

PROYECTO FINAL

BASE DE DATOS

SISTEMA DE GESTIÓN DE COMPRA Y VENTA DE INSUMOS DE COMPUTACIÓN

PROYECTO FINAL DE CODERHOUSE SQL



Link Repositorio de Github: <https://github.com/jgromerou/CursoSql>

Estudiante: Romero Uro, Juan Gerardo

Comisión: 31270

Profesor: Rodas, Miguel

CONTENIDO:

01 INTRODUCCIÓN	3
02 OBJETIVO	3
03 SITUACIÓN PROBLEMÁTICA	4
04 MODELO DE NEGOCIO	4
05 DIAGRAMA E - R	4
06 LISTADO DE TABLAS CON DESCRIPCIÓN DE SU ESTRUCTURA	7
07 SCRIPT DE CREACIÓN DE OBJETOS DE LA BASE DE DATOS	9
Creación del Script SQL - Schema y Uso del mismo:	9
Creación del Script SQL - Tablas:	10
Script SQL: Inserción de datos en cada tabla:	12
LISTADO DE VISTAS: (se crean con la BBDD seleccionada)	14
Script SQL - Creación de Funciones:	19
Script SQL Creación de Procedimientos Almacenados:	22
Script SQL Creación de Triggers:	26
Creación de Users:	32
DAR PERMISOS A LOS USERS CREADOS:	33
08 SCRIPTS DE INSERCIÓN DE DATOS	38
Backup y Restauración:	38
Backup:	38
Restauración:	40
09 INFORMES GENERADOS EN BASE A LA INFORMACIÓN ALMACENADAS EN LAS TABLAS	55
10 HERRAMIENTAS Y TECNOLOGÍAS:	56

01 INTRODUCCIÓN

Para la creación de esta base de datos surgió de la idea de organizar y controlar las compras y ventas de los productos realizados por los empleados del negocio para mantener un control y organización sobre los productos y empleados, donde cada empleado tiene un rol. Se llevan a cabo vistas importantes y una auditoria sobre los empleados y ventas de productos.

02 OBJETIVO

El objetivo de este proyecto es crear una base de datos de compras y ventas de productos de Computación, donde sus empleados son quienes se encargan de cargar datos como por ejemplo las compras y ventas de los productos, además cuenta con un control de stock y cada empleado tiene su propio usuario con un determinado rol(comprador, vendedor, administrador).

03 SITUACIÓN PROBLEMÁTICA

Surgió la necesidad de contar con un sistema para poder gestionar las operaciones como por ejemplo: compras y ventas de los productos, control de stock y de los empleados en general.

Se crea esta BBDD para evitar anomalías y redundancias de datos como sucede en Excel.

04 MODELO DE NEGOCIO

El negocio se ocupa de la compra y venta de productos de computación de origen nacional y extranjero.

05 DIAGRAMA E - R

Primer Modelo:

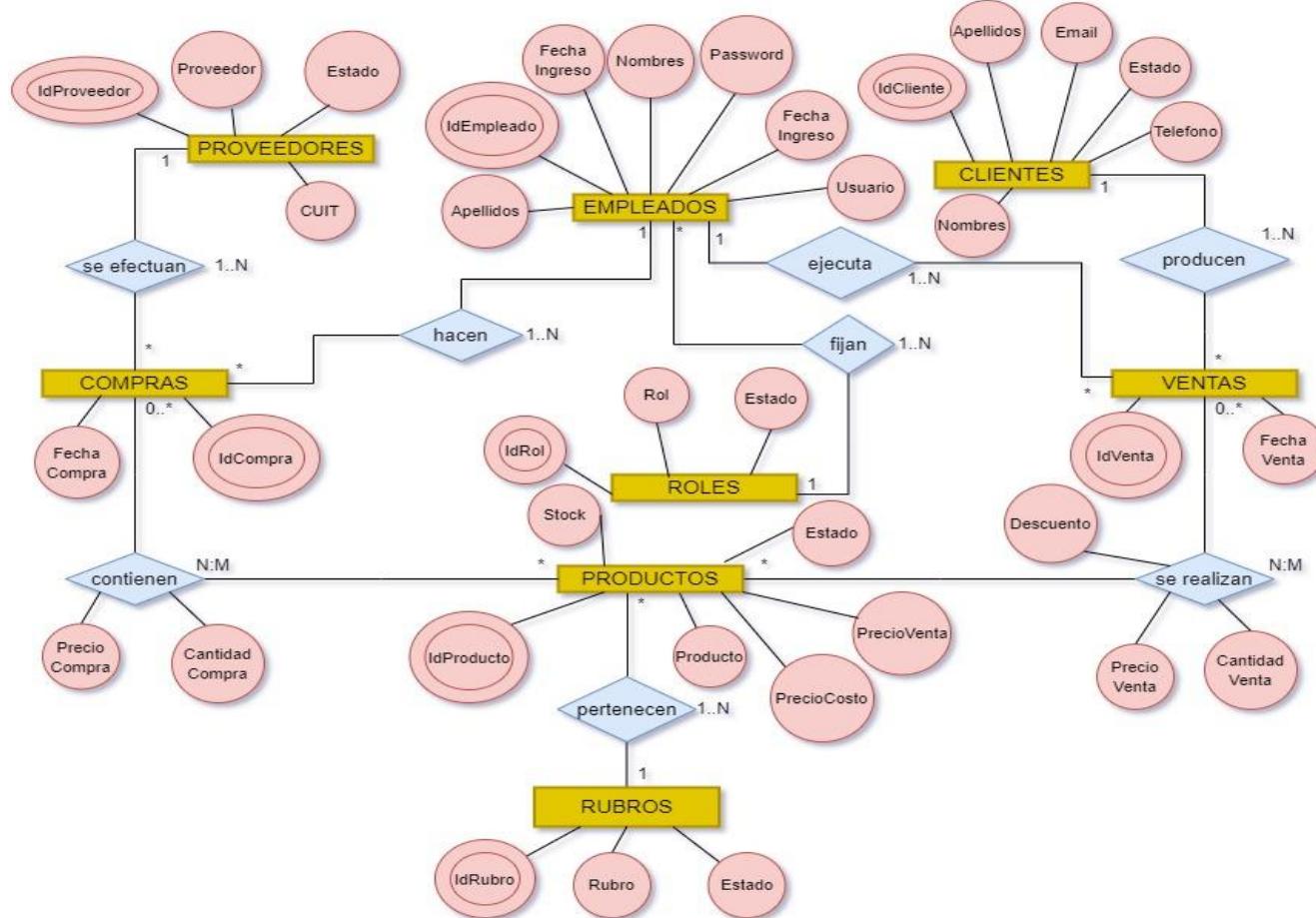
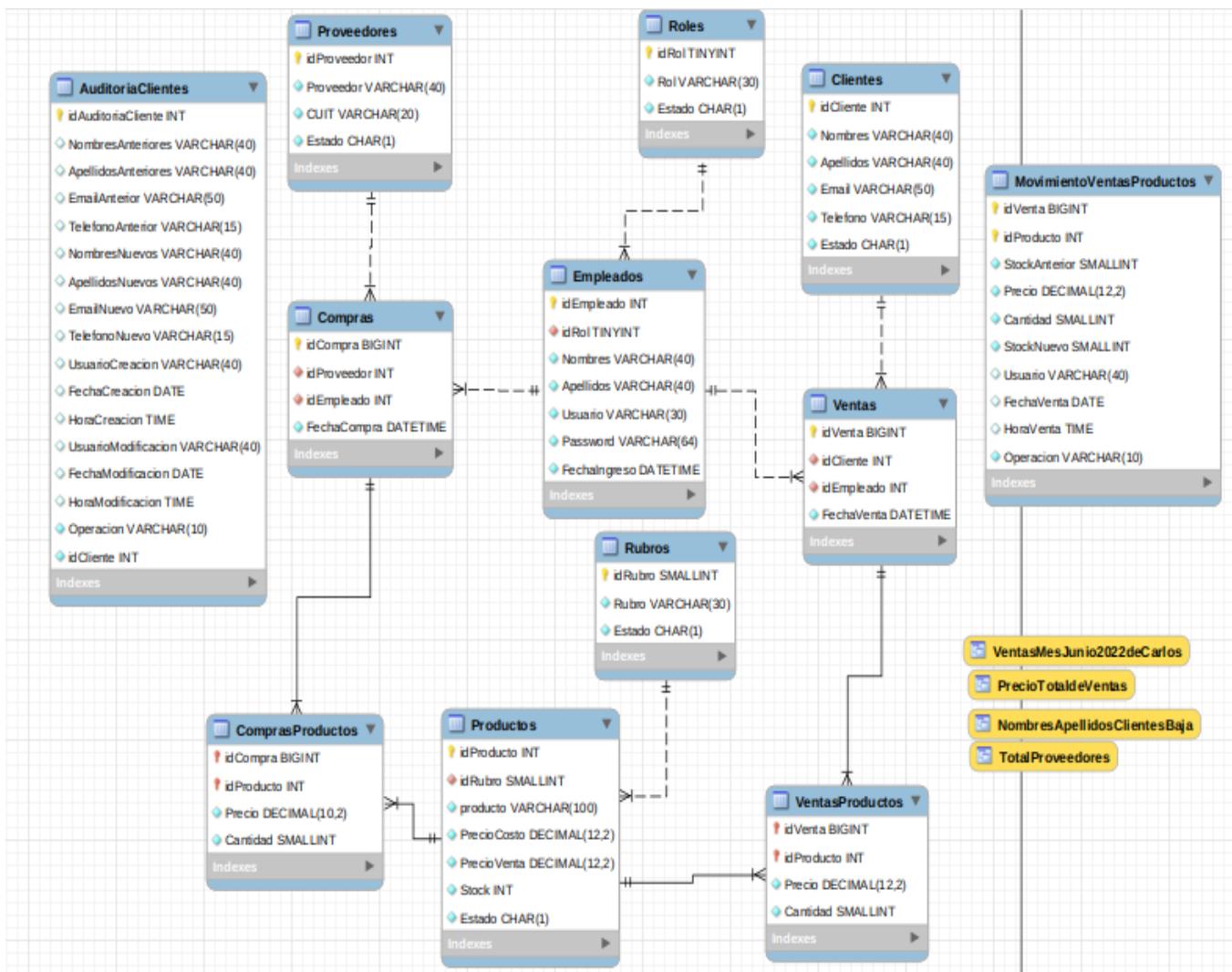


DIAGRAMA E-R (Modelo Físico)



06 LISTADO DE TABLAS CON DESCRIPCIÓN DE SU ESTRUCTURA

Tablas del sistema de compra venta de productos de Computación - Romero Uro

Roles: roles de los empleados.				
idRol (PK)	TINYINT	id de rol del empleado		
Rol	VARCHAR(30)	Rol del empleado		
Estado	CHAR(1)	Estado de Rol del empleado ->	A: alta, B: baja	
Empleados: son los únicos que tienen accesos al sistema con un determinado rol				
idEmpleado (PK)	INT	ID del empleado		
idRol (FK)	TINYINT	ID de rol		
Nombres	VARCHAR(40)	Nombre/s del empleado		
Apellidos	VARCHAR(40)	Apellido/s del empleado		
Usuario	VARCHAR(30)	Usuario de ingreso al sistema		
Password	VARCHAR(64)	Password de ingreso al sistema		
FechaIngreso	DATETIME	Fecha de ingreso al sistema		
Rubros: esta tabla contiene los rubros de los productos				
idRubro (PK)	SMALLINT	ID del rubro		
Nombre	VARCHAR(30)	Nombre de rubro		
Estado	CHAR(1)	Estado de rubro	A: alta, B: baja	
Productos: contiene los productos de Computación.				
idProducto (PK)	INT	ID del producto		
idRubro (FK)	SMALLINT	ID del rubro de producto		
Producto	VARCHAR(40)	Nombre del producto		
PrecioCosto	DECIMAL(12,2)	Precio Costo del producto		
PrecioVenta	DECIMAL(12,2)	Precio Venta del Producto		
Stock	INT	Stock		
Estado	CHAR(1)	Estado del producto ----->	A: alta, B: baja	

Proveedores: Esta tabla almacena los proveedores		
idProveedor (PK)	INT	ID del proveedor
CUIT	VARCHAR(20)	CUIT del proveedor
Proveedor	VARCHAR(40)	Nombre del proveedor
Estado	CHAR(1)	Estado del proveedor

Compras: se registran las compras de los productos de Computación		
idCompra (PK)	BIGINT	ID de compra
idProveedor (FK)	INT	ID de proveedor
idEmpleado (FK)	INT	ID de Empleado
FechaCompra	DATETIME	Fecha de Compra

ComprasProductos: Relación entre tablas compras y productos		
idCompra (FK)	BIGINT	ID de compra
idProducto (FK)	INT	ID de producto
Precio	DECIMAL(12,2)	Precio de Compra de un producto
Cantidad	SMALLINT	Cantidad de producto/s de una compra

Clientes: Esta tabla almacena los clientes		
idCliente (PK)	INT	ID del cliente
Nombres	VARCHAR(40)	Nombre/s de Cliente
Apellidos	VARCHAR(40)	Apellido/s de Cliente
Email	VARCHAR(50)	Email del cliente
Telefono	VARCHAR(15)	Telefono del cliente
Estado	CHAR(1)	Estado de cliente

Ventas: se registran las ventas de los productos de Computación		
idVenta (PK)	BIGINT	ID de venta
idCliente (FK)	INT	ID de cliente
idEmpleado (FK)	INT	ID de Empleado
FechaVenta	DATETIME	Fecha de Venta

VentasProductos: Relación entre tablas ventas y productos

idVenta (FK)	BIGINT	ID de Venta	
idProducto (FK)	INT	ID de Producto	
Precio	DECIMAL(12,2)	Precio de producto de una venta	
Cantidad	SMALLINT	Cantidad de producto/s de una venta	

07 SCRIPT DE CREACIÓN DE OBJETOS DE LA BASE DE DATOS

Creación del Script SQL - Schema y Uso del mismo:



```
1 -- CREAR SCHEMA
2 • CREATE SCHEMA `compraventainsumosromerouro`;
3 • USE `compraventainsumosromerouro`;
```

