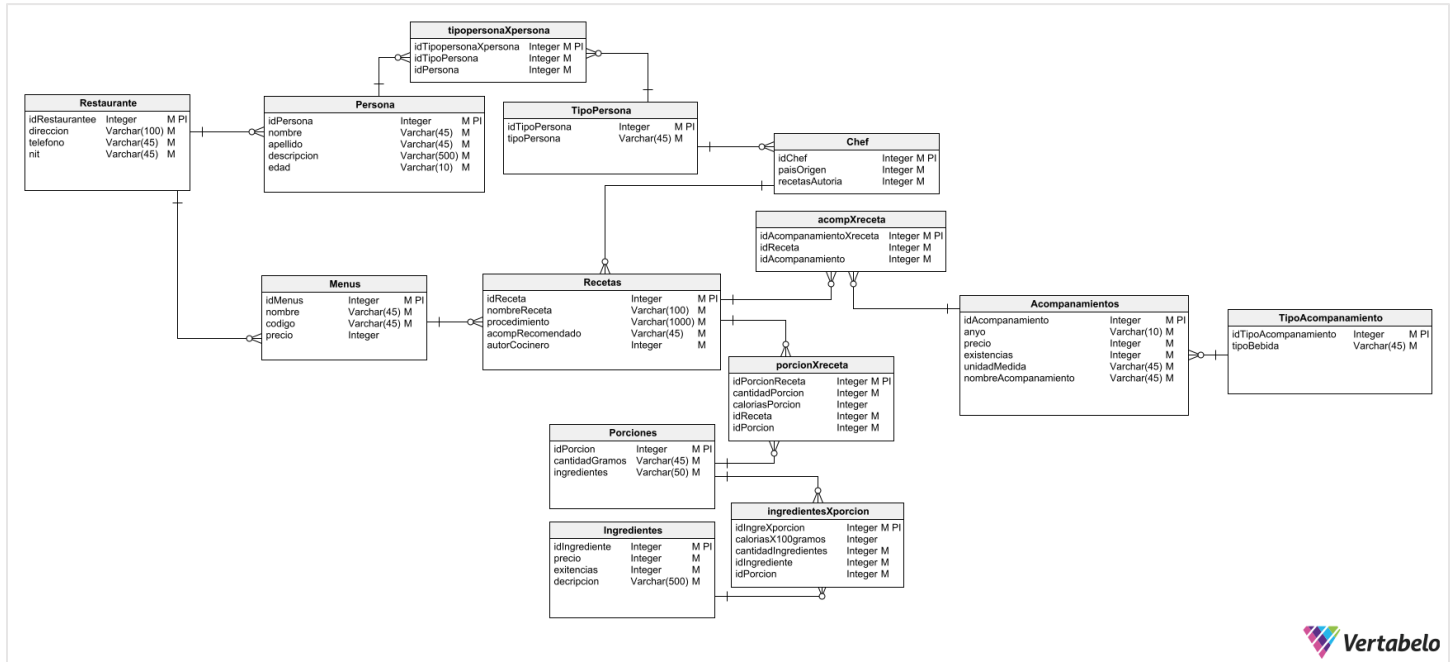


# TALLER BASES DE DATOS:



## Casos de aplicación de la sentencia DDL ALTER en MySQL y PostgreSQL.

Ing. Luis Felipe Narváez Gómez. E-mail: luis.narvaez@usantoto.edu.co. Cod: 2312660. Facultad de Ingeniería de Sistemas.

La base de datos que tenemos de restaurantes corresponde a “db\_restaurant”, su estructura en el diagrama entidad relación es el siguiente:



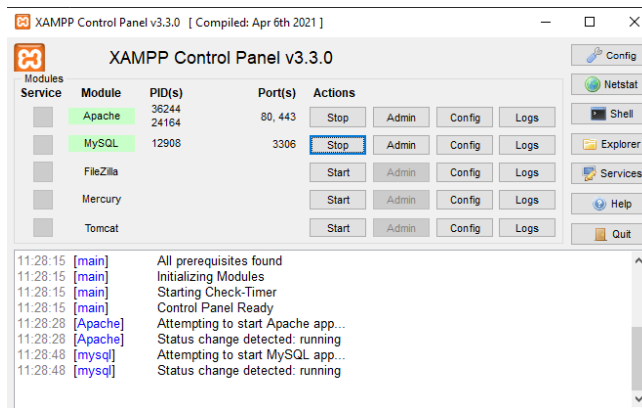
Esta base de datos fue generada en Vertabelo, este recurso web tiene la capacidad y función de generar el modelo y físico y a partir del generar código SQL respectivo para crear la base de datos en diferentes motores como IBM DB2, PostgreSQL, Oracle Database, Microsoft SQL Server, MySQL, HSQLDB, SQLite, Amazon Redshift, BigQuery y Snowflake. Utilizando esta herramienta podemos tener para este ejercicio la Database para trabajarla con MySQL Xampp y PostgreSQL PGAdmin 4.

	<b>bd_restaurant_Physical_Export_MySQL_v4_create.sql</b> Tipo: SQL Text File	Fecha de modificación: 20/09/2021 12:55 Tamaño: 7,11 KB
	<b>bd_restaurant_Physical_Export_PostgreSQL13_v4_create.sql</b> Tipo: SQL Text File	Fecha de modificación: 22/10/2021 11:09 Tamaño: 9,70 KB

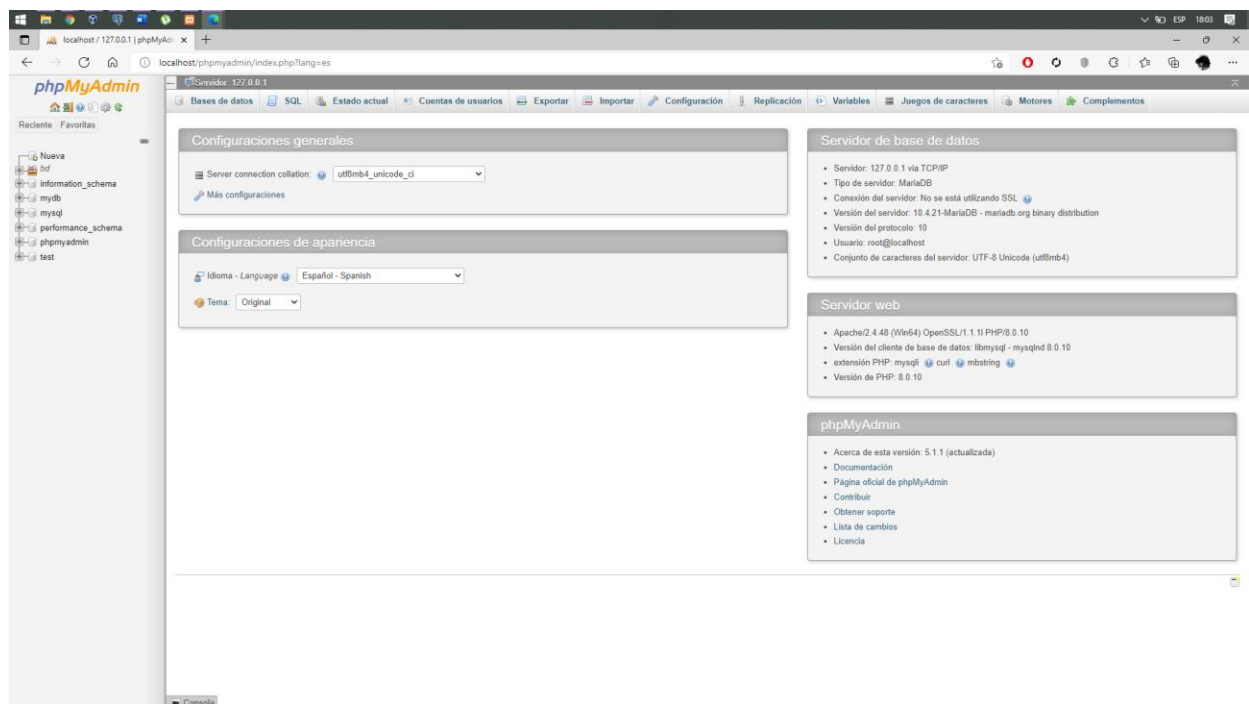
De esta manera este Ejercicio se dividirá en dos fases, la solución dentro de MySQL y el desarrollo dentro de PostgreSQL.

# MySQL XAMPP v3.3.0

Haciendo uso de la GUI de Xampp podemos montar esta base de datos, para esto debemos lazar primeramente el servicio dentro de nuestra computadora.





