaster 500W
	10	7900.00	SSD 480GB
	1	9800.00	Motherboard B450 PRO-M2 MAX ...
	27	11000.00	Gabinete AeroCool Cronus V1 Bla...

At the bottom of the interface, there is a tab labeled 'Vista_Productos_Baratos 6' with a close button (x).

Visualiza los productos por debajo del precio promedio de todos los productos. Parecido a los productos más baratos pero con lógica distinta.

Vista producto más caro



```
219 -- Vista producto mas caro
220 • DROP VIEW IF EXISTS Vista_Producto_Mas_Caro;
221 • CREATE VIEW
222 Vista_Producto_Mas_Caro AS
223 SELECT producto_id, precio_prod AS Precio, nombre_prod AS Producto
224 FROM productos
225 WHERE precio_prod = (SELECT MAX(precio_prod) FROM productos);
226
227 -- SELECT * FROM Vista_Producto_Mas_Caro;
```

Result Grid			
Filter Rows: <input type="text"/>			
Export:  Wrap Cell Content: 			
	producto_id	Precio	Producto
▶	15	195000.00	Nvidia RTX 3080 10GB

Vista sencilla que solo muestra el producto de mayor valor.

Vista empleado más viejo

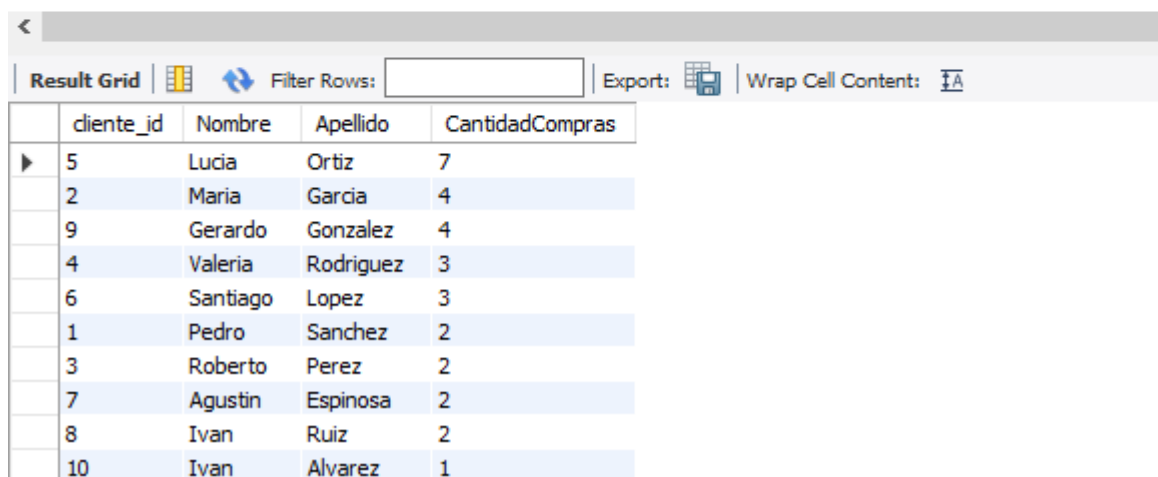
```
230 -- Vista empleado mas viejo
231 • DROP VIEW IF EXISTS Vista_Empleado_Viejo;
232 • CREATE VIEW
233 Vista_Empleado_Viejo AS
234 SELECT empleado_id, nombre_emp AS Nombre, apellido_emp AS Apellido, fecha_ingreso AS FechaIngreso
235 FROM empleados
236 WHERE fecha_ingreso = (SELECT MIN(fecha_ingreso) FROM empleados);
237
238 -- SELECT * FROM Vista_Empleado_Viejo;
```

Result Grid				
Filter Rows: <input type="text"/>				
Export:  Wrap Cell Content: 				
	empleado_id	Nombre	Apellido	FechaIngreso
▶	9	Agustin	Benitez	2020-03-13

Vista del empleado más longevo. Es hora de darle un bono!

