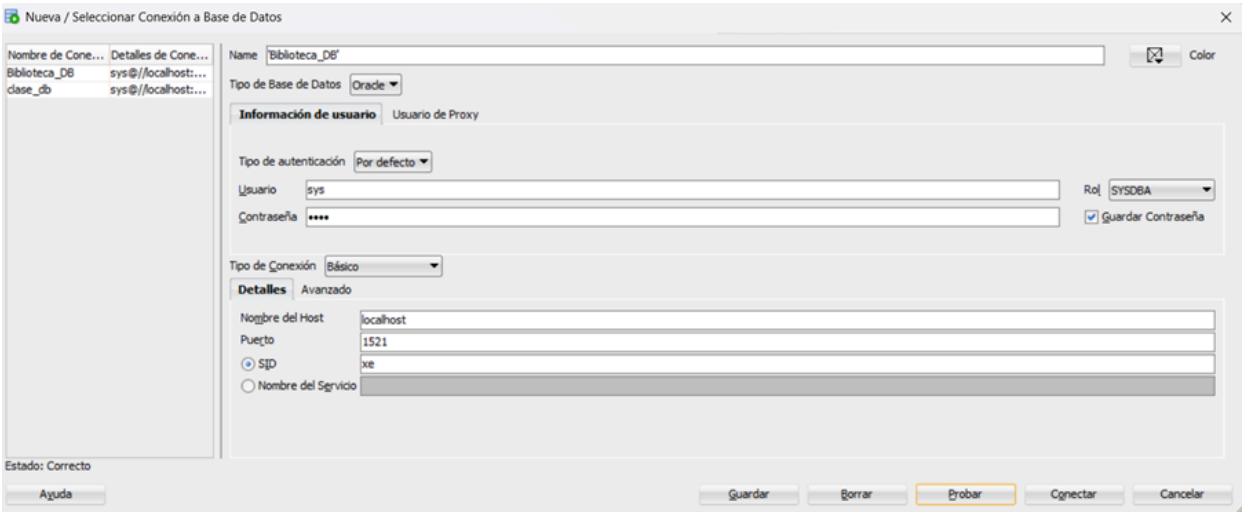
Ejercicios de Base de datos

50 ejercicios sobre Bases de Datos en Oracle 21C XE

1. Crea una nueva conexión en SQL Developer para conectarte a Oracle Database 21C XE. Llamala 'Biblioteca_DB'.



//—----------

2. Crear la tabla Libros con las siguientes columnas y restricciones:

LibroID (NUMBER, PK, NOT NULL)

Titulo (VARCHAR2(100), NOT NULL)

Autor (VARCHAR2(100), NOT NULL)

Genero (VARCHAR2(50))

FechaPublicacion (DATE)

Precio (NUMBER(10,2), DEFAULT 0)

```
CREATE TABLE Libros (
    LibroID NUMBER PRIMARY KEY NOT NULL,
    Titulo VARCHAR2(100) NOT NULL,
    Autor VARCHAR2(100) NOT NULL,
    Genero VARCHAR2(50),
    FechaPublicacion DATE
```

);

//----------------------------------

ALTER TABLE Libros

ADD Precio NUMBER(10,2) DEFAULT 0;

// modifico la tabla porque se me ha olvidado completarlo bien

//----------------------------------

3. Inserta al menos 5 registros en la tabla 'Libros'.

LibroID	Titulo	Autor	Genero	FechaPublicacion	Precio
1	El Quijote	Miguel de Cervantes	Clásico	1605-01-16	25.50
2	1984	George Orwell	Distopía	1949-06-08	15.99
3	Cien Años de Soledad	Gabriel García Márquez	Realismo Mágico	1967-05-30	20.75
4	El Hobbit	J.R.R. Tolkien	Fantasía	1937-09-21	18.20
5	Orgullo y Prejuicio	Jane Austen	Romance	1813-01-28	10.50

INSERT INTO Libros (LibroID, Titulo, Autor, Genero, FechaPublicacion)

VALUES (1, 'El Quijote', 'Miguel de Cervantes', 'Clásico', TO_DATE('1605-01-16', 'YYYY-MM-DD'));

INSERT INTO Libros (LibroID, Titulo, Autor, Genero, FechaPublicacion)