Creación del Script SQL - Tablas:

```
5
6    -- TABLA: Clientes
7  ● (−) CREATE TABLE Clientes(
8      idCliente      INT      NOT NULL,
9      Nombres        VARCHAR(40) NOT NULL,
10     Apellidos      VARCHAR(40) NOT NULL,
11     Email          VARCHAR(50) NOT NULL,
12     Telefono       VARCHAR(15) NOT NULL,
13     Estado         CHAR(1)   NOT NULL,
14     PRIMARY KEY (idCliente)
15 )ENGINE=INNODB
16 ;
17
18    -- TABLA: Compras
19  ● (−) CREATE TABLE Compras(
20      idCompra       BIGINT   NOT NULL,
21      idProveedor    INT      NOT NULL,
22      idEmpleado     INT      NOT NULL,
23      FechaCompra    DATETIME NOT NULL,
24      PRIMARY KEY (idCompra)
25 )ENGINE=INNODB
26 ;
27
28    -- TABLA: ComprasProductos
29  ● (−) CREATE TABLE ComprasProductos(
30      idCompra       BIGINT   NOT NULL,
31      idProducto     INT      NOT NULL,
32      Precio         DECIMAL(10, 2) NOT NULL,
33      Cantidad       SMALLINT NOT NULL,
34      PRIMARY KEY (idCompra, idProducto)
35 )ENGINE=INNODB
36 ;
37
```

```
36    ;
37
38    -- TABLA: Empleados
39  ● (−) CREATE TABLE Empleados(
40      idEmpleado     INT      NOT NULL,
41      idRol          TINYINT  NOT NULL,
42      Nombres        VARCHAR(40) NOT NULL,
43      Apellidos      VARCHAR(40) NOT NULL,
44      Usuario        VARCHAR(30) NOT NULL,
45      Password       VARCHAR(64) NOT NULL,
46      FechaIngreso   DATETIME NOT NULL,
47      PRIMARY KEY (idEmpleado)
48 )ENGINE=INNODB
49 ;
50
51    -- TABLA: Productos
52  ● (−) CREATE TABLE Productos(
53      idProducto     INT      NOT NULL,
54      idRubro        SMALLINT NOT NULL,
55      producto       VARCHAR(100) NOT NULL,
56      PrecioCosto   DECIMAL(12,2) NOT NULL,
57      PrecioVenta   DECIMAL(12,2) NOT NULL,
58      Stock          INT      NOT NULL,
59      Estado         CHAR(1)   NOT NULL,
60      PRIMARY KEY (idProducto)
61 )ENGINE=INNODB
62 ;
63
```



```
-- TABLA: Proveedores
CREATE TABLE Proveedores(
    idProveedor INT NOT NULL,
    Proveedor VARCHAR(40) NOT NULL,
    CUIT VARCHAR(20) NOT NULL,
    Estado CHAR(1) NOT NULL,
    PRIMARY KEY (idProveedor)
)ENGINE=INNODB;
;

-- TABLA: Roles
CREATE TABLE Roles(
    idRol TINYINT NOT NULL,
    Rol VARCHAR(30) NOT NULL,
    Estado CHAR(1) NOT NULL,
    PRIMARY KEY (idRol)
)ENGINE=INNODB;
;

-- TABLA: Rubros
CREATE TABLE Rubros(
    idRubro SMALLINT NOT NULL,
    Rubro VARCHAR(30) NOT NULL,
    Estado CHAR(1) NOT NULL,
    PRIMARY KEY (idRubro)
)ENGINE=INNODB;
;
```

```
91
92      -- TABLA: Ventas
93  ●   CREATE TABLE Ventas(
94      idVenta BIGINT NOT NULL,
95      idCliente INT NOT NULL,
96      idEmpleado INT NOT NULL,
97      FechaVenta DATETIME NOT NULL,
98      PRIMARY KEY (idVenta)
99  )ENGINE=INNODB;
100 ;
101
102      -- TABLA: VentasProductos
103  ●   CREATE TABLE VentasProductos(
104      idVenta BIGINT NOT NULL,
105      idProducto INT NOT NULL,
106      Precio DECIMAL(12, 2) NOT NULL,
107      Cantidad SMALLINT NOT NULL,
108      PRIMARY KEY (idVenta, idProducto)
109  )ENGINE=INNODB;
110 ;
111
```

```

111
112 /*Crear Tabla AuditoriaClientes*/
113 CREATE TABLE `AuditoriaClientes` (
114     `idAuditoriaCliente` int NOT NULL AUTO_INCREMENT,
115     `NombresAnteriores` varchar(40) DEFAULT NULL,
116     `ApellidosAnteriores` varchar(40) DEFAULT NULL,
117     `EmailAnterior` varchar(50) DEFAULT NULL,
118     `TelefonoAnterior` varchar(15) DEFAULT NULL,
119     `NombresNuevos` varchar(40) DEFAULT NULL,
120     `ApellidosNuevos` varchar(40) DEFAULT NULL,
121     `EmailNuevo` varchar(50) DEFAULT NULL,
122     `TelefonoNuevo` varchar(15) DEFAULT NULL,
123     `UsuarioCreacion` varchar(40) DEFAULT NULL,
124     `FechaCreacion` date DEFAULT NULL,
125     `HoraCreacion` time DEFAULT NULL,
126     `UsuarioModificacion` varchar(40) DEFAULT NULL,
127     `FechaModificacion` date DEFAULT NULL,
128     `HoraModificacion` time DEFAULT NULL,
129     `Operacion` varchar(10) NOT NULL,
130     `idCliente` int NOT NULL,
131     PRIMARY KEY (`idAuditoriaCliente`)
132 ) ENGINE=InnoDB;
133

/*Crear Tabla MovimientoVentasProductos*/
134 CREATE TABLE `MovimientoVentasProductos` (
135     `idVenta` bigint NOT NULL,
136     `idProducto` int NOT NULL,
137     `StockAnterior` smallint NOT NULL,
138     `Precio` decimal(12,2) NOT NULL,
139     `Cantidad` smallint NOT NULL,
140     `StockNuevo` smallint NOT NULL,
141     `Usuario` varchar(40) DEFAULT NULL,
142     `FechaVenta` date DEFAULT NULL,
143     `HoraVenta` time DEFAULT NULL,
144     `Operacion` varchar(10) NOT NULL,
145     PRIMARY KEY (`idVenta`,`idProducto`)
146 ) ENGINE=InnoDB;

```

En este caso se crean 12 Tablas para la BBDD `compraventainsumosromerouro`. Cada una con sus respectivos datos que luego serán llenados por la información que le corresponda a cada una.

Son las Tablas que inicialmente se pensaron en el Primer Modelo Entidad Relación.

Script SQL: Inserción de datos en cada tabla:

```

137
138 • INSERT INTO `Roles` (`idRol`, `Rol`, `Estado`) VALUES (1,'ADMIN','A');
139 • INSERT INTO `Roles` (`idRol`, `Rol`, `Estado`) VALUES (2,'EMPLEADO','A');
140 • INSERT INTO `Roles` (`idRol`, `Rol`, `Estado`) VALUES (3,'TEST','A');
141 • INSERT INTO `Roles` (`idRol`, `Rol`, `Estado`) VALUES (4,'TEST 2','B');
142
143 • INSERT INTO `Empleados` (`idEmpleado`, `idRol`, `Nombres`, `Apellidos`, `Usuario`, `Password`, `FechaIngreso`) VALUES (1,2,'Gabriela','Gouth','sgouth0','Y4l5R8Z1','2022-08-17 15:34:00');
144 • INSERT INTO `Empleados` (`idEmpleado`, `idRol`, `Nombres`, `Apellidos`, `Usuario`, `Password`, `FechaIngreso`) VALUES (2,2,'Carlos','Murny','cnurnyl','hx8gPfx0uz','2022-06-29 15:33:02');
145 • INSERT INTO `Empleados` (`idEmpleado`, `idRol`, `Nombres`, `Apellidos`, `Usuario`, `Password`, `FechaIngreso`) VALUES (3,2,'Benjamin','Shafier','bschafier2','1rYHuyPpSdv','2022-09-06 13:02:04');
146 • INSERT INTO `Empleados` (`idEmpleado`, `idRol`, `Nombres`, `Apellidos`, `Usuario`, `Password`, `FechaIngreso`) VALUES (4,2,'Bautista','Senerad','bsenerad3','y1Z5hL6','2022-04-03 14:02:10');
147 • INSERT INTO `Empleados` (`idEmpleado`, `idRol`, `Nombres`, `Apellidos`, `Usuario`, `Password`, `FechaIngreso`) VALUES (5,2,'Blanca','Yelop','byellep4','BgEnq0','2022-01-18 10:02:21');
148 • INSERT INTO `Empleados` (`idEmpleado`, `idRol`, `Nombres`, `Apellidos`, `Usuario`, `Password`, `FechaIngreso`) VALUES (6,2,'Juan','Penton','jpenton5','4rcZ47Mc','2022-12-12 08:01:02');
149 • INSERT INTO `Empleados` (`idEmpleado`, `idRol`, `Nombres`, `Apellidos`, `Usuario`, `Password`, `FechaIngreso`) VALUES (7,2,'Ricardo','Bastone','rbastone6','3L9b2G','2022-10-10 09:02:30');
150 • INSERT INTO `Empleados` (`idEmpleado`, `idRol`, `Nombres`, `Apellidos`, `Usuario`, `Password`, `FechaIngreso`) VALUES (8,2,'Liam','Hudson','lhudson7','qSWJGUx1l','2022-11-07 11:03:06');
151 • INSERT INTO `Empleados` (`idEmpleado`, `idRol`, `Nombres`, `Apellidos`, `Usuario`, `Password`, `FechaIngreso`) VALUES (9,2,'Ezequiel','Armer','earner8','vABK2ZU8db','2022-04-02 14:03:05');
152 • INSERT INTO `Empleados` (`idEmpleado`, `idRol`, `Nombres`, `Apellidos`, `Usuario`, `Password`, `FechaIngreso`) VALUES (10,2,'Florencia','Tunley','ftunley9','YsJNbq7n4a','2022-04-13 13:03:07');
153 • INSERT INTO `Empleados` (`idEmpleado`, `idRol`, `Nombres`, `Apellidos`, `Usuario`, `Password`, `FechaIngreso`) VALUES (11,2,'Juan','Penton','jpenton53','4rcZ47Mc','2022-12-12 14:15:03');
154 • INSERT INTO `Empleados` (`idEmpleado`, `idRol`, `Nombres`, `Apellidos`, `Usuario`, `Password`, `FechaIngreso`) VALUES (12,2,'Ricardo Jóse','Bastone perez','rbastone63','3L9b2G','2022-10-10 13:11:30');
155 • INSERT INTO `Empleados` (`idEmpleado`, `idRol`, `Nombres`, `Apellidos`, `Usuario`, `Password`, `FechaIngreso`) VALUES (13,2,'Liam Nicole','Hudson','lhudson73','qSWJGUx1l','2022-11-07 11:20:16');
156 • INSERT INTO `Empleados` (`idEmpleado`, `idRol`, `Nombres`, `Apellidos`, `Usuario`, `Password`, `FechaIngreso`) VALUES (14,2,'Ezequiel José','Armer','earner83','vABK2ZU8db','2022-04-02 10:50:05');
157 • INSERT INTO `Empleados` (`idEmpleado`, `idRol`, `Nombres`, `Apellidos`, `Usuario`, `Password`, `FechaIngreso`) VALUES (15,1,'Juan','Romero','jromero15','esNbq7n4a','2022-01-29 10:10:20');

338
339 • INSERT INTO `Proveedores` (`idProveedor`, `Proveedor`, `CUIT`, `Estado`) VALUES (1,'Tucuman Ofertas','28334455663','A');
340 • INSERT INTO `Proveedores` (`idProveedor`, `Proveedor`, `CUIT`, `Estado`) VALUES (2,'Exo Soluciones Informaticas','20151617483','A');
341 • INSERT INTO `Proveedores` (`idProveedor`, `Proveedor`, `CUIT`, `Estado`) VALUES (3,'Ceven','27234242221','A');
342 • INSERT INTO `Proveedores` (`idProveedor`, `Proveedor`, `CUIT`, `Estado`) VALUES (4,'Computronic SRL','28345234241','A');
343 • INSERT INTO `Proveedores` (`idProveedor`, `Proveedor`, `CUIT`, `Estado`) VALUES (5,'Nh Computación','20342532523','A');
344 • INSERT INTO `Proveedores` (`idProveedor`, `Proveedor`, `CUIT`, `Estado`) VALUES (6,'Compuciti','32423423424','B');
345 • INSERT INTO `Proveedores` (`idProveedor`, `Proveedor`, `CUIT`, `Estado`) VALUES (7,'Tecnologic','32423424241','A');
346 • INSERT INTO `Proveedores` (`idProveedor`, `Proveedor`, `CUIT`, `Estado`) VALUES (8,'Amazon','34324324324','A');
347 • INSERT INTO `Proveedores` (`idProveedor`, `Proveedor`, `CUIT`, `Estado`) VALUES (9,'Alum Computación','28343242352','B');
348 • INSERT INTO `Proveedores` (`idProveedor`, `Proveedor`, `CUIT`, `Estado`) VALUES (10,'SYSCOM','20345324324','A');
349 • INSERT INTO `Proveedores` (`idProveedor`, `Proveedor`, `CUIT`, `Estado`) VALUES (11,'Compuamaq','22343243242','A');
350 • INSERT INTO `Proveedores` (`idProveedor`, `Proveedor`, `CUIT`, `Estado`) VALUES (12,'Punto Com punto ar','33234234234','A');
351 • INSERT INTO `Proveedores` (`idProveedor`, `Proveedor`, `CUIT`, `Estado`) VALUES (13,'Pronet','234233532522','A');
352 • INSERT INTO `Proveedores` (`idProveedor`, `Proveedor`, `CUIT`, `Estado`) VALUES (14,'Mundo PC','22342342555','B');
353 • INSERT INTO `Proveedores` (`idProveedor`, `Proveedor`, `CUIT`, `Estado`) VALUES (15,'Mercado Libre','22342343245','A');
354
355 • INSERT INTO `Clientes` (`idCliente`, `Nombres`, `Apellidos`, `Email`, `Telefono`, `Estado`) VALUES (1,'Cort','Cogle','ccogle0@last.fm','269-433-3884','A');
356 • INSERT INTO `Clientes` (`idCliente`, `Nombres`, `Apellidos`, `Email`, `Telefono`, `Estado`) VALUES (2,'Petronilla','Foro','pfrolgchicagotribune.com','265-763-9016','A');
357 • INSERT INTO `Clientes` (`idCliente`, `Nombres`, `Apellidos`, `Email`, `Telefono`, `Estado`) VALUES (3,'Adeline','Beattie','beattie2@alex.com','513-698-3743','A');
358 • INSERT INTO `Clientes` (`idCliente`, `Nombres`, `Apellidos`, `Email`, `Telefono`, `Estado`) VALUES (4,'Ruggiero','Rosenfeld','rrosenfeld3@jiathis.com','226-126-2672','B');
359 • INSERT INTO `Clientes` (`idCliente`, `Nombres`, `Apellidos`, `Email`, `Telefono`, `Estado`) VALUES (5,'Bermetta','Crews','bcrews4@wikimedia.org','807-436-6134','B');
360 • INSERT INTO `Clientes` (`idCliente`, `Nombres`, `Apellidos`, `Email`, `Telefono`, `Estado`) VALUES (6,'Flory','MacMaster','fmacmaster5@wp.com','701-788-0167','A');
361 • INSERT INTO `Clientes` (`idCliente`, `Nombres`, `Apellidos`, `Email`, `Telefono`, `Estado`) VALUES (7,'Trey','Patkin','tpatkin6@tamu.edu','360-844-7252','A');
362 • INSERT INTO `Clientes` (`idCliente`, `Nombres`, `Apellidos`, `Email`, `Telefono`, `Estado`) VALUES (8,'Dinirity','Markel','dmarkel7@google.co.uk','140-838-7270','A');
363 • INSERT INTO `Clientes` (`idCliente`, `Nombres`, `Apellidos`, `Email`, `Telefono`, `Estado`) VALUES (9,'Roxanna','Mara','mara8@cornell.edu','764-148-6397','A');
364 • INSERT INTO `Clientes` (`idCliente`, `Nombres`, `Apellidos`, `Email`, `Telefono`, `Estado`) VALUES (10,'Ailina','Gamage','agamage9@ocon.ne.jp','398-327-2429','A');
365 • INSERT INTO `Clientes` (`idCliente`, `Nombres`, `Apellidos`, `Email`, `Telefono`, `Estado`) VALUES (11,'Cully','Cocking','ccocking@va.gov','748-271-1565','A');
366 • INSERT INTO `Clientes` (`idCliente`, `Nombres`, `Apellidos`, `Email`, `Telefono`, `Estado`) VALUES (12,'Correy','Wortt','cworttbgravatar.com','287-364-0543','A');
367 • INSERT INTO `Clientes` (`idCliente`, `Nombres`, `Apellidos`, `Email`, `Telefono`, `Estado`) VALUES (13,'Nikolaus','Tutin','ntutinc@newyorker.com','941-689-1061','A');
368 • INSERT INTO `Clientes` (`idCliente`, `Nombres`, `Apellidos`, `Email`, `Telefono`, `Estado`) VALUES (14,'Renate','Tetlow','rtetlowd@slate.com','666-317-3679','A');
369 • INSERT INTO `Clientes` (`idCliente`, `Nombres`, `Apellidos`, `Email`, `Telefono`, `Estado`) VALUES (15,'Livy','Stoeckle','lstoekle@sakura.ne.jp','300-994-0533','A');
370
371 • INSERT INTO `Rubros` (`idRubro`, `Rubro`, `Estado`) VALUES (1,'Impresora','A');
372 • INSERT INTO `Rubros` (`idRubro`, `Rubro`, `Estado`) VALUES (2,'Mouse','A');
373 • INSERT INTO `Rubros` (`idRubro`, `Rubro`, `Estado`) VALUES (3,'Teclado','A');
374 • INSERT INTO `Rubros` (`idRubro`, `Rubro`, `Estado`) VALUES (4,'Discos Rígidos','A');
375 • INSERT INTO `Rubros` (`idRubro`, `Rubro`, `Estado`) VALUES (5,'Memoria RAM','A');
376 • INSERT INTO `Rubros` (`idRubro`, `Rubro`, `Estado`) VALUES (6,'Placa de Video','A');
377 • INSERT INTO `Rubros` (`idRubro`, `Rubro`, `Estado`) VALUES (7,'Monitor','A');
378 • INSERT INTO `Rubros` (`idRubro`, `Rubro`, `Estado`) VALUES (8,'Cables VGA','A');
379 • INSERT INTO `Rubros` (`idRubro`, `Rubro`, `Estado`) VALUES (9,'Cables HDMI','A');
380 • INSERT INTO `Rubros` (`idRubro`, `Rubro`, `Estado`) VALUES (10,'Cargador Notebook','A');
381 • INSERT INTO `Rubros` (`idRubro`, `Rubro`, `Estado`) VALUES (11,'Tablet','A');
382 • INSERT INTO `Rubros` (`idRubro`, `Rubro`, `Estado`) VALUES (12,'Pendrive','A');
383 • INSERT INTO `Rubros` (`idRubro`, `Rubro`, `Estado`) VALUES (13,'Procesador','A');
384 • INSERT INTO `Rubros` (`idRubro`, `Rubro`, `Estado`) VALUES (14,'Modem','A');
385 • INSERT INTO `Rubros` (`idRubro`, `Rubro`, `Estado`) VALUES (15,'Placa de Red','A');
386
387 • INSERT INTO `Productos` (`idProducto`, `idRubro`, `producto`, `PrecioCosto`, `PrecioVenta`, `Stock`, `Estado`) VALUES (1,4,'Disco WD 1Tb',4800.00,8000.00,50,'A');
388 • INSERT INTO `Productos` (`idProducto`, `idRubro`, `producto`, `PrecioCosto`, `PrecioVenta`, `Stock`, `Estado`) VALUES (2,1,'Impresora Epson Tx241',9000.00,15000.00,40,'A');
389 • INSERT INTO `Productos` (`idProducto`, `idRubro`, `producto`, `PrecioCosto`, `PrecioVenta`, `Stock`, `Estado`) VALUES (3,2,'Mouse Genius Dx300',1800.00,3000.00,30,'A');
390 • INSERT INTO `Productos` (`idProducto`, `idRubro`, `producto`, `PrecioCosto`, `PrecioVenta`, `Stock`, `Estado`) VALUES (4,2,'Mouse Logitech G605',4200.00,7000.00,40,'B');
391 • INSERT INTO `Productos` (`idProducto`, `idRubro`, `producto`, `PrecioCosto`, `PrecioVenta`, `Stock`, `Estado`) VALUES (5,4,'Disco WD Green 4000 GB 550',6000.00,10000.00,55,'A');
392 • INSERT INTO `Productos` (`idProducto`, `idRubro`, `producto`, `PrecioCosto`, `PrecioVenta`, `Stock`, `Estado`) VALUES (6,5,'Memoria RAM 2GB HyperX Fury',3600.00,6000.00,50,'A');
393 • INSERT INTO `Productos` (`idProducto`, `idRubro`, `producto`, `PrecioCosto`, `PrecioVenta`, `Stock`, `Estado`) VALUES (7,6,'Monitor 22"',Samsung D3x1,24000.00,40000.00,30,'B');
394 • INSERT INTO `Productos` (`idProducto`, `idRubro`, `producto`, `PrecioCosto`, `PrecioVenta`, `Stock`, `Estado`) VALUES (8,12,'Pendrive 32 Gb Kingston d300',12000.00,20000.00,45,'A');
395 • INSERT INTO `Productos` (`idProducto`, `idRubro`, `producto`, `PrecioCosto`, `PrecioVenta`, `Stock`, `Estado`) VALUES (9,6,'Placa de Video GTX 3060 OC',96000.00,160000.00,10,'A');
396 • INSERT INTO `Productos` (`idProducto`, `idRubro`, `producto`, `PrecioCosto`, `PrecioVenta`, `Stock`, `Estado`) VALUES (10,3,'Teclado Genius KB300',3600.00,6000.00,25,'A');
397 • INSERT INTO `Productos` (`idProducto`, `idRubro`, `producto`, `PrecioCosto`, `PrecioVenta`, `Stock`, `Estado`) VALUES (11,11,'Notebook HP 15.6'' Intel i3 256 Gb SSD 16 Gb RAM',120000.00,200000.00,6,'A');
398 • INSERT INTO `Productos` (`idProducto`, `idRubro`, `producto`, `PrecioCosto`, `PrecioVenta`, `Stock`, `Estado`) VALUES (12,5,'Memoria RAM 8GB Kingston ValueRAM DDR3',3000.00,5000.00,20,'A');
399 • INSERT INTO `Productos` (`idProducto`, `idRubro`, `producto`, `PrecioCosto`, `PrecioVenta`, `Stock`, `Estado`) VALUES (13,12,'Pendrive 16 Gb Kingston x40',7200.00,12000.00,25,'A');
400 • INSERT INTO `Productos` (`idProducto`, `idRubro`, `producto`, `PrecioCosto`, `PrecioVenta`, `Stock`, `Estado`) VALUES (14,6,'Placa de Video GT 1060 4 Gb',30000.18,50000.30,30,'A');
401 • INSERT INTO `Productos` (`idProducto`, `idRubro`, `producto`, `PrecioCosto`, `PrecioVenta`, `Stock`, `Estado`) VALUES (15,14,'Modem Tp-Link 300Mbps WR-850RE',2400.30,4000.50,40,'A');
...

