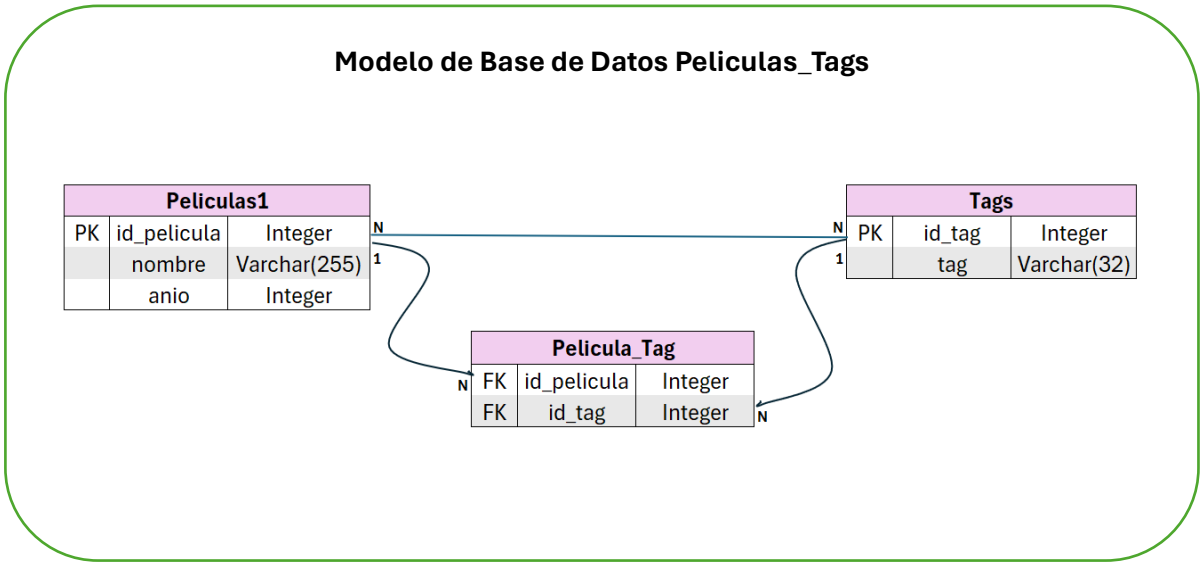
Prueba M5- Fundamentos de bases de datos relacionales

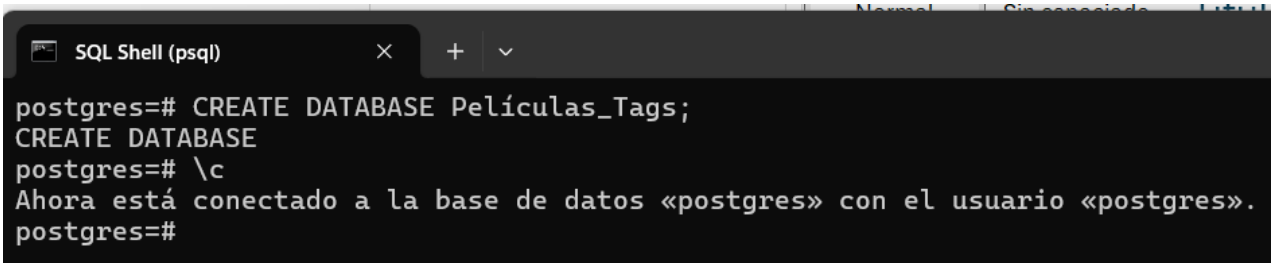
Parte I

1. Crea el modelo (revisa bien cuál es el tipo de relación antes de crearlo), respeta las claves primarias, foráneas y tipos de datos.



