

Documentación Taller #5

Para la primera actividad se nos pide utilizar el ejercicio de la librería realizado en clase con anterioridad.

Se nos piden los siguientes requerimientos:

- Complete la información para las tablas autor, libro, cliente, editorial, libro_cliente, libro_autor y teléfono_cliente con al menos (5,20,7,4,10,10, 12) registros respectivamente usando **Únicamente** comandos SQL creados por usted.
- realice 5 consultas que me permitan conocer el nombre y la fecha de nacimiento de cada escritor, la cantidad de libros diferentes vendidos, el nombre de su cliente acompañado de su número telefónico, el nombre del libro acompañado por su autor o sus autores, el nombre de las editoriales que han logrado vender libros.
- Realice las dos vistas que considere sean las más importantes y explique el motivo de su selección.

Para la solución del primer punto lo que hice fue empezar a poblar la base de datos con comandos SQL de la siguiente manera:

Para la tabla de autor:

The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' pane shows a tree view with 'libreriaBuscalibre' and 'libreriaadb'. The main area displays a SQL script with five INSERT statements for the 'autor' table, followed by a SELECT * query. The result grid below shows the data inserted into the 'autor' table.

id	fecha de nacimiento	nacionalidad	nombre
A001	01/01/1980	Español	Juan Pérez
A002	02/02/1985	Mexicano	María García
A003	03/03/1990	Argentino	Carlos Rodríguez
A004	04/04/1995	Colombiano	Sofía Martínez
A005	05/05/2000	Venezolano	José López
NULL	NULL	NULL	NULL

INSERT INTO se utiliza para insertar nuevos registros en una tabla. En este caso, se están insertando múltiples registros al mismo tiempo. En esta imagen podemos ver que cada registro representa un autor con su id, fecha de nacimiento, nacionalidad y nombre. Luego de la inserción, se ejecuta una consulta que muestra todos los registros de la tabla "autor" mediante la cláusula "SELECT *".

Para la tabla Editorial:

The screenshot shows a database management interface with a 'Navigator' pane on the left displaying a schema named 'libreriaBuscalibre'. The main area shows a SQL script for 'Query 1' with the following content:

```
141 • INSERT INTO LibreriaBuscaLibre.Editorial (nombre, ciudad, complemento, Telefono)
142 VALUES ('Editorial A', 'Ciudad A', 'Complemento A', '555-555-5555'),
143 ('Editorial B', 'Ciudad B', 'Complemento B', '555-555-5556'),
144 ('Editorial C', 'Ciudad C', 'Complemento C', '555-555-5557'),
145 ('Editorial D', 'Ciudad D', 'Complemento D', '555-555-5558');
146 • SELECT *
147 FROM LibreriaBuscaLibre.Editorial;
```

Below the script, the 'Result Grid' displays the data for the 'Editorial' table:

nombre	ciudad	complemento	Telefono
Editorial A	Ciudad A	Complemento A	555-555-5555
Editorial B	Ciudad B	Complemento B	555-555-5556
Editorial C	Ciudad C	Complemento C	555-555-5557
Editorial D	Ciudad D	Complemento D	555-555-5558
NULL	NULL	NULL	NULL

Aquí en esta imagen podemos observar que cada registro representa una editorial con su nombre, ciudad, complemento y número de teléfono. Luego de la inserción, se ejecuta una consulta que muestra todos los registros de la tabla "Editorial" mediante la cláusula "SELECT *" mas el nombre de la tabla.

Para la tabla Libro:

The screenshot shows the same database management interface. The SQL script for 'Query 1' now includes an insert statement for the 'Libro' table:

```
166 ('5836914', 'Libro P', '525', 'Editorial D'),
167 ('3691470', 'Libro Q', '275', 'Editorial A'),
168 ('1470258', 'Libro R', '375', 'Editorial B'),
169 ('7025836', 'Libro S', '475', 'Editorial C'),
170 ('5836919', 'Libro T', '575', 'Editorial D');
171 • SELECT *
172 FROM LibreriaBuscaLibre.libro;
173
174 • INSERT INTO LibreriaBuscaLibre.libro_autor (ISBN_libro, id_a
175 VALUES ('1234567890', 'A1'),
```

The 'Result Grid' displays the data for the 'Libro' table:

ISBN	titulo	numero_paginas	nombre_editorial
0987654	Libro B	300	Editorial B
1029381	Libro C	400	Editorial C
1234567890	Libro A	200	Editorial A
1470258	Libro R	375	Editorial B
147058	Libro N	325	Editorial B
1472583	Libro K	475	Editorial C
1597532	Libro J	375	Editorial B
2583691	Libro M	225	Editorial A
3691470	Libro Q	275	Editorial A
3692581	Libro L	575	Editorial D
5432167	Libro D	500	Editorial D
5836914	Libro P	525	Editorial D
5836919	Libro T	575	Editorial D
67890ae	Libro E	250	Editorial A
7025836	Libro S	475	Editorial C
70258f6	Libro O	425	Editorial C
7894561	Libro I	275	Editorial A

En la imagen anterior vemos que Cada registro representa un libro con su ISBN, título, número de páginas y nombre de la editorial. Luego de la inserción, se ejecuta una consulta que muestra todos los registros de la tabla "libro" mediante la cláusula "SELECT *" mas el nombre de la tabla que en este caso sería Libro.

Para la tabla Libro_autor:

The screenshot shows a database management interface with a 'Navigator' pane on the left and a 'Query 1' editor on the right. The 'Navigator' pane shows a tree of schemas, including 'libreriaBuscalibre', 'libreriadb', 'parquenorte', 'sakila', 'sys', and 'world'. The 'Query 1' editor shows a SQL script with the following content:

```

181      ('70258f6', 'A001'),
182      ('147058', 'A002'),
183      ('3692581', 'A003'),
184      ('2583691', 'A004'),
185      ('1472583', 'A005');
186      SELECT *
187      FROM LibreriaBuscaLibre.libro_autor;
188

```

Below the query editor, the 'Result Grid' shows the results of the query. The grid has two columns: 'ISBN_libro' and 'id_autor'. The results are as follows:

ISBN_libro	id_autor
5836919	A001
70258f6	A001
147058	A002
7025836	A002
1470258	A003
3692581	A003
2583691	A004
3691470	A004
1472583	A005
5836914	A005
NULL	NULL

At the bottom of the interface, the 'Schema: libreriadb' is selected, and the 'Output' pane shows the query name 'libro_autor 4'.

Para esta tabla se inserta diez registros en la tabla "libro_autor" de la base de datos "LibreriaBuscaLibre". Cada registro representa la relación entre un libro y un autor, identificados por su ISBN y su ID respectivamente. Luego de la inserción, se ejecuta una consulta que muestra todos los registros de la tabla "libro_autor" mediante la cláusula "SELECT *" mas el nombre de su respectiva tabla.

Para la tabla Cliente:

The screenshot displays a database management interface. On the left, the 'SCHEMAS' pane shows a tree view with databases like 'libreriaBuscaLibre', 'libreriaDb', and 'mydb'. Under 'mydb', there are several tables including 'enfermero', 'factura', 'medicamento', 'medicamento_has_p', 'medico', 'paciente', 'procedimiento', 'procedimiento_has_p', 'telefono_enfermero', 'telefono_medico', and 'telefono_paciente'. The 'Tables' folder is expanded, and 'telefono_paciente' is selected. Below the tree, the 'Administration' tab is active, showing 'Schemas' and 'Information' sections. The 'No object selected' message is visible.

The main pane shows a SQL script titled 'Script generado libreria1'. The script contains an INSERT statement followed by a SELECT statement. The INSERT statement inserts seven records into the 'cliente' table of the 'LibreriaBuscaLibre' database. The SELECT statement retrieves all records from the 'cliente' table.

```
190 • INSERT INTO `LibreriaBuscaLibre`.`cliente` (`cedula`, `nombre`)
191 VALUES
192 ("1000000000", "Juan"),
193 ("2000000000", "María"),
194 ("3000000000", "Pedro"),
195 ("4000000000", "Sofía"),
196 ("5000000000", "Carlos"),
197 ("6000000000", "Ana"),
198 ("7000000000", "Luis");
199 • SELECT *
200 FROM `LibreriaBuscaLibre`.`cliente`;
201
```

Below the script, the 'Result Grid' shows the results of the SELECT statement. The grid has two columns: 'cedula' and 'nombre'. It displays seven rows of data, corresponding to the records inserted by the script. The last row shows 'NULL' for both 'cedula' and 'nombre'.

cedula	nombre
1000000000	Juan
2000000000	María
3000000000	Pedro
4000000000	Sofía
5000000000	Carlos
6000000000	Ana
7000000000	Luis
NULL	NULL

En la imagen anterior se insertan siete registros en la tabla "cliente" de la base de datos "LibreriaBuscaLibre". Cada registro representa un cliente identificado por su número de cédula y su nombre. Luego de la inserción, se ejecuta una consulta que muestra todos los registros de la tabla "cliente" mediante la cláusula "SELECT *" mas el nombre de la tabla.