```

```

482
483 • INSERT INTO `Compras` (`idCompra`, `idProveedor`, `idEmpleado`, `FechaCompra`) VALUES (1,2,3,'2022-06-28 15:11:02');
484 • INSERT INTO `Compras` (`idCompra`, `idProveedor`, `idEmpleado`, `FechaCompra`) VALUES (2,1,4,'2022-06-28 15:11:33');
485 • INSERT INTO `Compras` (`idCompra`, `idProveedor`, `idEmpleado`, `FechaCompra`) VALUES (3,5,4,'2022-06-28 15:11:58');
486 • INSERT INTO `Compras` (`idCompra`, `idProveedor`, `idEmpleado`, `FechaCompra`) VALUES (4,6,2,'2022-06-28 15:12:58');
487 • INSERT INTO `Compras` (`idCompra`, `idProveedor`, `idEmpleado`, `FechaCompra`) VALUES (5,7,3,'2022-06-28 15:15:58');
488
489 • INSERT INTO `ComprasProductos` (`idCompra`, `idProducto`, `Precio`, `Cantidad`) VALUES (1,1,4800.00,2);
490 • INSERT INTO `ComprasProductos` (`idCompra`, `idProducto`, `Precio`, `Cantidad`) VALUES (1,3,1800.00,3);
491 • INSERT INTO `ComprasProductos` (`idCompra`, `idProducto`, `Precio`, `Cantidad`) VALUES (1,5,6000.00,1);
492 • INSERT INTO `ComprasProductos` (`idCompra`, `idProducto`, `Precio`, `Cantidad`) VALUES (1,6,3600.00,4);
493 • INSERT INTO `ComprasProductos` (`idCompra`, `idProducto`, `Precio`, `Cantidad`) VALUES (2,1,4800.00,4);
494 • INSERT INTO `ComprasProductos` (`idCompra`, `idProducto`, `Precio`, `Cantidad`) VALUES (3,1,4800.00,3);
495 • INSERT INTO `ComprasProductos` (`idCompra`, `idProducto`, `Precio`, `Cantidad`) VALUES (3,4,4200.00,2);
496 • INSERT INTO `ComprasProductos` (`idCompra`, `idProducto`, `Precio`, `Cantidad`) VALUES (3,6,3600.00,5);
497 • INSERT INTO `ComprasProductos` (`idCompra`, `idProducto`, `Precio`, `Cantidad`) VALUES (3,10,3600.00,7);
498 • INSERT INTO `ComprasProductos` (`idCompra`, `idProducto`, `Precio`, `Cantidad`) VALUES (4,1,4800.00,4);
499 • INSERT INTO `ComprasProductos` (`idCompra`, `idProducto`, `Precio`, `Cantidad`) VALUES (4,10,3600.00,6);
500 • INSERT INTO `ComprasProductos` (`idCompra`, `idProducto`, `Precio`, `Cantidad`) VALUES (4,11,120000.00,2);
501 • INSERT INTO `ComprasProductos` (`idCompra`, `idProducto`, `Precio`, `Cantidad`) VALUES (5,1,4800.00,3);
502 • INSERT INTO `ComprasProductos` (`idCompra`, `idProducto`, `Precio`, `Cantidad`) VALUES (5,10,3600.00,2);
503 • INSERT INTO `ComprasProductos` (`idCompra`, `idProducto`, `Precio`, `Cantidad`) VALUES (5,15,2400.30,4);
504
505 • INSERT INTO `Ventas` (`idVenta`, `idCliente`, `idEmpleado`, `FechaVenta`) VALUES (1,1,2,'2022-06-28 15:01:02');
506 • INSERT INTO `Ventas` (`idVenta`, `idCliente`, `idEmpleado`, `FechaVenta`) VALUES (2,2,2,'2022-06-28 15:02:00');
507 • INSERT INTO `Ventas` (`idVenta`, `idCliente`, `idEmpleado`, `FechaVenta`) VALUES (3,3,1,'2022-06-28 15:02:31');
508 • INSERT INTO `Ventas` (`idVenta`, `idCliente`, `idEmpleado`, `FechaVenta`) VALUES (4,3,3,'2022-06-28 15:03:24');
509 • INSERT INTO `Ventas` (`idVenta`, `idCliente`, `idEmpleado`, `FechaVenta`) VALUES (5,4,4,'2022-06-28 15:04:00');
510 • INSERT INTO `Ventas` (`idVenta`, `idCliente`, `idEmpleado`, `FechaVenta`) VALUES (6,2,2,'2022-06-28 15:04:59');

511
512 • INSERT INTO `VentasProductos` (`idVenta`, `idProducto`, `Precio`, `Cantidad`) VALUES (1,2,15000.00,1);
513 • INSERT INTO `VentasProductos` (`idVenta`, `idProducto`, `Precio`, `Cantidad`) VALUES (2,6,6000.00,2);
514 • INSERT INTO `VentasProductos` (`idVenta`, `idProducto`, `Precio`, `Cantidad`) VALUES (2,9,160000.00,1);
515 • INSERT INTO `VentasProductos` (`idVenta`, `idProducto`, `Precio`, `Cantidad`) VALUES (3,3,6000.00,3);
516 • INSERT INTO `VentasProductos` (`idVenta`, `idProducto`, `Precio`, `Cantidad`) VALUES (4,1,8000.00,2);
517 • INSERT INTO `VentasProductos` (`idVenta`, `idProducto`, `Precio`, `Cantidad`) VALUES (4,5,10000.00,1);
518 • INSERT INTO `VentasProductos` (`idVenta`, `idProducto`, `Precio`, `Cantidad`) VALUES (4,8,2000.00,4);
519 • INSERT INTO `VentasProductos` (`idVenta`, `idProducto`, `Precio`, `Cantidad`) VALUES (4,10,6000.00,3);
520 • INSERT INTO `VentasProductos` (`idVenta`, `idProducto`, `Precio`, `Cantidad`) VALUES (5,4,7000.00,2);
521 • INSERT INTO `VentasProductos` (`idVenta`, `idProducto`, `Precio`, `Cantidad`) VALUES (5,5,10000.00,1);
522 • INSERT INTO `VentasProductos` (`idVenta`, `idProducto`, `Precio`, `Cantidad`) VALUES (6,1,8000.00,2);
523 • INSERT INTO `VentasProductos` (`idVenta`, `idProducto`, `Precio`, `Cantidad`) VALUES (6,5,10000.00,1);
524 • INSERT INTO `VentasProductos` (`idVenta`, `idProducto`, `Precio`, `Cantidad`) VALUES (6,7,40000.00,1);
525 • INSERT INTO `VentasProductos` (`idVenta`, `idProducto`, `Precio`, `Cantidad`) VALUES (6,8,2000.00,4);
526 • INSERT INTO `VentasProductos` (`idVenta`, `idProducto`, `Precio`, `Cantidad`) VALUES (6,9,160000.00,1);
527

```

Para la inserción de datos, se utilizó la instrucción INSERT INTO:

`INSERT INTO Nombre_Tabla (PrimeraColumna, SegundaColumna, etc.) VALUES ("Dato1","Dato2",etc);`

Se insertan al comienzo 15 Registros para cada una de las Tablas y sus correspondientes columnas.

LISTADO DE VISTAS: (se crean con la BBDD seleccionada)

➤ Primera Vista: TotalProveedores.

OBJETIVO: Mostrar el Total de Proveedores.

DESCRIPCIÓN: La idea principal de esta vista es que la persona que vaya a utilizar la base de datos sepa el Número total de Proveedores.

TABLA QUE COMPONE A LA VISTA: La tabla que componen la vista, TotalProveedores es: Proveedores.

Creación del Script para la Vista - TotalProveedores:

```
1      /*Total de Proveedores*/  
2 •  CREATE OR REPLACE VIEW TotalProveedores AS  
3          (SELECT COUNT(*) AS TotalProveedores FROM Proveedores);
```

EJEMPLO VISTA, TotalProveedores:

#	TotalProveedores
1	15

➤ Segunda Vista: NombresApellidosClientesBaja

OBJETIVO: Mostrar Nombres y Apellidos dados de baja.

DESCRIPCIÓN: La idea principal de esta vista es que la persona que vaya a utilizar la base de datos sepa los Nombres y Apellidos de Clientes dados de baja.

TABLA QUE COMPONE A LA VISTA: La tabla que componen la vista, NombresApellidosClientesBaja es: Clientes.

Creación del Script para la Vista - NombresApellidosClientesBaja:

```
/*Nombres y Apellidos de Clientes dados de Baja */  
CREATE OR REPLACE VIEW NombresApellidosClientesBaja AS  
  (SELECT DISTINCT c.Nombres,c.Apellidos  FROM Clientes c WHERE c.Estado = 'B')|
```

EJEMPLO VISTA, NombresApellidosClientesBaja:

Result Grid		Filter Rows:
#	Nombres Apellidos	
1	Ruggiero Rosenfeld	
2	Bernetta Crews	

➤ Tercera Vista: MostrarEmpleadosRolActivo

OBJETIVO: Mostrar idEmpleado, idRol, Usuario, Nombres, Apellidos, Password y FechaIngreso con Rol Activo.

DESCRIPCIÓN: La idea principal de esta vista es que la persona que vaya a utilizar la base de datos sepa id del Empleado, idRol, Usuario, los Nombres, Apellidos, Password y Fecha de Ingreso de los Empleados con Rol Activo.

TABLA QUE COMPONE A LA VISTA: Las tablas que componen la vista, **MostrarEmpleadosRolActivo** es: Empleados y Roles.

Creación del Script para la Vista - MostrarEmpleadosRolActivo:

```
/* Mostrar los Empleados con roles activos */  
CREATE OR REPLACE VIEW MostrarEmpleadosRolActivo AS  
  (SELECT *  
   FROM Empleados  
   WHERE idRol IN (SELECT idRol FROM Roles WHERE Estado = 'A')  
   ORDER BY Empleados.idEmpleado);
```

EJEMPLO VISTA, MostrarEmpleadosRolActivo:

#	idEmpleado	idRol	Nombres	Apellidos	Usuario	Password	FechaIngreso
1	1	2	Gabriela	Gouth	sgouth0	Y4i5R8Z1j	2022-08-17 15:34:00
2	2	2	Carlos	Nurny	cnurny1	hx8gPFx0uz	2022-06-29 15:33:02
3	3	2	Benjamin	Shafier	bshafier2	1rYHuyPpSdv	2022-09-06 13:02:04
4	4	2	Bautista	Semerad	bsemerad3	yiZSrbL6	2022-04-03 14:02:10
5	5	2	Blanca	Yellep	byellep4	BgEnqQ	2022-01-18 10:02:21
6	6	2	Juan	Pentony	jpentony5	4rcZ47Mc	2022-12-12 08:01:02
7	7	2	Ricardo	Bastone	rbastone6	3L9b2G	2022-10-10 09:02:30
8	8	2	Liam	Hudson	lhudson7	qSWJGXUsll	2022-11-07 11:03:06
9	9	2	Ezequiel	Armer	earmer8	vA8K2ZU8db	2022-04-02 14:03:05
10	10	2	Florencia	Tunsley	ftunsley9	YsJW8q7N4a	2022-04-13 13:03:07
11	11	2	Juan	Pentony	jpentony53	4rcZ47Mc	2022-12-12 14:15:03
12	12	2	Ricard...	Baston...	rbastone63	3L9b2G	2022-10-10 13:11:30
13	13	2	Liam N...	Hudson	lhudson73	qSWJGXUsll	2022-11-07 11:20:16
14	14	2	Ezequi...	Armer	earmer83	vA8K2ZU8db	2022-04-02 10:50:05
15	15	1	Juan	Romero	jromero15	esJW8q7N4a	2022-01-29 10:10:20

➤ Cuarta Vista: VentasMesJunio2022deCarlos

OBJETIVO: Obtener el total de las ventas por el rango de fechas y por el nombre de empleados.

