INSTITUTO TECNOLOGICO DE MEXICO

INGENIERO EN SISTEMAS COMPUTACIONALES

Armas Diaz Erick Hidekio N.ctrol : 23490411

FUNDAMENTOS DE BASE DE DATOS

Tarea 2 Unidad 2 Fecha: 21/03/25

Codigos SQLS y sus Entidades

1.Sistema de Gestion de Inventarios

Entidades:

- Producto (id, nombre, precio, id_categoria, id_proveedor)
- Categoría (id, nombre)
- Proveedor (id, nombre, contacto)
- Inventario (id_producto, stock, ubicacion)

```
CREATE TABLE Categoria (
  id SERIAL PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL
);


CREATE TABLE Proveedor (
  id SERIAL PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  contacto VARCHAR(100)
);
```

```
CREATE TABLE Producto (
   id SERIAL PRIMARY KEY,
   nombre VARCHAR(100) NOT NULL,
   precio DECIMAL(10,2) NOT NULL,
   id_categoria INT REFERENCES Categoria(id),
   id_proveedor INT REFERENCES Proveedor(id)
);

CREATE TABLE Inventario (
   id_producto INT PRIMARY KEY REFERENCES
Producto(id),
   stock INT NOT NULL,
   ubicacion VARCHAR(100) NOT NULL
);
```

ERD y Consultas

2.Sistema de Gestion de Eventos.

Codigo SQL y Entidades.

Entidades:

- Evento (id, nombre, fecha, id_ubicacion, id_organizador)
- Ubicación (id, direccion)
- Organizador (id, nombre, contacto)
- Participante (id, nombre, email)
- Registro (id_evento, id_participante)

```
CREATE TABLE Ubicacion (
  id SERIAL PRIMARY KEY,
  direccion VARCHAR(255) NOT NULL
);

CREATE TABLE Organizador (
  id SERIAL PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  contacto VARCHAR(100)
);

CREATE TABLE Evento (
  id SERIAL PRIMARY KEY,
```

```
    nombre VARCHAR(100) NOT NULL,
    fecha DATE NOT NULL,
    id_ubicacion INT REFERENCES Ubicacion(id),
    id_organizador INT REFERENCES Organizador(id)
);

CREATE TABLE Participante (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL
);

CREATE TABLE Registro (
    id_evento INT REFERENCES Evento(id),
    id_participante INT REFERENCES Participante(id),
    PRIMARY KEY (id_evento, id_participante)
);
```

ERD y Consultas.

```sql
-- Obtener productos con sus categorías y proveedores
SELECT p.nombre AS producto, c.nombre AS categoria, pr.nombre AS proveedor
FROM Producto p
INNER JOIN Categoria c ON p.id_categoria = c.id
INNER JOIN Proveedor pr ON p.id_proveedor = pr.id
ORDER BY p.nombre;

-- Contar cuántos productos hay en cada categoría
SELECT c.nombre AS categoria, COUNT(p.id) AS total_productos
FROM Categoria c
LEFT JOIN Producto p ON c.id = p.id_categoria
GROUP BY c.nombre;

-- Productos con stock menor a 20 unidades
SELECT p.nombre AS producto, i.stock
FROM Producto p
```

Data Output | Messages | Notifications

	producto character varying (100)	categoria character varying (100)	proveedor character varying (100)
1	Balón de Fútbol Adidas	Deportes y Fitness	Adidas Group
2	Escritorio Moderna	Muebles	IKEA Furniture
3	Libro: Inteligencia Artificial	Libros y Educación	Amazon Books
4	Nike Air Max 2023	Ropa y Moda	Nike Inc.
5	Samsung Galaxy S23	Electrónica	Samsung Electronics

```sql
-- Contar cuántos productos hay en cada categoría
SELECT c.nombre AS categoria, COUNT(p.id) AS total_productos
FROM Categoria c
LEFT JOIN Producto p ON c.id = p.id_categoria
GROUP BY c.nombre;

-- Productos con stock menor a 20 unidades
SELECT p.nombre AS producto, i.stock
FROM Producto p
```

Data Output | Messages | Notifications

	categoria character varying (100)	total_productos bigint
1	Juguetes y Juegos	0
2	Alimentos y Bebidas	0
3	Electrónica	1
4	Deportes y Fitness	1
5	Automotriz	0
6	Ropa y Moda	1
7	Muebles	1
8	Salud y Belleza	0
9	Libros y Educación	1
10	Oficina y Papelería	0

```sql
-- Productos con stock menor a 20 unidades
SELECT p.nombre AS producto, i.stock
FROM Producto p
INNER JOIN Inventario i ON p.id = i.id_producto
WHERE i.stock < 20;
```

