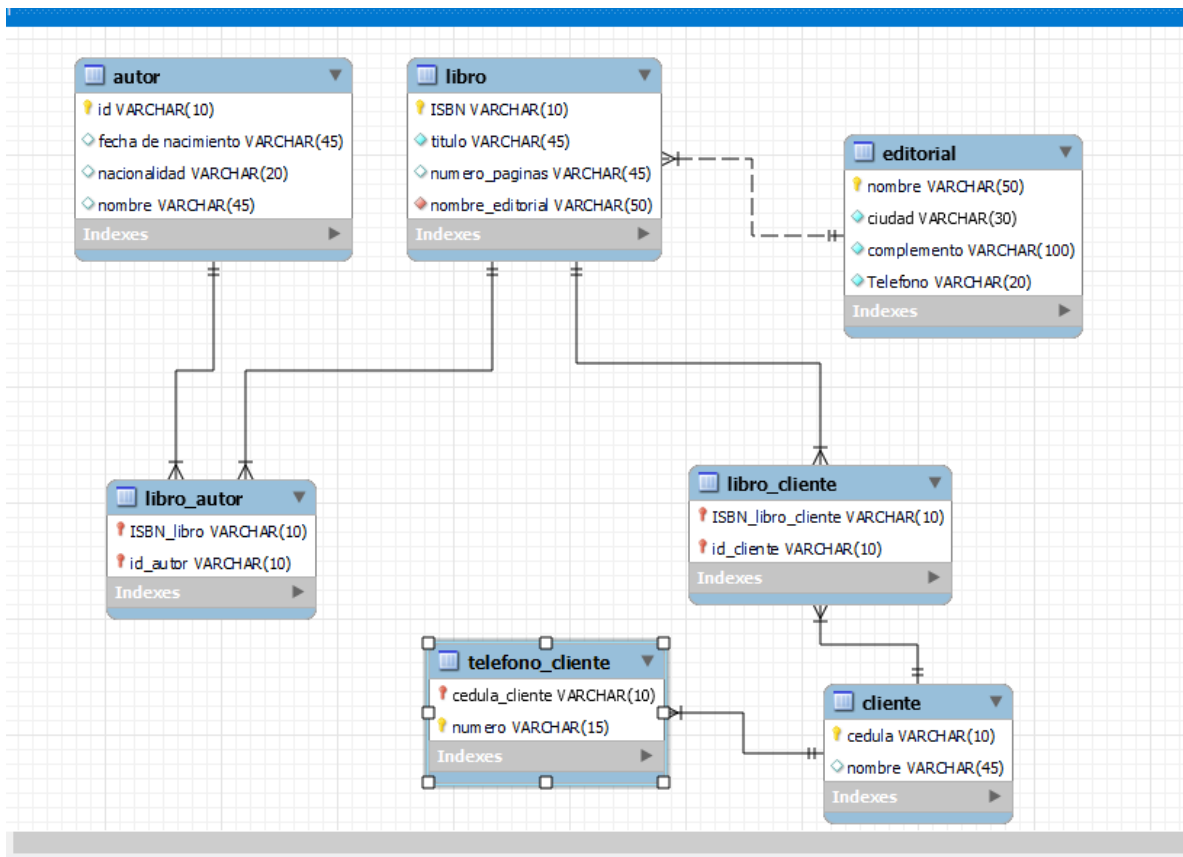


DOCUMENTACION TALLER #5 BASES DE DATOS ACTIVIDAD 1:

Para comenzar a realizar la primera actividad correspondiente a la librería empecé por validar el modelo relacional que el script me generó por medio de la ingeniería reversa, y esto lo hice con el fin de poder visualizar más fácilmente las tablas, los atributos, las llaves y las relaciones correspondientes entre las entidades antes de empezar a insertar los registros en la base de datos y procurar así mantener un orden y hacer un buen trabajo.

Este fue el modelo relacional que el script me generó:



Tras esto, continúe entonces llenando la base de datos es decir insertando el número de registros correspondientes, y la forma en que lo hice fue por medio de la sentencia INSERT TO + (nombre_de_la_tabla)+ (atributos de la tabla)+VALUES(ingresando aquí los valores que asignaré) y finalmente para tener una vista de esto que estaba haciendo utilice la sentencia SELECT *(que quiere decir en vagas palabras tráeme todo o muéstreme todo de) + FROM (nombre_de_la_tabla).

Empecé entonces por insertar los datos para las tablas principales, para así llevar un mejor orden.

- La primera en tener registros fue la de **Autor** y estas fueron las sentencias y registros que utilicé:

Nota: al ingresar los datos correspondientes para esta tabla , tuve una dificultad y fue a causa de que al nombrar el atributo fecha de nacimiento en la tabla autor, se hizo sin uso de camelCase o de alguna u otra forma más fácil de identificar y nombrar , entonces, cuando pasaba el atributo en el insert , tratando de llenarlo, no era posible porque no era la sintaxis esperada en Sql o simplemente no reconocía esa columna , entonces la forma de solucionarlo fue agregando las comillas simples invertidas haciendo alusión a los atributos creados con anterioridad . Pero, también evidencie que es posible hacer esto de ambas formas con las comillas o exactamente igual al nombre del atributo desde que sea posible según el caso.

