



PROYECTO BASE DE DATOS

INTEGRANTES:

- DANIEL DIAZ PAPU GEI
- MATEO BARBA

ÍNDICE

- 01** CREACION DE TABLAS
- 02** CREACION DE ROLES
- 03** ASIGNACION DE PERMISOS
- 04** El nivel de detalle
- 05** Uso de ejemplos
- 06** Tono y estilo
- 07** Evitar ambigüedades
- 08** Paso a paso



01. CREACION DE TABLAS

```
4      -- 1. Tabla de géneros
5 • Ⓛ CREATE TABLE generos (
6          genero_id INT AUTO_INCREMENT PRIMARY KEY,
7          descripcion VARCHAR(20) UNIQUE NOT NULL
8      );
9
10     -- 2. Tabla de estados de matrícula
11 • Ⓛ CREATE TABLE estados_matricula (
12         estado_id INT AUTO_INCREMENT PRIMARY KEY,
13         descripcion VARCHAR(30) UNIQUE NOT NULL
14     );
15
16     -- 3. REPRESENTANTES
17 • Ⓛ CREATE TABLE representantes (
18         representante_id INT AUTO_INCREMENT PRIMARY KEY,
19         nombres VARCHAR(100) NOT NULL,
20         apellidos VARCHAR(100) NOT NULL,
21         telefono VARCHAR(15) NOT NULL CHECK (telefono REGEXP '^[0-9]{10}$'),
22         correo VARCHAR(100) NOT NULL UNIQUE
23     );
24
25     -- 4. ESTUDIANTES
26 • Ⓛ CREATE TABLE estudiantes (
27         estudiante_id INT AUTO_INCREMENT PRIMARY KEY,
28         cedula VARCHAR(10) UNIQUE NOT NULL CHECK (cedula REGEXP '^[0-9]{10}$'),
29         genero_id INT,
30         nombres VARCHAR(100) NOT NULL,
31         apellidos VARCHAR(100) NOT NULL,
32         fecha_nacimiento DATE NOT NULL,
33         direccion TEXT NOT NULL,
34         telefono VARCHAR(15) CHECK (telefono REGEXP '^[0-9]{10}$'),
35         correo VARCHAR(100) UNIQUE,
36         representante_id INT,
37         FOREIGN KEY (genero_id) REFERENCES generos(genero_id),
38         FOREIGN KEY (representante_id) REFERENCES representantes(representante_id)
39     );
```



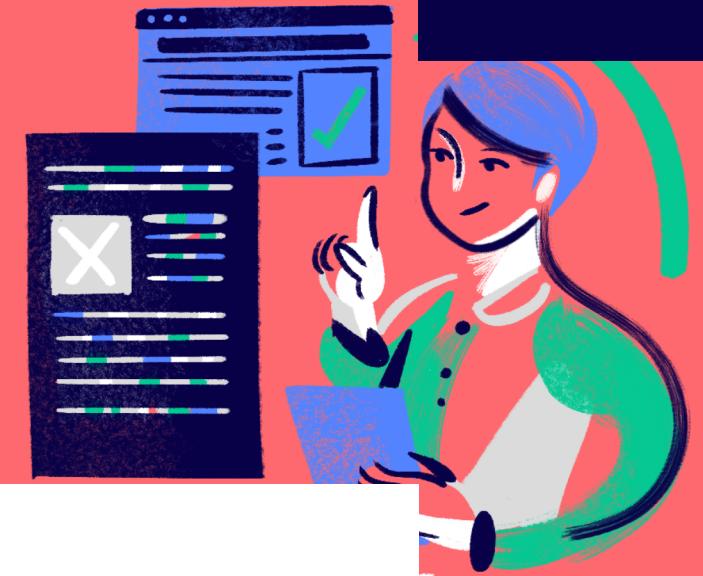
01. CREACION DE TABLAS

```
-- 5. DOCENTES
CREATE TABLE docentes (
    docente_id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    apellidos VARCHAR(100) NOT NULL,
    titulo_obtenido VARCHAR(200) NOT NULL,
    telefono VARCHAR(15) CHECK (telefono REGEXP '^[0-9]{10}$'),
    correo VARCHAR(100) UNIQUE
);
```

```
-- 6. MATERIAS
CREATE TABLE materias (
    materia_id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL UNIQUE,
    descripcion TEXT,
    horas_semanales INT NOT NULL CHECK (horas_semanales > 0)
);
```

```
-- 7. PARALELOS
CREATE TABLE paralelos (
    paralelo_id INT AUTO_INCREMENT PRIMARY KEY,
    nombre_paralelo VARCHAR(10) NOT NULL,
    grado VARCHAR(10) NOT NULL CHECK (grado IN ('Primero', 'Segundo', 'Tercero', 'Cuarto', 'Quinto', 'Sexto', 'Séptimo')),
    cupo_maximo INT CHECK (cupo_maximo > 0)
);

-- 8. HORARIOS
CREATE TABLE horarios (
    horario_id INT AUTO_INCREMENT PRIMARY KEY,
    dia VARCHAR(20) NOT NULL CHECK (dia IN ('Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes')),
    hora_inicio TIME NOT NULL,
    hora_fin TIME NOT NULL,
    CHECK (hora_inicio < hora_fin)
);
```



01. CREACION DE TABLAS

-- 9. ASIGNACION DE MATERIAS

```
CREATE TABLE asignacion_materias (
    asignacion_id INT AUTO_INCREMENT PRIMARY KEY,
    materia_id INT NOT NULL,
    docente_id INT,
    paralelo_id INT NOT NULL,
    horario_id INT NOT NULL,
    FOREIGN KEY (materia_id) REFERENCES materias(materia_id) ON DELETE CASCADE,
    FOREIGN KEY (docente_id) REFERENCES docentes(docente_id) ON DELETE SET NULL,
    FOREIGN KEY (paralelo_id) REFERENCES paralelos(paralelo_id) ON DELETE CASCADE,
    FOREIGN KEY (horario_id) REFERENCES horarios(horario_id) ON DELETE CASCADE
);
```

