Departamento de
Tecnología Digital

Tecsup TECNOLOGÍA CON SENTIDO

PROYECTO FINAL

Base de Datos de
Sistema Administrativo
Universitario

DOCENTE:

Suárez García, Gonzalo

CURSO:

Base de Datos
2 - C24 - Secciones A-B

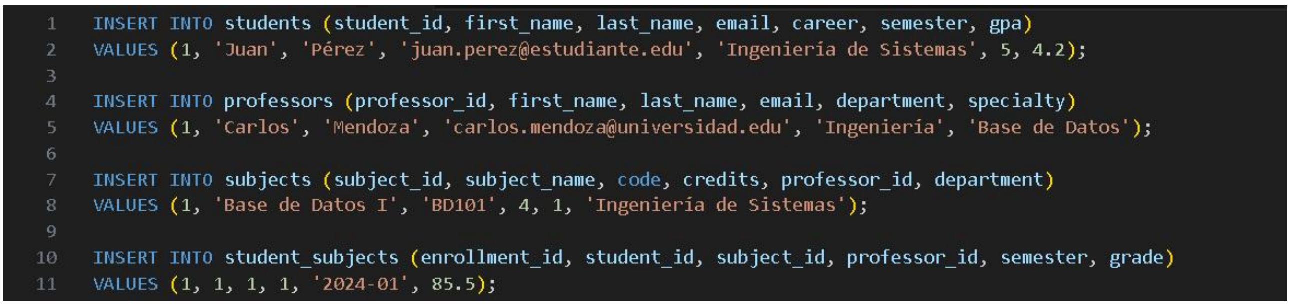
Integrantes :

- Suarez Lavado, Pedro Junior
- Chuquirima Alanya, Augusto Benjamin
- Prieto Huiza, Alexis Stephano

Entrega

1) Inserción, actualización y eliminación de datos

INSERT:

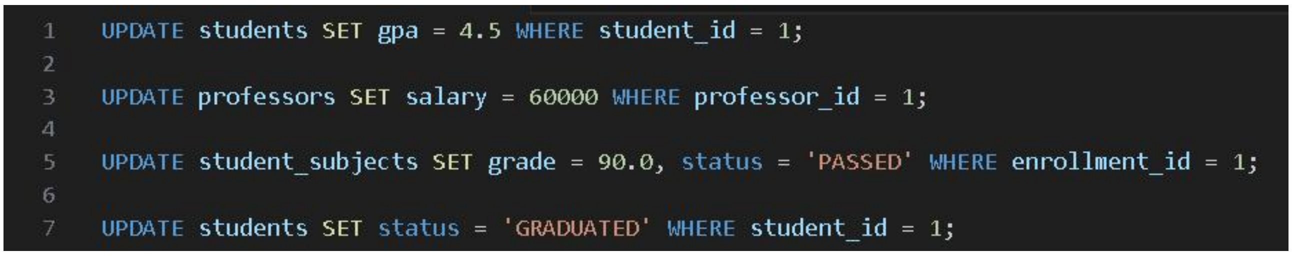


```sql
INSERT INTO students (student_id, first_name, last_name, email, career, semester, gpa)
VALUES (1, 'Juan', 'Pérez', 'juan.perez@estudiante.edu', 'Ingeniería de Sistemas', 5, 4.2);

INSERT INTO professors (professor_id, first_name, last_name, email, department, specialty)
VALUES (1, 'Carlos', 'Mendoza', 'carlos.mendoza@universidad.edu', 'Ingeniería', 'Base de Datos');

INSERT INTO subjects (subject_id, subject_name, code, credits, professor_id, department)
VALUES (1, 'Base de Datos I', 'BD101', 4, 1, 'Ingeniería de Sistemas');

INSERT INTO student_subjects (enrollment_id, student_id, subject_id, professor_id, semester, grade)
VALUES (1, 1, 1, 1, '2024-01', 85.5);
```

SCRIPT INSERT:

```sql
INSERT INTO students (student_id, first_name, last_name, email, career, semester, gpa)
VALUES (1, 'Juan', 'Pérez', 'juan.perez@estudiante.edu', 'Ingeniería de Sistemas', 5, 4.2);

INSERT INTO professors (professor_id, first_name, last_name, email, department, specialty)
VALUES (1, 'Carlos', 'Mendoza', 'carlos.mendoza@universidad.edu', 'Ingeniería', 'Base de Datos');

INSERT INTO subjects (subject_id, subject_name, code, credits, professor_id, department)
VALUES (1, 'Base de Datos I', 'BD101', 4, 1, 'Ingeniería de Sistemas');

INSERT INTO student_subjects (enrollment_id, student_id, subject_id, professor_id, semester, grade)
VALUES (1, 1, 1, 1, '2024-01', 85.5);
```

UPDATE:

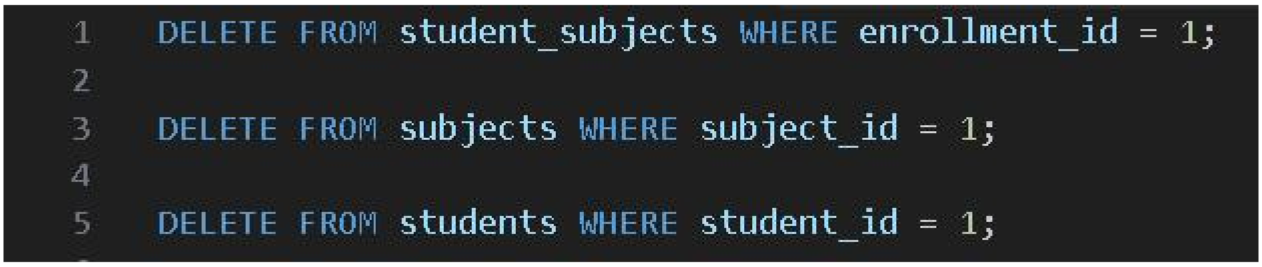


```sql
UPDATE students SET gpa = 4.5 WHERE student_id = 1;

UPDATE professors SET salary = 60000 WHERE professor_id = 1;

UPDATE student_subjects SET grade = 90.0, status = 'PASSED' WHERE enrollment_id = 1;

UPDATE students SET status = 'GRADUATED' WHERE student_id = 1;
```

SCRIPT UPDATE:

```sql
UPDATE students SET gpa = 4.5 WHERE student_id = 1;

UPDATE professors SET salary = 60000 WHERE professor_id = 1;

UPDATE student_subjects SET grade = 90.0, status = 'PASSED' WHERE enrollment_id = 1;

UPDATE students SET status = 'GRADUATED' WHERE student_id = 1;
```

DELETE:



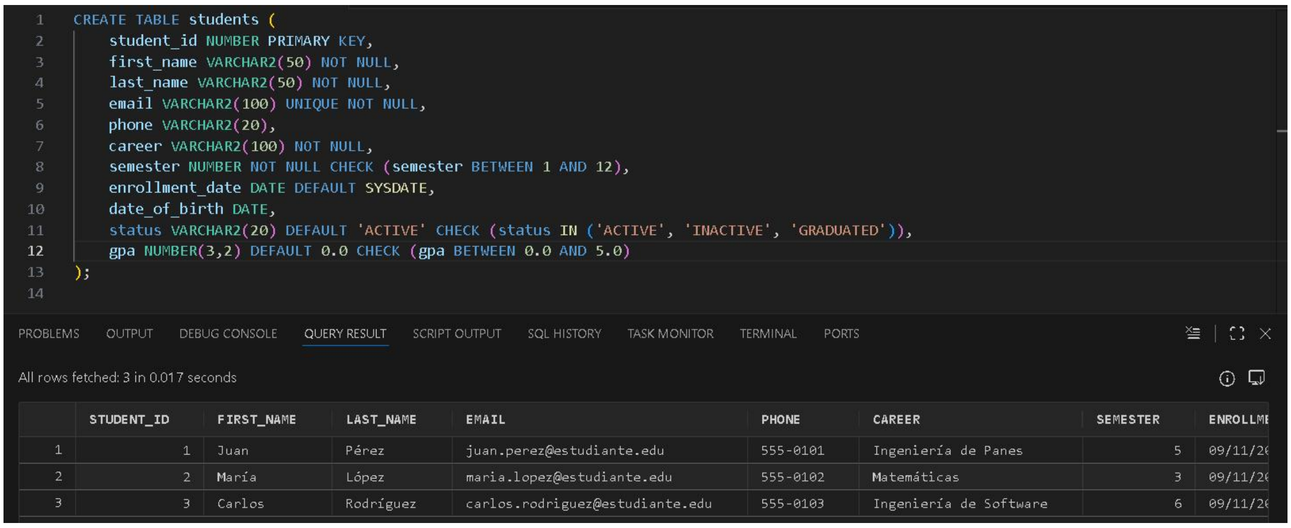
```
DELETE FROM student_subjects WHERE enrollment_id = 1;

DELETE FROM subjects WHERE subject_id = 1;

DELETE FROM students WHERE student_id = 1;
```

SCRIPT DELETE:

```
DELETE FROM student_subjects WHERE enrollment_id = 1;

DELETE FROM subjects WHERE subject_id = 1;

DELETE FROM students WHERE student_id = 1;
```

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

Tabla Students:



```
CREATE TABLE students (
    student_id NUMBER PRIMARY KEY,
    first_name VARCHAR2(50) NOT NULL,
    last_name VARCHAR2(50) NOT NULL,
    email VARCHAR2(100) UNIQUE NOT NULL,
    phone VARCHAR2(20),
    career VARCHAR2(100) NOT NULL,
    semester NUMBER NOT NULL CHECK (semester BETWEEN 1 AND 12),
    enrollment_date DATE DEFAULT SYSDATE,
    date_of_birth DATE,
    status VARCHAR2(20) DEFAULT 'ACTIVE' CHECK (status IN ('ACTIVE', 'INACTIVE', 'GRADUATED')),
    gpa NUMBER(3,2) DEFAULT 0.0 CHECK (gpa BETWEEN 0.0 AND 5.0)
);
```