Aquí una foto del atributo y/o columna que sql no me encontraba si lo seguía escribiendo **sin** las comillas:

```
-----  
CREATE TABLE IF NOT EXISTS `LibreriaBuscaLibre`.`autor` (  
  `id` VARCHAR(10) NOT NULL,  
  `fecha de nacimiento` VARCHAR(45) NULL,  
  `nacionalidad` VARCHAR(20) NULL,  
  `nombre` VARCHAR(45) NULL,  
  PRIMARY KEY (`id`))  
ENGINE = InnoDB;
```

Así **era** como estaba tratando de ingresar los datos, pero aquel error persistía:

```
INSERT INTO autor (id, fecha de nacimiento ,nacionalidad, nombre)  
VALUES  
  ('A001', '1980-01-01', 'Español', 'Juan Garcia '),  
  ('A002', '1985-05-12', 'Mexicano', 'María Rodríguez'),  
  ('A003', '1990-03-20', 'Argentino', 'Pedro Pérez'),  
  ('A004', '1995-07-15', 'Colombiano', 'Sofía García'),  
  ('A005', '1988-09-10', 'Peruano', 'Carlos Rodríguez');
```

Entonces las sentencias correctas que finalmente usé para agregar mis datos fueron estas:

INSERT INTO autor (`id`, `fecha de nacimiento`, `nacionalidad`, `nombre`)

VALUES

('A001', '1980-01-01', 'Español', 'Juan García'),

('A002', '1985-05-12', 'Mexicano', 'María Rodríguez'),

('A003', '1990-03-20', 'Argentino', 'Pedro Pérez'),

('A004', '1995-07-15', 'Colombiano', 'Sofía García'),

('A005', '1988-09-10', 'Peruano', 'Carlos Rodríguez');

Primero validé el ingreso de los datos de esta forma:

Query 1 Script generado libreria

ENGINE = InnoDB;

124

125

126

127 • SET SQL_MODE=@OLD_SQL_MODE;

128 • SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;

129 • SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

130

131 • INSERT INTO autor (`id`, `fecha de nacimiento`, `nacionalidad`, `nombre`)

132 VALUES

133 ('A001', '1980-01-01', 'Español', 'Juan García'),

134 ('A002', '1985-05-12', 'Mexicano', 'María Rodríguez'),

135 ('A003', '1990-03-20', 'Argentino', 'Pedro Pérez'),

136 ('A004', '1995-07-15', 'Colombiano', 'Sofía García'),

137 ('A005', '1988-09-10', 'Peruano', 'Carlos Rodríguez');

138

139 • SELECT*

140 FROM autor;

Output

Action Output

#	Time	Action	Message	Duration / Fetch
13	11:20:42	SET SQL_MODE=@OLD_SQL_MODE	0 row(s) affected	0.000 sec
14	11:20:42	SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS	0 row(s) affected	0.000 sec
15	11:20:42	SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS	0 row(s) affected	0.000 sec
16	12:17:14	INSERT INTO autor (`id`, `fecha de nacimiento`, `nacionalidad`, `nombre`) VALUES ('A001', '1980-01-01', 'Español', 'Juan García'), ('A002', '1985-05-12', 'Mexicano', 'María Rodríguez'), ('A003', '1990-03-20', 'Argentino', 'Pedro Pérez'), ('A004', '1995-07-15', 'Colombiano', 'Sofía García'), ('A005', '1988-09-10', 'Peruano', 'Carlos Rodríguez');	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.188 sec

Y luego lo que procedo a hacer es ver esos datos ingresados en la tabla:

137 ('A005', '1988-09-10', 'Peruano', 'Carlos Rodríguez');

138

139

140 • SELECT*

141 FROM autor;

Result Grid

	id	fecha de nacimiento	nacionalidad	nombre
▶	A001	1980-01-01	Español	Juan García
	A002	1985-05-12	Mexicano	María Rodríguez
	A003	1990-03-20	Argentino	Pedro Pérez
	A004	1995-07-15	Colombiano	Sofía García
	A005	1988-09-10	Peruano	Carlos Rodríguez

autor 1

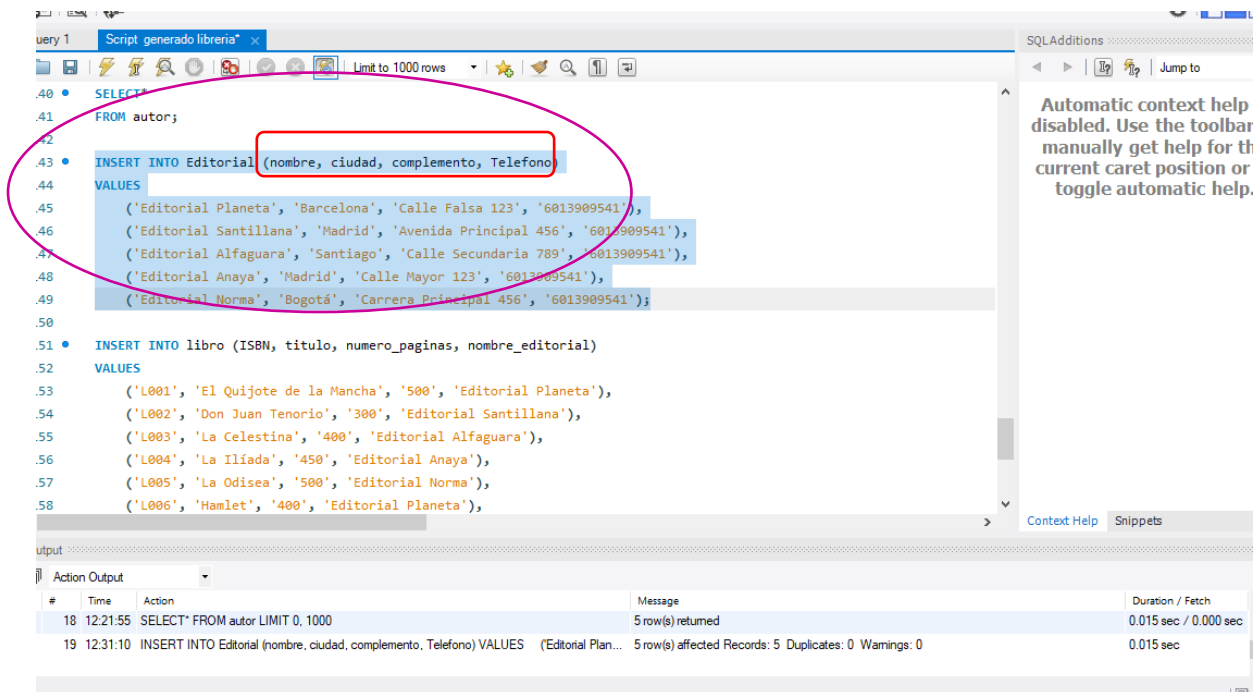
Output

- Luego, continué insertando los registros a la tabla de **editorial** y en esta ocasión no tuve inconveniente al llamar los atributos dentro de mi sentencia INSERT TO sin las comillas pues no había ningún problema con los caracteres que creé los atributos:

INSERT INTO Editorial (nombre, ciudad, complemento, Telefono)

VALUES

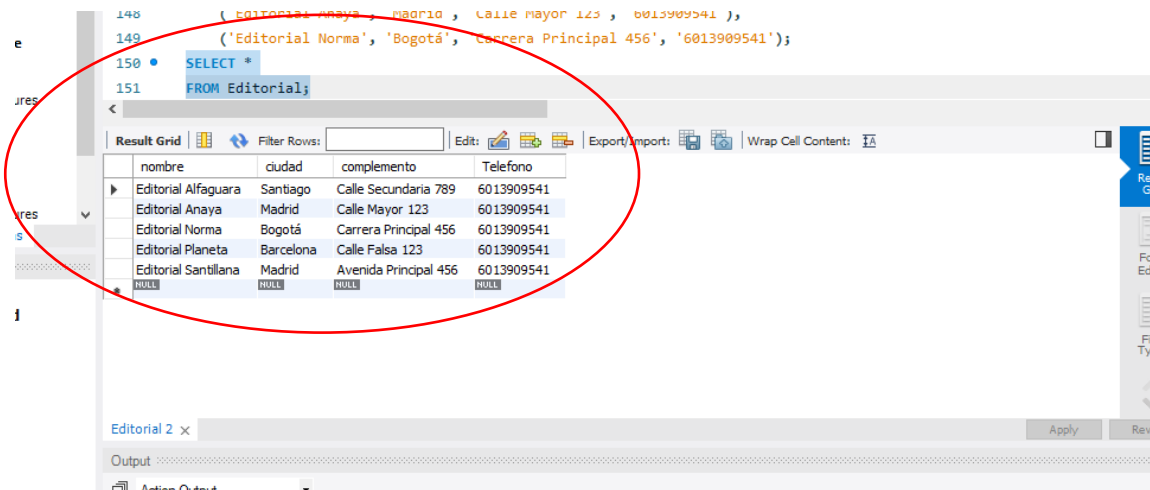
('Editorial Planeta', 'Barcelona', 'Calle Falsa 123', '6013909541'),
('Editorial Santillana', 'Madrid', 'Avenida Principal 456', '6013909541'),
('Editorial Alfaguara', 'Santiago', 'Calle Secundaria 789', '6013909541'),
('Editorial Anaya', 'Madrid', 'Calle Mayor 123', '6013909541'),
('Editorial Norma', 'Bogotá', 'Carrera Principal 456', '6013909541');



```
Query 1 Script generado libreria x
Limit to 1000 rows
40 SELECT
41 FROM autor;
42
43 INSERT INTO Editorial (nombre, ciudad, complemento, Telefono)
44 VALUES
45 ('Editorial Planeta', 'Barcelona', 'Calle Falsa 123', '6013909541'),
46 ('Editorial Santillana', 'Madrid', 'Avenida Principal 456', '6013909541'),
47 ('Editorial Alfaguara', 'Santiago', 'Calle Secundaria 789', '6013909541'),
48 ('Editorial Anaya', 'Madrid', 'Calle Mayor 123', '6013909541'),
49 ('Editorial Norma', 'Bogotá', 'Carrera Principal 456', '6013909541');
50
51 INSERT INTO libro (ISBN, titulo, numero_paginas, nombre_editorial)
52 VALUES
53 ('L001', 'El Quijote de la Mancha', '500', 'Editorial Planeta'),
54 ('L002', 'Don Juan Tenorio', '300', 'Editorial Santillana'),
55 ('L003', 'La Celestina', '400', 'Editorial Alfaguara'),
56 ('L004', 'La Iliada', '450', 'Editorial Anaya'),
57 ('L005', 'La Odisea', '500', 'Editorial Norma'),
58 ('L006', 'Hamlet', '400', 'Editorial Planeta');

Output
Action Output
# Time Action Message Duration / Fetch
18 12:21:55 SELECT* FROM autor LIMIT 0, 1000 5 row(s) returned 0.015 sec / 0.000 sec
19 12:31:10 INSERT INTO Editorial (nombre, ciudad, complemento, Telefono) VALUES ('Editorial Plan... 5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0 0.015 sec
```

De la misma manera validé los datos dentro de la tabla, usando la sentencia SELECT * FROM y esto fue lo que observé:



Después, continúe ingresando los demás registros pero a las tablas principales restantes, y esto, reitero nuevamente, lo hago con la intención de procurar un orden y de no tener ningún enredo a la hora de hacer uso de las llaves foráneas.

➤ Para la tabla **libros** estas fueron las sentencias que utilice:

INSERT INTO libro (ISBN, titulo, numero_paginas, nombre_editorial)

VALUES

('L001', 'El Quijote de la Mancha', '500', 'Editorial Planeta'),

('L002', 'Don Juan Tenorio', '300', 'Editorial Santillana'),

('L003', 'La Celestina', '400', 'Editorial Alfaguara'),

('L004', 'La Ilíada', '450', 'Editorial Anaya'),

('L005', 'La Odisea', '500', 'Editorial Norma'),

('L006', 'Hamlet', '400', 'Editorial Planeta'),

('L007', 'Othello', '450', 'Editorial Santillana'),

('L008', 'Romeo y Julieta', '500', 'Editorial Alfaguara'),

('L009', 'Macbeth', '450', 'Editorial Anaya'),

('L010', 'La Divina Comedia', '450', 'Editorial Norma'),

('L011', 'Los Argonautas', '400', 'Editorial Planeta'),

('L012', 'La Isla del Tesoro', '300', 'Editorial Santillana'),

('L013', 'Matar a un ruiseñor', '450', 'Editorial Alfaguara'),

('L014', 'El Gran Gatsby', '400', 'Editorial Anaya'),

('L015', '1984', '450', 'Editorial Norma'),

('L016', 'Una tempestad', '450', 'Editorial Planeta'),

('L017', 'La Guerra y la Paz', '500', 'Editorial Santillana'),

('L018', 'Anna Karenina', '450', 'Editorial Alfaguara'),

('L019', 'Madame Bovary', '400', 'Editorial Anaya'),

('L020', 'El Lobo Estepario', '450', 'Editorial Norma');