-- 10. MATRICULAS

```
CREATE TABLE matriculas (
    matricula_id INT AUTO_INCREMENT PRIMARY KEY,
    estudiante_id INT NOT NULL,
    paralelo_id INT NOT NULL,
    fecha_matricula DATE NOT NULL,
    estado_id INT NOT NULL,
    FOREIGN KEY (estudiante_id) REFERENCES estudiantes(estudiante_id) ON DELETE CASCADE,
    FOREIGN KEY (paralelo_id) REFERENCES paralelos(paralelo_id) ON DELETE CASCADE,
    FOREIGN KEY (estado_id) REFERENCES estados_matricula(estado_id)
);
```

-- 11. CALIFICACIONES

```
CREATE TABLE calificaciones (
    calificacion_id INT AUTO_INCREMENT PRIMARY KEY,
    estudiante_id INT NOT NULL,
    asignacion_id INT NOT NULL,
    trimestre INT NOT NULL CHECK (trimestre BETWEEN 1 AND 3),
    nota DECIMAL(5,2) NOT NULL CHECK (nota BETWEEN 0 AND 10),
    FOREIGN KEY (estudiante_id) REFERENCES estudiantes(estudiante_id) ON DELETE CASCADE,
    FOREIGN KEY (asignacion_id) REFERENCES asignacion_materias(asignacion_id) ON DELETE CASCADE
);
```

-- 12. ASISTENCIAS

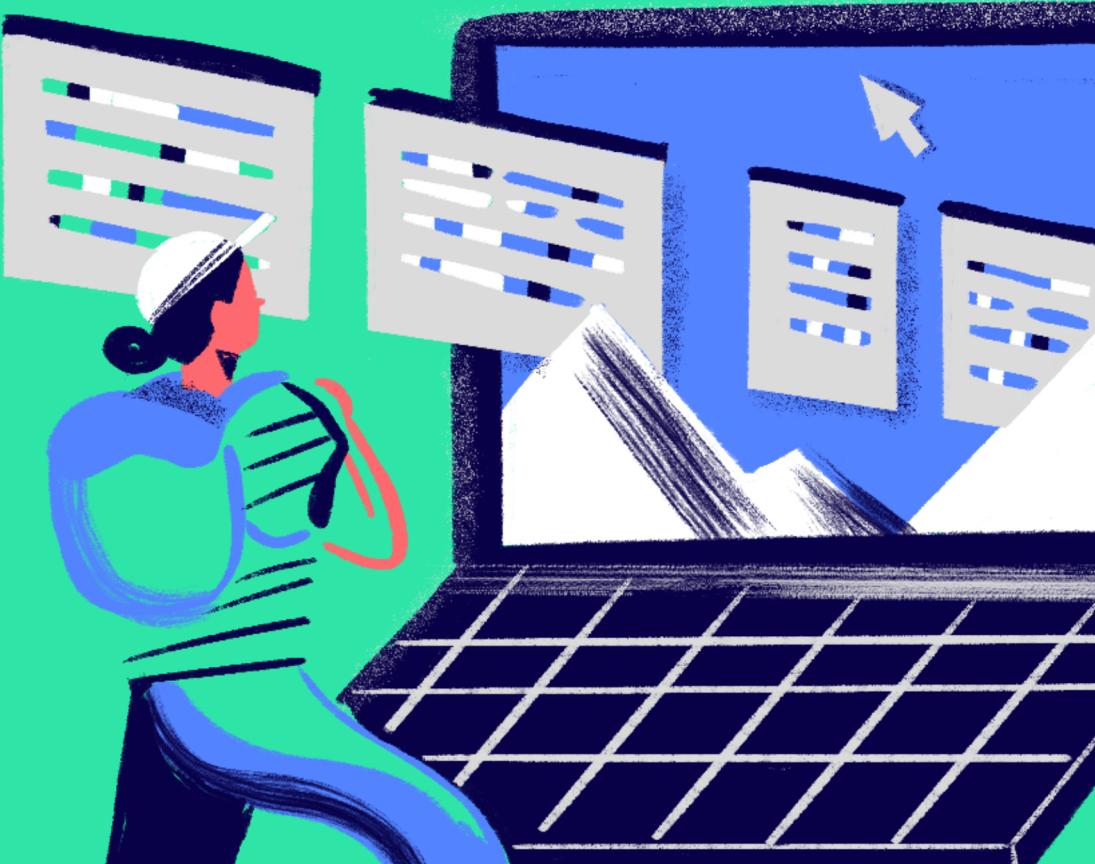
```
CREATE TABLE asistencias (
    asistencia_id INT AUTO_INCREMENT PRIMARY KEY,
    estudiante_id INT NOT NULL,
    asignacion_id INT NOT NULL,
    fecha_asistencia DATE NOT NULL,
    presente BOOLEAN NOT NULL DEFAULT FALSE,
    FOREIGN KEY (estudiante_id) REFERENCES estudiantes(estudiante_id) ON DELETE CASCADE,
    FOREIGN KEY (asignacion_id) REFERENCES asignacion_materias(asignacion_id) ON DELETE CASCADE
);
```



02. CREACION DE ROLES

```
CREATE ROLE administrador;  
CREATE ROLE secretaria;  
CREATE ROLE docente;  
CREATE ROLE representante;  
CREATE ROLE auditor;
```

buenas prácticas para administrar
permisos y mantener la seguridad
organizada.



