



**Instituto Tecnológico de Mexicali**

**Alumna:**

**Jhanya Karina Castro Vidal**

**Num.Control:**

**23490030**

**Materia:**

**Fundamentos de Base de Datos**

**Maestro:**

**Jose Ramon Bogarin Valenzuela**

**“Examen Final U4”**

**Viernes 23 de marzo de 2025**

**11:59 pm**

## Problema Técnico: Gestionando la Información de una Universidad

Imagina que estás trabajando en el equipo de desarrollo de software para una universidad. La universidad necesita un sistema para gestionar la información de sus estudiantes, los cursos que ofrecen, las inscripciones de los estudiantes a los cursos, los profesores que imparten los cursos y los departamentos a los que pertenecen los profesores. Tu tarea es diseñar y trabajar con la base de datos que almacenará toda esta información.

### Objetivo General:

Diseñar una base de datos relacional y realizar diversas operaciones para gestionar la información de la universidad. Esto incluye la creación y modificación de la estructura de las tablas, la manipulación de los datos y la realización de consultas complejas para obtener información específica.

### Pasos a Seguir:

Los estudiantes deben seguir estos pasos para resolver el problema:

1. **Entendiendo las Entidades:** Identificar los elementos principales que necesita gestionar la universidad y la información relevante para cada uno.

- **Estudiantes:**

Información sobre los alumnos: id\_estudiante, nombre, apellido, fecha de nacimiento, dirección, ciudad y email.

- **Departamentos:**

Áreas o divisiones académicas de la universidad: id\_departamento, nombre del departamento y edificio.

- **Cursos:**

Materias que se imparten: id\_curso, id\_departamento (curso se relaciona con departamento), nombre del curso, descripción, créditos y semestre.

- **Inscripciones:**

Registro de estudiantes inscritos en cursos: id\_inscripcion, id\_estudiante (se relaciona con estudiante), id\_curso (se relaciona con curso), fecha de inscripción y calificación.

- **Profesores:**

Maestros de la universidad: id\_profesor, nombre, apellido, titulo, id\_departamento (profesor se relaciona con departamento)

- **Aulas:**

Espacio físico donde se imparten los cursos: id\_aula, nombre del aula, capacidad y la ubicación.

- **Horarios:**

Clases con fechas y horas: id\_horario, id\_curso ( horario se relaciona con curso), id\_aula (horario se relaciona con aula), fecha de inicio, fecha de fin, hora de inicio y hora fin.

- **Programas de Estudio:**

Planes de estudio: id\_programa, nombre del programa y descripción.

2. **Diseñando las Tablas (MDL):** Organizar la información en tablas, definiendo campos, tipos de datos y claves principales, y establecer las relaciones entre las tablas.

3. **Creando las Tablas (DDL):** Utilizar un sistema de gestión de bases de datos para crear las tablas definidas en el paso anterior.

Creamos cada tabla con sus claves primarias, claves foráneas, y sus atributos, y luego utilizamos el `select * from` para que nos muestre que la tabla fue realizada. Algunos atributos son únicos como lo es el correo en la tabla de estudiantes, esto nos indica que su valor es único que otro estudiante no puede tener ese mismo correo.

Haremos lo mismo para las siguientes tablas.

- **Creación tabla Estudiantes**

```
--Tabla de estudiantes
CREATE TABLE estudiantes (
  id_estudiante SERIAL PRIMARY KEY,
  nombre_e VARCHAR (50) NOT NULL,
  apellido_e VARCHAR (50) NOT NULL,
  fecha_nacimiento DATE NOT NULL,
  direccion VARCHAR (50),
  ciudad VARCHAR (50) NOT NULL,
  email VARCHAR (50) UNIQUE NOT NULL
);

select * from estudiantes;
```

|               |                        |                        |                  |                        |                        |                        |
|---------------|------------------------|------------------------|------------------|------------------------|------------------------|------------------------|
| id_estudiante | nombre_e               | apellido_e             | fecha_nacimiento | direccion              | ciudad                 | email                  |
| [PK] integer  | character varying (50) | character varying (50) | date             | character varying (50) | character varying (50) | character varying (50) |

- **Creación tabla Departamentos**

```
--Tabla de departamentos
CREATE TABLE departamentos(
  id_departamento SERIAL PRIMARY KEY,
  nombre_departamento VARCHAR(50) UNIQUE NOT NULL,
  edificio VARCHAR (50) NOT NULL
);

select * from departamentos;
```



```
select * from inscripciones;
```

| id_inscripcion | id_estudiante | id_curso | fecha_inscripcion | calificacion |
|----------------|---------------|----------|-------------------|--------------|
| [PK] integer   | integer       | integer  | date              | integer      |