Para la tabla **Libro_Cliente**:

The screenshot shows a database management interface with a 'Navigator' pane on the left and a 'Query 1' pane on the right. The 'Navigator' pane shows a tree of schemas, including 'libreriaBuscalibre', 'libreriaadb', 'parquenorte', 'sakila', 'sys', and 'world'. The 'Query 1' pane shows a SQL script with the following content:

```
208 ('5836914', '5000000000'),
209 ('70258f6', '6000000000'),
210 ('147058', '7000000000'),
211 ('3692581', '1000000000'),
212 ('2583691', '2000000000'),
213 ('1472583', '3000000000');
214 SELECT *
215 FROM `LibreriaBuscaLibre`.`libro_cliente`;
```

Below the script, the 'Result Grid' shows the results of the query. The grid has two columns: 'ISBN_libro_cliente' and 'id_cliente'. The results are as follows:

ISBN_libro_cliente	id_cliente
3692581	1000000000
5836919	1000000000
2583691	2000000000
7025836	2000000000
1470258	3000000000
1472583	3000000000
3691470	4000000000
5836914	5000000000
70258f6	6000000000
147058	7000000000
NULL	NULL

En este caso, se están insertando múltiples registros al mismo tiempo, cada uno con dos valores: **ISBN_libro_cliente** y **id_cliente**.

La consulta **SELECT * FROM** se utiliza para seleccionar todos los registros de una tabla. En este caso, se seleccionan todos los registros de la tabla **libro_cliente**.

Y finalmente para la **tabla Telefono_Cliente**:

The screenshot shows a database management interface with a 'Navigator' pane on the left and a 'Query 1' pane on the right. The 'Navigator' pane shows a tree of schemas: 'libreria_buscabilibre' (containing Tables, Views, Stored Procedures, and Functions), 'libreria_db', 'parquenorte', 'sakila', 'sys', and 'world'. The 'Query 1' pane shows a SQL script with lines 224 to 231. The script contains a series of INSERT statements followed by a SELECT statement. The results of the SELECT statement are displayed in a 'Result Grid' below the script. The 'Result Grid' has two columns: 'cedula_cliente' and 'numero'. The results show 10 rows of data, each representing a client and their phone numbers.

```
224 ("4000000000", "555-555-6789"),
225 ("5000000000", "555-555-7890"),
226 ("6000000000", "555-555-8901"),
227 ("7000000000", "555-555-9012"),
228 ("4000000000", "555-555-0123"),
229 ("7000000000", "555-555-1234"),
230 ("6000000000", "555-555-2345");
231 • SELECT *
```

cedula_cliente	numero
1000000000	555-555-1234
1000000000	555-555-2345
2000000000	555-555-3456
2000000000	555-555-4567
3000000000	555-555-5678
4000000000	555-555-0123
4000000000	555-555-6789
5000000000	555-555-7890
6000000000	555-555-2345
6000000000	555-555-8901
7000000000	555-555-1234
7000000000	555-555-9012

En este caso, se están insertando múltiples registros al mismo tiempo, cada uno con dos valores: **cedula_cliente** y **numero**.

Cada conjunto de valores representa un número de teléfono y su correspondiente cliente. Por ejemplo, el primer conjunto de valores representa dos números de teléfono diferentes que pertenecen al mismo cliente con cédula "1000000000".

La consulta **SELECT * FROM** se utiliza para seleccionar todos los registros de una tabla. En este caso, se seleccionan todos los registros de la tabla **telefono_cliente**.

Después de poblar la base de datos de la librería mi siguiente paso fue hacer las respectivas consultas que en el taller se piden.

Lo hice de la siguiente manera:

CONSULTAS - El nombre y la fecha de nacimiento de cada escritor:

The screenshot shows a database management interface. On the left is a 'SCHEMAS' tree with a search bar 'Filter objects'. It contains a tree for 'libreriaabusalibre' with sub-items 'Tables', 'Views', 'Stored Procedures', and 'Functions'. Below it is 'libreriadb' and 'mydb'. Under 'mydb', there is a 'Tables' folder containing: 'enfermero', 'factura', 'medicamento', 'medicamento_has_p', 'medico', 'paciente', 'procedimiento', 'procedimiento_has_p', 'telefono_enfermero', 'telefono_medico', and 'telefono_paciente'. There is also a 'Views' folder. At the bottom of the left pane are tabs for 'Administration' and 'Schemas', and an 'Information' section.

The main area is titled 'Script: generado libreria1'. It contains a list of SQL queries. Lines 234-236 show a comment: `-----#CONSULTAS - El nombre y la fecha de nacimiento de cada escritor -----`. Lines 237-241 show five SELECT queries, each selecting 'nombre' and 'fecha_de_nacimiento' from the 'autor' table for a specific 'id' ('A001' through 'A005'). Line 242 is blank. Lines 244-245 show another comment: `-----#CONSULTAS - La cantidad de libros diferentes vendidos -----`.

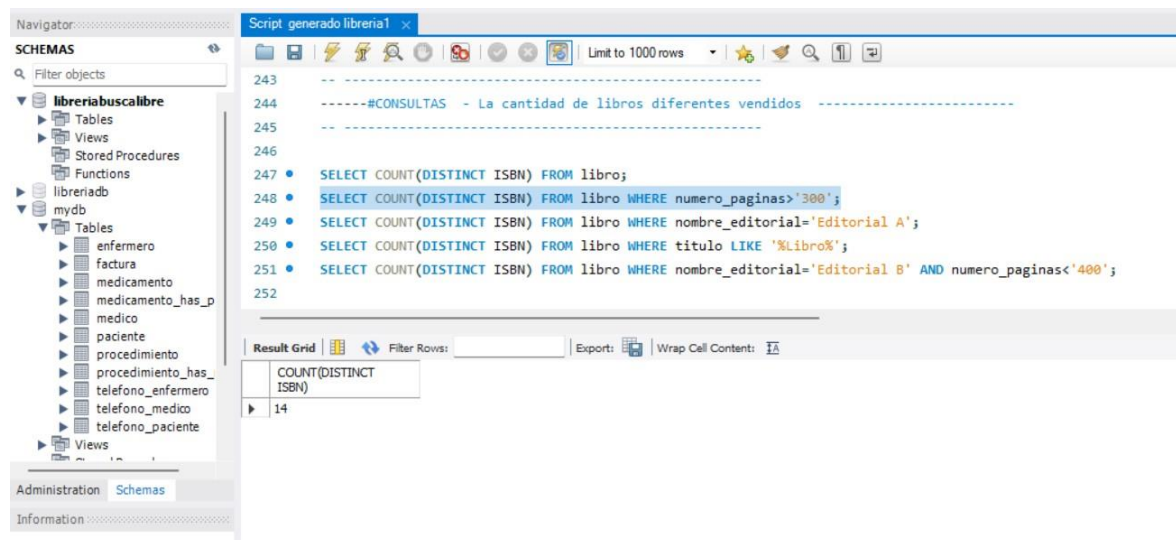
Below the queries is a 'Result Grid' section. It has a 'Filter Rows' input and an 'Export' button. The grid shows two columns: 'nombre' and 'fecha_de_nacimiento'. The first row contains the data: 'Juan Pérez' and '01/01/1980'.

En la imagen anterior se seleccionan el nombre y la fecha de nacimiento de cinco autores diferentes de la tabla autor. Cada consulta selecciona un autor específico según su id.

Cada consulta tiene la misma estructura básica: utiliza la cláusula SELECT para seleccionar dos columnas específicas (nombre y fecha_de_nacimiento) de la tabla autor, y luego utiliza la cláusula WHERE para especificar el autor específico que se desea seleccionar, utilizando su id.

CONSULTAS - La cantidad de libros diferentes vendidos:

Se cuentan todos los libros únicos en la tabla libro que tienen más de 300 páginas:



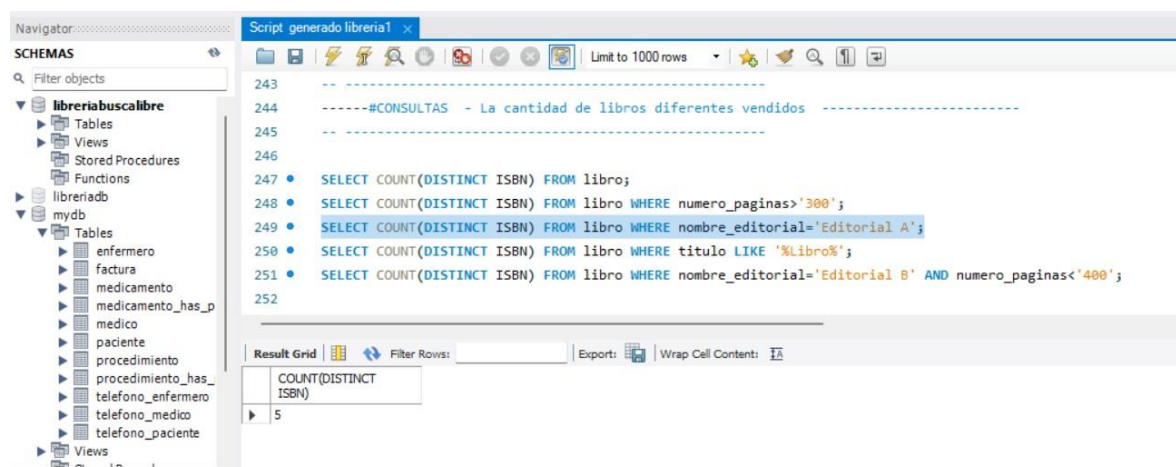
The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including tables like 'enfermero', 'factura', 'medicamento', and 'paciente'. The main area displays a SQL script titled 'Script generado libreria1'. The script contains several SQL queries, with the following query highlighted:

```
SELECT COUNT(DISTINCT ISBN) FROM libro WHERE numero_paginas > 300;
```

Below the script, the 'Result Grid' shows the results of the query:

COUNT(DISTINCT ISBN)
14

Se cuentan todos los libros únicos en la tabla libro que son publicados por la editorial "Editorial A":



The screenshot shows the same database management tool interface. The SQL script in the main area is the same as in the previous screenshot, but the query highlighted is:

```
SELECT COUNT(DISTINCT ISBN) FROM libro WHERE nombre_editorial = 'Editorial A';
```

Below the script, the 'Result Grid' shows the results of the query:

COUNT(DISTINCT ISBN)
5

se cuentan todos los libros únicos en la tabla **libro**.