Luego por medio de algún navegador, podemos entrar a la dirección local de este servicio siendo “[localhost / 127.0.0.1 | phpMyAdmin 5.1.1](http://localhost/127.0.0.1/phpMyAdmin/5.1.1)” la cual nos mostrara lo siguiente:



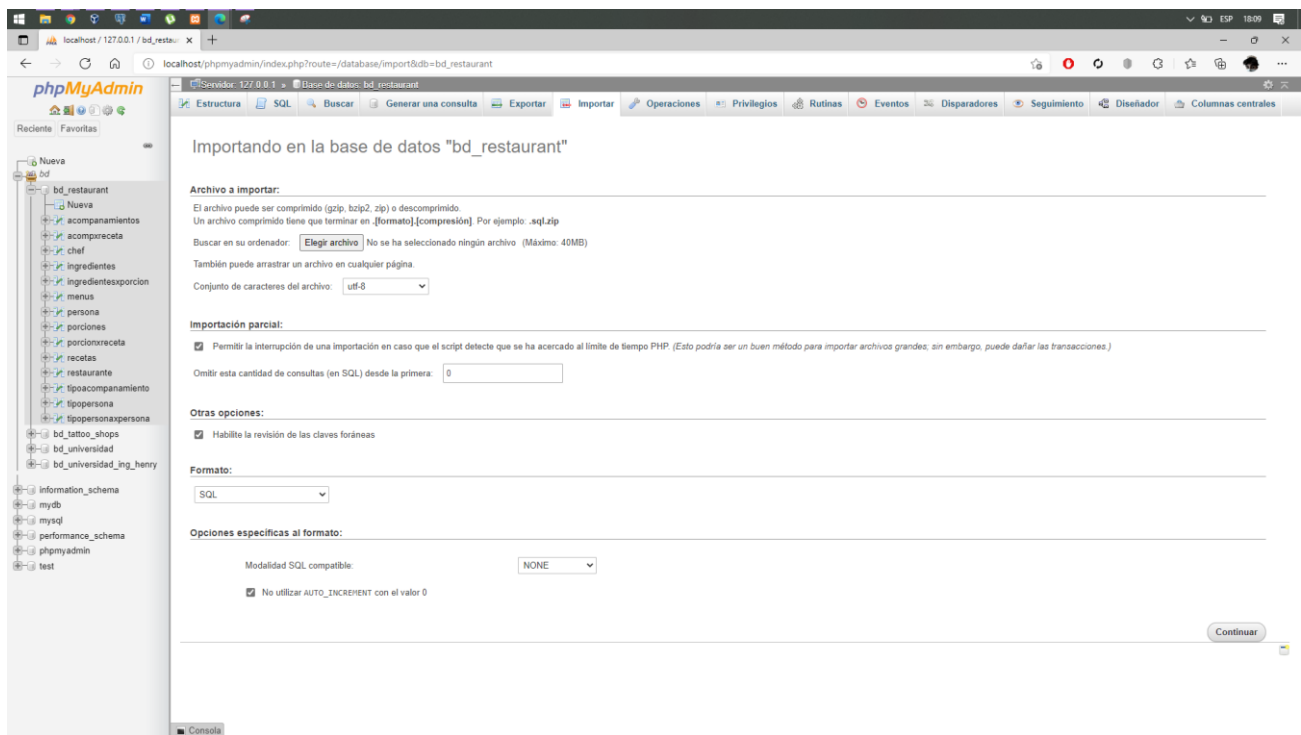
Para crear la nueva base de datos, basta con ir al menú lateral de la pantalla, seleccionar la herramienta vertical que dice “nueva” y escribir el nombre de la base de datos y un cotejamiento de “utf8mb4\_general\_ci” el cual nos permitirá ingresar datos que contengan caracteres especiales de algunos idiomas, como el español, portugués, francés, etc. Lenguas románicas o del árbol de lenguas nórdicas, las cuales utilizan tildes, la “ñ”, aspectos como “”, etc.

## Bases de datos

 Crear base de datos 

Nombre de la base de datos

Una vez creada, podemos seleccionarla en el mismo menú lateral en vertical de la pantalla y dentro de ella, seleccionamos en la barra de menús superior la opción de “importar”. En ella solo debemos elegir nuestro archivo de extensión “SQL” y que el formato este seleccionado precisamente en este tipo de extensión.



Luego de esto, solo queda ejecutar el código y se montaran las tablas y restricciones propias de la Base de Datos. Podemos comprobar cada una de las existencias de estas tablas y sus atributos o columnas mediante consultas SQL, la siguiente imagen corresponde a las tablas existentes dentro de la base de datos en el Motor MySQL de Xampp:

Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
<input type="checkbox"/> <b>acompanamientos</b>	Examinar  Estructura  Buscar  Insertar  Vaciar  Eliminar	0	InnoDB	utf8mb4_general_ci	32.0 KB	-
<input type="checkbox"/> <b>acompxreceta</b>	Examinar  Estructura  Buscar  Insertar  Vaciar  Eliminar	0	InnoDB	utf8mb4_general_ci	48.0 KB	-
<input type="checkbox"/> <b>chef</b>	Examinar  Estructura  Buscar  Insertar  Vaciar  Eliminar	0	InnoDB	utf8mb4_general_ci	32.0 KB	-
<input type="checkbox"/> <b>ingredientes</b>	Examinar  Estructura  Buscar  Insertar  Vaciar  Eliminar	0	InnoDB	utf8mb4_general_ci	16.0 KB	-
<input type="checkbox"/> <b>ingredientesexporcion</b>	Examinar  Estructura  Buscar  Insertar  Vaciar  Eliminar	0	InnoDB	utf8mb4_general_ci	48.0 KB	-
<input type="checkbox"/> <b>menus</b>	Examinar  Estructura  Buscar  Insertar  Vaciar  Eliminar	0	InnoDB	utf8mb4_general_ci	32.0 KB	-
<input type="checkbox"/> <b>persona</b>	Examinar  Estructura  Buscar  Insertar  Vaciar  Eliminar	0	InnoDB	utf8mb4_general_ci	32.0 KB	-
<input type="checkbox"/> <b>porciones</b>	Examinar  Estructura  Buscar  Insertar  Vaciar  Eliminar	0	InnoDB	utf8mb4_general_ci	16.0 KB	-
<input type="checkbox"/> <b>porcionxreceta</b>	Examinar  Estructura  Buscar  Insertar  Vaciar  Eliminar	0	InnoDB	utf8mb4_general_ci	48.0 KB	-
<input type="checkbox"/> <b>recetas</b>	Examinar  Estructura  Buscar  Insertar  Vaciar  Eliminar	0	InnoDB	utf8mb4_general_ci	48.0 KB	-
<input type="checkbox"/> <b>restaurante</b>	Examinar  Estructura  Buscar  Insertar  Vaciar  Eliminar	0	InnoDB	utf8mb4_general_ci	16.0 KB	-
<input type="checkbox"/> <b>tipoacompanamiento</b>	Examinar  Estructura  Buscar  Insertar  Vaciar  Eliminar	0	InnoDB	utf8mb4_general_ci	16.0 KB	-
<input type="checkbox"/> <b>tipopersona</b>	Examinar  Estructura  Buscar  Insertar  Vaciar  Eliminar	0	InnoDB	utf8mb4_general_ci	16.0 KB	-
<input type="checkbox"/> <b>tipopersonaxpersona</b>	Examinar  Estructura  Buscar  Insertar  Vaciar  Eliminar	0	InnoDB	utf8mb4_general_ci	48.0 KB	-
<b>14 tablas</b>	<b>Número de filas</b>	<b>0</b>	<b>InnoDB</b>	<b>utf8mb4_general_ci</b>	<b>448.0 KB</b>	<b>0 B</b>

La siguiente Imagen Corresponde a las consultas realizadas para consultar los elementos dentro de las tablas. Ya que estas no están pobladas, solo se mostrarán los nombres de las columnas o los nombres de los atributos en cada tabla:

```

SELECT * FROM restaurante;
SELECT * FROM persona;
SELECT * FROM tipopersonaxpersona;
SELECT * FROM tipopersona;
SELECT * FROM chef;
SELECT * FROM menus;
SELECT * FROM recetas;
SELECT * FROM porcionxreceta;
SELECT * FROM porciones;
SELECT * FROM ingredientesxporcion;
SELECT * FROM ingredientes;
SELECT * FROM acompxreceta;
SELECT * FROM acompanamientos;
SELECT * FROM tipoacompanamiento;

```

Estos son los resultados de cada una de las consultas:

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0011 segundos.)

`SELECT * FROM restaurante;`

Perfilando [ [Editar en línea](#) ] [ [Editar](#) ] [ [Explicar SQL](#) ] [ [Crear código PHP](#) ] [ [Actualizar](#) ]

`idRestaurante direccion telefono nit`

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0016 segundos.)

`SELECT * FROM persona;`

Perfilando [ [Editar en línea](#) ] [ [Editar](#) ] [ [Explicar SQL](#) ] [ [Crear código PHP](#) ] [ [Actualizar](#) ]

`idPersona nombre apellido descripcion edad Restaurante_idRestaurante`

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0006 segundos.)

`SELECT * FROM tipopersonaxpersona;`

Perfilando [ [Editar en línea](#) ] [ [Editar](#) ] [ [Explicar SQL](#) ] [ [Crear código PHP](#) ] [ [Actualizar](#) ]

`idTipopersonaxpersona idTipoPersona idPersona Persona_idPersona TipoPersona_idTipoPersona`

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0006 segundos.)

`SELECT * FROM tipopersona;`

Perfilando [ [Editar en línea](#) ] [ [Editar](#) ] [ [Explicar SQL](#) ] [ [Crear código PHP](#) ] [ [Actualizar](#) ]

`idTipoPersona tipoPersona`

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0006 segundos.)

`SELECT * FROM chef;`

Perfilando [ [Editar en línea](#) ] [ [Editar](#) ] [ [Explicar SQL](#) ] [ [Crear código PHP](#) ] [ [Actualizar](#) ]

`idChef paisOrigen recetasAutoria TipoPersona_idTipoPersona`

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0006 segundos.)

`SELECT * FROM menus;`

Perfilando [ [Editar en línea](#) ] [ [Editar](#) ] [ [Explicar SQL](#) ] [ [Crear código PHP](#) ] [ [Actualizar](#) ]

`idMenus nombre codigo precio Restaurante_idRestaurante`

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0006 segundos.)

`SELECT * FROM recetas;`

Perfilando [ [Editar en línea](#) ] [ [Editar](#) ] [ [Explicar SQL](#) ] [ [Crear código PHP](#) ] [ [Actualizar](#) ]

`idReceta nombreReceta procedimiento acompRecomendado autorCocinero Menus_idMenus Chef_idChef`

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0007 segundos.)

`SELECT * FROM porcionxreceta;`

Perfilando [ [Editar en línea](#) ] [ [Editar](#) ] [ [Explicar SQL](#) ] [ [Crear código PHP](#) ] [ [Actualizar](#) ]

`idPorcionReceta cantidadPorcion caloriasPorcion idReceta idPorcion Recetas_idReceta Porciones_idPorcion`

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0006 segundos.)