1a. Crearemos en SQL Shell la base de datos Peliculas_Tags

```
CREATE DATABASE Peliculas_Tags;
\c
```



1b. En SQL Shell se crean las tablas peliculas1– tags - pelicula_Tag

```
CREATE TABLE peliculas1(id_pelicula SERIAL PRIMARY KEY, nombre
varchar(255) NOT NULL, anio INT);

CREATE TABLE tags(id_tag SERIAL PRIMARY KEY, tag varchar(32) NOT NULL);

CREATE TABLE pelicula_tag(
id_pelicula INTEGER REFERENCES peliculas1(id_pelicula),
id_tag INTEGER REFERENCES Tags(id_tag),
PRIMARY KEY (id_pelicula, id_tag));
```

```
postgres=# CREATE TABLE peliculas1(id_pelicula SERIAL PRIMARY KEY, nombre varchar(255) NOT NULL, anio INT);
CREATE TABLE
postgres=# CREATE TABLE tags(id_tag SERIAL PRIMARY KEY, tag varchar(32) NOT NULL);
CREATE TABLE
postgres=# CREATE TABLE pelicula_tag(
postgres(# id_pelicula INTEGER REFERENCES peliculas1(id_pelicula),
postgres(# id_tag INTEGER REFERENCES Tags(id_tag),
postgres(#      PRIMARY KEY (id_pelicula, id_tag));
CREATE TABLE
postgres=#
```

2. Inserta 5 películas y 5 tags, la primera película tiene que tener 3 tags asociados, la segunda película debe tener dos tags asociados.

```
-- Insertamos 5 películas

INSERT INTO peliculas1 (nombre, anio) VALUES
('El Viaje Fantástico', 2021),
('La Odisea del Espacio', 2022),
('Los Jardines de la Luna', 2023),
('El Despertar del Leviatán', 2024),
('Las Puertas de Piedra', 2025);
```

```
postgres=# SELECT * FROM Peliculas1;
 id_pelicula |          nombre           | anio
-------------+---------------------------+------
           1 | El Viaje Fantástico       | 2021
           2 | La Odisea del Espacio     | 2022
           3 | Los Jardines de la Luna   | 2023
           4 | El Despertar del Leviatán | 2024
           5 | Las Puertas de Piedra     | 2025
(5 filas)
```

```
-- Insertamos 5 tags
```

```
INSERT INTO tags (tag) VALUES
('Ciencia Ficción'),
('Aventura'),
('Fantasía'),
('Espacio'),
('Misterio');
```

```
postgres=# SELECT * FROM Tags;
 id_tag |       tag
--------+-----------------
      1 | Ciencia Ficción
      2 | Aventura
      3 | Fantasía
      4 | Espacio
      5 | Misterio
(5 filas)
```

```
-- Asociamos 3 tags a la primera película (ID 1)

(1, 1), -- 'El Viaje Fantástico - Ciencia Ficción'
(1, 2), -- 'El Viaje Fantástico - Aventura'
(1, 3); -- 'El Viaje Fantástico - Fantasía'

-- Asociamos 2 tags a la segunda película (ID 2)

INSERT INTO pelicula_tag (id_pelicula, id_tag) VALUES
(2, 4), -- 'La Odisea del Espacio - Espacio'
(2, 5); -- 'La Odisea del Espacio - Misterio'
```

```
postgres=# INSERT INTO pelicula_tag (id_pelicula, id_tag) VALUES
postgres-# (1, 1), -- 'Ciencia Ficción'
postgres-# (1, 2), -- 'Aventura'
postgres-# (1, 3); -- 'Fantasía'
INSERT 0 3
postgres=# INSERT INTO pelicula_tag (id_pelicula, id_tag) VALUES
postgres-# (2, 4), -- 'Espacio'
postgres-# (2, 5); -- 'Misterio'
INSERT 0 2
```

Evidencia de que la primera película tiene 3 tags asociados y la segunda película tiene dos tags asociados.

```
postgres=# SELECT * FROM Pelicula_Tag;
 id_pelicula | id_tag
-------------+--------
           1 |      1
           1 |      2
           1 |      3
           2 |      4
           2 |      5
(5 filas)
```