Data Output | Messages | Notifications

	producto character varying (100)	stock integer
1	Escritorio Moderna	15

2. Sistema de Gestión de Eventos

Entidades:

- Evento (id, nombre, fecha, id_ubicacion, id_organizador)
- Ubicación (id, direccion)
- Organizador (id, nombre, contacto)
- Participante (id, nombre, email)
- Registro (id_evento, id_participante)

Sql.

```
CREATE TABLE Ubicacion (
  id SERIAL PRIMARY KEY,
  direccion VARCHAR(255) NOT NULL
);

CREATE TABLE Organizador (
  id SERIAL PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  contacto VARCHAR(100)
);

CREATE TABLE Evento (
```

```
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    fecha DATE NOT NULL,
    id_ubicacion INT REFERENCES Ubicacion(id),
    id_organizador INT REFERENCES Organizador(id)
);

CREATE TABLE Participante (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL
);

CREATE TABLE Registro (
    id_evento INT REFERENCES Evento(id),
    id_participante INT REFERENCES Participante(id),
    PRIMARY KEY (id_evento, id_participante)
);
```

ERD y Consultas.

```sql
-- Listar eventos con la cantidad de participantes
SELECT e.nombre AS evento, COUNT(r.id_participante) AS cantidad_participantes
FROM Evento e
LEFT JOIN Registro r ON e.id = r.id_evento
GROUP BY e.nombre
ORDER BY cantidad_participantes DESC;

-- Listar participantes inscritos en la "Feria Internacional del Libro"
SELECT p.nombre, p.email
FROM Registro r
INNER JOIN Participante p ON r.id_participante = p.id
INNER JOIN Evento e ON r.id_evento = e.id
WHERE e.nombre = 'Feria Internacional del Libro';

-- Mostrar eventos que no tienen participantes registrados
SELECT e.nombre AS evento
```

Data Output | Messages | Notifications

	evento character varying (100)	cantidad_participantes bigint
1	Concierto Coldplay	2
2	Feria Internacional del Libro	2
3	Congreso de Innovación Tecnológica	1

```sql
-- Listar participantes inscritos en la "Feria Internacional del Libro"
SELECT p.nombre, p.email
FROM Registro r
INNER JOIN Participante p ON r.id_participante = p.id
INNER JOIN Evento e ON r.id_evento = e.id
WHERE e.nombre = 'Feria Internacional del Libro';

-- Mostrar eventos que no tienen participantes registrados
SELECT e.nombre AS evento
```

Data Output | Messages | Notifications

	nombre character varying (100)	email character varying (100)
1	Carlos Duarte	carlos.duarte@email.com
2	Laura Pineda	laura.pineda@email.com

```sql
-- Mostrar eventos que no tienen participantes registrados
SELECT e.nombre AS evento
FROM Evento e
LEFT JOIN Registro r ON e.id = r.id_evento
WHERE r.id_participante IS NULL;
```

Data Output | Messages | Notifications

	evento character varying (100)

3. Plataforma de Sreaming de Musica.

Codigo SQL y Entidades.

Entidades:

- Usuario (id, nombre, email)
- Artista (id, nombre)
- Álbum (id, nombre, id_artista)
- Canción (id, nombre, id_album)
- Historial (id_usuario, id_cancion, fecha)

SQL.

```
CREATE TABLE Usuario (
   id SERIAL PRIMARY KEY,
   nombre VARCHAR(100) NOT NULL,
   email VARCHAR(100) UNIQUE NOT NULL
);


CREATE TABLE Artista (
   id SERIAL PRIMARY KEY,
   nombre VARCHAR(100) NOT NULL
);
```

```
CREATE TABLE Album (
   id SERIAL PRIMARY KEY,
   nombre VARCHAR(100) NOT NULL,
   id_artista INT REFERENCES Artista(id)
);

CREATE TABLE Cancion (
   id SERIAL PRIMARY KEY,
   nombre VARCHAR(100) NOT NULL,
   id_album INT REFERENCES Album(id)
);

CREATE TABLE Historial (
   id_usuario INT REFERENCES Usuario(id),
   id_cancion INT REFERENCES Cancion(id),
   fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

ERD y Consultas.

