```sql
/*

 Ejercicio 1: ¿Cuántos registros tiene cada tabla? Responder en un solo resultado.

*/

SELECT 'autor' AS nombre_tabla, COUNT(*) AS cuenta FROM autor
UNION
SELECT 'categoria', COUNT(*) FROM categoria
UNION
SELECT 'inventario', COUNT(*) FROM inventario
UNION
SELECT 'libro', COUNT(*) FROM libro
UNION
SELECT 'pais', COUNT(*) FROM pais
UNION
SELECT 'prestamo', COUNT(*) FROM prestamo
UNION
SELECT 'usuario', COUNT(*) FROM usuario;


/*

 Ejercicio 2: ¿Cuántos autores se tienen por cada nacionalidad en la biblioteca?

*/

SELECT pais.nombre, COUNT(*)
FROM autor
INNER JOIN pais ON autor.nacionalidad_id = pais.id
GROUP BY pais.nombre
ORDER BY pais.nombre;


/*

 Ejercicio 3: ¿Cuántos usuarios se tienen por cada nacionalidad en la biblioteca?

*/
WITH usuarios_por_pais AS (
    SELECT nacionalidad_id, COUNT(*) AS cuenta_usuarios
    FROM usuario
    GROUP BY nacionalidad_id
)

SELECT pais.nombre, usuarios_por_pais.cuenta_usuarios
FROM usuarios_por_pais
INNER JOIN pais ON usuarios_por_pais.nacionalidad_id = pais.id;

/*

 Ejercicio 4: ¿Cuáles son los detalles del autor más viejo en la bilbioteca? ¿Cuáles son los detalles del autor
 más joven?

*/

SELECT *
FROM autor
WHERE autor.fecha_nacimiento = (SELECT MIN(fecha_nacimiento) FROM autor)

UNION

SELECT *
FROM autor
WHERE autor.fecha_nacimiento = (SELECT MAX(fecha_nacimiento) FROM autor);

```

```sql
/*

 Ejercicio 5: En promedio, ¿cuántos ejemplares se compran (por parte de la biblioteca) por libro?

*/
WITH libros_agrupados AS (
    SELECT libro_id, COUNT(*) AS cuenta_libros
    FROM inventario
    GROUP BY libro_id
)

SELECT AVG(cuenta_libros)
FROM libros_agrupados;


/*

 Ejercicio 6: La biblioteca desea tener un reporte de todos los ejemplares que se han perdido. Incluir en el reporte
 el ISBN, título y autor de libro así como su último usuario.

*/
WITH CTE_prestamo AS (
    SELECT *
    FROM prestamo
    WHERE prestamo.fecha_retorno IS NULL
)
SELECT libro.isbn,
       libro.titulo,
       autor.nombre,
       autor.apellido,
       usuario.nombre,
       usuario.apellido
FROM CTE_prestamo
--INNER JOIN inventario ON inventario.libro_id = CTE_prestamo.inventario_id
INNER JOIN inventario ON inventario.id = CTE_prestamo.inventario_id
INNER JOIN libro ON inventario.libro_id = libro.id
INNER JOIN autor ON libro.autor_id = autor.id
INNER JOIN usuario ON CTE_prestamo.usuario_id = usuario.id;

/*

 Ejercicio 7: ¿Cuantos préstamos se tienen por cada nacionalidad de usuario?

*/

SELECT pais.nombre, COUNT(*)
FROM prestamo
INNER JOIN usuario ON prestamo.usuario_id = usuario.id
INNER JOIN pais ON usuario.nacionalidad_id = pais.id
GROUP BY pais.nombre
ORDER BY COUNT(*) DESC;

/*

 Ejercicio 8: ¿De qué nacionalidad son los autores de los cuáles se han prestado más libros?