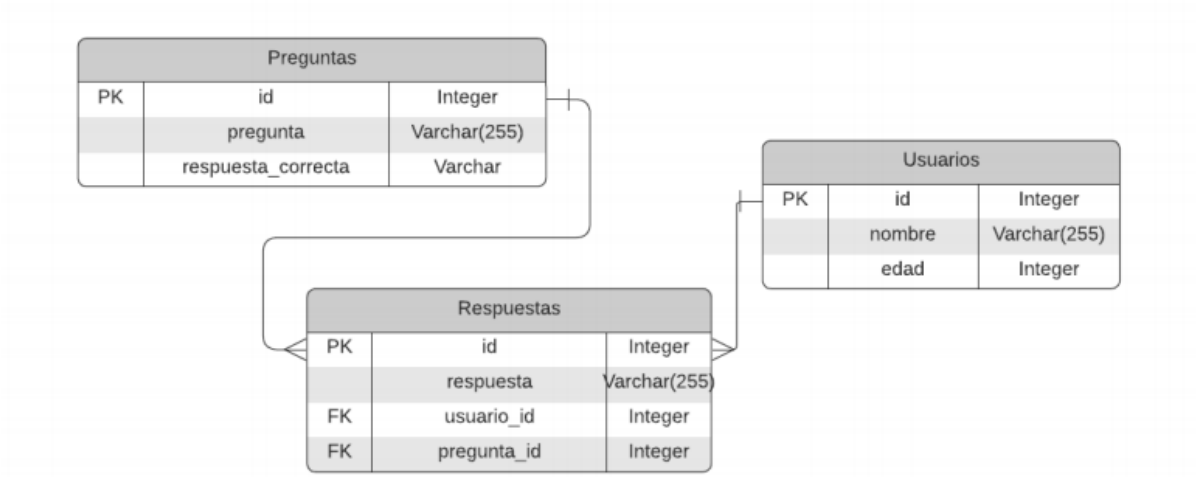
3. Contemos la cantidad de tags que tiene cada película. Si una película no tiene tags debe mostrar 0.

```
SELECT p.nombre, COUNT(pt.id_tag) AS cantidad_tags
FROM peliculas1 p
LEFT JOIN pelicula_tag pt ON p.id_pelicula = pt.id_pelicula
GROUP BY p.id_pelicula
ORDER BY p.nombre;
```

```
postgres=# SELECT p.nombre, COUNT(pt.id_tag) AS cantidad_tags
postgres-# FROM peliculas1 p
postgres-# LEFT JOIN pelicula_tag pt ON p.id_pelicula = pt.id_pelicula
postgres-# GROUP BY p.id_pelicula
postgres-# ORDER BY p.nombre;
          nombre           | cantidad_tags
---------------------------+---------------
 El Despertar del Leviatán |             0
 El Viaje Fantástico       |             3
 La Odisea del Espacio     |             2
 Las Puertas de Piedra     |             0
 Los Jardines de la Luna   |             0
(5 filas)
```

Parte II

Dado el siguiente modelo:



4. Crea las tablas respetando los nombres, tipos, claves primarias y foráneas y tipos de datos.

```
CREATE TABLE Preguntas (
    id INT PRIMARY KEY,
    pregunta VARCHAR(255),
    respuesta_correcta VARCHAR(255)
);

CREATE TABLE Usuarios (
    id INT PRIMARY KEY,
    nombre VARCHAR(255),
    edad INT
);

CREATE TABLE Respuestas (
    id INT PRIMARY KEY,
    respuesta VARCHAR(255),
    nombre_id INT,
    pregunta_id INT,
    FOREIGN KEY (nombre_id) REFERENCES Usuarios(id),
    FOREIGN KEY (pregunta_id) REFERENCES Preguntas(id)
);
```

Evidencia

```
postgres=# CREATE TABLE Preguntas (
postgres(#     id INT PRIMARY KEY,
postgres(#     pregunta VARCHAR(255),
postgres(#     respuesta_correcta VARCHAR(255)
postgres(# );
CREATE TABLE
postgres=#
postgres=# CREATE TABLE Usuarios (
postgres(#     id INT PRIMARY KEY,
postgres(#     nombre VARCHAR(255),
postgres(#     edad INT
postgres(# );
CREATE TABLE
postgres=#
postgres=# CREATE TABLE Respuestas (
postgres(#     id INT PRIMARY KEY,
postgres(#     respuesta VARCHAR(255),
postgres(#     nombre_id INT,
postgres(#     pregunta_id INT,
postgres(#     FOREIGN KEY (nombre_id) REFERENCES Usuarios(id),
postgres(#     FOREIGN KEY (pregunta_id) REFERENCES Preguntas(id)
postgres(# );
CREATE TABLE
postgres=#
```

5. Agrega 5 registros a la tabla preguntas, de los cuales:
    a. 1. La primera pregunta debe ser contestada correctamente por dos usuarios distintos
    b. 2. La pregunta 2 debe ser contestada correctamente por un usuario.
    c. 3. Los otros dos registros deben ser respuestas incorrectas.