The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including tables like 'enfermero', 'factura', 'medicamento', 'medicamento_has_p', 'medico', 'paciente', 'procedimiento', 'procedimiento_has_telefono_enfermero', 'telefono_medico', and 'telefono_paciente'. The main area displays a SQL script titled 'Script generado libreria1'. The script contains several queries, including a query to count distinct ISBNs from the 'libro' table. The results pane at the bottom shows the output of the first query: 'COUNT(DISTINCT ISBN)' with a value of 20.

```
243 -----#CONSULTAS - La cantidad de libros diferentes vendidos -----
244 -----#CONSULTAS - La cantidad de libros diferentes vendidos -----
245 -----#CONSULTAS - La cantidad de libros diferentes vendidos -----
246
247 • SELECT COUNT(DISTINCT ISBN) FROM libro;
248 • SELECT COUNT(DISTINCT ISBN) FROM libro WHERE numero_paginas>'300';
249 • SELECT COUNT(DISTINCT ISBN) FROM libro WHERE nombre_editorial='Editorial A';
250 • SELECT COUNT(DISTINCT ISBN) FROM libro WHERE titulo LIKE '%Libro%';
251 • SELECT COUNT(DISTINCT ISBN) FROM libro WHERE nombre_editorial='Editorial B' AND numero_paginas<'400';
252
```

Result Grid
COUNT(DISTINCT ISBN)
20

Y en la siguiente se muestran todos los libros únicos en la tabla libro que tienen "Libro" en su título:

The screenshot shows the same database management tool interface as the previous one. The SQL script titled 'Script generado libreria1' now includes a query to count distinct ISBNs from the 'libro' table where the title contains 'Libro'. The results pane at the bottom shows the output of this query: 'COUNT(DISTINCT ISBN)' with a value of 20.

```
246
247 • SELECT COUNT(DISTINCT ISBN) FROM libro;
248 • SELECT COUNT(DISTINCT ISBN) FROM libro WHERE numero_paginas>'300';
249 • SELECT COUNT(DISTINCT ISBN) FROM libro WHERE nombre_editorial='Editorial A';
250 • SELECT COUNT(DISTINCT ISBN) FROM libro WHERE titulo LIKE '%Libro%';
251 • SELECT COUNT(DISTINCT ISBN) FROM libro WHERE nombre_editorial='Editorial B' AND numero_paginas<'400';
252
253 -----#CONSULTAS - El nombre de su cliente acompañado de su numero telefónico -----
254 -----#CONSULTAS - El nombre de su cliente acompañado de su numero telefónico -----
255 -----#CONSULTAS - El nombre de su cliente acompañado de su numero telefónico -----
```

Result Grid
COUNT(DISTINCT ISBN)
20

CONSULTAS - El nombre de su cliente acompañado de su número telefónico:

```
254 -----#CONSULTAS - El nombre de su cliente acompañado de su numero telefónico
255 -----
256
257 • SELECT c.nombre, t.numero
258 FROM cliente c
259 JOIN telefono_cliente t ON c.cedula = t.cedula_cliente;
260
261 • SELECT c.nombre, GROUP_CONCAT(t.numero SEPARATOR ', ') AS numeros
262 FROM cliente c
263 JOIN telefono_cliente t ON c.cedula = t.cedula_cliente
264 GROUP BY c.nombre;
265
266 • SELECT c.nombre, t.numero
267 FROM cliente c
268 LEFT JOIN telefono_cliente t ON c.cedula = t.cedula_cliente;
269
270 • SELECT c.nombre, t.numero
271 FROM telefono_cliente t
272 JOIN cliente c ON t.cedula_cliente = c.cedula;
273
274 • SELECT c.nombre, t.numero
275 FROM telefono_cliente t
276 RIGHT JOIN cliente c ON t.cedula_cliente = c.cedula;
277 ~~~
```

Estas son consultas SQL que involucran la combinación (join) de dos tablas cliente y telefono_cliente, y en cada consulta se seleccionan ciertas columnas de estas tablas.

-La primera consulta es una inner join, que devuelve el nombre del cliente y su número de teléfono.

-La segunda consulta es también un inner join, pero agrupa por nombre de cliente y usa la función GROUP_CONCAT para concatenar todos los números de teléfono del cliente separados por una coma en una sola cadena.

-La tercera consulta es una left join, que devuelve todos los clientes y sus números de teléfono si tienen uno. Si un cliente no tiene un número de teléfono asociado en la tabla telefono_cliente, todavía se mostrará su nombre en la consulta, pero los valores correspondientes de la tabla telefono_cliente serán nulos.

-La cuarta consulta es igual que la primera consulta, solo que cambia el orden en que se unen las tablas.

-La quinta consulta es una right join, devuelve todos los números de teléfono y los nombres de los clientes que tienen números de teléfono asociados en la tabla telefono_cliente. Si un número de teléfono no tiene un cliente asociado en la tabla cliente, todavía se mostrará en la consulta, pero los valores correspondientes de la tabla cliente serán nulos.

En la siguiente imagen se muestra el resultado de la primera:

The screenshot shows a database management interface with a left sidebar containing a tree view of schemas. The main area displays a SQL script with two queries. The first query is a right join between 'cliente' and 'telefono_cliente' tables. The second query is a right join between 'cliente' and 'telefono_cliente' tables, using 'GROUP_CONCAT' to concatenate phone numbers. The results are shown in a table with two columns: 'nombre' and 'numero'.

Script: generado libreria1

```
254 -----#CONSULTAS - El nombre de su cliente acompañado de su numero telefonico --
255 -----
256
257 • SELECT c.nombre, t.numero
258 FROM cliente c
259 JOIN telefono_cliente t ON c.cedula = t.cedula_cliente;
260
261 • SELECT c.nombre, GROUP_CONCAT(t.numero SEPARATOR ', ') AS numeros
262 FROM cliente c
263 JOIN telefono_cliente t ON c.cedula = t.cedula_cliente
```

Result Grid

nombre	numero
Juan	555-555-1234
Juan	555-555-2345
María	555-555-3456
María	555-555-4567
Pedro	555-555-5678
Sofía	555-555-0123
Sofía	555-555-6789
Carlos	555-555-7890
Ana	555-555-2345
Ana	555-555-8901
Luis	555-555-1234
Luis	555-555-9012

CONSULTAS - El nombre del libro acompañado por su autor o sus autores:

The screenshot shows a database management interface with a Navigator pane on the left and a Script editor on the right. The Navigator pane displays a schema named 'libreriaBuscalibre' with various tables and views. The Script editor shows a SQL query that joins three tables: 'LibreriaBuscaLibre.libro', 'LibreriaBuscaLibre.libro_autor', and 'LibreriaBuscaLibre.autor'. The query selects the title and author name from the 'libro' table, joined with the 'libro_autor' table on ISBN, and the 'autor' table on author ID. The results are displayed in a grid below the script.

```
277
278
279  -- ----#CONSULTAS - El nombre del libro acompañado por su autor o sus autores -
280  -- ----
281
282  • SELECT libro.titulo, autor.nombre
283     FROM LibreriaBuscaLibre.libro
284    JOIN LibreriaBuscaLibre.libro_autor ON libro.ISBN = libro_autor.ISBN_libro
285    JOIN LibreriaBuscaLibre.autor ON libro_autor.id_autor = autor.id;
286
```

titulo	nombre
Libro T	Juan Pérez
Libro O	Juan Pérez
Libro N	María García
Libro S	María García
Libro R	Carlos Rodríguez
Libro L	Carlos Rodríguez
Libro M	Sofía Martínez
Libro Q	Sofía Martínez
Libro K	José López
Libro P	José López

Esta consulta es una combinación de tres tablas, libro, libro_autor, y autor, mediante los campos clave ISBN, ISBN_libro, id_autor, e id, respectivamente.

La consulta devuelve los nombres de los autores y los títulos de los libros que escribieron, coincidiendo los ISBN de la tabla libro con los ISBN de la tabla libro_autor, y los ids de los autores de la tabla autor con los ids de los autores de la tabla libro_autor.

CONSULTAS -El nombre de las editoriales que han logrado vender libros:

The screenshot shows a database management interface with a Navigator pane on the left and a Script editor on the right. The Navigator pane displays a schema named 'libreriaBuscalibre' with various tables and views. The Script editor shows a SQL query that selects distinct editorial names from the 'libro' table. The results are displayed in a grid below the script.

```
286
287
288  -- ----#CONSULTAS -El nombre de las editoriales que han logrado vender libros
289  -- ----
290  • SELECT DISTINCT nombre_editorial
291     FROM LibreriaBuscaLibre.libro;
292
293
294  -- ----#-VISTAS que considero son las más importantes-----
295  -- ----
```

nombre_editorial
Editorial A
Editorial B
Editorial C
Editorial D

En la imagen anterior la consulta devuelve los nombres de las editoriales que aparecen en la tabla **libro** de la base de datos **LibreriaBuscaLibre**. La cláusula **DISTINCT** se utiliza para asegurar que no se repitan nombres de editoriales en los resultados.