- Creacion tabla Profesores

```
--Tabla profesores
CREATE TABLE profesores(
id_profesor SERIAL PRIMARY KEY,
nombre_p VARCHAR(50) NOT NULL,
apellido_p VARCHAR(50) NOT NULL,
titulo_profesor VARCHAR(50),
id_departamento INT NOT NULL,
FOREIGN KEY (id_departamento) REFERENCES departamentos(id_departamento)
);
```

```
Select * from profesores;
```

| id_profesor  | nombre_p               | apellido_p             | titulo_profesor        | id_departamento |
|--------------|------------------------|------------------------|------------------------|-----------------|
| [PK] integer | character varying (50) | character varying (50) | character varying (50) | integer         |

- Creacion tabla Aulas

```
CREATE TABLE aulas(
id_aula SERIAL PRIMARY KEY,
nombre_aula VARCHAR(50) NOT NULL UNIQUE,
capacidad int NOT NULL,
ubicacion VARCHAR(50) NOT NULL
);
```

```
select * from aulas;
```

| id_aula      | nombre_aula            | capacidad | ubicacion              |
|--------------|------------------------|-----------|------------------------|
| [PK] integer | character varying (50) | integer   | character varying (50) |

- Creación tabla horarios

```
CREATE TABLE horarios (  
id_horario SERIAL PRIMARY KEY,  
id_curso INT NOT NULL,  
id_aula INT NOT NULL,  
fecha_inicio DATE NOT NULL,  
fecha_fin DATE NOT NULL,  
hora_inicio time NOT NULL,  
hora_fin time NOT NULL  
);
```

```
select * from horarios;
```

|                            |                     |                    |                      |                   |                                       |                                    |
|----------------------------|---------------------|--------------------|----------------------|-------------------|---------------------------------------|------------------------------------|
| id_horario<br>[PK] integer | id_curso<br>integer | id_aula<br>integer | fecha_inicio<br>date | fecha_fin<br>date | hora_inicio<br>time without time zone | hora_fin<br>time without time zone |
|----------------------------|---------------------|--------------------|----------------------|-------------------|---------------------------------------|------------------------------------|

- Creación tabla intermedia CursosProfesores

```
CREATE TABLE cursos_profesores(  
id_curso_profesor SERIAL PRIMARY KEY,  
id_curso INT,  
id_profesor INT,  
FOREIGN KEY (id_curso) REFERENCES cursos(id_curso),  
FOREIGN KEY (id_profesor) REFERENCES profesores(id_profesor)  
);
```

```
select * from cursos_profesores;
```

|                                   |                     |                        |
|-----------------------------------|---------------------|------------------------|
| id_curso_profesor<br>[PK] integer | id_curso<br>integer | id_profesor<br>integer |
|-----------------------------------|---------------------|------------------------|

Cuando una tabla no contiene valores propios significa que es una tabla intermedia, la tabla intermedia nos indica que se hace la relación muchos a muchos.  
En esta tabla nos dice que muchos profesores pueden impartir muchos cursos.  
Se usa la misma lógica para cada tabla intermedia que se ha realizado.  
Tabla intermedia: Indica muchos a muchos.

- Creacion tabla ProgramasEstudio

```
CREATE TABLE programasestudio(
id_programa SERIAL PRIMARY KEY,
nombre_programa VARCHAR(50) UNIQUE NOT NULL,
descripcion_p TEXT
);
```

```
select * from programasestudio;
```

|    |                             |   |   |   |                       |    |    |   |     |
|----|-----------------------------|---|---|---|-----------------------|----|----|---|-----|
| ≡+ | 📄                           | ▼ | 📄   | ▼ | 🗑️                    | 🗄️ | ⬇️ | 📈 | SQL |
|    | id_programa<br>[PK] integer |   | nombre_programa<br>character varying (50) |   | descripcion_p<br>text |    |    |   |     |

- Creacion tabla intermedia ProgramasCursos

```
CREATE TABLE programas_cursos(
id_programa_curso SERIAL PRIMARY KEY,
id_programa INT,
id_curso INT,
FOREIGN KEY (id_programa) REFERENCES programasestudio(id_programa),
FOREIGN KEY (id_curso) REFERENCES cursos (id_curso)
);
```

```
select*from programas_cursos;
```

|   |                                   |   |                        |   |                     |   |   |   |   |
|---|-----------------------------------|---|------------------------|---|---------------------|---|---|---|---|
| 📄 | 📄                                 | 📄 | 📄                      | 📄 | 📄                   | 📄 | 📄 | 📄 | 📄 |
|   | id_programa_curso<br>[PK] integer |   | id_programa<br>integer |   | id_curso<br>integer |   |   |   |   |



4. **Modificando las Tablas (DDL):** Realizar modificaciones a la estructura de las tablas, como agregar, modificar o eliminar campos, según sea necesario.