Desarrollo

-- Insertamos 5 preguntas en la tabla Preguntas con sus respuestas e ingresamos datos de usuarios

```
INSERT INTO Preguntas (id, pregunta, respuesta_correcta) VALUES
(1, '¿Cuál es la capital de Francia?', 'París'),
(2, '¿Cuántos continentes hay en el mundo?', '7'),
(3, '¿Cuál es el río más largo del mundo?', 'Amazonas'),
(4, '¿Qué año llegó el hombre a la luna?', '1969'),
(5, '¿Cuál es el elemento químico más abundante en la atmósfera terrestre?',
'Nitrógeno');
```

```
INSERT INTO Usuarios (id, nombre, edad) VALUES
(101, 'Roberto Carlos', '60'),
(102, 'Ricardo Arjona', '45'),
(103, 'Alejandro Sanz', '15'),
(104, 'Marcianeque', '20'),
(105, 'Luis Fonsi', '35');
```

```
postgres=# INSERT INTO Preguntas (id, pregunta, respuesta_correcta) VALUES
postgres-# (1, '¿Cuál es la capital de Francia?', 'París'),
postgres-# (2, '¿Cuántos continentes hay en el mundo?', '7'),
postgres-# (3, '¿Cuál es el río más largo del mundo?', 'Amazonas'),
postgres-# (4, '¿Qué año llegó el hombre a la luna?', '1969'),
postgres-# (5, '¿Cuál es el elemento químico más abundante en la atmósfera terrestre?', 'Nitrógeno');
INSERT 0 5
postgres=#
postgres=#
postgres=#
postgres=#
postgres=# INSERT INTO Usuarios (id, nombre, edad) VALUES
postgres-# (101, 'Roberto Carlos', '60'),
postgres-# (102, 'Ricardo Arjona', '45'),
postgres-# (103, 'Alejandro Sanz', '15'),
postgres-# (104, 'Marcianeque', '20'),
postgres-# (105, 'Luis Fonsi', '35');
INSERT 0 5
```

-- La pregunta 1 es contestada correctamente por 2 usuarios distintos

INSERT INTO Respuestas (id, respuesta, nombre_id, pregunta_id) VALUES
(1, 'París', 101, 1),
(2, 'París', 102, 1);

```
postgres=# INSERT INTO Respuestas (id, respuesta, nombre_id, pregunta_id) VALUES
postgres-# (1, 'París', 101, 1),
postgres-# (2, 'París', 102, 1);
INSERT 0 2
postgres=# SELECT * FROM Respuestas;
 id | respuesta | nombre_id | pregunta_id
----+-----------+-----------+-------------
  1 | París     |       101 |           1
  2 | París     |       102 |           1
(2 filas)
```

-- La pregunta 2 es contestada correctamente por 1 usuario

INSERT INTO Respuestas (id, respuesta, nombre_id, pregunta_id) VALUES
(3, '7', 103, 2);

```
postgres=# INSERT INTO Respuestas (id, respuesta, nombre_id, pregunta_id) VALUES
postgres-# (3, '7', 103, 2);
```

```
postgres=# SELECT * FROM Respuestas;
 id | respuesta | nombre_id | pregunta_id
----+-----------+-----------+-------------
  1 | París     |       101 |           1
  2 | París     |       102 |           1
  3 | 7         |       103 |           2
(3 filas)
```