VALUES (2, '1984', 'George Orwell', 'Distopía', TO_DATE('1949-06-08', 'YYYY-MM-DD'));

INSERT INTO Libros (LibroID, Titulo, Autor, Genero, FechaPublicacion)

VALUES (3, 'Cien Años de Soledad', 'Gabriel García Márquez', 'Realismo Mágico', TO_DATE('1967-05-30', 'YYYY-MM-DD'));

INSERT INTO Libros (LibroID, Titulo, Autor, Genero, FechaPublicacion)

VALUES (4, 'El Hobbit', 'J.R.R. Tolkien', 'Fantasía', TO_DATE('1937-09-21', 'YYYY-MM-DD'));

INSERT INTO Libros (LibroID, Titulo, Autor, Genero, FechaPublicacion)

VALUES (5, 'Orgullo y Prejuicio', 'Jane Austen', 'Romance', TO_DATE('1813-01-28', 'YYYY-MM-DD'));

//----------------------------------

UPDATE Libros

SET Precio = 23.50

WHERE LibroID = 1;

UPDATE Libros

SET Precio = 15.99

WHERE LibroID = 2;

UPDATE Libros

SET Precio = 20.75

WHERE LibroID = 3;

UPDATE Libros

SET Precio = 18.20

WHERE LibroID = 4;

UPDATE Libros

SET Precio = 10.50

WHERE LibroID = 5;

//inserto los datos que no pude poner antes porque no tenia puesta la columna

//----------------------------------

4. Crear la tabla Clientes con las siguientes columnas y restricciones:

ClienteID (NUMBER, PK, NOT NULL)
Nombre (VARCHAR2(50), NOT NULL)
Direccion (VARCHAR2(100))
Telefono (VARCHAR2(15))
FechaRegistro (DATE, DEFAULT SYSDATE)
//----------------------------------

CREATE TABLE Clientes (
  ClienteID NUMBER PRIMARY KEY NOT NULL,

```
    Nombre VARCHAR2(50) NOT NULL,
    Direccion VARCHAR2(100),
    Telefono VARCHAR2(15),
    FechaRegistro DATE DEFAULT SYSDATE
);
//------------------------------
```

5. Inserta al menos 5 registros en la tabla 'Usuarios'.

ClienteID	Nombre	Direccion	Telefono	FechaRegistro
1	Ana López	Calle Falsa 123	600123456	2024-01-01
2	Juan Pérez	Av. Siempre Viva 45	690987654	2024-02-01
3	María González	Plaza Mayor s/n	610456789	2024-03-01
4	Luis Rodríguez	Calle Real 77	620123789	2024-04-01
5	Carmen Jiménez	Barrio Antiguo 32	630654321	2024-05-01

```
//------------------------------
INSERT INTO Clientes (ClienteID, Nombre, Direccion, Telefono, FechaRegistro)
VALUES (1, 'Ana López', 'Calle Falsa 123', '600123456', TO_DATE('2024-01-01', 'YYYY-MM-DD'));
INSERT INTO Clientes (ClienteID, Nombre, Direccion, Telefono, FechaRegistro)
VALUES (2, 'Juan Pérez', 'Av. Siempre Viva 45', '690987654', TO_DATE('2024-02-01', 'YYYY-MM-DD'));
INSERT INTO Clientes (ClienteID, Nombre, Direccion, Telefono, FechaRegistro)
VALUES (3, 'María González', 'Plaza Mayor s/n', '610456789', TO_DATE('2024-03-01', 'YYYY-MM-DD'));
INSERT INTO Clientes (ClienteID, Nombre, Direccion, Telefono, FechaRegistro)
VALUES (4, 'Luis Rodríguez', 'Calle Real 77', '620123789', TO_DATE('2024-04-01', 'YYYY-MM-DD'));
INSERT INTO Clientes (ClienteID, Nombre, Direccion, Telefono, FechaRegistro)
VALUES (5, 'Carmen Jiménez', 'Barrio Antiguo 32', '630654321', TO_DATE('2024-05-01', 'YYYY-MM-DD'));
//---------------------------------
```