Vista de cantidad de compras por cliente

```
241 -- Vista cantidad de compras por cliente ordenados de forma descendente
242 • DROP VIEW IF EXISTS Vista_Compras_Cliente;
243 • CREATE VIEW Vista_Compras_Cliente AS
244 SELECT Compras.cliente_id, Clientes.nombre_cliente AS Nombre,
245 Clientes.apellido_cliente AS Apellido, COUNT(*) AS CantidadCompras
246 FROM compras
247 INNER JOIN Clientes ON Compras.cliente_id = Clientes.cliente_id
248 GROUP BY cliente_id
249 ORDER BY CantidadCompras DESC;
250
251
252 -- SELECT * FROM Vista_Compras_Cliente;
```

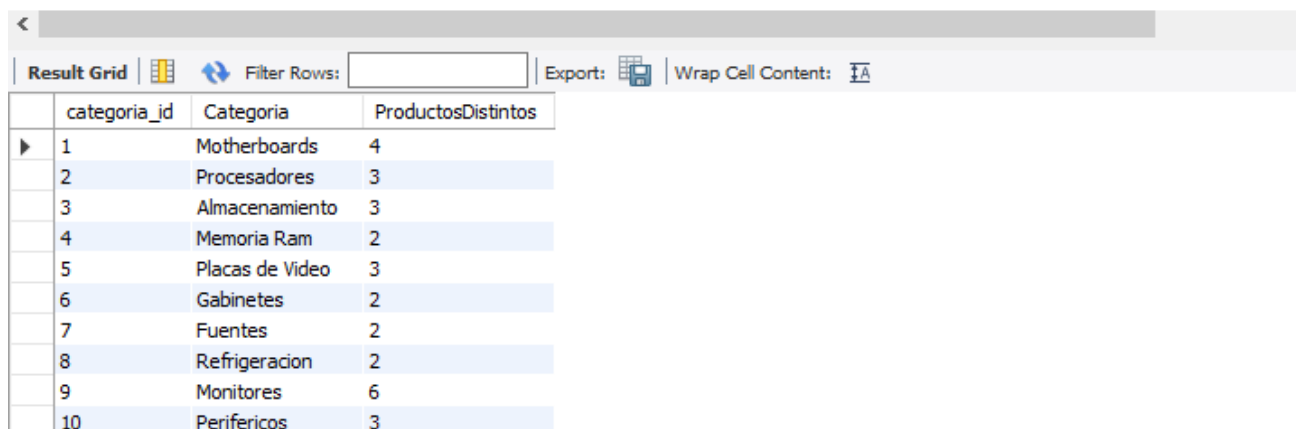


	cliente_id	Nombre	Apellido	CantidadCompras
▶	5	Lucia	Ortiz	7
	2	Maria	Garcia	4
	9	Gerardo	Gonzalez	4
	4	Valeria	Rodriguez	3
	6	Santiago	Lopez	3
	1	Pedro	Sanchez	2
	3	Roberto	Perez	2
	7	Agustin	Espinosa	2
	8	Ivan	Ruiz	2
	10	Ivan	Alvarez	1

Muestra el nombre, apellido y cantidad de compras de cada cliente. Se le agrego también un ordenado descendente por la cantidad.

Vista de cantidad de productos distintos por categoría

```
254 -- Vista de cantidad de productos distintos por categoria ordenados de forma ascendente
255 • DROP VIEW IF EXISTS Vista_Productos_Categoria;
256 • CREATE VIEW Vista_Productos_Categoria AS
257 SELECT Productos.categoria_id, Categorias.nombre_cat AS Categoria,
258 COUNT(*) AS ProductosDistintos
259 FROM productos
260 INNER JOIN Categorias ON Productos.categoria_id = Categorias.categoria_id
261 GROUP BY categoria_id
262 ORDER BY categoria_id ASC;
263
264 -- SELECT * FROM Vista_Productos_Categoria;
```



categoria_id	Categoria	ProductosDistintos
1	Motherboards	4
2	Procesadores	3
3	Almacenamiento	3
4	Memoria Ram	2
5	Placas de Video	3
6	Gabinetes	2
7	Fuentes	2
8	Refrigeracion	2
9	Monitores	6
10	Perifericos	3

Muestra la cantidad de producto diferentes por cada categoría (no tiene en cuenta el stock). Tal vez haya que comprar más gabinetes y fuentes. Muy poca variedad.

