Instituto Tecnológico de Mexicali

Ingeniería Sistemas Computacionales

Fundamentos de Base de Datos

Tema:

“U3_Tarea#1”

Estudiante:

Andrés Mojica Medina

No. De control: C21490782

Docente:

José Ramón Bogarín Valenzuela

Mexicali, B.C., 10 de Abril de 2025.

Query original

```
-- Crear tablas

CREATE TABLE estudiantes (

  id SERIAL PRIMARY KEY,

  nombre VARCHAR(100),

  email VARCHAR(100),

  fecha_nacimiento DATE

);



CREATE TABLE cursos (

  id SERIAL PRIMARY KEY,

  nombre_curso VARCHAR(100),

  duracion_meses INT

);



CREATE TABLE matriculas (

  id SERIAL PRIMARY KEY,

  id_estudiante INT REFERENCES estudiantes(id),

  id_curso INT REFERENCES cursos(id),

  fecha_matricula DATE
```

```
);


-- Insertar datos en estudiantes
INSERT INTO estudiantes (nombre, email, fecha_nacimiento) VALUES
('Ana Torres', 'ana@example.com', '1998-03-12'),
('Luis Gómez', 'luis@example.com', '2000-07-22'),
('Carla Ruiz', 'carla@example.com', '1995-11-05');


-- Insertar datos en cursos
INSERT INTO cursos (nombre_curso, duracion_meses) VALUES
('Bases de Datos', 4),
('Programación Web', 6);


-- Insertar datos en matriculas
INSERT INTO matriculas (id_estudiante, id_curso, fecha_matricula)
VALUES
(1, 1, '2025-01-10'),
(2, 1, '2025-01-12'),
(3, 2, '2025-02-05'),
(1, 2, '2025-02-10');
```

```sql
-- Consultas CLE


-- Estudiantes matriculados en "Bases de Datos"
SELECT e.nombre
FROM estudiantes e
JOIN matriculas m ON e.id = m.id_estudiante
JOIN cursos c ON c.id = m.id_curso
WHERE c.nombre_curso = 'Bases de Datos';


-- Cursos con cantidad de estudiantes matriculados
SELECT c.nombre_curso, COUNT(m.id_estudiante) AS total_estudiantes
FROM cursos c
LEFT JOIN matriculas m ON c.id = m.id_curso
GROUP BY c.nombre_curso;


-- Estudiantes mayores de 25 años
SELECT nombre, fecha_nacimiento,
       DATE_PART('year', AGE(fecha_nacimiento)) AS edad
FROM estudiantes
WHERE DATE_PART('year', AGE(fecha_nacimiento)) > 25;
```

```sql
-- Edad promedio de los estudiantes

SELECT ROUND(AVG(DATE_PART('year', AGE(fecha_nacimiento)))) AS
edad_promedio

FROM estudiantes;



-- Estudiantes ordenados por fecha de nacimiento

SELECT nombre, fecha_nacimiento

FROM estudiantes

ORDER BY fecha_nacimiento ASC;
```

Parte 1: Verificación y ajustes de estructura (LDD)

1. Agregar columna "teléfono" a la tabla estudiantes si no existe

`ALTER TABLE ADD COLUMN`: Agrega una columna a una tabla si no estaba previamente.

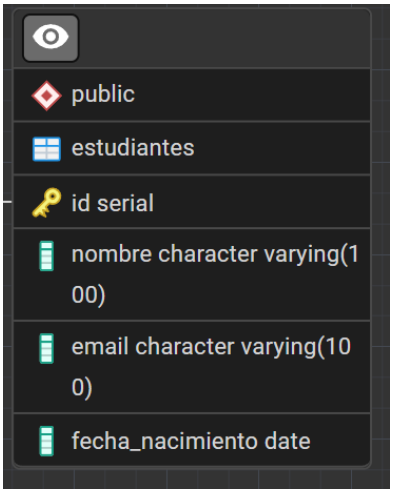
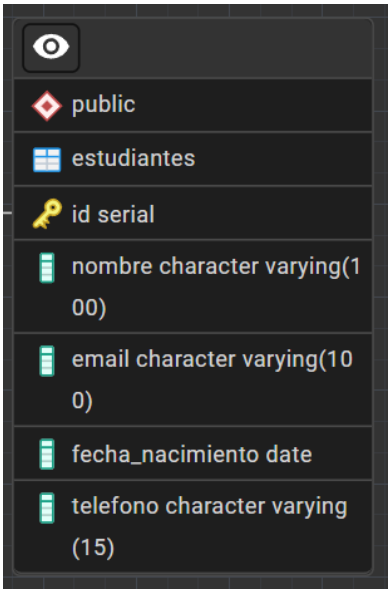