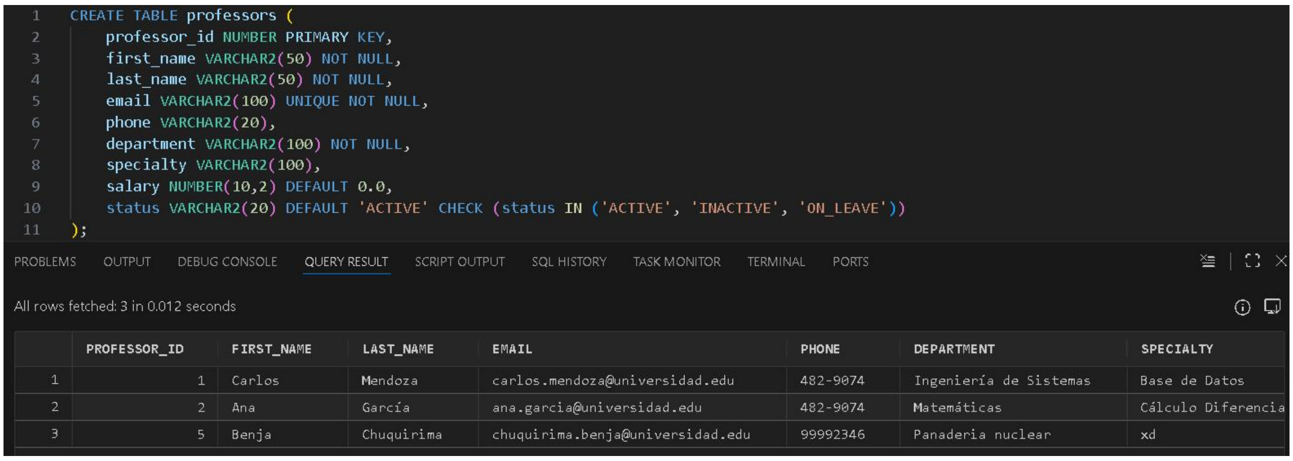
All rows fetched: 3 in 0.017 seconds

	STUDENT_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE	CAREER	SEMESTER	ENROLLME
1	1	Juan	Pérez	juan.perez@estudiante.edu	555-0101	Ingeniería de Panes	5	09/11/2
2	2	María	López	maria.lopez@estudiante.edu	555-0102	Matemáticas	3	09/11/2
3	3	Carlos	Rodríguez	carlos.rodriguez@estudiante.edu	555-0103	Ingeniería de Software	6	09/11/2

Script Tabla Students:

```
CREATE TABLE students (
    student_id NUMBER PRIMARY KEY,
    first_name VARCHAR2(50) NOT NULL,
    last_name VARCHAR2(50) NOT NULL,
    email VARCHAR2(100) UNIQUE NOT NULL,
    phone VARCHAR2(20),
    career VARCHAR2(100) NOT NULL,
    semester NUMBER NOT NULL CHECK (semester BETWEEN 1 AND 12),
    enrollment_date DATE DEFAULT SYSDATE,
    date_of_birth DATE,
    status VARCHAR2(20) DEFAULT 'ACTIVE' CHECK (status IN ('ACTIVE', 'INACTIVE', 'GRADUATED')),
    gpa NUMBER(3,2) DEFAULT 0.0 CHECK (gpa BETWEEN 0.0 AND 5.0)
);
```

Tabla Professors:



```
CREATE TABLE professors (
    professor_id NUMBER PRIMARY KEY,
    first_name VARCHAR2(50) NOT NULL,
    last_name VARCHAR2(50) NOT NULL,
    email VARCHAR2(100) UNIQUE NOT NULL,
    phone VARCHAR2(20),
    department VARCHAR2(100) NOT NULL,
    specialty VARCHAR2(100),
    salary NUMBER(10,2) DEFAULT 0.0,
    status VARCHAR2(20) DEFAULT 'ACTIVE' CHECK (status IN ('ACTIVE', 'INACTIVE', 'ON_LEAVE'))
);
```

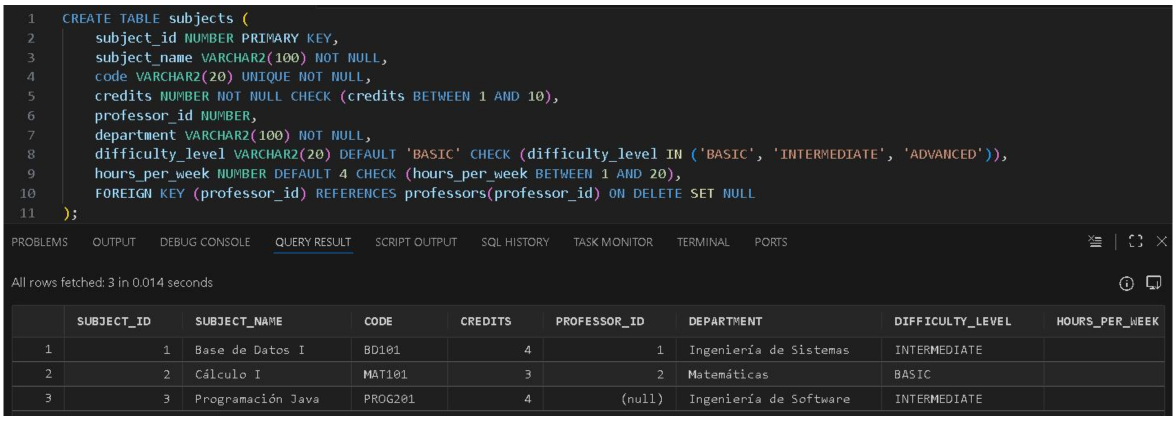
All rows fetched: 3 in 0.012 seconds

	PROFESSOR_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE	DEPARTMENT	SPECIALTY
1	1	Carlos	Mendoza	carlos.mendoza@universidad.edu	482-9074	Ingeniería de Sistemas	Base de Datos
2	2	Ana	García	ana.garcia@universidad.edu	482-9074	Matemáticas	Cálculo Diferencia
3	5	Benja	Chuquirima	chuquirima.benja@universidad.edu	99992346	Panaderia nuclear	xd

Script Tabla Professors:

```
CREATE TABLE professors (
    professor_id NUMBER PRIMARY KEY,
    first_name VARCHAR2(50) NOT NULL,
    last_name VARCHAR2(50) NOT NULL,
    email VARCHAR2(100) UNIQUE NOT NULL,
    phone VARCHAR2(20),
    department VARCHAR2(100) NOT NULL,
    specialty VARCHAR2(100),
    salary NUMBER(10,2) DEFAULT 0.0,
    status VARCHAR2(20) DEFAULT 'ACTIVE' CHECK (status IN ('ACTIVE', 'INACTIVE', 'ON_LEAVE'))
);
```

Tabla Subjects:



```
CREATE TABLE subjects (
    subject_id NUMBER PRIMARY KEY,
    subject_name VARCHAR2(100) NOT NULL,
    code VARCHAR2(20) UNIQUE NOT NULL,
    credits NUMBER NOT NULL CHECK (credits BETWEEN 1 AND 10),
    professor_id NUMBER,
    department VARCHAR2(100) NOT NULL,
    difficulty_level VARCHAR2(20) DEFAULT 'BASIC' CHECK (difficulty_level IN ('BASIC', 'INTERMEDIATE', 'ADVANCED')),
    hours_per_week NUMBER DEFAULT 4 CHECK (hours_per_week BETWEEN 1 AND 20),
    FOREIGN KEY (professor_id) REFERENCES professors(professor_id) ON DELETE SET NULL
);
```

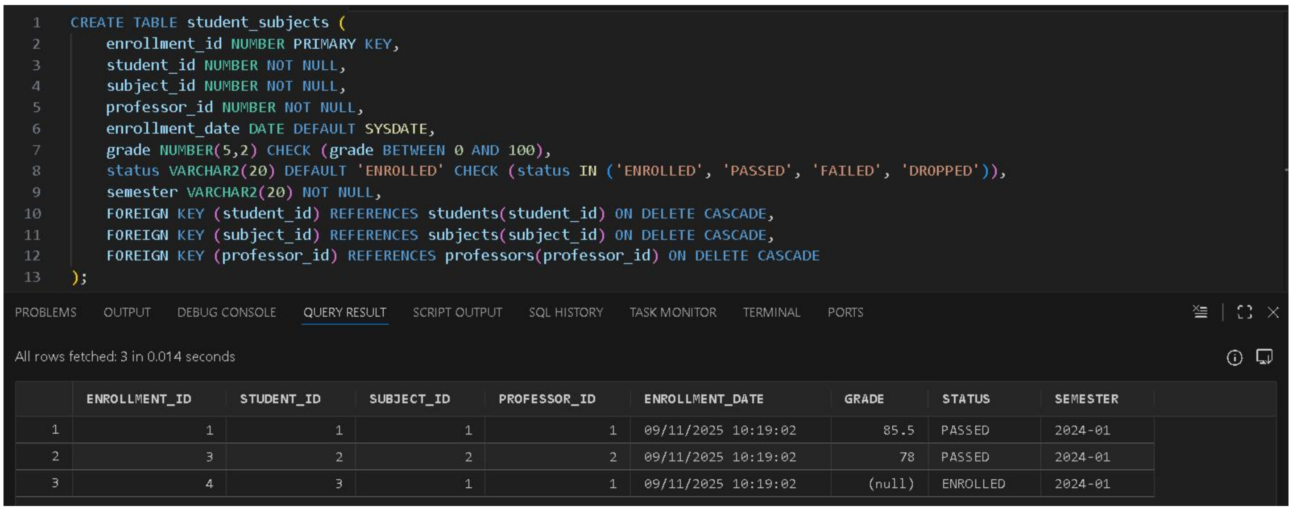
All rows fetched: 3 in 0.014 seconds

	SUBJECT_ID	SUBJECT_NAME	CODE	CREDITS	PROFESSOR_ID	DEPARTMENT	DIFFICULTY_LEVEL	HOURS_PER_WEEK
1	1	Base de Datos I	BD101	4	1	Ingeniería de Sistemas	INTERMEDIATE	
2	2	Cálculo I	MAT101	3	2	Matemáticas	BASIC	
3	3	Programación Java	PROG201	4	(null)	Ingeniería de Software	INTERMEDIATE	

