Instituto Tecnologico De Mexicali

Revilla Moraila Joel Alberto

23490369

Fundamentos de Base de Datos

Bogarin Valenzuela Jose Ramon

Examen Final De la Unidad 4

20/05/2025

1.- Identificar Entidades:

Tablas Principales

Estudiantes: Almacena información personal de los estudiantes.

Departamentos: Contiene los departamentos académicos y su ubicación.

Cursos: Define los cursos ofrecidos, con su descripción, créditos y departamento asociado.

Profesores: Registra los profesores y su departamento.

Aulas: Información sobre las aulas disponibles.

ProgramasEstudio: Define los programas académicos ofrecidos.

Tablas de Relación

Inscripciones: Relaciona estudiantes con cursos (matrículas).

Horarios: Asignar cursos a aulas con sus horarios específicos.

CursosProfesores: Relación muchos-a-muchos entre cursos y profesores.

ProgramasCursos: Relación muchos-a-muchos entre programas y cursos.

2.- Diseñando Las Tablas:

Estudiantes

Nombre	Dato	PK / FK
IDEstudiantes	serial	primary key
Nombre	varchar	
Apellido	varchar	
FechaNacimiento	Date	
Direccion	varchar	
Ciudad	varchar	
Email	varchar	

Cursos

Nombre	Dato	PK / FK
IDCurso	Serial	Primary key
NombreCurso	varchar	
Descripcion	Text	
Creditos	Int	
Semestre	Varchar	
IDDepartamento	Int	Foreign key

Inscripciones

Nombre	Dato	PK / FK
IDInscripcion	Serial	Primary Key
IDEstudiante	Int	Foreign Key
IDCurso	Int	Foreing Key
FechaInscripcion	Date	
Calificaion	Decimal	

Profesores

Nombre	Dato	PK /FK
IDProfesores	Serial	Primary Key
Nombre	Varchar	
Apellido	Varchar	
Titulo	Varchar	
IDDepartamento	Int	Foreign Key

Departamentos

Nombre	Dato	PK /FK
IDDepartamento	Serial	Primary Key
NombreDepartamento	Varchar	
Edificio	Varchar	

Aulas

Nombre	Dato	PK / FK
IDAula	Serial	Primary Key
NombreAula	Varchar	
Capacidad	Int	
Ubicacion	Varchar	

Horarios

Nombre	Dato	PK / FK
IDHorario	Serial	Primary key
IDCurso	Serial	Foreing key
IDAula	Serial	Foreing key
FechaInicio	Date	
FechaFin	Date	
HoraInicio	Time	
HoraFin	Time	

Tabla CursosProfesores

Nombre	Dato	PK / FK
IDcursoProfesor	Serial	Primary Key
IDCurso	Serial	Foreing Key
IDProfesor	Serial	Foreing Key