03. ASIGNAMOS PERMISOS

-- ADMINISTRADOR

```
GRANT ALL PRIVILEGES ON educacion.* TO administrador;
```

-- SECRETARÍA

```
GRANT SELECT, INSERT, UPDATE ON educacion.estudiantes TO secretaria;
```

```
GRANT SELECT, INSERT, UPDATE ON educacion.matriculas TO secretaria;
```

```
GRANT SELECT, INSERT, UPDATE ON educacion.asignacion_materias TO secretaria;
```

-- DOCENTE

```
GRANT SELECT, INSERT, UPDATE ON educacion.calificaciones TO docente;
```

```
GRANT SELECT, INSERT, UPDATE ON educacion.asistencias TO docente;
```

```
GRANT SELECT ON educacion.estudiantes TO docente;
```

```
GRANT SELECT ON educacion.materias TO docente;
```

-- REPRESENTANTE

```
GRANT SELECT ON educacion.calificaciones TO representante;
```

```
GRANT SELECT ON educacion.asistencias TO representante;
```

```
GRANT SELECT ON educacion.estudiantes TO representante;
```

-- AUDITOR

```
GRANT SELECT ON educacion.* TO auditor;
```

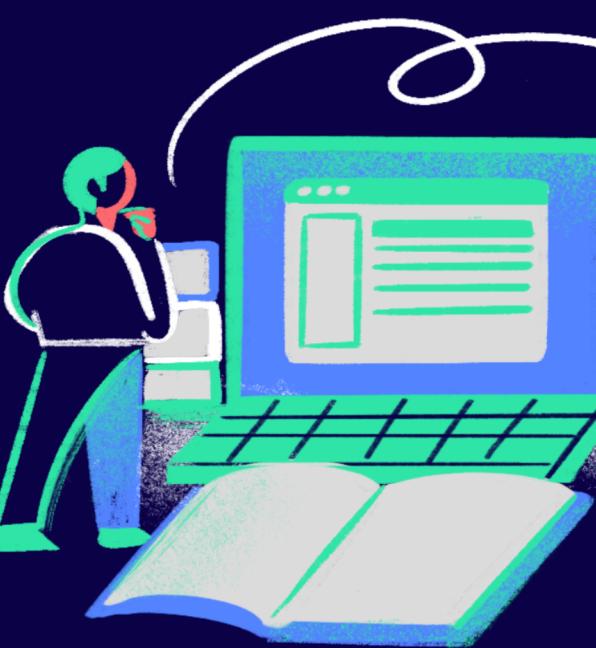
TODOS LOS
PRIVILEGIOS

MODIFICAR(INSERTAR
Y ACTUALIZAR)

MODIFICAR Y LEER

LEER

LEER



04. CREACION DE USUARIOS



```
158      -- Crear usuarios  
159 •  CREATE USER 'admin_user'@'localhost' IDENTIFIED BY 'admin123';  
160 •  CREATE USER 'sec_user'@'localhost' IDENTIFIED BY 'sec123';  
161 •  CREATE USER 'docente_user'@'localhost' IDENTIFIED BY 'docente123';  
162 •  CREATE USER 'rep_user'@'localhost' IDENTIFIED BY 'rep123';  
163 •  CREATE USER 'auditor_user'@'localhost' IDENTIFIED BY 'auditor123';
```

CON ESTO SOLO PERMITIMOS PERSONAS AUTORIZADAS PUEDEN ENTRAR A LA BASE DE DATOS.

CADA USUARIO TIENE UN NOMBRE DE USUARIO (USERNAME) Y UNA CONTRASEÑA (PASSWORD).

04. EL NIVEL DE DETALLE

ASIGNACION



```
-- Asignar roles  
GRANT administrador TO 'admin_user'@'localhost';  
GRANT secretaria TO 'sec_user'@'localhost';  
GRANT docente TO 'docente_user'@'localhost';  
GRANT representante TO 'rep_user'@'localhost';  
GRANT auditor TO 'auditor_user'@'localhost';  
  
-- Activar el rol por defecto  
SET DEFAULT ROLE administrador TO 'admin_user'@'localhost';  
SET DEFAULT ROLE secretaria TO 'sec_user'@'localhost';  
SET DEFAULT ROLE docente TO 'docente_user'@'localhost';  
SET DEFAULT ROLE representante TO 'rep_user'@'localhost';  
SET DEFAULT ROLE auditor TO 'auditor_user'@'localhost';
```

Con:
-GRANT-
Se usa para conceder permisos o roles a usuario que nos nosotros deseemos