-- Los otros dos registros son respuestas incorrectas
-- Respuesta incorrecta para la pregunta 5
-- Respuesta incorrecta para la pregunta 3

```
INSERT INTO Respuestas (id, respuesta, nombre_id, pregunta_id) VALUES
(4, 'Nilo', 104, 3),
(5, 'Hidrógeno', 105, 5);
```

```
postgres=# INSERT INTO Respuestas (id, respuesta, nombre_id, pregunta_id) VALUES
postgres-# (4, 'Nilo', 104, 3),
postgres-# (5, 'Hidrógeno', 105, 5);
INSERT 0 2
postgres=# SELECT * FROM Respuestas;
 id | respuesta | nombre_id | pregunta_id
----+-----------+-----------+-------------
  1 | París     |       101 |           1
  2 | París     |       102 |           1
  3 | 7         |       103 |           2
  4 | Nilo      |       104 |           3
  5 | Hidrógeno |       105 |           5
(5 filas)
```

6. Cuenta la cantidad de respuestas correctas totales por usuario (independiente de la pregunta).

```
SELECT
usuarios.nombre,
COUNT(respuestas.respuesta) FILTER (WHERE respuestas.respuesta =
preguntas.respuesta_correcta) AS respuestas_correctas
FROM Usuarios
LEFT JOIN
    respuestas ON respuestas.nombre_id = usuarios.id
LEFT JOIN
    preguntas ON respuestas.pregunta_id = preguntas.id
GROUP BY
    usuarios.nombre;
```

```
postgres-#     usuarios.nombre;
     nombre     | respuestas_correctas
----------------+----------------------
 Marcianeque    |                    0
 Roberto Carlos |                    1
 Alejandro Sanz |                    1
 Luis Fonsi     |                    0
 Ricardo Arjona |                    1
(5 filas)
```

--Otras consultas

```
SELECT nombre_id, SUM(CASE WHEN respuesta = 'París' THEN 1 ELSE 0 END) AS respuestas_correctas
FROM Respuestas
GROUP BY nombre_id;

SELECT nombre_id, SUM(CASE WHEN respuesta = '7' THEN 1 ELSE 0 END) AS respuestas_correctas
FROM Respuestas
GROUP BY nombre_id;
```

```
postgres=# SELECT nombre_id, SUM(CASE WHEN respuesta = 'París' THEN 1 ELSE 0 END) AS respuestas_correctas
postgres-# FROM Respuestas
postgres-# GROUP BY nombre_id;
 nombre_id | respuestas_correctas
-----------+----------------------
       101 |                    1
       103 |                    0
       104 |                    0
       105 |                    0
       102 |                    1
(5 filas)


postgres=#
postgres=# SELECT nombre_id, SUM(CASE WHEN respuesta = '7' THEN 1 ELSE 0 END) AS respuestas_correctas
postgres-# FROM Respuestas
postgres-# GROUP BY nombre_id;
 nombre_id | respuestas_correctas
-----------+----------------------
       101 |                    0
       103 |                    1
       104 |                    0
       105 |                    0
       102 |                    0
(5 filas)
```

7. Por cada pregunta, en la tabla preguntas, cuenta cuántos usuarios tuvieron la respuesta correcta.

```
SELECT
    preguntas.pregunta,
    COUNT(usuarios.id) FILTER (WHERE respuestas.respuesta = preguntas.respuesta_correcta) AS respuestas_correctas
FROM
    preguntas
LEFT JOIN
    respuestas ON respuestas.pregunta_id = preguntas.id
LEFT JOIN
    usuarios ON respuestas.nombre_id = usuarios.id
GROUP BY
    preguntas.pregunta;
```

```
postgres=# SELECT
postgres-#     preguntas.pregunta,
postgres-#     COUNT(usuarios.id) FILTER (WHERE respuestas.respuesta = preguntas.respuesta_correcta) AS respuestas_corre
ctas
postgres-# FROM
postgres-#     preguntas
postgres-# LEFT JOIN
postgres-#     respuestas ON respuestas.pregunta_id = preguntas.id
postgres-# LEFT JOIN
postgres-#     usuarios ON respuestas.nombre_id = usuarios.id
postgres-# GROUP BY
postgres-#     preguntas.pregunta;
                            pregunta                             | respuestas_correctas
-----------------------------------------------------------------+----------------------
 ¿Cuál es el río más largo del mundo?                            |                    0
 ¿Cuál es la capital de Francia?                                 |                    2
 ¿Qué año llegó el hombre a la luna?                             |                    0
 ¿Cuántos continentes hay en el mundo?                           |                    1
 ¿Cuál es el elemento químico más abundante en la atmósfera terrestre? |                    0
(5 filas)
```

```
--Otra forma de consulta

SELECT p.id AS pregunta_id,
       COUNT(DISTINCT r.nombre_id) AS usuarios_respuesta_correcta
FROM Preguntas p
LEFT JOIN Respuestas r ON p.id = r.pregunta_id
WHERE r.respuesta IN ('París', '7')
GROUP BY p.id;
```

```
postgres=# SELECT p.id AS pregunta_id,
postgres-#        COUNT(DISTINCT r.nombre_id) AS usuarios_respuesta_correcta
postgres-# FROM Preguntas p
postgres-# LEFT JOIN Respuestas r ON p.id = r.pregunta_id
postgres-# WHERE r.respuesta IN ('París', '7')
postgres-# GROUP BY p.id;
 pregunta_id | usuarios_respuesta_correcta
-------------+-----------------------------
           1 |                           2
           2 |                           1
(2 filas)
```