Y así es como se ven estos registros en la tabla:

1/4 ('L020', 'El Lobo Estepario', '450', 'Editorial Norma');

175 • SELECT *

ISBN	titulo	numero_paginas	nombre_editorial
L001	El Quijote de la Mancha	500	Editorial Planeta
L002	Don Juan Tenorio	300	Editorial Santillana
L003	La Celestina	400	Editorial Alfaguara
L004	La Ilíada	450	Editorial Anaya
L005	La Odisea	500	Editorial Norma
L006	Hamlet	400	Editorial Planeta
L007	Othello	450	Editorial Santillana
L008	Romeo y Julieta	500	Editorial Alfaguara
L009	Macbeth	450	Editorial Anaya
L010	La Divina Comedia	450	Editorial Norma
L011	Los Argonautas	400	Editorial Planeta
L012	La Isla del Tesoro	300	Editorial Santillana
L013	Matar a un ruiseñor	450	Editorial Alfaguara

libro 3 x Apply Revert Co

Output Action Output

173 ('L019', 'Madame Bovary', '400', 'Editorial Anaya'),

174 ('L020', 'El Lobo Estepario', '450', 'Editorial Norma');

175 • SELECT *

ISBN	titulo	numero_paginas	nombre_editorial
L009	Macbeth	450	Editorial Anaya
L010	La Divina Comedia	450	Editorial Norma
L011	Los Argonautas	400	Editorial Planeta
L012	La Isla del Tesoro	300	Editorial Santillana
L013	Matar a un ruiseñor	450	Editorial Alfaguara
L014	El Gran Gatsby	400	Editorial Anaya
L015	1984	450	Editorial Norma
L016	Una tempestad	450	Editorial Planeta
L017	La Guerra y la Paz	500	Editorial Santillana
L018	Anna Karenina	450	Editorial Alfaguara
L019	Madame Bovary	400	Editorial Anaya
L020	El Lobo Estepario	450	Editorial Norma
*	NULL	NULL	NULL

libro 3 x

- Registros para la tabla **clientes**:

```
INSERT INTO LibreriaBuscaLibre.cliente (cedula, nombre)

VALUES

("1234567890", "Juan Perez"),

("2345678901", "Maria Rodriguez"),

("3456789012", "Carlos González"),

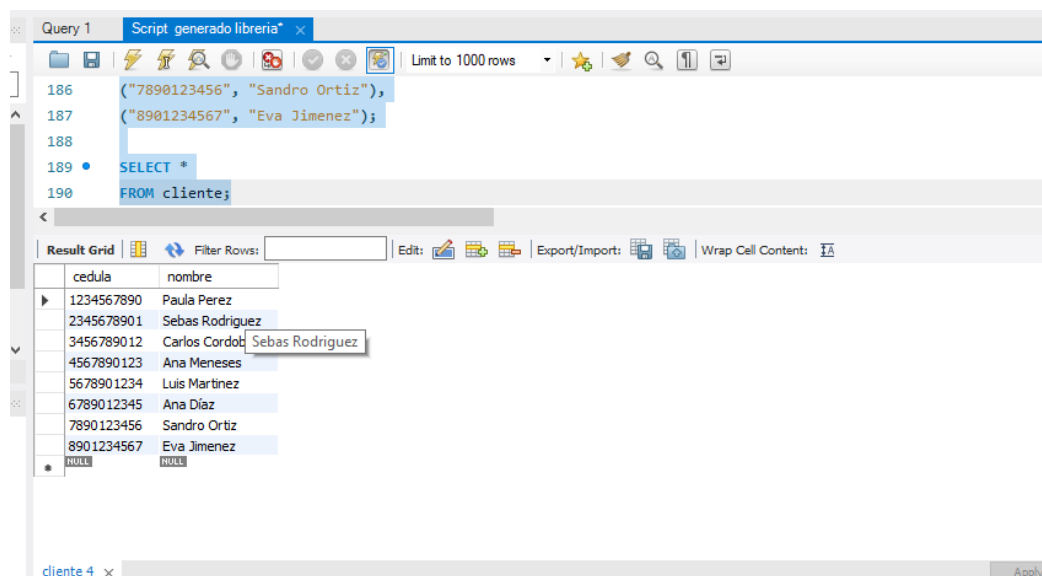
("4567890123", "Ana García"),

("5678901234", "Pedro Martinez"),

("6789012345", "Sofía Díaz"),

("7890123456", "Luis Ortiz"),

("8901234567", "Isabel Jimenez");
```



- Inserto registros para la tabla **libro_cliente** que me recibe las dos llaves de mis entidades correspondientes: libro + cliente

```
INSERT INTO libro_cliente (ISBN_libro_cliente, id_cliente)

VALUES

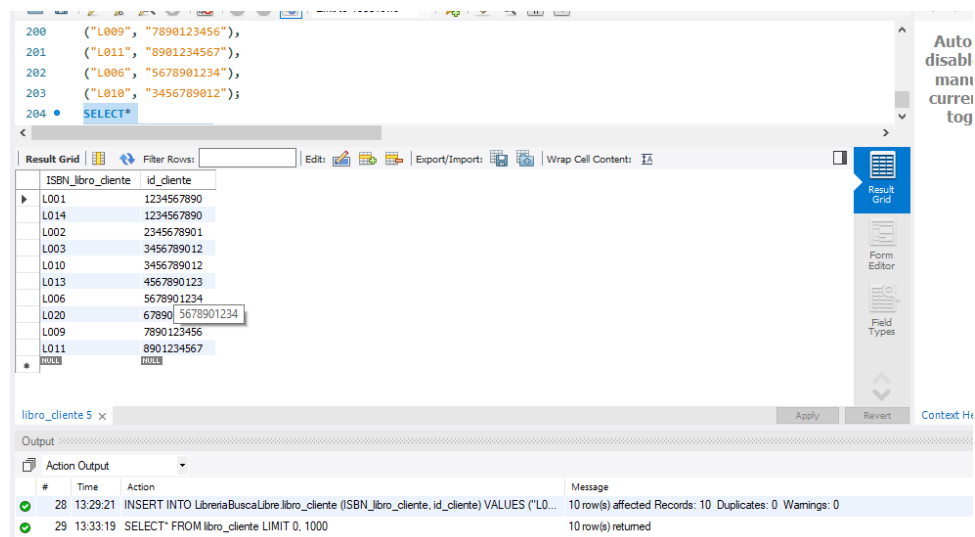
("L001", "1234567890"),

("L002", "2345678901"),

("L003", "3456789012"),
```

```
( "L013", "4567890123"),
( "L014", "1234567890"),
( "L020", "6789012345"),
( "L009", "7890123456"),
( "L011", "8901234567"),
( "L006", "5678901234"),
( "L010", "3456789012");
```