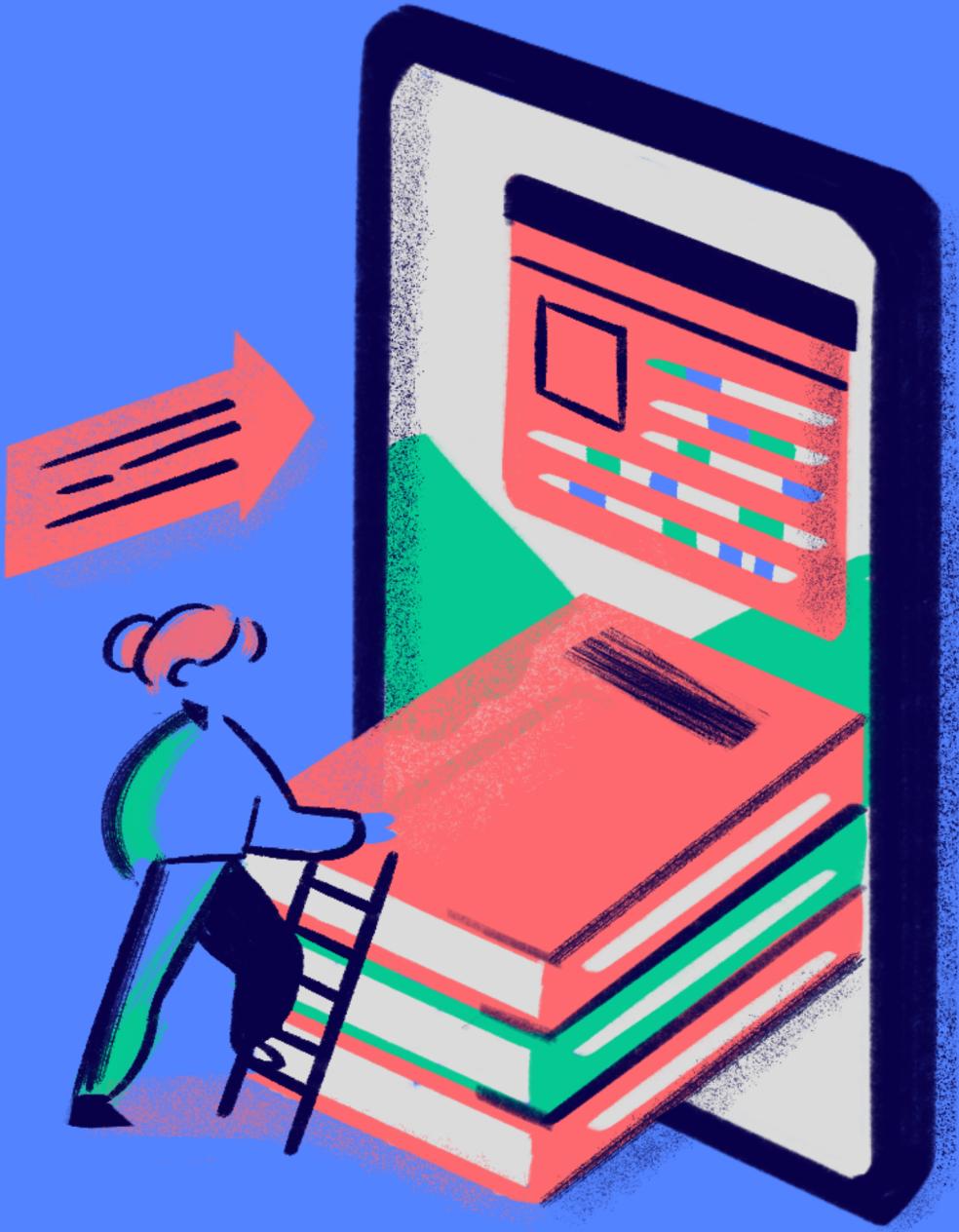
Con:
-SET DEFAULT ROLE-
Se usa para especificar que rol o roles se activan automaticamente cuando un usuario inicie sesion en la base de datos

05. USUARIOS

Solo es una tabla para registrar que rol tiene cada usuario, y agregamos los 5 registros que tenemos.

```
CREATE TABLE auditoria_usuarios_roles (
    id INT AUTO_INCREMENT PRIMARY KEY,
    usuario VARCHAR(50) NOT NULL,
    rol VARCHAR(50) NOT NULL,
    fecha_asignacion DATETIME DEFAULT CURRENT_TIMESTAMP
);

INSERT INTO auditoria_usuarios_roles (usuario, rol) VALUES
('admin_user', 'administrador'),
('sec_user', 'secretaria'),
('docente_user', 'docente'),
('rep_user', 'representante'),
('auditor_user', 'auditor');
```



05. USUARIOS

Crea una tabla en la cual podemos darle roles a quien quiera registrarse

```
CREATE TABLE usuarios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    usuario VARCHAR(50) UNIQUE NOT NULL,
    contrasena VARBINARY(256) NOT NULL, -- Guardamos con SHA2
    correo VARBINARY(256),           -- Encriptado con AES
    telefono VARBINARY(256),         -- Encriptado con AES
    rol ENUM('administrador', 'docente', 'representante', 'secretaria','auditor') NOT NULL,
    creado_en TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- CREACION USUARIOS (EJEMPLO)

-- Insertar un docente
INSERT INTO usuarios (nombre, usuario, contrasena, correo, telefono, rol)
VALUES (
    'Carlos Pérez',
    'cperez',
    SHA2('miClaveDocente123', 256),          -- Contraseña segura
    AES_ENCRYPT('carlos.perez@escuela.edu.ec', 'LlaveAES123'), -- Correo encriptado
    AES_ENCRYPT('0987654321', 'LlaveAES123'),      -- Teléfono encriptado
    'docente'
);
```