- **Agregar Tablas:**

- Tabla: Campus
  - IDCampus (Clave Principal)
  - NombreCampus
  - DireccionCampus

```
CREATE TABLE campus (  
id_campus SERIAL PRIMARY KEY,  
nombre_campus VARCHAR(50) NOT NULL,  
direccion_campus VARCHAR(100) NOT NULL  
);
```

```
select * from campus;
```

| id_campus    | nombre_campus          | direccion_campus        |
|--------------|------------------------|-------------------------|
| [PK] integer | character varying (50) | character varying (100) |

- Tabla: Carreras
  - IDCarrera (Clave Principal)
  - NombreCarrera
  - TituloOtorgado

```
--Creacion tabla Carreras  
CREATE TABLE carreras (  
id_carrera SERIAL PRIMARY KEY,  
nombre_carrera VARCHAR(50) NOT NULL,  
titulo_otorgado VARCHAR(100) NOT NULL  
);
```

```
select* from carreras;
```

| id_carrera   | nombre_carrera         | titulo_otorgado         |
|--------------|------------------------|-------------------------|
| [PK] integer | character varying (50) | character varying (100) |

- Agregar una relación de muchos a muchos entre Estudiantes y Carreras

```
--Tabla intermedia estudiantes_carreras;
CREATE TABLE estudiantes_carreras (
id_estudiante INT NOT NULL,
id_carrera INT NOT NULL,
PRIMARY KEY (id_estudiante, id_carrera),
FOREIGN KEY (id_estudiante) REFERENCES estudiantes(id_estudiante),
FOREIGN KEY (id_carrera) REFERENCES carreras(id_carrera)
);|
```

```
select*from estudiantes_carreras;
```

| id_estudiante | id_carrera   |
|---------------|--------------|
| [PK] integer  | [PK] integer |

- **Modificar Tablas:**

1. En la tabla **Estudiantes**, agregar una clave foránea **IDCarrera** que haga referencia a la tabla **Carreras**.

```
ALTER TABLE estudiantes
ADD COLUMN id_carrera INT,
ADD CONSTRAINT fk_estudiantes_carrera
FOREIGN KEY (id_carrera) REFERENCES carreras(id_carre
```

```
Select* from estudiantes;
```

| id_estudiante                 | nombre_e                           | apellido_e                           | fecha_nacimiento         | direccion                            |
|-------------------------------|------------------------------------|--------------------------------------|--------------------------|--------------------------------------|
| id_estudiante<br>[PK] integer | nombre_e<br>character varying (50) | apellido_e<br>character varying (50) | fecha_nacimiento<br>date | direccion<br>character varying (100) |

|                        |                        |            |
|------------------------|------------------------|------------|
| ciudad                 | email                  | id_carrera |
| character varying (50) | character varying (50) | integer    |

2. En la tabla `Cursos`, agregar una columna `IDCampus` como clave foránea, referenciando la tabla `Campus`.

```
ALTER TABLE cursos
ADD COLUMN id_campus INT,
ADD CONSTRAINT fk_cursos_campus
FOREIGN KEY (id_campus) REFERENCES campus(id_campus);
```

```
SELECT * FROM cursos;
```

|                          |                            |  |                       |                     |
|--------------------------|----------------------------|--|-----------------------|---------------------|
| id_curso<br>[PK] integer | id_departamento<br>integer | nombre_curso<br>character varying (50) | descripcion_c<br>text | creditos<br>integer |
| creditos<br>integer      | semestre<br>integer        | id_campus<br>integer                   |                       |                     |

3. Modificar la tabla `Profesores` para incluir un campo `Email`

```
ALTER table profesores
ADD column email VARCHAR(50) UNIQUE NOT NULL;
SELECT * FROM profesores;
```

- **Eliminar Tablas/Campos**

- Eliminar la columna `Ciudad` de la tabla `Estudiantes`.

```
--Eliminar la columna Ciudad de la tabla estudiantes
ALTER TABLE estudiantes
DROP COLUMN ciudad;
```

```
select* from estudiantes;
```

|                          |                                     |                                 |                     |
|--------------------------|-------------------------------------|---------------------------------|---------------------|
| fecha_nacimiento<br>date | direccion<br>character varying (50) | email<br>character varying (50) | id_carre<br>integer |
| fecha_nacimiento<br>date | direccion<br>character varying (50) | email<br>character varying (50) | id_carre<br>integer |

- Eliminar la tabla Aulas

```
--Eliminar tabla Aulas
DROP TABLE aulas;
```

```
143 --Eliminar tabla Aulas
144 DROP TABLE aulas;
145 select *from aulas;
146
147
148
```

Data Output Messages Notifications

```
ERROR: no existe la relación «aulas»
LINE 1: select *from aulas;
                        ^
```

```
SQL state: 42P01
Character: 14
```

2. **Insertando Datos:** Insertar datos de ejemplo en las tablas para representar la información de la universidad.

Insertamos algunos datos de ejemplo, con los cambios que se realizaron en algunas tablas y no utilizamos horarios porque se pidió que se eliminara la tabla Aulas. Como la tabla horarios depende de Aulas esta quedara invalida.