`SELECT * FROM porciones;`

Perfilando [ [Editar en línea](#) ] [ [Editar](#) ] [ [Explicar SQL](#) ] [ [Crear código PHP](#) ] [ [Actualizar](#) ]

`idPorcion cantidadGramos ingredientes`

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0008 segundos.)

`SELECT * FROM ingredientesxporcion;`

Perfilando [ [Editar en línea](#) ] [ [Editar](#) ] [ [Explicar SQL](#) ] [ [Crear código PHP](#) ] [ [Actualizar](#) ]

`idIngredXporcion caloriasX100gramos cantidadIngredientes idIngrediente idPorcion Porciones_idPorcion Ingredientes_idIngrediente`

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0009 segundos.)

`SELECT * FROM ingredientes;`

Perfilando [ [Editar en línea](#) ] [ [Editar](#) ] [ [Explicar SQL](#) ] [ [Crear código PHP](#) ] [ [Actualizar](#) ]

`idIngrediente precio existencias descripcion`

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0010 segundos.)

`SELECT * FROM acompxreceta;`

Perfilando [ [Editar en línea](#) ] [ [Editar](#) ] [ [Explicar SQL](#) ] [ [Crear código PHP](#) ] [ [Actualizar](#) ]

`idAcompanamientoXreceta idReceta idAcompanamiento Recetas_idReceta Acompanamientos_idAcompanamiento`

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0007 segundos.)

`SELECT * FROM acompanamientos;`

Perfilando [ [Editar en línea](#) ] [ [Editar](#) ] [ [Explicar SQL](#) ] [ [Crear código PHP](#) ] [ [Actualizar](#) ]

`idAcompanamiento anyo precio existencias unidadMedida nombreAcompanamiento TipoAcompanamiento_idTipoAcompanamiento`

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0008 segundos.)

`SELECT * FROM tipoacompanamiento;`

Perfilando [ [Editar en línea](#) ] [ [Editar](#) ] [ [Explicar SQL](#) ] [ [Crear código PHP](#) ] [ [Actualizar](#) ]

`idTipoAcompanamiento tipoBebida`

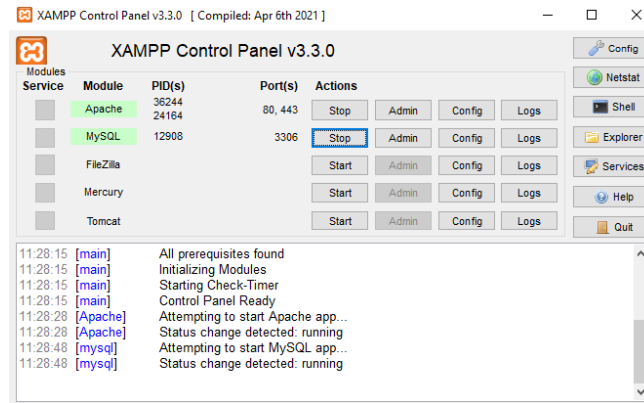
Ahora bien, para movernos con mayor libertad por las tablas y la misma base de datos podemos transportarnos a un entorno de trabajo distinto a Xampp como lo es la Consola de Comandos. Para esto debemos primero acceder a la ruta donde tenemos instalado Xampp y abrir el directorio de bin.

```

C:\Users\ruiiso>cd..
C:\Users>cd..
C:\>D:
D:\>cd D:\Software\Xampp\mysql\bin

```

Una vez hecho esto, ejecutamos nuestro Xampp y activamos tanto Apache como MySQL.



Una vez hecho esto, podemos iniciar el motor de MySQL para trabajar desde la consola de comandos.

```
D:\Software\Xampp\mysql\bin>mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 18
Server version: 10.4.21-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

Ahora si podemos cambiar a la base de datos que estamos trabajando con el comando “USE nombre\_database”, de esta manera empezaremos a hacer tanto consultas como transformaciones a la base de datos.

```
MariaDB [(none)]> USE bd_restaurant
Database changed
MariaDB [bd_restaurant]>
```

Ahora bien, podemos realizar las consultas de los datos de las tablas para cerciorarnos que estas tengan los atributos que estamos utilizando.

```
DESCRIBE restaurante;
DESCRIBE persona;
DESCRIBE tipopersonaxpersona;
DESCRIBE tipopersona;
DESCRIBE chef;
DESCRIBE menus;
DESCRIBE recetas;
DESCRIBE porcionxreceta;
DESCRIBE porciones;
DESCRIBE ingredientesxporcion;
DESCRIBE ingredientes;
DESCRIBE acompxreceta;
DESCRIBE acompanamientos;
DESCRIBE tipoacompanamiento;
```

Su resultado sería el siguiente:

```
MariaDB [bd_restaurant]> SHOW FULL tables FROM bd_restaurant
-> ;
```

Tables_in_bd_restaurant	Table_type
acompanamientos	BASE TABLE
acompxreceta	BASE TABLE
chef	BASE TABLE
ingredientes	BASE TABLE
ingredientesxporcion	BASE TABLE
menus	BASE TABLE
persona	BASE TABLE
porciones	BASE TABLE
porcionxreceta	BASE TABLE
recetas	BASE TABLE
restaurante	BASE TABLE
tipoacompanamiento	BASE TABLE
tipopersona	BASE TABLE
tipopersonaxpersona	BASE TABLE

14 rows in set (0.192 sec)

```
MariaDB [bd_restaurant]> DESCRIBE restaurante;
```

Field	Type	Null	Key	Default	Extra
idRestaurantee	int(11)	NO	PRI	NULL	
direccion	varchar(100)	NO		NULL	
telefono	varchar(45)	NO		NULL	
nit	varchar(45)	NO		NULL	

4 rows in set (0.104 sec)

```
MariaDB [bd_restaurant]> DESCRIBE persona;
```

Field	Type	Null	Key	Default	Extra
idPersona	int(11)	NO	PRI	NULL	
nombre	varchar(45)	NO		NULL	
apellido	varchar(45)	NO		NULL	
descripcion	varchar(500)	NO		NULL	
edad	varchar(10)	NO		NULL	
Restaurante_idRestaurantee	int(11)	NO	MUL	NULL	

6 rows in set (0.007 sec)

```
MariaDB [bd_restaurant]> DESCRIBE tipopersonaxpersona;
```

Field	Type	Null	Key	Default	Extra
idTipopersonaxpersona	int(11)	NO	PRI	NULL	
idTipoPersona	int(11)	NO		NULL	
idPersona	int(11)	NO		NULL	
Persona_idPersona	int(11)	NO	MUL	NULL	
TipoPersona_idTipoPersona	int(11)	NO	MUL	NULL	

5 rows in set (0.007 sec)

```
MariaDB [bd_restaurant]> DESCRIBE tipopersona;
```

Field	Type	Null	Key	Default	Extra
idTipoPersona	int(11)	NO	PRI	NULL	
tipoPersona	varchar(45)	NO		NULL	

2 rows in set (0.160 sec)

```
MariaDB [bd_restaurant]> DESCRIBE chef;
```

Field	Type	Null	Key	Default	Extra
idChef	int(11)	NO	PRI	NULL	
paisOrigen	int(11)	NO		NULL	
recetasAutoria	int(11)	NO		NULL	
TipoPersona_idTipoPersona	int(11)	NO	MUL	NULL	

4 rows in set (0.006 sec)

```
MariaDB [bd_restaurant]> DESCRIBE menus;
```

Field	Type	Null	Key	Default	Extra
idMenu	int(11)	NO	PRI	NULL	
nombre	varchar(45)	NO		NULL	
codigo	varchar(45)	NO		NULL	
precio	int(11)	YES		NULL	
Restaurante_idRestaurantee	int(11)	NO	MUL	NULL	

5 rows in set (0.007 sec)

```
MariaDB [bd_restaurant]> DESCRIBE recetas;
```

Field	Type	Null	Key	Default	Extra
idReceta	int(11)	NO	PRI	NULL	
nombreReceta	varchar(100)	NO		NULL	
procedimiento	varchar(1000)	NO		NULL	
acompxRecomendado	varchar(45)	NO		NULL	
autorCocinero	int(11)	NO		NULL	
Menus_idMenu	int(11)	NO	MUL	NULL	
Chef_idChef	int(11)	NO	MUL	NULL	

7 rows in set (0.007 sec)

```
MariaDB [bd_restaurant]> DESCRIBE porcionxreceta;
```

Field	Type	Null	Key	Default	Extra
idPorcionReceta	int(11)	NO	PRI	NULL	
cantidadPorcion	int(11)	NO		NULL	
caloriasPorcion	int(11)	YES		NULL	
idReceta	int(11)	NO		NULL	
idPorcion	int(11)	NO		NULL	
Recetas_idReceta	int(11)	NO	MUL	NULL	
Porciones_idPorcion	int(11)	NO	MUL	NULL	

7 rows in set (0.006 sec)

```
MariaDB [bd_restaurant]> DESCRIBE porciones;
```

Field	Type	Null	Key	Default	Extra
idPorcion	int(11)	NO	PRI	NULL	
cantidadGramos	varchar(45)	NO		NULL	
ingredientes	varchar(50)	NO		NULL	

3 rows in set (0.117 sec)

```
MariaDB [bd_restaurant]> DESCRIBE ingredientesxporcion;
```

Field	Type	Null	Key	Default	Extra
idIngrexporcion	int(11)	NO	PRI	NULL	
caloriasX100gramos	int(11)	YES		NULL	
cantidadIngredientes	int(11)	NO		NULL	
idIngrediente	int(11)	NO		NULL	
idPorcion	int(11)	NO		NULL	
Porciones_idPorcion	int(11)	NO	MUL	NULL	
Ingredientes_idIngrediente	int(11)	NO	MUL	NULL	

7 rows in set (0.006 sec)