Vista de gasto total por compra

```

267 -- Vista de gasto total por compra ordenando por monto de forma descendente
268 -- trayendo id, nombre y apellido del cliente
269 • DROP VIEW IF EXISTS Vista_Gasto_Compra;
270 • CREATE VIEW Vista_Gasto_Compra AS
271 SELECT Clientes.cliente_id, Clientes.nombre_cliente AS Nombre,
272 Clientes.apellido_cliente AS Apellido, Compras.compra_id,
273 SUM(Productos.precio_prod*DetallesCompras.cantidad) AS TotalGastoCompra
274 FROM DetallesCompras
275 INNER JOIN Productos ON DetallesCompras.producto_id = Productos.producto_id
276 INNER JOIN Compras ON DetallesCompras.compra_id = Compras.compra_id
277 INNER JOIN Clientes ON Compras.cliente_id = Clientes.cliente_id
278 GROUP BY compra_id
279 ORDER BY TotalGastoCompra DESC;
280
281 -- SELECT * FROM Vista_Gasto_Compra;

```

Result Grid					
		Filter Rows:			
		Export:			
		Wrap Cell Content:			
	cliente_id	Nombre	Apellido	compra_id	TotalGastoCompra
▶	4	Valeria	Rodriguez	3	422000.00
	5	Lucia	Ortiz	1	406400.00
	1	Pedro	Sanchez	6	361300.00
	3	Roberto	Perez	7	255000.00
	9	Gerardo	Gonzalez	2	147000.00
	5	Lucia	Ortiz	8	140100.00
	2	Maria	Garcia	5	78500.00
	7	Agustin	Espinosa	10	71300.00
	5	Lucia	Ortiz	4	22200.00
	4	Valeria	Rodriguez	9	18000.00

Gasto total por compra. Se puede repetir el cliente. Ordenados de forma descendente.

6. Funciones

Se generaron 2 funciones.

Función de stock del producto en el local asignado

```

287  -- Funcion stock_producto. Me trae la cantidad de stock del producto en el local asignado
288  -- Primer parametro = id del local, Segundo parametro = id del producto.
289  -- Si devuelve null es porque no hay stock de ese producto en el local asignado
290  • DROP FUNCTION IF EXISTS stock_producto;
291  DELIMITER $$
292  • CREATE FUNCTION stock_producto (param_local INT, param_producto INT)
293  RETURNS INT
294  READS SQL DATA
295  BEGIN
296      DECLARE resultado INT;
297      SET resultado = (SELECT LocalesProductos.stock AS Stock
298                      FROM productos
299                      INNER JOIN LocalesProductos
300                      ON LocalesProductos.producto_id = Productos.producto_id
301                      WHERE LocalesProductos.local_id = param_local
302                      AND LocalesProductos.producto_id = param_producto);
303      RETURN resultado;
304  END $$
305  DELIMITER ;
306
307  • -- Llamado a la funcion "stock_producto"
308  -- SELECT stock_producto(4,17) AS Stock;
309

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	Stock			
▶	6			

Ingresando el ID del local y el ID del producto podremos saber la cantidad en stock de ese producto en ese local. Si entrega null es porque no hay.

Función de monto de dinero total por local

```

311 -- Funcion acumulado_local me devuelve el monto de dinero total
312 -- de todos los productos que se encuentran en ese local
313 -- Suma de todos los productos del precio del producto
314 -- multiplicado por el stock del local que se asigna por parametro
315 DROP FUNCTION IF EXISTS acumulado_local;
316 DELIMITER $$
317 • CREATE FUNCTION acumulado_local (param_local INT)
318 RETURNS DECIMAL(12,2)
319 READS SQL DATA
320 BEGIN
321     DECLARE sumaTotal INT;
322     SET sumaTotal = (SELECT SUM(Productos.precio_prod * LocalesProductos.stock)
323                     FROM LocalesProductos
324                     INNER JOIN Locales ON LocalesProductos.local_id = Locales.local_id
325                     INNER JOIN Productos ON Productos.producto_id = LocalesProductos.producto_id
326                     WHERE LocalesProductos.local_id = param_local);
327     RETURN sumaTotal;
328 END $$
329 DELIMITER ;
330
331 • -- Llamado a la funcion "acumulado_local"
332 -- SELECT acumulado_local(5) AS TotalAcumulado;
333

```

Result Grid		Filter Rows: <input type="text"/>	Export:	Wrap Cell Content:
	TotalAcumulado			
▶	2812100.00			

Me devuelve la suma de todos los productos disponibles en ese local por la cantidad en stock. Puede ser un dato interesante para una mejor distribución de productos si comparásemos cuanto hay en cada local.

7. Stored Procedures

Stored Procedure productos más caros o más baratos