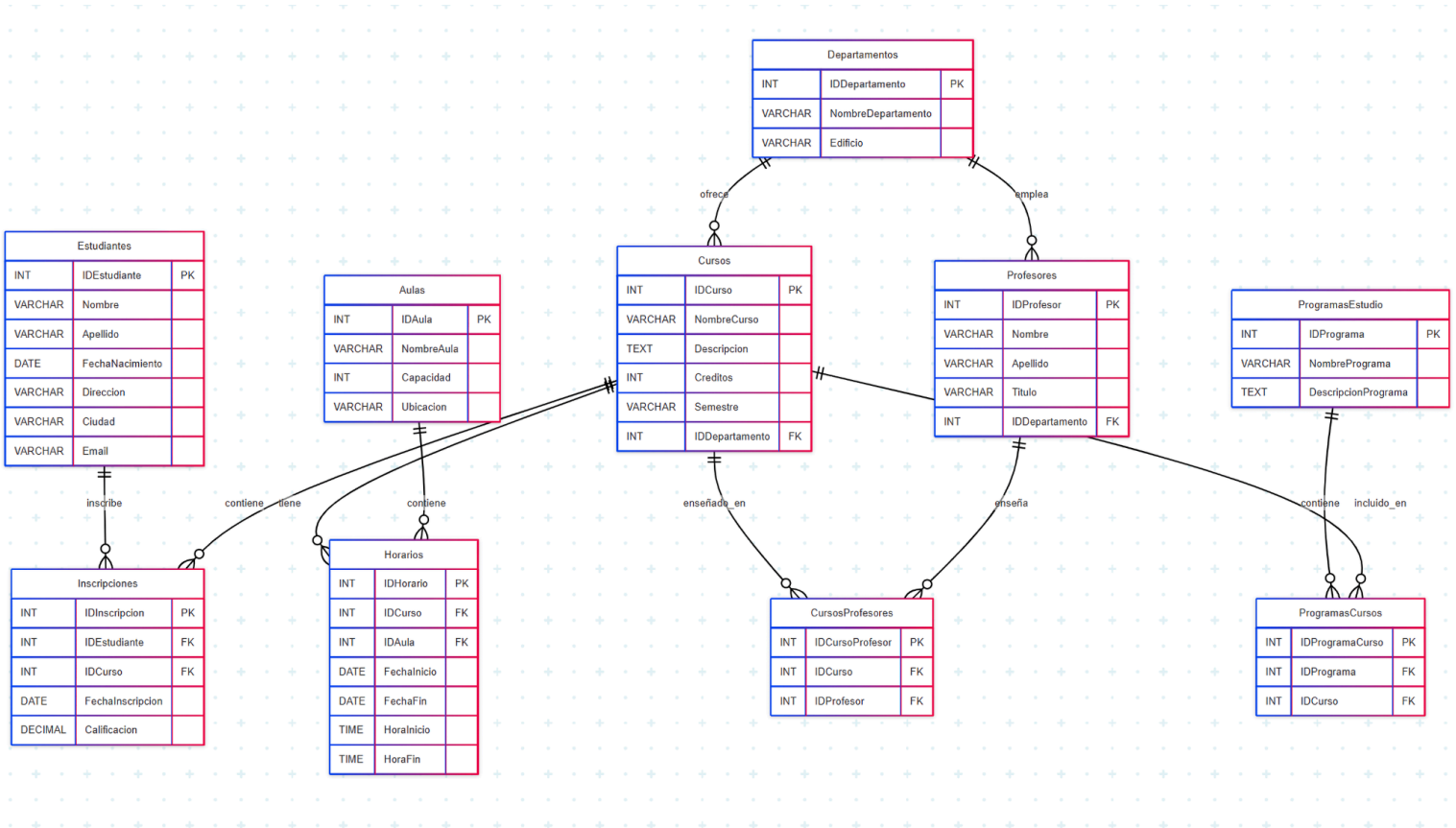
Tabla ProgramasEstudio

Nombre	Dato	PK / FK
IDPrograma	Serial	Primary Key
NombrePrograma	Varchar	
DescripcionPrograma	Text	

Tabla ProgramasCursos

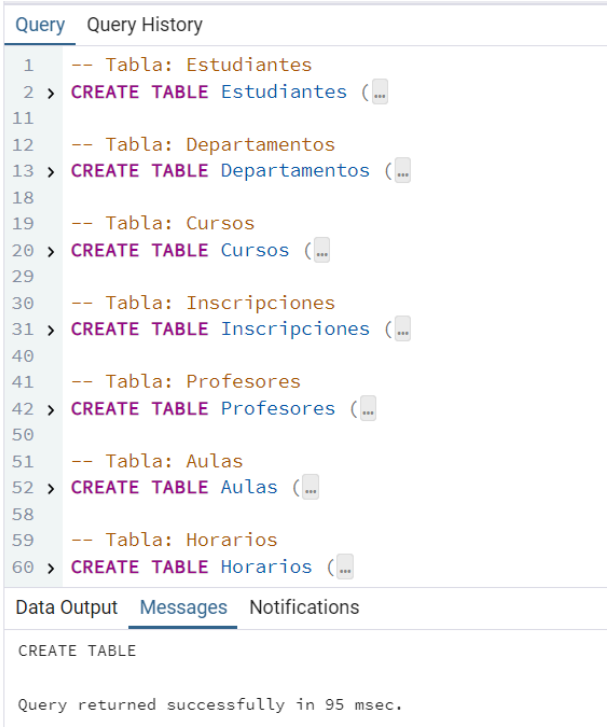
Nombre	Dato	PK / FK
IDProgramaCurso	Serial	Primary Key
IDPrograma	Serial	Foreign Key
IDCurso	Serial	Foreign Key