Validación del ingreso de los registros en tabla:



➤ Ingreso registros para la tabla **teléfono_cliente** :

Estas fueron las sentencias que usé:

INSERT INTO telefono_cliente (cedula_cliente, numero)

VALUES

```
( "1234567890", "555-555-1212"),
( "2345678901", "555-555-1213"),
( "3456789012", "555-555-1214"),
( "4567890123", "555-555-1215"),
( "5678901234", "555-555-1216"),
```

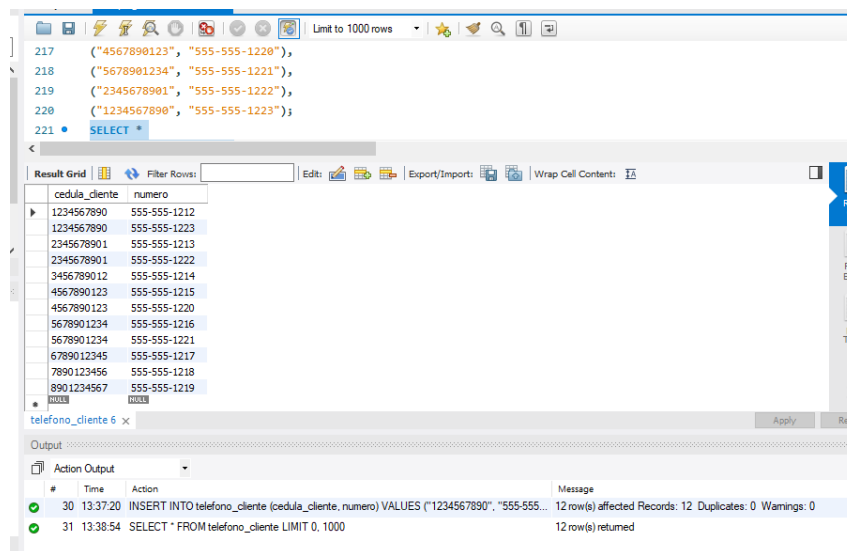


```

("6789012345", "555-555-1217"),
("7890123456", "555-555-1218"),
("8901234567", "555-555-1219"),
("4567890123", "555-555-1220"),
("5678901234", "555-555-1221"),
("2345678901", "555-555-1222"),
("1234567890", "555-555-1223");

```

Y así, valide el correcto ingreso de estos registros en la tabla:



Y, por último, poblé la **tabla libro_autor** que corresponde a una relación de escritura:

Estas fueron las sentencias que usé:

```
INSERT INTO libro_autor (ISBN_libro, id_autor)
```

```
VALUES
```

```

("L001", "A001"),
("L002", "A002"),
("L010", "A003"),
("L011", "A004"),
("L015", "A005"),
("L006", "A001"),

```

("L004", "A002"),

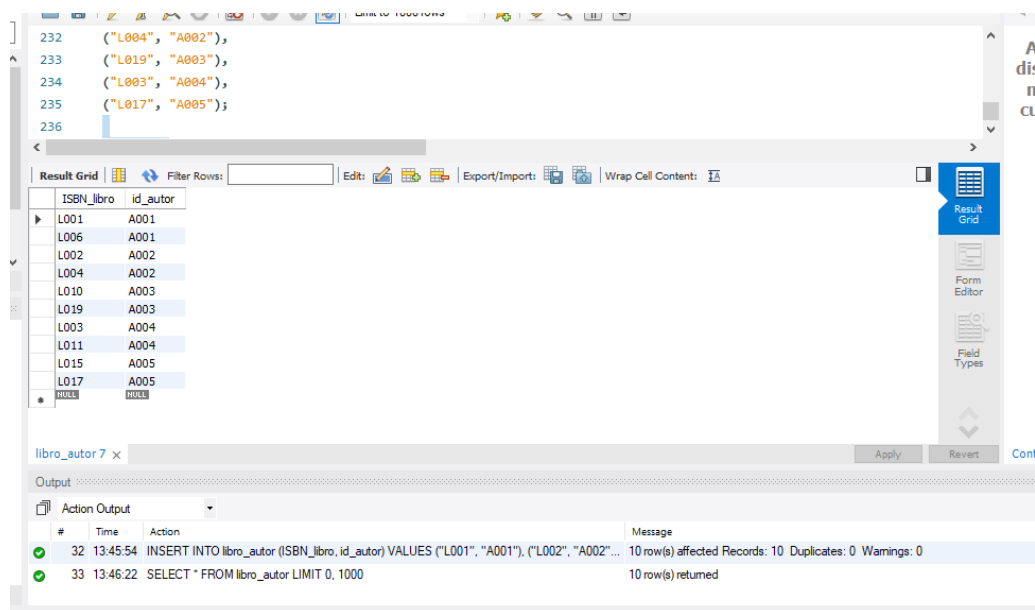
("L019", "A003"),

("L003", "A004"),

("L017", "A005");

Para las tablas que se creaban ya sea por atributos multivariados o por relaciones muchos a muchos, tenía que tener mucho cuidado a la hora de ingresar los datos pues primero que todo debía validar que ya había pasado datos a las entidades principales y tras esto , pasar los datos correspondientes para las llaves foráneas , es decir datos ya existentes.