```

337 -- Traer los n productos mas caros o los n productos mas baratos.
338 -- Segundo parametro = 1 para los mas caros(descendente), 0 para los baratos(ascendente).
339 -- Cualquier otro numero no trae nada
340 DROP PROCEDURE IF EXISTS productos_caros_baratos;
341 DELIMITER $$
342 • CREATE PROCEDURE productos_caros_baratos (IN cantidad_productos INT, IN orden INT)
343 BEGIN
344     IF orden = 1 THEN
345         SELECT producto_id, nombre_prod AS Producto, precio_prod AS Precio
346         FROM productos
347         ORDER BY precio_prod DESC
348         LIMIT cantidad_productos;
349     ELSEIF orden = 0 THEN
350         SELECT producto_id, nombre_prod AS Producto, precio_prod AS Precio
351         FROM productos
352         ORDER BY precio_prod ASC
353         LIMIT cantidad_productos;
354     END IF;
355 END$$
356 DELIMITER ;
357
358 • -- CALL productos_caros_baratos(15,0);

```

Result Grid Filter Rows: Export: Wrap Cell Content: IA			
	producto_id	Producto	Precio
▶	18	Pasta Termica	1000.00
	19	FAN 120MM	2700.00
	8	SSD 120GB	2900.00
	23	Teclado MK110	4100.00
	9	SSD 240GB	4200.00
	21	Mouse Corsair M55 PRO	4800.00
	22	Mouse PAD Cooler Master	5000.00
	11	DDR4 8GB 2400MHz	6000.00
	20	HYPER 212 SPECTRUM	6800.00
	5	Micro Intel Celeron G5905 3.5GHz	6900.00

Me devuelve una tabla con los productos más caros o más baratos asignado a través de uno de los parámetros y la cantidad que deseamos ver, también ingresado por parámetro. El primer parámetro corresponde a la cantidad de productos, el segundo parámetro para asignarlo de forma ascendente o descendente.

Stored Procedure inserción de empleado

```
362 -- Insertar empleado
363 DROP PROCEDURE IF EXISTS agregar_empleado;
364 DELIMITER $$
365 CREATE PROCEDURE agregar_empleado (IN id INT, IN localId INT, IN cargoId INT,
366 IN nombre varchar(20), IN apellido varchar(20), IN dni INT, IN tel varchar(20), IN fecha Date)
367 BEGIN
368     INSERT INTO Empleados (Empleado_ID, LOCAL_ID, CARGO_ID, NOMBRE_EMP, APELLIDO_EMP, DNI, TEL_EMP, FECHA_INGRESO)
369     VALUES (id, localId, cargoId, nombre, apellido, dni, tel, fecha);
370 END$$
371 DELIMITER ;
372 -- CALL agregar_empleado('50', '5', '2', 'Jorge', 'Prueba', '20500200', '5', '2020-05-15');
373 -- SELECT * FROM Empleados;
```

Result Grid

EMPLEADO_ID	LOCAL_ID	CARGO_ID	NOMBRE_EMP	APELLIDO_EMP	DNI	TEL_EMP	FECHA_INGRESO
50	5	2	Jorge	Prueba	20500200	5	2020-05-15
20	5	2	Rodolfo	Andrade	39552184	3411789400	2022-02-04

Insertar empleado en la tabla de empleados.

Stored Procedure cantidad de productos por categoría