3.- Creando Tablas:



Version en UML, (simplemente para usarse de referencia)

Query




```
-- Tabla: Estudiantes
CREATE TABLE Estudiantes (
    IDEstudiante SERIAL PRIMARY KEY,
    Nombre VARCHAR(100),
    Apellido VARCHAR(100),
    FechaNacimiento DATE,
    Direccion VARCHAR(200),
    Ciudad VARCHAR(100),
    Email VARCHAR(100)
);

-- Tabla: Departamentos
CREATE TABLE Departamentos (
    IDDepartamento SERIAL PRIMARY KEY,
    NombreDepartamento VARCHAR(100),
    Edificio VARCHAR(100)
);

-- Tabla: Cursos
CREATE TABLE Cursos (
    IDCurso SERIAL PRIMARY KEY,
    NombreCurso VARCHAR(100),
    Descripcion TEXT,
    Creditos INT,
    Semestre VARCHAR(20),
    IDDepartamento INT,
    FOREIGN KEY (IDDepartamento) REFERENCES Departamentos(IDDepartamento)
);
```

```
-- Tabla: Inscripciones
CREATE TABLE Inscripciones (
    IDInscripcion SERIAL PRIMARY KEY,
    IDEstudiante INT,
    IDCurso INT,
    FechaInscripcion DATE,
    Calificacion DECIMAL(4,2),
    FOREIGN KEY (IDEstudiante) REFERENCES Estudiantes(IDEstudiante),
    FOREIGN KEY (IDCurso) REFERENCES Cursos(IDCurso)
);

-- Tabla: Profesores
CREATE TABLE Profesores (
    IDProfesor SERIAL PRIMARY KEY,
    Nombre VARCHAR(100),
    Apellido VARCHAR(100),
    Titulo VARCHAR(100),
    IDDepartamento INT,
    FOREIGN KEY (IDDepartamento) REFERENCES Departamentos(IDDepartamento)
);

-- Tabla: Aulas
CREATE TABLE Aulas (
    IDAula SERIAL PRIMARY KEY,
    NombreAula VARCHAR(100),
    Capacidad INT,
    Ubicacion VARCHAR(100)
);

-- Tabla: Horarios
CREATE TABLE Horarios (
    IDHorario SERIAL PRIMARY KEY,
    IDCurso SERIAL,
    IDAula SERIAL,
    FechaInicio DATE,
    FechaFin DATE,
    HoraInicio TIME,
    HoraFin TIME,
    FOREIGN KEY (IDCurso) REFERENCES Cursos(IDCurso),
    FOREIGN KEY (IDAula) REFERENCES Aulas(IDAula)
);

-- Tabla Intermedia: CursosProfesores
CREATE TABLE CursosProfesores (
    IDCursoProfesor SERIAL PRIMARY KEY,
    IDCurso Serial,
    IDProfesor Serial,
```

```
    FOREIGN KEY (IDCurso) REFERENCES Cursos(IDCurso),
    FOREIGN KEY (IDProfesor) REFERENCES Profesores(IDProfesor)
);

-- Tabla: ProgramasEstudio
CREATE TABLE ProgramasEstudio (
    IDPrograma SERIAL PRIMARY KEY,
    NombrePrograma VARCHAR(100),
    DescripcionPrograma TEXT
);

-- Tabla Intermedia: ProgramasCursos
CREATE TABLE ProgramasCursos (
    IDProgramaCurso SERIAL PRIMARY KEY,
    IDPrograma SERIAL,
    IDCurso SERIAL,
    FOREIGN KEY (IDPrograma) REFERENCES ProgramasEstudio(IDPrograma),
    FOREIGN KEY (IDCurso) REFERENCES Cursos(IDCurso)
);
```

