Iniciamos el ejercicio ejecutando el archivo sql para crear el schema, luego agregamos 5 autores, dejo el archivo del código usado para agregar estos 5 autores.

Procedemos a agregar 7 editoriales, 20 libros, 7 clientes, 10 libro_clientes, 10 libro_autor y 12 telefono_cliente

Luego de esto debemos hacer las consultas propuestas:

1. Conocer el nombre y la fecha de nacimiento de cada escritor:

Al ingresar el código de búsqueda:

```
1 •  select nombre,fecha_nacimiento
2    from autor;
```

Lo hice de esta manera ya que me gusta ver que voy a seleccionar y debajo de donde lo voy a hacer.

Nos arroja los siguientes datos:

nombre	fecha_nacimiento
Jorge Luis Borges	1899-08-24
Federico García Lorca	1898-06-05
Octavio Paz	1914-03-31
Julio Cortázar	1914-08-26
Gabriel García Márquez	1927-03-06

Mostrándonos lo que queríamos, los nombres de los autores y las fechas de nacimiento

2. la cantidad de libros diferentes vendidos:

Al ingresar el código de búsqueda:

```sql
SELECT l.ISBN, l.titulo, COUNT(DISTINCT lc.ISBN_libro_cliente) as cantidad_libros_distintos
FROM libro l
JOIN libro_cliente lc ON l.ISBN = lc.ISBN_libro_cliente
GROUP BY l.ISBN, l.titulo;
```

Utilizamos el alias de l y lc para interactuar con la tabla libro y libro_cliente, además de usar la función count que cuenta lo que se use como parámetro, nos piden lo que es distinto por lo que le indicamos al programa que solo llame los ISBN_libro_cliente que sean distintos, lo que nos resulta:

ISBN	titulo	cantidad_libros_distintos
9788397934	Final del juego	1
9788408094	Relato de un náufrago	1
9788437603	Gypsy Ballads	1
9788437609	Otras inquisiciones	1
9788437612	Poemas	1
9788437616	Laberinto	1
9788437618	El Aleph	1
9788458094	Cien años de soledad	1
9788472230	Obras completas	1
9788497938	La casa de Asterión	1

Por otro lado, solo podemos obtener el número con el siguiente código:

```sql
SELECT COUNT(distinct lc.ISBN_libro_cliente) as cantidad_libros_distintos
FROM libro_cliente lc;
```

Que nos devuelve:

cantidad_libros_distintos
10

3. el nombre del libro acompañado por su autor o sus autores:

Ejecutamos el código en sql:

```sql
select l.titulo as nombre_libro, group_concat(a.nombre) as nombre_autores
from libro l
join libro_autor la on l.ISBN=la.ISBN_libro
join autor a on la.id_autor=a.id
group by l.titulo;
```

Seleccionamos el titulo del libro, luego le decimos que si en los autores hay más de un valor los concatene por medio de una coma, luego le indicamos que solo nos traiga los libros que tengan asignado un autor, y así mismo de los autores solo nos traiga el nombre de los autores que tengan un libro asignado.

Obtenemos al ejecutar este código:

nombre_libro	nombre_autores
Cien años de soledad	Octavio Paz
El Aleph	Federico García Lorca
Final del juego	Jorge Luis Borges
Gypsy Ballads	Octavio Paz
La casa de Asterión	Gabriel García Márquez
Laberinto	Jorge Luis Borges
Obras completas	Julio Cortázar
Otras inquisiciones	Julio Cortázar
Poemas	Gabriel García Márquez
Relato de un náufrago	Federico García Lorca

4. el nombre de las editoriales que han logrado vender libros:

Diseñamos la consulta que haremos por medio de sql:

```sql
select l.nombre_editorial as nombre_editorial, group_concat(distinct(l.titulo)) as nombre_libros
from libro l
join libro_cliente lc on l.ISBN=lc.ISBN_libro_cliente
group by l.nombre_editorial;
```

En donde seleccionamos el nombre de la editorial, agrupamos por el título (Añadimos el distinct para que no nos repita títulos en la concatenación), que tomamos de libro y solo los que tengan un cliente asignado

```sql
CREATE VIEW libros AS
SELECT l.ISBN, l.titulo, group_concat(a.nombre) as nombre_autores
FROM libro l
join libro_autor la on l.ISBN=la.ISBN_libro
JOIN autor a ON a.id=la.id_autor
GROUP BY l.titulo, l.ISBN,a.nombre;
```