Script Tabla Subjects:

```sql
CREATE TABLE subjects (
    subject_id NUMBER PRIMARY KEY,
    subject_name VARCHAR2(100) NOT NULL,
    code VARCHAR2(20) UNIQUE NOT NULL,
    credits NUMBER NOT NULL CHECK (credits BETWEEN 1 AND 10),
    professor_id NUMBER,
    department VARCHAR2(100) NOT NULL,
    difficulty_level VARCHAR2(20) DEFAULT 'BASIC' CHECK (difficulty_level IN ('BASIC', 'INTERMEDIATE',
'ADVANCED')),
    hours_per_week NUMBER DEFAULT 4 CHECK (hours_per_week BETWEEN 1 AND 20),
    FOREIGN KEY (professor_id) REFERENCES professors(professor_id) ON DELETE SET NULL
);
```

Tabla Student_Subjects:



```sql
CREATE TABLE student_subjects (
    enrollment_id NUMBER PRIMARY KEY,
    student_id NUMBER NOT NULL,
    subject_id NUMBER NOT NULL,
    professor_id NUMBER NOT NULL,
    enrollment_date DATE DEFAULT SYSDATE,
    grade NUMBER(5,2) CHECK (grade BETWEEN 0 AND 100),
    status VARCHAR2(20) DEFAULT 'ENROLLED' CHECK (status IN ('ENROLLED', 'PASSED', 'FAILED', 'DROPPED')),
    semester VARCHAR2(20) NOT NULL,
    FOREIGN KEY (student_id) REFERENCES students(student_id) ON DELETE CASCADE,
    FOREIGN KEY (subject_id) REFERENCES subjects(subject_id) ON DELETE CASCADE,
    FOREIGN KEY (professor_id) REFERENCES professors(professor_id) ON DELETE CASCADE
);
```

All rows fetched: 3 in 0.014 seconds

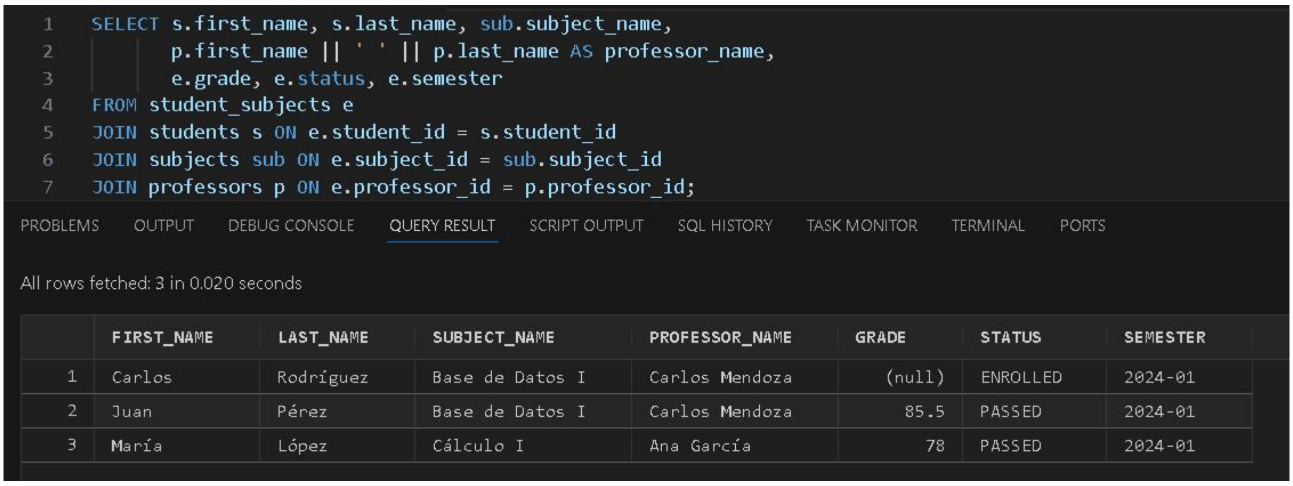
	ENROLLMENT_ID	STUDENT_ID	SUBJECT_ID	PROFESSOR_ID	ENROLLMENT_DATE	GRADE	STATUS	SEMESTER
1	1	1	1	1	09/11/2025 10:19:02	85.5	PASSED	2024-01
2	3	2	2	2	09/11/2025 10:19:02	78	PASSED	2024-01
3	4	3	1	1	09/11/2025 10:19:02	(null)	ENROLLED	2024-01

Script Tabla Student_Subjects:

```sql
CREATE TABLE student_subjects (
    enrollment_id NUMBER PRIMARY KEY,
    student_id NUMBER NOT NULL,
    subject_id NUMBER NOT NULL,
    professor_id NUMBER NOT NULL,
    enrollment_date DATE DEFAULT SYSDATE,
    grade NUMBER(5,2) CHECK (grade BETWEEN 0 AND 100),
    status VARCHAR2(20) DEFAULT 'ENROLLED' CHECK (status IN ('ENROLLED', 'PASSED', 'FAILED', 'DROPPED')),
    semester VARCHAR2(20) NOT NULL,
    FOREIGN KEY (student_id) REFERENCES students(student_id) ON DELETE CASCADE,
    FOREIGN KEY (subject_id) REFERENCES subjects(subject_id) ON DELETE CASCADE,
    FOREIGN KEY (professor_id) REFERENCES professors(professor_id) ON DELETE CASCADE
);
```

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

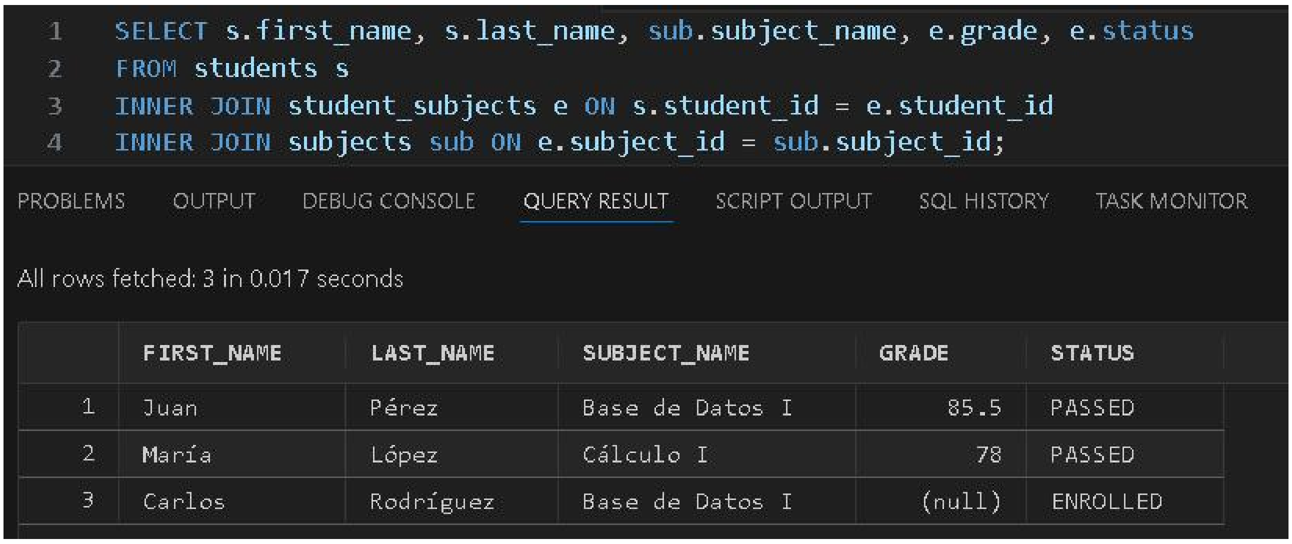
Consultas Joins:



```sql
SELECT s.first_name, s.last_name, sub.subject_name,
       p.first_name || ' ' || p.last_name AS professor_name,
       e.grade, e.status, e.semester
FROM student_subjects e
JOIN students s ON e.student_id = s.student_id
JOIN subjects sub ON e.subject_id = sub.subject_id
JOIN professors p ON e.professor_id = p.professor_id;
```

All rows fetched: 3 in 0.020 seconds

	FIRST_NAME	LAST_NAME	SUBJECT_NAME	PROFESSOR_NAME	GRADE	STATUS	SEMESTER
1	Carlos	Rodríguez	Base de Datos I	Carlos Mendoza	(null)	ENROLLED	2024-01
2	Juan	Pérez	Base de Datos I	Carlos Mendoza	85.5	PASSED	2024-01
3	María	López	Cálculo I	Ana García	78	PASSED	2024-01



```sql
SELECT s.first_name, s.last_name, sub.subject_name, e.grade, e.status
FROM students s
INNER JOIN student_subjects e ON s.student_id = e.student_id
INNER JOIN subjects sub ON e.subject_id = sub.subject_id;
```

All rows fetched: 3 in 0.017 seconds

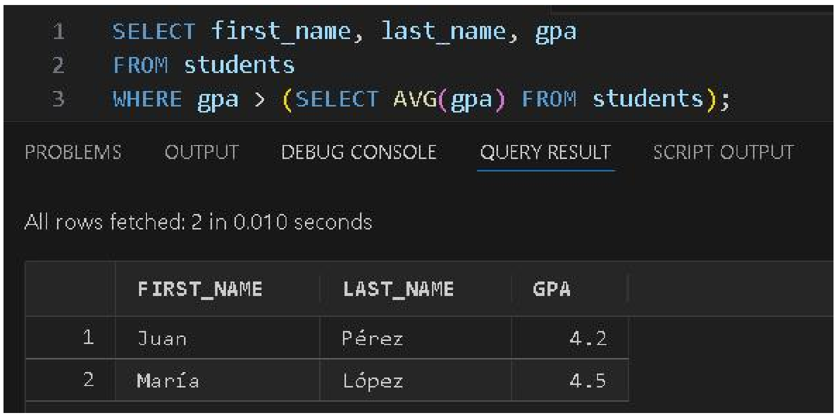
	FIRST_NAME	LAST_NAME	SUBJECT_NAME	GRADE	STATUS
1	Juan	Pérez	Base de Datos I	85.5	PASSED
2	María	López	Cálculo I	78	PASSED
3	Carlos	Rodríguez	Base de Datos I	(null)	ENROLLED