En resumen, esta consulta retorna una lista de editoriales diferentes que han publicado libros en la librería representada en la base de datos **LibreriaBuscaLibre**.

Para el siguiente paso que es buscar las vistas más importantes según mi criterio tuve que investigar un poco más sobre este tema ya que no me quedaba muy claro y encontré lo siguiente:

que es una vista en esta base de datos?

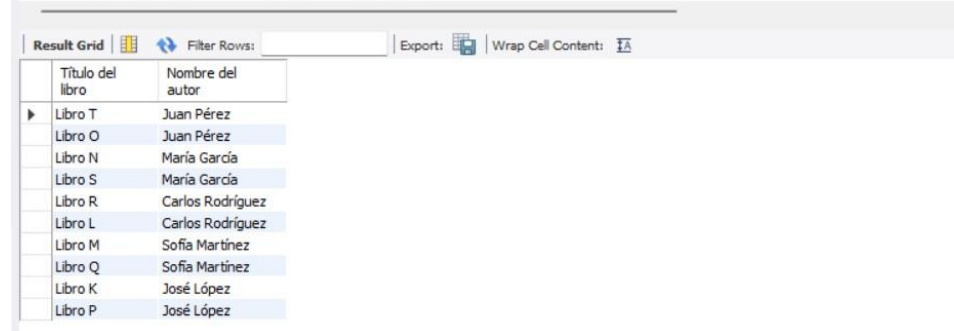
En esta base de datos, una vista es una consulta guardada en el sistema que puede ser llamada y utilizada como si fuera una tabla. Es decir, una vista permite acceder a los datos de varias tablas como si fuera una única tabla, con una estructura y un formato determinados. Las vistas se utilizan para simplificar las consultas y para proteger los datos originales, ya que las vistas pueden limitar los campos y las filas que se muestran al usuario.

Las vistas que considero son las más importantes en esta base de datos:

- una vista importante podría ser una que muestre toda la información sobre los libros y sus autores. Esta vista combinaría información de la tabla "libro" y la tabla "libro_autor".

Porque de esta forma podría mostrar los ISBN de los libros, títulos, nombres de los editoriales, y los nombres de los autores.

```
296  -----
297  -- ----#-VISTA de los libros y sus autores:-----
298  -----
299  • CREATE VIEW vista_libros_autores AS
300  SELECT libro.titulo AS 'Título del libro', autor.nombre AS 'Nombre del autor'
301  FROM libro
302  INNER JOIN libro_autor ON libro.ISBN = libro_autor.ISBN_libro
303  INNER JOIN autor ON libro_autor.id_autor = autor.id;
304  • SELECT * FROM vista_libros_autores;
```



The screenshot shows a database interface with a SQL query editor and a results grid. The query creates a view named 'vista_libros_autores' that joins the 'libro' and 'autor' tables. The results grid displays the data from this view, showing columns for 'Título del libro' and 'Nombre del autor'.

Título del libro	Nombre del autor
Libro T	Juan Pérez
Libro O	Juan Pérez
Libro N	María García
Libro S	María García
Libro R	Carlos Rodríguez
Libro L	Carlos Rodríguez
Libro M	Sofía Martínez
Libro Q	Sofía Martínez
Libro K	José López
Libro P	José López

- Otra vista importante podría ser una que muestre información sobre los clientes y los libros que han comprado. Esta vista combinaría información de las tablas "cliente" y "libro_cliente" para mostrar la cédula de los clientes y los ISBN de los libros que han comprado.

```

306  -- -----
307  -- ----#CONSULTAS  -VISTA de los clientes y los libros que han comprado:-----
308  -- -----
309  • CREATE VIEW vista_clientes_libros AS
310  SELECT cliente.nombre AS 'Nombre del cliente', libro.titulo AS 'Título del libro'
311  FROM cliente
312  INNER JOIN libro_cliente ON cliente.cedula = libro_cliente.id_cliente
313  INNER JOIN libro ON libro_cliente.ISBN_libro_cliente = libro.ISBN;
314  • SELECT * FROM vista_clientes_libros;

```

Result Grid		
Filter Rows: <input type="text"/>		
Export: <input type="button" value=""/>		
Wrap Cell Content: <input type="button" value=""/>		
	Nombre del cliente	Título del libro
▶	Juan	Libro L
	Juan	Libro T
	María	Libro M
	María	Libro S
	Pedro	Libro R
	Pedro	Libro K
	Sofía	Libro Q
	Carlos	Libro P
	Ana	Libro O
	Luis	Libro N

Actividad 2

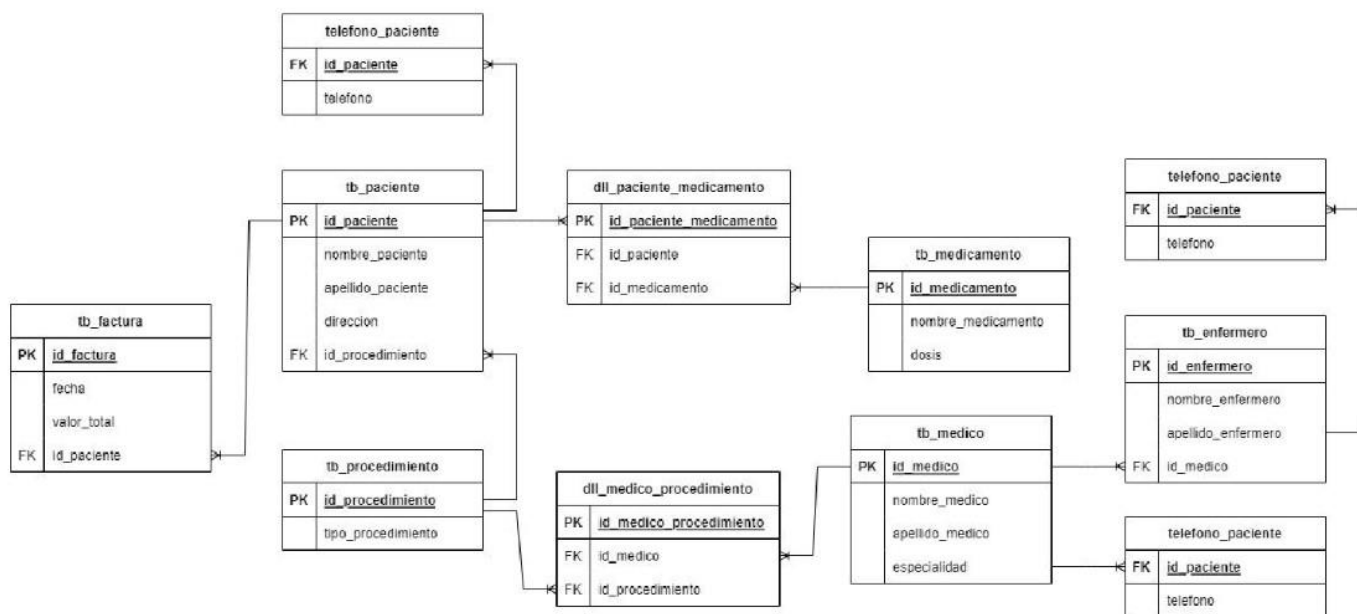
Para esta actividad me encargue de revisar la actividad del hospital entregada por un compañero.

La actividad tiene los siguientes requerimientos:

- Convierta el MR en una base de datos en MySQL utilizando sentencias SQL o el diagrama EER.
- Complete la información para las tablas realizadas con al menos 5 registros por tabla.

- realice una consulta que me permita conocer que medicamentos a tomado cada paciente y la dosis suministrada.
- realice una consulta que me permita conocer que enfermeros estuvieron en los procedimientos de los pacientes.
- Realice las tres vistas que considere sean las más importantes y explique el motivo de su selección.