4.- Modificar las Tablas DDL:

Agregar tablas:

Query | Query History

```
1  -- Tabla: Campus
2  CREATE TABLE Campus (
3      IDCampus SERIAL PRIMARY KEY,
4      NombreCampus VARCHAR(100) NOT NULL,
5      DireccionCampus VARCHAR(200) NOT NULL
6  );
7
8  -- Tabla: Carreras
9  CREATE TABLE Carreras (
10     IDCarrera SERIAL PRIMARY KEY,
11     NombreCarrera VARCHAR(100) NOT NULL,
12     TituloOtorgado VARCHAR(100) NOT NULL
13 );
14
15 -- Tabla intermedia: EstudiantesCarreras (relación muchos a muchos)
16 CREATE TABLE EstudiantesCarreras (
17     IDEstudianteCarrera SERIAL PRIMARY KEY,
18     IDEstudiante INT NOT NULL,
19     IDCarrera INT NOT NULL,
20     FechaInicio DATE,
21     FechaFin DATE,
22     FOREIGN KEY (IDEstudiante) REFERENCES Estudiantes(IDEstudiante),
23     FOREIGN KEY (IDCarrera) REFERENCES Carreras(IDCarrera),
24     UNIQUE (IDEstudiante, IDCarrera) -- Evita duplicados
25 );
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 308 msec.
```

las tablas EstudianteCarreras, campus y Carreras fueron añadidas

Modificar Tablas:

```sql
-- Eliminar columna Ciudad de Estudiantes
ALTER TABLE Estudiantes DROP COLUMN Ciudad;

-- Agregar campo Email a Profesores
ALTER TABLE Profesores ADD COLUMN Email VARCHAR(100);

-- Agregar IDCampus a Cursos
ALTER TABLE Cursos ADD COLUMN IDCampus INT;
ALTER TABLE Cursos
ADD CONSTRAINT fk_cursos_campus
FOREIGN KEY (IDCampus) REFERENCES Campus(IDCampus);

-- Agregar IDCarrera a Estudiantes (aunque también tenemos la relación muchos a muchos)
ALTER TABLE Estudiantes ADD COLUMN IDCarrera INT;
ALTER TABLE Estudiantes
ADD CONSTRAINT fk_estudiantes_carrera
FOREIGN KEY (IDCarrera) REFERENCES Carreras(IDCarrera);
```

Data Output | Messages | Notifications

```
ALTER TABLE

Query returned successfully in 77 msec.
```

los campos fueron cambiados, para que así concuerden con las nuevas tablas añadidas

Eliminar Tablas:

Query | Query History

```sql
-- Eliminar tabla Aulas (y sus referencias)
-- Primero necesitamos eliminar las dependencias en Horarios
ALTER TABLE Horarios DROP CONSTRAINT horarios_idaula_fkey;
DROP TABLE Aulas;
```

Data Output | Messages | Notifications

```
DROP TABLE

Query returned successfully in 51 msec.
```

Quitamos la tabla aula junto a sus dependencias.

5.- Inserción de Datos:

añadimos datos o información dentro de las tablas, para lógicamente no tener ningún elemento vacío.

