EVALUACIÓN

Base de Datos Aplicadas Semana 3

Nombre del estudiante: Jesus Antonio Navarrete Castillo

Fecha de entrega: 20/01/2025

Carrera: Analista de Datos

1. **Crea una base de datos con sus tablas y restricciones para garantizar la eficiencia en el análisis de datos. Crea una base de datos llamada "MusicStreamingDB" con sus tablas para almacenar datos de reproducción, usuarios y canciones, con las columnas adecuadas para cada tipo de información. Establece restricciones, como claves primarias y claves foráneas, para garantizar la integridad de los datos.**

R:

```
-- Crear la base de datos:
CREATE DATABASE MusicStreamingDB;

 -- Usar la base de datos
USE MusicStreamingDB;

-- Tabla Usuarios
CREATE TABLE Usuarios (
id_usuario INT AUTO_INCREMENT PRIMARY KEY,
nombre VARCHAR(100) NOT NULL,
email VARCHAR(150) UNIQUE NOT NULL,
fecha_registro DATE NOT NULL,
pais VARCHAR(50)
);

-- Tabla Canciones
CREATE TABLE Canciones (
id_cancion INT AUTO_INCREMENT PRIMARY KEY,
titulo VARCHAR(150) NOT NULL,
artista VARCHAR(100) NOT NULL,
album VARCHAR(100),
genero VARCHAR(50),
duracion TIME NOT NULL,
fecha_lanzamiento DATE
);
```

```
-- Tabla Reproducciones
CREATE TABLE Reproducciones (
id_reproduccion INT AUTO_INCREMENT PRIMARY KEY,
id_usuario INT NOT NULL,
id_cancion INT NOT NULL,
fecha_reproduccion DATETIME NOT NULL,
dispositivo VARCHAR(50),
FOREIGN KEY (id_usuario) REFERENCES Usuarios(id_usuario) ON DELETE CASCADE,
FOREIGN KEY (id_cancion) REFERENCES Canciones(id_cancion) ON DELETE CASCADE
);
```

Descripción del diseño

1. **Tabla de usuarios** :
   - Contiene información de los usuarios, como su nombre, correo electrónico (único), fecha de registro y país.
   - El campo id_usuarioes la clave primaria.
2. **Tabla de Canciones** :
   - Guarde información relevante sobre cada canción, como el título, artista, álbum, género, duración y fecha de lanzamiento.
   - El campo id_canciones la clave primaria.
3. **Tabla Reproducciones** :
   - Registre cada reproducción realizada por un usuario.
   - Contiene claves foráneas que enlazan con Usuariosy Cancionespara identificar quién reproduce qué canción.
   - Incluye información adicional como la fecha de reproducción y el dispositivo usado.

Restricciones implementadas

1. **Integridad referencial** :
   - Las claves foráneas garantizan la relación entre las tablas y eliminan automáticamente los datos relacionados cuando un usuario o una canción es eliminada.
2. **Unicidad** :
   - El correo electrónico en la tabla Usuarioses único para evitar duplicados.
3. **Integridad de datos** :
   - Las columnas importantes, como nombre, email, tituloy duracion, no permiten valores nulos para evitar registros incompletos.

2. Relaciona la modificación de datos con la inserción de estos, actualización de registros y eliminación de datos. Inserta registros de reproducción, usuarios y canciones en las tablas correspondientes utilizando la instrucción INSERT. Asegúrate de mantener actualizada la base de datos con las nuevas reproducciones y usuarios registrados. Actualiza registros de usuarios según sea necesario utilizando la instrucción UPDATE. Elimina registros obsoletos o no deseados de la base de datos utilizando la instrucción DELETE FROM.

R:

1. Inserción de registros

Insertemos datos en las tablas Usuarios , **CancCanciones , yReproducciones .

Insertar usuarios:

INSERT INTO Usuarios (nombre, email, fecha_registro, pais)

VALUES

('Juan Pérez', 'juan.perez@example.com', '2025-01-01', 'Chile'),

('Ana Gómez', 'ana.gomez@example.com', '2025-01-05', 'México'),

('Carlos López', 'carlos.lopez@example.com', '2025-01-10', 'España');

Insertar canciones:

INSERT INTO Canciones (titulo, artista, album, genero, duracion, fecha_lanzamiento)

VALUES

(

'Shape of You', 'Ed Sheeran', 'Divide', 'Pop', '00:03:53', '2017-01-06'),

(

(

'Blinding Lights', 'The Weeknd', 'After Hours', 'Synthwave', '00:03:20', '2019-11-29'),

('Bohemian Rhapsody', 'Queen', 'A Night at the Opera', 'Rock', '00:05:55', '1975-10-31');

```
INSERT INTO Reproducciones (id_usuario, id_cancion, fecha_reproduccion, dispositivo)
VALUES
    (1, 1, '2025-01-15 14:23:00', 'Móvil'),
    (2, 2, '2025-01-15 15:10:00', 'PC'),
    (3, 3, '2025-01-15 16:45:00', 'Tablet');
```

Actualizar el país de un usuario:

```
UPDATE Usuarios
SET pais = 'Argentina'
WHERE email = 'ana.gomez@example.com';
```