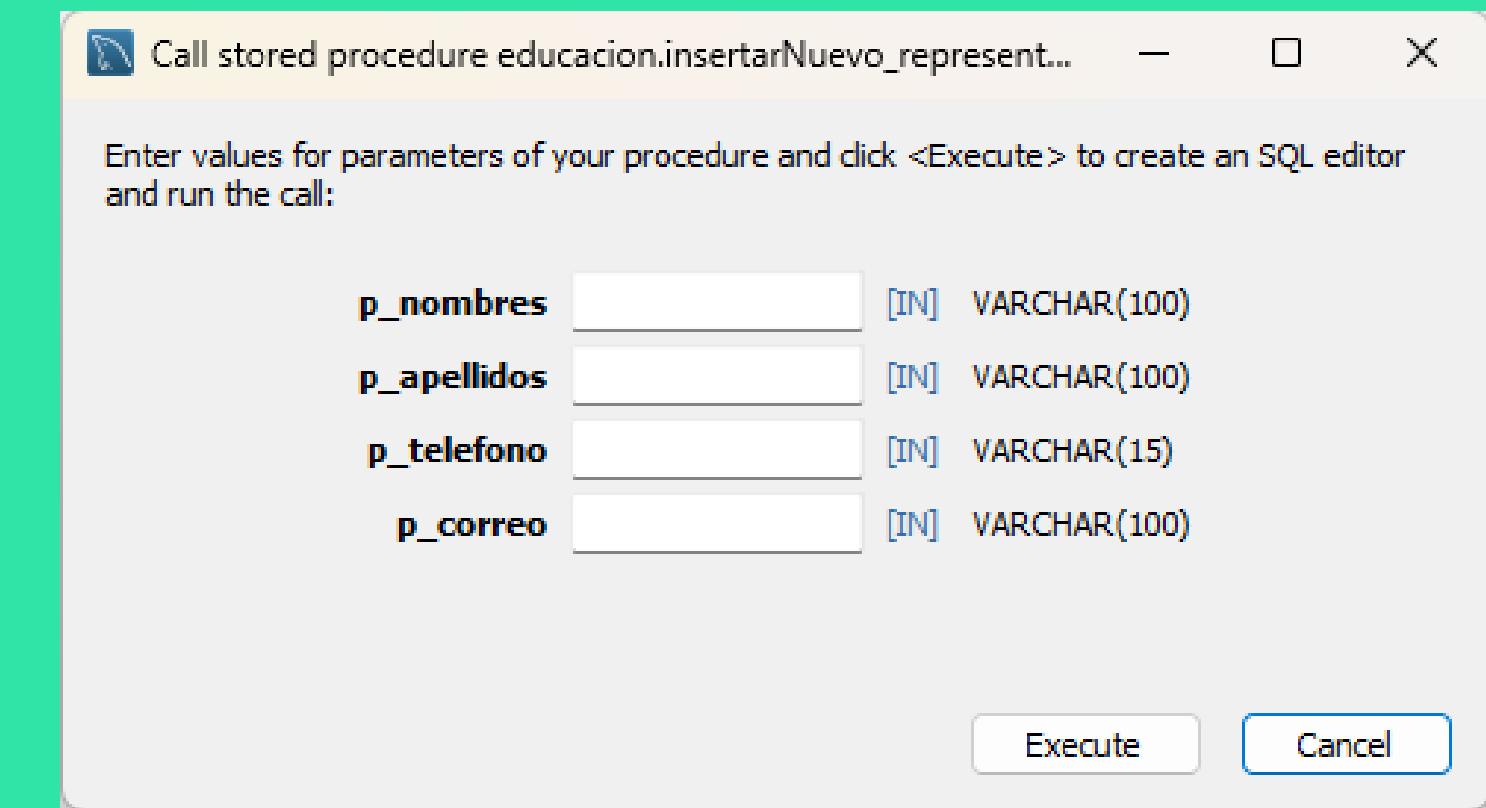
Se usa encriptación, dependiendo del dato que estemos usando como por ejemplo:

- SHA-2 se usa para verificar la integridad de los datos, asegurando que no hayan sido modificados.
- AES se usa para proteger la confidencialidad de los datos, haciéndolos ilegibles sin la clave.

06. PROCEDIMIENTOS ALMACENADOS

```
-- 1. AGREGAR UN NUEVO REPRESENTANTE
delimiter //

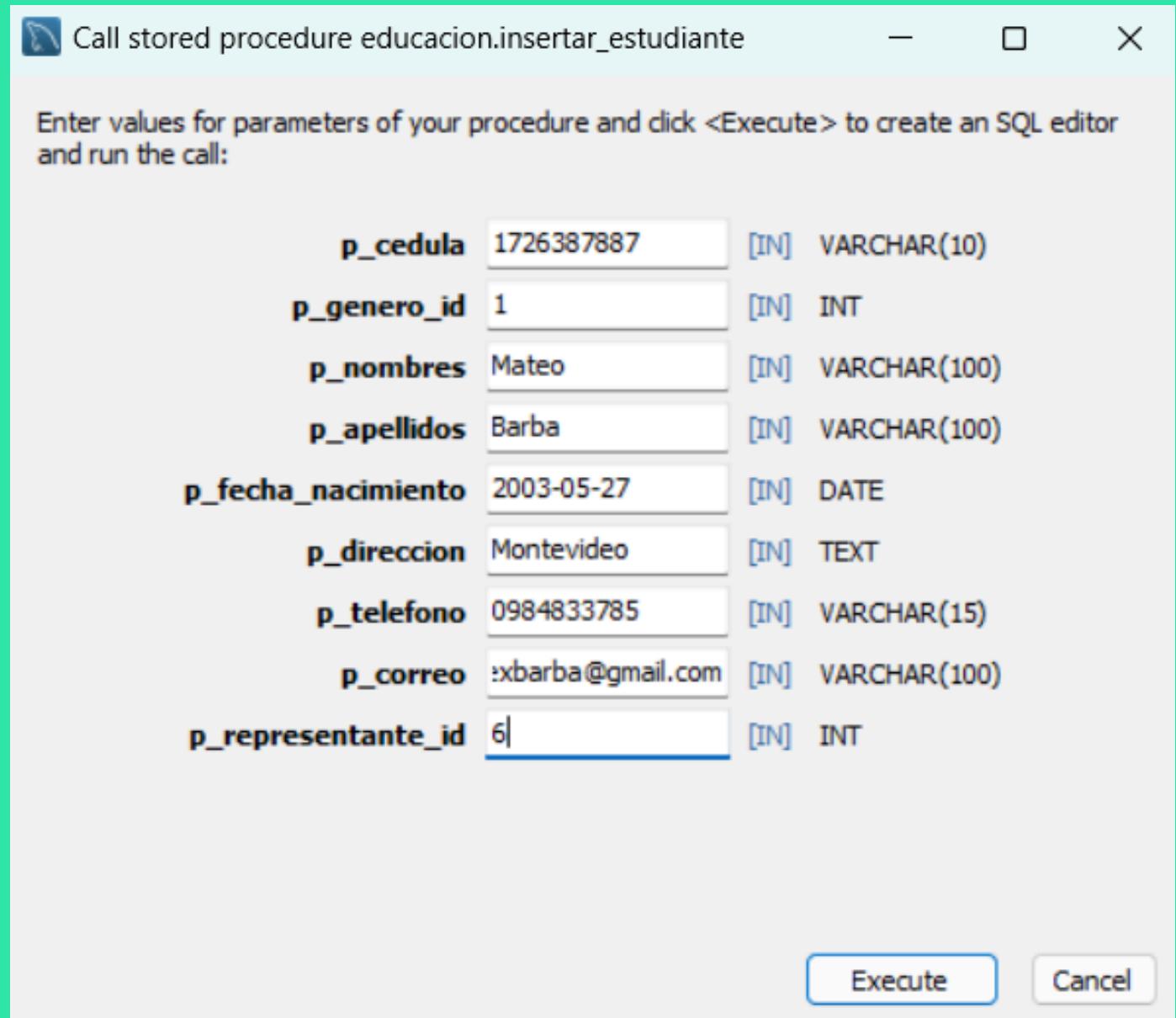
CREATE PROCEDURE insertarNuevo_representante(
    IN p_nombres VARCHAR(100),
    IN p_apellidos VARCHAR(100),
    IN p_telefono VARCHAR(15),
    IN p_correo VARCHAR(100)
)
BEGIN
    -- Insertar el nuevo representante en la tabla 'representantes'
    INSERT INTO representantes (nombres, apellidos, telefono, correo)
        VALUES (p_nombres, p_apellidos, p_telefono, p_correo);
END //
delimiter ;
```