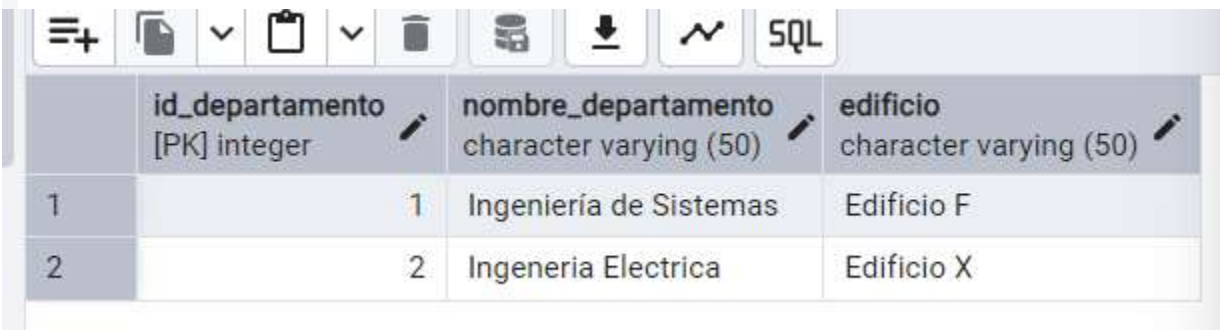
## Ejemplos:

```
-- Insertar campus
INSERT INTO campus (nombre_campus, direccion_campus)
VALUES
('TecNM Campus Mexicali', 'Av. Tecnológico s/n, Mexicali, BC'),
('TecNM Campus Tijuana', 'Calz. del Tecnológico 12950, Tijuana, BC');
-- TecNM Campus Tijuana
select* from campus;
```



|   | id_campus<br>[PK] integer | nombre_campus<br>character varying (50) | direccion_campus<br>character varying (100) |
|---|---------------------------|---|---|
| 1 | 1                         | TecNM Campus Mexicali                   | Av. Tecnológico s/n, Mexicali, BC           |
| 2 | 2                         | TecNM Campus Tijuana                    | Calz. del Tecnológico 12950, Tijuana, BC    |

```
-- INSERTAR DEPARTAMENTOS
INSERT INTO departamentos (nombre_departamento, edificio) VALUES
('Ingeniería de Sistemas', 'Edificio F'),
('Ingeniería Eléctrica', 'Edificio X');
select*from departamentos;
```



|   | id_departamento<br>[PK] integer | nombre_departamento<br>character varying (50) | edificio<br>character varying (50) |
|---|---------------------------------|---|------------------------------------|
| 1 | 1                               | Ingeniería de Sistemas                        | Edificio F                         |
| 2 | 2                               | Ingeniería Eléctrica                          | Edificio X                         |

```
-- INSERTAR CARRERAS
INSERT INTO carreras (nombre_carrera, titulo_otorgado) VALUES
('Ingeniería en Sistemas Computacionales', 'Ingeniero en Sistemas Computacionales'),
('Ingeniería Eléctrica', 'Ingeniero Eléctrico');
SELECT*FROM carreras;
```



|   | id_carrera<br>[PK] integer | nombre_carrera<br>character varying (50) | titulo_otorgado<br>character varying (100) |
|---|----------------------------|--|--|
| 1 | 1                          | Ingeniería en Sistemas Computacionales   | Ingeniero en Sistemas Computacionales      |
| 2 | 2                          | Ingeniería Eléctrica                     | Ingeniero Eléctrico                        |

```
-- INSERTAR PROGRAMAS DE ESTUDIO
INSERT INTO programasesudio (nombre_programa, descripcion_p) VALUES
('Programa de Desarrollo de Software', 'Formación para desarrolladores de hardware y software'),
('Ingeniería Eléctrica', 'Programa de formación en Ingeniería Eléctrica');
SELECT * FROM programasesudio;
```

|   | id_programa<br>[PK] integer | nombre_programa<br>character varying (50) | descripcion_p<br>text                                 |
|---|-----------------------------|---|---|
| 1 | 1                           | Programa de Desarrollo de Software        | Formación para desarrolladores de hardware y software |
| 2 | 2                           | Ingeniería Eléctrica                      | Programa de formación en Ingeniería Eléctrica         |

```
-- Inserción de estudiantes
INSERT INTO estudiantes (nombre_e, apellido_e, fecha_nacimiento, direccion, email, id_carrera) VALUES
('Carlos', 'Gómez', '2000-05-12', 'Calle Falsa 123', 'a2549085@itmexicali.edu.mx', 1),
('Ana', 'López', '1999-08-23', 'Av. Siempre Viva 742', 'a2249065@itmexicali.edu.mx', 2),
('José', 'Sánchez', '2002-07-18', 'Av. Principal 321', 'a9988776@itmexicali.edu.mx', 2);
SELECT * FROM estudiantes;
```