```
MariaDB [bd_restaurant]> DESCRIBE ingredientes;
```

Field	Type	Null	Key	Default	Extra
idIngrediente	int(11)	NO	PRI	NULL	
precio	int(11)	NO		NULL	
exitencias	int(11)	NO		NULL	
decripcion	varchar(500)	NO		NULL	

4 rows in set (0.116 sec)

```
MariaDB [bd_restaurant]> DESCRIBE acompxreceta;
```

Field	Type	Null	Key	Default	Extra
idAcompanamientoXreceta	int(11)	NO	PRI	NULL	
idReceta	int(11)	NO		NULL	
idAcompanamiento	int(11)	NO		NULL	
Recetas_idReceta	int(11)	NO	MUL	NULL	
Acompanamientos_idAcompanamiento	int(11)	NO	MUL	NULL	

5 rows in set (0.006 sec)



```
MariaDB [bd_restaurant]> DESCRIBE acompanamientos;
```

Field	Type	Null	Key	Default	Extra
idAcompanamiento	int(11)	NO	PRI	NULL	
anyo	varchar(10)	NO		NULL	
precio	int(11)	NO		NULL	
existencias	int(11)	NO		NULL	
unidadMedida	varchar(45)	NO		NULL	
nombreAcompanamiento	varchar(45)	NO		NULL	
TipoAcompanamiento_idTipoAcompanamiento	int(11)	NO	MUL	NULL	

```
7 rows in set (0.109 sec)
```

```
MariaDB [bd_restaurant]> DESCRIBE tipoacompanamiento;
```

Field	Type	Null	Key	Default	Extra
idTipoAcompanamiento	int(11)	NO	PRI	NULL	
tipoBebida	varchar(45)	NO		NULL	

```
2 rows in set (0.124 sec)
```

**AGREGAR CAMPO:** Ahora supongamos que, en nuestra base de datos, El Gerente del restaurante nos pide añadir un nuevo atributo a la tabla de Chef, el cual será Genero. Genero para facilitar la interacción podrá ser definido como carácter simple.

```
MariaDB [bd_restaurant]> DESCRIBE chef;
```

Field	Type	Null	Key	Default	Extra
idChef	int(11)	NO	PRI	NULL	
paisOrigen	int(11)	NO		NULL	
recetasAutoria	int(11)	NO		NULL	
TipoPersona_idTipoPersona	int(11)	NO	MUL	NULL	

```
4 rows in set (0.004 sec)
```

```
MariaDB [bd_restaurant]> ALTER TABLE chef ADD generoChef VARCHAR(25) AFTER paisOrigen;
```

```
Query OK, 0 rows affected (0.247 sec)
```

```
Records: 0 Duplicates: 0 Warnings: 0
```

```
MariaDB [bd_restaurant]> DESCRIBE chef;
```

Field	Type	Null	Key	Default	Extra
idChef	int(11)	NO	PRI	NULL	
paisOrigen	int(11)	NO		NULL	
generoChef	varchar(25)	YES		NULL	
recetasAutoria	int(11)	NO		NULL	
TipoPersona_idTipoPersona	int(11)	NO	MUL	NULL	

```
5 rows in set (0.003 sec)
```

El comando utilizado fue el siguiente:

**ALTER TABLE** chef **ADD** generoChef **VARCHAR(25)** **AFTER** paisOrigen;

Lo que traduce este comando sería algo como: transfórmame la tabla “chef” añadiéndome una nueva columna o atributo llamado “generoChef” el cual será de tipo “varchar” que tendrá “25 caracteres” y este después del atributo “paísOrigen”. Al no especificarle nada más, hará que la misma por defecto sea un campo que si pueda ser nulo y no sea una llave primaria.

**QUITAR UN CAMPO:** Ahora supongamos que, tras una queja de los empleados, la decisión del Gerente de añadir a la base de datos el campo de Genero, fue mal recibida, en su argumento defienden que los comensales no les debe importar que genero tiene la persona que prepare un platillo específico. Por tal motivo se nos pide borrar este campo nuevamente de la tabla “chef”.

```
MariaDB [bd_restaurant]> DESCRIBE chef;
```

Field	Type	Null	Key	Default	Extra
idChef	int(11)	NO	PRI	NULL	
paisOrigen	int(11)	NO		NULL	
generoChef	varchar(25)	YES		NULL	
recetasAutoria	int(11)	NO		NULL	
TipoPersona_idTipoPersona	int(11)	NO	MUL	NULL	

```
5 rows in set (0.004 sec)
```

```
MariaDB [bd_restaurant]> ALTER TABLE chef DROP COLUMN generoChef;
```

```
Query OK, 0 rows affected (0.226 sec)
```

```
Records: 0 Duplicates: 0 Warnings: 0
```

```
MariaDB [bd_restaurant]> DESCRIBE chef;
```

Field	Type	Null	Key	Default	Extra
idChef	int(11)	NO	PRI	NULL	
paisOrigen	int(11)	NO		NULL	
recetasAutoria	int(11)	NO		NULL	
TipoPersona_idTipoPersona	int(11)	NO	MUL	NULL	

```
4 rows in set (0.004 sec)
```

El comando utilizado es:

```
ALTER TABLE chef DROP COLUMN generoChef;
```

Lo que traduce este comando es algo como: transfórmame la tabla “chef” eliminando la columna “generoChef”.

**RENOMBRAR UN CAMPO:** trabajando con la base de datos nos damos cuenta que la llave foránea “TipoPersona\_idTipoPersona” posee un nombre muy largo para trabajar con ella y no es fácilmente identificable, por tal motivo se resuelve renombrar el campo.

```
MariaDB [bd_restaurant]> DESCRIBE chef;
+-----+-----+-----+-----+-----+-----+
| Field          | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idChef         | int(11)| NO   | PRI | NULL    |       |
| paisOrigen     | int(11)| NO   |     | NULL    |       |
| recetasAutoria | int(11)| NO   |     | NULL    |       |
| TipoPersona_idTipoPersona | int(11)| NO   | MUL | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.003 sec)

MariaDB [bd_restaurant]> ALTER TABLE chef CHANGE TipoPersona_idTipoPersona FK_TipoPersona INT;
Query OK, 0 rows affected (0.785 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [bd_restaurant]> DESCRIBE chef;
+-----+-----+-----+-----+-----+-----+
| Field          | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idChef         | int(11)| NO   | PRI | NULL    |       |
| paisOrigen     | int(11)| NO   |     | NULL    |       |
| recetasAutoria | int(11)| NO   |     | NULL    |       |
| FK_TipoPersona | int(11)| YES  | MUL | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.003 sec)
```

El comando utilizado es:

```
ALTER TABLE chef CHANGE TipoPersona_idTipoPersona FK_TipoPersona INT;
```

Este anterior comando traduce algo como: Transformame una tabla con nombre “chef” y cambia el atributo “TipoPersona\_idTipoPersona” por (marcado por un espacio en el comando) “FK\_TipoPersona” que será de tipo “INT”, este reemplazo se dará en la misma columna sin afectar los datos ya existentes en la misma, manteniendo la ubicación. Si no es dado otro aspecto más, se tomará por defecto el estado de “Key” previamente establecido y dejando que si puedan existir datos nulos.

**CAMBIAR EL TIPO DE DATO A UNO EXISTENTE:** con el ultimo cambio que hemos realizado, hemos dejado la cualidad a “FK\_TipoPersona” de contener datos nulos, este aspecto no es requerido en nuestra base de datos pues una persona registrada dentro del restaurante debe poderse identificar dentro de uno de los diferentes roles en la empresa, sea cual sea. Por tal motivo es necesario cambiar el tipo de dato de este atributo.

```
MariaDB [bd_restaurant]> DESCRIBE chef;
+-----+-----+-----+-----+-----+-----+
| Field          | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idChef         | int(11)| NO   | PRI | NULL    |       |
| paisOrigen     | int(11)| NO   |     | NULL    |       |
| recetasAutoria | int(11)| NO   |     | NULL    |       |
| FK_TipoPersona | int(11)| YES  | MUL | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.098 sec)

MariaDB [bd_restaurant]> ALTER TABLE chef MODIFY FK_TipoPersona INT NOT NULL;
Query OK, 0 rows affected (0.710 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [bd_restaurant]> DESCRIBE chef;
+-----+-----+-----+-----+-----+-----+
| Field          | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idChef         | int(11)| NO   | PRI | NULL    |       |
| paisOrigen     | int(11)| NO   |     | NULL    |       |
| recetasAutoria | int(11)| NO   |     | NULL    |       |
| FK_TipoPersona | int(11)| NO   | MUL | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.085 sec)
```



El comando utilizado fue:

```
ALTER TABLE chef MODIFY FK_TipoPersona INT NOT NULL;
```

Este comando traducido dicta algo como: transfórmame la tabla denominada “chef” modificando el atributo “FK\_TipoPersona” haciendo que sea de tipo “INT” y no pueda albergar datos nulos.

**CREAR UN ÍNDICE:** El Gerente del restaurante solicita que podamos encontrar los datos de los cocineros de manera más eficiente, para la forma más sencilla es asignar un número de identificación único para cada persona, esto se logra manteniendo un índice para la tabla de chef.