Validación:



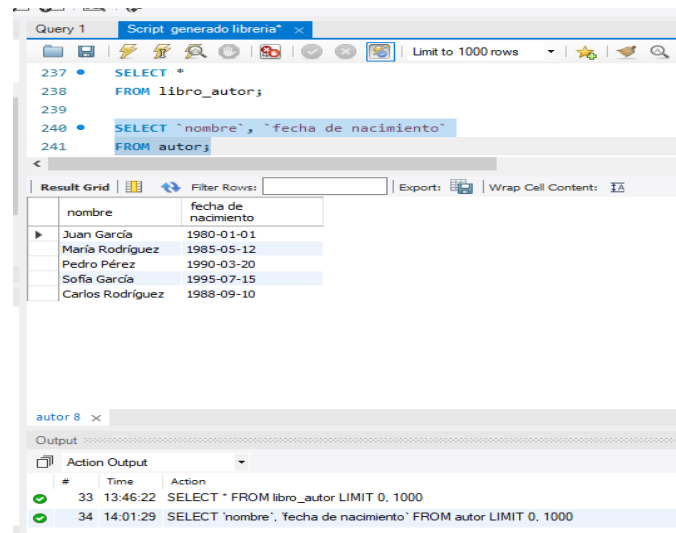
Luego de haber poblado completamente la base de datos de la librería, procedí a construir las sentencias para hacer allí mismo las consultas adecuadas:

- **Conocer el nombre y fecha de nacimiento de cada escritor:** para hacer esta consulta lo primero que intente hacer fue partir de la sentencia SELECT porque sabía que por ahí iba la cosa, con SELECT estaba ya diciéndole al programa “muéstrame esto”, “consúltame esto” y entonces con el FROM terminé de completar la consulta pues lo que hice allí fue pasarle el nombre de la tabla en donde iba a buscar esos atributos o aquello que deseaba ver . Finalmente, la consulta quedo así:

```
SELECT `nombre`, `fecha de nacimiento`
```

```
FROM autor;
```

Y así se veía:

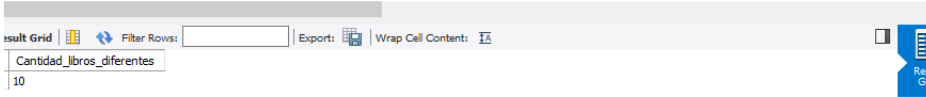


- **Cantidad de libros diferentes vendidos:** para hacer esta consulta, lo primero que hice fue empezar a profundizar mucho más en las sentencias para hacer las respectivas consultas. Entonces, con base en lo anterior lo que hice fue lo siguiente:
 1. Debía usar una sentencia **SELECT**
 2. También una función contadora, en este caso un **COUNT ()**
 3. Debía usar, un **as**, un alias, para dar un nombre a la columna que fuese a obtener como resultado
 4. Use la sentencia **DISTINCT** para hacer referencia a elementos diferentes
 5. Y finalmente, lo que hice fue identificar aquella columna de importancia, aquella con la que debía efectuar todo mi flujo para hacer una buena consulta , entonces me pregunté:¿Cómo puedo identificar en este contexto de la librería , la venta de libros diferentes , es decir únicos? Y así la respuesta fue con la llave primaria de la tabla, el **ISBN** es único para cada libro , entonces por ahí fue que logre armar bien mi consulta , quedando de la siguiente manera:

```
SELECT COUNT(DISTINCT ISBN_libro_cliente) as Cantidad_libros_diferentes
FROM libro_cliente;
```

Obteniendo así mi consulta:

```
13
14 -- consulta para conocer la cantidad de libros vendidos diferentes
15 • SELECT COUNT(DISTINCT ISBN_libro_cliente) as Cantidad_libros_diferentes
16 FROM libro_cliente;
17
18 -- consulta para conocer el nombre de su cliente acompañado de su numero telefónico
```



The screenshot shows a database query interface. At the top, there is a SQL query editor with three lines of code. The first line is a comment. The second line is a SQL query to count distinct ISBNs from a table named 'libro_cliente'. The third line is another comment. Below the editor, there is a 'Result Grid' section. It contains a table with one column named 'Cantidad_libros_diferentes' and one row with the value '10'. To the right of the grid, there are buttons for 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below the grid, there is a 'Result 1' tab and a 'Read' button.

- **Nombre del cliente acompañado de su número telefónico:**

Para comenzar a crear esta consulta, tenía muy claro que debía ser con la sentencia JOIN pero no estaba muy segura de cómo usarla para este caso, así que investigue un poco y, en mis palabras, la use para especificar cómo es que las filas de la tabla cliente y la tabla telefono_cliente se iban a relacionar entre sí. Usé también la sentencia ON para especificar la condición que se usa para unir estas filas, en este caso, que los valores en la columna cedula de la tabla cliente coincidan con los valores en la columna cedula_cliente de la tabla telefono_cliente.

Usé por supuesto la cláusula SELECT para especificar qué columnas incluiría en el resultado. En este caso, seleccioné entonces las columnas nombre de la tabla cliente y la columna numero de la tabla telefono_cliente.

El resultado será finalmente una tabla con dos columnas, nombre del cliente y su número, que incluye una fila para cada combinación única de valores en las dos tablas donde se cumpla la condición que especifiqué en la cláusula ON.