Tabla antes



2. Evitar nombres duplicados en la tabla cursos

`ADD CONSTRAINT UNIQUE`: Se asegura de que los valores de una columna sean únicos y no puedan repetirse.

```sql
-- 2. Evitar nombres duplicados en la tabla cursos
ALTER TABLE cursos
ADD CONSTRAINT nombre_curso_unico UNIQUE (nombre_curso);
```

Parte 2: Carga y ajuste de datos (LMD)

1. Actualizar el email de Luis Gómez

`UPDATE SET WHERE`: Permite actualizar datos específicos en una tabla, localizándolos con condiciones.

```
86  -- 1. Actualizar el email de Luis Gómez
87  UPDATE estudiantes
88  SET email = 'luisgomez@universidad.edu'
89  WHERE nombre = 'Luis Gómez';
90
```

Data Output | Messages | Notifications

SQL

	id [PK] integer	nombre character varying (100)	email character varying (100)	fecha_nacimiento date	telefono character varying (15)
1	2	Luis Gómez	luis@example.com	2000-07-22	6861112222

2. Registrar nueva matrícula de Carla Ruiz en "Bases de Datos"

`INSERT INTO SELECT FROM JOIN WHERE`: Se utiliza para insertar datos relacionados entre varias tablas a partir de una condición.

```
-- 2. Registrar nueva matrícula de Carla Ruiz en "Bases de Datos"
INSERT INTO matriculas (id_estudiante, id_curso, fecha_matricula)
SELECT e.id, c.id, '2025-04-01'
FROM estudiantes e
JOIN cursos c ON c.nombre_curso = 'Bases de Datos'
WHERE e.nombre = 'Carla Ruiz';
```

SQL

	id_matricula integer	estudiante character varying (100)	curso character varying (100)	fecha_matricula date
1	1	Ana Torres	Bases de Datos	2025-01-10
2	2	Luis Gómez	Bases de Datos	2025-01-12
3	3	Carla Ruiz	Programación Web	2025-02-05
4	4	Ana Torres	Programación Web	2025-02-10

Datos antes

Data Output | Messages | Notifications

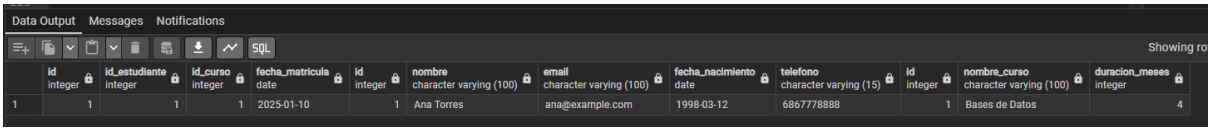
	id_matricula integer	estudiante character varying (100)	curso character varying (100)	fecha_matricula date
1	1	Ana Torres	Bases de Datos	2025-01-10
2	2	Luis Gómez	Bases de Datos	2025-01-12
3	3	Carla Ruiz	Programación Web	2025-02-05
4	4	Ana Torres	Programación Web	2025-02-10
5	5	Carla Ruiz	Bases de Datos	2025-04-01

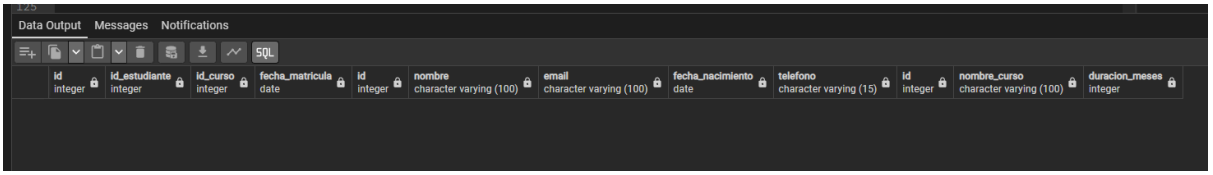
Datos despues

3. Eliminar la matrícula de Ana Torres en "Bases de Datos"

`DELETE FROM WHERE SELECT`: Elimina registros específicos combinando condiciones y subconsultas para obtener los IDs.

```sql
-- 3. Eliminar la matrícula de Ana Torres en "Bases de Datos"
DELETE FROM matriculas
WHERE id_estudiante = (
    SELECT id FROM estudiantes WHERE nombre = 'Ana Torres'
)
AND id_curso = (
    SELECT id FROM cursos WHERE nombre_curso = 'Bases de Datos'
);
```