```
MariaDB [bd_restaurant]> DESCRIBE chef;
+-----+-----+-----+-----+-----+-----+
| Field          | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idChef         | int(11)| NO   | PRI | NULL    |       |
| paisOrigen     | int(11)| NO   |     | NULL    |       |
| recetasAutoria | int(11)| NO   |     | NULL    |       |
| FK_TipoPersona | int(11)| NO   | MUL | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.085 sec)

MariaDB [bd_restaurant]> ALTER TABLE chef ADD INDEX (idChef);
Query OK, 0 rows affected (0.239 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [bd_restaurant]> DESCRIBE chef;
+-----+-----+-----+-----+-----+-----+
| Field          | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idChef         | int(11)| NO   | PRI | NULL    |       |
| paisOrigen     | int(11)| NO   |     | NULL    |       |
| recetasAutoria | int(11)| NO   |     | NULL    |       |
| FK_TipoPersona | int(11)| NO   | MUL | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.040 sec)
```

El comando utilizado es:

```
ALTER TABLE chef ADD INDEX (idChef);
```

Este comando añade un índice que crea un código único ordenado por cada dato ingresado al momento de poblar la tabla, esto generalmente se hace con la columna que es la llave primaria de la tabla.

El comando traducido se da como: Transfórmame una tabla llamada “chef” y añádeme un índice en la columna “idChef”.

Ahora bien, crearemos una tabla flotante dentro de la base de datos para hacer ejercicios de llave primaria (crear y eliminar) y ejercicios de llave foránea (crear y eliminar). Las tablas actuales son las siguientes:

```
MariaDB [bd_restaurant]> SHOW FULL tables FROM bd_restaurant;
+-----+-----+
| Tables_in_bd_restaurant | Table_type |
+-----+-----+
| acompanamientos         | BASE TABLE |
| acompxreceta             | BASE TABLE |
| chef                     | BASE TABLE |
| ingredientes             | BASE TABLE |
| ingredientesxporcion     | BASE TABLE |
| menus                    | BASE TABLE |
| persona                  | BASE TABLE |
| porciones                | BASE TABLE |
| porcionxreceta           | BASE TABLE |
| recetas                  | BASE TABLE |
| restaurante              | BASE TABLE |
| tipoacompanamiento       | BASE TABLE |
| tipopersona              | BASE TABLE |
| tipopersonaxpersona      | BASE TABLE |
+-----+-----+
14 rows in set (0.001 sec)
```

**CREAR NUEVA TABLA:** CREATE TABLE tabla\_ejemplo ( atributo\_Uno INT, atributo\_DOS INT );

```
MariaDB [bd_restaurant]> CREATE TABLE tabla_ejemplo (
-> atributo_Uno INT,
-> atributo_DOS INT
-> );
Query OK, 0 rows affected (0.296 sec)

MariaDB [bd_restaurant]> SHOW FULL tables FROM bd_restaurant;
+-----+-----+
| Tables_in_bd_restaurant | Table_type |
+-----+-----+
| acompanamientos         | BASE TABLE |
| acompxreceta             | BASE TABLE |
| chef                     | BASE TABLE |
| ingredientes             | BASE TABLE |
| ingredientesxporcion     | BASE TABLE |
| menus                    | BASE TABLE |
| persona                  | BASE TABLE |
| porciones                | BASE TABLE |
| porcionxreceta           | BASE TABLE |
| recetas                  | BASE TABLE |
| restaurante              | BASE TABLE |
| tabla_ejemplo            | BASE TABLE |
| tipoacompanamiento       | BASE TABLE |
| tipopersona              | BASE TABLE |
| tipopersonaxpersona      | BASE TABLE |
+-----+-----+
15 rows in set (0.001 sec)
```

**CREAR LLAVE PRIMARIA:** La tabla como podemos ver a continuación, solo posee dos atributos no una llave primaria, por tanto, debemos crear una.

```
MariaDB [bd_restaurant]> DESCRIBE tabla_ejemplo;
+-----+-----+-----+-----+-----+-----+
| Field      | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| atributo_Uno | int(11) | YES  |     | NULL    |       |
| atributo_DOS | int(11) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.004 sec)

MariaDB [bd_restaurant]> ALTER TABLE tabla_ejemplo ADD idEjemplo INT NOT NULL;
Query OK, 0 rows affected (0.133 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [bd_restaurant]> ALTER TABLE tabla_ejemplo ADD PRIMARY KEY (idEjemplo);
Query OK, 0 rows affected (0.696 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [bd_restaurant]> DESCRIBE tabla_ejemplo;
+-----+-----+-----+-----+-----+-----+
| Field      | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| atributo_Uno | int(11) | YES  |     | NULL    |       |
| atributo_DOS | int(11) | YES  |     | NULL    |       |
| idEjemplo    | int(11) | NO   | PRI | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.024 sec)
```

Aquí se utilizaron dos comandos, uno para crear el atributo y otro para que este sea una llave primaria o PRIMARY KEY.

```
ALTER TABLE tabla_ejemplo ADD idEjemplo INT NOT NULL;
ALTER TABLE tabla_ejemplo ADD PRIMARY KEY (idEjemplo);
```

**BORRAR LLAVE PRIMARIA:** como observamos, se cometió un “error”, realmente no es necesario que la llave primaria este en una determinada posición dentro de la tabla, sin embargo, por convenciones de trabajado con las bases de datos, como una regla no escrita, la llave primaria se deja como el primer atributo dentro de una tabla. Por tanto, podemos eliminar el campo previamente creado como la llave primaria y volverlo a crear cerciorándonos que tenga la posición inicial.

```

MariaDB [bd_restaurant]> DESCRIBE tabla_ejemplo;
+-----+-----+-----+-----+-----+-----+
| Field      | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| atributo_Uno | int(11) | YES  |     | NULL    |       |
| atributo_DOS | int(11) | YES  |     | NULL    |       |
| idEjemplo   | int(11) | NO   | PRI | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.024 sec)

MariaDB [bd_restaurant]> ALTER TABLE tabla_ejemplo DROP PRIMARY KEY;
Query OK, 0 rows affected (0.987 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [bd_restaurant]> DESCRIBE tabla_ejemplo;
+-----+-----+-----+-----+-----+-----+
| Field      | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| atributo_Uno | int(11) | YES  |     | NULL    |       |
| atributo_DOS | int(11) | YES  |     | NULL    |       |
| idEjemplo   | int(11) | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.007 sec)

MariaDB [bd_restaurant]> ALTER TABLE tabla_ejemplo DROP COLUMN idEjemplo;
Query OK, 0 rows affected (0.160 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [bd_restaurant]> DESCRIBE tabla_ejemplo;
+-----+-----+-----+-----+-----+-----+
| Field      | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| atributo_Uno | int(11) | YES  |     | NULL    |       |
| atributo_DOS | int(11) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.004 sec)

```

El primer comando utilizado quita la etiqueta que identifica un campo como llave primaria, el segundo simplemente elimina el atributo de la tabla.

**ALTER TABLE** tabla\_ejemplo **DROP PRIMARY KEY**;  
**ALTER TABLE** tabla\_ejemplo **DROP COLUMN** idEjemplo;

Para crear de nuevo el campo, pero ahora que este en primer lugar utilizaremos los siguientes comandos:

**ALTER TABLE** tabla\_ejemplo **ADD** idEjemplo **INT NOT NULL FIRST**;  
**ALTER TABLE** tabla\_ejemplo **ADD PRIMARY KEY** (idEjemplo);

```

MariaDB [bd_restaurant]> DESCRIBE tabla_ejemplo;
+-----+-----+-----+-----+-----+-----+
| Field      | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| atributo_Uno | int(11) | YES  |     | NULL    |       |
| atributo_DOS | int(11) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.003 sec)

MariaDB [bd_restaurant]> ALTER TABLE tabla_ejemplo ADD idEjemplo INT NOT NULL FIRST;
Query OK, 0 rows affected (0.193 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [bd_restaurant]> ALTER TABLE tabla_ejemplo ADD PRIMARY KEY (idEjemplo);
Query OK, 0 rows affected (0.634 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [bd_restaurant]> DESCRIBE tabla_ejemplo;
+-----+-----+-----+-----+-----+-----+
| Field      | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idEjemplo   | int(11) | NO   | PRI | NULL    |       |
| atributo_Uno | int(11) | YES  |     | NULL    |       |
| atributo_DOS | int(11) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.003 sec)

```

**RENOMBRAR TABLA:** Para facilitar un poco más el manejo de la tabla podemos renombrarla. Esto lo que hace es crear una nueva tabla con un “CREATE TABLE” con el nombre de la tabla que hemos dado, con los atributos de la tabla original de la que estamos solicitando y movemos todos los datos hacia la nueva tabla. La tabla original se elimina y queda la nueva reemplazando la anterior.

**RENAME TABLE** tabla\_ejemplo **TO** ejemplos;

```
MariaDB [bd_restaurant]> DESCRIBE tabla_ejemplo;
+-----+-----+-----+-----+-----+-----+
| Field      | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idEjemplo  | int(11) | NO   | PRI | NULL    |       |
| atributo_Uno | int(11) | YES  |     | NULL    |       |
| atributo_DOS | int(11) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.003 sec)
```

```
MariaDB [bd_restaurant]> RENAME TABLE tabla_ejemplo TO ejemplos;
Query OK, 0 rows affected (0.232 sec)
```

```
MariaDB [bd_restaurant]> DESCRIBE ejemplos;
+-----+-----+-----+-----+-----+-----+
| Field      | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idEjemplo  | int(11) | NO   | PRI | NULL    |       |
| atributo_Uno | int(11) | YES  |     | NULL    |       |
| atributo_DOS | int(11) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.008 sec)
```

**CREAR LLAVES FORANEAS:** las llaves foráneas nos permiten traer ciertos datos que están enlazados con la información que tenemos en una tabla que por su naturaleza propia pertenecen a otra tabla.