representante_id	nombres	apellidos	telefono	correo
92	Eduardo	Hernandez	0974076175	e.hernandez@yahoo.com
93	Luca	Mendez	0912746232	lucamendez7395@yahoo.com
94	Liberto	Ferrer	0938366954	l.ferrer1058@hotmail.com
95	Ignacio	Gimenez	0936194326	igimenez5171@outlook.com
96	José	Rodriguez	0993529667	j.rodriguez@gmail.com
97	Diego	Diaz	0993747682	ddiaz@yahoo.com
98	Morena	Gonzalez	0936274137	m-gonzalez875@hotmail.com
99	Reno	Miguel	0946431607	r.miguel4240@hotmail.com
100	Felipe	Ortiz	0949605684	f.ortiz@outlook.com
101	Hector ...	Diaz Sand...	0986731490	sandiaz@
102	Hector ...	Diaz Sand...	0986731490	sandiaz@gmail.com
*	NULL	NULL	NULL	NULL

06. PROCEDIMIENTOS ALMACENADOS

```
-- 2. AÑADIR UN NUEVO ESTUDIANTE
delimiter //
CREATE PROCEDURE insertar_estudiante(
    IN p_cedula VARCHAR(10),
    IN p_genero_id INT,
    IN p_nombres VARCHAR(100),
    IN p_apellidos VARCHAR(100),
    IN p_fecha_nacimiento DATE,
    IN p_direccion TEXT,
    IN p_telefono VARCHAR(15),
    IN p_correo VARCHAR(100),
    IN pRepresentante_id INT
)
BEGIN
    INSERT INTO estudiantes (cedula, genero_id, nombres, apellidos, fecha_nacimiento, direccion, telefono, correo, representante_id)
    VALUES (p_cedula, p_genero_id, p_nombres, p_apellidos, p_fecha_nacimiento, p_direccion, p_telefono, p_correo, pRepresentante_id);
END //
delimiter ;
```

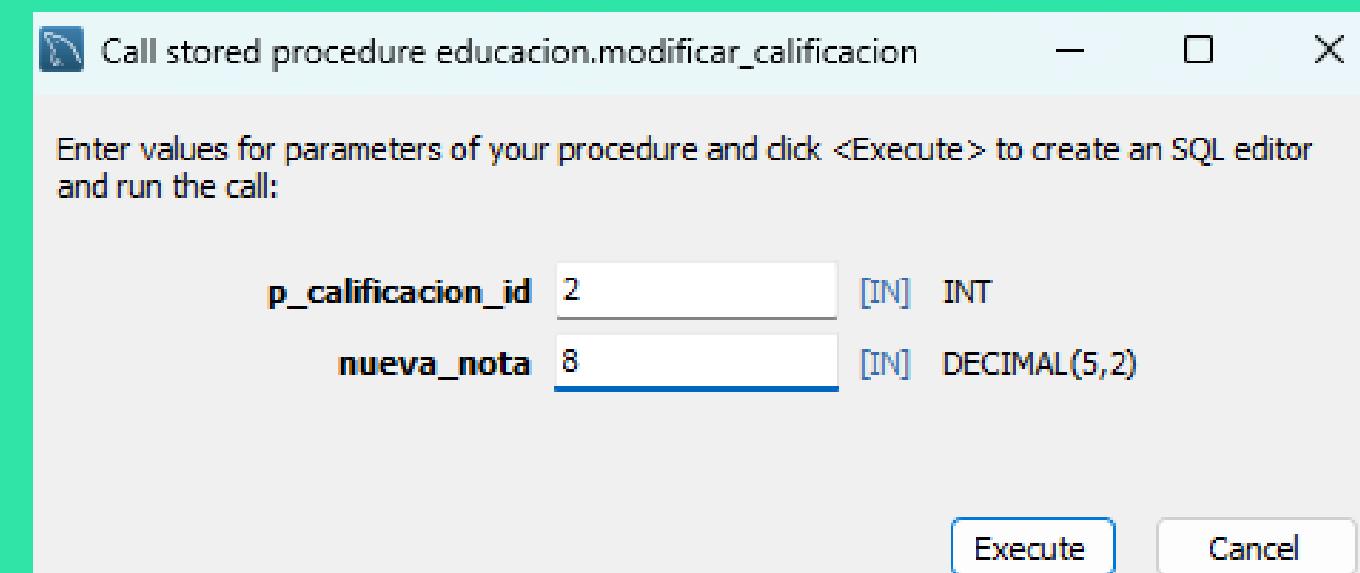


ID	Cédula	Género	Nombres	Apellidos	Fecha Nacimiento	Dirección	Teléfono	Correo	Representante ID
198	1722569272	1	Fernanda	Crespo	2016-01-24	Ap #692-9186 Curabitur Ave	0936866360	c_fernanda@hotmail.com	25
199	1784862079	1	Lucero	Alvarez	2018-09-28	P.O. Box 582, 9955 Arcu Street	0928154653	l_alvarez8605@outlook.com	25
200	1788170873	2	Marina	Muñoz	2018-09-11	An #203-9097 Fermentum Ave	0954435236	marina.muoz4152@hotmail.com	78
203	1726387887	1	Mateo	Barba	2003-05-27	Montevideo	0984833785	alexbarba@gmail.com	6

