

# LOBSTER S.R.L

## Proyecto Rubini

Base de datos realizada para el departamento de compras de una industria farmacéutica con el fin de poder visualizar los usuarios que están trabajando en el departamento junto al stock de materia prima que contempla la empresa pasando por el proceso de compra

CODER HOUSE

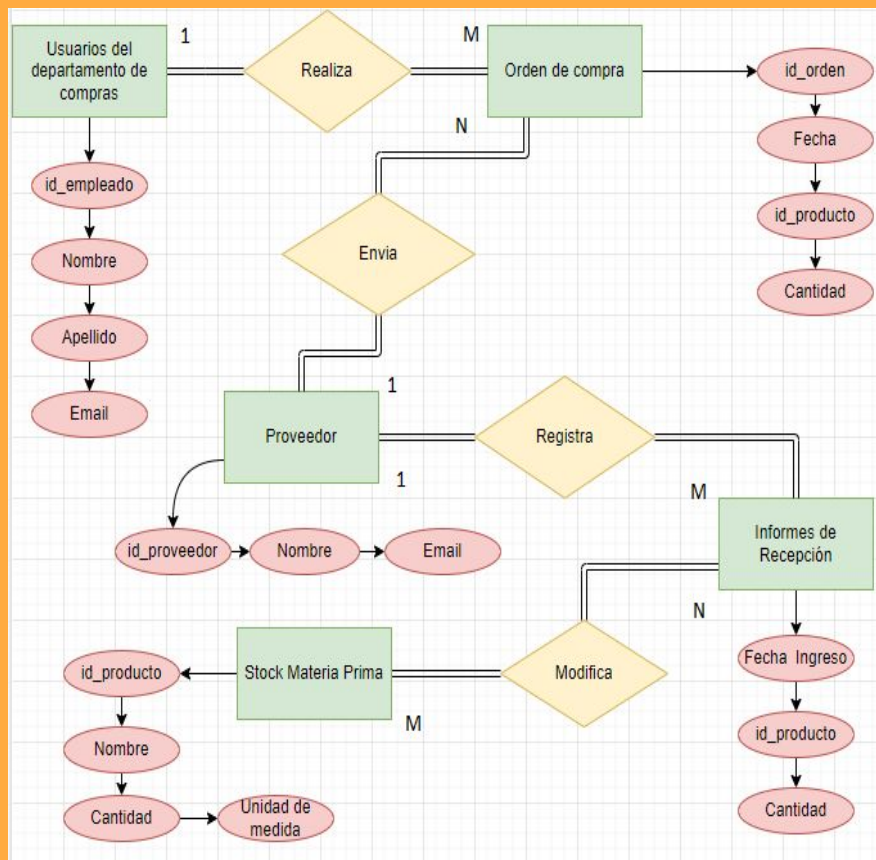


## ***Situación Problemática***

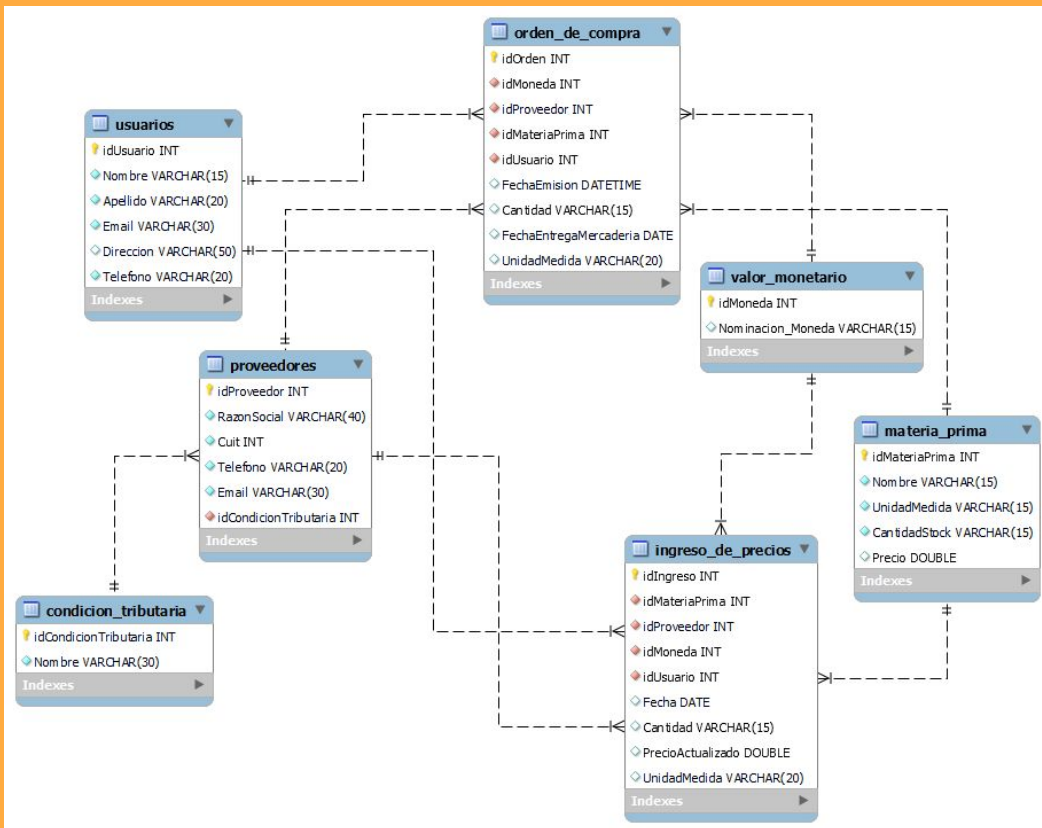
El departamento de compras no contaba con un sistema tan detallado del stock con el cuál contaba y tampoco con un sistema de orden de compra para poder enviar formalmente al proveedor

compras para poder enviar formalmente al proveedor  
del stock con el cual contaba y tampoco con un sistema de orden de