DESCRIPCIÓN: La idea principal de esta vista es que la persona que vaya a utilizar la base de datos sepa el total de las ventas por rango de fecha y por el nombre del empleado como por ejemplo: Ventas del mes de Junio del año 2022 de los empleados llamados Carlos.

TABLA QUE COMPONE A LA VISTA: Las tablas que componen la vista, **VentasMesJunio2022deCarlos** es: Empleados y Ventas.

Creación del Script para la Vista - VentasMesJunio2022deCarlos:

```
/*Ventas del mes de Junio del año 2022 de los empleados llamados Carlos */
CREATE OR REPLACE VIEW VentasMesJunio2022deCarlos AS
  (SELECT
    v.FechaVenta AS FechaVenta,
    v.idVenta AS idVenta,
    v.idEmpleado AS NroEmpleado,
    e.Nombres AS Nombres,
    e.Apellidos AS Apellidos
   FROM
    (Ventas v
   JOIN Empleados e
     ON ((v.idEmpleado = e.idEmpleado)))
  WHERE
    ((v.FechaVenta BETWEEN '2022-06-01' AND '2022-06-30')
     AND (e.Nombres LIKE '%Carlos%'))
  ORDER BY v.FechaVenta);
```

EJEMPLO VISTA, VentasMesJunio2022deCarlos:

#	FechaVenta	idVenta	NroEmpleado	Nombres	Apellidos
1	2022-06-28 15:01:02	1	2	Carlos	Nurny
2	2022-06-28 15:02:00	2	2	Carlos	Nurny
3	2022-06-28 15:04:59	6	2	Carlos	Nurny

➤ Quinta Vista: PrecioTotaldeVentas

OBJETIVO: Mostrar los precios totales de las Ventas.

DESCRIPCIÓN: La idea principal de esta vista es que la persona que vaya a utilizar la base de datos sepa el total de las ventas realizadas.

TABLA QUE COMPONE A LA VISTA: Las tablas que componen la vista, **PrecioTotaldeVentas** es: Ventas y VentasProductos.

Creación del Script para la Vista - PrecioTotaldeVentas:

```
1  /*Precio Total por Venta*/
2 • CREATE OR REPLACE VIEW PrecioTotaldeVentas AS
3      (SELECT
4          v.idVenta AS idVenta,
5          SUM(vp.Precio) AS PrecioTotal
6      FROM
7          (Ventas v
8              JOIN VentasProductos vp
9                  ON (v.idVenta = vp.idVenta))
10         GROUP BY v.idVenta);
```

EJEMPLO VISTA, PrecioTotaldeVentas:

#	idVenta	PrecioTotal
1	1	15000.00
2	2	166000.00
3	3	6000.00
4	4	26000.00
5	5	17000.00
6	6	227500.00

Script SQL - Creación de Funciones:

Con el objetivo de procesar y manipular datos de forma procedural y eficiente, se presentan las siguientes funciones para la Base de Datos.

• Función saberEstadoProveedor:

OBJETIVO: Saber el Estado de un Proveedor.

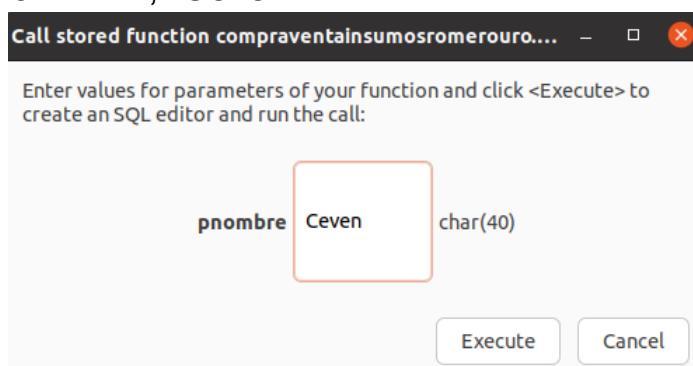
DESCRIPCIÓN: Para realizar la función **saberEstadoProveedor**, se debe ingresar el parámetro pnombre (de tipo CHAR(40)). El resultado que arroja la función es del tipo CHAR(255) donde si está activo devuelve “El proveedor ,‘pnombre’, está activo” y si está dado de baja: “El proveedor ,‘pnombre’, no está activo”.

TABLA QUE INTERVIENE EN LA FUNCIÓN: La tabla que interviene en dicha función es Proveedores.

Creación del Script para la Función - saberEstadoProveedor:

```
1  /*Para saber el Estado de un Proveedor*/
2  DELIMITER $$ 
3 • CREATE FUNCTION saberEstadoProveedor(pnombre char(40))
4  RETURNS CHAR(255)
5  READS SQL DATA
6  BEGIN
7      DECLARE cantidad int DEFAULT 0;
8      SET cantidad = (SELECT count(*) FROM Proveedores WHERE Proveedor=pnombre AND Estado='A');
9      IF(cantidad=1) THEN
10         RETURN CONCAT('El proveedor ',pnombre,' está activo');
11     ELSE
12         RETURN CONCAT('El proveedor ',pnombre,' no está activo');
13     END IF;
14 END $$
```

EJEMPLO DE saberEstadoProveedor: Nos saldrá un recuadro como el siguiente donde deberemos colocar el Nombre del Proveedor que queremos. En este caso será uno que tenemos en la BBDD, “Ceven”.



Resultado de Ejecutar la Función saberEstadoProveedor:

```
#  compraventainsumosromerourou.sa
1  El proveedor Ceven está activo
```

- **Función saberVentasEmpleadoporsuNombre:**

OBJETIVO: Saber las Ventas de un Empleado por su Nombre.

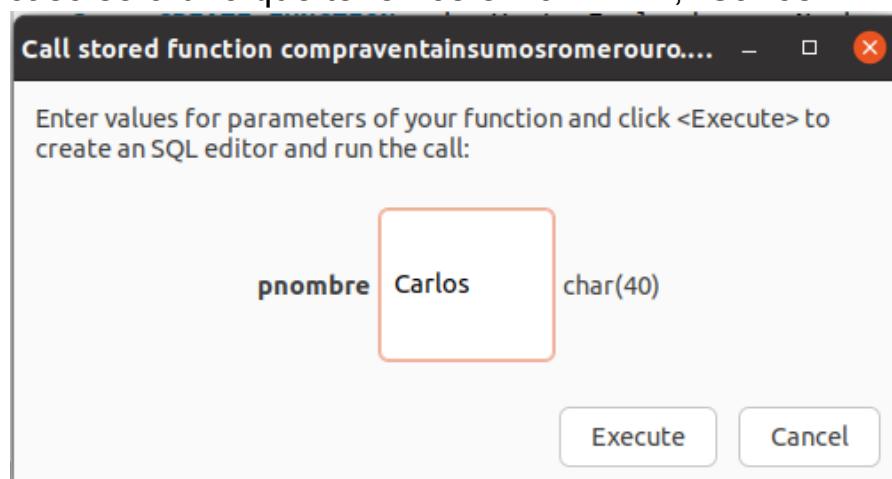
DESCRIPCIÓN: Para realizar la función **saberVentasEmpleadoporsuNombre**, se debe ingresar el parámetro *pnombre* (de tipo CHAR(40)). El resultado que arroja la función es del tipo CHAR(255) donde devuelve el siguiente mensaje: “*,pnombre, hizo n ventas*”.

TABLAS QUE INTERVIENEN EN LA FUNCIÓN: Las tablas que intervienen en dicha función son: Empleados y Ventas.

Creación del Script para la Función - saberVentasEmpleadoporsuNombre:

```
1  /*Para saber cuantas ventas hizo un/a empleado/a por su nombre */
2  DELIMITER $$ 
3 • CREATE FUNCTION saberVentasEmpleadoporsuNombre(pnombre char(40))
4  RETURNS CHAR(255)
5  READS SQL DATA
6  BEGIN
7      DECLARE cantidad int DEFAULT 0;
8      SET cantidad = (SELECT count(*) FROM Ventas v
9                  INNER JOIN Empleados e
10                 ON v.idEmpleado = e.idEmpleado
11                 WHERE e.Nombres=pnombre);
12      RETURN CONCAT(' ',pnombre,' realizó ',cantidad,' ventas');
13 END $$
```

EJEMPLO DE saberVentasEmpleadoporsuNombre: Nos saldrá un recuadro como el siguiente donde deberemos colocar el Nombre del Empleado que queremos saber la cantidad de Ventas que hizo. En este caso será uno que tenemos en la BBDD, “Carlos”.



Resultado de Ejecutar la Función saberVentasEmpleadoporsuNombre:

```
# compraventainsumosromerouro.sa
1 Carlos realizó 3 ventas
```

Script SQL Creación de Procedimientos Almacenados:

❖ 1° PROCEDIMIENTO ALMACENADO sp_get_empleados_ordenamiento:

OBJETIVO: Mostrar el ordenamiento de la tabla Empleados indicando el campo y el tipo de orden (Ascendente o Descendente).

DESCRIPCIÓN: Para utilizar el procedimiento almacenado, se debe indicar a través de un parámetro el campo de ordenamiento de una tabla y mediante un segundo parámetro, si el orden es Ascendente o Descendente. El resultado que arrojará el procedimiento será mostrar una tabla con el orden indicado por parámetro.

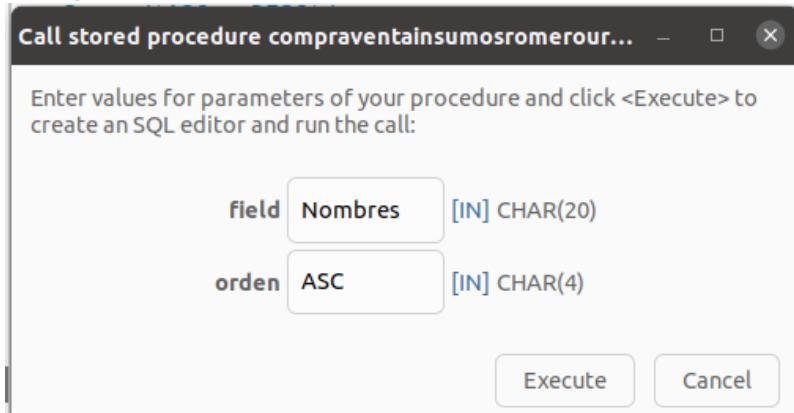
TABLAS QUE INTERVIENEN EN EL PROCEDIMIENTO: La tabla que interviene en este procedimiento es Empleados.

Creación del Script para PROCEDIMIENTO ALMACENADO sp_get_empleados_ordenamiento:

```
/*1º Stored Procedure: Permite indicar a través de un parámetro el campo de ordenamiento
y mediante un segundo parámetro, si el orden es Ascendente o Descendente.*/
/*ASC o DESC*/
DELIMITER $$

CREATE DEFINER=`root`@`%` PROCEDURE `sp_get_empleados_ordenamiento`(
    IN field CHAR(20),
    IN orden CHAR(4)
)
BEGIN
    IF field <> '' THEN
        SET @rubro_order = concat('ORDER BY ',field,' ',orden,'');
    ELSE
        SET @rubro_order = '';
    END IF;
    SET @clausula = concat('SELECT * FROM Empleados ',@rubro_order);
    PREPARE runSQL FROM @clausula;
    EXECUTE runSQL;
    DEALLOCATE PREPARE runSQL;
END$$
```

Ejecutemos el Script para el PROCEDIMIENTO ALMACENADO `sp_get_empleados_ordenamiento`:



Finalmente el Resultado que obtenemos es el siguiente:
Está ordenado por el campo Nombres de Forma Ascendente(A-Z)

#	idEmplead	idRol	Nombres	Apellidos	Usuario	Password	Fechaingreso
1	4	2	Bautista	Semerad	bsem erad3	yiZSrbL6	2022-04-03 14:02:10
2	3	2	Benjamin	Shafier	bshafier2	1rYHuyPpSdv	2022-09-06 13:02:04
3	5	2	Blanca	Yellep	byellep4	BgEnqQ	2022-01-18 10:02:21
4	2	2	Carlos	Nurny	cnurny1	hx8gPFx0uz	2022-06-29 15:33:02
5	9	2	Ezequiel	Armer	earmer8	vA8K2ZU8db	2022-04-02 14:03:05
6	14	2	Ezequiel Jose	Armer	earmer83	vA8K2ZU8db	2022-04-02 10:50:05
7	10	2	Florencia	Tunsley	ftunsley9	YsJW8q7N4a	2022-04-13 13:03:07
8	1	2	Gabriela	Gouth	sgouth0	Y4i5R8Z1j	2022-08-17 15:34:00
9	6	2	Juan	Pentony	jpentony5	4rcZ47Mc	2022-12-12 08:01:02
10	11	2	Juan	Pentony	jpentony53	4rcZ47Mc	2022-12-12 14:15:03
11	15	1	Juan	Romero	jromero15	esJW8q7N4a	2022-01-29 10:10:20
12	8	2	Liam	Hudson	lhudson7	qSWJGXUsll	2022-11-07 11:03:06
13	13	2	Liam Nicole	Hudson	lhudson73	qSWJGXUsll	2022-11-07 11:20:16
14	7	2	Ricardo	Bastone	rbastone6	3L9b2G	2022-10-10 09:02:30
15	12	2	Ricardo Jóse	Bastone...	rbastone63	3L9b2G	2022-10-10 13:11:30

❖ 2º PROCEDIMIENTO ALMACENADO `sp_insert_rubro`:

OBJETIVO: Permite ingresar un nuevo rubro.

DESCRIPCIÓN: Para utilizar el procedimiento almacenado, se debe indicar a través de un primer parámetro el nombre del nuevo Rubro y un segundo parámetro el Estado (Alta (A) o Baja(B)). El resultado que

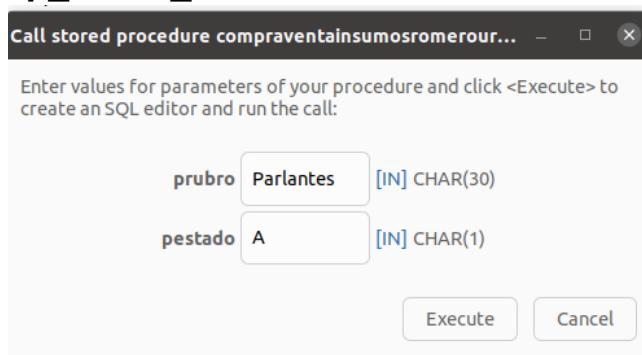
arrojará el procedimiento mostrar un mensaje diciendo que se ingresó correctamente el nombre de Rubro y su respectivo Id.

TABLAS QUE INTERVIENEN EN EL PROCEDIMIENTO: La tabla que interviene en este procedimiento es Roles.

Creación del Script para PROCEDIMIENTO ALMACENADO sp_insert_rubro:

```
1  /*2° Stored Procedure: Permite ingresar un nuevo rubro*/
2  /*Parámetros a ingresar son: Rubro, Estado(A:Alta, B:Baja)*/
3  DELIMITER $$
4  • CREATE DEFINER='root'@`%` PROCEDURE `sp_insert_rubro`(
5      IN prubro CHAR(30),
6      IN pestado CHAR(1)
7  )
8  BEGIN
9      DECLARE ultimoIDmasuno smallint;
10     SET @clausula = concat('SELECT MAX(idRubro) FROM Rubros INTO @ultimoID');
11     PREPARE runSQL FROM @clausula;
12     EXECUTE runSQL;
13     DEALLOCATE PREPARE runSQL;
14     set ultimoIDmasuno=@ultimoID+1;
15     SET @ftexsql=CONCAT('INSERT INTO Rubros (idRubro,Rubro,Estado) VALUES (','"',"CAST(ultimoIDmasuno AS CHAR),"','"',prubro,'"','"',pestado,'"')');
16     PREPARE fodsSQL FROM @ftexsql;
17     EXECUTE fodsSQL;
18     DEALLOCATE PREPARE fodsSQL;
19     SELECT 'Se ingresó satisfactoriamente el nuevo rubro de Nombre:',prubro,'con el ID: ',ultimoIDmasuno,'';
20
21 END$$
```

Ejecutemos el Script para el PROCEDIMIENTO ALMACENADO sp_insert_rubro:



Finalmente el Resultado que obtenemos es el siguiente:

```
#  Se ingresó satisfactoriamente el nuevo rubro de Noml prubro      con el ID: ultimoidmasuno
1  Se ingresó satisfactoriamente el nuevo rubro de Nombre: Parlantes    con el ID: 17
```

❖ 3° PROCEDIMIENTO ALMACENADO sp_delete_rubro:

OBJETIVO: Permite eliminar un rubro por su id.