06. PROCEDIMIENTOS ALMACENADOS

	calificacion_id	estudiante_id	asignacion_id	trimestre	nota
▶	1	1	1	1	9.00
	2	2	25	3	3.00
	3	3	30	2	6.00
	4	4	10	1	7.00

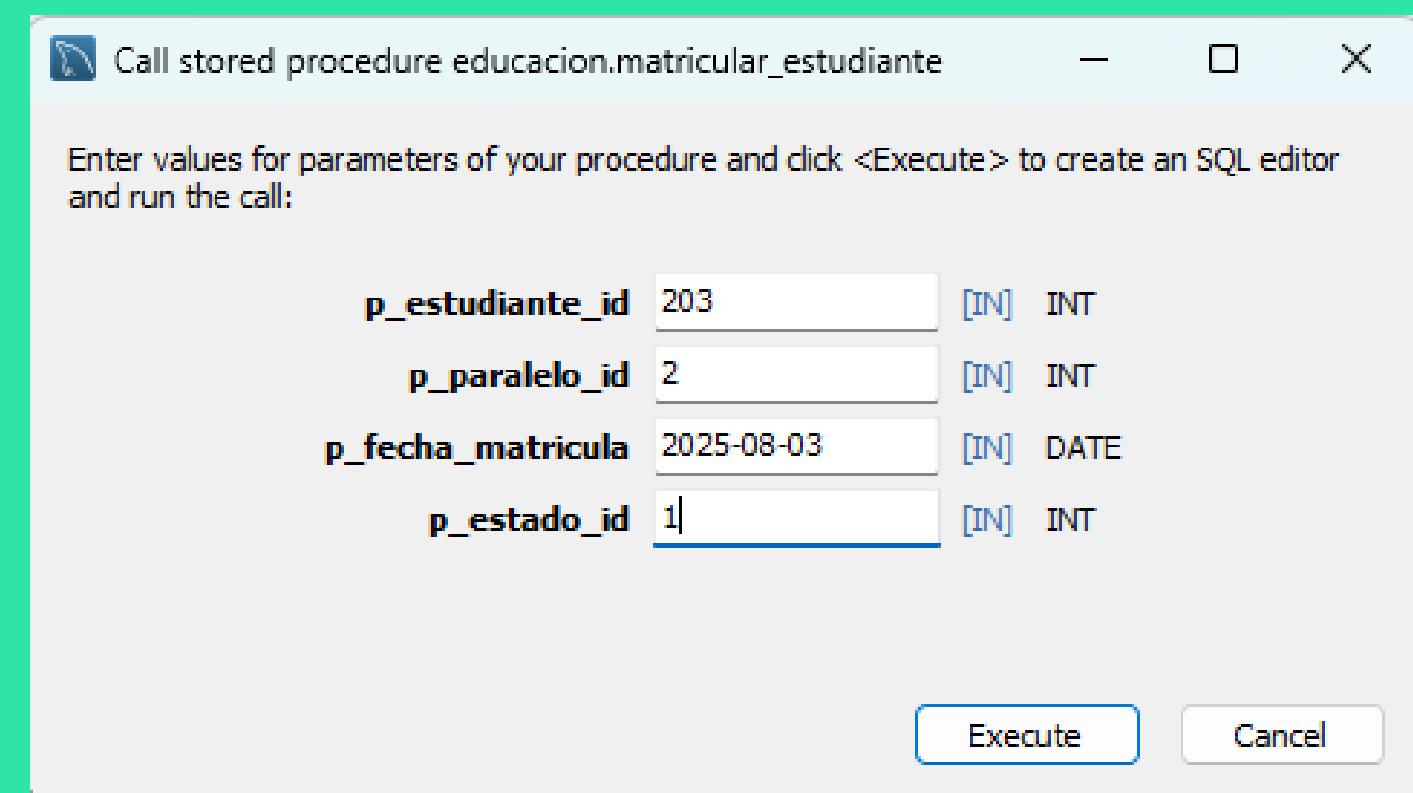
```
-- 3. modificar una calificación  
DELIMITER //  
CREATE PROCEDURE modificar_calificación(  
    IN p_calificación_id INT, -- Cambié el nombre del parámetro a 'p_calificación_id'  
    IN nueva_nota DECIMAL(5,2)  
)  
BEGIN  
    UPDATE calificaciones  
    SET nota = nueva_nota  
    WHERE calificación_id = p_calificación_id; -- Usamos 'p_calificación_id' aquí  
END //  
DELIMITER ;
```



	calificacion_id	estudiante_id	asignacion_id	trimestre	nota
▶	1	1	1	1	9.00
	2	2	25	3	8.00
	3	3	30	2	6.00
	4	4	10	1	7.00

06. PROCEDIMIENTOS ALMACENADOS

```
-- 4. matricular un estudiante
delimiter //
CREATE PROCEDURE matricular_estudiante(
    IN p_estudiante_id INT,
    IN p_paralelo_id INT,
    IN p_estado_id INT
)
BEGIN
    INSERT INTO matriculas (estudiante_id, paralelo_id, fecha_matricula, estado_id)
    VALUES (p_estudiante_id, p_paralelo_id, CURDATE(), p_estado_id);
END //
delimiter ;
```