|   | id_estudiante<br>[PK] integer | nombre_e<br>character varying (50) | apellido_e<br>character varying (50) | fecha_nacimiento<br>date | direccion<br>character varying (50) | email<br>character varying (50) | id_carrera<br>integer |
|---|-------------------------------|------------------------------------|--------------------------------------|--------------------------|-------------------------------------|---------------------------------|-----------------------|
| 1 | 1                             | Carlos                             | Gómez                                | 2000-05-12               | Calle Falsa 123                     | a2549085@itmexicali.edu.mx      | 1                     |
| 2 | 2                             | Ana                                | López                                | 1999-08-23               | Av. Siempre Viva 742                | a2249065@itmexicali.edu.mx      | 2                     |
| 3 | 3                             | José                               | Sánchez                              | 2002-07-18               | Av. Principal 321                   | a9988776@itmexicali.edu.mx      | 2                     |

```
-- INSERCIÓN DE PROFESORES se incluyó el email
INSERT INTO profesores (nombre_p, apellido_p, titulo_profesor, id_departamento, email) VALUES
('Luis', 'Martínez', 'Maestría en Ingeniería del Software', 1, 'lmartinez@itmexicali.edu.mx'),
('Mario', 'Fernández', 'Doctorado en Ingeniería Eléctrica', 2, 'mfernandez@itmexicali.edu.mx');
SELECT * FROM profesores;
```

|   | id_profesor<br>[PK] integer | nombre_p<br>character varying (50) | apellido_p<br>character varying (50) | titulo_profesor<br>character varying (50) | id_departamento<br>integer | email<br>character varying (50) |
|---|-----------------------------|------------------------------------|--------------------------------------|---|----------------------------|---------------------------------|
| 1 | 1                           | Luis                               | Martínez                             | Maestría en Ingeniería del Software       | 1                          | lmartinez@itmexicali.edu.mx     |
| 2 | 2                           | Mario                              | Fernández                            | Doctorado en Ingeniería Eléctrica         | 2                          | mfernandez@itmexicali.edu.mx    |

```
-- CURSOS incluyendo id_campus
INSERT INTO cursos (id_departamento, nombre_curso, descripcion_c, credits, semestre, id_campus) VALUES
(1, 'Programación I', 'Introducción a la programación', 5, 1, 1),
(2, 'Circuitos Eléctricos I', 'Comportamiento de los Circuitos Eléctricos', 5, 4, 2);
SELECT * FROM cursos;
```

|   | id_curso<br>[PK] integer | id_departamento<br>integer | nombre_curso<br>character varying (50) | descripcion_c<br>text                      | creditos<br>integer | semestre<br>integer | id_campus<br>integer |
|---|--------------------------|----------------------------|--|--|---------------------|---------------------|----------------------|
| 1 | 1                        | 1                          | Programación I                         | Introducción a la programación             | 5                   | 1                   | 1                    |
| 2 | 2                        | 2                          | Circuitos Electricos I                 | Comportamiento de los Circuitos Electricos | 5                   | 4                   | 2                    |

-- INSCRIPCIONES

```
INSERT INTO inscripciones (id_estudiante, id_curso, fecha_inscripcion, calificacion) VALUES
(1, 1, '2025-01-15', 85),
(2, 2, '2025-01-16', 90),
(3, 2, '2025-02-01', 75);
SELECT* FROM inscripciones;
```

|   | id_inscripcion<br>[PK] integer | id_estudiante<br>integer | id_curso<br>integer | fecha_inscripcion<br>date | calificacion<br>integer |
|---|--------------------------------|--------------------------|---------------------|---------------------------|-------------------------|
| 1 | 4                              | 1                        | 1                   | 2025-01-15                | 85                      |
| 2 | 5                              | 2                        | 2                   | 2025-01-16                | 90                      |
| 3 | 6                              | 3                        | 2                   | 2025-02-01                | 75                      |

-- CURSOS\_PROFESORES

```
INSERT INTO cursos_profesores (id_curso, id_profesor) VALUES
(1, 1),
(2, 2);
select*from cursos_profesores;
```

|   | id_curso_profesor<br>[PK] integer | id_curso<br>integer | id_profesor<br>integer |
|---|-----------------------------------|---------------------|------------------------|
| 1 | 1                                 | 1                   | 1                      |
| 2 | 2                                 | 2                   | 2                      |

```
-- PROGRAMAS_CURSOS
INSERT INTO programas_cursos (id_programa, id_curso) VALUES
(1, 1),
(2, 2);
select*from programas_cursos;
```

|   | id_programa_curso<br>[PK] integer | id_programa<br>integer | id_curso<br>integer |
|---|-----------------------------------|------------------------|---------------------|
| 1 | 1                                 | 1                      | 1                   |
| 2 | 2                                 | 2                      | 2                   |

```
-- ESTUDIANTES_CARRERAS
INSERT INTO estudiantes_carreras (id_estudiante, id_carrera) VALUES
(1, 1),
(2, 2),
(3, 2);
select*from estudiantes_carreras;
```

|   | id_estudiante<br>[PK] integer | id_carrera<br>[PK] integer |
|---|-------------------------------|----------------------------|
| 1 | 1                             | 1                          |
| 2 | 2                             | 2                          |
| 3 | 3                             | 2                          |