Se crea la vista que muestra los libros y sus autores, esta información puede ser importante para los compradores al saber que libros tiene la librería.

```sql
CREATE VIEW clientes AS
SELECT c.nombre AS nombre_cliente,
       GROUP_CONCAT(DISTINCT tc.numero ORDER BY tc.numero ASC SEPARATOR ', ') AS telefonos,
       GROUP_CONCAT(DISTINCT l.titulo ORDER BY l.titulo ASC SEPARATOR ', ') AS libros_comprados
FROM cliente c
JOIN telefono_cliente tc ON c.cedula = tc.cedula_cliente
JOIN libro_cliente lc ON c.cedula = lc.id_cliente
JOIN libro l ON lc.ISBN_libro_cliente = l.ISBN
GROUP BY c.cedula;
```

Se crea la vista para ver los clientes que han comprado libros, mostrando sus números de teléfono y el título de los libros que ha comprado.

Se crean las consultas solicitadas en el segundo punto:

1) realice una consulta que me permita conocer que enfermeros estuvieron en los procedimientos de los pacientes.

```sql
select e.nombre as nombre_enfermero, e.idenfermero as id_enfermero,mp.idprocedimiento_medico as id_procedimiento,p.nombre as r
from enfermero e
join medico_procedimiento mp on mp.idmedico_procedimiento=e.idMedico_enfermero
join paciente p on p.idProcedimiento_paciente=mp.idprocedimiento_medico
group by e.nombre, e.idenfermero,mp.idmedico_procedimiento,mp.idprocedimiento_medico, p.nombre;
```

2) realice una consulta que me permita conocer que medicamentos a tomado cada paciente y la dosis suministrada.

```sql
select p.nombre, group_concat(m.nombre) as medicamentos, group_concat(m.dosis) as dosis
from paciente p
join medicamento_paciente mp on p.idPaciente= mp.idPaciente_medicamento
join medicamento m on m.idMedicamento=mp.idMedicamento_paciente
group by p.nombre;
```

Ahora creamos la vista de los médicos, su especialidad y sus números telefónicos, esta vista nos permite conocer los médicos que tiene el hospital y sus especialidades para poder tomar decisiones en cuanto a procedimientos, por otro lado, también se añade el número para poder tener una forma de contacto sin tener que hacer otra consulta

```sql
create view medicos as
select m.idmedico,m.nombre, m.especialidad as especialidad, group_concat(mt.medico_telefono) as telefono_Medico
from medico m
join medico_telefono mt on mt.idmedico_telefono=m.idmedico
group by m.idmedico,m.nombre,m.especialidad,mt.medico_telefono
```

Creamos otra vista la que nos permite ver los pacientes que tenemos, el procedimiento que se le está haciendo, el médico a cargo y su enfermero esto nos permite controlar los pacientes y lo que necesitan, en dado caso poder buscar un reemplazo de medico o asignación de más enfermeros

```sql
create view pacientes as
select p.idPaciente,p.nombre, pr.tipo, group_concat(m.nombre) as nombre_medico,group_concat(e.nombre) as nombre_enfermero
from paciente p
join procedimiento pr on pr.idProcedimiento=p.idProcedimiento_paciente
join medico_procedimiento mp on mp.idprocedimiento_medico=p.idProcedimiento_paciente
join medico m on m.idmedico=mp.idprocedimiento_medico
join enfermero e on e.idMedico_enfermero=m.idmedico
group by p.idPaciente;
```

Creamos la ultima vista que nos permite ver los pacientes las facturas, los medicamentos y sus dosis. Esto nos permite hacer un análisis de las tendencias en los pacientes, medicamentos, las dosis y su costo

```sql
create view facturas as
select p.idPaciente, p.nombre as nombre_paciente, f.idfactura, f.valorTotal,m.nombre as nombre_medicamento,m.dosis
from paciente p
join factura f on f.idPaciente_factura=p.idPaciente
join medicamento_paciente mp on mp.idPaciente_medicamento=p.idPaciente
join medicamento m on mp.idPaciente_medicamento=m.idMedicamento
group by p.idPaciente,f.idfactura;
```

Seguimos con el tercer punto que nos indica crear cuatro procedimientos para poder agregar, actualizar, consultar y borrar

1) agregar:

```sql
DELIMITER //
CREATE PROCEDURE insertar_autor(
    IN id VARCHAR(10),
    IN nacionalidad VARCHAR(20),
    IN fecha_nacimiento varchar(45),
    IN nombre varchar(45)
)
BEGIN
    INSERT INTO autor (id,fecha_nacimiento,nacionalidad,nombre)
    VALUES (id,fecha_nacimiento,nacionalidad,nombre);
END //
DELIMITER ;
```

