

## INSTITUTO TECNOLÓGICO DE MEXICALI

Carrera:

Ing. en Sistemas Computacionales

Materia:

Fundamentos de base de datos.

Alumno:

Hernandez Garcia Martin 22490354

Actividad:

Tarea 1 Unidad 3

Profesor:

Jose Ramón Bogarin Valenzuela.

Mexicali, Baja California a 10 de Abril del 2025.



## QUERY Ejemplo

-- 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');
```

-- 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;
```

-- 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;
```

## Problema a resolver: "Analítica y Gestión Académica"

### Contexto

Una institución educativa quiere aprovechar su sistema de base de datos para obtener información útil sobre sus estudiantes, los cursos ofrecidos y las matrículas realizadas. Como analista de datos, se te solicita realizar una serie de tareas para mejorar la toma de decisiones académicas.

### Parte 1: Verificación y Ajustes de Estructura (LDD)

1. Verifica si la base de datos contiene una columna para almacenar el número de teléfono de los estudiantes. Si no existe, agrégala a la tabla estudiantes.
2. Modifica la tabla cursos para que el nombre del curso no pueda repetirse.

### Parte 2: Carga y Ajuste de Datos (LMD)

1. Actualiza el email de "Luis Gómez" a [luisgomez@universidad.edu](mailto:luisgomez@universidad.edu).



2. Registra una nueva matrícula para "Carla Ruiz" en el curso "Bases de Datos", con fecha 2025-04-01.
3. Elimina la matrícula de "Ana Torres" del curso "Bases de Datos".

### Parte 3: Consultas Avanzadas (CLE)

1. Muestra un listado con el nombre de cada estudiante, el nombre del curso al que está matriculado y la fecha de matrícula.
2. Muestra cuántos cursos ha tomado cada estudiante.
3. Calcula la edad actual de cada estudiante y ordénalos de mayor a menor edad.
4. Muestra qué curso tiene más estudiantes matriculados.
5. Calcula el porcentaje de estudiantes matriculados respecto al total de estudiantes para cada curso.

### Query Resueltos

#### Parte 1

1. Verifica si la base de datos contiene una columna para almacenar el número de teléfono de los estudiantes. Si no existe, agrégala a la tabla estudiantes.

```
SELECT * FROM estudiantes;  
ALTER TABLE estudiantes  
ADD COLUMN telefono VARCHAR(20);
```

	id [PK] integer	nombre character varying (100)	email character varying (100)	fecha_nacimiento date	telefono character varying (50)
1	1	Ana Torres	ana@example.com	1998-03-12	[null]
2	3	Carla Ruiz	carla@example.com	1995-11-05	[null]
3	2	Luis Gómez	luis@example.com	2000-07-22	[null]

2. Modifica la tabla cursos para que el nombre del curso no pueda repetirse.

```
ALTER TABLE cursos  
ADD UNIQUE (nombre_curso);
```

#### Parte 2

1. Actualiza el email de "Luis Gómez" a [luisgomez@universidad.edu](mailto:luisgomez@universidad.edu)



```
UPDATE estudiantes  
SET email = 'luisgomez@universidad.edu'  
WHERE nombre = 'Luis Gómez';
```

	id [PK] integer	nombre character varying (100)	email character varying (100)	fecha_nacimiento date	telefono character varying (50)
1	1	Ana Torres	ana@example.com	1998-03-12	[null]
2	3	Carla Ruiz	carla@example.com	1995-11-05	[null]
3	2	Luis Gómez	luisgomez@universidad.edu	2000-07-22	[null]

2. Registra una nueva matrícula para "Carla Ruiz" en el curso "Bases de Datos", con fecha 2025-04-01.

```
INSERT INTO matriculas (id_estudiante, id_curso, fecha_matricula)  
VALUES (  
(SELECT id FROM estudiantes WHERE nombre = 'Carla Ruiz'),  
(SELECT id FROM cursos WHERE nombre_curso = 'Bases de Datos'),  
'2025-04-01'  
);
```

	id [PK] integer	id_estudiante integer	id_curso integer	fecha_matricula date
1	1	1	1	2025-01-10
2	2	2	1	2025-01-12
3	3	3	2	2025-02-05
4	4	1	2	2025-02-10
5	5	3	1	2025-04-01

3. Elimina la matrícula de "Ana Torres" del curso "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');
```

	id [PK] integer	id_estudiante integer	id_curso integer	fecha_matricula date
1	2	2	1	2025-01-12
2	3	3	2	2025-02-05
3	4	1	2	2025-02-10
4	5	3	1	2025-04-01

**Parte 3**

1. Muestra un listado con el nombre de cada estudiante, el nombre del curso al que está matriculado y la fecha de matrícula.

```

SELECT e.nombre AS estudiante, c.nombre_curso AS curso, m.fecha_matricula
FROM matriculas m
INNER JOIN estudiantes e ON e.id = m.id_estudiante
INNER JOIN cursos c ON c.id = m.id_curso;
  
```

	estudiante character varying (100)	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. Muestra cuántos cursos ha tomado 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;
  
```



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

3. Calcula la edad actual de cada estudiante y ordénalos de mayor a menor edad.

```
SELECT nombre,  
DATE_PART('year', AGE(fecha_nacimiento)) AS edad  
FROM estudiantes  
ORDER BY edad DESC;
```

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



4. Muestra qué curso tiene más estudiantes matriculados.

```
SELECT c.nombre_curso, COUNT(m.id_estudiante) AS total_estudiantes  
FROM cursos c  
INNER JOIN matriculas m ON c.id = m.id_curso  
GROUP BY c.nombre_curso  
ORDER BY total_estudiantes DESC  
LIMIT 1;
```

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

5. Calcula el porcentaje de estudiantes matriculados respecto al total de estudiantes para cada curso.

```
SELECT c.nombre_curso,
ROUND(COUNT(DISTINCT m.id_estudiante) * 100.0 / (SELECT COUNT(*) FROM estudiantes), 2) AS porcentaje
FROM cursos c
LEFT JOIN matriculas m ON c.id = m.id_curso
GROUP BY c.nombre_curso;
```

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