Script Consultas Joins:

```sql
SELECT s.first_name, s.last_name, sub.subject_name, e.grade, e.status
FROM students s
INNER JOIN student_subjects e ON s.student_id = e.student_id
INNER JOIN subjects sub ON e.subject_id = sub.subject_id;
```

```sql
SELECT s.first_name, s.last_name, sub.subject_name,
       p.first_name || ' ' || p.last_name AS professor_name,
       e.grade, e.status, e.semester
FROM student_subjects e
JOIN students s ON e.student_id = s.student_id
JOIN subjects sub ON e.subject_id = sub.subject_id
JOIN professors p ON e.professor_id = p.professor_id;
```

Subconsultas:

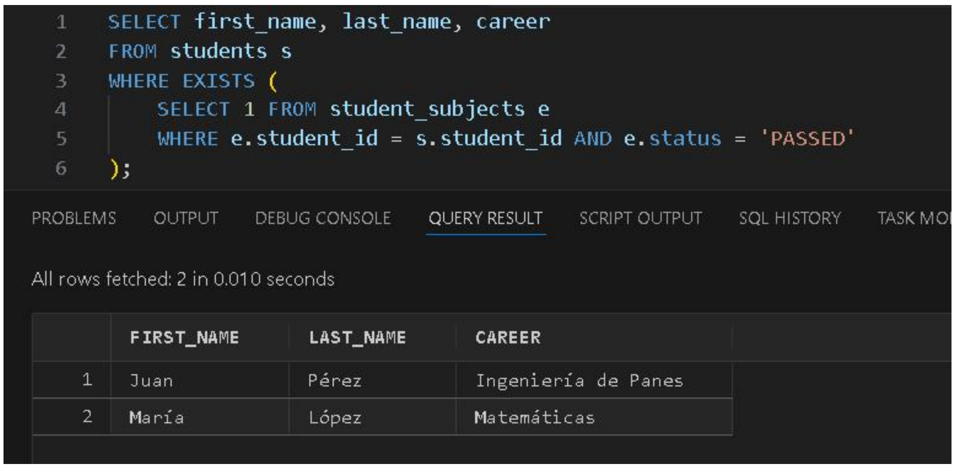


```sql
SELECT first_name, last_name, gpa
FROM students
WHERE gpa > (SELECT AVG(gpa) FROM students);
```

PROBLEMS OUTPUT DEBUG CONSOLE QUERY RESULT SCRIPT OUTPUT

All rows fetched: 2 in 0.010 seconds

	FIRST_NAME	LAST_NAME	GPA
1	Juan	Pérez	4.2
2	María	López	4.5



```sql
SELECT first_name, last_name, career
FROM students s
WHERE EXISTS (
    SELECT 1 FROM student_subjects e
    WHERE e.student_id = s.student_id AND e.status = 'PASSED'
);
```

PROBLEMS OUTPUT DEBUG CONSOLE QUERY RESULT SCRIPT OUTPUT SQL HISTORY TASK MOI

All rows fetched: 2 in 0.010 seconds

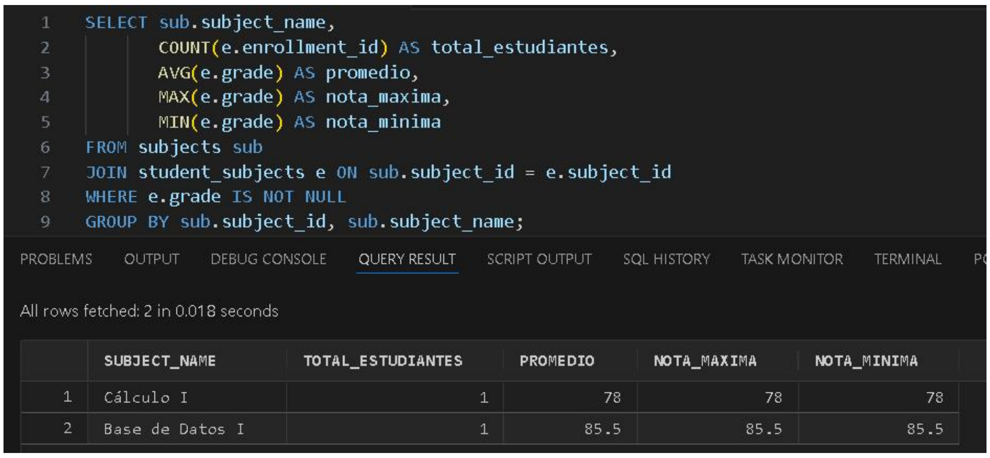
	FIRST_NAME	LAST_NAME	CAREER
1	Juan	Pérez	Ingeniería de Panes
2	María	López	Matemáticas

Script Subconsultas:

```sql
SELECT first_name, last_name, gpa
FROM students
WHERE gpa > (SELECT AVG(gpa) FROM students);
```

```sql
SELECT first_name, last_name, career
FROM students s
WHERE EXISTS (
    SELECT 1 FROM student_subjects e
    WHERE e.student_id = s.student_id AND e.status = 'PASSED'
);
```

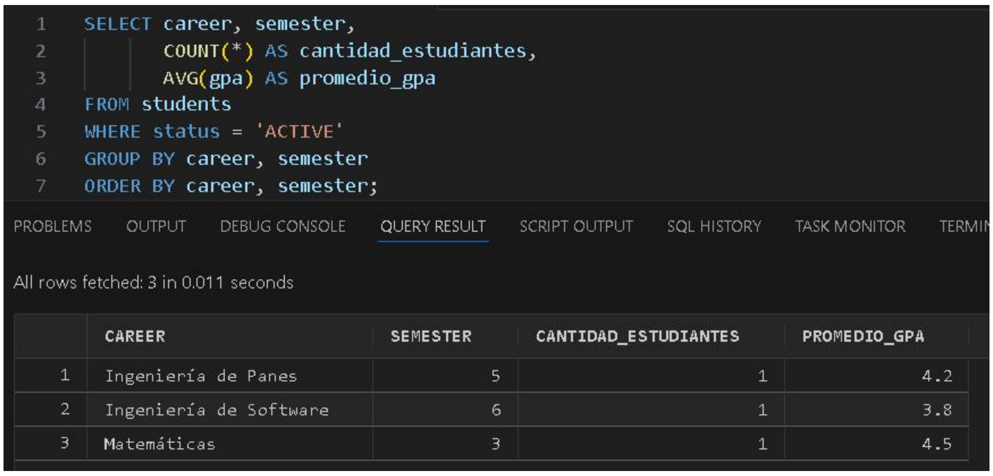
Funciones agregadas:



```
SELECT sub.subject_name,
       COUNT(e.enrollment_id) AS total_estudiantes,
       AVG(e.grade) AS promedio,
       MAX(e.grade) AS nota_maxima,
       MIN(e.grade) AS nota_minima
FROM subjects sub
JOIN student_subjects e ON sub.subject_id = e.subject_id
WHERE e.grade IS NOT NULL
GROUP BY sub.subject_id, sub.subject_name;
```

All rows fetched: 2 in 0.018 seconds

	SUBJECT_NAME	TOTAL_ESTUDIANTES	PROMEDIO	NOTA_MAXIMA	NOTA_MINIMA
1	Cálculo I	1	78	78	78
2	Base de Datos I	1	85.5	85.5	85.5



```
SELECT career, semester,
       COUNT(*) AS cantidad_estudiantes,
       AVG(gpa) AS promedio_gpa
FROM students
WHERE status = 'ACTIVE'
GROUP BY career, semester
ORDER BY career, semester;
```

All rows fetched: 3 in 0.011 seconds

	CAREER	SEMESTER	CANTIDAD_ESTUDIANTES	PROMEDIO_GPA
1	Ingeniería de Panes	5	1	4.2
2	Ingeniería de Software	6	1	3.8
3	Matemáticas	3	1	4.5

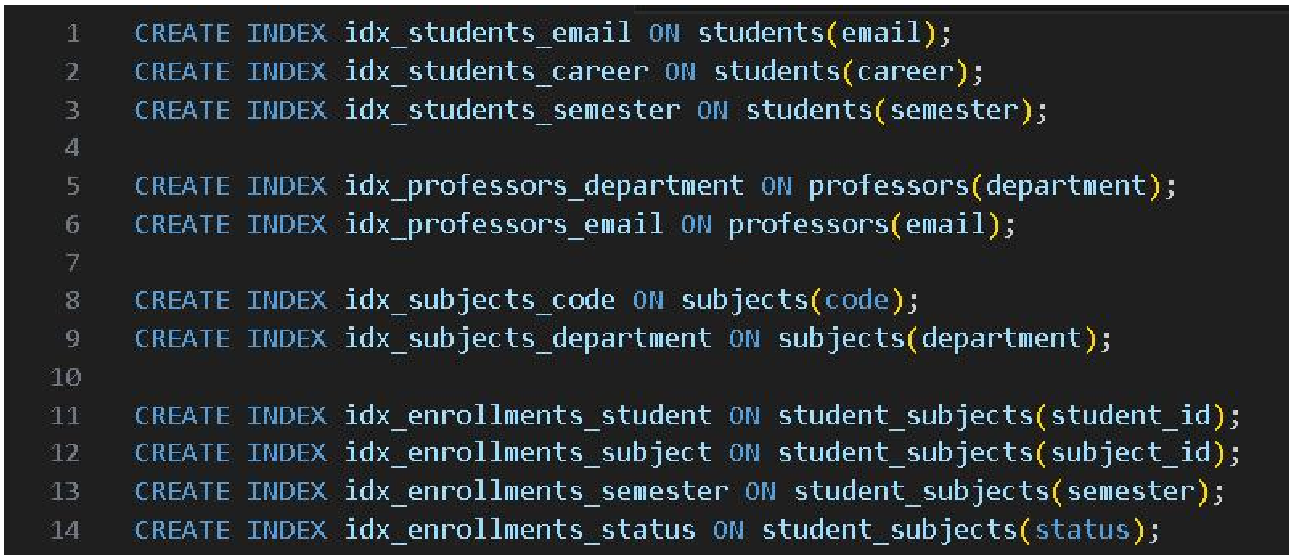
Script Funciones agregadas:

```
SELECT sub.subject_name,
       COUNT(e.enrollment_id) AS total_estudiantes,
       AVG(e.grade) AS promedio,
       MAX(e.grade) AS nota_maxima,
       MIN(e.grade) AS nota_minima
FROM subjects sub
JOIN student_subjects e ON sub.subject_id = e.subject_id
WHERE e.grade IS NOT NULL
GROUP BY sub.subject_id, sub.subject_name;
```

```
SELECT career, semester,
       COUNT(*) AS cantidad_estudiantes,
       AVG(gpa) AS promedio_gpa
FROM students
WHERE status = 'ACTIVE'
GROUP BY career, semester
ORDER BY career, semester;
```