6.- Actualizando Datos:

Actualización 1, sobre datos acerca de estudiantes

Query | Query History

```sql
-- Corregir el email de un estudiante (error tipográfico)
UPDATE Estudiantes
SET Email = 'mrodriguez@universidad.edu'
WHERE IDEstudiante = 4;

-- Actualizar la dirección de un estudiante que se mudó
UPDATE Estudiantes
SET Direccion = 'Av. Nueva Vida 789, Urb. Las Flores'
WHERE IDEstudiante = 3;

-- Cambiar la carrera principal de un estudiante (de Psicología a Derecho)
UPDATE Estudiantes
SET IDCarrera = 3
WHERE IDEstudiante = 5;
```

Data Output | Messages | Notifications

```
UPDATE 1

Query returned successfully in 144 msec.
```

Actualización 2, sobre la información académica

Query | Query History

```sql
-- Corregir una calificación erronea
UPDATE Inscripciones
SET Calificacion = 17.5
WHERE IDInscripcion = 1;

-- Cambiar el semestre de un curso
UPDATE Cursos
SET Semestre = '2023-2'
WHERE IDCurso = 5;

-- Actualizar los créditos de un curso
UPDATE Cursos
SET Creditos = 5
WHERE NombreCurso = 'Programación I';
```

Data Output | Messages | Notifications

```
UPDATE 1

Query returned successfully in 106 msec.
```

Actualización 3, acerca de los datos de los profesores

Query | Query History

```sql
-- Actualizar el título de un profesor
UPDATE Profesores
SET Titulo = 'PhD en Inteligencia Artificial'
WHERE IDProfesor = 1;

-- Corregir email de un profesor
UPDATE Profesores
SET Email = 'carlos.mendoza@universidad.edu'
WHERE IDProfesor = 1;

-- Cambiar departamento de un profesor
UPDATE Profesores
SET IDDepartamento = 4
WHERE IDProfesor = 5;
```

Data Output | Messages | Notifications

```
UPDATE 1

Query returned successfully in 72 msec.
```

Actualización 4, acerca de las relaciones sobre muchos a muchos

Query | Query History

```sql
-- Actualizar fecha de finalización para una carrera completada
UPDATE EstudiantesCarreras
SET FechaFin = '2023-12-20'
WHERE IDEstudianteCarrera = 5;

-- Cambiar el estado de una doble carrera (de activa a abandonada)
UPDATE EstudiantesCarreras
SET FechaFin = '2023-06-30'
WHERE IDEstudianteCarrera = 6;
```

Data Output | Messages | Notifications

```
UPDATE 1

Query returned successfully in 67 msec.
```

Actualización 5, acerca de información institucional

Query | Query History

```sql
-- Cambiar el nombre de un campus
UPDATE Campus
SET NombreCampus = 'Campus Principal'
WHERE IDCampus = 1;

-- Actualizar la dirección de un campus
UPDATE Campus
SET DireccionCampus = 'Av. Universitaria 1234, Ciudad Universitaria Ampliada'
WHERE IDCampus = 1;

-- Cambiar el título otorgado por una carrera
UPDATE Carreras
SET TituloOtorgado = 'Licenciado en Psicología'
WHERE IDCarrera = 5;
```

Data Output | Messages | Notifications

```
UPDATE 1

Query returned successfully in 73 msec.
```

Actualización 6, acerca de cursos

Query | Query History

```sql
-- Cambiar la descripción de un programa
UPDATE ProgramasEstudio
SET DescripcionPrograma = 'Programa para estudiantes con promedio mayor a 16.0'
WHERE IDPrograma = 1;

-- Eliminar un curso de un programa especial
DELETE FROM ProgramasCursos
WHERE IDPrograma = 3 AND IDCurso = 5;
```

Data Output | Messages | Notifications

```
DELETE 1

Query returned successfully in 73 msec.
```

7,- Eliminación de datos no relevantes

Eliminar Estudiantes inactivos

Query | Query History

```sql
-- Eliminar estudiantes que no tienen inscripciones a cursos
DELETE FROM Estudiantes
WHERE IDEstudiante NOT IN (SELECT DISTINCT IDEstudiante FROM Inscripciones);

-- Eliminar estudiantes que dejaron la universidad hace más de 5 años
-- (Primero eliminamos sus relaciones en EstudiantesCarreras)
DELETE FROM EstudiantesCarreras
WHERE IDEstudiante IN (
    SELECT IDEstudiante
    FROM Estudiantes
    WHERE FechaNacimiento < '1995-01-01'
);

DELETE FROM Estudiantes
WHERE FechaNacimiento < '1995-01-01';
```