# PRIMER DIAGRAMA



# DIAGRAMA MODIFICADO



## **TABLAS DETALLADAS**

| <b>USUARIOS DEPARTAMENTO DE COMPRAS</b> |       |              |                           |  |
|---|-------|--------------|---------------------------|--|
| Nombre de Campo                         | Clave | Tipo de Dato | Características           | Descripción  |
| ID_Usuario                              | PK    | INT          | NOT_NULL / AUTO_INCREMENT | EN ESTA TABLA SE VAN A DETALLAR TODOS LOS EMPLEADOS QUE SEAN DEL DEPARTAMENTO DE COMPRAS |
| Nombre                                  |       | VARCHAR (15) | NOT_NULL                  |  |
| Apellido                                |       | VARCHAR (20) | NOT_NULL                  |  |
| Email                                   |       | VARCHAR (30) | NOT_NULL                  |  |
| Dirección                               |       | VARCHAR (50) | NOT_NULL                  |  |
| Teléfono                                |       | VARCHAR (20) | NOT_NULL                  |  |
|   |       |              |                           |  |

| <b>VALOR MONETARIO</b> |       |              |                           |   |
|------------------------|-------|--------------|---------------------------|---|
| Nombre de Campo        | Clave | Tipo de Dato | Características           | Descripción   |
| id_moneda              | PK    | INT          | NOT_NULL / AUTO_INCREMENT | EN ESTA TABLA SE VA A DETALLAR EL TIPO DE MONEDA QUE SE UTILIZO EN LA ORDEN DE COMPRA |
| Nominación_Moneda      |       | VARCHAR (15) |                           |   |
|                        |       |              |                           |   |
|                        |       |              |                           |   |
|                        |       |              |                           |   |
|                        |       |              |                           |   |
|                        |       |              |                           |   |

| MATERIA PRIMA         |       |               |                           |  |
|-----------------------|-------|---------------|---------------------------|--|
| Nombre de Campo       | Clave | Tipo de Dato  | Características           | Descripción  |
| idMateriaPrima        | PK    | INT           | NOT_NULL / AUTO_INCREMENT | EN ESTA TABLA SE VA A DETALLAR EL STOCK DE LA MATERIA PRIMA                          |
| Nombre                |       | VARCHAR (100) |                           |  |
| Unidadmedida          |       | VARCHAR (15)  | NOT_NULL                  |  |
| CantidadStock         |       | VARCHAR (15)  | NOT_NULL                  |  |
| Precio                |       | VARCHAR (50)  | NOT_NULL                  |  |
|                       |       |               |                           |  |
|                       |       |               |                           |  |
|                       |       |               |                           |  |
| CONDICIÓN TRIBUTARIA  |       |               |                           |  |
| Nombre de Campo       | Clave | Tipo de Dato  | Características           | Descripción  |
| idCondiciónTributaria | PK    | INT           | NOT_NULL / AUTO_INCREMENT | EN ESTA TABLA SE VAN A DETALLAR LOS PORCENTAJES IMPOSITIVOS QUE TENGA CADA PROVEEDOR |
| Categoria             |       | VARCHAR (30)  | NOT_NULL                  |  |
|                       |       |               |                           |  |
|                       |       |               |                           |  |
|                       |       |               |                           |  |
|                       |       |               |                           |  |
|                       |       |               |                           |  |
|                       |       |               |                           |  |

**PROVEEDORES**

| Nombre de Campo         | Clave | Tipo de Dato  | Características           | Descripción  |
|-------------------------|-------|---------------|---------------------------|--|
| idProveedor             | PK    | INT           | NOT_NULL / AUTO_INCREMENT | EN ESTA TABLA SE VAN A DETALLAR TODOS LOS PROVEEDORES QUE FUERON CARGADOS EN EL SISTEMA PARA POSTERIORES COMPRAS |
| RazonSocial             |       | VARCHAR (40)  | NOT_NULL                  |  |
| Cuit                    |       | VARCHAR (100) | NOT_NULL                  |  |
| Teléfono                |       | VARCHAR (100) | NOT_NULL                  |  |
| Email                   |       | VARCHAR (100) | NOT_NULL                  |  |
| id_condición_tributaria | FK    | INT           | NOT_NULL                  |  |
|                         |       |               |                           |  |