6. Crea una tabla llamada 'Prestamos' con las siguientes columnas:
   - ID_PRESTAMO (PK, NUMBER, NOT NULL)
   - ID_USUARIO (FK, NUMBER, NOT NULL)
   - ID_LIBRO (FK, NUMBER, NOT NULL)
   - FECHA_PRESTAMO (DATE, NOT NULL)
   - FECHA_DEVOLUCION (DATE)

```
//---------------------------------

CREATE TABLE Prestamos (
    ID_PRESTAMO NUMBER PRIMARY KEY NOT NULL,
    ID_USUARIO NUMBER NOT NULL,
```

```
   ID_LIBRO NUMBER NOT NULL,
   FECHA_PRESTAMO DATE NOT NULL,
   FECHA_DEVOLUCION DATE,
     CONSTRAINT FK_ID_USUARIO FOREIGN KEY (ID_USUARIO) REFERENCES Clientes (ClienteID),
   CONSTRAINT FK_ID_LIBRO FOREIGN KEY (ID_LIBRO) REFERENCES Libros (LibroID)
);
//--------------------------------------
```

7. Inserta al menos 5 registros en la tabla 'Prestamos'.

PrestamoID	LibroID	ClienteID	FechaPrestamo	FechaDevolucion
1	1	1	2024-01-05	2024-01-20
2	2	2	2024-02-10	2024-02-25
3	3	3	2024-03-15	NULL
4	4	4	2024-04-01	NULL
5	5	5	2024-05-10	2024-05-25

```
//--------------------------------------

INSERT INTO Prestamos (ID_PRESTAMO, ID_LIBRO, ID_USUARIO, FECHA_PRESTAMO, FECHA_DEVOLUCION)
VALUES (1, 1, 1, TO_DATE('2024-01-05', 'YYYY-MM-DD'), TO_DATE('2024-01-20', 'YYYY-MM-DD'));
INSERT INTO Prestamos (ID_PRESTAMO, ID_LIBRO, ID_USUARIO, FECHA_PRESTAMO, FECHA_DEVOLUCION)
VALUES (2, 2, 2, TO_DATE('2024-02-10', 'YYYY-MM-DD'), TO_DATE('2024-02-25', 'YYYY-MM-DD'));
INSERT INTO Prestamos (ID_PRESTAMO, ID_LIBRO, ID_USUARIO, FECHA_PRESTAMO, FECHA_DEVOLUCION)
VALUES (3, 3, 3, TO_DATE('2024-03-15', 'YYYY-MM-DD'), NULL);
INSERT INTO Prestamos (ID_PRESTAMO, ID_LIBRO, ID_USUARIO, FECHA_PRESTAMO, FECHA_DEVOLUCION)
VALUES (4, 4, 4, TO_DATE('2024-04-01', 'YYYY-MM-DD'), NULL);
INSERT INTO Prestamos (ID_PRESTAMO, ID_LIBRO, ID_USUARIO, FECHA_PRESTAMO, FECHA_DEVOLUCION)
VALUES (5, 5, 5, TO_DATE('2024-05-10', 'YYYY-MM-DD'), TO_DATE('2024-05-25', 'YYYY-MM-DD'));
//--------------------------------------
```

8. Realiza una consulta para obtener todos los registros de la tabla 'Libros'.

```
//--------------------------------------
SELECT * FROM Libros;
//--------------------------------------
```