*/

SELECT autor.nacionalidad_id, pais.nombre, COUNT(*)
FROM prestamo
INNER JOIN inventario ON prestamo.inventario_id = inventario.id
INNER JOIN libro ON inventario.libro_id = libro.id
INNER JOIN autor ON libro.autor_id = autor.id
INNER JOIN pais ON autor.nacionalidad_id = pais.id
```

```sql
GROUP BY autor.nacionalidad_id, pais.nombre
ORDER BY COUNT(*) DESC;

/*

 Ejercicio 9: ¿Cuál es el hash de contraseña más común entre los usuarios?

*/

SELECT contrasena, COUNT(*)
FROM usuario
GROUP BY contrasena
HAVING COUNT(*) > 1
ORDER BY COUNT(*) DESC;

/*

 Ejercicio 10: Ejecuta un ataque de tipo rainbow table. Con los hashes precomputados en la tabla 'rainbow',
 proveer los correos de usuario y contraseñas (no hashes) de los usuarios vulerables a este hackeo.

*/

SELECT usuario.correo, rainbow.contrasena
FROM usuario
INNER JOIN rainbow ON usuario.contrasena = rainbow.hash
ORDER BY correo;

/*

 Ejercicio 11: Proporciona los detalles de los usuarios que no se encuentran vulnerados por el rainbow table attack.
 Utiliza un LEFT JOIN o RIGHT JOIN para hacerlo.

*/

SELECT usuario.*
FROM usuario
LEFT JOIN rainbow ON usuario.contrasena = rainbow.hash
WHERE rainbow.id IS NULL
ORDER BY usuario.id;

/*

 Ejercicio 12: La biblioteca cree que el COVID19 se puede contagiar a través de libros. Proporciona una lista de todos
 los pares de usuarios que hayan pedido en préstamo el mismo ejemplar con menos de 4 días de diferencia entre la
 fecha de devolución del primer usuario y la fecha de inicio del segundo usuario. Recordar que el usuario que
 tuvo primero el libro es el potencial contagiador, el que lo pidió después es el potencial contagiado. Favor de omitir
 los casos en que el contagiador es el mismo usuario que el contagiado.

*/

SELECT contagiador.inventario_id,
       contagiador.usuario_id AS contagiador_id,
       contagiado.usuario_id AS contagiado_id,
       contagiador.fecha_retorno,
       contagiado.fecha_inicio,
       contagiado.fecha_inicio - contagiador.fecha_retorno AS diferencia_dias
FROM prestamo contagiador
INNER JOIN prestamo contagiado
    ON contagiador.inventario_id = contagiado.inventario_id
    AND contagiado.fecha_inicio > contagiador.fecha_retorno
```

```sql
    AND (contagiado.fecha_inicio - contagiador.fecha_retorno) < 4
    AND contagiador.usuario_id != contagiado.usuario_id
ORDER BY inventario_id, contagiador_id, contagiado_id;

/*

 Ejercicio 13: ¿Cuál es el libro más solicitado?

*/

SELECT inventario.libro_id, libro.titulo, COUNT(*)
FROM prestamo
INNER JOIN inventario ON prestamo.inventario_id = inventario.id
INNER JOIN libro ON inventario.libro_id = libro.id
GROUP BY inventario.libro_id, libro.titulo
ORDER BY COUNT(*) DESC;

/*

 Ejercicio 14: ¿Cuál es el autor cuyos ejemplares se han perdido más? ¿Cuántos han sido?

*/

WITH prestamos_perdidos AS (
   SELECT *
    FROM prestamo
    WHERE fecha_retorno IS NULL
        AND fecha_fin < NOW()
),

recuento_prestamos_perdidos_por_autor AS (
    SELECT autor.id, autor.nombre, autor.apellido, COUNT(*)
    FROM prestamos_perdidos
    INNER JOIN inventario ON prestamos_perdidos.inventario_id = inventario.id
    INNER JOIN libro ON inventario.libro_id = libro.id
    INNER JOIN autor ON libro.autor_id = autor.id
    GROUP BY autor.id, autor.nombre, autor.apellido
)

SELECT *
FROM autor
WHERE id IN (
    SELECT id
    FROM recuento_prestamos_perdidos_por_autor
    WHERE "count" = (SELECT MAX("count") FROM recuento_prestamos_perdidos_por_autor)
);

/*

 Ejercicio 15: Obtener un reporte por usuario (usuario_id, libro_id) donde se especifique los libros distintos que
 haya solicitado alguna vez.