El MR del compañero es el siguiente:

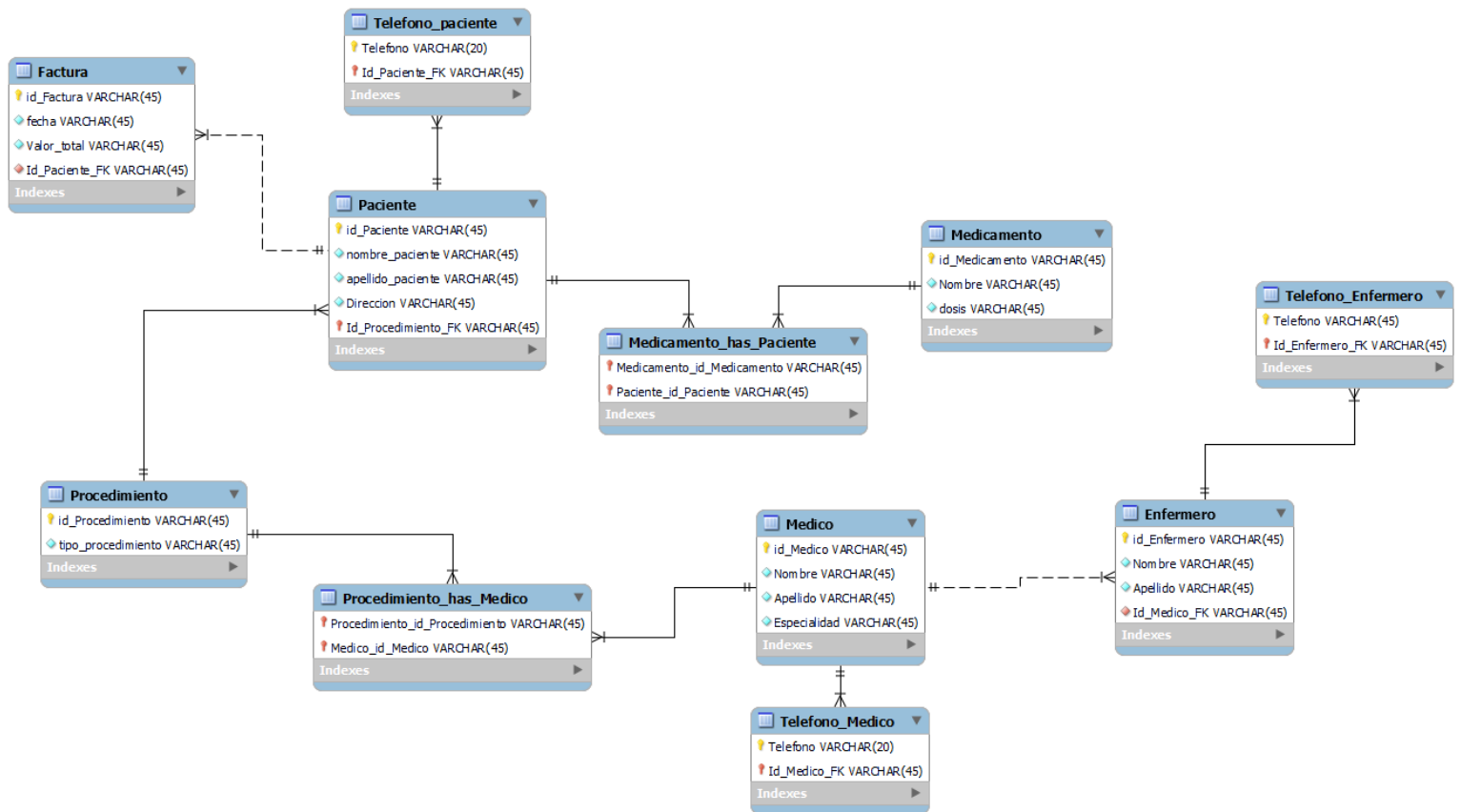


En este punto me di cuenta de que había varios errores en este Modelo relacional como por ejemplo algunas clases están mal nombradas y son repetidas como la de Telefono_Paciente.

Acá se debe aclarar o nombrar las clases acorde a lo que están relacionadas por ejemplo en vez de poner teléfono paciente al médico era Telefono_Medico para que sea correcto y lógico.

Otro error que encontré es que la tabla intermedia producto de la relación muchos a muchos entre tb_Medico y tb_Procedimiento esta mal ya que solo debe tener el id de la tabla de medico y otro id de la tabla procedimiento.

Después de notar estos errores procedí a hacer mi diagrama EER y quedo de la siguiente manera:



Después de esto aplique la ingeniería hacia adelante para poder crear el script:

Aquí adjunto algunas imágenes del resultado del script:

```

1  -- MySQL Workbench Forward Engineering
2
3  • SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
4  • SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
5  • SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE
6
7  -----
8  -- Schema mydb
9  -----
10
11 -----
12 -- Schema mydb
13 -----
14 • CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
15 • USE `mydb` ;
16
17 -----
18 -- Table `mydb`.`Medico`
19 -----
20 • CREATE TABLE IF NOT EXISTS `mydb`.`Medico` (
21     `id_Medico` VARCHAR(45) NOT NULL,
22     `Nombre` VARCHAR(45) NOT NULL,
23     `Apellido` VARCHAR(45) NOT NULL,
24     `Especialidad` VARCHAR(45) NOT NULL,
25     PRIMARY KEY (`id_Medico`))
26 ENGINE = InnoDB;
27
28 -----
29 -----
30
31 • CREATE TABLE IF NOT EXISTS `mydb`.`Procedimiento` (
32     `id_Procedimiento` VARCHAR(45) NOT NULL,
33     `tipo_procedimiento` VARCHAR(45) NOT NULL,
34     PRIMARY KEY (`id_Procedimiento`))
35 ENGINE = InnoDB;
36
37 -----
38 -----
39 -- Table `mydb`.`Procedimiento_has_Medico`
40 -----
41 -----
42 • CREATE TABLE IF NOT EXISTS `mydb`.`Procedimiento_has_Medico` (
43     `Procedimiento_id_Procedimiento` VARCHAR(45) NOT NULL,
44     `Medico_id_Medico` VARCHAR(45) NOT NULL,
45     PRIMARY KEY (`Procedimiento_id_Procedimiento`, `Medico_id_Medico`),
46     INDEX `fk_Procedimiento_has_Medico_Medico1_idx` (`Medico_id_Medico` ASC) VISIBLE,
47     INDEX `fk_Procedimiento_has_Medico_Procedimiento_idx` (`Procedimiento_id_Procedimiento` ASC) VISIBLE,
48     CONSTRAINT `fk_Procedimiento_has_Medico_Procedimiento`
49         FOREIGN KEY (`Procedimiento_id_Procedimiento`)
50         REFERENCES `mydb`.`Procedimiento` (`id_Procedimiento`)
51         ON DELETE NO ACTION
52         ON UPDATE NO ACTION,
53     CONSTRAINT `fk_Procedimiento_has_Medico_Medico1`
54         FOREIGN KEY (`Medico_id_Medico`)
55         REFERENCES `mydb`.`Medico` (`id_Medico`)
56         ON DELETE NO ACTION
57         ON UPDATE NO ACTION)
58 ENGINE = InnoDB;
59

```

```

Script_Hospital x
Limit to 1000 rows
175 REFERENCES `mydb`.`Medico` (`id_Medico`)
176 ON DELETE NO ACTION
177 ON UPDATE NO ACTION)
178 ENGINE = InnoDB;
179
180
181 -----
182 -- Table `mydb`.`Telefono_Enfermero`
183 -----
184 • CREATE TABLE IF NOT EXISTS `mydb`.`Telefono_Enfermero` (
185     `Telefono` VARCHAR(45) NOT NULL,
186     `Id_Enfermero_FK` VARCHAR(45) NOT NULL,
187     PRIMARY KEY (`Telefono`, `Id_Enfermero_FK`),
188     INDEX `Id_Enfermero_FK_idx` (`Id_Enfermero_FK` ASC) VISIBLE,
189     CONSTRAINT `Id_Enfermero_FK`
190     FOREIGN KEY (`Id_Enfermero_FK`)
191     REFERENCES `mydb`.`Enfermero` (`id_Enfermero`)
192     ON DELETE NO ACTION
193     ON UPDATE NO ACTION)
194 ENGINE = InnoDB;
195
196
197 • SET SQL_MODE=@OLD_SQL_MODE;
198 • SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
199 • SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
200

```

Después de esto empecé a poblar la base de datos del hospital con 5 registros por tabla:

Para la tabla Medico:

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows the database structure, including the 'mydb' database and its tables. The 'Script_Hospital' window displays the following SQL script:

```

200
201 #Poblando la base de datos
202
203 • INSERT INTO `mydb`.`Medico` (`id_Medico`, `Nombre`, `Apellido`, `Especialidad`)
204 VALUES
205     ('1', 'Juan', 'Perez', 'Cardiología'),
206     ('2', 'Maria', 'Gomez', 'Pediatria'),
207     ('3', 'Luis', 'Martinez', 'Cirugia'),
208     ('4', 'Ana', 'Ruiz', 'Oftalmología'),
209     ('5', 'Carlos', 'Garcia', 'Dermatología');
210 • SELECT *

```

Below the script, the 'Result Grid' window shows the data inserted into the 'Medico' table:

id_Medico	Nombre	Apellido	Especialidad
1	Juan	Perez	Cardiología
2	Maria	Gomez	Pediatria
3	Luis	Martinez	Cirugia
4	Ana	Ruiz	Oftalmología
5	Carlos	Garcia	Dermatología

Para la imagen anterior La instrucción **INSERT INTO** especifica los nombres de columna y sus valores correspondientes que deben insertarse en la tabla **Medico**. La instrucción **SELECT** recupera todas las columnas y filas de datos de la tabla **Medico**.