4) Índices y transaccionesreación de tablas

Indices:

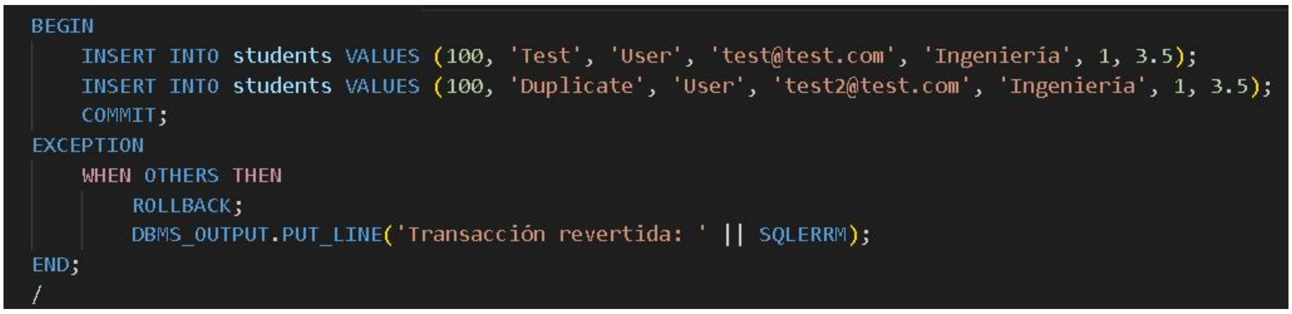


```
CREATE INDEX idx_students_email ON students(email);
CREATE INDEX idx_students_career ON students(career);
CREATE INDEX idx_students_semester ON students(semester);

CREATE INDEX idx_professors_department ON professors(department);
CREATE INDEX idx_professors_email ON professors(email);

CREATE INDEX idx_subjects_code ON subjects(code);
CREATE INDEX idx_subjects_department ON subjects(department);

CREATE INDEX idx_enrollments_student ON student_subjects(student_id);
CREATE INDEX idx_enrollments_subject ON student_subjects(subject_id);
CREATE INDEX idx_enrollments_semester ON student_subjects(semester);
CREATE INDEX idx_enrollments_status ON student_subjects(status);
```

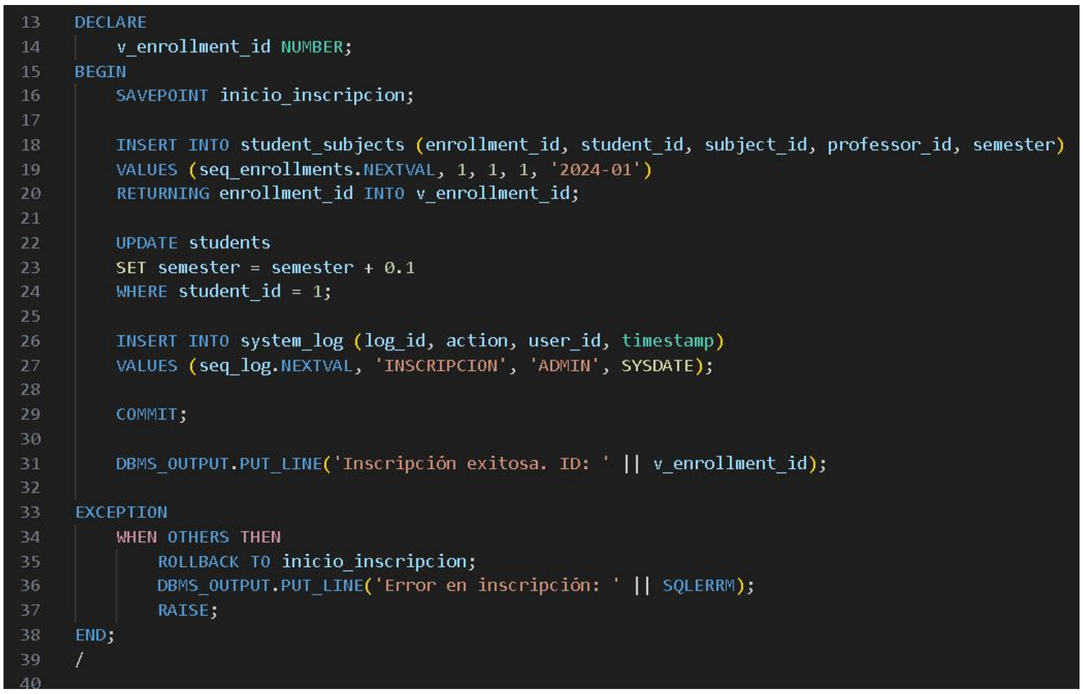
Script Indices:

```
CREATE INDEX idx_students_email ON students(email);
CREATE INDEX idx_students_career ON students(career);
CREATE INDEX idx_students_semester ON students(semester);

CREATE INDEX idx_professors_department ON professors(department);
CREATE INDEX idx_professors_email ON professors(email);

CREATE INDEX idx_subjects_code ON subjects(code);
CREATE INDEX idx_subjects_department ON subjects(department);

CREATE INDEX idx_enrollments_student ON student_subjects(student_id);
CREATE INDEX idx_enrollments_subject ON student_subjects(subject_id);
CREATE INDEX idx_enrollments_semester ON student_subjects(semester);
CREATE INDEX idx_enrollments_status ON student_subjects(status);
```

Transacciones:



```
BEGIN
    INSERT INTO students VALUES (100, 'Test', 'User', 'test@test.com', 'Ingeniería', 1, 3.5);
    INSERT INTO students VALUES (100, 'Duplicate', 'User', 'test2@test.com', 'Ingeniería', 1, 3.5);
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Transacción revertida: ' || SQLERRM);
END;
/
```



```
DECLARE
    v_enrollment_id NUMBER;
BEGIN
    SAVEPOINT inicio_inscripcion;

    INSERT INTO student_subjects (enrollment_id, student_id, subject_id, professor_id, semester)
    VALUES (seq_enrollments.NEXTVAL, 1, 1, 1, '2024-01')
    RETURNING enrollment_id INTO v_enrollment_id;

    UPDATE students
    SET semester = semester + 0.1
    WHERE student_id = 1;

    INSERT INTO system_log (log_id, action, user_id, timestamp)
    VALUES (seq_log.NEXTVAL, 'INSCRIPCION', 'ADMIN', SYSDATE);

    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Inscripción exitosa. ID: ' || v_enrollment_id);

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK TO inicio_inscripcion;
        DBMS_OUTPUT.PUT_LINE('Error en inscripción: ' || SQLERRM);
        RAISE;
END;
/
```

Script Transacciones:

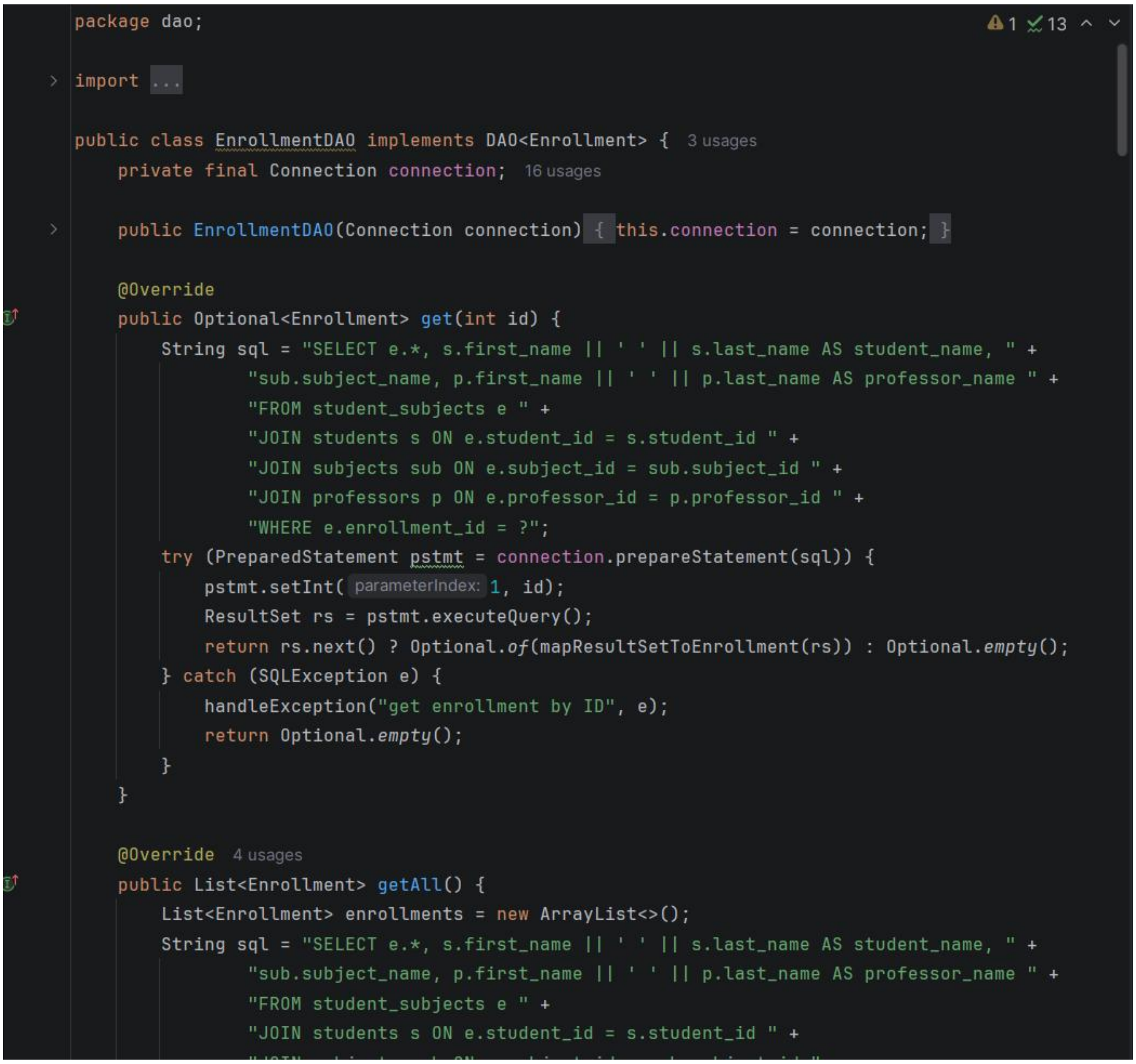
```
BEGIN
    INSERT INTO students VALUES (100, 'Test', 'User', 'test@test.com', 'Ingeniería', 1, 3.5);
    INSERT INTO students VALUES (100, 'Duplicate', 'User', 'test2@test.com', 'Ingeniería', 1, 3.5);
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Transacción revertida: ' || SQLERRM);
END;
/
```

```
DECLARE
    v_enrollment_id NUMBER;
BEGIN
    SAVEPOINT inicio_inscripcion;
    INSERT INTO student_subjects (enrollment_id, student_id, subject_id, professor_id, semester)
    VALUES (seq_enrollments.NEXTVAL, 1, 1, 1, '2024-01')
    RETURNING enrollment_id INTO v_enrollment_id;

    UPDATE students
    SET semester = semester + 0.1
    WHERE student_id = 1;
    INSERT INTO system_log (log_id, action, user_id, timestamp)
    VALUES (seq_log.NEXTVAL, 'INSCRIPCION', 'ADMIN', SYSDATE);
    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Inscripción exitosa. ID: ' || v_enrollment_id);

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK TO inicio_inscripcion;
        DBMS_OUTPUT.PUT_LINE('Error en inscripción: ' || SQLERRM);
        RAISE;
END;
/
```

5) Operaciones CRUD desde la aplicación

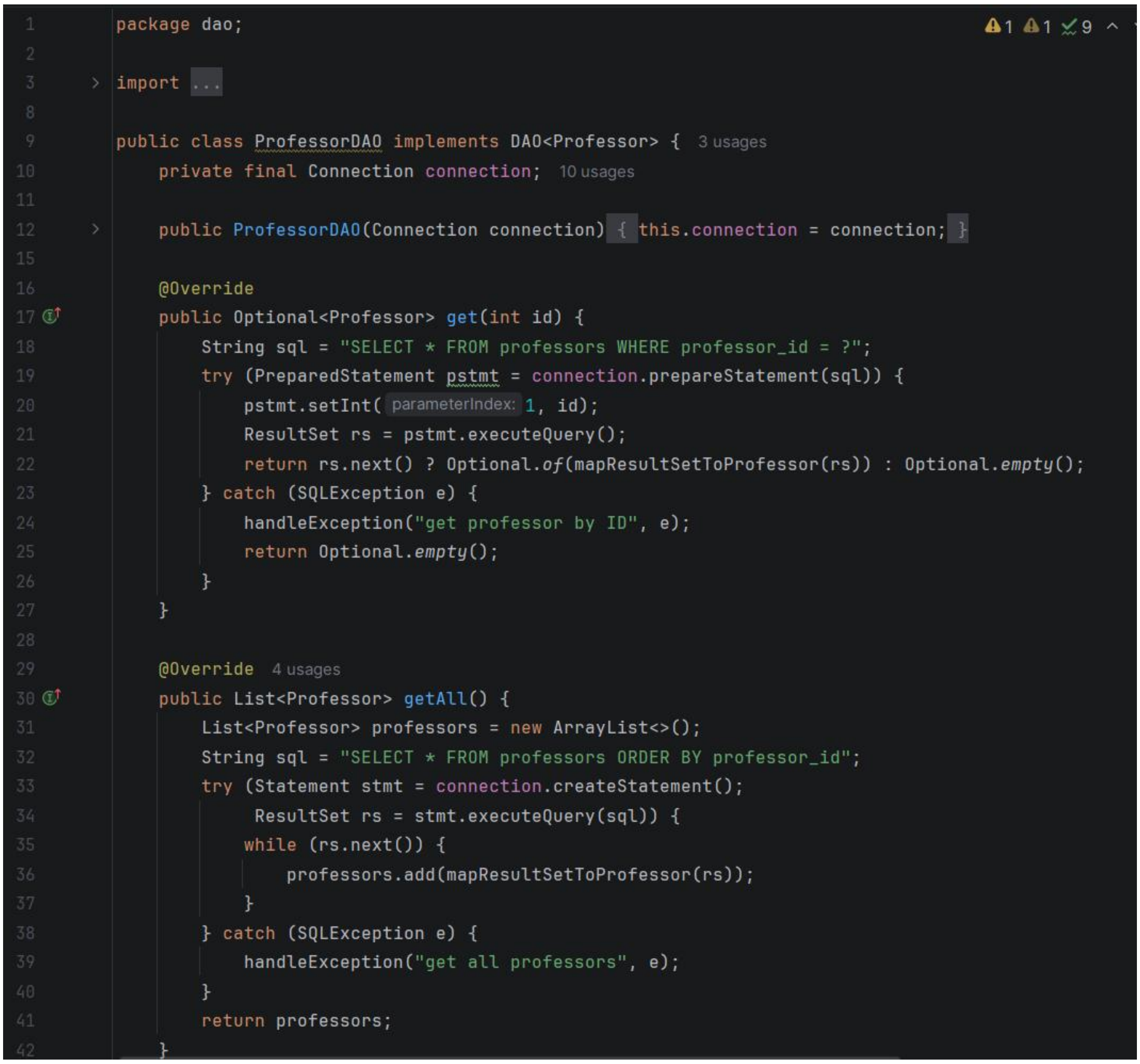
En mi aplicacion todos las operaciones Crud estaran en todo apartado con nombre DAO.

EnrollmentDAO:



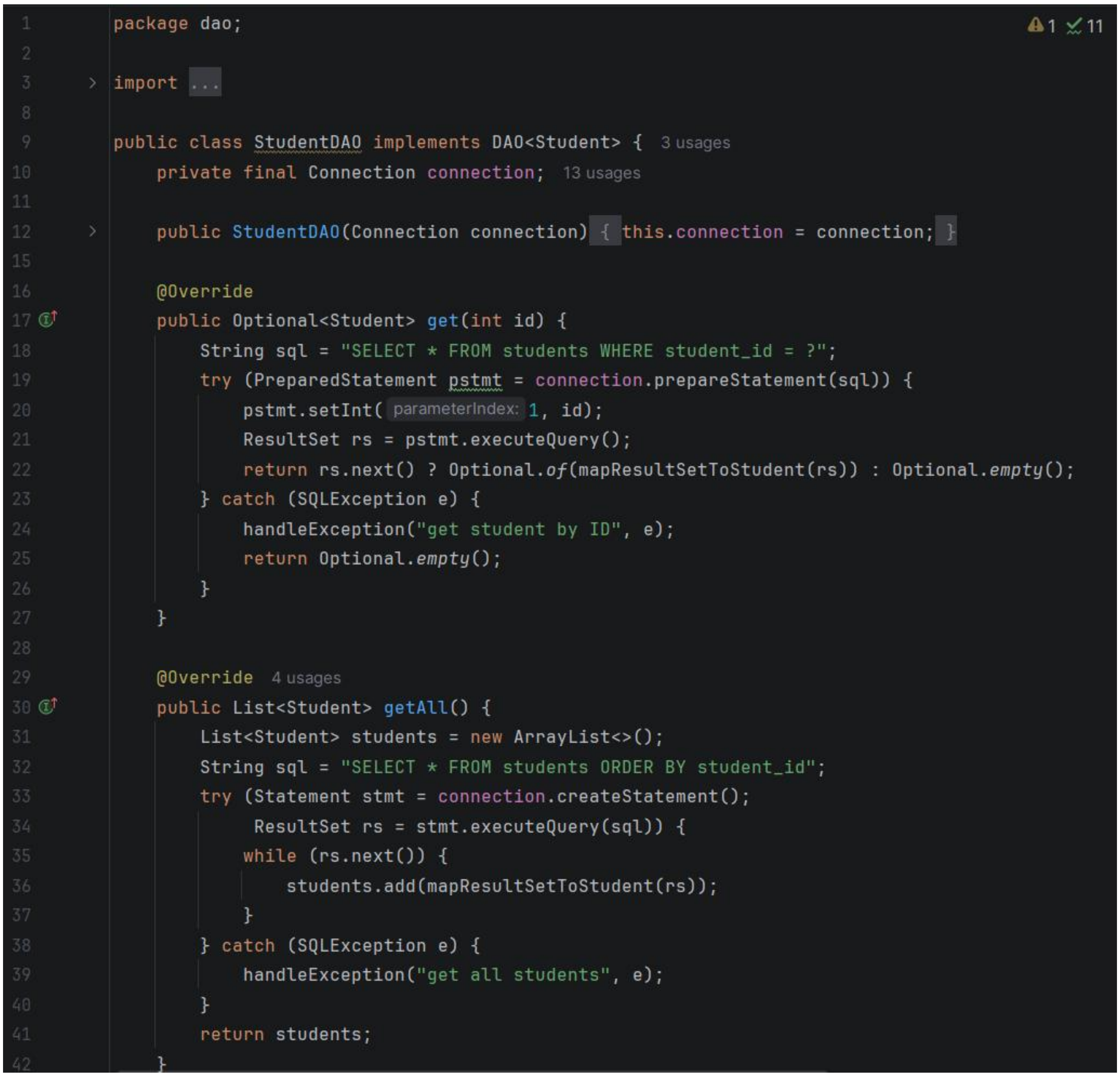
```java
package dao;

import ...

public class EnrollmentDAO implements DAO<Enrollment> {  3 usages
    private final Connection connection;  16 usages

    public EnrollmentDAO(Connection connection) { this.connection = connection; }

    @Override
    public Optional<Enrollment> get(int id) {
        String sql = "SELECT e.*, s.first_name || ' ' || s.last_name AS student_name, " +
                "sub.subject_name, p.first_name || ' ' || p.last_name AS professor_name " +
                "FROM student_subjects e " +
                "JOIN students s ON e.student_id = s.student_id " +
                "JOIN subjects sub ON e.subject_id = sub.subject_id " +
                "JOIN professors p ON e.professor_id = p.professor_id " +
                "WHERE e.enrollment_id = ?";
        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
            pstmt.setInt( parameterIndex: 1, id);
            ResultSet rs = pstmt.executeQuery();
            return rs.next() ? Optional.of(mapResultSetToEnrollment(rs)) : Optional.empty();
        } catch (SQLException e) {
            handleException("get enrollment by ID", e);
            return Optional.empty();
        }
    }

    @Override  4 usages
    public List<Enrollment> getAll() {
        List<Enrollment> enrollments = new ArrayList<>();
        String sql = "SELECT e.*, s.first_name || ' ' || s.last_name AS student_name, " +
                "sub.subject_name, p.first_name || ' ' || p.last_name AS professor_name " +
                "FROM student_subjects e " +
                "JOIN students s ON e.student_id = s.student_id " +
```