```
MariaDB [bd_restaurant]> DESCRIBE ejemplos;
+-----+-----+-----+-----+-----+-----+
| Field      | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idEjemplo  | int(11) | NO   | PRI | NULL    |       |
| atributo_Uno | int(11) | YES  |     | NULL    |       |
| atributo_DOS | int(11) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.004 sec)
```

```
MariaDB [bd_restaurant]> ALTER TABLE ejemplos ADD FK_IdChef INT NOT NULL;
Query OK, 0 rows affected (0.156 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
MariaDB [bd_restaurant]> ALTER TABLE ejemplos
-> ADD CONSTRAINT FK_IdChef
-> FOREIGN KEY (FK_IdChef)
-> REFERENCES chef(idChef)
-> ON UPDATE CASCADE
-> ON DELETE RESTRICT;
Query OK, 0 rows affected (1.090 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
MariaDB [bd_restaurant]> DESCRIBE ejemplos;
+-----+-----+-----+-----+-----+-----+
| Field      | Type  | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| idEjemplo  | int(11) | NO   | PRI | NULL    |       |
| atributo_Uno | int(11) | YES  |     | NULL    |       |
| atributo_DOS | int(11) | YES  |     | NULL    |       |
| FK_IdChef  | int(11) | NO   | MUL | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.006 sec)
```

Los comandos utilizados son dos, el primero que crea el campo donde se destinara la llave foránea y el segundo corresponde a toda la configuración de la llave foránea.

**ALTER TABLE ejemplos ADD FK\_IdChef INT NOT NULL;**

El siguiente código se traduce como: Transfórmame una tabla llamada “ejemplos” en ella añádeme una restricción que se denominará “FK\_IdChef” que será de tipo llave foránea, la misma estará en la columna “FK\_IdChef” de la tabla “ejemplos” y tendrá como referencia los datos de la columna “idChef” de la tabla “Chef”. El comportamiento de actualización de datos se dará en cascada a partir de las hijas de la tabla “ejemplos” y en caso de ser eliminada su comportamiento será “restrictivo”.

```
ALTER TABLE ejemplos
ADD CONSTRAINT FK_IdChef
FOREIGN KEY (FK_IdChef)
REFERENCES chef(idChef)
ON UPDATE CASCADE
ON DELETE RESTRICT;
```

**ELIMINAR TABLA:** De esta manera ahora tenemos las siguientes tablas:

```
MariaDB [bd_restaurant]> SHOW FULL tables FROM bd_restaurant
-> ;
```

Tables_in_bd_restaurant	Table_type
acompanamientos	BASE TABLE
acompxreceta	BASE TABLE
chef	BASE TABLE
ejemplos	BASE TABLE
ingredientes	BASE TABLE
ingredientesxporcion	BASE TABLE
menus	BASE TABLE
persona	BASE TABLE
porciones	BASE TABLE
porcionxreceta	BASE TABLE
recetas	BASE TABLE
restaurante	BASE TABLE
tipoacompanamiento	BASE TABLE
tipopersona	BASE TABLE
tipopersonaxpersona	BASE TABLE

```
15 rows in set (0.002 sec)
```

Volveremos a su estado original eliminando la tabla “ejemplos”.

**DROP TABLE** ejemplos;  
**SHOW FULL tables FROM** bd\_restaurant;

Tables_in_bd_restaurant	Table_type
acompanamientos	BASE TABLE
acompxreceta	BASE TABLE
chef	BASE TABLE
ejemplos	BASE TABLE
ingredientes	BASE TABLE
ingredientesxporcion	BASE TABLE
menus	BASE TABLE
persona	BASE TABLE
porciones	BASE TABLE
porcionxreceta	BASE TABLE
recetas	BASE TABLE
restaurante	BASE TABLE
tipoacompanamiento	BASE TABLE
tipopersona	BASE TABLE
tipopersonaxpersona	BASE TABLE

```
15 rows in set (0.001 sec)
```

```
MariaDB [bd_restaurant]> DROP TABLE ejemplos;
Query OK, 0 rows affected (0.226 sec)
```

```
MariaDB [bd_restaurant]> SHOW FULL tables FROM bd_restaurant;
```

Tables_in_bd_restaurant	Table_type
acompanamientos	BASE TABLE
acompxreceta	BASE TABLE
chef	BASE TABLE
ingredientes	BASE TABLE
ingredientesxporcion	BASE TABLE
menus	BASE TABLE
persona	BASE TABLE
porciones	BASE TABLE
porcionxreceta	BASE TABLE
recetas	BASE TABLE
restaurante	BASE TABLE
tipoacompanamiento	BASE TABLE
tipopersona	BASE TABLE
tipopersonaxpersona	BASE TABLE

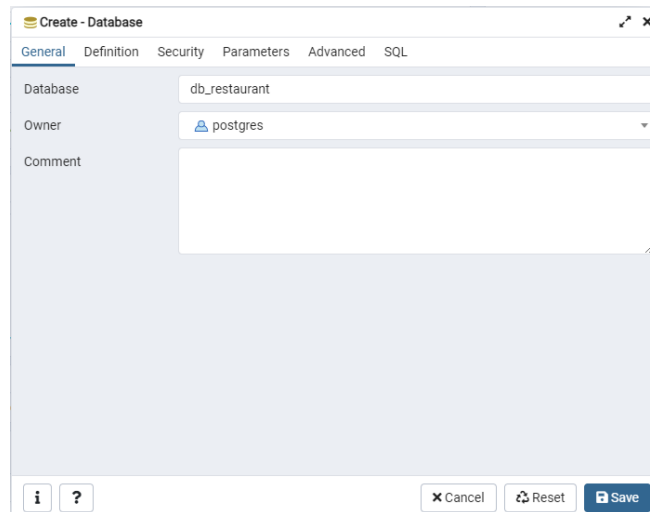
```
14 rows in set (0.001 sec)
```

## PostgreSQL PgAdmin 4.

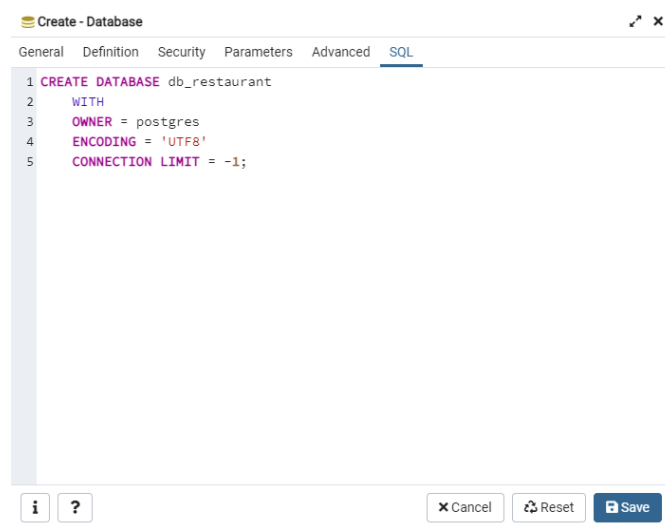
Ahora bien, si en cambio yo quisiera trabajar en PostgreSQL PGAdmin 4, tendría que crear la database. Lo primero que debemos hacer es dirigirnos al servidor que tenemos abierto, colocado en el menú lateral de la pantalla, seleccionamos “PostgreSQL 13”, dando clic derecho en el aparecerá un menú de tareas vertical donde podremos ubicar la opción “create” y en ella la opción de “Database”, la cual seleccionaremos al final.

Dado lo último se desplegará como aparece a continuación, una ventana donde podremos configurar esta base de datos. Para hacer mas sencillo este procedimiento, solo ubicaremos el nombre de la base de datos y el usuario que se encargara de ella, esto dentro de la pestaña “General”.

El nombre de la base de datos figura como un espacio continuo del atributo “Database” y el usuario que elegiremos para delegársela esta en el atributo de “Owner”. En este ultimo paso debemos tener cuidado en que no es recomendable que utilicemos el super usuario o el usuario administrador de PostgreSQL denominado “postgres”, con la finalidad de que configuraciones dadas específicamente para la base de datos en cuestión, no alteren el funcionamiento general de PostgreSQL.

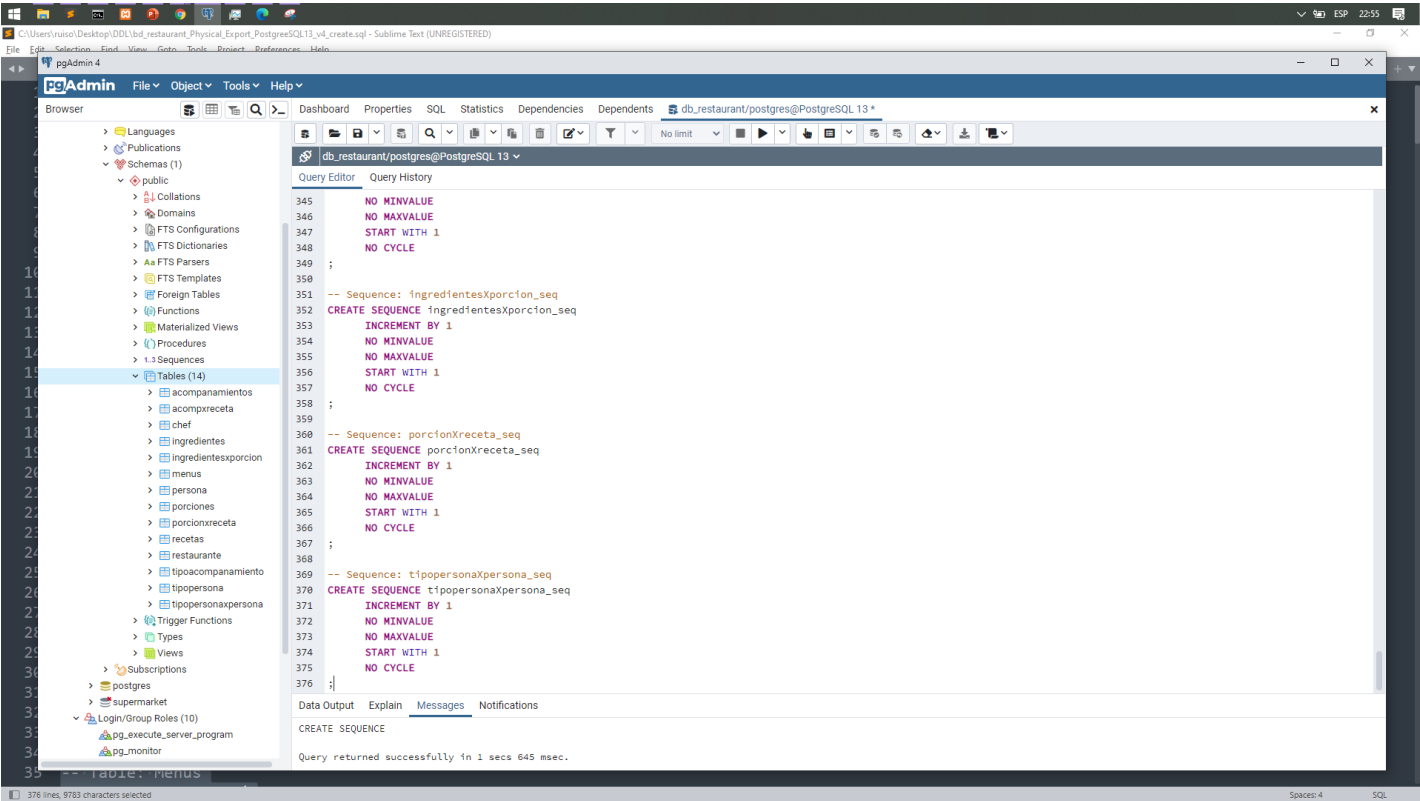


También podemos visualizar que en la pestaña “SQL” se nos dará el código SQL para crear esta misma base de datos que estamos haciendo por el método de GUI y que es internamente lo que ejecutara PgAdmin 4.



Una vez creada la base de datos debemos entrar a la barra de menús, el menú Tool y seleccionar la herramienta Query, en ella podremos traspasar las sentencias SQL que crearan nuestra base de datos. Las sentencias SQL las podemos hallar en el interior de nuestro archivo generado por Vertabelo, basta solo con abrirlo con cualquier editor de texto o programa que

trabaje con texto plano tal como “Bloc de Notas”. Pasaremos todas las instrucciones al espacio Query y las ejecutaremos, esperando que en el espacio inmediatamente inferior no salga ningún problema.



Podremos observar también de la anterior imagen que, en la parte lateral en vertical, en la sección de “Schemas” y las “tables”, estarán todas las entidades que hemos creado anteriormente. Exactamente los mismos ejemplos que desarrollamos en MySQL los desarrollaremos en PostgreSQL. El espacio de comandos aquí será el mismo dado dentro de la GUI de PgAdmin, la opción de Query que antes vimos, dentro de ella podemos hacer todas las consultas y transformación DDL y DML que queramos para este ejercicio.

**MOSTRAR TODAS LAS TABLAS:** Aquí es donde algunos comandos serán distintos entre MySQL y PostgreSQL, aunque en la mayoría son pequeños arreglos de sintaxis lo que diferenciara el SQL entre ambos motores de base de datos, pero en general es casi lo mismo en ambos.