```
376 -- Cantidad de productos por categoria ingresando nombre de la categoria
377 DROP PROCEDURE IF EXISTS productos_nombreCategoria;
378 DELIMITER $$
379 CREATE PROCEDURE productos_nombreCategoria (IN categoria VARCHAR(25))
380 BEGIN
381     SELECT COUNT(Productos.categoria_id) AS Cantidad, Categorias.nombre_cat AS Categoria
382     FROM productos
383     INNER JOIN categorias ON Productos.categoria_id = Categorias.categoria_id
384     WHERE nombre_cat LIKE CONCAT('%', categoria, '%')
385     GROUP BY Productos.categoria_id, Categorias.nombre_cat;
386 END $$
387 DELIMITER ;
388
389 -- CALL productos_nombreCategoria('Refrigeracion');
```

Result Grid

Cantidad	Categoria
2	Refrigeracion

Similar a la vista, nada más que acá ingresamos la categoría en sí que queremos saber el dato.

Stored Procedure cantidad de productos disponibles

```

392  -- Muestra todos los productos disponibles por local, stock, precio y nombre de producto,
393  -- ingresando el id del local. Ordenados de forma ascendente
394
395  DROP PROCEDURE IF EXISTS productos_local;
396  DELIMITER $$
397  • CREATE PROCEDURE productos_local (param_local INT)
398  • BEGIN
399      SELECT LocalesProductos.local_id, LocalesProductos.producto_id,
400      LocalesProductos.stock, Productos.precio_prod AS Precio, Productos.nombre_prod AS Producto
401      FROM LocalesProductos
402      INNER JOIN Productos ON LocalesProductos.producto_id = Productos.producto_id
403      WHERE local_id = param_local
404      ORDER BY Productos.precio_prod;
405  END $$
406  DELIMITER ;
407
408  • -- CALL productos_local(1);
409

```

Result Grid Filter Rows: Export: Wrap Cell Content:					
	local_id	producto_id	stock	Precio	Producto
▶	1	18	6	1000.00	Pasta Termica
	1	19	12	2700.00	FAN 120MM
	1	8	5	2900.00	SSD 120GB
	1	22	4	5000.00	Mouse PAD Cooler Master
	1	11	1	6000.00	DDR4 8GB 2400MHz
	1	20	8	6800.00	HYPER 212 SPECTRUM
	1	5	9	6900.00	Micro Intel Celeron G5905 3.5GHz
	1	16	16	7300.00	CoolerMaster 500W
	1	10	5	7900.00	SSD 480GB
	1	1	5	9800.00	Motherboard B450 PRO-M2 MAX AM4

Ingresando el ID del local, obtenemos el ID del producto, la cantidad de stock, precio y el nombre del producto.

Stored Procedure cantidad de empleados por cargo.

```
411 -- Muestra la cantidad total de empleados dependiendo el cargo ingresado por parametro
412
413 DROP PROCEDURE IF EXISTS empleados_cargo;
414 DELIMITER $$
415 • CREATE PROCEDURE empleados_cargo (IN CargoIn VARCHAR(10))
416 BEGIN
417     SELECT COUNT(Cargos.cargo_id) AS Cantidad, Cargos.cargo_employado AS Cargo
418     FROM Empleados
419     INNER JOIN Cargos ON Empleados.cargo_id = Cargos.cargo_id
420     WHERE Cargos.cargo_employado LIKE CONCAT('%', CargoIn, '%');
421 END $$
422 DELIMITER ;
423
424 • -- CALL empleados_cargo('Capataz');
```

<		
Result Grid Filter Rows: Export: Wrap Cell Content:		
	Cantidad	Cargo
▶	6	Capataz

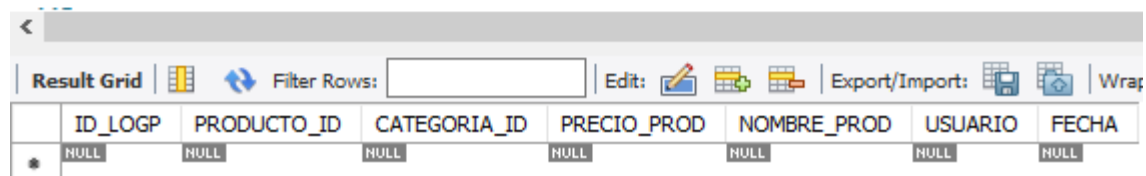
Ingresando el nombre del cargo nos devuelve la cantidad de empleados ocupando dicho cargo.

8. Triggers

Se generaron un total de 4 triggers. Para la inserción de nuevos productos y nuevos empleados se crearon 2 nuevas tablas "log"

Tabla log_productos