3. **Actualizando Datos:** Actualizar la información existente en las tablas para reflejar cambios o correcciones.

```
--ACTUALIZACIONES DE DATOS
--CAMBIAR LA CALIFICACION DE UN ESTUDIANTE
UPDATE inscripciones
SET calificacion = 95
WHERE id_estudiante = 1 AND id_curso = 1;
```

Para mostrar que si la haya cambiado:



```
SELECT *
FROM inscripciones
WHERE id_estudiante = 1 AND id_curso = 1;
```

Showing rows: 1 to 1 | Page No. 1

|   | id_inscripcion<br>[PK] integer | id_estudiante<br>integer | id_curso<br>integer | fecha_inscripcion<br>date | calificacion<br>integer |
|---|--------------------------------|--------------------------|---------------------|---------------------------|-------------------------|
| 1 | 4                              | 1                        | 1                   | 2025-01-15                | 95                      |

```
--CORRECCION DEL CORREO DE UN ESTUDIANTE
UPDATE estudiantes
SET email = 'a2249076@itmexicali.edu.mx'
WHERE id_estudiante = 3;
```

```
SELECT id_estudiante, nombre_e, apellido_e, email
FROM estudiantes
WHERE id_estudiante = 3;
```

Showing rows: 1 to 1 | Page No. 1

|   | id_estudiante<br>[PK] integer | nombre_e<br>character varying (50) | apellido_e<br>character varying (50) | email<br>character varying (50) |
|---|-------------------------------|------------------------------------|--------------------------------------|---------------------------------|
| 1 | 3                             | José                               | Sánchez                              | a2249076@itmexicali.edu.mx      |

4. **Eliminando Datos:** Eliminar registros de las tablas que ya no sean relevantes.

```
--ELIMINANDO DATOS
--ELIMINAR UNA INSCRIPCION DE UN ESTUDIANTE
DELETE FROM inscripciones
WHERE id_estudiante = 2;
SELECT * FROM inscripciones;
```

|   | id_inscripcion<br>[PK] integer | id_estudiante<br>integer | id_curso<br>integer | fecha_inscripcion<br>date | calificacion<br>integer |
|---|--------------------------------|--------------------------|---------------------|---------------------------|-------------------------|
| 1 | 6                              | 3                        | 2                   | 2025-02-01                | 75                      |
| 2 | 4                              | 1                        | 1                   | 2025-01-15                | 95                      |

5. **Realizando Consultas (Búsquedas):** Formular y ejecutar consultas para obtener información específica de la base de datos.

### Consultas Específicas:

Los estudiantes deben formular consultas para responder a las siguientes solicitudes:

1. **Selección Básica:** Muestra todos los nombres y apellidos de los estudiantes.

--Mostrar todos los nombres y los apellidos de los estudiantes

```
SELECT nombre_e, apellido_e  
FROM estudiantes;
```

Showing rows: 1 to 3  Page No: 1 of 1

|   | nombre_e<br>character varying (50)  | apellido_e<br>character varying (50)  |
|---|--|--|
| 1 | Carlos   | Gómez  |
| 2 | Ana  | López  |
| 3 | José   | Sánchez  |

Selecciona nombre y apellido de la tabla estudiantes y los muestra.

2. **Cláusula WHERE:** Encuentra todos los cursos que tienen 3 créditos.

```
--Cursos que tienen 3 créditos  
SELECT * FROM cursos  
WHERE credits = 3;
```

Selecciona la tabla cursos y el where busca todos los valores que sean igual a 3 y los muestra, pero como en este caso no tenemos cursos que contengan 3 créditos, no muestra nada.

| id_curso<br>(PK) integer | id_departamento<br>integer | nombre_curso<br>character varying (50) | descripcion_c<br>text | credits<br>integer | semestre<br>integer | id_campus<br>integer |
|--------------------------|----------------------------|--|-----------------------|--------------------|---------------------|----------------------|
|--------------------------|----------------------------|--|-----------------------|--------------------|---------------------|----------------------|

3. **INNER JOIN:** Obtén una lista que muestre el nombre del estudiante y el nombre del curso en el que está inscrito.

```
-- Lista que del estudiante y el curso en el que está inscrito
SELECT e.nombre_e, e.apellido_e, c.nombre_curso
FROM estudiantes e
INNER JOIN inscripciones i
ON e.id_estudiante = i.id_estudiante
INNER JOIN cursos c
ON i.id_curso = c.id_curso;
```

|   | nombre_e<br>character varying (50) | apellido_e<br>character varying (50) | nombre_curso<br>character varying (50) |
|---|------------------------------------|--------------------------------------|--|
| 1 | José                               | Sánchez                              | Circuitos Electricos I                 |
| 2 | Carlos                             | Gómez                                | Programación I                         |

Se usa un INNER JOIN que devolverá la fila donde hay coincidencia en ambas tablas.

