

1. Inserción, actualización y eliminación de datos

Inserción:

```
SQL> INSERT INTO Estudiante VALUES (seq_estudiante.NEXTVAL, 'María', 'Gonzales', '75345678', 'maria.gonzales@email.com');
1 row created.

SQL> INSERT INTO Estudiante VALUES (seq_estudiante.NEXTVAL, 'Juan', 'Pérez', '75456789', 'juan.perez@email.com');
1 row created.

SQL> INSERT INTO Docente VALUES (seq_docente.NEXTVAL, 'Mg. Sofia Castro', 'Inteligencia Artificial', 'sofia.castro@email.com');
1 row created.

SQL> INSERT INTO Curso VALUES (seq_curso.NEXTVAL, 'Machine Learning', 14, 'Online', 202);
1 row created.

SQL> INSERT INTO Matricula VALUES (seq_matricula.NEXTVAL, SYSDATE, 3, 102);
1 row created.

SQL> INSERT INTO Matricula VALUES (seq_matricula.NEXTVAL, SYSDATE, 4, 102);
1 row created.
```

Actualización:

```
SQL> UPDATE Estudiante SET email = 'greia.lopez.actualizado@email.com' WHERE id_estudiante = 1;
1 row updated.

SQL> UPDATE Curso SET modalidad = 'Presencial' WHERE id_curso = 100;
1 row updated.

SQL> UPDATE Docente SET especialidad = 'Data Science' WHERE id_docente = 202;
1 row updated.
```

Eliminación de datos:

```
SQL> DELETE FROM Matricula WHERE id_estudiante = 4 AND id_curso = 102;
1 row deleted.

SQL> DELETE FROM Estudiante WHERE id_estudiante = 4;
1 row deleted.
```

2. Restricciones: PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, CHECK

```
SQL> SELECT * FROM Matricula WHERE id_estudiante NOT IN (SELECT id_estudiante FROM Estudiante);
no rows selected

SQL> SELECT * FROM Matricula WHERE id_curso NOT IN (SELECT id_curso FROM Curso);
no rows selected

SQL>
SQL> -- UNIQUE: Verificar DNI duplicados
SQL> SELECT dni, COUNT(*) FROM Estudiante GROUP BY dni HAVING COUNT(*) > 1;
no rows selected

SQL>
SQL> -- CHECK: Verificar modalidades válidas
SQL> SELECT * FROM Curso WHERE modalidad NOT IN ('Presencial', 'Online');
no rows selected

SQL>
SQL> -- NOT NULL: Verificar campos obligatorios
SQL> SELECT * FROM Estudiante WHERE nombre IS NULL OR apellido IS NULL;
no rows selected
```

3. Consultas SELECT avanzadas: joins, subconsultas, funciones agregadas

-Joins:

```
SQL> SELECT
  2   e.nombre || ' ' || e.apellido AS estudiante,
  3   c.nombre AS curso,
  4   c.duracion_semanas,
  5   c.modalidad,
  6   d.nombre AS docente,
  7   d.especialidad,
  8   m.fecha AS fecha_matricula
  9 FROM Matricula m
 10 INNER JOIN Estudiante e ON m.id_estudiante = e.id_estudiante
 11 INNER JOIN Curso c ON m.id_curso = c.id_curso
 12 INNER JOIN Docente d ON c.id_docente = d.id_docente
 13 ORDER BY m.fecha DESC;
```