2) actualizar:

```
DELIMITER //
CREATE PROCEDURE actualizar_autor(
    IN id_a VARCHAR(10),
    IN nacionalidad_a VARCHAR(20),
    IN fecha_nacimiento_a varchar(45),
    IN nombre_a varchar(45)
)
BEGIN
    update autor set fecha_nacimiento=fecha_nacimiento_a,nacionalidad=nacionalidad_a,nombre=nombre_a WHERE id=id_a;
END //
DELIMITER ;
```

3) Consultar:

```
DELIMITER //
CREATE PROCEDURE consultar_autor(
    IN nombre_autor VARCHAR(45)
)
BEGIN
    SELECT *
    FROM autor
    WHERE nombre = nombre_autor;
END //
DELIMITER ;
```

4) Eliminar:

```
DELIMITER //
CREATE PROCEDURE consultar_autor(
    IN nombre_autor VARCHAR(45)
)
BEGIN
    SELECT *
    FROM autor
    WHERE nombre = nombre_autor;
END //
DELIMITER ;
```

Se crea una tabla que guarda los cambios y se crea un trigger para saber quien ha manipulado la base de datos de la librería

```sql
CREATE TABLE control_de_cambios_libreria (
  usuario VARCHAR(50),
  accion VARCHAR(50),
  fecha DATETIME
);
DELIMITER //
CREATE TRIGGER agregar_registro AFTER INSERT ON autor
FOR EACH ROW
BEGIN
  INSERT INTO control_de_cambios_libreria (usuario, accion, fecha)
  VALUES (USER(), "agregó un registro en la tabla autor", now());
END; //
DELIMITER ;
DELIMITER //
CREATE TRIGGER eliminar_registro AFTER DELETE ON autor
FOR EACH ROW
BEGIN
  INSERT INTO control_de_cambios_libreria (usuario, accion, fecha)
  VALUES (USER(), 'eliminó un registro en la tabla autor', NOW());
END; //
```

Se crean los procedimientos para la base de datos del hospital:

1) Agregar:

```sql
DELIMITER //
CREATE PROCEDURE insertar_paciente(
    IN nombre VARCHAR(20),
    IN direccion varchar(45),
    IN idProcedimiento varchar(45)
)
BEGIN
    INSERT INTO paciente (nombre,direccion,idProcedimiento_paciente)
    VALUES (nombre,direccion,idProcedimiento);
END //
DELIMITER ;
```

2) Actualizar:

```
DELIMITER //
CREATE PROCEDURE actualizar_paciente(
    IN id_a int,
    IN nombre_a VARCHAR(20),
    IN direccion_a varchar(45),
    IN idProcedimiento_a varchar(45)
)
BEGIN
    update paciente set nombre=nombre_a,direccion=direccion_a,idProcedimiento_paciente=idProcedimiento_a WHERE idP
END //
DELIMITER ;
```

3) Consultar:

```
DELIMITER //
CREATE PROCEDURE consultar_paciente(
    IN nombre_paciente VARCHAR(20)
)
BEGIN
    SELECT *
    FROM paciente
    WHERE nombre = nombre_paciente;
END //
DELIMITER ;
```

4) Eliminar:

```
DELIMITER //
CREATE PROCEDURE eliminar_paciente(
    IN id_paciente int
)
BEGIN
    delete from paciente where idPaciente=id_paciente;
END //
DELIMITER ;
```

Creamos la tabla que revisa las inserciones y eliminaciones en la tabla pacientes:

```
CREATE TABLE control_de_cambios_hospital(
  usuario VARCHAR(50),
  accion VARCHAR(50),
  fecha DATETIME
);
DELIMITER //
CREATE TRIGGER agregar_registro AFTER INSERT ON paciente
FOR EACH ROW
BEGIN
  INSERT INTO control_de_cambios_libreria (usuario, accion, fecha)
  VALUES (USER(), "agregó un registro en la tabla autor", now());
END; //
DELIMITER ;
DELIMITER //
CREATE TRIGGER eliminar_registro AFTER DELETE ON paciente
FOR EACH ROW
BEGIN
  INSERT INTO control_de_cambios_libreria (usuario, accion, fecha)
  VALUES (USER(), 'eliminó un registro en la tabla autor', NOW());
END; //
DELIMITER ;
```

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

Yo agregaría vistas para tener una información general del negocio, información que me permita analizar y tomar decisiones con respecto al futuro del negocio y las tendencias internas de los procesos y trabajadores