Hice lo mismo para las siguientes tablas:

Procedimiento:

The screenshot shows a SQL script in a text editor. The script contains an **INSERT INTO** statement for the **Procedimiento** table, followed by a **VALUES** clause with five rows of data. Below the script, a **Result Grid** displays the data being inserted.

```
213 • INSERT INTO `mydb`.`Procedimiento` (`id_Procedimiento`, `tipo_procedimiento`)
214 VALUES
215     ('1', 'Extracción de sangre'),
216     ('2', 'Radiografía'),
217     ('3', 'Colonoscopia'),
218     ('4', 'Cirugía de cataratas'),
219     ('5', 'Endoscopia');
220 • SELECT *
221 FROM `mydb`.`Procedimiento`;
222
223 • INSERT INTO `mydb`.`Procedimiento_has_Medico` (`Procedimiento_id_Procedimiento`, `Medico_id_Medico`)
```

id_Procedimiento	tipo_procedimiento
1	Extracción de sangre
2	Radiografía
3	Colonoscopia
4	Cirugía de cataratas
5	Endoscopia
NULL	NULL

Procedimiento_Medico:

The screenshot shows a SQL script in a text editor. The script contains an **INSERT INTO** statement for the **Procedimiento_has_Medico** table, followed by a **VALUES** clause with five rows of data. Below the script, a **Result Grid** displays the data being inserted.

```
222
223 • INSERT INTO `mydb`.`Procedimiento_has_Medico` (`Procedimiento_id_Procedimiento`, `Medico_id_Medico`)
224 VALUES
225     ('1', '1'),
226     ('2', '2'),
227     ('3', '3'),
228     ('4', '4'),
229     ('5', '5');
230 • SELECT *
231 FROM `mydb`.`Procedimiento_has_Medico`;
232
```

Procedimiento_id_Procedimiento	Medico_id_Medico
1	1
2	2
3	3
4	4
5	5
NULL	NULL

Paciente:

Navigator: SCHEMAS

Filter objects

- libreria_buscabre
 - Tables
 - Views
 - Stored Procedures
 - Functions
- libreria_db
 - mydb
 - enfermero
 - factura
 - medicamento
 - medicamento_has_p
 - medico
 - paciente
 - procedimiento
 - procedimiento_has_
 - telefono_enfermero
 - telefono_medico
 - telefono_paciente
 - Views

Administration Schemas

Information

No object selected

Script_Hospital

Limit to 1000 rows

```

228 ('4', '4'),
229 ('5', '5');
230 SELECT *
231 FROM `mydb`.`Procedimiento_has_Medico`;
232
233 INSERT INTO `mydb`.`Paciente` (`id_Paciente`, `nombre_paciente`, `apellido_paciente`, `Direccion`, `Id_Procedimiento_FK`)
234 VALUES
235 ('1', 'Juan', 'González', 'Calle 123', '1'),
236 ('2', 'María', 'Martínez', 'Avenida 456', '2'),
237 ('3', 'Pedro', 'García', 'Calle 789', '3'),
238 ('4', 'Ana', 'Rodríguez', 'Avenida 012', '4'),
239 ('5', 'Sofía', 'Sánchez', 'Calle 345', '5');
240 SELECT *
241 FROM `mydb`.`Paciente`;
242
243 TRUNCATE `mydb`.`Telefono_paciente` (`Telefono`, `Id_Paciente_Telefono`);
  
```

Result Grid

	id_Paciente	nombre_paciente	apellido_paciente	Direccion	Id_Procedimiento_FK
1	Juan	González	Calle 123	1	
2	María	Martínez	Avenida 456	2	
3	Pedro	García	Calle 789	3	
4	Ana	Rodríguez	Avenida 012	4	
5	Sofía	Sánchez	Calle 345	5	
6	NULL	NULL	NULL	NULL	

Telefono_Paciente:

Navigator: SCHEMAS

Filter objects

- libreria_buscabre
 - Tables
 - Views
 - Stored Procedures
 - Functions
- libreria_db
 - mydb
 - enfermero
 - factura
 - medicamento
 - medicamento_has_p
 - medico
 - paciente
 - procedimiento
 - procedimiento_has_
 - telefono_enfermero
 - telefono_medico
 - telefono_paciente
 - Views

Administration Schemas

Information

No object selected

Script_Hospital

Limit to 1000 rows

```

240 SELECT *
241 FROM `mydb`.`Paciente`;
242
243 INSERT INTO `mydb`.`Telefono_paciente` (`Telefono`, `Id_Paciente_Telefono`)
244 VALUES
245 ('555-1111', '1'),
246 ('555-2222', '2'),
247 ('555-3333', '3'),
248 ('555-4444', '4'),
249 ('555-5555', '5');
250 SELECT *
251 FROM `mydb`.`Telefono_paciente`;
252
253 INSERT INTO `mydb`.`Factura` (`id_Factura`, `fecha`, `Valor_total`, `Id_Paciente_Factura`)
254 VALUES
255 ('1', '2022-01-01', '100', '1');
  
```

Result Grid

	Telefono	Id_Paciente_Telefono
1	555-1111	1
2	555-2222	2
3	555-3333	3
4	555-4444	4
5	555-5555	5
6	NULL	NULL

Factura:

Navigator: SCHEMAS

Filter objects

- libreriaescalibre
 - Tables
 - Views
 - Stored Procedures
 - Functions
- libreriadb
 - mydb
 - Tables
 - enfermero
 - factura
 - medicamento
 - medicamento_has_p
 - medico
 - paciente
 - procedimiento
 - procedimiento_has_p
 - telefono_enfermero
 - telefono_medico
 - telefono_paciente
 - Views

Administration Schemas

Information

No object selected

Script_Hospital

Limit to 1000 rows

```

252
253 • INSERT INTO `mydb`.`Factura` (`id_Factura`, `fecha`, `Valor_total`, `Id_Paciente_Factura`)
254 VALUES
255     ('1', '2022-01-01', '100', '1'),
256     ('2', '2022-01-02', '200', '2'),
257     ('3', '2022-01-03', '300', '3'),
258     ('4', '2022-01-04', '400', '4'),
259     ('5', '2022-01-05', '500', '5');
260 • SELECT *
261 FROM `mydb`.`Factura`;
262
263 • INSERT INTO mydb.Medicamento (id_Medicamento, Nombre, dosis) VALUES
264     ('M001', 'Ibuprofeno', '400 mg'),
265     ('M002', 'Paracetamol', '500 mg'),
266     ('M003', 'Amoxicilina', '500 mg'),
267     ('M004', 'Omeprazol', '20 mg'),
268     ('M005', 'Diazepam', '5 mg');
  
```

Result Grid

id_Factura	fecha	Valor_total	Id_Paciente_Factura
1	2022-01-01	100	1
2	2022-01-02	200	2
3	2022-01-03	300	3
4	2022-01-04	400	4
5	2022-01-05	500	5
NULL	NULL	NULL	NULL

Medicamento:

Navigator: SCHEMAS

Filter objects

- libreriaescalibre
 - Tables
 - Views
 - Stored Procedures
 - Functions
- libreriadb
 - mydb
 - Tables
 - enfermero
 - factura
 - medicamento
 - medicamento_has_p
 - medico
 - paciente
 - procedimiento
 - procedimiento_has_p
 - telefono_enfermero
 - telefono_medico
 - telefono_paciente
 - Views

Administration Schemas

Information

No object selected

Script_Hospital

Limit to 1000 rows

```

261 FROM `mydb`.`Factura`;
262
263 • INSERT INTO mydb.Medicamento (id_Medicamento, Nombre, dosis) VALUES
264     ('M001', 'Ibuprofeno', '400 mg'),
265     ('M002', 'Paracetamol', '500 mg'),
266     ('M003', 'Amoxicilina', '500 mg'),
267     ('M004', 'Omeprazol', '20 mg'),
268     ('M005', 'Diazepam', '5 mg');
269 • SELECT *
270 FROM mydb.Medicamento;
271
272 • INSERT INTO Medicamento_has_Paciente (Medicamento_id_Medicamento, Paciente_id_Paciente)
273 VALUES ('M001', '1');
274 • INSERT INTO Medicamento_has_Paciente (Medicamento_id_Medicamento, Paciente_id_Paciente)
275 VALUES ('M002', '2');
276 • INSERT INTO Medicamento_has_Paciente (Medicamento_id_Medicamento, Paciente_id_Paciente)
  
```

Result Grid

id_Medicamento	Nombre	dosis
M001	Ibuprofeno	400 mg
M002	Paracetamol	500 mg
M003	Amoxicilina	500 mg
M004	Omeprazol	20 mg
M005	Diazepam	5 mg
NULL	NULL	NULL

Medicamento_paciente:

SCHEMAS

Filter objects

- libreriaescalibre
 - Tables
 - Views
 - Stored Procedures
 - Functions
- libreriadb
 - mydb
 - Tables
 - enfermero
 - factura
 - medicamento
 - medicamento_has_p
 - medico
 - paciente
 - procedimiento
 - procedimiento_has_
 - telefono_enfermero
 - telefono_medico
 - telefono_paciente
 - Views

Administration Schemas

Information

No object selected

```

270 FROM mydb.Medicamento;
271
272 • INSERT INTO Medicamento_has_Paciente (Medicamento_id_Medicamento, Paciente_id_Paciente)
273 VALUES ('M001', '1');
274 • INSERT INTO Medicamento_has_Paciente (Medicamento_id_Medicamento, Paciente_id_Paciente)
275 VALUES ('M002', '2');
276 • INSERT INTO Medicamento_has_Paciente (Medicamento_id_Medicamento, Paciente_id_Paciente)
277 VALUES ('M003', '3');
278 • INSERT INTO Medicamento_has_Paciente (Medicamento_id_Medicamento, Paciente_id_Paciente)
279 VALUES ('M004', '4');
280 • INSERT INTO Medicamento_has_Paciente (Medicamento_id_Medicamento, Paciente_id_Paciente)
281 VALUES ('M005', '5');
282 • SELECT *
283 FROM Medicamento_has_Paciente;
284
285 • INSERT INTO `Telefono_Medico` (`Telefono`, `Id_Medico_Telefono`)