4. **LEFT JOIN:** Muestra todos los estudiantes y, si están inscritos en algún curso, el nombre del curso. Si un estudiante no está inscrito en ningún curso, el campo del nombre del curso debe mostrar un valor que lo indique (ej: NULL).

```
--Mostrar todos los estudiantes esten o no esten inscritos a un curso
SELECT e.nombre_e, e.apellido_e, c.nombre_curso
FROM estudiantes e
LEFT JOIN inscripciones i
ON e.id_estudiante = i.id_estudiante
LEFT JOIN cursos c
ON i.id_curso = c.id_curso;
```

|   | nombre_e<br>character varying (50) | apellido_e<br>character varying (50) | nombre_curso<br>character varying (50) |
|---|------------------------------------|--------------------------------------|--|
| 1 | José                               | Sánchez                              | Circuitos Electricos I                 |
| 2 | Carlos                             | Gómez                                | Programación I                         |
| 3 | Ana                                | López                                | [null]                                 |

El LEFT JOIN te mostrara todos los estudiantes independientemente de si tienen un curso o no, la diferencia entre el INNER JOIN es que el JOIN te muestra solo donde hay coincidencia.

5. **RIGHT JOIN:** Lista todos los cursos y, si tienen estudiantes inscritos, el nombre de los estudiantes. Muestra todos los cursos, incluso si no tienen estudiantes inscritos actualmente.

```
--Mostrar todos los cursos haya o no haya estudiantes inscritos
SELECT c.nombre_curso, e.nombre_e, e.apellido_e
FROM estudiantes e
RIGHT JOIN inscripciones i
ON e.id_estudiante = i.id_estudiante
RIGHT JOIN cursos c
ON i.id_curso = c.id_curso;
```

El **RIGHT JOIN** mostrara los registros de la tabla derecha, que son los cursos, estos aparecerán todos sin importar que tenga o no estudiantes.

6. **GROUP BY y COUNT:** Calcula cuántos estudiantes están inscritos en cada curso. Muestra el nombre del curso y la cantidad de estudiantes.

```
--MOSTRAR CURSOS Y CANTIDAD DE ESTUDIANTES INSCRITOS
SELECT c.nombre_curso, COUNT(i.id_estudiante) AS total_estudiantes
FROM cursos c
LEFT JOIN inscripciones i
ON c.id_curso = i.id_curso
GROUP BY c.nombre_curso;
```

Showing rows | Page No. |

Paste options

|   | nombre_curso<br>character varying (50) | total_estudiantes<br>bigint |
|---|--|-----------------------------|
| 1 | Circuitos Electricos I                 | 1                           |
| 2 | Programación I                         | 1                           |

Selecciona el nombre del curso y cuenta las inscripciones de cada estudiante, generándole un alias para mostrar el total de estudiantes, de la tabla cursos utilizara el **LEFT JOIN** para mostrar los cursos sin estudiantes inscritos. El **group by** agrupa los resultados.


7. **BETWEEN:** Encuentra todos los estudiantes que nacieron entre el 1 de enero de 1995 y el 31 de diciembre de 1998.

```
--estudiantes que nacieron entre el 1 de enero de 1995 y el 31 de diciembre de 1999
SELECT nombre_e, apellido_e, fecha_nacimiento
FROM estudiantes
WHERE fecha_nacimiento BETWEEN '1995-01-01' AND '1998-12-31';
```

El BETWEEN selecciona a los estudiantes que nacieron entre '1995-01-01' y '1998-12-31';

8. **ORDER BY:** Muestra todos los cursos ordenados alfabéticamente por su nombre.

```
--AGRUPAR CURSOS DE MANERA ALFABETICAMENTE
SELECT nombre_curso
FROM cursos
ORDER BY nombre_curso ASC;
```

|   | nombre_curso<br>character varying (50)  |
|---|--|
| 1 | Circuitos Electricos I   |
| 2 | Programación I   |

El ORDER BY agrupa los nombres de los cursos y los representará de manera ascendente, de la A a la Z.