9. Realiza una consulta para obtener todos los registros de la tabla 'Usuarios'.

//---------------------------------------
```
SELECT * FROM Clientes;
```
//---------------------------------------
10. Mostrar solo los títulos y precios de los libros.
//---------------------------------------
```
SELECT Titulo, Precio FROM Libros;
```
//---------------------------------------
11. Ordenar los registros de la tabla Clientes por Nombre en orden alfabético.
//---------------------------------------
```
SELECT * FROM Clientes
ORDER BY Nombre ASC;
```
//---------------------------------------
12. Seleccionar los libros cuyo precio sea mayor a 15.
//---------------------------------------
```
SELECT * FROM Libros
WHERE Precio > 15;
```
//---------------------------------------
13. Contar cuántos préstamos hay en la tabla Prestamos.
//---------------------------------------
```
SELECT COUNT(*) FROM Prestamos;
```
//---------------------------------------
14. Mostrar los préstamos realizados después del 1 de marzo de 2024.
//---------------------------------------
```
SELECT * FROM Prestamos
WHERE FECHA_PRESTAMO > TO_DATE('2024-03-01', 'YYYY-MM-DD');
```
//---------------------------------------
15. Agregar una nueva columna llamada Email a la tabla Clientes.
//---------------------------------------
```
ALTER TABLE Clientes
ADD Email VARCHAR2(100);
```
//---------------------------------------
16. Insertar un nuevo registro en la tabla Clientes con un valor nulo en la columna Telefono.
//---------------------------------------
```
INSERT INTO Clientes (ClienteID, Nombre, Direccion, Telefono, FechaRegistro)
VALUES (6, 'Sandra Díaz', 'Calle Nueva 303', NULL, SYSDATE);
```
//---------------------------------------
17. Modificar el valor de Telefono para el cliente con ClienteID = 1 a 650000000.
//---------------------------------------
```
UPDATE Clientes
SET Telefono = '650000000'
WHERE ClienteID = 1;
```
//---------------------------------------
18. Borrar el préstamo con PrestamoID = 5.
//---------------------------------------
```
DELETE FROM Prestamos
WHERE ID_PRESTAMO = 5;
```
//---------------------------------------
19. Crear una secuencia llamada seq_Libros para generar valores únicos en la columna LibroID.
//---------------------------------------
```
CREATE SEQUENCE seq_Libros
```

```
START WITH 1
INCREMENT BY 1
NOCACHE
NOCYCLE;
//-------------------------------------
```
20. Crear una tabla Autores y establecer una relación con la tabla Libros.
```
//-------------------------------------
CREATE TABLE Autores (
   AutorID NUMBER PRIMARY KEY NOT NULL,
   Nombre VARCHAR2(100) NOT NULL,
   FechaNacimiento DATE
);

ALTER TABLE Libros
ADD AutorID NUMBER;

ALTER TABLE Libros
ADD CONSTRAINT FK_Autor
FOREIGN KEY (AutorID) REFERENCES Autores(AutorID);

INSERT INTO Autores (AutorID, Nombre, FechaNacimiento)
VALUES (1, 'Miguel de Cervantes', TO_DATE('1547-09-29', 'YYYY-MM-DD'));

INSERT INTO Autores (AutorID, Nombre, FechaNacimiento)
VALUES (2, 'Gabriel García Márquez', TO_DATE('1927-03-06', 'YYYY-MM-DD'));

UPDATE Libros
SET AutorID = 1
WHERE LibroID = 1;
UPDATE Libros
SET AutorID = 2
WHERE LibroID = 2;
//-------------------------------------
```
21. Realizar una combinación INNER JOIN entre Libros y Prestamos para mostrar los títulos de los libros prestados.
```
//-------------------------------------
SELECT L.Titulo
FROM Libros L
INNER JOIN Prestamos P ON L.LibroID = P.ID_LIBRO;
//-------------------------------------
```
22. Utilizar una subconsulta para encontrar los títulos de libros no prestados.
```
//-------------------------------------
SELECT Titulo
FROM Libros
WHERE LibroID NOT IN (
   SELECT ID_LIBRO
   FROM Prestamos
);
```

//------------------------------------

23. Mostrar el cliente que más libros ha prestado.

//------------------------------------

```
SELECT C.Nombre, COUNT(P.ID_PRESTAMO) AS NumeroDePrestamos
FROM Clientes C
JOIN Prestamos P ON C.ClienteID = P.ID_USUARIO
GROUP BY C.Nombre
ORDER BY NumeroDePrestamos DESC
FETCH FIRST 1 ROWS ONLY;
```

//------------------------------------

24. Crear una vista llamada LibrosPrestados que combine la información de las tablas Libros y Prestamos.

//------------------------------------

```
CREATE VIEW LibrosPrestados AS
SELECT L.LibroID, L.Titulo, L.Autor, L.Genero, L.FechaPublicacion, L.Precio, P.ID_PRESTAMO, P.ID_USUARIO, P.FECHA_PRESTAMO, P.FECHA_DEVOLUCION
FROM Libros L
INNER JOIN Prestamos P ON L.LibroID = P.ID_LIBRO;
```

//------------------------------------

25. Realizar una combinación LEFT JOIN entre las tablas Clientes y Prestamos para mostrar todos los clientes, incluyendo aquellos que no han realizado préstamos.