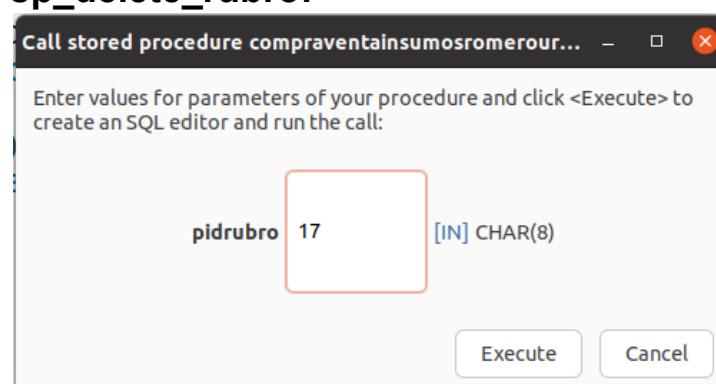
DESCRIPCIÓN: Para utilizar el procedimiento almacenado, se debe indicar a través de un parámetro el ID del Rubro que se desea eliminar. El resultado que arrojará el procedimiento será mostrar un mensaje diciendo que se eliminó correctamente el Rubro.

TABLAS QUE INTERVIENEN EN EL PROCEDIMIENTO: La tabla que interviene en este procedimiento es Roles.

Creación del Script para PROCEDIMIENTO ALMACENADO sp_delete_rubro:

```
1  /*3° Stored Procedure: Permite Eliminar un rubro por su ID*/
2  /*Parámetros a ingresar son: ID del rubro*/
3  DELIMITER $$ 
4  CREATE DEFINER=`root`@`%` PROCEDURE `sp_delete_rubro`(
5      IN pidrubro CHAR(8)
6  )
7  BEGIN
8      SET @ftexsql=CONCAT('DELETE FROM Rubros WHERE idRubro=(',pidrubro,')');
9      PREPARE fordsql FROM @ftexsql;
10     EXECUTE fordsql;
11     DEALLOCATE PREPARE fordsql;
12     SELECT 'Se eliminó satisfactoriamente el rubro de ID:',pidrubro,'';
13 END$$
14
```

Ejecutemos el Script para el PROCEDIMIENTO ALMACENADO sp_delete_rubro:



Finalmente el Resultado que obtenemos es el siguiente:

```
# Se eliminó satisfactoriamente el rubro de ID: pidrubro
1 Se eliminó satisfactoriamente el rubro de ID: 17
```

Script SQL Creación de Triggers:

- **1° TRIGGER:** `BEF_INS_ventasproductos_movimientoVentasProductos`

OBJETIVO: Crear un trigger después de la inserción de una venta de Productos en la Tabla MovimientoVentasProductos guardando datos importantes de la operación.

DESCRIPCIÓN: Se crea primero que nada una tabla llamada MovimientoVentasProductos la cual va a guardar la información necesaria que se genere al disparar el trigger, como se puede observar en la imagen. Una vez realizada la tabla se procede a escribir el código del trigger. El trigger denominado `BEF_INS_ventasproductos_movimientoVentasProductos`, lo que hace es que después de la inserción en la tabla VentasProductos se va a aplicar por cada fila insertar en la tabla MovimientoVentasProductos los datos pasados: idVenta, idProducto, StockAnterior, Precio, Cantidad, StockNuevo, Usuario que hizo la venta, FechaVenta, HoraVenta y Opreación. Se realiza una inserción para probar el funcionamiento de este trigger.

TABLAS QUE INTERVIENEN: La tabla que interviene es VentasProductos y MovimientoVentasProductos.

BENEFICIOS: Esto es beneficioso para por ejemplo cuando un usuario X agrega datos a la tabla VentasProductos, para saber qué datos relevantes se agregaron y la información del usuario así como la fecha y la hora en que se realizaron estos cambios. Tiene también un control que permite avisar a través de un mensaje si superó la cantidad de Producto a Vender de Stock.

Creación del Script para el TRIGGER :

```
7 •  CREATE TRIGGER BEF_INS_ventasproductos_movimientoVentasProductos
8      BEFORE INSERT
9      ON VentasProductos
10     FOR EACH ROW
11     BEGIN
12         /*DECLARACION DE VARIABLES*/
13         DECLARE _cantidadviejo INTEGER;
14         DECLARE _cantidadnueva INTEGER;
15         SELECT Stock INTO _cantidadviejo FROM Productos WHERE idProducto = new.idProducto;
16         /*SET Y CONSULTAS*/
17         SET _cantidadnueva = _cantidadviejo - new.Cantidad;
18         IF (_cantidadnueva<0) then
19             SIGNAL SQLSTATE '45000'
20             SET MESSAGE_TEXT = 'No existe esa cantidad de productos en el stock';
21         ELSE
22             /*MODIFICAR VALOR DE STOCK DEL PRODUCTO*/
23             UPDATE Productos SET Stock=_cantidadviejo - new.Cantidad WHERE idProducto = new.idProducto;
24             /*GUARDAR MOVIMIENTO DE LA VENTA DE PRODUCTO*/
25             INSERT INTO MovimientoVentasProductos(idVenta,idProducto,StockAnterior,Precio,Cantidad,
26             StockNuevo,Usuario,FechaVenta,HoraVenta,Operacion)
27             VALUES (NEW.idVenta,NEW.idProducto,_cantidadviejo,NEW.Precio,NEW.Cantidad,
28             _cantidadnueva,CURRENT_USER,CURRENT_DATE(),CURRENT_TIME(),'NUEVAVENTA');
29         END if;
30     END$$
31
```

EJEMPLO DEL TRIGGER

BEF_INS_ventasproductos_movimientoVentasProductos:

```
1    /* Verificamos 1º Trigger */
2 •  INSERT INTO VentasProductos (idVenta,idProducto,Precio,Cantidad) VALUES ('6', '3', '7500', '25');
3
```

Como Resultado Obtenemos en la Tabla MovimientoVentasProductos:

#	idVenta	idProduct	StockAnterio	Precio	Cantidad	StockNuevo	Usuario	FechaVenta	HoraVenta	Operacion
1	6	3	30	7500.00	25	5	root@%	2022-08-01	12:45:42	NUEVAVENTA
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

➤ **2º TRIGGER:** AFT_INS_Clientes_AuditoriaClientes

OBJETIVO: El trigger permite auditar a un nuevo cliente en la tabla AuditoriaClientes.

DESCRIPCIÓN: Se crea primero que nada una tabla llamada AuditoriaClientes la cual va a guardar la información necesaria que se genere al disparar el trigger, como se puede observar en la imagen. Una

vez realizada la tabla se procede a escribir el código del trigger. El trigger denominado `FT_INS_Clientes_AuditoriaCientes`, lo que hace es que después de la inserción en la tabla Clientes se va a aplicar por cada fila insertar en la tabla AuditoriaCientes los datos pasados: idAuditoriaCliente, NombresNuevos, ApellidosNuevos, EmailNuevo, TelefonoNuevo, UsuarioCreacion, FechaCreacion, HoraCreacion, Operacion:NUEVO, idCliente. Se realiza una inserción para probar el funcionamiento de este trigger.

TABLAS QUE INTERVIENEN: La tabla que interviene es Clientes y AuditoriaCliente.

BENEFICIOS: Esto es beneficioso para por ejemplo cuando un usuario X agrega datos a la tabla Cliente, para saber qué datos relevantes se agregaron y la información del usuario así como la fecha y la hora en que se realizaron estos cambios.

Creación del Script para el TRIGGER :

```

1  /*2º Trigger: Permite auditar a un nuevo cliente en la*/
2  /*tabla AuditoriaCientes*/
3 •  DROP TRIGGER IF EXISTS AFT_INS_Clientes_AuditoriaCientes;
4  DELIMITER $$ 
5 •  CREATE TRIGGER AFT_INS_Clientes_AuditoriaCientes
6      AFTER INSERT
7      ON Clientes
8      FOR EACH ROW
9      BEGIN
10      INSERT INTO AuditoriaCientes(NombresNuevos,ApellidosNuevos,EmailNuevo,
11      TelefonoNuevo,UsuarioCreacion,FechaCreacion,HoraCreacion,Operacion,idCliente)
12      VALUES (NEW.Nombres,NEW.Apellidos,NEW.Email,NEW.Telefono,CURRENT_USER,
13      CURRENT_DATE(),CURRENT_TIME(),'NUEVO',NEW.idCliente);
14  END$$

```

EJEMPLO DEL TRIGGER `AFT_INS_Clientes_AuditoriaCientes`:

```

1  /* Verificamos 2º Trigger */
2 •  INSERT INTO Clientes (idCliente,Nombres,Apellidos,Email,Telefono,Estado) VALUES ('16','adrian','rodriguez','arodriguez@gmail.com','323-222-2222','A');
3 •  INSERT INTO Clientes (idCliente,Nombres,Apellidos,Email,Telefono,Estado) VALUES ('17','jose','perea','jperea@gmail.com','5435-353-345','A');
4

```

Como Resultado Obtenemos en la Tabla MovimientoVentasProductos:

Nombre: ApellidosNuevo	EmailNuevo	TelefonoNuevo	UsuarioCreacion	FechaCreacion	HoraCreacion	UsuarioModificacio	FechaModificacio	HoraModificacio	Operacion	idCliente
rodriguez	arodriguez@gmail.com	323-222-2222	root@%	2022-08-01	12:45:42	NULL	NULL	NULL	NUEVO	16
perea	jperea@gmail.com	5435-353-345	root@%	2022-08-01	12:45:42	NULL	NULL	NULL	NUEVO	17

➤ **3° TRIGGER:** BEF_UPD_Clientes_AuditoriaClientes

OBJETIVO: El trigger permite auditar la modificación de un cliente en la tabla AuditoriaClientes.

DESCRIPCIÓN: Se crea primero que nada una tabla llamada AuditoriaClientes la cual va a guardar la información necesaria que se genere al disparar el trigger, como se puede observar en la imagen. Una vez realizada la tabla se procede a escribir el código del trigger. El trigger denominado BEF_UPD_Clientes_AuditoriaClientes, lo que hace es que después de la modificación en la tabla Clientes se va a aplicar por cada fila insertar en la tabla AuditoriaClientes los datos pasados: idAuditoriaCliente, NombresAnteriores, ApellidosAnteriores, EmailAnterior, TelefonoAnterior, ApellidosNuevos, EmailNuevo, TelefonoNuevo, UsuarioModificacion, FechaModificacion, HoraModificacion, Operacion:EDITAR, idCliente. Se realiza una inserción para probar el funcionamiento de este trigger.

TABLAS QUE INTERVIENEN: La tabla que interviene es Clientes y AuditoriaCliente.

BENEFICIOS: Esto es beneficioso para por ejemplo cuando un usuario X modifica datos a la tabla Cliente, para saber qué datos relevantes se modificaron y la información del usuario así como la fecha y la hora en que se realizaron estos cambios.

Creación del Script para el TRIGGER :

```
3 •  DROP TRIGGER IF EXISTS BEF_UPD_Clientes_AuditoriaClientes;
4   DELIMITER $$ 
5 •  CREATE TRIGGER BEF_UPD_Clientes_AuditoriaClientes
6    BEFORE UPDATE
7    ON Clientes
8    FOR EACH ROW
9    BEGIN
10   INSERT INTO AuditoriaClientes(NombresAnteriores,ApellidosAnteriores,
11   TelefonoAnterior,EmailAnterior,
12   NombresNuevos,ApellidosNuevos,EmailNuevo,
13   TelefonoNuevo,UsuarioModificacion,FechaModificacion,HoraModificacion,
14   Operacion,idCliente)
15   VALUES (OLD.Nombres,OLD.Apellidos,NEW.Telefono,NEW.Email,
16   NEW.Nombres,NEW.Apellidos,NEW.Email,NEW.Telefono,CURRENT_USER,
17   CURRENT_DATE(),current_time(),'EDITAR',NEW.idCliente);
18 END$$
```

EJEMPLO DEL TRIGGER BEF_UPD_Clientes_AuditoriaClientes:

```
1  /* Verificamos 2º Trigger */
2 •  INSERT INTO Clientes (idCliente,Nombres,Apellidos,Email,Telefono,Estado) VALUES ('16','adrian','rodriguez','arodriguez@gmail.com','323-222-2222','A');
3 •  INSERT INTO Clientes (idCliente,Nombres,Apellidos,Email,Telefono,Estado) VALUES ('17','jose','perea','jperea@gmail.com','5435-353-345','A');
4
```

Como Resultado Obtenemos en la Tabla MovimientoVentasProductos:

	idAuditoriaClient	NombresAnteriores	ApellidosAnteriores	EmailAnterior	TelefonoAnterior	NombresNuevo					
	3	adrian	rodriguez	arodriguez@gmail.com	323-222-2222	adrian					
	4	jose	perea	jperea@gmail.com	5435-353-345	jose luis					
	NULL	NULL	NULL	NULL	NULL	NULL	NULL				
	ApellidosNuevo	EmailNuevo	TelefonoNuevo	UsuarioCreacion	FechaCreacion	HoraCreacion	UsuarioModificacion	FechaModificacion	HoraModificacion	Operacion	idCliente
	rodriguez juarez	arodriguez@gmail.com	323-222-2222	NULL	NULL	NULL	root@%	2022-08-01	12:45:42	EDITAR	16
	perea	jperea@gmail.com	5435-353-345	NULL	NULL	NULL	root@%	2022-08-01	12:45:42	EDITAR	17
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

➤ 4º TRIGGER: AFT_DEL_Clientes_AuditoriaClientes

OBJETIVO: El trigger permite auditar el borrado de un cliente en la tabla AuditoriaClientes.

DESCRIPCIÓN: Se crea primero que nada una tabla llamada AuditoriaClientes la cual va a guardar la información necesaria que se genere al disparar el trigger, como se puede observar en la imagen. Una vez realizada la tabla se procede a escribir el código del trigger. El trigger denominado AFT_DEL_Clientes_AuditoriaClientes, lo que hace es que después de borrar un registro por ID en la tabla Clientes se va a aplicar en la tabla AuditoriaClientes los datos pasados: idAuditoriaCliente, NombresAnteriores, Apellidos Anteriores, Operacion:BORRAR, idCliente. Se realiza una inserción para probar el funcionamiento de este trigger.

TABLAS QUE INTERVIENEN: La tabla que interviene es Clientes y AuditoriaCliente.

BENEFICIOS: Esto es beneficioso para por ejemplo cuando un usuario X borrar un registro por ID a la tabla Cliente, para saber qué la información del usuario así como la fecha y la hora en que se realizaron estos cambios.

Creación del Script para el TRIGGER :

```
1  /*4º Trigger: Permite auditar el borrado de un cliente en la*/
2  /*tabla AuditoriaClientes*/
3 • DROP TRIGGER IF EXISTS AFT_DEL_Clientes_AuditoriaClientes;
4  DELIMITER $$
5 • CREATE TRIGGER AFT_DEL_Clientes_AuditoriaClientes
6      AFTER DELETE
7      ON Clientes
8      FOR EACH ROW
9      BEGIN
10         INSERT INTO AuditoriaClientes(NombresAnteriores,ApellidosAnteriores,
11         Operacion,idCliente)
12         VALUES (OLD.Nombres,OLD.Apellidos,
13         'BORRAR',OLD.idCliente);
14     END$$
```

EJEMPLO DEL TRIGGER BEF_UPD_Clientes_AuditoriaClientes:

```
1 • DELETE FROM Clientes WHERE (idCliente = '16');
2
```

Como Resultado Obtenemos en la Tabla MovimientoVentasProductos:

#	idAuditoriaClient	NombresAnterior	ApellidosAnterior	EmailAnterior	TelefonoAnterior	NombresNuevo	ApellidosNuevo	EmailNuevo	Telefono
1	5	adrian	rodriguez juarez	NULL	NULL	NULL	NULL	NULL	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

FechaCreacion	HoraCreacion	UsuarioModificacio	FechaModificacio	HoraModificacio	Operacion	idCliente
NULL	NULL	NULL	NULL	NULL	BORRAR	16
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Creación de Users:

- **PRIMER PASO:** Creación de Users
usuarioLectura@127.0.0.1 y usuarioCRU@localhost, como bien se muestra se decide hacerlo con dominio de localhost (es decir de forma local). Como por esta vez no se pedía en la entrega, se crearon los usuario con contraseña 'Clave123!'.
- **SEGUNDO PASO:** Verificar que realmente se crearon los Users detallados anteriormente.