**ORDENES DE COMPRA**

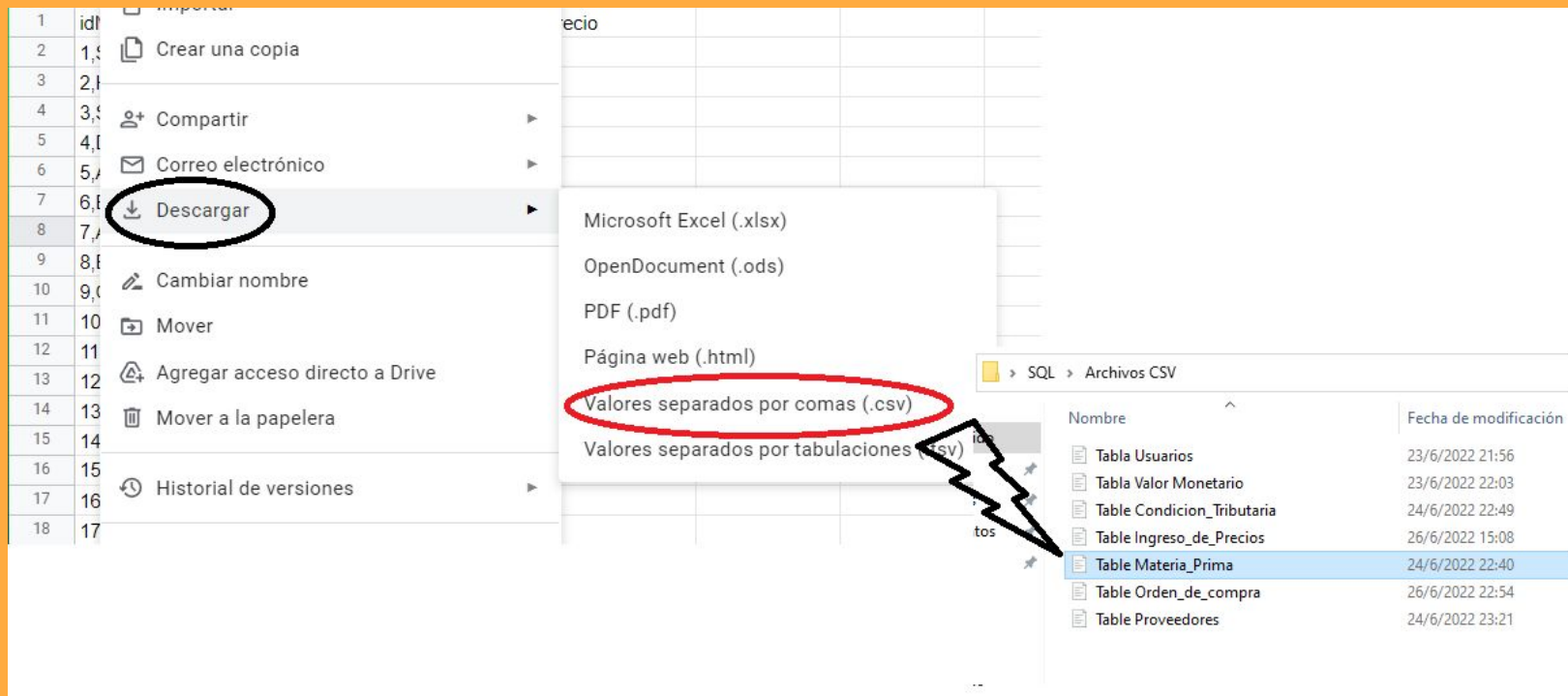
| Nombre de Campo        | Clave | Tipo de Dato | Características           | Descripción   |
|------------------------|-------|--------------|---------------------------|---|
| idOrden                | PK    | INT          | NOT_NULL / AUTO_INCREMENT | EN ESTA TABLA SE VAN A DETALLAR TODAS LAS ORDENES ENVIADAS A LOS DIFERENTES PROVEEDORES INDICANDO EL USUARIO QUE LO REALIZÓ |
| idMoneda               | FK    | INT          | NOT_NULL                  |   |
| idProveedor            | FK    | INT          | NOT_NULL                  |   |
| idMateriaPrima         | FK    | INT          | NOT_NULL                  |   |
| idUsuario              | FK    | INT          | NOT_NULL                  |   |
| FechaEmision           |       | DATETIME     |                           |   |
| Cantidad               |       | VARCHAR (15) |                           |   |
| FechaEntregaMercaderia |       | DATE         |                           |   |
| UnidadMedida           |       | VARCHAR (20) |                           |   |

### INGRESO DE PRECIOS

| Nombre de Campo   | Clave | Tipo de Dato | Características           | Descripción   |
|-------------------|-------|--------------|---------------------------|---|
| idIngreso         | PK    | INT          | NOT_NULL / AUTO_INCREMENT | EN ESTA TABLA SE VAN A DETALLAR TODOS LOS INGRESOS DE MATERIA PRIMA QUE SE VAYAN GENERANDO TENIENDO EN CUENTA EL USUARIO QUE LO CARGA |
| idMateriaPrima    | FK    | INT          | NOT_NULL                  |   |
| idProveedor       | FK    | INT          | NOT_NULL                  |   |
| IdMoneda          | FK    | INT          | NOT_NULL                  |   |
| idUsuario         | FK    | INT          | NOT_NULL                  |   |
| Fecha             |       | DATE         |                           |   |
| Cantidad          |       | VARCHAR (15) |                           |   |
| PrecioActualizado |       | DOUBLE       |                           |   |
| UnidadMedida      |       | VARCHAR (20) |                           |   |



# Inserción de Datos : PRIMER PASO



Como primer paso lo que debemos hacer es descargar el archivo xlsx en formato **.CSV** para poder generar el archivo en el cual importamos los datos de la tabla

## Inserción de Datos : SEGUNDO PASO

```
20 ● ○ Create table if not exists Materia_Prima(  
21     idMateriaPrima INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
22     Nombre varchar (100),  
23     UnidadMedida Varchar (15) not null,  
24     CantidadStock Varchar (15) not null,  
25     Precio varchar (50)  
26 );
```

Output

Action Output

| #   | Time     | Action  | Message           |
|-----|----------|---|-------------------|
| ✓ 1 | 20:59:46 | Create schema if not exists Prueba3   | 1 row(s) affected |
| ✓ 2 | 20:59:49 | use Prueba3   | 0 row(s) affected |
| ✓ 3 | 21:34:58 | Create table if not exists Materia_Prima(idMateriaPrima INT NOT NULL PRIMARY KEY AUTO_INCREMENT, Nombre varchar (100), UnidadMedida Va... | 0 row(s) affected |

Se crea la tabla de manera exitosa detallando las columnas y el tipo de dato que va a contener dicha tabla

## Inserción de Datos : *TERCER PASO*

79 `select * from Materia_Prima;`

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

| idMateriaPrima | Nombre | UnidadMedida | CantidadStock | Precio |
|----------------|--------|--------------|---------------|--------|
| NULL           | NULL   | NULL         | NULL          | NULL   |

Archivos CSV

Archivo Inicio Compartir Vista

SQL > Archivos CSV

| Nombre                     | Fecha de modificación | Tipo                  | Tamaño |
|----------------------------|-----------------------|-----------------------|--------|
| Tabla Usuarios             | 23/6/2022 21:56       | Archivo de origen ... | 2 KB   |
| Tabla Valor Monetario      | 23/6/2022 22:03       | Archivo de origen ... | 1 KB   |
| Table Condicion_Tributaria | 24/6/2022 22:49       | Archivo de origen ... | 1 KB   |
| Table Ingreso de Precios   | 26/6/2022 15:08       | Archivo de origen ... | 1 KB   |
| Table Materia_Prima        | 24/6/2022 22:40       | Archivo de origen ... | 1 KB   |
| Table Orden_de_compra      | 26/6/2022 22:54       | Archivo de origen ... | 2 KB   |

Le hacemos una consulta a la tabla para verificar los datos, donde podremos visualizar el botón de atajo que nos permite poder importar el archivo .CSV creado anteriormente

## ***Inserción de Datos : CUARTO PASO***

**Table Data Import**

Select File to Import

Table Data Import allows you to easily import CSV, JSON datafiles. You can also create destination table on the fly.