```

Result Grid

Medicamento_id_Medicamento	Paciente_id_Paciente
M001	1
M002	2
M003	3
M004	4
M005	5
NULL	NULL

Telefono_Medico:

Navigator

SCHEMAS

Filter objects

- libreriaescalibre
 - Tables
 - Views
 - Stored Procedures
 - Functions
- libreriadb
 - mydb
 - Tables
 - enfermero
 - factura
 - medicamento
 - medicamento_has_p
 - medico
 - paciente
 - procedimiento
 - procedimiento_has_
 - telefono_enfermero
 - telefono_medico
 - telefono_paciente
 - Views

Administration Schemas

Information

No object selected

```

282 • SELECT *
283 FROM Medicamento_has_Paciente;
284
285 • INSERT INTO `Telefono_Medico` (`Telefono`, `Id_Medico_Telefono`)
286 VALUES
287 ('1234567890', '1'),
288 ('0987654321', '2'),
289 ('5554443333', '3'),
290 ('1112223333', '4'),
291 ('4445556666', '5');
292 • SELECT *
293 FROM `Telefono_Medico`;
294
295 • INSERT INTO `Enfermero` (`id_Enfermero`, `Nombre`, `Apellido`, `Id_Medico_FK`)
296 VALUES
297 ('1', 'Juan', 'Díaz', '1')

```

Result Grid

Telefono	Id_Medico_Telefono
1234567890	1
0987654321	2
5554443333	3
1112223333	4
4445556666	5
NULL	NULL

Telefono_Enfermero:

The screenshot shows a database management interface with a left-hand 'SCHEMAS' pane and a main 'Script_Hospital' editor. The 'SCHEMAS' pane shows a tree view with 'mydb' expanded, listing tables like 'enfermero', 'factura', 'medicamento', 'telefono_enfermero', etc. The 'Script_Hospital' editor contains SQL code for inserting data into the 'telefono_enfermero' table and selecting it. The 'Result Grid' at the bottom displays the results of the SELECT query.

SQL Script:

```
303 FROM `Enfermero`;  
304  
305 • INSERT INTO `Telefono_Enfermero` (`Telefono`, `Id_Enfermero_FK`)  
306 VALUES  
307 ('123-456-7890', '1'),  
308 ('234-567-8901', '2'),  
309 ('345-678-9012', '3'),  
310 ('456-789-0123', '4'),  
311 ('567-890-1234', '5');  
312 • SELECT *  
313 FROM `Telefono_Enfermero`;  
314  
315  
316 #CONSULTAS  
317 -----  
318 Describa los medicamentos que ha tomado cada paciente y la dosis
```

Result Grid:

Telefono	Id_Enfermero_FK
123-456-7890	1
234-567-8901	2
345-678-9012	3
456-789-0123	4
567-890-1234	5
NULL	NULL

CONSULTAS - Para conocer los medicamentos que ha tomado cada paciente y la dosis suministrada.

The screenshot shows a database management interface with a left-hand sidebar displaying a schema tree. The main area contains a SQL editor with a query that uses an INNER JOIN to retrieve patient names, medication names, and dosages. Below the editor, a 'Result Grid' displays the query results in a table format.

SQL Query:

```

312 • SELECT *
313 FROM `Telefono_Enfermero`;
314
315
316 #CONSULTAS
317 -----
318 -- Para conocer los medicamentos que ha tomado cada paciente y la dosis suministrada.
319 -----
320 • SELECT p.nombre_paciente, m.Nombre, m.dosis
321 FROM Paciente p
322 INNER JOIN Medicamento_has_Paciente mp ON p.id_Paciente = mp.Paciente_id_Paciente
323 INNER JOIN Medicamento m ON mp.Medicamento_id_Medicamento = m.id_Medicamento;
324
325 -----
326 -- Para conocer los enfermeros que estuvieron en los procedimientos de los pacientes
327 -----
  
```

Result Grid:

nombre_paciente	Nombre	dosis
Juan	Ibuprofeno	400 mg
María	Paracetamol	500 mg
Pedro	Amoxicilina	500 mg
Ana	Omeprazol	20 mg
Sofía	Diazepam	5 mg

En la imagen anterior se utiliza una cláusula INNER JOIN para recuperar información relacionada entre tres tablas: Paciente, Medicamento_has_Paciente, y Medicamento. La consulta selecciona los nombres de los pacientes (p.nombre_paciente), los nombres de los medicamentos (m.Nombre), y las dosis correspondientes a cada medicamento que se les ha recetado (m.dosis).

La cláusula INNER JOIN se utiliza para combinar las filas de las tablas relacionadas entre sí según las condiciones especificadas. En este caso, se utiliza para unir las filas de la tabla Paciente con las de la tabla Medicamento_has_Paciente, y luego unir las filas resultantes con las de la tabla Medicamento, utilizando los campos de relación (id_Paciente y Paciente_id_Paciente para unir la tabla Paciente y Medicamento_has_Paciente, y id_Medicamento y Medicamento_id_Medicamento para unir la tabla Medicamento_has_Paciente y Medicamento, respectivamente).

CONSULTAS - Para conocer los enfermeros que estuvieron en los procedimientos de los pacientes:

The screenshot shows a database management tool interface. On the left is a 'SCHEMAS' tree with folders for 'libreria', 'mydb', and 'Views'. The 'mydb' folder is expanded, showing tables like 'enfermero', 'factura', 'medicamento', 'medicamento_has_p', 'medico', 'paciente', 'procedimiento', 'procedimiento_has_', 'telefono_enfermero', 'telefono_medico', and 'telefono_paciente'. The main area displays a SQL script in a window titled 'Script_Hospital'. The script is a query to find nurses involved in patient procedures. Below the script, the 'Result Grid' shows the output of the query.

```
324
325
326 -- Para conocer los enfermeros que estuvieron en los procedimientos de los pacientes
327
328 SELECT P.nombre_paciente, P.apellido_paciente, PR.tipo_procedimiento, M.Nombre AS NombreMedico, M.Apellido AS Ap
329 FROM Paciente AS P
330 INNER JOIN Procedimiento AS PR ON P.Id_Procedimiento_FK = PR.id_Procedimiento
331 INNER JOIN Procedimiento_has_Medico AS PM ON PR.id_Procedimiento = PM.Procedimiento_id_Procedimiento
332 INNER JOIN Medico AS M ON PM.Medico_id_Medico = M.id_Medico
333 INNER JOIN Enfermero AS E ON M.id_Medico = E.Id_Medico_FK;
334
335
336 #VISTAS
337
338 -- Vista para mostrar la información del paciente, la factura correspondiente y el total de la factura
339
```

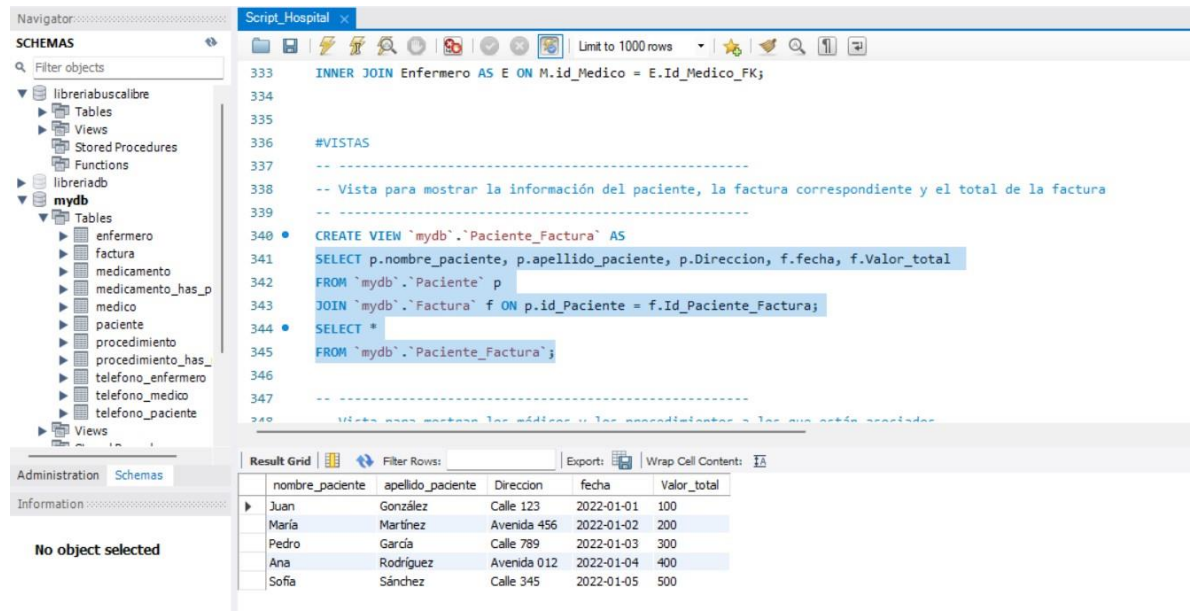
nombre_paciente	apellido_paciente	tipo_procedimiento	NombreMedico	ApellidoMedico	NombreEnfermero	ApellidoEnfermero
Juan	González	Extracción de sangre	Juan	Perez	Juan	Pérez
María	Martínez	Radiografía	María	Gomez	María	García
Pedro	García	Colonoscopia	Luis	Martínez	Luis	González
Ana	Rodríguez	Cirugía de cataratas	Ana	Ruiz	Ana	Martínez
Sofía	Sánchez	Endoscopia	Carlos	García	Carlos	Ruiz

En la imagen anterior se utiliza múltiples cláusulas INNER JOIN para recuperar información relacionada entre varias tablas: Paciente, Procedimiento, Procedimiento_has_Medico, Medico, y Enfermero. La consulta selecciona el nombre y apellido del paciente (P.nombre_paciente, P.apellido_paciente), el tipo de procedimiento (PR.tipo_procedimiento), el nombre y apellido del médico responsable (M.Nombre, M.Apellido), y el nombre y apellido del enfermero asignado (E.Nombre, E.Apellido).