Data Output | Messages | Notifications

```
DELETE 0

Query returned successfully in 140 msec.
```

Eliminar cursos obsoletos

Query | Query History

```sql
-- Eliminar cursos que no tienen inscripciones ni están en programas
DELETE FROM Cursos
WHERE IDCurso NOT IN (
    SELECT DISTINCT IDCurso FROM Inscripciones
) AND IDCurso NOT IN (
    SELECT DISTINCT IDCurso FROM ProgramasCursos
);

-- Eliminar cursos que no se ofrecen desde hace más de 3 años
-- (Primero eliminamos relaciones en CursosProfesores y Horarios)
DELETE FROM CursosProfesores
WHERE IDCurso IN (
    SELECT IDCurso
    FROM Cursos
    WHERE Semestre < '2020-2'
);

DELETE FROM Horarios...

DELETE FROM Cursos
WHERE Semestre < '2020-2';
```

Data Output | Messages | Notifications

```
DELETE 0

Query returned successfully in 63 msec.
```

Eliminar Relaciones obsoletas:

Query | Query History

```
1  -- Eliminar inscripciones con calificación 0 (consideradas no válidas)
2  DELETE FROM Inscripciones
3  WHERE Calificacion = 0;
4
5  -- Eliminar relaciones estudiantes-carreras terminadas hace más de 10 años
6  DELETE FROM EstudiantesCarreras
7  WHERE FechaFin < '2013-01-01';
```

Data Output | Messages | Notifications

```
DELETE 0

Query returned successfully in 128 msec.
```

Eliminación de Cursos inactivos

Query | Query History

```
1  -- Eliminar programas que no tienen cursos asociados
2  DELETE FROM ProgramasEstudio
3  WHERE IDPrograma NOT IN (
4      SELECT DISTINCT IDPrograma FROM ProgramasCursos
5  );
6
7  -- Eliminar relaciones programas-cursos para cursos ya eliminados
8  DELETE FROM ProgramasCursos
9  WHERE IDCurso NOT IN (SELECT IDCurso FROM Cursos);
```

Data Output | Messages | Notifications

```
DELETE 0

Query returned successfully in 55 msec.
```

Eliminar profes sin cursos

Query | Query History

```
1  -- Eliminar profesores que no están asignados a ningún curso
2  DELETE FROM Profesores
3  WHERE IDProfesor NOT IN (
4      SELECT DISTINCT IDProfesor FROM CursosProfesores
5  );
```

Data Output | Messages | Notifications

```
DELETE 0

Query returned successfully in 45 msec.
```

Eliminar Campus sin Cursos

Query | Query History

```sql
-- Eliminar campus que no tienen cursos asignados
DELETE FROM Campus
WHERE IDCampus NOT IN (
    SELECT DISTINCT IDCampus FROM Cursos WHERE IDCampus IS NOT NULL
);
```

Data Output | Messages | Notifications

```
DELETE 0

Query returned successfully in 139 msec.
```

8.- Realizar consultas:

Nombres y Apellidos de estudiantes:

Query | Query History

```
1  SELECT Nombre, Apellido
2  FROM Estudiantes;
```

Data Output | Messages | Notifications

	nombre character varying (100)	apellido character varying (100)
1	Juan	Pérez
2	Sofía	García
3	Laura	Martínez
4	Miguel	López
5	Diego	Ramírez

where con 3 creditos

Query | Query History

```
1  SELECT NombreCurso, Creditos
2  FROM Cursos
3  WHERE Creditos = 3;
```

Data Output | Messages | Notifications

	nombrecurso character varying (100)	creditos integer

inner join con estudiantes y sus cursos:

Query | Query History

```
1  SELECT e.Nombre, e.Apellido, c.NombreCurso
2  FROM Estudiantes e
3  INNER JOIN Inscripciones i ON e.IDEstudiante = i.IDEstudiante
4  INNER JOIN Cursos c ON i.IDCurso = c.IDCurso;
```

Data Output | Messages | Notifications

	nombre character varying (100)	apellido character varying (100)	nombrecurso character varying (100)
1	Sofía	García	Anatomía Humana
2	Miguel	López	Derecho Civil
3	Laura	Martínez	Contabilidad Financiera
4	Diego	Ramírez	Psicología General
5	Juan	Pérez	Derecho Civil
6	Sofía	García	Psicología General
7	Juan	Pérez	Programación I

left join con todos los estudiantes:

Query | Query History

```
1 SELECT e.Nombre, e.Apellido, COALESCE(c.NombreCurso, 'No inscrito') AS Curso
2 FROM Estudiantes e
3 LEFT JOIN Inscripciones i ON e.IDEstudiante = i.IDEstudiante
4 LEFT JOIN Cursos c ON i.IDCurso = c.IDCurso;
```

Data Output | Messages | Notifications

	nombre character varying (100)	apellido character varying (100)	curso character varying
1	Sofía	García	Anatomía Humana
2	Miguel	López	Derecho Civil
3	Laura	Martínez	Contabilidad Financiera
4	Diego	Ramírez	Psicología General
5	Juan	Pérez	Derecho Civil
6	Sofía	García	Psicología General
7	Juan	Pérez	Programación I

right join

Query | Query History

```
1 SELECT c.NombreCurso, COALESCE(e.Nombre || ' ' || e.Apellido, 'Sin estudiantes') AS Estudiante
2 FROM Estudiantes e
3 RIGHT JOIN Inscripciones i ON e.IDEstudiante = i.IDEstudiante
4 RIGHT JOIN Cursos c ON i.IDCurso = c.IDCurso;
```

Data Output | Messages | Notifications

	nombrecurso character varying (100)	estudiante text
1	Anatomía Humana	Sofía García
2	Derecho Civil	Miguel López
3	Contabilidad Financiera	Laura Martínez
4	Psicología General	Diego Ramírez
5	Derecho Civil	Juan Pérez
6	Psicología General	Sofía García
7	Programación I	Juan Pérez

group by y count

Query | Query History

```
1 SELECT c.NombreCurso, COUNT(i.IDEstudiante) AS CantidadEstudiantes
2 FROM Cursos c
3 LEFT JOIN Inscripciones i ON c.IDCurso = i.IDCurso
4 GROUP BY c.NombreCurso;
```

Data Output | Messages | Notifications

	nombrecurso character varying (100)	cantidadestudiantes bigint
1	Contabilidad Financiera	1
2	Psicología General	2
3	Derecho Civil	2
4	Programación I	1
5	Anatomía Humana	1

between

Query | Query History

```
1 SELECT Nombre, Apellido, FechaNacimiento
2 FROM Estudiantes
3 WHERE FechaNacimiento BETWEEN '1995-01-01' AND '1998-12-31';
```

Data Output | Messages | Notifications

	nombre character varying (100)	apellido character varying (100)	fechanacimiento date

order by:

Query | Query History

```
1 SELECT NombreCurso, Creditos, Semestre
2 FROM Cursos
3 ORDER BY NombreCurso ASC;
```