File Path: C:\Users\nixco\Desktop\SQL\Archivos CSV\Table Materia\_Prima.csv Browse...

**Table Data Import**

Select Destination

Select destination table and additional options.

☒ Use existing table: prueba3.materia\_prima

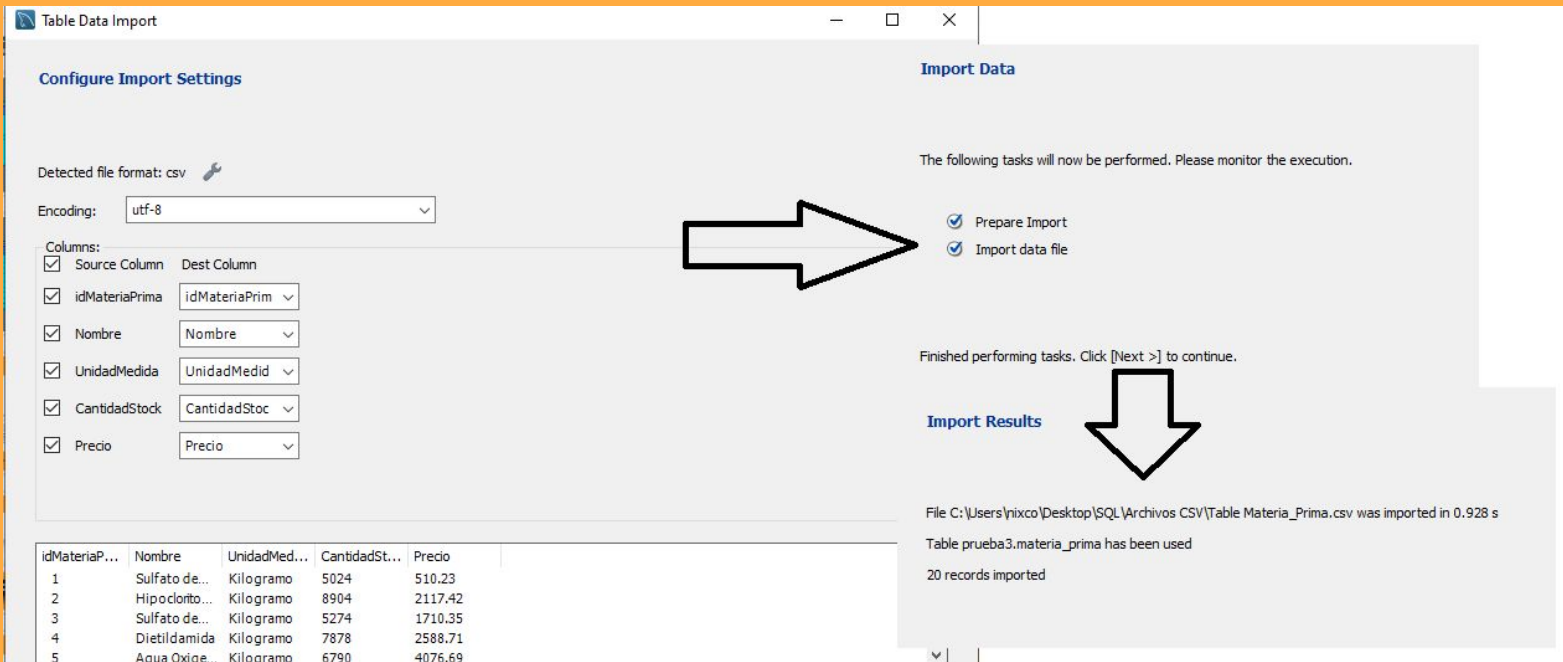
☐ Create new table: prueba3 . Table Materia\_Prima

☐ Truncate table before import

Ingresamos la ruta donde se encuentra nuestro archivo .CSV

Paso posterior se indica que tabla existente deberían agregarse los datos

# Inserción de Datos : QUINTO PASO



**Table Data Import**

**Configure Import Settings**

Detected file format: csv

Encoding: utf-8

Columns:

| Source Column                                      | Dest Column   |
|--|---------------|
| <input checked="" type="checkbox"/> idMateriaPrima | idMateriaPrim |
| <input checked="" type="checkbox"/> Nombre         | Nombre        |
| <input checked="" type="checkbox"/> UnidadMedida   | UnidadMedid   |
| <input checked="" type="checkbox"/> CantidadStock  | CantidadStoc  |
| <input checked="" type="checkbox"/> Precio         | Precio        |

**Import Data**

The following tasks will now be performed. Please monitor the execution.

- ☒ Prepare Import
- ☒ Import data file

Finished performing tasks. Click [Next >] to continue.

**Import Results**

File C:\Users\nixco\Desktop\SQL\Archivos CSV\Table Materia\_Prima.csv was imported in 0.928 s

Table prueba3.materia\_prima has been used

20 records imported

| idMateriaP... | Nombre         | UnidadMed... | CantidadSt... | Precio  |
|---------------|----------------|--------------|---------------|---------|
| 1             | Sulfato de...  | Kilogramo    | 5024          | 510.23  |
| 2             | Hipoclorito... | Kilogramo    | 8904          | 2117.42 |
| 3             | Sulfato de...  | Kilogramo    | 5274          | 1710.35 |
| 4             | Dietildamida   | Kilogramo    | 7878          | 2588.71 |
| 5             | Agua Oxige...  | Kilogramo    | 6790          | 4076.69 |

Último paso: Visualización breve de datos chequeando que las columnas contienen los mismos nombres... Datos insertados de manera exitosa!!!