La cláusula INNER JOIN se utiliza para combinar las filas de las tablas relacionadas entre sí según las condiciones especificadas. En este caso, se utiliza para unir las filas de la tabla Paciente con las de la tabla Procedimiento mediante el campo Id_Procedimiento_FK, luego unir las filas resultantes con las de la tabla Procedimiento_has_Medico utilizando el campo id_Procedimiento, luego unir las filas resultantes con las de la tabla Medico utilizando el campo Medico_id_Medico, y finalmente unir las filas resultantes con las de la tabla Enfermero utilizando el campo Id_Medico_FK. Al utilizar los alias AS en las tablas, se hace más fácil y legible el manejo de los nombres de estas.

- Realice las tres vistas que considere sean las más importantes y explique el motivo de su selección.

1. Vista para mostrar la información del paciente, la factura correspondiente y el total de la factura



The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including tables like 'enfermero', 'factura', 'medicamento', 'paciente', and 'procedimiento'. The main editor window shows the SQL script for creating the view 'Paciente_Factura'. The script includes an INNER JOIN between the 'Enfermero' table and the 'Paciente_Factura' view, followed by a comment and the view definition. The view definition uses a SELECT statement to retrieve patient information, factura details, and the total value. The 'Result Grid' at the bottom displays the data returned by the view.

```

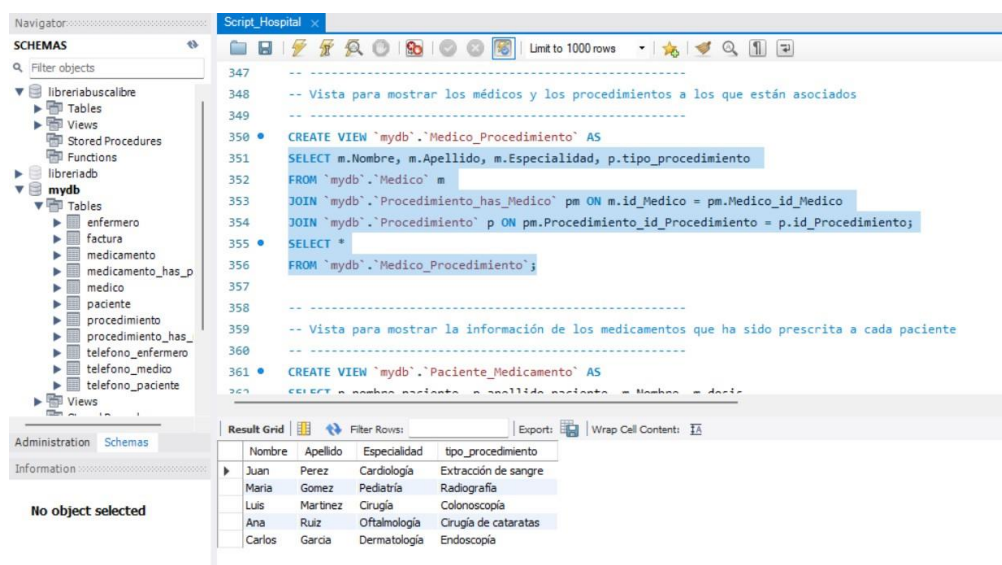
333 INNER JOIN Enfermero AS E ON M.id_Medico = E.Id_Medico_FK;
334
335
336 #VISTAS
337
338 -- Vista para mostrar la información del paciente, la factura correspondiente y el total de la factura
339
340 CREATE VIEW `mydb`.`Paciente_Factura` AS
341 SELECT p.nombre_paciente, p.apellido_paciente, p.Direccion, f.fecha, f.Valor_total
342 FROM `mydb`.`Paciente` p
343 JOIN `mydb`.`Factura` f ON p.id_Paciente = f.Id_Paciente_Factura;
344
345 SELECT *
346 FROM `mydb`.`Paciente_Factura`;
347
348
349

```

nombre_paciente	apellido_paciente	Direccion	fecha	Valor_total
Juan	González	Calle 123	2022-01-01	100
Maria	Martínez	Avenida 456	2022-01-02	200
Pedro	García	Calle 789	2022-01-03	300
Ana	Rodríguez	Avenida 012	2022-01-04	400
Sofía	Sánchez	Calle 345	2022-01-05	500

Me parece que esta vista es importante porque permite obtener información del paciente y la factura correspondiente en una sola tabla, con los campos más relevantes de ambas. Además, se incluye el total de la factura.

2. Vista para mostrar los médicos y los procedimientos a los que están asociados:



The screenshot shows the same database management tool interface. The main editor window displays the SQL script for creating the view 'Medico_Procedimiento'. The script includes a comment and the view definition. The view definition uses a SELECT statement to retrieve doctor information and the type of procedure. The 'Result Grid' at the bottom displays the data returned by the view.

```

347
348 -- Vista para mostrar los médicos y los procedimientos a los que están asociados
349
350 CREATE VIEW `mydb`.`Medico_Procedimiento` AS
351 SELECT m.Nombre, m.Apellido, m.Especialidad, p.tipo_procedimiento
352 FROM `mydb`.`Medico` m
353 JOIN `mydb`.`Procedimiento_has_Medico` pm ON m.id_Medico = pm.Medico_id_Medico
354 JOIN `mydb`.`Procedimiento` p ON pm.Procedimiento_id_Procedimiento = p.id_Procedimiento;
355
356 SELECT *
357 FROM `mydb`.`Medico_Procedimiento`;
358
359 -- Vista para mostrar la información de los medicamentos que ha sido prescrita a cada paciente
360
361 CREATE VIEW `mydb`.`Paciente_Medicamento` AS
362 SELECT p.nombre_paciente, p.apellido_paciente, m.Nombre, m.dosis

```

Nombre	Apellido	Especialidad	tipo_procedimiento
Juan	Perez	Cardiología	Extracción de sangre
Maria	Gomez	Pediatría	Radiografía
Luis	Martinez	Crugía	Colonoscopia
Ana	Ruiz	Oftalmología	Crugía de cataratas
Carlos	Garcia	Dermatología	Endoscopia

Me parece que esta vista es importante porque permite obtener la información de los médicos y los procedimientos a los que están asociados en una sola tabla.

3. Vista para mostrar la información de los medicamentos que ha sido prescrita a cada paciente:

The screenshot shows a database management interface with a 'Script_Hospital' window. The 'Schemas' pane on the left shows a database named 'mydb' with various tables and views. The main window displays SQL code for creating a view. The code includes joins for 'Procedimiento_has_Medico', 'Procedimiento', 'Paciente', 'Medicamento_has_Paciente', and 'Medicamento'. The view 'Paciente_Medicamento' is created to show patient names, surnames, medication names, and dosages. Below the code, a 'Result Grid' shows the data returned by the view.

```
353 JOIN `mydb`.`Procedimiento_has_Medico` pm ON m.id_Medico = pm.Medico_id_Medico
354 JOIN `mydb`.`Procedimiento` p ON pm.Procedimiento_id_Procedimiento = p.id_Procedimiento;
355 • SELECT *
356 FROM `mydb`.`Medico_Procedimiento`;
357
358 -----
359 -- Vista para mostrar la información de los medicamentos que ha sido prescrita a cada paciente
360 -----
361 • CREATE VIEW `mydb`.`Paciente_Medicamento` AS
362 SELECT p.nombre_paciente, p.apellido_paciente, m.Nombre, m.dosis
363 FROM `mydb`.`Paciente` p
364 JOIN `mydb`.`Medicamento_has_Paciente` mp ON p.id_Paciente = mp.Paciente_id_Paciente
365 JOIN `mydb`.`Medicamento` m ON mp.Medicamento_id_Medicamento = m.id_Medicamento;
366 • SELECT *
367 FROM `mydb`.`Paciente_Medicamento`;
>>>
```

nombre_paciente	apellido_paciente	Nombre	dosis
Juan	González	Ibuprofeno	400 mg
María	Martínez	Paracetamol	500 mg
Pedro	García	Amoxicilina	500 mg
Ana	Rodríguez	Omeprazol	20 mg
Sofía	Sánchez	Diazepam	5 mg

Me parece que esta vista es importante porque permite obtener la información de los medicamentos que ha sido prescrita a cada paciente en una sola tabla con sus dosis.

¿Qué le agregaría al modelo para dar más información y esa información cual sería?

Se podrían incluir detalles adicionales sobre los procedimientos, como la duración, el costo, los requisitos previos, etc. Además, se podría agregar información sobre la fecha de inicio y finalización del tratamiento de cada paciente. También se podría incluir información adicional sobre los medicamentos, como la frecuencia y duración de la prescripción, posibles efectos secundarios y restricciones de uso.