ProfessorDAO:



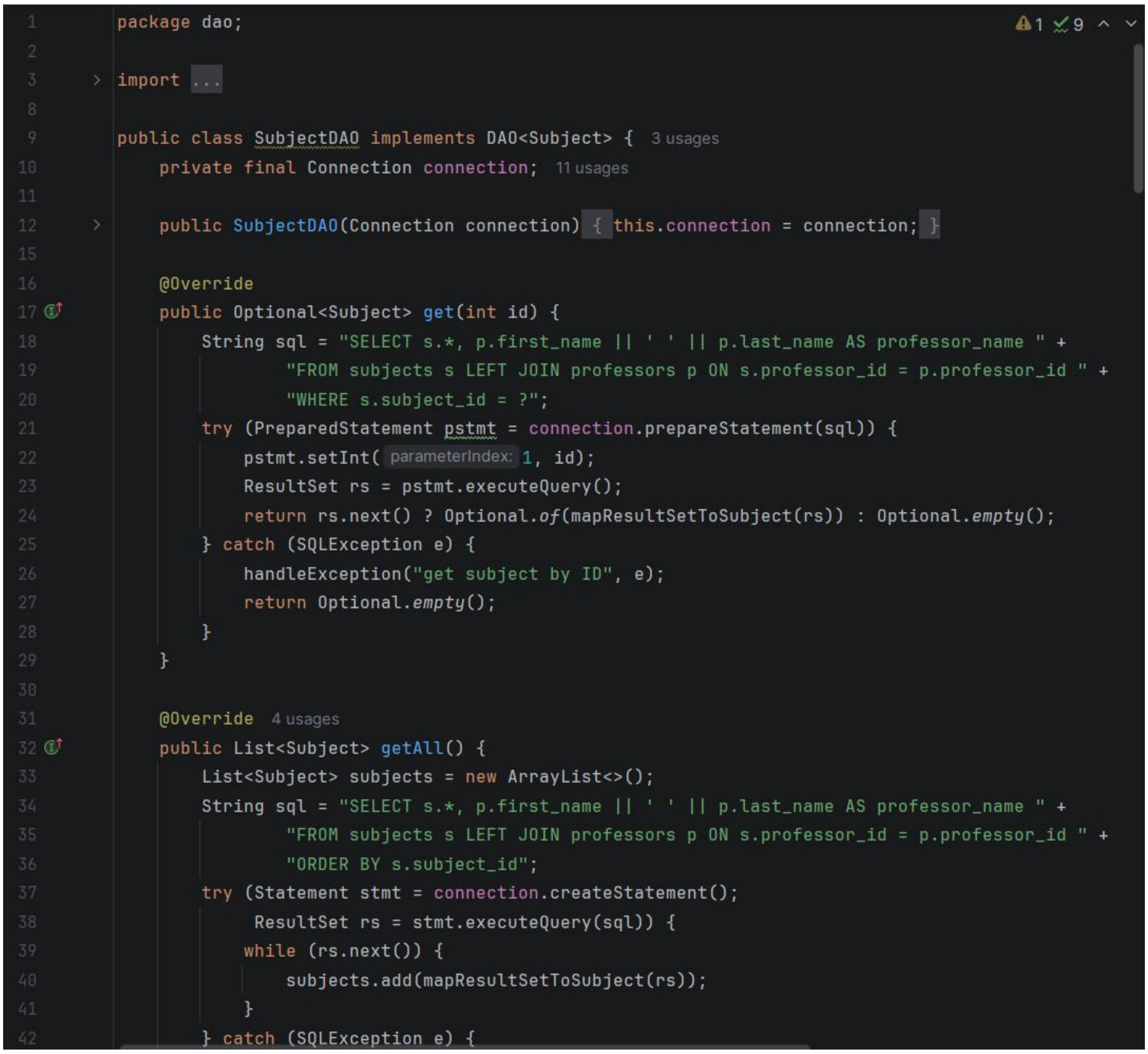
```java
package dao;

import ...

public class ProfessorDAO implements DAO<Professor> {  3 usages
    private final Connection connection;  10 usages

    public ProfessorDAO(Connection connection) { this.connection = connection; }

    @Override
    public Optional<Professor> get(int id) {
        String sql = "SELECT * FROM professors WHERE professor_id = ?";
        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
            pstmt.setInt( parameterIndex: 1, id);
            ResultSet rs = pstmt.executeQuery();
            return rs.next() ? Optional.of(mapResultSetToProfessor(rs)) : Optional.empty();
        } catch (SQLException e) {
            handleException("get professor by ID", e);
            return Optional.empty();
        }
    }

    @Override  4 usages
    public List<Professor> getAll() {
        List<Professor> professors = new ArrayList<>();
        String sql = "SELECT * FROM professors ORDER BY professor_id";
        try (Statement stmt = connection.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                professors.add(mapResultSetToProfessor(rs));
            }
        } catch (SQLException e) {
            handleException("get all professors", e);
        }
        return professors;
    }
```

StudentDAO:



```
package dao;

import ...

public class StudentDAO implements DAO<Student> {  3 usages
    private final Connection connection;  13 usages

    public StudentDAO(Connection connection) { this.connection = connection; }

    @Override
    public Optional<Student> get(int id) {
        String sql = "SELECT * FROM students WHERE student_id = ?";
        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
            pstmt.setInt( parameterIndex: 1, id);
            ResultSet rs = pstmt.executeQuery();
            return rs.next() ? Optional.of(mapResultSetToStudent(rs)) : Optional.empty();
        } catch (SQLException e) {
            handleException("get student by ID", e);
            return Optional.empty();
        }
    }

    @Override  4 usages
    public List<Student> getAll() {
        List<Student> students = new ArrayList<>();
        String sql = "SELECT * FROM students ORDER BY student_id";
        try (Statement stmt = connection.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                students.add(mapResultSetToStudent(rs));
            }
        } catch (SQLException e) {
            handleException("get all students", e);
        }
        return students;
    }
```

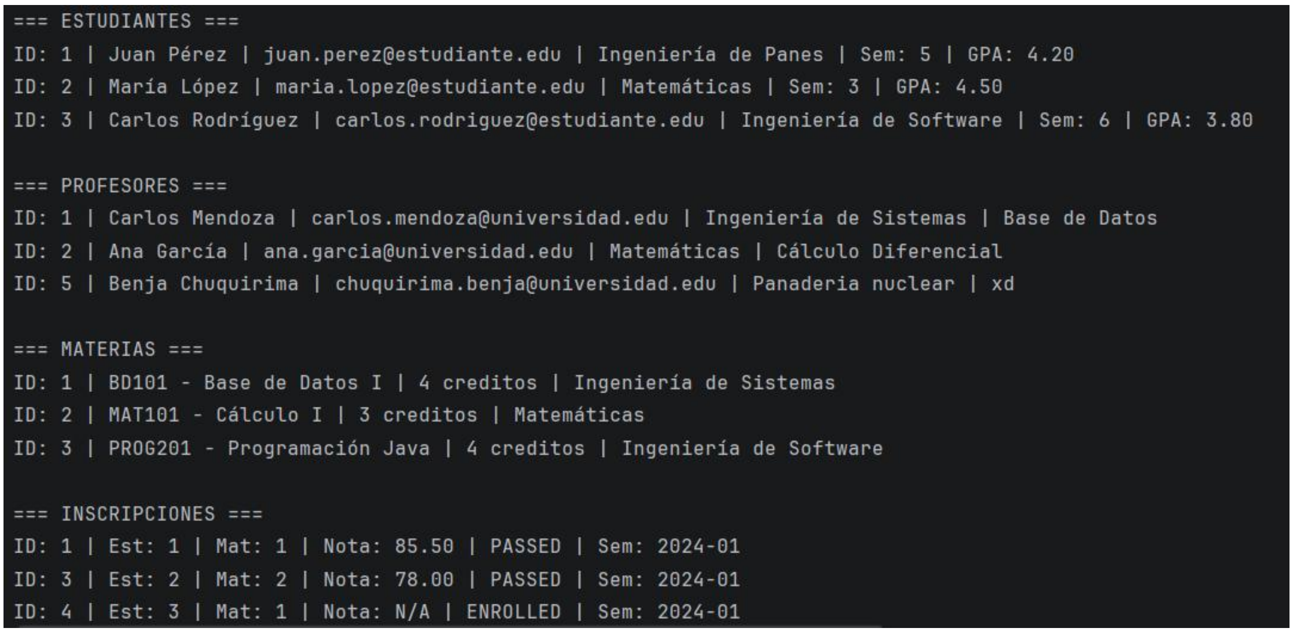
SubjectDAO:



```
package dao;

import ...

public class SubjectDAO implements DAO<Subject> {  3 usages
    private final Connection connection;  11 usages

    public SubjectDAO(Connection connection) { this.connection = connection; }

    @Override
    public Optional<Subject> get(int id) {
        String sql = "SELECT s.*, p.first_name || ' ' || p.last_name AS professor_name " +
                "FROM subjects s LEFT JOIN professors p ON s.professor_id = p.professor_id " +
                "WHERE s.subject_id = ?";
        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
            pstmt.setInt( parameterIndex: 1, id);
            ResultSet rs = pstmt.executeQuery();
            return rs.next() ? Optional.of(mapResultSetToSubject(rs)) : Optional.empty();
        } catch (SQLException e) {
            handleException("get subject by ID", e);
            return Optional.empty();
        }
    }

    @Override  4 usages
    public List<Subject> getAll() {
        List<Subject> subjects = new ArrayList<>();
        String sql = "SELECT s.*, p.first_name || ' ' || p.last_name AS professor_name " +
                "FROM subjects s LEFT JOIN professors p ON s.professor_id = p.professor_id " +
                "ORDER BY s.subject_id";
        try (Statement stmt = connection.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                subjects.add(mapResultSetToSubject(rs));
            }
        } catch (SQLException e) {
```