```
1 SELECT *
2 FROM pg_catalog.pg_tables
3 WHERE schemaname != 'pg_catalog' AND
4       schemaname != 'information_schema';
```

	schemaname name	tablename name	tableowner name	tablespace name	hasindexes boolean	hasrules boolean	hastriggers boolean	rowsecurity boolean
1	public	tipoacompanamiento	postgres	[null]	true	false	true	false
2	public	acompanamientos	postgres	[null]	true	false	true	false
3	public	tipopersona	postgres	[null]	true	false	true	false
4	public	chef	postgres	[null]	true	false	true	false
5	public	restaurante	postgres	[null]	true	false	true	false
6	public	menus	postgres	[null]	true	false	true	false
7	public	persona	postgres	[null]	true	false	true	false
8	public	recetas	postgres	[null]	true	false	true	false
9	public	acompxreceta	postgres	[null]	true	false	true	false
10	public	ingredientes	postgres	[null]	true	false	true	false
11	public	ingredientesporcion	postgres	[null]	true	false	true	false
12	public	porciones	postgres	[null]	true	false	true	false
13	public	porcionxreceta	postgres	[null]	true	false	true	false
14	public	tipopersonaxpersona	postgres	[null]	true	false	true	false



```

SELECT *
FROM pg_catalog.pg_tables
WHERE schemaname != 'pg_catalog' AND schemaname != 'information_schema';

```

## MOSTRAR ATRIBUTOS DE LAS TABLAS:

1

SELECT table\_name, column\_name, data\_type

2

FROM information\_schema.columns

3

WHERE table\_name = 'restaurante';

Data Output

Explain

Messages

Notifications

	table_name name	column_name name	data_type character varying
1	restaurante	idrestaurantee	integer
2	restaurante	direccion	character varying
3	restaurante	telefono	character varying
4	restaurante	nit	character varying

1

SELECT table\_name, column\_name, data\_type

2

FROM information\_schema.columns

3

WHERE table\_name = 'persona';

Data Output

Explain

Messages

Notifications

	table_name name	column_name name	data_type character varying
1	persona	idpersona	integer
2	persona	nombre	character varying
3	persona	apellido	character varying
4	persona	descripcion	character varying
5	persona	edad	character varying
6	persona	restaurante_idrestaurantee	integer

1

SELECT table\_name, column\_name, data\_type

2

FROM information\_schema.columns

3

WHERE table\_name = 'tipopersonaxpersona';

Data Output

Explain

Messages

Notifications

	table_name name	column_name name	data_type character varying
1	tipopersonaxpersona	idtipopersonaxpersona	integer
2	tipopersonaxpersona	idtipopersona	integer
3	tipopersonaxpersona	idpersona	integer
4	tipopersonaxpersona	persona_idpersona	integer
5	tipopersonaxpersona	tipopersona_idtipopersona	integer

1

SELECT

table\_name,

column\_name,

data\_type

2

FROM

information\_schema.columns

3

WHERE

table\_name = 'tipopersona';

Data Output

Explain

Messages

Notifications

	table_name name	column_name name	data_type character varying
1	tipopersona	idtipopersona	integer
2	tipopersona	tipopersona	character varying

1

SELECT

table\_name,

column\_name,

data\_type

2

FROM

information\_schema.columns

3

WHERE

table\_name = 'chef';

Data Output

Explain

Messages

Notifications

	table_name name	column_name name	data_type character varying
1	chef	idchef	integer
2	chef	paisorigen	integer
3	chef	recetasautoria	integer
4	chef	tipopersona_idtipopersona	integer

1

SELECT

table\_name,

column\_name,

data\_type

2

FROM

information\_schema.columns

3

WHERE

table\_name = 'menus';

Data Output

Explain

Messages

Notifications

	table_name name	column_name name	data_type character varying
1	menus	idmenus	integer
2	menus	nombre	character varying
3	menus	codigo	character varying
4	menus	precio	integer
5	menus	restaurante_idrestaurantee	integer

```

1 SELECT table_name, column_name, data_type
2 FROM information_schema.columns
3 WHERE table_name = 'recetas';

```

Data Output	Explain	Messages	Notifications
table_name name	column_name name	data_type character varying	
1	recetas	idreceta	integer
2	recetas	nombrereceta	character varying
3	recetas	procedimiento	character varying
4	recetas	acomprecomendado	character varying
5	recetas	autorcocinero	integer
6	recetas	menus_idmenus	integer
7	recetas	chef_idchef	integer

```

1 SELECT table_name, column_name, data_type
2 FROM information_schema.columns
3 WHERE table_name = 'porcionxreceta';

```

Data Output	Explain	Messages	Notifications
table_name name	column_name name	data_type character varying	
1	porcionxreceta	idporcionreceta	integer
2	porcionxreceta	cantidadporcion	integer
3	porcionxreceta	caloriasporcion	integer
4	porcionxreceta	idreceta	integer
5	porcionxreceta	idporcion	integer
6	porcionxreceta	recetas_idreceta	integer
7	porcionxreceta	porciones_idporcion	integer

```

1 SELECT table_name, column_name, data_type
2 FROM information_schema.columns
3 WHERE table_name = 'porciones';

```

Data Output	Explain	Messages	Notifications
table_name name	column_name name	data_type character varying	
1	porciones	idporcion	integer
2	porciones	cantidadgramos	character varying
3	porciones	ingredientes	character varying

```

1 SELECT table_name, column_name, data_type
2 FROM information_schema.columns
3 WHERE table_name = 'ingredientesxporcion';

```

Data Output	Explain	Messages	Notifications
table_name name	column_name name	data_type character varying	
1	ingredientesxporcion	idingrexporcion	integer
2	ingredientesxporcion	caloriasx100gramos	integer
3	ingredientesxporcion	cantidadingredientes	integer
4	ingredientesxporcion	idingrediente	integer
5	ingredientesxporcion	idporcion	integer
6	ingredientesxporcion	porciones_idporcion	integer
7	ingredientesxporcion	ingredientes_idingrediente	integer

```

1 SELECT table_name, column_name, data_type
2 FROM information_schema.columns
3 WHERE table_name = 'ingredientes';

```

Data Output	Explain	Messages	Notifications
table_name name	column_name name	data_type character varying	
1	ingredientes	idingrediente	integer
2	ingredientes	precio	integer
3	ingredientes	existencias	integer
4	ingredientes	decripcion	character varying

```

1 SELECT table_name, column_name, data_type
2 FROM information_schema.columns
3 WHERE table_name = 'acompxreceta';

```

Data Output	Explain	Messages	Notifications
table_name name	column_name name	data_type character varying	
1	acompxreceta	idacompanamientoxreceta	integer
2	acompxreceta	idreceta	integer
3	acompxreceta	idacompanamiento	integer
4	acompxreceta	recetas_idreceta	integer
5	acompxreceta	acompanamientos_idacompanamiento	integer

```

1 SELECT table_name, column_name, data_type
2 FROM information_schema.columns
3 WHERE table_name = 'acompanamientos';

```

Data Output	Explain	Messages	Notifications
table_name name	column_name name	data_type character varying	
1	acompanamientos	idacompanamiento	integer
2	acompanamientos	anyo	character varying
3	acompanamientos	precio	integer
4	acompanamientos	existencias	integer
5	acompanamientos	unidadmedida	character varying
6	acompanamientos	nombreakompanamiento	character varying
7	acompanamientos	tipocompanamiento_idtipocompanamiento	integer