Las sentencias utilizadas son las siguientes:

```
1  /*Creación de usuarios nuevos*/
2 • CREATE USER usuarioLectura@127.0.0.1 IDENTIFIED BY 'Clave123!';
3 • CREATE USER usuarioCRU@127.0.0.1 IDENTIFIED BY 'Clave123!';
4
5  /*Comprobamos que los Users fueron creados con éxito*/
6  /*Para verificar que los Users fueron creados usamos la siguiente sentencia*/
7 • SELECT * FROM mysql.user;
```

El Resultado obtenido es el siguiente:

#	Host	User
1	%	root
2	127.0.0.1	usuarioCRU
3	127.0.0.1	usuarioLectura
4	127.0.0.1	usuarioLectura2
5	localhost	debian-sys-maint
6	localhost	mysql.infoschema
7	localhost	mysql.session
8	localhost	mysql.sys
*	NULL	NULL

DAR PERMISOS A LOS USERS CREADOS:

OBJETIVO: Como bien dice el título se trata de crear permisos para cada uno de los usuarios creados anteriormente.

PRIMER PERMISO OTORGADO A

usuarioLectura@localhost: Se otorgan permisos de Lectura para este User de

la siguiente manera: GRANT SELECT ON *.* TO

usuarioLectura@127.0.0.1;. Lo que estamos realizando con esta sentencia de GRANT SELECT, es justamente eso, darle permisos de Lectura al usuario usuarioLectura@localhost.

VERIFICACIÓN: Con la siguiente sentencia SHOW GRANTS FOR usuarioLectura@localhost; verificamos si efectivamente se creó el permiso de Lectura.

En las siguientes imágenes vemos las sentencias utilizadas y el resultado obtenido:

```
1  /*Permiso de sólo lectura*/
2 • GRANT SELECT ON *.* TO usuarioLectura@127.0.0.1;
3
4  /*Mostrar los permisos de los nuevos usuarios para verificar */
5 • SHOW GRANTS FOR usuarioLectura@127.0.0.1;

#
# Grants for usuarioLectura@127.0.0.1
1 GRANT SELECT ON *.* TO `usuarioLectura`@`127.0.0.1`
```

PRIMEROS PERMISOS OTORGADOS a usuarioCRU@localhost: Se otorgan permisos de Lectura, Inserción y Modificación para este User de la siguiente manera: GRANT SELECT, INSERT, UPDATE ON *.* TO usuarioCRU@127.0.0.1;. Con esta sentencia estamos otorgando al User usuarioCRU@localhost esos permisos anteriormente mencionados al principio.

VERIFICACIÓN: Con la siguiente sentencia SHOW GRANTS FOR usuarioCRU@localhost; verificamos si efectivamente se crearon los permisos de Lectura, Inserción y Modificación.

La siguiente imagen muestra las sentencias mencionadas y los resultados obtenidos:

```
1  /*Permiso de insert, update*/
2 • GRANT SELECT, INSERT, UPDATE ON *.* TO usuarioCRU@127.0.0.1;
3
4  /*Mostrar los permisos de los nuevos usuarios para verificar */
5 • SHOW GRANTS FOR usuarioCRU@127.0.0.1;

#
# Grants for usuarioCRU@127.0.0.1
1 GRANT SELECT, INSERT, UPDATE ON *.* TO `usuarioCRU`@`127.0.0.1`
```

SENTENCIAS DE REVOKE PARA AMBOS USERS: Sólo a modo de prueba se crean dos sentencias de REVOKE, para ambos Users. Debido a que no tienen derechos de eliminación no se van a ver reflejados en ningún lado. Pero como método de prueba parecía interesante hacerlo.

La siguiente imagen muestra las sentencias utilizadas para este caso de REVOKE:

```
1  /*Revocar permisos de eliminar registros de tablas*/
2 •  REVOKE DELETE ON *.* FROM usuarioLectura@127.0.0.1;
3 •  REVOKE DELETE ON *.* FROM usuarioCRU@127.0.0.1;
4
```

Sentencias Del Sublenguaje TCL:

➤ **OBJETIVO:** Elegir dos tablas de la BBDD. Realizar una serie de modificaciones en los registros, controladas por transacciones.

PRIMERA PARTE: En la Primera Tabla, la cual es , como la misma tiene registros, se deben eliminar algunos de ellos iniciando previamente una transacción. Se debe dejar en una línea siguiente, comentado la sentencia Rollback, y en una línea posterior, la sentencia Commit.

- **PRIMER PASO:** Para comenzar cualquier transacción debemos escribir la siguiente sentencia: START TRANSACTION;
- **SEGUNDO PASO:** Escribimos la sentencia SET FOREIGN_KEY_CHECKS=0;, para no tener problemas a la hora de realizar la eliminación con las FK.
- **TERCER PASO:** Realizar la eliminación de un registro que ya tenemos en nuestra BBDD, en nuestra tabla VentasProductos. Para ello usamos la siguiente sentencia:

```
DELETE FROM VentasProductos WHERE idVenta = 6 AND idProducto=9;
```

- **CUARTO PASO:** Verificar que esos registros recién eliminados, efectivamente se hayan borrado con la siguiente sentencia:

```
SELECT * FROM VentasProductos;
```

- **ÚLTIMOS PASOS:** Para terminar debemos dejar comentadas las sentencias ROLLBACK; y COMMIT;.

En las siguiente imagen vemos lo realizado y explicado en los pasos anteriores:

```
1 SET AUTOCOMMIT = 0;
2 -- SELECT @@AUTOCOMMIT;
3
4 /* Tabla VentasProductos */
5 • START TRANSACTION;
6 •     DELETE FROM VentasProductos WHERE idVenta = 6 AND idProducto=9;
7 •     ROLLBACK;
8 •     COMMIT;
9
```

La siguiente imagen muestra el antes de la tabla VentasProductos:

Podemos observar como existen 16 registros en la misma.

#	idVenta	idProducto	Precio	Cantidad
1	2		15000.00	1
2	6		6000.00	2
2	9		160000.00	1
3	3		6000.00	3
4	1		8000.00	2
4	5		10000.00	1
4	8		2000.00	4
4	10		6000.00	3
5	4		7000.00	2
5	5		10000.00	1
6	1		8000.00	2
6	3		7500.00	25
6	5		10000.00	1
6	7		40000.00	1
6	8		2000.00	4
6	9		160000.00	1
	NULL	NULL	NULL	NULL

**La siguiente imagen muestra el después de la tabla VentasProductos al realizar los pasos anteriores:
Y ahora observamos que solamente quedan 15 registros.**

#	idVenta	idProducto	Precio	Cantidad
1	1	2	15000.00	1
2	2	6	6000.00	2
3	2	9	160000.00	1
4	3	3	6000.00	3
5	4	1	8000.00	2
6	4	5	10000.00	1
7	4	8	2000.00	4
8	4	10	6000.00	3
9	5	4	7000.00	2
10	5	5	10000.00	1
11	6	1	8000.00	2
12	6	3	7500.00	25
13	6	5	10000.00	1
14	6	7	40000.00	1
15	6	8	2000.00	4
	NULL	NULL	NULL	NULL

SEGUNDA PARTE: En la segunda tabla, la cual es ComprasProductos, insertamos 8 nuevos registros iniciando también una transacción. Agregaremos un savepoint a posteriori de la inserción del registro #4 y otro savepoint a posteriori del registro #8. También debemos agregar en una línea comentada la sentencia de eliminación del savepoint de los primeros 4 registros insertados

- **PRIMER PASO:** Para comenzar cualquier transacción debemos escribir la siguiente sentencia: START TRANSACTION;
- **SEGUNDO PASO:** Realizamos la inserción de los primeros 4 registros, debido a que hay que colocar un SAVEPOINT a posteriori de la inserción del registro #4.
- **TERCER PASO:** Colocar la sentencia de SAVEPOINT + nombre;, en nuestro caso es SAVEPOINT Lote 1;. El nombre se debe a lo explicado en el SEGUNDO PASO.
- **CUARTO PASO:** Realizamos la inserción de los últimos 4 registros.
- **QUINTO PASO:** Colocamos la sentencia del SAVEPOINT: SAVEPOINT Lote 2; , el nombre se debe a que hay que agregar un SAVEPOINT a posteriori del registro #8.
- **SEXTO PASO:** Para verificar que los 8 registros efectivamente se agregaron realizamos:

```
SELECT * FROM ComprasProductos;
```
- **ÚLTIMOS PASOS:** Escribimos como comentario la sentencia para eliminar el SAVEPOINT de los primeros 4 registros insertados: RELEASE SAVEPOINT Lote 1;.

En las siguiente imagen vemos lo realizado y explicado en los pasos anteriores:

```
1 • USE compraventainsumosromeroouro;
2
3 • SET AUTOCOMMIT = 0;
4 -- SELECT @@AUTOCOMMIT;
5
6 /* Tabla ComprasProductos */
7 • START TRANSACTION;
8 •     INSERT INTO ComprasProductos (idCompra,idProducto,Precio,Cantidad) VALUES (5,2,2500,3);
9 •     INSERT INTO ComprasProductos (idCompra,idProducto,Precio,Cantidad) VALUES (5,3,2600,2);
10 •    INSERT INTO ComprasProductos (idCompra,idProducto,Precio,Cantidad) VALUES (5,4,2700,1);
11 •    INSERT INTO ComprasProductos (idCompra,idProducto,Precio,Cantidad) VALUES (5,5,3000,4);
12 •    SAVEPOINT LOTE_1;
13 •    INSERT INTO ComprasProductos (idCompra,idProducto,Precio,Cantidad) VALUES (5,6,1500,5);
14 •    INSERT INTO ComprasProductos (idCompra,idProducto,Precio,Cantidad) VALUES (5,7,3500,6);
15 •    INSERT INTO ComprasProductos (idCompra,idProducto,Precio,Cantidad) VALUES (5,8,5500,7);
16 •    INSERT INTO ComprasProductos (idCompra,idProducto,Precio,Cantidad) VALUES (5,9,3500,8);
17 •    SAVEPOINT LOTE_2;
18 •    RELEASE SAVEPOINT LOTE_1;
19 -- ROLLBACK TO LOTE_1; /*Para Verificar que ya no existe el SAVEPOINT LOTE_1*/
20
```

La siguiente imagen muestra el antes de la tabla ComprasProductos:

Podemos observar como existen 10 registros en la misma.

	idCompra	idProducto	Precio	Cantidad
1	1		4800.00	2
1	3		1800.00	3
1	5		6000.00	1
1	6		3600.00	4
2	1		4800.00	4
3	1		4800.00	3
3	4		4200.00	2
3	6		3600.00	5
3	10		3600.00	7
4	1		4800.00	4
4	10		3600.00	6
4	11		120000.00	2
5	1		4800.00	3
5	10		3600.00	2
5	15		2400.30	4
	NULL	NULL	NULL	NULL

La siguiente imagen muestra el después de la tabla Compras al realizar los pasos anteriores:

Y ahora observamos que tenemos 23 registros.

<u>idCompra</u>	<u>idProducto</u>	<u>Precio</u>	<u>Cantidad</u>
3	4	4200.00	2
3	6	3600.00	5
3	10	3600.00	7
4	1	4800.00	4
4	10	3600.00	6
4	11	120000.00	2
5	1	4800.00	3
5	2	2500.00	3
5	3	2600.00	2
5	4	2700.00	1
5	5	3000.00	4
5	6	1500.00	5
5	7	3500.00	6
5	8	5500.00	7
5	9	3500.00	8
5	10	3600.00	2
5	15	2400.30	4
NULL	NULL	NULL	NULL

08 SCRIPTS DE INSERCIÓN DE DATOS

Backup y Restauración:

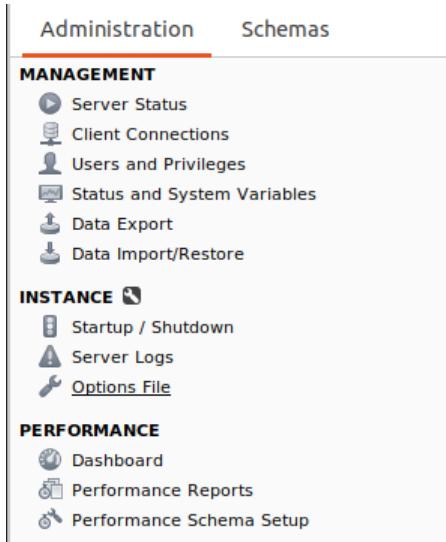
- **OBJETIVO:** Generar un Backup de la BBDD, incluyendo en éste solamente las tablas. El backup debe incluir sólo los datos, dejando de lado su estructura. No se incluye debido a que ya tenemos la BBDD creada cuando se hace el Backup. También realizar la Importación o Restauración de esos Datos salvados.

Backup:

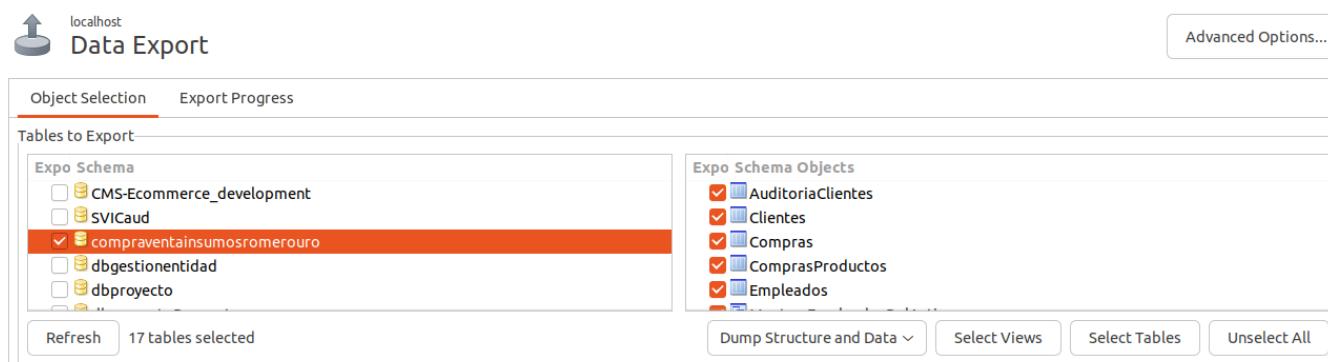
- **PRIMER PASO:** Para realizar el Backup de nuestra BBDD, lo que hacemos primero que nada es ir a la pestaña de Administración (en la parte izquierda al lado de Schemas).

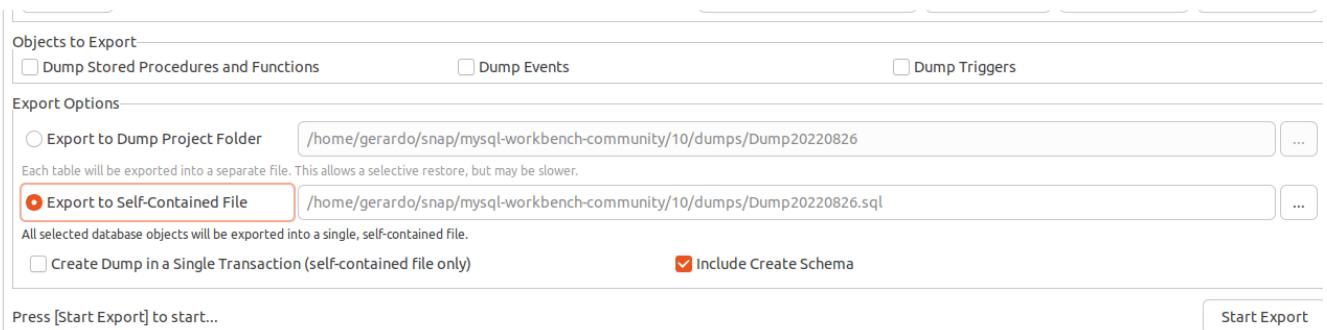
- **SEGUNDO PASO:** Nos dirigimos a la parte en la que dice MANAGEMENT, ahí le damos click en donde dice “Data Export”.