Data Output | Messages | Notifications

	nombrecurso character varying (100)	creditos integer	semestre character varying (20)
1	Anatomía Humana	6	2023-1
2	Contabilidad Financiera	4	2023-2
3	Derecho Civil	5	2023-2
4	Programación I	5	2023-1
5	Psicología General	4	2023-2

estudiantes con mas inscripciones

Query | Query History

```
1  WITH InscripcionesPorEstudiante AS (
2      SELECT e.IDEstudiante, e.Nombre, e.Apellido, COUNT(i.IDInscripcion) AS TotalInscripciones
3      FROM Estudiantes e
4      LEFT JOIN Inscripciones i ON e.IDEstudiante = i.IDEstudiante
5      GROUP BY e.IDEstudiante, e.Nombre, e.Apellido
6  )
7  SELECT Nombre, Apellido, TotalInscripciones
8  FROM InscripcionesPorEstudiante
9  ORDER BY TotalInscripciones DESC
10 LIMIT 3;
```

Data Output | Messages | Notifications

	nombre character varying (100)	apellido character varying (100)	totalinscripciones bigint
1	Sofía	García	2
2	Juan	Pérez	2
3	Laura	Martínez	1

Consulta compleja 1:

```
WITH EstudiantesPorCurso AS (
    SELECT c.IDCurso, c.NombreCurso, c.IDDepartamento, COUNT(i.IDEstudiante) AS CantEstudiantes
    FROM Cursos c
    LEFT JOIN Inscripciones i ON c.IDCurso = i.IDCurso
    GROUP BY c.IDCurso, c.NombreCurso, c.IDDepartamento
),
MaxEstudiantesPorDepto AS (
    SELECT IDDepartamento, MAX(CantEstudiantes) AS MaxEstudiantes
    FROM EstudiantesPorCurso
    GROUP BY IDDepartamento
)
SELECT d.NombreDepartamento, ec.NombreCurso, ec.CantEstudiantes
FROM Departamentos d
JOIN EstudiantesPorCurso ec ON d.IDDepartamento = ec.IDDepartamento
JOIN MaxEstudiantesPorDepto med ON ec.IDDepartamento = med.IDDepartamento AND ec.CantEstudiantes = med.MaxEstudiantes;
```

	nombredepartamento character varying (100)	nombrecurso character varying (100)	cantestudiantes bigint
1	Ciencias Jurídicas	Derecho Civil	2
2	Psicología	Psicología General	2
3	Administración	Contabilidad Financiera	1
4	Medicina	Anatomía Humana	1
5	Ciencias de la Computación	Programación I	1

Consulta compleja 2:

```
SELECT p.Nombre, p.Apellido, COUNT(cp.IDCurso) AS CantidadCursos
FROM Profesores p
JOIN CursosProfesores cp ON p.IDProfesor = cp.IDProfesor
GROUP BY p.IDProfesor, p.Nombre, p.Apellido
HAVING COUNT(cp.IDCurso) > 2;
```

	nombre character varying (100)	apellido character varying (100)	cantidadcursos bigint

consulta compleja 3:

```
WITH PromediosPorCurso AS (
    SELECT pc.IDPrograma, i.IDCurso, AVG(i.Calificacion) AS Promedio
    FROM ProgramasCursos pc
    JOIN Inscripciones i ON pc.IDCurso = i.IDCurso
    GROUP BY pc.IDPrograma, i.IDCurso
),
MaxPromedioPorPrograma AS (
    SELECT IDPrograma, MAX(Promedio) AS MaxPromedio
    FROM PromediosPorCurso
    GROUP BY IDPrograma
)
SELECT pe.NombrePrograma, c.NombreCurso, ppc.Promedio
FROM ProgramasEstudio pe
JOIN PromediosPorCurso ppc ON pe.IDPrograma = ppc.IDPrograma
JOIN MaxPromedioPorPrograma mpp ON ppc.IDPrograma = mpp.IDPrograma AND ppc.Promedio = mpp.MaxPromedio
JOIN Cursos c ON ppc.IDCurso = c.IDCurso
ORDER BY pe.NombrePrograma;
```

	nombreprograma character varying (100)	nombrecurso character varying (100)	promedio numeric
1	Doble Titulación	Contabilidad Financiera	17.0000000000000000
2	Intercambio Internacional	Derecho Civil	13.7500000000000000
3	Programa de Honor	Anatomía Humana	18.0000000000000000