//------------------------------------

```
SELECT C.ClienteID, C.Nombre, C.Direccion, C.Telefono, C.FechaRegistro, P.ID_PRESTAMO, P.ID_LIBRO, P.FECHA_PRESTAMO, P.FECHA_DEVOLUCION

FROM Clientes C

LEFT JOIN Prestamos P ON C.ClienteID = P.ID_USUARIO;
```

//------------------------------------

26. Realizar una combinación RIGHT JOIN entre las tablas Prestamos y Libros para mostrar todos los libros, incluyendo aquellos que no han sido prestados.

//------------------------------------

```
SELECT L.LibroID, L.Titulo, L.Autor, L.Genero, L.FechaPublicacion, L.Precio, P.ID_PRESTAMO, P.ID_USUARIO, P.FECHA_PRESTAMO, P.FECHA_DEVOLUCION

FROM Libros L

RIGHT JOIN Prestamos P ON L.LibroID = P.ID_LIBRO;
```

//------------------------------------

27. Crear una consulta que utilice la combinación NATURAL JOIN para relacionar Prestamos con Libros.

//----------

```
SELECT *

FROM Prestamos

NATURAL JOIN Libros;
```

//----------

28. Crear una consulta que utilice la cláusula USING para combinar las tablas Prestamos y Libros por el campo LibroID.

//----------

```
SELECT *

FROM Prestamos

JOIN Libros

USING (LibroID);
```

//----------

29. Crear una subconsulta correlativa que encuentre los libros cuyo precio sea mayor que el promedio de los libros prestados.

//----------

```
SELECT L.LibroID, L.Titulo, L.Precio

FROM Libros L

WHERE L.Precio > (

   SELECT AVG(L2.Precio)

   FROM Libros L2

   JOIN Prestamos P ON L2.LibroID = P.ID_LIBRO

   WHERE L2.LibroID = L.LibroID

);
```

//----------

30. Agregar un índice único a la columna Titulo en la tabla Libros.

//----------

```
CREATE UNIQUE INDEX idx_unique_titulo

ON Libros (Titulo);

//-------------------------------------
```

31. Actualizar el precio de los libros del género "Fantasía" aumentando su valor en un 10%.

```
//-------------------------------------

UPDATE Libros

SET Precio = Precio * 1.10

WHERE Genero = 'Fantasía';

//-------------------------------------
```

32. Crear una restricción CHECK en la tabla Libros que asegure que el precio no sea negativo.

```
//-------------------------------------

ALTER TABLE Libros

ADD CONSTRAINT chk_precio_no_negativo CHECK (Precio >= 0);

//-------------------------------------
```

33. Crear una tabla Multas con las siguientes columnas:

MultaID (NUMBER, PK)

PrestamoID (NUMBER, FK, referencia a Prestamos.PrestamoID)

Monto (NUMBER, NOT NULL)

FechaMulta (DATE, DEFAULT SYSDATE)

```
//-------------------------------------

CREATE TABLE Multas (

    MultaID NUMBER PRIMARY KEY,

    PrestamoID NUMBER,

    Monto NUMBER NOT NULL,

    FechaMulta DATE DEFAULT SYSDATE,

    CONSTRAINT fk_prestamo
```

```
    FOREIGN KEY (PrestamoID)
    REFERENCES Prestamos (ID_PRESTAMO)
)
```

//----------

34. Insertar registros en la tabla Multas, calculando automáticamente el monto como $5 por cada día de retraso (diferencia entre SYSDATE y FechaDevolucion).

//----------

```
CREATE SEQUENCE seq_Multas
START WITH 1
INCREMENT BY 1;

INSERT INTO Multas (MultaID, PrestamoID, Monto, FechaMulta)
SELECT seq_Multas.NEXTVAL,  -- Genera un valor único para MultaID
    P.ID_PRESTAMO,
    (SYSDATE - P.FECHA_DEVOLUCION) * 5 AS Monto,
    SYSDATE
FROM Prestamos P
WHERE P.FECHA_DEVOLUCION < SYSDATE;
```

//----------

35. Crear una vista llamada ClientesConMultas que muestre la información de los clientes y sus respectivas multas.

//----------

```
CREATE VIEW ClientesConMultas AS
SELECT C.ClienteID, C.Nombre, C.Direccion, C.Telefono, C.FechaRegistro,
    M.MultaID, M.Monto, M.FechaMulta
FROM Clientes C
LEFT JOIN Prestamos P ON C.ClienteID = P.ID_USUARIO  -- Cambia CLIENTEID por el nombre correcto
```

```
LEFT JOIN Multas M ON P.ID_PRESTAMO = M.PrestamoID;
```