Como se muestra en la siguiente imagen:



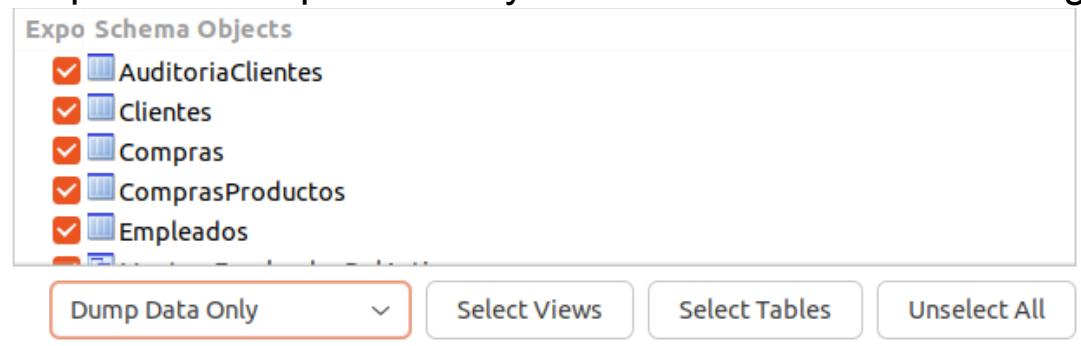
TERCER PASO: Se nos abrirá la siguiente ventana, conteniendo todos los Schemas que tenemos en ese momento, y una serie de opciones para Exportar los Datos, donde se marca nuestra BBDD a exportar e “Include Schema Create”.





- **CUARTO PASO:** En esta parte elegimos el Schema que queremos Exportar o al cual queremos hacerle el Backup. Así que simplemente la seleccionamos.

- **CUARTO PASO:** Ahora es momento de elegir si queremos Exportar los Datos con la Estructura, la Estructura sola o los Datos solos. En nuestro caso se nos había pedido SOLAMENTE los Datos de las Tablas. Por lo que Elegimos la Opción: "Dump Data Only" como se muestra en la imagen:



- **ÚLTIMOS PASOS:** A esta altura ya estamos listos para darle al Botón Start Export. Una vez que hagamos esto. En la pestaña "Export Progress" vamos a ver el Progreso de la Exportación, así como el Status que muestra la cantidad de archivos exportados. Así como la hora de realización de la Exportación, las Tablas elegidas, entre otra información, como se muestra en la imagen:

The screenshot shows the MySQL Workbench Data Export interface. At the top, there's a header with a logo, the text "localhost", and "Data Export". Below the header, there are two tabs: "Object Selection" and "Export Progress", with "Export Progress" being the active tab. Under "Export Progress", it says "Export Completed" and "Status: 17 of 17 exported.". A "Log" section contains the following text:

```
18:50:16 Dumping compraventainsumosromerouro (all tables)
Running: /snap/mysql-workbench-community/10/usr/lib/mysql-workbench/mysqldump --defaults-file="/tmp/tmp34n6sgm/extraparams.cnf" --host=127.0.0.1 --port=3306 --default-character-set=utf8 --user=root --protocol=tcp --column-statistics=FALSE --no-create-info=TRUE --skip-triggers "compraventainsumosromerouro"
18:50:16 Export of /home/gerardo/snap/mysql-workbench-community/10/dumps/Dump20220826.sql has finished
```

Restauración:

● **PRIMER PASO:** Una vez realizado el Backup tenemos un archivo .sql con las Tablas que deseábamos Recuperar. Como se ve en la imagen, tenemos un solo archivo .sql con las Tablas a Recuperar:

Nombre	Tamaño	Modificación
compraventainsumosBKP.sql	17,8 kB	1 ago

Lo que podemos hacer en un principio es abrirlo en un Editor de Texto para ver lo que contiene, que en nuestro caso es lo siguiente:

```
-- MySQL dump 10.13 Distrib 8.0.29, for Linux (x86_64)
--
-- Host: 127.0.0.1 Database: compraventainsumosromerouro
-- -----
-- Server version      8.0.19-0ubuntu5

/*!40101 SET
@OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET
@OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET
@OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40103 SET NAMES utf8 */;
```

```

/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET
@OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS,
UNIQUE_CHECKS=0 */;
/*!40014 SET
@OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECK
S, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES,
SQL_NOTES=0 */;

-- CREATE DATABASE
CREATE DATABASE IF NOT EXISTS
`compraventainsumosromerouro`;
USE `compraventainsumosromerouro`;

-- 
-- Tabla `AuditoriaClientes`
-- 

DROP TABLE IF EXISTS `AuditoriaClientes`;
/*!40101 SET @saved_cs_client    = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `AuditoriaClientes` (
  `idAuditoriaCliente` int NOT NULL AUTO_INCREMENT,
  `NombresAnteriores` varchar(40) DEFAULT NULL,
  `ApellidosAnteriores` varchar(40) DEFAULT NULL,
  `EmailAnterior` varchar(50) DEFAULT NULL,
  `TelefonoAnterior` varchar(15) DEFAULT NULL,
  `NombresNuevos` varchar(40) DEFAULT NULL,
  `ApellidosNuevos` varchar(40) DEFAULT NULL,
  `EmailNuevo` varchar(50) DEFAULT NULL,
  `TelefonoNuevo` varchar(15) DEFAULT NULL,
  `UsuarioCreacion` varchar(40) DEFAULT NULL,
  `FechaCreacion` date DEFAULT NULL,
  `HoraCreacion` time DEFAULT NULL,
  `UsuarioModificacion` varchar(40) DEFAULT NULL,

```

```
`FechaModificacion` date DEFAULT NULL,  
 `HoraModificacion` time DEFAULT NULL,  
 `Operacion` varchar(10) NOT NULL,  
 `idCliente` int NOT NULL,  
 PRIMARY KEY (`idAuditoriaCliente`)  
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT  
 CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;  
/*!40101 SET character_set_client = @saved_cs_client */;
```

```
--  
-- Dumping data for table `AuditoriaClientes`  
--
```

```
LOCK TABLES `AuditoriaClientes` WRITE;  
/*!40000 ALTER TABLE `AuditoriaClientes` DISABLE KEYS */;  
INSERT INTO `AuditoriaClientes` VALUES  
(1,NULL,NULL,NULL,NULL,'adrian','rodriguez','arodriguez@gmail.  
com','323-222-2222','root@%', '2022-08-01','12:45:42',NULL,NULL,  
NULL,'NUEVO',16),(2,NULL,NULL,NULL,NULL,'jose','perea','jpere  
a@gmail.com','5435-353-345','root@%', '2022-08-01','12:45:42',NU  
LL,NULL,NULL,'NUEVO',17),(3,'adrian','rodriguez','arodriguez@g  
mail.com','323-222-2222','adrian','rodriguez  
juarez','arodriguez@gmail.com','323-222-2222',NULL,NULL,NULL,  
'root@%', '2022-08-01','12:45:42','EDITAR',16),(4,'jose','perea','jper  
ea@gmail.com','5435-353-345','jós  
e luis','perea','jperea@gmail.com','5435-353-345',NULL,NULL,NULL,  
'root@%', '2022-08-01','12:45:42','EDITAR',17),(5,'adrian','rodriguez  
juarez',NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,N  
ULL,NULL,NULL,'BORRAR',16);  
/*!40000 ALTER TABLE `AuditoriaClientes` ENABLE KEYS */;  
UNLOCK TABLES;
```

```
--  
-- Tabla `Clientes`  
--
```

```
DROP TABLE IF EXISTS `Clientes`;  
/*!40101 SET @saved_cs_client    = @@character_set_client */;  
/*!50503 SET character_set_client = utf8mb4 */;  
CREATE TABLE `Clientes` (
```

```
`idCliente` int NOT NULL,  
`Nombres` varchar(40) NOT NULL,  
`Apellidos` varchar(40) NOT NULL,  
`Email` varchar(50) NOT NULL,  
`Telefono` varchar(15) NOT NULL,  
`Estado` char(1) NOT NULL,  
PRIMARY KEY (`idCliente`),  
UNIQUE KEY `UI_Email` (`Email`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4  
COLLATE=utf8mb4_0900_ai_ci;  
/*!40101 SET character_set_client = @saved_cs_client */;
```

```
--  
-- Dumping data for table `Clientes`  
--
```

```
LOCK TABLES `Clientes` WRITE;  
/*!40000 ALTER TABLE `Clientes` DISABLE KEYS */;  
INSERT INTO `Clientes` VALUES  
(1,'Cort','Cogle','ccogle0@last.fm','269-433-3884','A'),(2,'Petronilla','Foro','pforo1@chicagotribune.com','265-763-9016','A'),(3,'Adeline','Beattie','abeattie2@alexa.com','513-698-3743','A'),(4,'Ruggiero','Rosenfeld','rrosenfeld3@jiathis.com','226-126-2672','B'),(5,'Bernetta','Crews','bcrews4@wikimedia.org','807-436-6134','B'),(6,'Flory','MacMaster','fmacmaster5@wp.com','701-788-0167','A'),(7,'Trey','Patkin','tpatkin6@tamu.edu','360-844-7252','A'),(8,'Dimitry','Markel','dmarkel7@google.co.uk','140-838-7270','A'),(9,'Roxanna','Mara','rmara8@cornell.edu','764-148-6397','A'),(10,'Ailina','Gamage','agamage9@ocn.ne.jp','390-327-2429','A'),(11,'Cully','Cocking','ccockinga@va.gov','748-271-1565','A'),(12,'Correy','Wortt','cworttb@gravatar.com','287-364-0543','A'),(13,'Nikolaus','Tutin','ntutinc@newyorker.com','941-689-1061','A'),(14,'Renate','Tetlow','rtetlowd@slate.com','666-317-3679','A'),(15,'Livvy','Stoeckle','lstoecklee@sakura.ne.jp','300-994-0533','A'),(17,'José Luis','Perea','jperea@gmail.com','5435-353-345','A');  
/*!40000 ALTER TABLE `Clientes` ENABLE KEYS */;  
UNLOCK TABLES;
```

```
--  
-- Table structure for table `Compras`
```

```
--  
  
DROP TABLE IF EXISTS `Compras`;  
/*!40101 SET @saved_cs_client      = @@character_set_client */;  
/*!50503 SET character_set_client = utf8mb4 */;  
CREATE TABLE `Compras` (  
  `idCompra` bigint NOT NULL,  
  `idProveedor` int NOT NULL,  
  `idEmpleado` int NOT NULL,  
  `FechaCompra` datetime NOT NULL,  
  PRIMARY KEY (`idCompra`),  
  KEY `RefidEmpleado` (`idEmpleado`),  
  KEY `RefidProveedor` (`idProveedor`),  
  CONSTRAINT `Comp_empl` FOREIGN KEY (`idEmpleado`)  
    REFERENCES `Empleados` (`idEmpleado`),  
  CONSTRAINT `Comp_prov` FOREIGN KEY (`idProveedor`)  
    REFERENCES `Proveedores` (`idProveedor`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4  
COLLATE=utf8mb4_0900_ai_ci;  
/*!40101 SET character_set_client = @saved_cs_client */;
```

```
--  
-- Dumping data for table `Compras`  
--
```

```
LOCK TABLES `Compras` WRITE;  
/*!40000 ALTER TABLE `Compras` DISABLE KEYS */;  
INSERT INTO `Compras` VALUES (1,2,3,'2022-06-28  
15:11:02'),(2,1,4,'2022-06-28 15:11:33'),(3,5,4,'2022-06-28  
15:11:58'),(4,6,2,'2022-06-28 15:12:58'),(5,7,3,'2022-06-28  
15:15:58');  
/*!40000 ALTER TABLE `Compras` ENABLE KEYS */;  
UNLOCK TABLES;
```

```
--  
-- Tabla `ComprasProductos`  
--
```

```
DROP TABLE IF EXISTS `ComprasProductos`;  
/*!40101 SET @saved_cs_client      = @@character_set_client */;
```

```
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `ComprasProductos` (
  `idCompra` bigint NOT NULL,
  `idProducto` int NOT NULL,
  `Precio` decimal(10,2) NOT NULL,
  `Cantidad` smallint NOT NULL,
  PRIMARY KEY (`idCompra`,`idProducto`),
  KEY `RefComprasProductos` (`idProducto`),
  KEY `RefidCompras` (`idCompra`),
  CONSTRAINT `Cp_comp` FOREIGN KEY (`idCompra`)
    REFERENCES `Compras` (`idCompra`),
  CONSTRAINT `Cp_prod` FOREIGN KEY (`idProducto`)
    REFERENCES `Productos` (`idProducto`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
```

```
--  
-- Dumping data for table `ComprasProductos`
```

```
--  
LOCK TABLES `ComprasProductos` WRITE;
/*!40000 ALTER TABLE `ComprasProductos` DISABLE KEYS */;
INSERT INTO `ComprasProductos` VALUES
(1,1,4800.00,2),(1,3,1800.00,3),(1,5,6000.00,1),(1,6,3600.00,4),(2,
1,4800.00,4),(3,1,4800.00,3),(3,4,4200.00,2),(3,6,3600.00,5),(3,10,
3600.00,7),(4,1,4800.00,4),(4,10,3600.00,6),(4,11,120000.00,2),(5,
1,4800.00,3),(5,10,3600.00,2),(5,15,2400.30,4);
/*!40000 ALTER TABLE `ComprasProductos` ENABLE KEYS */;
UNLOCK TABLES;
```

```
--  
-- Tabla `Empleados`
```

```
--  
DROP TABLE IF EXISTS `Empleados`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `Empleados` (
  `idEmpleado` int NOT NULL,
```

```

`idRol` tinyint NOT NULL,
`Nombres` varchar(40) NOT NULL,
`Apellidos` varchar(40) NOT NULL,
`Usuario` varchar(30) NOT NULL,
`Password` varchar(64) NOT NULL,
`FechalIngreso` datetime NOT NULL,
PRIMARY KEY (`idEmpleado`),
UNIQUE KEY `UniqueIndex_Usuario` (`Usuario`),
KEY `RefEmpleadoidRol` (`idRol`),
CONSTRAINT `Empl_rol` FOREIGN KEY (`idRol`)
REFERENCES `Roles` (`idRol`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
```

-- Dumping data for table `Empleados`

```

LOCK TABLES `Empleados` WRITE;
/*!40000 ALTER TABLE `Empleados` DISABLE KEYS */;
INSERT INTO `Empleados` VALUES
(1,2,'Gabriela','Gouth','sgouth0','Y4i5R8Z1j','2022-08-17
15:34:00'),(2,2,'Carlos','Nurny','cnurny1','hx8gPFx0uz','2022-06-29
15:33:02'),(3,2,'Benjamin','Shafier','bshafier2','1rYHuyPpSdv','2022
-09-06
13:02:04'),(4,2,'Bautista','Semerad','bsemerad3','yiZSrbL6','2022-0
4-03
14:02:10'),(5,2,'Blanca','Yellep','byellep4','BgEnqQ','2022-01-18
10:02:21'),(6,2,'Juan','Pentony','jpentyony5','4rcZ47Mc','2022-12-12
08:01:02'),(7,2,'Ricardo','Bastone','rbastone6','3L9b2G','2022-10-1
0
09:02:30'),(8,2,'Liam','Hudson','lhudson7','qSWJGXUsll','2022-11-0
7
11:03:06'),(9,2,'Ezequiel','Armer','earmer8','vA8K2ZU8db','2022-04-
02
14:03:05'),(10,2,'Florencia','Tunsley','ftunsley9','YsJW8q7N4a','202
2-04-13
13:03:07'),(11,2,'Juan','Pentony','jpentyony53','4rcZ47Mc','2022-12-
12 14:15:03'),(12,2,'Ricardo Jóse','Bastone