# VISTAS / VIEWS

```
269 • create or replace view Orden_de_compra_proveedor_view
270 as
271     select o.FechaEmision,o.Cantidad,o.FechaEntregaMercaderia,o.idMateriaPrima,p.RazonSocial,p.Cuit from Orden_de_compra o inner join Proveedores p on o.idProveedor = p.idProveedor;
272 /* select * from Orden_de_compra_proveedor_view; */
273
274
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

|   | FechaEmision        | Cantidad | FechaEntregaMercaderia | idMateriaPrima | RazonSocial | Cuit       |
|---|---------------------|----------|------------------------|----------------|-------------|------------|
| ▶ | 2022-03-23 07:46:41 | 52       | 2021-03-30             | 5              | Span        | 2312801930 |
|   | 2021-12-18 09:54:17 | 32       | 2021-12-25             | 2              | Trippledex  | 80013279   |
|   | 2021-09-29 10:13:18 | 15       | 2021-10-03             | 7              | Cardguard   | 5120778062 |
|   | 2022-02-26 18:56:57 | 36       | 2021-02-28             | 1              | Regrant     | 8027326788 |
|   | 2022-05-09 11:40:35 | 84       | 2021-05-11             | 19             | Alpha       | 1174978562 |
|   | 2021-11-04 02:09:32 | 3        | 2021-11-14             | 8              | Pannier     | 7700533024 |
|   | 2022-05-14 05:43:24 | 98       | 2021-05-20             | 6              | Alpha       | 9402735550 |

CODER HOUSE

Las vistas fueron creadas para que los usuarios puedan visualizar una tabla o una unión de tablas mediante un **JOIN** con el fin de conservar la seguridad de los datos y para que la BDD tenga un mejor rendimiento

# DIFERENTES VISTAS

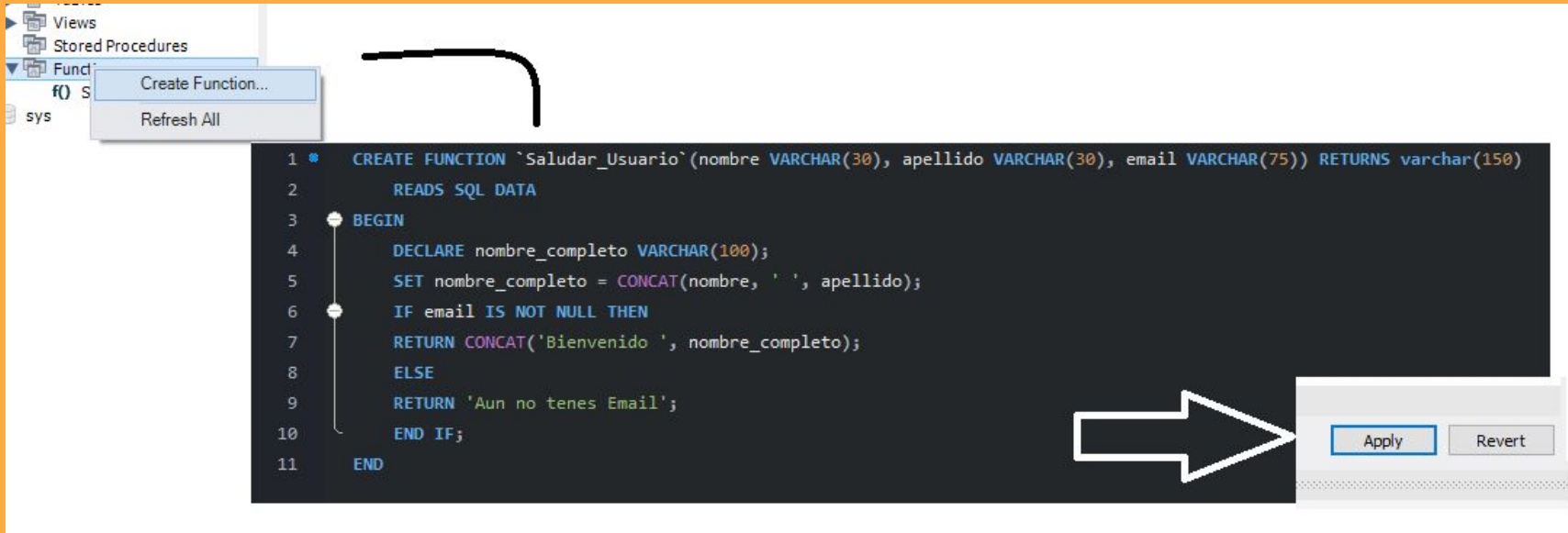
```
create or replace view Materia_Prima_y_Usuarios_view
as
    select m.idMateriaPrima,m.Nombre,m.CantidadStock,u.idUsuario from Materia_Prima m join Usuarios u where m.idMateriaPrima >= 10;

update Materia_Prima_y_Usuarios_view
set CantidadStock = True
where idMateriaPrima <= 10 or idUsuario = 1 ;

/* select * from Materia_Prima_y_Usuarios_view; */
```

```
create or replace view Costo_Materia_Prima
as
    select distinct * from Materia_Prima where Precio >= 1000 or 4576;
/* select * from Costo_Materia_Prima; */
```

# FUNCIONES



The screenshot shows a database management interface. On the left, a tree view displays 'Views', 'Stored Procedures', and 'Functions'. A context menu is open over the 'Functions' folder, with 'Create Function...' selected. The main area is a SQL editor with the following code:

```
1 * CREATE FUNCTION `Saludar_Usuario` (nombre VARCHAR(30), apellido VARCHAR(30), email VARCHAR(75)) RETURNS varchar(150)
2   READS SQL DATA
3   BEGIN
4     DECLARE nombre_completo VARCHAR(100);
5     SET nombre_completo = CONCAT(nombre, ' ', apellido);
6     IF email IS NOT NULL THEN
7       RETURN CONCAT('Bienvenido ', nombre_completo);
8     ELSE
9       RETURN 'Aun no tenes Email';
10    END IF;
11  END
```

Below the code editor, there are 'Apply' and 'Revert' buttons. A large white arrow points from the code editor to the 'Apply' button. A black bracket is drawn above the code editor, spanning from the 'Create Function...' menu item to the code.

Las funciones se crean con la plantilla que nos deja utilizar mysql, se definen los parámetros con los tipos de datos detallando y declarando la función que queremos que realice la consulta y usar el Apply para que nos cree dicha función



# STORED PROCEDURES PASO UNO

CODER HOUSE



The screenshot shows the SQL Server Enterprise Manager interface. In the left-hand tree view, the 'Stored Procedures' folder is expanded. A right-click context menu is open over this folder, with the 'Create Stored Procedure...' option highlighted. A red line points from this menu option to the SQL code block below.