Entonces, la consulta quedó de la siguiente manera:

```
SELECT cliente.nombre, telefono_cliente.numero
```

```
FROM cliente
```

```
INNER JOIN telefono_cliente ON cliente.cedula = telefono_cliente.cedula_cliente;
```

Y así se ven los registros en la tabla:

```
247
248 -- consulta para conocer el nombre de su cliente acompañado de su numero telefónico
249 • SELECT cliente.nombre, telefono_cliente.numero
250 FROM cliente
251 INNER JOIN telefono_cliente ON cliente.cedula = telefono_cliente.cedula_cliente;
252
253 -- consulta nombre del libro acompañado de su autor
254 • SELECT libro.titulo, autor.nombre
```

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

nombre	numero
Paula Perez	555-555-1212
Paula Perez	555-555-1223
Sebas Rodriguez	555-555-1213
Sebas Rodriguez	555-555-1222
Carlos Cordoba	555-555-1214

Result 2 x

Output

```
247
248 -- consulta para conocer el nombre de su cliente acompañado de su numero telefónico
249 • SELECT cliente.nombre, telefono_cliente.numero
250 FROM cliente
251 INNER JOIN telefono_cliente ON cliente.cedula = telefono_cliente.cedula_cliente;
252
253 -- consulta nombre del libro acompañado de su autor
254 • SELECT libro.titulo, autor.nombre
```

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

nombre	numero
Carlos Cordoba	555-555-1214
Ana Meneses	555-555-1215
Ana Meneses	555-555-1220
Luis Martinez	555-555-1216
Luis Martinez	555-555-1221
Ana Diaz	555-555-1217
Sandro Ortiz	555-555-1218
Eva Jimenez	555-555-1219

Result 2 x

Output

- **Nombre del libro acompañado por su autor o sus autores:**

Para hacer esta consulta, uní tres tablas: "libro", "libro_autor" y "autor", utilizando las cláusulas o sentencias JOIN de la siguiente manera:

Uní primero la tabla "libro" con la tabla "libro_autor" utilizando la cláusula JOIN y su respectiva igualdad de valores en la columna "ISBN". Luego, uní la tabla "libro_autor" con la tabla "autor" utilizando también el JOIN y escribiendo su respectiva igualdad de valores en la columna "id_autor".

Así, el resultado final que tuve fue una tabla que contiene dos columnas: "titulo" y "nombre", donde "titulo" es el título del libro y "nombre" es el nombre del autor.

Estas fueron las sentencias:

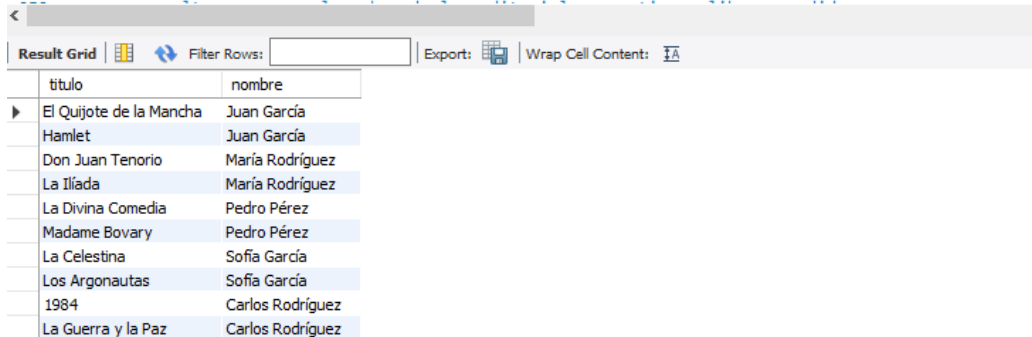
```
SELECT libro.titulo, autor.nombre
FROM libro
```

JOIN libro_autor ON libro.ISBN = libro_autor.ISBN_libro

JOIN autor ON libro_autor.id_autor = autor.id;

Así se ve la nueva tabla:

```
253 -- consulta nombre del libro acompañado de su autor
254 • SELECT libro.titulo, autor.nombre
255 FROM libro
256 JOIN libro_autor ON libro.ISBN = libro_autor.ISBN_libro
257 JOIN autor ON libro_autor.id_autor = autor.id;
258
```



The screenshot shows a database query result grid with two columns: 'titulo' and 'nombre'. The results are as follows:

titulo	nombre
El Quijote de la Mancha	Juan García
Hamlet	Juan García
Don Juan Tenorio	María Rodríguez
La Ilíada	María Rodríguez
La Divina Comedia	Pedro Pérez
Madame Bovary	Pedro Pérez
La Celestina	Sofía García
Los Argonautas	Sofía García
1984	Carlos Rodríguez
La Guerra y la Paz	Carlos Rodríguez

- **Nombre de las editoriales que han logrado vender libros.:**

Para hacer esta consulta uní tres tablas de mi interés: "libro", "editorial" y "libro_cliente", utilizando las cláusulas INNER JOIN.

Y para comenzar, junté la tabla "libro" con la tabla "editorial" utilizando la cláusula INNER JOIN y escribiendo la igualdad de valores en la columna "nombre_editorial". Luego, uní la tabla "libro" con la tabla "libro_cliente" utilizando también el INNER JOIN y escribiendo la igualdad de valores en la columna "ISBN".

Finalmente, usé la cláusula DISTINCT para obtener una lista de nombres de editoriales únicos que han vendido al menos un libro.

El resultado que obtuve fue la tabla que contiene una sola columna "nombre", donde cada fila corresponde a un nombre de editorial único que ha vendido al menos un libro.

Esta fue la consulta creada:

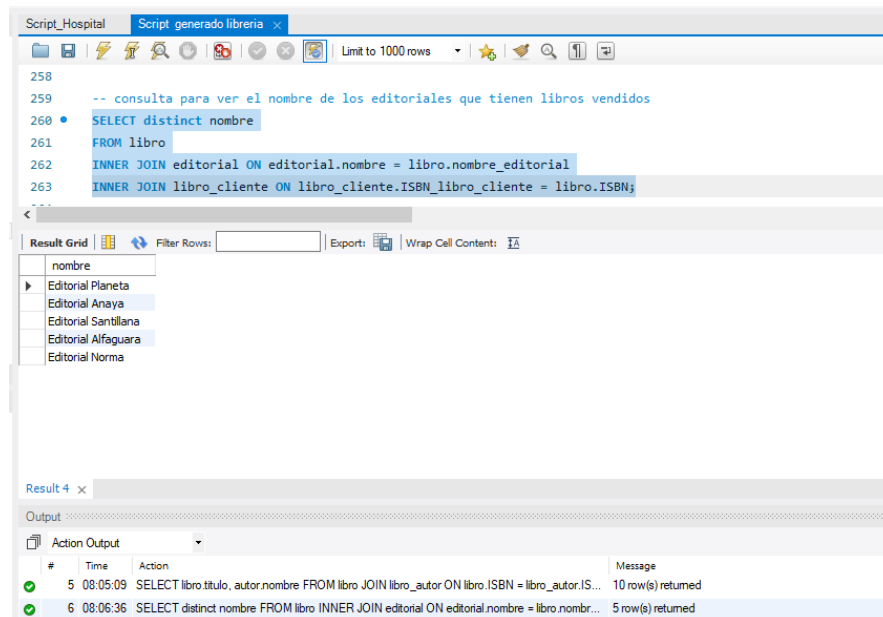
SELECT distinct nombre

FROM libro

INNER JOIN editorial ON editorial.nombre = libro.nombre_editorial

INNER JOIN libro_cliente ON libro_cliente.ISBN_libro_cliente = libro.ISBN;