//----------

36. Eliminar la vista ClientesConMultas.

//----------

```
DROP VIEW ClientesConMultas;
```

//----------

37. Crear una secuencia llamada seq_Multas para generar valores automáticos en la columna MultaID.

//----------

```
CREATE SEQUENCE seq_Multas

START WITH 1

INCREMENT BY 1

NOCACHE

NOCYCLE;
```

//----------

38. Crear una tabla Categorias con las siguientes columnas:

CategoriaID (NUMBER, PK)

NombreCategoria (VARCHAR2(50), NOT NULL)

//----------

```
CREATE TABLE Categorias (

   CategoriaID NUMBER PRIMARY KEY,

   NombreCategoria VARCHAR2(50) NOT NULL

);
```

//----------

39. Inserta los siguientes registros en la tabla categorías:

CategoriaID	NombreCategoria
1	Literatura
2	Ciencia
3	Historia
4	Tecnología

//----------

INSERT INTO Categorias (CategoriaID, NombreCategoria)

VALUES (1, 'Literatura');

INSERT INTO Categorias (CategoriaID, NombreCategoria)

VALUES (2, 'Ciencia');

INSERT INTO Categorias (CategoriaID, NombreCategoria)

VALUES (3, 'Historia');

INSERT INTO Categorias (CategoriaID, NombreCategoria)

VALUES (4, 'Tecnologia');

//----------

40. Agregar una columna CategoriaID en la tabla Libros como clave foránea, referenciando Categorias.CategoriaID.

//----------

ALTER TABLE Libros

ADD CategoriaID NUMBER;

ALTER TABLE Libros

ADD CONSTRAINT fk_categoria

FOREIGN KEY (CategoriaID)

REFERENCES Categorias (CategoriaID);

//----------

41. Actualizar los registros de la tabla Libros para asignarles categorías basadas en su género.

//-------------------------------------

```
UPDATE Libros

SET CategoriaID =

   CASE

        WHEN Genero = 'Fantasía' THEN 1   -- Asignar 'Literatura' (CategoriaID 1) para el género 'Fantasía'

        WHEN Genero = 'Ciencia' THEN 2     -- Asignar 'Ciencia' (CategoriaID 2) para el género 'Ciencia'

         WHEN Genero = 'Historia' THEN 3    -- Asignar 'Historia' (CategoriaID 3) para el género 'Historia'

          WHEN Genero = 'Tecnología' THEN 4 -- Asignar 'Tecnología' (CategoriaID 4) para el género 'Tecnología'

       ELSE NULL                      -- Si el género no coincide, asigna NULL

   END;
```

//-------------------------------------

42. Realizar una combinación FULL OUTER JOIN entre las tablas Clientes y Prestamos.

//-------------------------------------

```
SELECT C.ClienteID, C.Nombre, C.Direccion, C.Telefono, C.FechaRegistro,

                    P.ID_PRESTAMO, P.ID_USUARIO, P.ID_LIBRO, P.FECHA_PRESTAMO, P.FECHA_DEVOLUCION

FROM Clientes C

FULL OUTER JOIN Prestamos P

ON C.ClienteID = P.ID_USUARIO;
```

//-------------------------------------

43. Crear una consulta que utilice la cláusula UNION para combinar los nombres de los autores y los nombres de los clientes en una sola lista.

//------------------------------------

```
SELECT Autor AS Nombre

FROM Libros

UNION

SELECT Nombre

FROM Clientes;
```

//------------------------------------

44. Crear una consulta que utilice la cláusula INTERSECT para encontrar libros cuyo título y autor coincidan con los de otros registros en otra base de datos hipotética.