Actualizar el álbum de una canción:

```
UPDATE Canciones
SET album = 'The Best of Queen'
WHERE titulo = 'Bohemian Rhapsody';
```

Actualizar la fecha y dispositivo de una reproducción:

```
UPDATE Reproducciones
SET fecha_reproduccion = '2025-01-16 10:00:00', dispositivo = 'Smart TV'
WHERE id_reproduccion = 1;
```

Eliminar un usuario por correo electrónico:

```
DELETE FROM Usuarios WHERE email = 'carlos.lopez@example.com';
```

Eliminar una canción que no será más reproducida:

```
DELETE FROM Canciones
WHERE titulo = 'Shape of You';
```

Eliminar reproducciones antiguas (criterio de antigüedad)

```
DELETE FROM Reproducciones WHERE fecha_reproduccion < '2025-01-01';
```

3. Usa las sentencias del SQL considerando consultas SELECT, el filtrado de datos, resultados, agrupaciones, entidades y relaciones entre entidades. Realiza consultas SELECT para extraer información relevante, como las canciones más reproducidas, los usuarios más activos y el total de reproducciones por género. Utiliza el filtrado de datos con la cláusula WHERE para obtener resultados específicos, como las canciones de un artista particular o los usuarios que escucharon una canción específica. Agrupa datos utilizando la cláusula GROUP BY, por ejemplo, para obtener el total de reproducciones por género de música. Establece relaciones entre entidades utilizando claves foráneas, por ejemplo, relacionando las reproducciones con los usuarios y las canciones.

R:

1. Consultas básicas SELECT

Canciones más reproducidas:

```
SELECT
    C.titulo AS Cancion,
    C.artista AS Artista,
    COUNT(R.id_reproduccion) AS Total_Reproducciones
FROM
    Reproducciones R
JOIN
    Canciones C ON R.id_cancion = C.id_cancion
GROUP BY
    C.id_cancion
ORDER BY
    Total_Reproducciones DESC
LIMIT 5;
```

Usuarios más activo:

```
SELECT
    U.nombre AS Usuario,
    U.email AS Email,
    COUNT(R.id_reproduccion) AS Total_Reproducciones
FROM
    Reproducciones R
JOIN
    Usuarios U ON R.id_usuario = U.id_usuario
GROUP BY
    U.id_usuario
ORDER BY
    Total_Reproducciones DESC
LIMIT 5;
```

2. Filtrado de datos con DONDE

Canciones de un artista en particular:

```
SELECT
    titulo AS Cancion,
    album AS Album,
    fecha_lanzamiento AS Fecha_Lanzamiento
FROM
    Canciones
WHERE
    artista = 'Queen';
```

Usuarios que escuchan una canción específica:

```
SELECT
    U.nombre AS Usuario,
    U.email AS Email,
    R.fecha_reproduccion AS Fecha_Reproduccion,
    R.dispositivo AS Dispositivo
FROM
    Reproducciones R
JOIN
    Usuarios U ON R.id_usuario = U.id_usuario
JOIN
    Canciones C ON R.id_cancion = C.id_cancion
WHERE
    C.titulo = 'Bohemian Rhapsody';
```

3. Agrupación de datos con GROUP BY

Total de reproducciones por género:

```
SELECT
    C.genero AS Genero,
    COUNT(R.id_reproduccion) AS Total_Reproducciones
FROM
    Reproducciones R
JOIN
    Canciones C ON R.id_cancion = C.id_cancion
GROUP BY
    C.genero
ORDER BY
    Total_Reproducciones DESC;
```

Reproducciones por fecha:

```
SELECT
    DATE(R.fecha_reproduccion) AS Fecha,
    COUNT(R.id_reproduccion) AS Total_Reproducciones
FROM
    Reproducciones R
GROUP BY
    Fecha
ORDER BY
    Fecha DESC;
```

4. Relaciones entre entidades

Lista de canciones reproducidas por cada usuario:

```
SELECT
    U.nombre AS Usuario,
    C.titulo AS Cancion,
    C.artista AS Artista,
    R.fecha_reproduccion AS Fecha_Reproduccion
FROM
    Reproducciones R
JOIN
    Usuarios U ON R.id_usuario = U.id_usuario
JOIN
    Canciones C ON R.id_cancion = C.id_cancion
ORDER BY
    Usuario, Fecha_Reproduccion;
```

Géneros más escuchados por usuario:

```
SELECT
    U.nombre AS Usuario,
    C.genero AS Genero,
    COUNT(R.id_reproduccion) AS Total_Reproducciones
FROM
    Reproducciones R
JOIN
    Usuarios U ON R.id_usuario = U.id_usuario
JOIN
    Canciones C ON R.id_cancion = C.id_cancion
GROUP BY
    U.id_usuario, C.genero
ORDER BY
    Usuario, Total_Reproducciones DESC;
```

5. Consultas más avanzadas

Top 3 géneros más populares globalmente

```
SELECT
    C.genero AS Genero,
    COUNT(R.id_reproduccion) AS Total_Reproducciones
FROM
    Reproducciones R
JOIN
    Canciones C ON R.id_cancion = C.id_cancion
GROUP BY
    C.genero
ORDER BY
    Total_Reproducciones DESC
LIMIT 3;
```