*/

SELECT DISTINCT prestamo.usuario_id, inventario.libro_id
FROM prestamo
INNER JOIN inventario ON prestamo.inventario_id = inventario.id
ORDER BY prestamo.usuario_id, inventario.libro_id;


/*

 Ejercicio 16: ¿Hay algún libro que no se haya prestado ninguno de sus ejemplares?

*/
```

```sql

SELECT *
FROM libro WHERE id IN (
    SELECT libro.id
    FROM libro
    INNER JOIN inventario ON libro.id = inventario.libro_id
    LEFT JOIN prestamo ON inventario.id = prestamo.inventario_id
    GROUP BY libro.id
    HAVING COUNT(prestamo.id) = 0
);

/*

 Ejercicio 17: ¿De los préstamos que se regresan tarde (pero que sí se regresan) cuál es el promedio, mínimo y
 máximo de devolución tardía?

*/

SELECT ROUND(AVG(fecha_retorno - prestamo.fecha_fin), 2),
       MIN(fecha_retorno - prestamo.fecha_fin),
       MAX(fecha_retorno - prestamo.fecha_fin)
FROM prestamo
WHERE fecha_retorno IS NOT NULL
    AND fecha_retorno > fecha_fin;

/*

 Ejercicio 18: ¿Cuántos ejemplares de libros se han perdido por año, usando la fecha de inicio como referencia?

*/

WITH ejemplares_no_devueltos AS (
    SELECT *
    FROM prestamo
    WHERE fecha_retorno IS NULL
        AND NOW() > fecha_fin
)

SELECT EXTRACT(YEAR FROM fecha_inicio) AS "año",
       COUNT(*) AS conteo
FROM ejemplares_no_devueltos
GROUP BY EXTRACT(YEAR FROM fecha_inicio)
ORDER BY "año" DESC;

/*

 Ejercicio 19: La biblioteca desea comenzar un sistema de recomendaciones de libros. Genera un reporte por
 usuario con libros recomendados para el (usuario_id, libro_id). Este sistema debe funcionar usando como base libros
 prestados a otros usuarios que hayan pedido en préstamo algún libro en común con el usuario en cuestión. Favor de
 omitir de la lista de recomendaciones, los libros que alguna vez el usuario ya haya leído.

*/

-- tabla extendida
WITH prestamo_usuario AS (
    SELECT DISTINCT prestamo.usuario_id, inventario.libro_id
    FROM prestamo
    INNER JOIN inventario
        ON prestamo.inventario_id = inventario.id
),

```

```sql
-- sacar los pares de usuarios que tienen gustos en común
usuarios_gusto_comun AS (
    SELECT DISTINCT original.usuario_id original_id,
                    recomendador.usuario_id recomendador_id
    FROM prestamo_usuario original
    INNER JOIN prestamo_usuario recomendador
        ON original.libro_id = recomendador.libro_id
        AND original.usuario_id != recomendador.usuario_id
    WHERE original.usuario_id < recomendador.usuario_id
)

SELECT DISTINCT original.usuario_id,
                recomendador.libro_id
FROM prestamo_usuario AS original
INNER JOIN usuarios_gusto_comun
    ON original.usuario_id = usuarios_gusto_comun.original_id
INNER JOIN prestamo_usuario AS recomendador
    ON usuarios_gusto_comun.recomendador_id = recomendador.usuario_id
    AND recomendador.libro_id != original.libro_id
ORDER BY original.usuario_id, recomendador.libro_id;

```