```
433 DROP TABLE IF EXISTS log_productos;
434 CREATE TABLE log_productos (
435     ID_LOGP INT PRIMARY KEY AUTO_INCREMENT,
436     PRODUCTO_ID INT NOT NULL,
437     CATEGORIA_ID INT NOT NULL,
438     PRECIO_PROD DECIMAL(10 , 2 ) NOT NULL,
439     NOMBRE_PROD VARCHAR(80) NOT NULL UNIQUE,
440     USUARIO VARCHAR(50),
441     FECHA DATETIME
442 );
443
444 -- SELECT * FROM log_productos;
```



The screenshot shows a database management tool interface. The top part displays SQL code for creating a table named 'log_productos'. The code includes a 'DROP TABLE IF EXISTS' statement followed by a 'CREATE TABLE' statement with columns: 'ID_LOGP' (INT, PRIMARY KEY, AUTO_INCREMENT), 'PRODUCTO_ID' (INT, NOT NULL), 'CATEGORIA_ID' (INT, NOT NULL), 'PRECIO_PROD' (DECIMAL(10, 2), NOT NULL), 'NOMBRE_PROD' (VARCHAR(80), NOT NULL, UNIQUE), 'USUARIO' (VARCHAR(50)), and 'FECHA' (DATETIME). Below the code, there is a comment '-- SELECT * FROM log_productos;'. The bottom part of the screenshot shows a 'Result Grid' with a table structure preview. The table has columns: 'ID_LOGP', 'PRODUCTO_ID', 'CATEGORIA_ID', 'PRECIO_PROD', 'NOMBRE_PROD', 'USUARIO', and 'FECHA'. The first row shows all columns as 'NULL'.

ID_LOGP	PRODUCTO_ID	CATEGORIA_ID	PRECIO_PROD	NOMBRE_PROD	USUARIO	FECHA
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Aquí se insertaran los datos cuando se inserte un nuevo producto. Tomará el usuario que lo ha hecho y la fecha en el momento de la inserción.

Trigger inserción producto AFTER

```
446 • DROP TRIGGER IF EXISTS log_insercion_producto;
447 • CREATE
448     TRIGGER log_insercion_producto
449     AFTER INSERT ON productos FOR EACH ROW
450     INSERT INTO log_productos VALUES (DEFAULT, new.producto_id,
451     new.categoria_id, new.precio_prod, new.nombre_prod, user(), now());
452
453     -- SELECT * FROM log_productos;
454     -- INSERT INTO productos VALUES (31, 2, 50000.00, "PRUEBAPRODUCTO");
455     -- INSERT INTO productos VALUES (32, 2, 50000.00, "PRUEBAPRODUCTO2");
456     -- INSERT INTO productos VALUES (33, 2, 50000.00, "PRUEBAPRODUCTO3");
```

Result Grid							
Filter Rows: <input type="text"/>							
Edit: Export/Import: Wrap Cell Content:							
	ID_LOGP	PRODUCTO_ID	CATEGORIA_ID	PRECIO_PROD	NOMBRE_PROD	USUARIO	FECHA
▶	1	31	2	50000.00	PRUEBAPRODUCTO	root@localhost	2022-08-01 10:37:41
	2	32	2	50000.00	PRUEBAPRODUCTO2	root@localhost	2022-08-01 10:37:45
	3	33	2	50000.00	PRUEBAPRODUCTO3	root@localhost	2022-08-01 10:37:50
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Si la inserción en la tabla productos fue satisfactoria se disparara el trigger y e inyectara los datos en la tabla log_productos.

Tabla log_empleados

```

463 • DROP TABLE IF EXISTS log_empleados;
464 • CREATE TABLE log_empleados (
465     ID_LOGE INT PRIMARY KEY AUTO_INCREMENT,
466     EMPLEADO_ID INT NOT NULL,
467     LOCAL_ID INT NOT NULL,
468     CARGO_ID INT NOT NULL,
469     NOMBRE_EMP VARCHAR(20) NOT NULL,
470     APELLIDO_EMP VARCHAR(20) NOT NULL,
471     DNI INT NOT NULL UNIQUE,
472     TEL_EMP VARCHAR(20) NOT NULL UNIQUE,
473     USUARIO VARCHAR(50),
474     FECHA_INGRESO DATETIME
475 );
476

```

Aquí se insertaran los datos cuando se inserte un nuevo empleado. Tomará el usuario que lo ha hecho y la fecha en el momento de la inserción como fecha de ingreso.

Trigger inserción empleado AFTER