8. Implementa borrado en cascada de las respuestas al borrar un usuario y borrar el primer usuario para probar la implementación.

```
ALTER TABLE Respuestas
ADD CONSTRAINT fk_nombre
FOREIGN KEY (nombre_id)
REFERENCES usuarios(id)
ON DELETE CASCADE;

DELETE FROM respuestas WHERE nombre_id = 101;
DELETE FROM Usuarios WHERE id = 101;
```

```
postgres=# DELETE FROM Usuarios WHERE id = 101;
DELETE 1
postgres=# DELETE FROM respuestas WHERE nombre_id = 101;
DELETE 0
postgres=# SELECT * FROM Usuarios;
 id  |      nombre      | edad
-----+------------------+------
 102 | Ricardo Arjona   |   45
 103 | Alejandro Sanz   |   15
 104 | Marcianeque      |   20
 105 | Luis Fonsi       |   35
(4 filas)
```

9. Crea una restricción que impida insertar usuarios menores de 18 años en la base de datos.

```
ALTER TABLE Usuarios ADD CONSTRAINT edad CHECK (edad > 18);

UPDATE Usuarios
SET edad = '25'
WHERE id = 103;

ALTER TABLE Usuarios ADD CONSTRAINT edad CHECK (edad > 18);
```

```
postgres=# ALTER TABLE Usuarios ADD CONSTRAINT edad CHECK (edad > 18);
ERROR:  la restricción check «edad» de la relación «usuarios» es violada por alguna fila
postgres=# SELECT * FROM Usuarios;
 id  |      nombre      | edad
-----+------------------+------
 102 | Ricardo Arjona   |   45
 103 | Alejandro Sanz   |   15
 104 | Marcianeque      |   20
 105 | Luis Fonsi       |   35
(4 filas)


postgres=# UPDATE Usuarios
postgres-# SET edad = '25'
postgres-# WHERE id = 103;
UPDATE 1
postgres=#
postgres=# ALTER TABLE Usuarios ADD CONSTRAINT edad CHECK (edad > 18);
ALTER TABLE
postgres=# SELECT * FROM Usuarios;
 id  |      nombre      | edad
-----+------------------+------
 102 | Ricardo Arjona   |   45
 104 | Marcianeque      |   20
 105 | Luis Fonsi       |   35
 103 | Alejandro Sanz   |   25
(4 filas)
```

10. Altera la tabla existente de usuarios agregando el campo email con la restricción de único.

```
ALTER TABLE usuarios
ADD email VARCHAR(255) UNIQUE;
```

Este comando SQL altera la tabla usuarios para añadir una nueva columna llamada email.

```
postgres=# ALTER TABLE usuarios
postgres-# ADD email VARCHAR(255) UNIQUE;
ALTER TABLE
postgres=# SELECT * FROM Usuarios;
 id  |     nombre      | edad | email
-----+-----------------+------+-------
 102 | Ricardo Arjona  |   45 |
 104 | Marcianeque     |   20 |
 105 | Luis Fonsi      |   35 |
 103 | Alejandro Sanz  |   25 |
(4 filas)
```

Gracias!!!!