```

```
perez','rbastone63','3L9b2G','2022-10-10 13:11:30'),(13,2,'Liam
Nicole','Hudson','lhudson73','qSWJGXUsII','2022-11-07
11:20:16'),(14,2,'Ezequiel
Jose','Armer','earmer83','vA8K2ZU8db','2022-04-02
10:50:05'),(15,1,'Juan','Romero','jromero15','esJW8q7N4a','2022-0
1-29 10:10:20');
/*!40000 ALTER TABLE `Empleados` ENABLE KEYS */;
UNLOCK TABLES;
```

```
--  
-- Tabla `MovimientoVentasProductos`  
--
```

```
DROP TABLE IF EXISTS `MovimientoVentasProductos`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `MovimientoVentasProductos` (
  `idVenta` bigint NOT NULL,
  `idProducto` int NOT NULL,
  `StockAnterior` smallint NOT NULL,
  `Precio` decimal(12,2) NOT NULL,
  `Cantidad` smallint NOT NULL,
  `StockNuevo` smallint NOT NULL,
  `Usuario` varchar(40) DEFAULT NULL,
  `FechaVenta` date DEFAULT NULL,
  `HoraVenta` time DEFAULT NULL,
  `Operacion` varchar(10) NOT NULL,
  PRIMARY KEY (`idVenta`,`idProducto`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
```

```
--  
-- Dumping data for table `MovimientoVentasProductos`  
--
```

```
LOCK TABLES `MovimientoVentasProductos` WRITE;
/*!40000 ALTER TABLE `MovimientoVentasProductos` DISABLE
KEYS */;
```

```

INSERT INTO `MovimientoVentasProductos` VALUES
(6,3,30,7500.00,25,5,'root@%','2022-08-01','12:45:42','NUEVAVEN
TA');
/*!40000 ALTER TABLE `MovimientoVentasProductos` ENABLE
KEYS */;
UNLOCK TABLES;

-- 
-- Tabla structure for table `Productos`
-- 

DROP TABLE IF EXISTS `Productos`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `Productos` (
  `idProducto` int NOT NULL,
  `idRubro` smallint NOT NULL,
  `producto` varchar(100) NOT NULL,
  `PrecioCosto` decimal(12,2) NOT NULL,
  `PrecioVenta` decimal(12,2) NOT NULL,
  `Stock` int NOT NULL,
  `Estado` char(1) NOT NULL,
  PRIMARY KEY (`idProducto`),
  UNIQUE KEY `UniqueIndex_Producto` (`producto`),
  KEY `RefProductosidRubro` (`idRubro`),
  CONSTRAINT `Prod_rubro` FOREIGN KEY (`idRubro`)
    REFERENCES `Rubros` (`idRubro`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

-- 
-- Dumping data for table `Productos`
-- 

LOCK TABLES `Productos` WRITE;
/*!40000 ALTER TABLE `Productos` DISABLE KEYS */;
INSERT INTO `Productos` VALUES (1,4,'Disco WD
1Tb',4800.00,8000.00,50,'A'),(2,1,'Impresora Epson

```

Tx241',9000.00,15000.00,40,'A'),(3,2,'Mouse Genius
Dx300',1800.00,3000.00,5,'A'),(4,2,'Mouse Logitech
G605',4200.00,7000.00,40,'B'),(5,4,'Disco WD Green 480Gb
SSD',6000.00,10000.00,55,'A'),(6,5,'Memoria RAM 2GB HyperX
Fury',3600.00,6000.00,50,'A'),(7,6,'Monitor 22\" Samsung
Dx3',24000.00,40000.00,30,'B'),(8,12,'Pendrive 32 Gb Kingston
d300',1200.00,2000.00,45,'A'),(9,6,'Placa de Video GTX 3060
OC',96000.00,160000.00,10,'A'),(10,3,'Teclado Genius
KB300',3600.00,6000.00,25,'A'),(11,11,'Notebook HP 15\6 Intel I3
256 Gb SSD 16 Gb
RAM',120000.00,200000.00,6,'A'),(12,5,'Memoria RAM 8GB
Kingston ValueRAM
DDR3',3000.00,5000.00,20,'A'),(13,12,'Pendrive 16 Gb Kingston
x40',7200.00,12000.00,25,'A'),(14,6,'Placa de Video GT 1060 4
Gb',30000.18,50000.30,30,'A'),(15,14,'Modem Tp-Link 300Mb
WR-850RE',2400.30,4000.50,40,'A');
/*!40000 ALTER TABLE `Productos` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Tabla `Proveedores`
--

DROP TABLE IF EXISTS `Proveedores`;
/*!40101 SET @saved_cs_client = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `Proveedores` (
 `idProveedor` int NOT NULL,
 `Proveedor` varchar(40) NOT NULL,
 `CUIT` varchar(20) NOT NULL,
 `Estado` char(1) NOT NULL,
 PRIMARY KEY (`idProveedor`),
 UNIQUE KEY `UniqueIndex_Proveedor` (`Proveedor`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `Proveedores`
--

```
LOCK TABLES `Proveedores` WRITE;
/*!40000 ALTER TABLE `Proveedores` DISABLE KEYS */;
INSERT INTO `Proveedores` VALUES (1,'Tucumán Ofertas','20334455663','A'),(2,'Exo Soluciones Informáticas','20151617483','A'),(3,'Ceven','27234242221','A'),(4,'Computronic SRL','20345234241','A'),(5,'Nh Computación','20342532523','A'),(6,'Compuciti','32423423424','B'),(7,'Tecnologic','32423424241','A'),(8,'Amazon','34324324324','A'),(9,'Alum Computación','20343242352','B'),(10,'SYSCOM','20345324324','A'),(11,'Compumaq','22343243242','A'),(12,'Punto Com punto ar','33234234234','A'),(13,'Pronet','23423532522','A'),(14,'Mundo PC','22343242555','B'),(15,'Mercado Libre','22342343245','A');
/*!40000 ALTER TABLE `Proveedores` ENABLE KEYS */;
UNLOCK TABLES;
```

```
--
```

```
-- Tabla `Roles`
```

```
--
```

```
DROP TABLE IF EXISTS `Roles`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `Roles` (
  `idRol` tinyint NOT NULL,
  `Rol` varchar(30) NOT NULL,
  `Estado` char(1) NOT NULL,
  PRIMARY KEY (`idRol`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
```

```
--
```

```
-- Dumping data for table `Roles`
```

```
--
```

```
LOCK TABLES `Roles` WRITE;
/*!40000 ALTER TABLE `Roles` DISABLE KEYS */;
```

```

INSERT INTO `Roles` VALUES
(1,'ADMIN','A'),(2,'EMPLEADO','A'),(3,'TEST','A'),(4,'TEST 2','B');
/*!40000 ALTER TABLE `Roles` ENABLE KEYS */;
UNLOCK TABLES;

-- 
-- Tabla `Rubros`
-- 

DROP TABLE IF EXISTS `Rubros`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `Rubros` (
  `idRubro` smallint NOT NULL,
  `Rubro` varchar(30) NOT NULL,
  `Estado` char(1) NOT NULL,
  PRIMARY KEY (`idRubro`),
  UNIQUE KEY `UniqueIndex_Rubro` (`Rubro`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

-- 
-- Dumping data for table `Rubros`
-- 

LOCK TABLES `Rubros` WRITE;
/*!40000 ALTER TABLE `Rubros` DISABLE KEYS */;
INSERT INTO `Rubros` VALUES
(1,'Impresora','A'),(2,'Mouse','A'),(3,'Teclado','A'),(4,'Discos Rígidos','A'),(5,'Memoria RAM','A'),(6,'Placa de Video','A'),(7,'Monitor','A'),(8,'Cables VGA','A'),(9,'Cables HDMI','A'),(10,'Cargador Notebook','A'),(11,'Notebook','A'),(12,'Pendrive','A'),(13,'Procesador ','A'),(14,'Modem','A'),(15,'Placa de Red','A');
/*!40000 ALTER TABLE `Rubros` ENABLE KEYS */;
UNLOCK TABLES;

-- 
-- Tabla `Ventas`
-- 
```

```
--  
  
DROP TABLE IF EXISTS `Ventas`;  
/*!40101 SET @saved_cs_client      = @@character_set_client */;  
/*!50503 SET character_set_client = utf8mb4 */;  
CREATE TABLE `Ventas` (  
    `idVenta` bigint NOT NULL,  
    `idCliente` int NOT NULL,  
    `idEmpleado` int NOT NULL,  
    `FechaVenta` datetime NOT NULL,  
    PRIMARY KEY (`idVenta`),  
    KEY `RefVentasidEmpleado` (`idEmpleado`),  
    KEY `RefVentasidCliente` (`idCliente`),  
    CONSTRAINT `Venta_Cli` FOREIGN KEY (`idCliente`)  
        REFERENCES `Clientes` (`idCliente`),  
    CONSTRAINT `Venta_Empl` FOREIGN KEY (`idEmpleado`)  
        REFERENCES `Empleados` (`idEmpleado`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4  
COLLATE=utf8mb4_0900_ai_ci;  
/*!40101 SET character_set_client = @saved_cs_client */;
```

```
--  
-- Dumping data for table `Ventas`  
--
```

```
LOCK TABLES `Ventas` WRITE;  
/*!40000 ALTER TABLE `Ventas` DISABLE KEYS */;  
INSERT INTO `Ventas` VALUES (1,1,2,'2022-06-28  
15:01:02'),(2,2,2,'2022-06-28 15:02:00'),(3,3,1,'2022-06-28  
15:02:31'),(4,3,3,'2022-06-28 15:03:24'),(5,4,4,'2022-06-28  
15:04:00'),(6,2,2,'2022-06-28 15:04:59');  
/*!40000 ALTER TABLE `Ventas` ENABLE KEYS */;  
UNLOCK TABLES;
```

```
--  
-- Tabla `VentasProductos`  
--
```

```
DROP TABLE IF EXISTS `VentasProductos`;
```

```

/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `VentasProductos` (
  `idVenta` bigint NOT NULL,
  `idProducto` int NOT NULL,
  `Precio` decimal(12,2) NOT NULL,
  `Cantidad` smallint NOT NULL,
  PRIMARY KEY (`idVenta`,`idProducto`),
  KEY `RefVentasidProductos` (`idProducto`),
  KEY `RefVentas` (`idVenta`),
  CONSTRAINT `Vp_prod` FOREIGN KEY (`idProducto`)
    REFERENCES `Productos` (`idProducto`),
  CONSTRAINT `Vp_venta` FOREIGN KEY (`idVenta`)
    REFERENCES `Ventas` (`idVenta`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
```

```
--  
-- Dumping data for table `VentasProductos`  
--
```

```

LOCK TABLES `VentasProductos` WRITE;
/*!40000 ALTER TABLE `VentasProductos` DISABLE KEYS */;
INSERT INTO `VentasProductos` VALUES
(1,2,15000.00,1),(2,6,6000.00,2),(2,9,160000.00,1),(3,3,6000.00,3),
(4,1,8000.00,2),(4,5,10000.00,1),(4,8,2000.00,4),(4,10,6000.00,3),
(5,4,7000.00,2),(5,5,10000.00,1),(6,1,8000.00,2),(6,3,7500.00,25),
(6,5,10000.00,1),(6,7,40000.00,1),(6,8,2000.00,4),(6,9,160000.00,1);
/*!40000 ALTER TABLE `VentasProductos` ENABLE KEYS */;
UNLOCK TABLES;
```

```
-- Dump completed on 2022-08-01 12:51:12
```

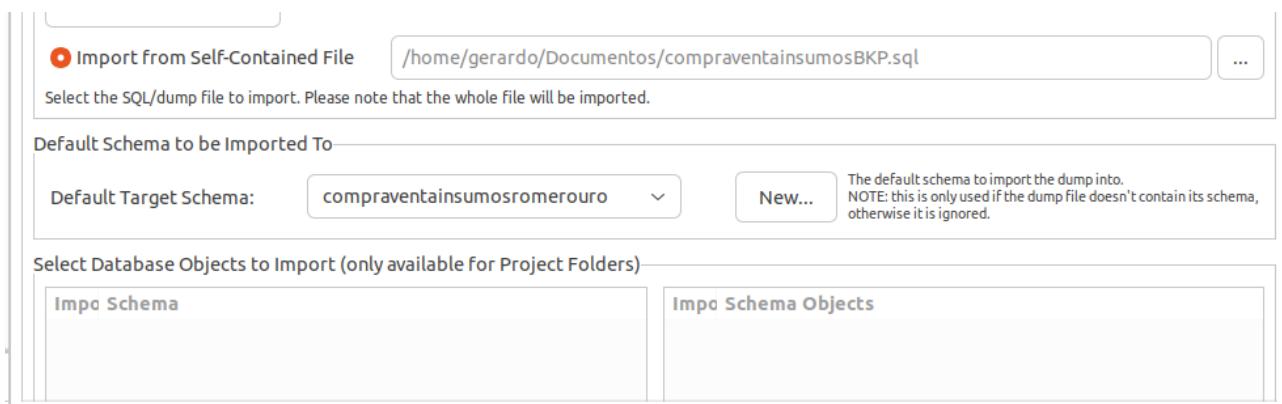
SEGUNDO PASO: Realizaremos la Importación de los Datos.
 Para ello nos dirigimos a la misma Pestaña que para Exportar, sólo que esta vez elegimos “Data Import/Restore” como se muestra en la imagen:

The screenshot shows the MySQL Workbench Administration interface. At the top, there are two tabs: "Administration" (which is selected) and "Schemas". Below the tabs, there are three main sections: "MANAGEMENT" (with icons for Server Status, Client Connections, Users and Privileges, Status and System Variables, Data Export, and Data Import/Restore), "INSTANCE" (with icons for Startup / Shutdown, Server Logs, and Options File), and "PERFORMANCE" (with icons for Dashboard, Performance Reports, and Performance Schema Setup).

TERCER PASO: Se nos abrirá la siguiente ventana, conteniendo la forma de Importación. Hay dos opciones, la primera es para si tenemos archivos por separado de las Tablas importarlas de esa forma. La segunda sería la que vamos a utilizar, ya que es la que permite seleccionar el archivo .sql que tenemos con todas las Tablas. Una vez realizado esto elegimos el archivo, en nuestro caso va a ser el que se muestra en la imagen:

The screenshot shows the "Data Import" dialog box. At the top, it says "localhost" and "Data Import". There are two tabs: "Import from Disk" (selected) and "Import Progress". Under "Import Options", there are two radio button options: "Import from Dump Project Folder" (with a dropdown menu showing "/home/gerardo/snap/mysql-workbench-community/10/dumps") and "Import from Self-Contained File" (with a dropdown menu showing "/home/gerardo/Documentos/compraventainsumosBKP.sql"). Below each option is a note: "Select the Dump Project Folder to import. You can do a selective restore." and "Select the SQL/dump file to import. Please note that the whole file will be imported." There are also "Load Folder Contents" and "... More Options" buttons.

Aquí lo que hacemos es elegir la BBDD (compraventainsumosromerouro) y antes de realizar nada yo borré las Tablas que voy a Importar, y las cree nuevamente pero sin los datos, de esa forma no recibiré ningún error de Datos Duplicados, etc. Y le damos a “Start Import”:



ÚLTIMOS PASOS: En la pestaña “Import Progress” vamos a ver el Progreso de la Importación, así como el Status que muestra la cantidad de archivos exportados. Así como la hora de realización de la Importación, entre otra información, como se muestra en la imagen:

Creating schema compraventainsumosromerouro2

```
19:01:14 Restoring /home/gerardo/Documentos/compraventainsumosBKP.sql
Running: /snap/mysql-workbench-community/10/usr/lib/mysql-workbench/mysql --defaults-file="/tmp/tmp9edsa67x/extraparams.cnf" --
protocol=tcp --host=127.0.0.1 --user=root --port=3306 --default-character-set=utf8 --comments --database=compraventainsumosromerouro2
< "/home/gerardo/Documentos/compraventainsumosBKP.sql"
19:01:16 Import of /home/gerardo/Documentos/compraventainsumosBKP.sql has finished
```

09 INFORMES GENERADOS EN BASE A LA INFORMACIÓN ALMACENADAS EN LAS TABLAS

- Según la Vista **TotalProveedores**, podemos saber el total de proveedores actual que tiene registrado la BBDD.
- De la Vista **NombresApellidosClientesBaja**, la persona que vaya a utilizar la base de datos sepa los Nombres y Apellidos de Clientes dados de baja.
- Mediante la Vista **MostrarEmpleadosRolActivo**, la persona puede ver los Empleado que tienen Rol Activo, o sea, que tiene el rol dado de alta.

- Si observamos la Vista **VentasMesJunio2022deCarlos**, la idea principal de esta vista es que la persona que vaya a utilizar la base de datos sepa el total de las ventas por rango de fecha y por el nombre del empleado como por ejemplo: Ventas del mes de Junio del año 2022 de los empleados llamados Carlos.
- Según la vista, **PrecioTotaldeVentas**, la idea principal de es que la persona que vaya a utilizar la base de datos sepa el total de las ventas realizadas.

10 HERRAMIENTAS Y TECNOLOGÍAS:

- Para el Model ER, el inicial se utilizó la Herramienta app.diagrams.net.
- Para realizar todo el Script SQL y el segundo Modelo ER, se trabajó con el Gestor MySQL.
- Para los Datos, que al final no se utilizaron, para la Importación de Datos, se utilizó Excel Online.
- Para los PDF's, se utilizó Google Presentation.
- Para guardar el repositorio del proyecto se usó Github.
- Zoom: clases brindadas por CoderHouse.
- Chat de Coderhouse.