```

477 • DROP TRIGGER IF EXISTS log_insercion_empleado;
478 • CREATE
479     TRIGGER log_insercion_empleado
480     AFTER INSERT ON empleados FOR EACH ROW
481     INSERT INTO log_empleados VALUES (DEFAULT , new.EMPLEADO_ID , new.LOCAL_ID ,
482     new.CARGO_ID , new.NOMBRE_EMP , new.APELLIDO_EMP , new.DNI , new.TEL_EMP , USER() , NOW());
483
484     -- SELECT * FROM log_empleados;
485     -- INSERT INTO empleados VALUES (21, 3, 1, "NOMBREEMP", "APELLIDOEMP", 27000000, 3410000001, now());
486     -- INSERT INTO empleados VALUES (22, 3, 1, "NOMBREEMP", "APELLIDOEMP", 27000001, 3410000002, now());
487     -- INSERT INTO empleados VALUES (23, 3, 1, "NOMBREEMP", "APELLIDOEMP", 27000002, 3410000003, now());

```

Result Grid										
	ID_LOGE	EMPLEADO_ID	LOCAL_ID	CARGO_ID	NOMBRE_EMP	APELLIDO_EMP	DNI	TEL_EMP	USUARIO	FECHA_INGRESO
▶	23	23	3	1	NOMBREEMP	APELLIDOEMP	27000002	3410000003	root@localhost	2022-08-01 10:43:26
	22	22	3	1	NOMBREEMP	APELLIDOEMP	27000001	3410000002	root@localhost	2022-08-01 10:43:23
	21	21	3	1	NOMBREEMP	APELLIDOEMP	27000000	3410000001	root@localhost	2022-08-01 10:43:19

Trigger de espacios vacíos BEFORE

```
494 • DROP TRIGGER IF EXISTS chequeo_empleado_vacio;
495
496 DELIMITER $$
497 • CREATE TRIGGER chequeo_empleado_vacio
498 BEFORE INSERT ON empleados
499 FOR EACH ROW
500 BEGIN
501 IF new.nombre_emp = ' ' THEN
502     signal sqlstate '45000';
503 END IF;
504 END $$
505 DELIMITER $$
506
507 -- INSERT INTO empleados VALUES (22, 3, 1, " ", "APELLIDOEMP", 27000000, 3410000000, now());
508
```

Checkea que no haya espacios antes de la inserción de datos en la tabla empleados.

Trigger precio menor a 1000 BEFORE

```
511 DROP TRIGGER IF EXISTS chequeo_precio;
512
513 DELIMITER $$
514 • CREATE TRIGGER chequeo_precio
515 BEFORE INSERT ON productos
516 FOR EACH ROW
517 BEGIN
518 IF NEW.precio_prod < 1000 THEN
519     SIGNAL SQLSTATE '45000';
520 END IF;
521 END $$
522 DELIMITER $$
523
524 • -- INSERT INTO productos VALUES (33, 4, 5, 900, 5, "PRUEBAPRODUCTO");
```

Si se ingresa un producto con un precio menor a 1000 saltara un error delimitado por usuario en consola.

✖ 1850 10:49:13 INSERT INTO productos VALUES (33, 4, 900, "PRUEBAPRODUCTO") Error Code: 1644. Unhandled user-defined exception condition

9. Sublenguaje DCL. Creación de usuarios y privilegios.

Se crearon dos usuarios. A uno solo se le entrego el privilegio de “solo lectura” y al otro se le otorgo la lectura sobre todas las tablas, inserción de datos en tablas y modificación de datos de las tablas.