```
CREATE DEFINER='root'@'localhost' PROCEDURE `Insert_Proveedores_Descendente`(IN idCondicionTributaria INT, RazonSocial VARCHAR (100), Cuit VARCHAR(100))
BEGIN
    set @idProveedor = (select max(idProveedor) from Proveedores)+1;
    insert into Proveedores (idCondicionTributaria, idProveedor, RazonSocial, Cuit) values (idCondicionTributaria, @idProveedor, RazonSocial, Cuit);
    select * from Proveedores order by idProveedor DESC;
END
```

Primer paso, creamos el stored procedures, indicando en la sintaxis: nombre del sp, los parámetros que vamos a utilizar, declaramos la variable y armamos la query que vamos a solicitar, en este caso es que ingrese un nuevo dato y me los ordene de forma descendente para poder visualizar como primer dato el nuevo que ingrese

# STORED PROCEDURES SEGUNDO PASO

CODER HOUSE

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Review the SQL Script to be Applied on the Database

Online DDL

Algorithm: Default Lock Type: Default

```
1 USE `lobster`;
2 DROP procedure IF EXISTS `Insert_Proveedores_Descendente`;
3
4 USE `lobster`;
5 DROP procedure IF EXISTS `lobster`.`Insert_Proveedores_Descendente`;
6 ;
7
8 DELIMITER $$
9 USE `lobster` $$
10 CREATE DEFINER=`root`@`localhost` PROCEDURE `Insert_Proveedores_Descendente`
11 BEGIN
12     set @idProveedor = (select max(idProveedor) from Proveedores)+1;
13     insert into Proveedores (idCondicionTributaria, idProveedor, RazonSocial, Cuit)
14     select * from Proveedores order by idProveedor DESC;
15 END$$
16
17 DELIMITER ;
```

Back Apply Cancel

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Applying SQL script to the database

The following tasks will now be executed. Please monitor the execution. Press Show Logs to see the execution logs.

☒ Execute SQL Statements

SQL script was successfully applied to the database.

Show Logs

Back Finish Cancel

Segundo paso, le damos a aplicar y si toda la sintaxis está correcta debería dejarnos aplicar correctamente nuestro script en la BDD

debería dejarnos aplicar correctamente nuestro script en la BDD

# STORED PROCEDURES SEGUNDO PASO

CODER HOUSE

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Review the SQL Script to be Applied on the Database

Online DDL

Algorithm: Default Lock Type: Default

```
1 USE `lobster`;
2 DROP procedure IF EXISTS `Insert_Proveedores_Descendente`;
3
4 USE `lobster`;
5 DROP procedure IF EXISTS `lobster`.`Insert_Proveedores_Descendente`;
6 ;
7
8 DELIMITER $$
9 USE `lobster` $$
10 CREATE DEFINER=`root`@`localhost` PROCEDURE `Insert_Proveedores_Descendente`
11 BEGIN
12     set @idProveedor = (select max(idProveedor) from Proveedores)+1;
13     insert into Proveedores (idCondicionTributaria, idProveedor, RazonSocial, Cuit)
14     select * from Proveedores order by idProveedor DESC;
15 END$$
16
17 DELIMITER ;
```

Back Apply Cancel

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Applying SQL script to the database

The following tasks will now be executed. Please monitor the execution. Press Show Logs to see the execution logs.

☒ Execute SQL Statements

SQL script was successfully applied to the database.

Show Logs

Back Finish Cancel

Segundo paso, le damos a aplicar y si toda la sintaxis está correcta debería dejarnos aplicar correctamente nuestro script en la BDD

debería dejarnos aplicar correctamente nuestro script en la BDD

# STORED PROCEDURES SEGUNDO PASO

CODER HOUSE

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Review the SQL Script to be Applied on the Database

Online DDL

Algorithm: Default Lock Type: Default

```
1 USE `lobster`;  
2 DROP procedure IF EXISTS `Insert_Proveedores_Descendente`;  
3  
4 USE `lobster`;  
5 DROP procedure IF EXISTS `lobster`.`Insert_Proveedores_Descendente`;  
6 ;  
7  
8 DELIMITER $$  
9 USE `lobster` $$  
10 CREATE DEFINER=`root`@`localhost` PROCEDURE `Insert_Proveedores_Descendente`  
11 BEGIN  
12     set @idProveedor = (select max(idProveedor) from Proveedores)+1;  
13     insert into Proveedores (idCondicionTributaria, idProveedor, RazonSocial, Cuit)  
14     select * from Proveedores order by idProveedor DESC;  
15 END$$  
16  
17 DELIMITER ;
```

Back Apply Cancel

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Applying SQL script to the database

The following tasks will now be executed. Please monitor the execution. Press Show Logs to see the execution logs.

☒ Execute SQL Statements

SQL script was successfully applied to the database.

Show Logs

Back Finish Cancel

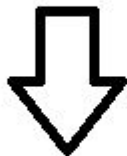
Segundo paso, le damos a aplicar y si toda la sintaxis está correcta debería dejarnos aplicar correctamente nuestro script en la BDD

debería dejarnos aplicar correctamente nuestro script en la BDD

## STORED PROCEDURES TERCER PASO

CODER HOUSE