Parte 3: Consultas avanzadas (CLE)

1. Listado con nombre del estudiante, nombre del curso y fecha de matrícula

`SELECT JOIN ON`: Permite combinar datos de varias tablas relacionadas para mostrar resultados conjuntos.

```sql
-- 1. Listado con nombre del estudiante, nombre del curso y fecha de matrícula
SELECT e.nombre AS estudiante, c.nombre_curso, m.fecha_matricula
FROM estudiantes e
JOIN matriculas m ON e.id = m.id_estudiante
JOIN cursos c ON c.id = m.id_curso;
```

Data Output | Messages | Notifications

	estudiante character varying (100)	nombre_curso character varying (100)	fecha_matricula date
1	Luis Gómez	Bases de Datos	2025-01-12
2	Carla Ruiz	Programación Web	2025-02-05
3	Ana Torres	Programación Web	2025-02-10
4	Carla Ruiz	Bases de Datos	2025-04-01

2. Cantidad de cursos tomados por cada estudiante

`COUNT GROUP BY`: Cuenta cuántas veces aparece cada estudiante en la tabla de matrículas.

```sql
-- 2. Cantidad de cursos tomados por cada estudiante
SELECT e.nombre, COUNT(m.id_curso) AS cursos_tomados
FROM estudiantes e
LEFT JOIN matriculas m ON e.id = m.id_estudiante
GROUP BY e.nombre;
```

Data Output | Messages | Notifications

	nombre character varying (100)	cursos_tomados bigint
1	Carla Ruiz	2
2	Luis Gómez	1
3	Ana Torres	1

3. Edad actual de cada estudiante ordenada de mayor a menor

`AGE, DATE_PART, ORDER BY`: Se calcula la edad y se ordenan los resultados desde el estudiante más grande al más joven.

```sql
-- 3. Edad actual de cada estudiante ordenada de mayor a menor
SELECT nombre,
       DATE_PART('year', AGE(fecha_nacimiento)) AS edad
FROM estudiantes
ORDER BY edad DESC;
```

Data Output | Messages | Notifications

	nombre character varying (100)	edad double precision
1	Carla Ruiz	29
2	Ana Torres	27
3	Luis Gómez	24

4. Curso con más estudiantes matriculados

`GROUP BY`, `ORDER BY DESC`, `LIMIT`: Agrupa por curso, cuenta estudiantes, y selecciona el que más tiene.

```sql
-- 4. Curso con más estudiantes matriculados
SELECT c.nombre_curso, COUNT(m.id_estudiante) AS total_estudiantes
FROM cursos c
JOIN matriculas m ON c.id = m.id_curso
GROUP BY c.nombre_curso
ORDER BY total_estudiantes DESC
LIMIT 1;
```

Data Output | Messages | Notifications

	nombre_curso character varying (100)	total_estudiantes bigint
1	Programación Web	2

5. Porcentaje de estudiantes matriculados respecto al total por curso

`ROUND`, `COUNT`, subconsulta `SELECT`: Calcula qué porcentaje de todos los estudiantes está matriculado en cada curso.

```sql
-- 5. Porcentaje de estudiantes matriculados respecto al total por curso
SELECT
  c.nombre_curso,
  COUNT(DISTINCT m.id_estudiante) AS estudiantes_matriculados,
  ROUND(
    (COUNT(DISTINCT m.id_estudiante) * 100.0) /
    (SELECT COUNT(*) FROM estudiantes), 2
  ) AS porcentaje_matriculados
FROM cursos c
LEFT JOIN matriculas m ON c.id = m.id_curso
GROUP BY c.nombre_curso;
```

Data Output | Messages | Notifications

	nombre_curso character varying (100)	estudiantes_matriculados bigint	porcentaje_matriculados numeric
1	Bases de Datos	2	66.67
2	Programación Web	2	66.67