```sql
-- Listar canciones reproducidas por cada usuario
SELECT u.nombre AS usuario, c.nombre AS cancion, a.nombre AS album, ar.nombre AS artista
FROM Historial h
INNER JOIN Usuario u ON h.id_usuario = u.id
INNER JOIN Cancion c ON h.id_cancion = c.id
INNER JOIN Album a ON c.id_album = a.id
INNER JOIN Artista ar ON a.id_artista = ar.id;

-- Obtener el número total de reproducciones por canción
SELECT c.nombre AS cancion, COUNT(h.id_usuario) AS reproducciones
FROM Historial h
INNER JOIN Cancion c ON h.id_cancion = c.id
GROUP BY c.nombre
ORDER BY reproducciones DESC;

-- Mostrar los usuarios que han reproducido más de 3 canciones
```

Data Output | Messages | Notifications

	usuario character varying (100)	cancion character varying (100)	album character varying (100)	artista character varying (100)
1	Mario López	Gimme Shelter	Let It Bleed	The Rolling Stones
2	Paula Herrera	Shake It Off	1989	Taylor Swift
3	Rodrigo Torres	Just The Way You Are	Doo-Wops & Hooligans	Bruno Mars
4	Elena Martínez	Don't Start Now	Future Nostalgia	Dua Lipa
5	Andrés Orozco	Tití Me Preguntó	Un Verano Sin Ti	Bad Bunny

```sql
-- Obtener el número total de reproducciones por canción
SELECT c.nombre AS cancion, COUNT(h.id_usuario) AS reproducciones
FROM Historial h
INNER JOIN Cancion c ON h.id_cancion = c.id
GROUP BY c.nombre
ORDER BY reproducciones DESC;

-- Mostrar los usuarios que han reproducido más de 3 canciones
SELECT u.nombre, COUNT(h.id_cancion) AS total_reproducciones
FROM Historial h
INNER JOIN Usuario u ON h.id_usuario = u.id
GROUP BY u.nombre
HAVING COUNT(h.id_cancion) > 3;
```

Data Output | Messages | Notifications

	cancion character varying (100)	reproducciones bigint
1	Don't Start Now	1
2	Tití Me Preguntó	1
3	Shake It Off	1
4	Gimme Shelter	1
5	Just The Way You Are	1

```sql
SELECT u.nombre, COUNT(h.id_cancion) AS total_reproducciones
FROM Historial h
INNER JOIN Usuario u ON h.id_usuario = u.id
GROUP BY u.nombre
HAVING COUNT(h.id_cancion) > 3;
```

Data Output | Messages | Notifications

	nombre character varying (100)	total_reproducciones bigint
1	Mario López	4

4. Sistema de Control de Proyectos.

Codigo SQL y Entidades

. Proyecto (id, nombre, id_empleado)

. Empleado (id, nombre, puesto)

. Tarea (id, descripcion, id_proyecto, fecha_vencimiento, estado)

SQL.

```
CREATE TABLE Empleado (
   id SERIAL PRIMARY KEY,
   nombre VARCHAR(100) NOT NULL,
   puesto VARCHAR(100) NOT NULL
);

CREATE TABLE Proyecto (
   id SERIAL PRIMARY KEY,
   nombre VARCHAR(100) NOT NULL,
   id_empleado INT REFERENCES Empleado(id)
);

CREATE TABLE Tarea (
```

id SERIAL PRIMARY KEY,

descripcion TEXT NOT NULL,

id_proyecto INT REFERENCES Proyecto(id),

fecha_vencimiento DATE NOT NULL,

estado VARCHAR(20) CHECK (estado IN ('pendiente', 'en progreso', 'completada'))

);

ERD y Consultas.

```sql
-- Mostrar tareas pendientes ordenadas por fecha
SELECT t.descripcion, p.nombre AS proyecto, t.fecha_vencimiento
FROM Tarea t
INNER JOIN Proyecto p ON t.id_proyecto = p.id
WHERE t.estado = 'pendiente'
ORDER BY t.fecha_vencimiento ASC;

-- Contar cuántos empleados hay por cada puesto
SELECT e.puesto, COUNT(e.id) AS cantidad_empleados
FROM Empleado e
GROUP BY e.puesto;

-- Listar los proyectos asignados a "Lucía Fernández"
SELECT p.nombre AS proyecto
FROM Proyecto p
INNER JOIN Empleado e ON p.id_empleado = e.id
```

Data Output | Messages | Notifications

	descripcion text	proyecto character varying (100)	fecha_vencimiento date
1	Diseño de interfaz	Desarrollo de App	2024-07-01