```
call Insert_Proveedores_Descendente (3, "Prueba2", 48339558);  
SELECT * FROM PROVEEDORES;
```



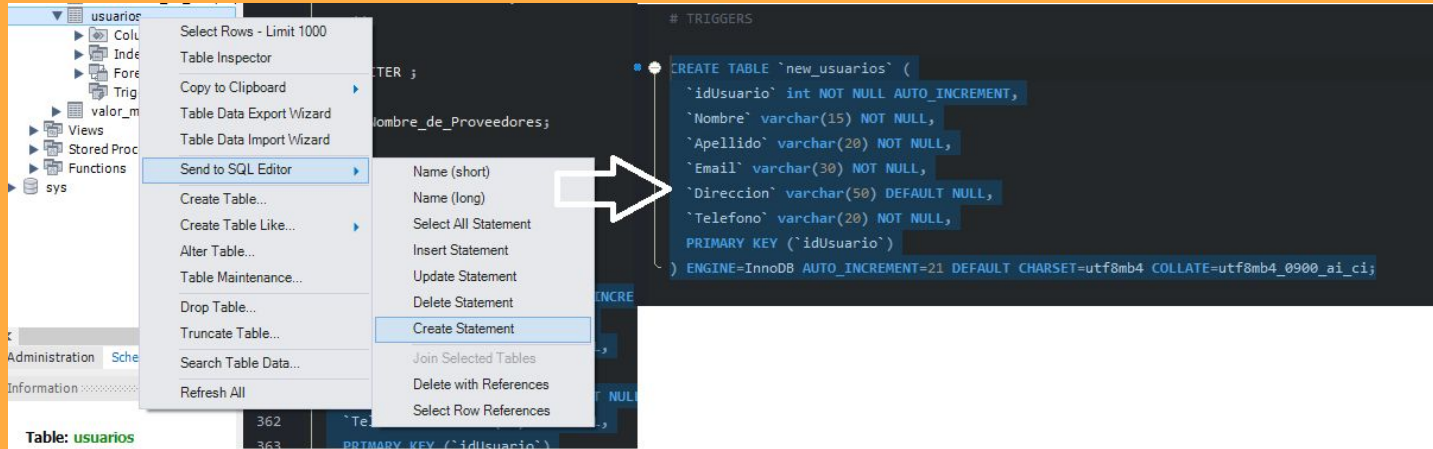
|      |         |          |      |      |      |
|------|---------|----------|------|------|------|
| 24   | Prueba2 | 48339558 | NULL | NULL | 3    |
| NULL | NULL    | NULL     | NULL | NULL | NULL |

Tercer paso, si todo nos da correcto, deberíamos hacerle la query respectiva, en la cual nos va a insertar nuestro dato a la vez que lo va a ordenar de forma ascendente para poder visualizar nuestro nuevo dato en primera fila

dato en primera fila

a ordenar de forma ascendente para poder visualizar nuestro nuevo

# TRIGGERS PASO UNO



Primer paso, debemos crear una tabla espejo en la cual vayamos a agregar los nuevos usuarios que se van a ingresar...

agregar los nuevos usuarios que se van a ingresar...

## TRIGGERS PASO DOS

CODER HOUSE

```
#SENTENCIA TRIGGER  
CREATE TRIGGER AFT_INS_Usuarios_log  
AFTER INSERT ON usuarios  
FOR EACH ROW  
INSERT INTO new_usuarios  
VALUES (NEW.idUsuario, NEW.Nombre, NEW.Apellido, NEW.Email, NEW.Direccion, NEW.Telefono);
```

Como segundo paso, deberemos crear la sentencia del trigger con la sentencia que se muestra en la foto

## TRIGGERS PASO TRES

CODER HOUSE

```
INSERT INTO Usuarios  
VALUES (21, "Nicolas", "Rubini", "nrubini@surfactan.com.ar", "calle falsa 1234", 1189768596);
```

```
1 * SELECT * FROM lobster.new_usuarios;
```



| Result Grid                                  |           |         |          |                          |                  |            |
|--|-----------|---------|----------|--------------------------|------------------|------------|
| Filter Rows: <input type="text"/>            |           |         |          |                          |                  |            |
| Edit:    Export/Import:   Wrap Cell Content: |           |         |          |                          |                  |            |
|  | idUsuario | Nombre  | Apellido | Email                    | Direccion        | Telefono   |
| ▶  | 21        | Nicolas | Rubini   | nrubini@surfactan.com.ar | calle falsa 1234 | 1189768596 |
| *  | NULL      | NULL    | NULL     | NULL                     | NULL             | NULL       |

Utilizamos el insert para crear nuestro nuevo dato y realizamos el correspondiente llamado a la tabla...

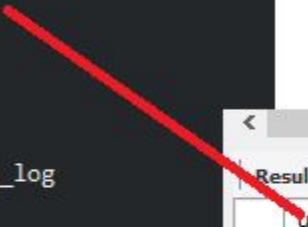
En la cual, podemos visualizar el nuevo insert



## TRIGGERS PASO CUATRO

CODER HOUSE

```
CREATE TABLE logs(  
    usuario Varchar(100),  
    Fecha date,  
    Hora varchar(100)  
);  
  
CREATE TRIGGER BEF_DEL_Usuarios_log  
BEFORE DELETE ON usuarios  
FOR EACH ROW  
INSERT INTO logs  
values (USER(), curdate(), curtime());  
  
#TEST DE TRIGGER DELETE  
DELETE FROM Usuarios where idUsuario = 22;
```



|   | usuario        | Fecha      | Hora     |
|---|----------------|------------|----------|
| ▶ | root@localhost | 2022-07-13 | 20:07:40 |

Por último, definimos y creamos una tabla donde se va a almacenar los datos de quien haga cambios en la tabla llamada LOG

Y utilizamos el before para que el trigger se active después de ejecutar el DML

# CREACIÓN DE USUARIOS

## PRIMER PASO

CODER HOUSE

```
#CREACION DE USUARIOS  
USE mysql;  
#SE CREAN LOS 2 USUARIOS CON SU RESPECTIVA CONTRASEÑA  
CREATE USER Nicolas@127.0.0.1 IDENTIFIED BY "1234";  
CREATE USER Roberto@127.0.0.1 IDENTIFIED BY "12345";  
SELECT * FROM USER;
```

Como primer paso, lo que debemos hacer es crear la sentencia de creación de usuarios identificándose con una contraseña propia.

## CREACIÓN DE USUARIOS SEGUNDO PASO

```
#AL PRIMER USUARIO SE LE DA PERMISO SOLAMENTE DE LECTURA  
GRANT SELECT ON Lobster.* TO Nicolas@127.0.0.1;  
#AL SEGUNDO USUARIO SE LE DA PERMISO DE LECTURA, INSERCIÓN DE DATOS Y MODIFICACION  
GRANT SELECT, INSERT, UPDATE ON Lobster.* TO Roberto@127.0.0.1;
```

Segundo Paso, es darle los permisos correspondientes a cada usuario utilizando "GRANT" y dándole permisos sobre una base de datos específica, pero teniendo acceso a las tablas

## CREACIÓN DE USUARIOS TERCER PASO

CODER HOUSE