Aquí la imagen de la nueva tabla:



Las vistas generadas mas importantes para mi fueron estas dos:

- ✓ vista para los libros más comprados por los clientes: decidí crear esta vista , porque teniendo en cuenta que estamos construyendo una base de datos para un sistema de librería , seria muy beneficioso tener un registro de la preferencia de tipos de literatura de nuestros clientes , eso podría servir y ser muy importante en un negocio de librería , por ejemplo para saber a que cliente se le notifican nuevos libros y de que tipo.

Estas fueron las sentencias:

```
CREATE VIEW librosPreferencia_clientes AS
```

```
SELECT cliente.nombre, libro.titulo
```

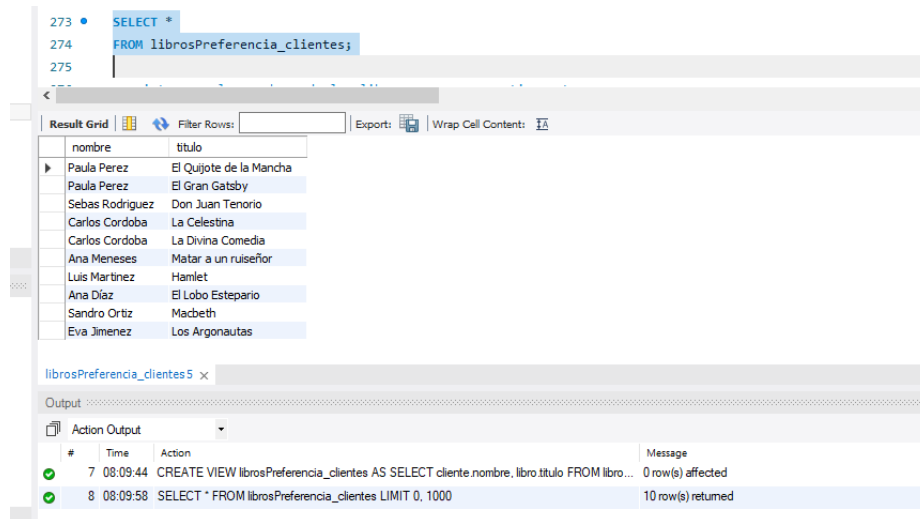
```
FROM libro_cliente
```

```
INNER JOIN libro ON libro_cliente.ISBN_libro_cliente = libro.ISBN
```

```
INNER JOIN cliente ON cliente.cedula = libro_cliente.id_cliente;
```

Este código que hice, creé una vista llamada "librosPreferencia_clientes" que muestra los nombres de los clientes y los títulos de los libros que han comprado. La vista la creé utilizando los datos de dos tablas: "libro_cliente" y "cliente", y una tabla "libro" que se une a través del campo "ISBN_libro_cliente".

La cláusula SELECT, super importante también, selecciona los campos "nombre" de la tabla "cliente" y "titulo" de la tabla "libro" y , finalmente el INNER JOIN lo usé para combinar las tres tablas en una sola vista utilizando las claves primarias y extrayendo solo los campos necesarios.



- ✓ vista para los nombres de los libros con su respectivo autor: considero que la creación de esta vista es muy importante teniendo en cuenta el funcionamiento de un sistema de librerías, pues tener acceso fácil y rápido a una tabla que contenga el nombre del libro y su autor , me permite tener a la mano , fácilmente, la información necesaria para empezar a hacer la búsqueda de un libro , por ejemplo en caso de tener el nombre del libro pero no el autor o viceversa, se empieza por ahí , y esta tabla resume buenos datos.

Estas fueron las sentencias que utilice para crear la vista :

```
CREATE VIEW informacion_libro AS

SELECT libro.titulo, autor.nombre

FROM libro

JOIN libro_autor ON libro.ISBN = libro_autor.ISBN_libro

JOIN autor ON libro_autor.id_autor = autor.id;
```

Y, para hacer esto creé la vista a partir de tres tablas: "libro", "libro_autor" y "autor". Luego, usé el JOIN para combinar estas tres tablas en una sola tabla virtual, así que finalmente la lógica para construirla fue esta:

La columna "titulo" la seleccioné de la tabla "libro", mientras que la columna "nombre" la seleccioné de la tabla "autor". La tabla "libro_autor" la utilicé para unir la tabla "libro" y la tabla "autor" utilizando el campo "ISBN_libro" y "id_autor".

```
283 • SELECT *
284 FROM informacion_libro;
285
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [↗](#)

titulo	nombre
El Quijote de la Mancha	Juan García
Hamlet	Juan García
Don Juan Tenorio	María Rodríguez
La Iliada	María Rodríguez
La Divina Comedia	Pedro Pérez
Madame Bovary	Pedro Pérez
La Celestina	Sofía García
Los Argonautas	Sofía García
1984	Carlos Rodríguez
La Guerra y la Paz	Carlos Rodríguez

informacion_libro 6 x

Output

Action Output

#	Time	Action	Message
✓ 9	08:12:35	CREATE VIEW informacion_libro AS SELECT libro.titulo, autor.nombre FROM libro JOIN libro...	0 row(s) affected
✓ 10	08:12:39	SELECT * FROM informacion_libro LIMIT 0, 1000	10 row(s) returned