ESTUDIANTE

CURSO

DURACION_SEMANAS MODALIDAD

DOCENTE

ESPECIALIDAD

FECHA_MA

María Gonzales

Machine Learning

14 Online

ESTUDIANTE

CURSO

DURACION_SEMANAS MODALIDAD

DOCENTE

ESPECIALIDAD

FECHA_MA

Mg. Sofía Castro

Data Science

15/11/25

ESTUDIANTE

CURSO

DURACION_SEMANAS MODALIDAD

DOCENTE

ESPECIALIDAD

FECHA_MA

Grecia López

Programación en Python

12 Presencial

ESTUDIANTE

CURSO

DURACION_SEMANAS MODALIDAD

DOCENTE

ESPECIALIDAD

FECHA_MA

Dr. Ana Ruiz

Desarrollo

15/11/25

ESTUDIANTE

CURSO

DURACION_SEMANAS MODALIDAD

DOCENTE

ESPECIALIDAD

FECHA_MA

Carlos Mendoza

Bases de Datos

10 Presencial

ESTUDIANTE

CURSO

DURACION_SEMANAS MODALIDAD

DOCENTE

ESPECIALIDAD

FECHA_MA

Ing. Luis Torres

Bases de Datos

15/11/25

-Subconsultas:

```
SQL> SELECT e.nombre, e.apellido, e.email
2 FROM Estudiante e
3 WHERE e.id_estudiante IN (
4     SELECT m.id_estudiante
5     FROM Matricula m
6     GROUP BY m.id_estudiante
7     HAVING COUNT(*) > 1
8 );

no rows selected

SQL>
SQL> -- Cursos con más estudiantes matriculados
SQL> SELECT c.nombre AS curso,
2     (SELECT COUNT(*) FROM Matricula m WHERE m.id_curso = c.id_curso) AS total_estudiantes
3 FROM Curso c
4 ORDER BY total_estudiantes DESC;

CURSO
-----
TOTAL_ESTUDIANTES
-----
Programación en Python
1

Machine Learning
1

Bases de Datos
1
```

-Funciones Agregadas:

Estadísticas generales

```
SQL> SELECT
2     COUNT(DISTINCT e.id_estudiante) AS total_estudiantes,
3     COUNT(DISTINCT c.id_curso) AS total_cursos,
4     COUNT(DISTINCT d.id_docente) AS total_docentes,
5     COUNT(*) AS total_matriculas,
6     AVG(c.duracion_semanas) AS duracion_promedio_cursos
7 FROM Estudiante e, Curso c, Docente d, Matricula m;

TOTAL_ESTUDIANTES TOTAL_CURSOS TOTAL_DOCENTES TOTAL_MATRICULAS
-----
DURACION_PROMEDIO_CURSOS
-----
3 3 3 81
12
```

Estudiantes por curso

```
SQL> SELECT
2     c.nombre AS curso,
3     COUNT(m.id_estudiante) AS cantidad_estudiantes,
4     c.modalidad
5 FROM Curso c
6 LEFT JOIN Matricula m ON c.id_curso = m.id_curso
7 GROUP BY c.id_curso, c.nombre, c.modalidad
8 ORDER BY cantidad_estudiantes DESC;

CURSO
-----
CANTIDAD_ESTUDIANTES MODALIDAD
-----
Programación en Python
1 Presencial

Machine Learning
1 Online

Bases de Datos
1 Presencial
```

4. Índices y transacciones

-Índices:

```
SQL> -- Índice para búsquedas por email
SQL> CREATE INDEX idx_estudiante_email ON Estudiante(email);

Index created.

SQL>
SQL> -- Índice compuesto para búsquedas frecuentes
SQL> CREATE INDEX idx_curso_modalidad_duracion ON Curso(modalidad, duracion_semanas);

Index created.

SQL>
SQL> -- Índice para consultas de matrículas por fecha
SQL> CREATE INDEX idx_matricula_fecha ON Matricula(fecha);

Index created.
```

-Transacciones:

```
SQL> DECLARE
2     v_estudiante_id NUMBER := 3;
3     v_curso1_id NUMBER := 100;
4     v_curso2_id NUMBER := 101;
5 BEGIN
6     -- Iniciar transacción
7     SAVEPOINT inicio_matricula;
8
9     -- Insertar primera matrícula
10    INSERT INTO Matricula VALUES (seq_matricula.NEXTVAL, SYSDATE, v_estudiante_id, v_curso1_id);
11
12    -- Insertar segunda matrícula
13    INSERT INTO Matricula VALUES (seq_matricula.NEXTVAL, SYSDATE, v_estudiante_id, v_curso2_id);
14
15    -- Confirmar transacción
16    COMMIT;
17    DBMS_OUTPUT.PUT_LINE('Matrículas realizadas exitosamente');
18
19 EXCEPTION
20     WHEN OTHERS THEN
21         -- Revertir en caso de error
22         ROLLBACK TO inicio_matricula;
23         DBMS_OUTPUT.PUT_LINE('Error en la transacción: ' || SQLERRM);
24 END;
25 /

PL/SQL procedure successfully completed.
```