ANALÍTICA Y GESTIÓN ACADÉMICA

Fundamentos de base de datos

10 DE ABRIL DE 2025

DAMIAN ALEXANDER LUGO AGUILAR- 23490386
Prof. Jose Ramon Bogarin Valenzuela

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. Nuestros querys iniciales quedarian de la siguiente manera:

```sql
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
);
```

```sql
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');
```

Y estas son algunos ejemplos de consultas avanzadas (CLE)

```sql
-- 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;
```

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.

```
35  --agregar a la tabla estudiantes columna numero de telefono
36  ALTER TABLE estudiantes
37  ADD numero INT;
```

Tambien agrege numeros a cada estudiante para comprobar si funcionaba

```
39  UPDATE estudiantes
40  SET numero= 123458
41  WHERE nombre='Carla Ruiz'
```

Solo se necesitaba cambiar el nombre por el que querias agregar el numero

	id [PK] integer	nombre character varying (100)	email character varying (100)	fecha_nacimiento date	numero integer
1	1	Ana Torres	ana@example.com	1998-03-12	123456
2	3	Carla Ruiz	carla@example.com	1995-11-05	123458
3	2	Luis Gómez	luisgomez@universidad.edu	2000-07-22	123457

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

```
43  --Modifica la tabla cursos para que el nombre del curso no pueda repetirse.
44  ALTER TABLE cursos
45  ADD UNIQUE (nombre_curso);
```

Parte 2: Carga y Ajuste de Datos (LMD)

1. Actualiza el email de "Luis Gómez" a luisgomez@universidad.edu.

```
48  --Actualiza el email de "Luis Gómez" a luisgomez@universidad.edu.
49  UPDATE estudiantes
50  SET email='luisgomez@universidad.edu'
51  WHERE nombre='Luis Gómez';
```

3	2	Luis Gómez	luisgomez@universidad.edu	2000-07-22	123457

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

```
52  --Registra una nueva matrícula para "Carla Ruiz" en el curso “Bases de Datos”, con fecha 2025-04-01
53  INSERT INTO matriculas (id_estudiante, id_curso, fecha_matricula) VALUES
54  (3, 2, '2025-04-01');
```

3. Elimina la matrícula de "Ana Torres" del curso “Bases de Datos”.

```
56  --Elimina la matrícula de "Ana Torres" del curso “Bases de Datos”.
57  DELETE FROM matriculas
58  WHERE id_estudiante=1; --Su id es el 1
```

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.

```sql
SELECT e.nombre AS estudiante, c.nombre_curso AS curso, m.fecha_matricula
FROM estudiantes e INNER JOIN matriculas m
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	Carla Ruiz	Programación Web	2025-04-01

2. Muestra cuántos cursos ha tomado cada estudiante.

```sql
--Muestra cuántos cursos ha tomado cada estudiante.
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;
```

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

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

```sql
--Calcula la edad actual de cada estudiante y ordénalos de mayor a menor edad.
SELECT nombre, fecha_nacimiento, DATE_PART('year', AGE(fecha_nacimiento)) AS edad
FROM estudiantes
ORDER BY fecha_nacimiento ASC;
```

	nombre character varying (100)	fecha_nacimiento date	edad double precision
1	Carla Ruiz	1995-11-05	29
2	Ana Torres	1998-03-12	27
3	Luis Gómez	2000-07-22	24

4. Muestra qué curso tiene más estudiantes matriculados.

```sql
--Muestra qué curso tiene más 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
ORDER BY c.nombre_curso DESC;--Si ponemos ASC mostrara la que tiene menos estudiantes matriculados
```

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

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

```sql
--Calcula el porcentaje de estudiantes matriculados respecto al total de estudiantes para cada curso.
SELECT c.id, c.nombre_curso AS curso, m.id_estudiante AS estudiantes_inscritos,
ROUND((COUNT(DISTINCT m.id_estudiante)* 100.0)/(SELECT COUNT(*)FROM estudiantes), 2) AS porcentaje_inscritos
FROM cursos c LEFT JOIN matriculas m ON c.id=m.id_curso
GROUP BY c.id, c.nombre_curso, m,id_estudiante;
```

	id integer	curso character varying (100)	estudiantes_inscritos integer	porcentaje_inscritos numeric
1	1	Bases de Datos	2	33.33
2	2	Programación Web	3	33.33