```
439 #VALIDACION DE PERMISOS A USUARIOS
440 SHOW GRANTS FOR Nicolas@127.0.0.1;
```

< Result Grid Filter Rows: Export: Wrap

|   | Grants for Nicolas@127.0.0.1   |
|---|--|
| ▶ | GRANT USAGE ON *,* TO `Nicolas`@`127.0.0.1`<br>GRANT SELECT ON `lobster`,* TO `Nicolas`@`... |

```
441 SHOW GRANTS FOR Roberto@127.0.0.1;
```

< Result Grid Filter Rows: Export: Wrap

|   | Grants for Roberto@127.0.0.1   |
|---|--|
| ▶ | GRANT USAGE ON *,* TO `Roberto`@`127.0....<br>GRANT SELECT, INSERT, UPDATE ON `lobster`... |

Como tercer paso, corremos la query para visualizar si los usuarios tienen aplicados de manera correcta los permisos que les cedimos.

# TRANSACCIÓN PRIMER TABLA

CODER HOUSE

```
#TRANSACCIONES DE ELIMINACION DE LA PRIMER TABLA
```

```
START TRANSACTION;
```

```
DELETE FROM usuarios
```

```
where id = 3, 4, 5, 6;
```

```
ROLLBACK;
```

```
COMMIT;
```

```
INSERT INTO Usuarios (`idUsuario`,`Nombre`,`Apellido`,`Email`,`Direccion`,`Telefono`) VALUES (3,'Michele','Hyams','mhyams2@ask.com','78 Dixon Plaza','+62-236-660-3623');
```

```
INSERT INTO Usuarios (`idUsuario`,`Nombre`,`Apellido`,`Email`,`Direccion`,`Telefono`) VALUES (4,'Barris','Deave','bdeave3@columbia.edu','03 Longview Park','+51-113-749-9114');
```

```
INSERT INTO Usuarios (`idUsuario`,`Nombre`,`Apellido`,`Email`,`Direccion`,`Telefono`) VALUES (5,'Melita','Isselee','misselee4@hubpages.com','99161 Village Green Parkway','+86-988-574-8747');
```

```
INSERT INTO Usuarios (`idUsuario`,`Nombre`,`Apellido`,`Email`,`Direccion`,`Telefono`) VALUES (6,'Mara','Rashleigh','mrashleigh5@bloomberg.com','40 Old Gate Drive','+234-328-691-0491');
```

Primero creamos una transacción en la cual eliminaremos un dato, seguido de un rollback por si nos equivocamos o un commit para confirmar el borrado del dato permanente

# TRANSACCIÓN SEGUNDA

## TABLA

CODER HOUSE

```
#TRANSACCION DE INSERCIÓN DE DATOS CON SAVEPOINTS
```

```
START TRANSACTION;
```

```
INSERT INTO Materia_Prima (`idMateriaPrima`,`Nombre`,`UnidadMedida`,`CantidadStock`,`Precio`) VALUES (100,'prueba1','Kilogramo','7878','2588.71');
```

```
INSERT INTO Materia_Prima (`idMateriaPrima`,`Nombre`,`UnidadMedida`,`CantidadStock`,`Precio`) VALUES (101,'prueba2','Kilogramo','7878','2588.71');
```

```
INSERT INTO Materia_Prima (`idMateriaPrima`,`Nombre`,`UnidadMedida`,`CantidadStock`,`Precio`) VALUES (102,'prueba3','Kilogramo','7878','2588.71');
```

```
INSERT INTO Materia_Prima (`idMateriaPrima`,`Nombre`,`UnidadMedida`,`CantidadStock`,`Precio`) VALUES (103,'Prueba4','Kilogramo','7878','2588.71');
```

```
SAVEPOINT Primero;
```

```
INSERT INTO Materia_Prima (`idMateriaPrima`,`Nombre`,`UnidadMedida`,`CantidadStock`,`Precio`) VALUES (104,'prueba5','Kilogramo','7878','2588.71');
```

```
INSERT INTO Materia_Prima (`idMateriaPrima`,`Nombre`,`UnidadMedida`,`CantidadStock`,`Precio`) VALUES (105,'prueba6','Kilogramo','7878','2588.71');
```

```
INSERT INTO Materia_Prima (`idMateriaPrima`,`Nombre`,`UnidadMedida`,`CantidadStock`,`Precio`) VALUES (106,'prueba7','Kilogramo','7878','2588.71');
```

```
INSERT INTO Materia_Prima (`idMateriaPrima`,`Nombre`,`UnidadMedida`,`CantidadStock`,`Precio`) VALUES (107,'prueba8','Kilogramo','7878','2588.71');
```

```
SAVEPOINT Segundo;
```

```
RELEASE SAVEPOINT Primero;
```

En la segunda transacción deberemos hacer inserción de datos en una tabla, utilizando savepoints cada 4 datos para poder volver o eliminar un bloque entero utilizando los guardados / savepoints... Eliminamos el primer savepoint, con lo cual, se eliminarían los primeros 4 datos que hayamos insertado

## *Informes acerca de la BDD*

Esta BDD nos brindará informes acerca de los usuarios con los cuales cuenta la empresa, sus proveedores (dando de alta nuevos), el stock de materia prima con el cual cuenta la empresa, y un registro de pedidos que se van a ir comprando para sumar al stock.

Son informes detallados de lo que puede contener el departamento de compras para facilitar datos de todas las transacciones que se realicen

# ***Herramientas y Tecnologías Utilizadas***

- 1) Diagrams.net = Creación de diagrama ER
- 2) Mockaroo = Creación de Inserts
- 3) Microsoft Excel = Creación de Tablas
- 4) MySql WorkBench = Creación de Scripts de la BDD
- 5) Git / GitHub = Creación de repositorios público para entrega del proyecto final