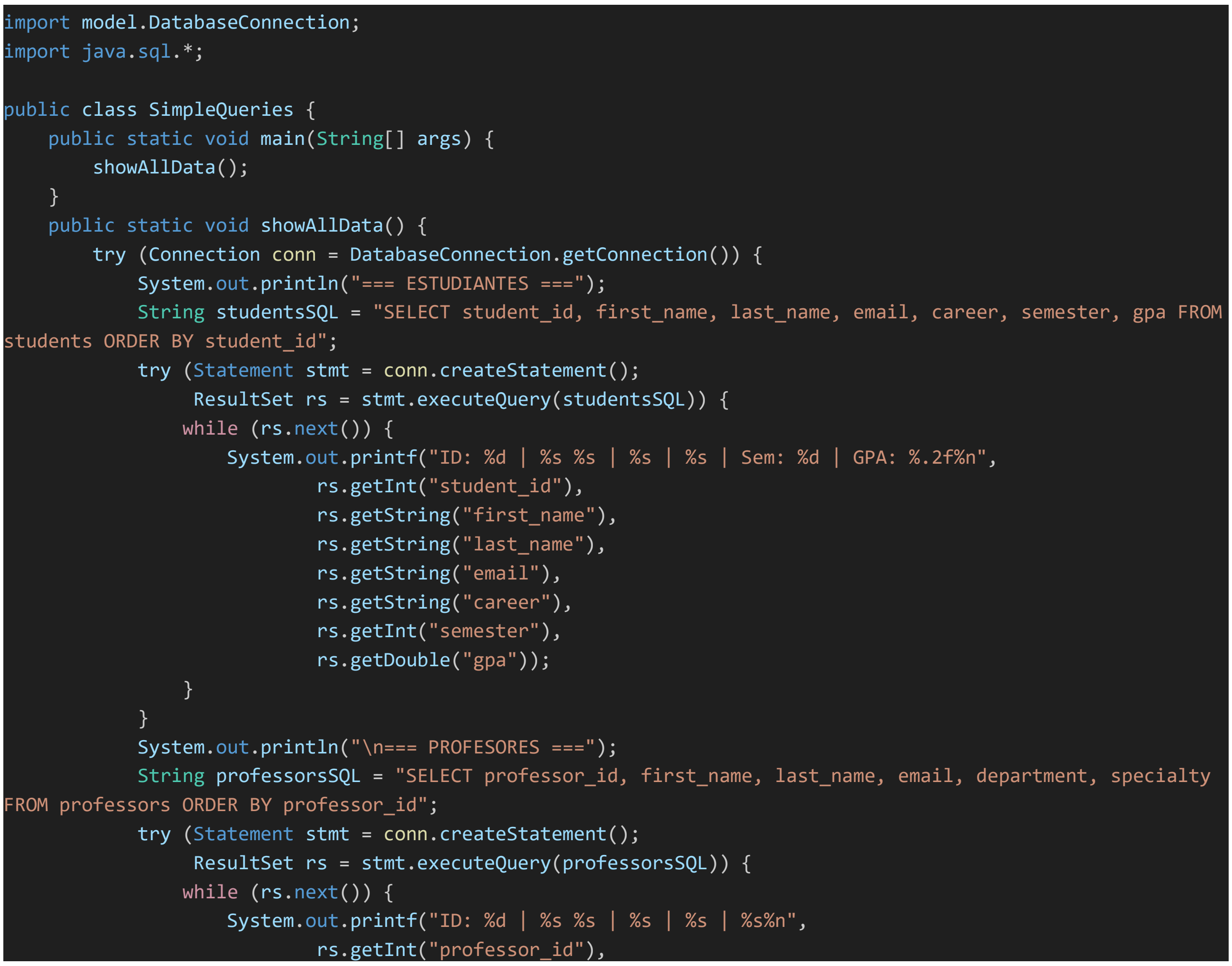
6) consulta simple para mostrar datos

Consulta Simple:



```
=== ESTUDIANTES ===
ID: 1 | Juan Pérez | juan.perez@estudiante.edu | Ingeniería de Panes | Sem: 5 | GPA: 4.20
ID: 2 | María López | maria.lopez@estudiante.edu | Matemáticas | Sem: 3 | GPA: 4.50
ID: 3 | Carlos Rodríguez | carlos.rodriguez@estudiante.edu | Ingeniería de Software | Sem: 6 | GPA: 3.80

=== PROFESORES ===
ID: 1 | Carlos Mendoza | carlos.mendoza@universidad.edu | Ingeniería de Sistemas | Base de Datos
ID: 2 | Ana García | ana.garcia@universidad.edu | Matemáticas | Cálculo Diferencial
ID: 5 | Benja Chuquirima | chuquirima.benja@universidad.edu | Panaderia nuclear | xd

=== MATERIAS ===
ID: 1 | BD101 - Base de Datos I | 4 creditos | Ingeniería de Sistemas
ID: 2 | MAT101 - Cálculo I | 3 creditos | Matemáticas
ID: 3 | PROG201 - Programación Java | 4 creditos | Ingeniería de Software

=== INSCRIPCIONES ===
ID: 1 | Est: 1 | Mat: 1 | Nota: 85.50 | PASSED | Sem: 2024-01
ID: 3 | Est: 2 | Mat: 2 | Nota: 78.00 | PASSED | Sem: 2024-01
ID: 4 | Est: 3 | Mat: 1 | Nota: N/A | ENROLLED | Sem: 2024-01
```

Script Consulta Simple:



```java
import model.DatabaseConnection;
import java.sql.*;

public class SimpleQueries {
    public static void main(String[] args) {
        showAllData();
    }
    public static void showAllData() {
        try (Connection conn = DatabaseConnection.getConnection()) {
            System.out.println("=== ESTUDIANTES ===");
            String studentsSQL = "SELECT student_id, first_name, last_name, email, career, semester, gpa FROM
students ORDER BY student_id";
            try (Statement stmt = conn.createStatement();
                 ResultSet rs = stmt.executeQuery(studentsSQL)) {
                while (rs.next()) {
                    System.out.printf("ID: %d | %s %s | %s | %s | Sem: %d | GPA: %.2f%n",
                            rs.getInt("student_id"),
                            rs.getString("first_name"),
                            rs.getString("last_name"),
                            rs.getString("email"),
                            rs.getString("career"),
                            rs.getInt("semester"),
                            rs.getDouble("gpa"));
                }
            }
            System.out.println("\n=== PROFESORES ===");
            String professorsSQL = "SELECT professor_id, first_name, last_name, email, department, specialty
FROM professors ORDER BY professor_id";
            try (Statement stmt = conn.createStatement();
                 ResultSet rs = stmt.executeQuery(professorsSQL)) {
                while (rs.next()) {
                    System.out.printf("ID: %d | %s %s | %s | %s | %s%n",
                            rs.getInt("professor_id"),
                            rs.getString("first_name"),
                            rs.getString("last_name"),
                            rs.getString("email"),
                            rs.getString("department"),
                            rs.getString("specialty"));
                }
            }
            System.out.println("\n=== MATERIAS ===");
            String subjectsSQL = "SELECT subject_id, code, subject_name, credits, department FROM subjects
ORDER BY subject_id";
            try (Statement stmt = conn.createStatement();
                 ResultSet rs = stmt.executeQuery(subjectsSQL)) {
                while (rs.next()) {
                    System.out.printf("ID: %d | %s - %s | %d creditos | %s%n",
                            rs.getInt("subject_id"),
                            rs.getString("code"),
                            rs.getString("subject_name"),
                            rs.getInt("credits"),
                            rs.getString("department"));
                }
            }
            System.out.println("\n=== INSCRIPCIONES ===");
            String enrollmentsSQL = "SELECT enrollment_id, student_id, subject_id, grade, status, semester
FROM student_subjects ORDER BY enrollment_id";
            try (Statement stmt = conn.createStatement();
                 ResultSet rs = stmt.executeQuery(enrollmentsSQL)) {
                while (rs.next()) {
                    String grade = rs.getDouble("grade") > 0 ? String.format("%.2f", rs.getDouble("grade")) :
"N/A";
                    System.out.printf("ID: %d | Est: %d | Mat: %d | Nota: %s | %s | Sem: %s%n",
                            rs.getInt("enrollment_id"),
                            rs.getInt("student_id"),
                            rs.getInt("subject_id"),
                            grade,
                            rs.getString("status"),
                            rs.getString("semester"));
                }
            }
        } catch (SQLException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

Link de Github:

https://github.com/IIK1ILLERII/University-Administrative-System.git