```
531      -- Creacion de Usuario
532      DROP USER IF EXISTS 'user_only_reading'@'localhost';
533      DROP USER IF EXISTS 'user_more_actions'@'localhost';
534      CREATE USER 'user_only_reading'@'localhost' identified by '1234';
535      CREATE USER 'user_more_actions'@'localhost' identified by '12345';
536
537      -- Checkeo de Usuarios
538      -- SELECT * FROM mysql.user;
539
540      -- ----- Permisos para 'user_only_reading' -----
541
542      -- Select sobre todas las tablas
543      GRANT SELECT ON empresa_pc.* TO 'user_only_reading'@'localhost';
544
545      -- ----- Permisos para 'user_more_actions' -----
546
547      -- Select sobre todas las tablas
548      GRANT SELECT ON empresa_pc.* TO 'user_more_actions'@'localhost';
549      -- Insercion sobre todas las tablas
550      GRANT INSERT ON empresa_pc.* TO 'user_more_actions'@'localhost';
551      -- Modificacion sobre todas las tablas
552      GRANT UPDATE ON empresa_pc.* TO 'user_more_actions'@'localhost';
553
554      -- Mostrar permisos
555      -- SHOW GRANTS FOR 'user_only_reading'@'localhost';
556      -- SHOW GRANTS FOR 'user_more_actions'@'localhost';
557
558      -- Refresh de privilegios
559      FLUSH PRIVILEGES;
```

10. Sublenguaje TCL

Se realizaron dos transacciones sencillas. En la primera, con una tabla existente, se insertó una nueva categoría y luego se la borro con un rollback. En la segunda transacción, se creó una nueva tabla “ClientesPremium”, se insertaron nuevos registros con savepoints y se habilitaron los rollbacks. En caso de volver por ejemplo al primer savepoint, se eliminaran los últimos 4 registros que fueron ingresados luego de este savepoint.

```
563     START TRANSACTION;
564     INSERT INTO CATEGORIAS VALUES (15, 'Sillas Gamers');
565     ROLLBACK;
566     -- COMMIT;
567     DELETE FROM CATEGORIAS WHERE CATEGORIA_ID=15;
568
569     -- SELECT * FROM ClientesPremium;
570
571     DROP TABLE IF EXISTS ClientesPremium;
572     CREATE TABLE ClientesPremium (
573     CLIENTEP_ID INT NOT NULL auto_increment primary key,
574     NOMBRE_CLIENTEP VARCHAR(20) NOT NULL,
575     APELLIDO_CLIENTEP VARCHAR(20) NOT NULL
576     );
577
578     START TRANSACTION;
579     INSERT INTO ClientesPremium VALUES (DEFAULT, 'Jose', 'Sanchez');
580     INSERT INTO ClientesPremium VALUES (DEFAULT, 'Leonel', 'Perez');
581     INSERT INTO ClientesPremium VALUES (DEFAULT, 'Sofia', 'Garcia');
582     INSERT INTO ClientesPremium VALUES (DEFAULT, 'Maria', 'Robledo');
583     SAVEPOINT primer_registro;
584     INSERT INTO ClientesPremium VALUES (DEFAULT, 'Roberto', 'Rojas');
585     INSERT INTO ClientesPremium VALUES (DEFAULT, 'Josefina', 'Valverde');
586     INSERT INTO ClientesPremium VALUES (DEFAULT, 'Uriel', 'Dominguez');
587     INSERT INTO ClientesPremium VALUES (DEFAULT, 'Gabriel', 'Godoy');
588     SAVEPOINT segundo_registro;
589     -- ROLLBACK TO primer_registro;
590     -- ROLLBACK TO segundo_registro;
591     -- RELEASE SAVEPOINT primer_registro;
592     -- COMMIT;
```

11. BackUp

Se anexara el BackUp del trabajo en la carpera de google Drive.

12. Herramientas utilizadas y sitios consultados.

- MySQL Workench 8.0
- Navicat 15
- Notepad++
- CMD (inserción de datos)
- Excel
- Paint (no hay que restarle crédito)
- Microsoft Word
- Curso de SQL Coderhouse
- <https://www.w3schools.com/sql/>
- <https://stackoverflow.com/>
- <https://database.guide/>
- <https://www.databasesstar.com/>

Especial mención y no menos profesional a los amigos que están en el tema y siempre están dispuestos a ayudar y aclarar conceptos.