9. **CTE:** Crea una tabla de expresión común que liste el número de inscripciones por estudiante. Luego, consulta esta CTE para encontrar los 3 estudiantes con más inscripciones, mostrando el nombre del estudiante y el número de inscripciones.

```
--crea una tabla temporal con número de inscripciones
--por estudiante y luego muestra los 3 con más inscripciones
WITH conteo_inscripciones AS (
    SELECT e.id_estudiante, e.nombre_e, COUNT(i.id_inscripcion) AS total_inscripciones
    FROM estudiantes e
    LEFT JOIN inscripciones i ON e.id_estudiante = i.id_estudiante
    GROUP BY e.id_estudiante, e.nombre_e
)
SELECT nombre_e, total_inscripciones
FROM conteo_inscripciones
ORDER BY total_inscripciones DESC
LIMIT 3;
```

|   | nombre_e<br>character varying (50) | total_inscripciones<br>bigint |
|---|------------------------------------|-------------------------------|
| 1 | José                               | 1                             |
| 2 | Carlos                             | 1                             |
| 3 | Ana                                | 0                             |

Primero crea una tabla temporal llamada IncripcionesPorEstudiante que calcula cuántas inscripciones tiene cada estudiante (COUNT). Luego, consulta esa tabla temporal para mostrar los 3 estudiantes con más inscripciones, ordenados de mayor a menor.

10. **Consulta Compleja 1:** Para cada departamento, muestra el nombre del departamento y el nombre del curso con la mayor cantidad de estudiantes inscritos.

```
WITH conteo AS (
    SELECT
        d.id_departamento,
        d.nombre_departamento,
        c.id_curso,
        c.nombre_curso,
        COUNT(i.id_estudiante) AS total_estudiantes
    FROM departamentos d
    JOIN cursos c ON d.id_departamento = c.id_departamento
    JOIN inscripciones i ON c.id_curso = i.id_curso
    GROUP BY d.id_departamento, d.nombre_departamento, c.id_curso, c.nombre_curso
),
maximos AS (
    SELECT
        id_departamento,
        MAX(total_estudiantes) AS max_estudiantes
    FROM conteo
    GROUP BY id_departamento
)
SELECT
    c.nombre_departamento,
    c.nombre_curso,
    c.total_estudiantes
FROM conteo c
JOIN maximos m ON c.id_departamento = m.id_departamento AND c.total_estudiantes = m.max_estudiantes;
```

|   | nombre_departamento<br>character varying (50) | nombre_curso<br>character varying (50) | total_estudiantes<br>bigint |
|---|---|--|-----------------------------|
| 1 | Ingeniería de Sistemas                        | Programación I                         | 1                           |
| 2 | Ingeniería Eléctrica                          | Circuitos Eléctricos I                 | 1                           |

conteo: calcula cuántos estudiantes hay en cada curso por departamento.

maximos: obtiene el número máximo de estudiantes por departamento.

Consulta final: junta ambas para sacar el curso más poblado por cada departamento

11. **Consulta Compleja 2:** Encuentra a los profesores que imparten más de dos cursos, mostrando su nombre, apellido y la cantidad de cursos que imparten.

```
-- muestra profesores que imparten más de dos cursos
SELECT p.nombre_p, p.apellido_p, COUNT(cp.id_curso) AS total_cursos
FROM profesores p
JOIN cursos_profesores cp
ON p.id_profesor = cp.id_profesor
GROUP BY p.id_profesor, p.nombre_p, p.apellido_p
HAVING COUNT(cp.id_curso) > 2;
```

| nombre_p<br>character varying (50) | apellido_p<br>character varying (50) | total_cursos<br>bigint |
|------------------------------------|--------------------------------------|------------------------|
|------------------------------------|--------------------------------------|------------------------|

12. **Consulta Compleja 3:** Lista los nombres de los programas de estudio y, para cada programa, el nombre del curso con el promedio de calificación más alto.



```

-- CTE para calcular el promedio de calificaciones por curso y programa
WITH promedios AS (
SELECT
p.id_programa, p.nombre_programa, c.id_curso, c.nombre_curso, AVG(i.calificacion) AS promedio_calificacion
FROM programasesudio p
JOIN programas_cursos pc
ON p.id_programa = pc.id_programa
JOIN cursos c ON pc.id_curso = c.id_curso
JOIN inscripciones i ON c.id_curso = i.id_curso
GROUP BY p.id_programa, p.nombre_programa, c.id_curso, c.nombre_curso
),
-- CTE para encontrar el máximo promedio por programa
maximos AS (
SELECT id_programa, MAX(promedio_calificacion) AS max_prom
FROM promedios
GROUP BY id_programa
)
-- Seleccionar solo los cursos con el promedio más alto por programa
SELECT
pr.nombre_programa,
pr.nombre_curso,
pr.promedio_calificacion

```

```

FROM promedios pr
JOIN maximos m
ON pr.id_programa = m.id_programa
AND pr.promedio_calificacion = m.max_prom;

```

|   | nombre_programa<br>character varying (50) | nombre_curso<br>character varying (50) | promedio_calificacion<br>numeric |
|---|---|--|----------------------------------|
| 1 | Programa de Desarrollo de Software        | Programación I                         | 95.0000000000000000              |
| 2 | Ingenieria Electrica                      | Circuitos Electricos I                 | 75.0000000000000000              |