```

1 SELECT table_name, column_name, data_type
2 FROM information_schema.columns
3 WHERE table_name = 'tipoacompanamiento';

```

Data Output Explain Messages Notifications

	table_name name	column_name name	data_type character varying
1	tipoacompanamiento	idtipoacompanamiento	integer
2	tipoacompanamiento	tipobebida	character varying

### AGREGAR UN CAMPO:

```

1 ALTER TABLE chef ADD COLUMN generoChef VARCHAR(25);
2 SELECT * FROM chef;

```

Data Output Explain Messages Notifications

	idchef [PK] integer	paisorigen integer	recetasautoria integer	tipopersona_idtipopersona integer	generochef character varying (25)

El comando utilizado fue:

```
ALTER TABLE chef ADD COLUMN generoChef VARCHAR(25);
```

### QUITAR UN CAMPO:

```

1 ALTER TABLE chef DROP COLUMN generoChef;
2 SELECT * FROM chef;

```

Data Output Explain Messages Notifications

	idchef [PK] integer	paisorigen integer	recetasautoria integer	tipopersona_idtipopersona integer

El comando utilizado fue:

```
ALTER TABLE chef DROP COLUMN generoChef;
```

### RENOMBRAR UN CAMPO:

```

1 ALTER TABLE chef RENAME COLUMN TipoPersona_idTipoPersona TO FK_TipoPersona;
2 SELECT * FROM chef;

```

Data Output Explain Messages Notifications









	idchef [PK] integer	paisorigen integer	recetasautoria integer	fk_tipopersona integer

El comando utilizado fue:

```
ALTER TABLE chef RENAME COLUMN TipoPersona_idTipoPersona TO FK_TipoPersona;
```

## CAMBIAR EL TIPO DE DATO A UNO EXISTENTE:

```
1 ALTER TABLE chef ALTER COLUMN FK_TipoPersona SET DATA TYPE INT;  
2 SELECT * FROM chef;
```









Data Output	Explain	Messages	Notifications
 idchef [PK] integer 	 paisorigen integer 	 recetasautoria integer 	 fk_tipopersona integer 

El comando utilizado fue:

```
ALTER TABLE chef ALTER COLUMN FK_TipoPersona SET DATA TYPE INT;
```

## CREAR UN ÍNDICE:

```
1 CREATE INDEX ON chef (idChef);  
2 SELECT * FROM chef;
```

Data Output	Explain	Messages	Notifications
 idchef [PK] integer 	 paisorigen integer 	 recetasautoria integer 	 fk_tipopersona integer 

El comando utilizado fue:

```
CREATE INDEX ON chef (idChef);
```

## CREAR UNA TABLA NUEVA:

Tables (15)

> acompanamientos

> acompxreceta

> chef

> ingredientes

> ingredientesxporcion

> menus

> persona

> porciones

> porcionxreceta

> recetas

> restaurante

> **tabla\_ejemplo**

> tipoacompanamiento

> tipopersona

> tipopersonaxpersona

```
1 CREATE TABLE tabla_ejemplo
2 (
3     atributo_Uno INT NOT NULL,
4     atributo_DOS INT NOT NULL
5 );
6 SELECT * FROM tabla_ejemplo;
```

Data Output

Explain

Messages

Notifications

atributo\_uno

integer

atributo\_dos

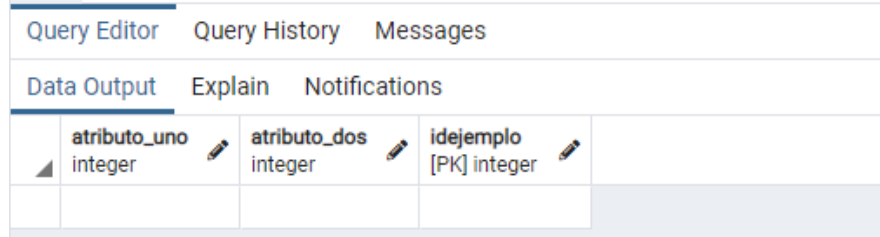
integer

El comando utilizado fue:

```
CREATE TABLE tabla_ejemplo (atributo_Uno INT NOT NULL, atributo_DOS INT NOT NULL);
```

### CREAR LLAVE PRIMARIA:

```
1 ALTER TABLE tabla_ejemplo ADD COLUMN idEjemplo INT;  
2 ALTER TABLE tabla_ejemplo ADD PRIMARY KEY (idEjemplo);  
3 SELECT * FROM tabla_ejemplo;
```

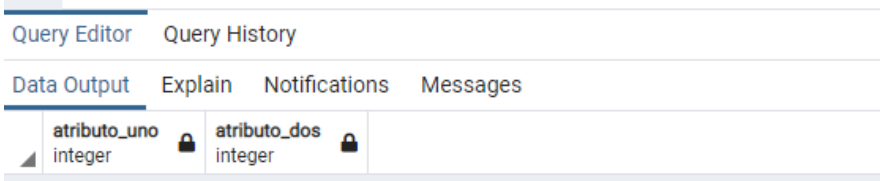


Los comandos utilizados fueron:

```
ALTER TABLE tabla_ejemplo ADD COLUMN idEjemplo INT;  
ALTER TABLE tabla_ejemplo ADD PRIMARY KEY (idEjemplo);
```

### BORRAR LLAVE PRIMARIA:

```
1 ALTER TABLE tabla_ejemplo DROP CONSTRAINT tabla_ejemplo_pkey ;  
2 ALTER TABLE tabla_ejemplo DROP COLUMN idEjemplo;  
3 SELECT * FROM tabla_ejemplo;
```

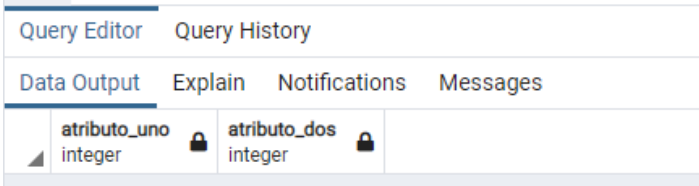


Los comandos utilizados fueron:

```
ALTER TABLE tabla_ejemplo DROP CONSTRAINT tabla_ejemplo_pkey ;  
ALTER TABLE tabla_ejemplo DROP COLUMN idEjemplo;
```

### RENOMBRER TABLA:

```
1 ALTER TABLE tabla_ejemplo RENAME TO ejemplos;  
2 SELECT * FROM ejemplos;
```







El comando utilizado es:

```
ALTER TABLE tabla_ejemplo RENAME TO ejemplos;
```

## CREAR LLAVE FORANEA:

```
1 ALTER TABLE ejemplos ADD COLUMN FK_Ejemplo INT;
2
3 ALTER TABLE ejemplos ADD CONSTRAINT FK_Ejemplo
4 FOREIGN KEY (FK_Ejemplo)
5 REFERENCES chef (idChef)
6 ON UPDATE CASCADE
7 ON DELETE RESTRICT;
8 ;
9 SELECT * FROM ejemplos;
```

Query Editor	Query History		
Data Output	Explain	Notifications	Messages
 atributo_uno integer	 atributo_dos integer	 fk_ejemplo integer	




Los comandos utilizados fueron:

```
ALTER TABLE ejemplos ADD COLUMN FK_Ejemplo INT;
```

```
ALTER TABLE ejemplos ADD CONSTRAINT FK_Ejemplo
FOREIGN KEY (FK_Ejemplo)
REFERENCES chef (idChef)
ON UPDATE CASCADE
ON DELETE RESTRICT;
```

## ELIMINAR LLAVE FORANEA:

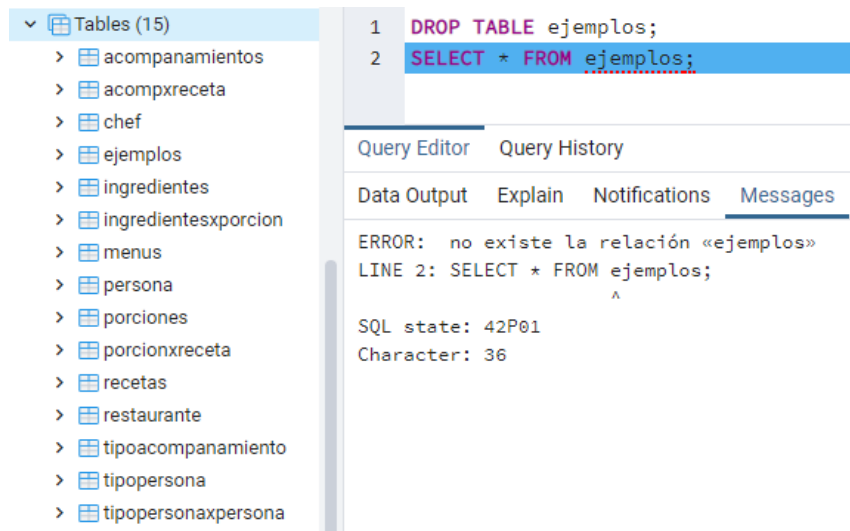
```
1 ALTER TABLE ejemplos DROP CONSTRAINT FK_ejemplo;
2 ALTER TABLE ejemplos DROP COLUMN FK_Ejemplo;
3 SELECT * FROM ejemplos;
```

Query Editor		Query History		
Data Output		Explain	Notifications	Messages
 atributo_uno integer	 atributo_dos integer			

Los comandos utilizados fueron:

```
ALTER TABLE ejemplos DROP CONSTRAINT FK_ejemplo;
ALTER TABLE ejemplos DROP COLUMN FK_Ejemplo;
```

## ELIMINA UNA TABLA:



El comando utilizado fue:

**DROP TABLE** ejemplos;