//------------------------------------

```
SELECT Titulo, Autor

FROM Libros

INTERSECT

SELECT Titulo, Autor

FROM OtraBaseDeDatos.Libros;
```

//------------------------------------

45. Crear una consulta que utilice la cláusula MINUS para encontrar clientes que no tienen préstamos activos.

//------------------------------------

```
SELECT ClienteID, Nombre

FROM Clientes

MINUS

SELECT C.ClienteID, C.Nombre

FROM Clientes C

JOIN Prestamos P ON C.ClienteID = P.ID_USUARIO

WHERE P.FECHA_DEVOLUCION IS NULL;
```

//------------------------------------

46. Crear una subconsulta en un INSERT para agregar libros automáticamente a la tabla Libros basándose en los títulos y autores proporcionados en otra tabla temporal.

```
//-------------------------------------

//-- Crear una tabla temporal como ejemplo

CREATE TABLE TempLibros (

   Titulo VARCHAR2(100),

   Autor VARCHAR2(100)

);



//-- Insertar algunos datos en la tabla temporal (solo para ejemplo)

INSERT INTO TempLibros (Titulo, Autor)

VALUES ('El Hobbit', 'J.R.R. Tolkien');

INSERT INTO TempLibros (Titulo, Autor)

VALUES ('1984', 'George Orwell');

INSERT INTO TempLibros (Titulo, Autor)

VALUES ('Cien años de soledad', 'Gabriel García Márquez');

//-------------------------------------
```

47. Crear una subconsulta en un DELETE para eliminar préstamos cuya fecha de devolución sea anterior al 1 de enero de 2024.

```
//-------------------------------------

DELETE FROM Prestamos

WHERE ID_PRESTAMO IN (

   SELECT ID_PRESTAMO

   FROM Prestamos

   WHERE FECHA_DEVOLUCION < TO_DATE('2024-01-01', 'YYYY-MM-DD')

);

//-------------------------------------
```

48. Crear una tabla Reseñas con las siguientes columnas:

ReseñaID (NUMBER, PK)

LibroID (NUMBER, FK, referencia a Libros.LibroID)

ClienteID (NUMBER, FK, referencia a Clientes.ClienteID)

Comentario (VARCHAR2(500))

FechaReseña (DATE, DEFAULT SYSDATE)

//----------

```
CREATE TABLE Reseñas (
    ReseñaID NUMBER PRIMARY KEY,
    LibroID NUMBER,
    ClienteID NUMBER,
    Comentario VARCHAR2(500),
    FechaReseña DATE DEFAULT SYSDATE,
    CONSTRAINT fk_libro FOREIGN KEY (LibroID) REFERENCES Libros(LibroID),
    CONSTRAINT fk_cliente FOREIGN KEY (ClienteID) REFERENCES Clientes(ClienteID)
);
```

//----------

49. Insertar registros en la tabla Reseñas donde cada cliente haga una reseña de un libro.

//----------

```
CREATE SEQUENCE seq_reseñas
START WITH 1
INCREMENT BY 1;
```

```
INSERT INTO Reseñas (ReseñaID, LibroID, ClienteID, Comentario, FechaReseña)
VALUES (seq_reseñas.NEXTVAL, 1, 1, 'Un libro fascinante que explora mundos mágicos.', SYSDATE);
```

```
INSERT INTO Reseñas (ReseñaID, LibroID, ClienteID, Comentario, FechaReseña)

VALUES (seq_reseñas.NEXTVAL, 2, 2, 'Una distopía increíblemente relevante para los tiempos modernos.', SYSDATE);


INSERT INTO Reseñas (ReseñaID, LibroID, ClienteID, Comentario, FechaReseña)

VALUES (seq_reseñas.NEXTVAL, 3, 3, 'Una obra maestra que captura la complejidad de la historia.', SYSDATE);


INSERT INTO Reseñas (ReseñaID, LibroID, ClienteID, Comentario, FechaReseña)

VALUES (seq_reseñas.NEXTVAL, 4, 4, 'Tecnología avanzada con un impacto significativo en la sociedad.', SYSDATE);


INSERT INTO Reseñas (ReseñaID, LibroID, ClienteID, Comentario, FechaReseña)

VALUES (seq_reseñas.NEXTVAL, 5, 5, 'Una narrativa profunda que cuestiona el poder y la corrupción.', SYSDATE);

//-------------------------------------
```

50. Crear una consulta que utilice una subconsulta correlativa para encontrar los clientes

```
//-------------------------------------

SELECT C.ClienteID, C.Nombre

FROM Clientes C

WHERE (SELECT COUNT(*)

    FROM Reseñas R

    WHERE R.ClienteID = C.ClienteID) > 1;

//-------------------------------------
```