```sql
ORDER BY c.fecha_vencimiento ASC;

-- Contar cuántos empleados hay por cada puesto
SELECT e.puesto, COUNT(e.id) AS cantidad_empleados
FROM Empleado e
GROUP BY e.puesto;

-- Listar los proyectos asignados a "Lucía Fernández"
SELECT p.nombre AS proyecto
FROM Proyecto p
INNER JOIN Empleado e ON p.id_empleado = e.id
```

Data Output | Messages | Notifications

	puesto character varying (100)	cantidad_empleados bigint
1	Diseñador UX	1
2	Desarrollador	1
3	Analista	1
4	Gerente	1
5	Tester	1

```sql
SELECT p.nombre AS proyecto
FROM Proyecto p
INNER JOIN Empleado e ON p.id_empleado = e.id
WHERE e.nombre = 'Isabel Castro';
```

Data Output | Messages | Notifications

	proyecto character varying (100)
1	Plataforma de E-learning

5.Sistema de Evaluacion Academica.

Codigos SQL y Entidades

Entidades:

- Estudiante (id, nombre, email)
- Curso (id, nombre)
- Profesor (id, nombre)
- Calificación (id_estudiante, id_curso, id_profesor, nota)

```
CREATE TABLE Estudiante (
  id SERIAL PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  email VARCHAR(100) UNIQUE NOT NULL
```

```
);

CREATE TABLE Curso (
   id SERIAL PRIMARY KEY,
   nombre VARCHAR(100) NOT NULL
);

CREATE TABLE Profesor (
   id SERIAL PRIMARY KEY,
   nombre VARCHAR(100) NOT NULL
);

CREATE TABLE Calificacion (
   id_estudiante INT REFERENCES Estudiante(id),
   id_curso INT REFERENCES Curso(id),
   id_profesor INT REFERENCES Profesor(id),
   nota DECIMAL(5,2) NOT NULL,
   PRIMARY KEY (id_estudiante, id_curso, id_profesor)
);
```

ERD y Consultas.

```sql
-- Obtener el promedio de calificaciones de cada estudiante
SELECT e.nombre, ROUND(AVG(c.nota), 2) AS promedio
FROM Calificacion c
INNER JOIN Estudiante e ON c.id_estudiante = e.id
GROUP BY e.nombre;

-- Listar los estudiantes con calificación mayor a 90
SELECT e.nombre, c.nota
FROM Calificacion c
INNER JOIN Estudiante e ON c.id_estudiante = e.id
WHERE c.nota > 90;

-- Mostrar la cantidad de estudiantes inscritos en cada curso
SELECT cu.nombre AS curso, COUNT(c.id_estudiante) AS total_estudiantes
FROM Calificacion c
INNER JOIN Curso cu ON c.id_curso = cu.id
```

Data Output | Messages | Notifications

	nombre character varying (100)	promedio numeric
1	Diego Castro	90.00
2	Jorge Morales	85.00
3	Valeria Contreras	92.00
4	Sebastián Vázquez	89.00
5	Gabriela Suárez	95.00

```sql
-- Listar los estudiantes con calificación mayor a 90
SELECT e.nombre, c.nota
FROM Calificacion c
INNER JOIN Estudiante e ON c.id_estudiante = e.id
WHERE c.nota > 90;

-- Mostrar la cantidad de estudiantes inscritos en cada curso
SELECT cu.nombre AS curso, COUNT(c.id_estudiante) AS total_estudiantes
FROM Calificacion c
INNER JOIN Curso cu ON c.id_curso = cu.id
GROUP BY cu.nombre
ORDER BY total_estudiantes DESC;
```

Data Output | Messages | Notifications

	nombre character varying (100)	nota numeric (5,2)
1	Valeria Contreras	92.00
2	Gabriela Suárez	95.00

```sql

-- Mostrar la cantidad de estudiantes inscritos en cada curso
SELECT cu.nombre AS curso, COUNT(c.id_estudiante) AS total_estudiantes
FROM Calificacion c
INNER JOIN Curso cu ON c.id_curso = cu.id
GROUP BY cu.nombre
ORDER BY total_estudiantes DESC;
```

Data Output | Messages | Notifications

	curso character varying (100)	total_estudiantes bigint
1	Álgebra Lineal	2
2	Historia Universal	2
3	Programación en Python	1