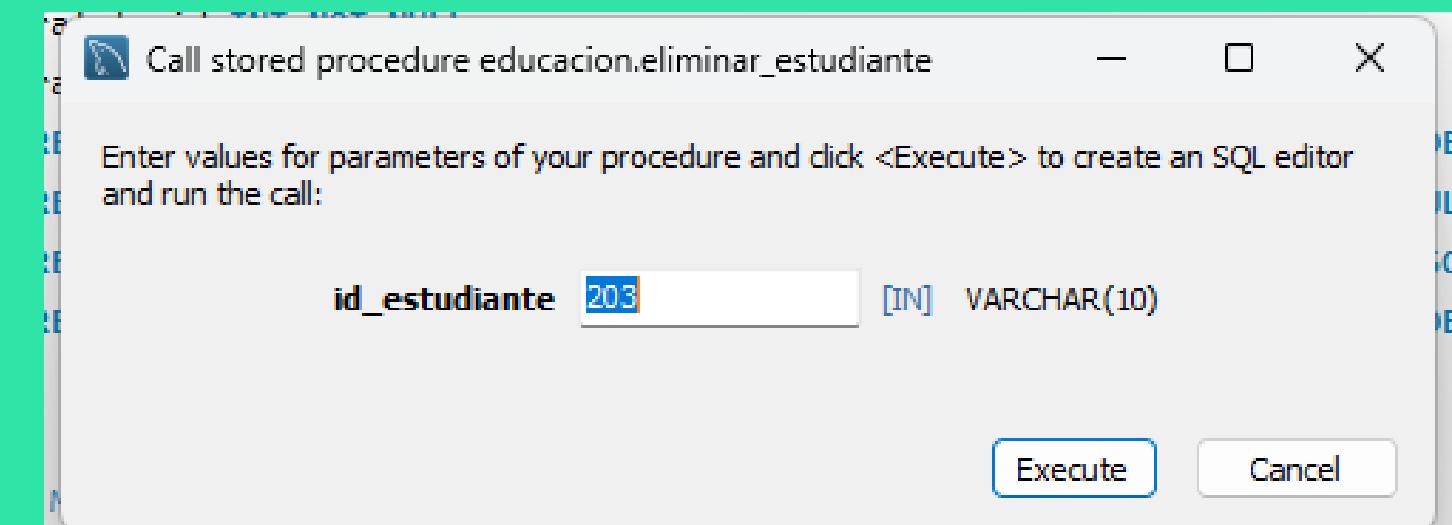


198	198	3	2025-07-06	3
199	199	2	2025-07-05	3
200	200	2	2025-07-13	2
201	203	2	2025-08-03	1
NULL	NULL	NULL	NULL	NULL

06. PROCEDIMIENTOS ALMACENADOS

```
-- 5. Eliminar Estudiante
DELIMITER //
CREATE PROCEDURE eliminar_estudiante(IN id_estudiante VARCHAR(10))
BEGIN
    DELETE FROM estudiantes WHERE estudiante_id = id_estudiante;
END //
DELIMITER ;
```

198	1722569272	1	Fernanda	Crespo	2016-01-24	Ap #692-9186 Curabitur Ave	0936866360	c_1
199	1784862079	1	Lucero	Alvarez	2018-09-28	P.O. Box 582, 9955 Arcu Street	0928154653	l_a
200	1788170873	2	Marina	Muñoz	2018-09-11	Ap #203-9097 Fermentum Ave	0954435236	ma
203	1726387887	1	Mateo	Barba	2003-05-27	Montevideo	0984833785	ale



197	1731100110	1	Aaron	Hurquez	2017-03-20	Ap #107-7180 Fidostr	0915177120	o_d
198	1722569272	1	Fernanda	Crespo	2016-01-24	Ap #692-9186 Curabitur Ave	0936866360	c_1
199	1784862079	1	Lucero	Alvarez	2018-09-28	P.O. Box 582, 9955 Arcu Street	0928154653	l_a
200	1788170873	2	Marina	Muñoz	2018-09-11	Ap #203-9097 Fermentum Ave	0954435236	ma

198	198	3	2025-07-06	3
199	199	2	2025-07-05	3
200	200	2	2025-07-13	2
NUL	NUL	NUL	NUL	NUL

07. FUNCIONES

```
-- 1. OBTENER EL ESTADO DE MATRICULA DE UN ESTUDIANTE
DELIMITER //
CREATE FUNCTION obtener_estado_matricula(estudiante_id INT)
RETURNS VARCHAR(30)
DETERMINISTIC
BEGIN
    DECLARE estado VARCHAR(30);
    SELECT sm.descripcion INTO estado
    FROM matriculas m
    JOIN estados_matricula sm ON m.estado_id = sm.estado_id
    WHERE m.estudiante_id = estudiante_id ORDER BY m.fecha_matricula DESC LIMIT 1;
    RETURN estado;
END //
DELIMITER ;
```

Call stored function educación.obtener_estado_matricula

Enter values for parameters of your function and click <Execute> to create an SQL editor and run the call:

estudiante_id INT

Execute Cancel

Result Grid | Filter Rows: | Export

	educación.obtener_estado_matricula(2)
▶	Retirada

07. FUNCIONES

```
-- 2. CONTAR CUANTOS ESTUDIANTES HAY EN UN PARALELO

DELIMITER //

CREATE FUNCTION contar_estudiantes_paralelo(f_paralelo_id INT)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE total INT;
    SELECT COUNT(*) INTO total
    FROM matriculas
    WHERE paralelo_id = f_paralelo_id;
    RETURN total;
END //
DELIMITER ;
```

Call stored function educación.contar_estudiantes_paralelo

Enter values for parameters of your function and click <Execute> to create an SQL editor and run the call:

f_paralelo_id INT

Execute Cancel

Result Grid | Filter Rows:

	educación.contar_estudiantes_paralelo(6)
▶	19

07. FUNCIONES

```
-- 3. cantidad de estudiantes por genero
DELIMITER //

CREATE FUNCTION contar_estudiantes_genero(p_genero_id INT)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE total INT;

    -- Contamos la cantidad de estudiantes según el género
    SELECT COUNT(*) INTO total
    FROM estudiantes
    WHERE genero_id = p_genero_id;

    -- Devolvemos el total
    RETURN total;
END //

DELIMITER ;
```

Call stored function educación.contar_estudiantes_genero

Enter values for parameters of your function and click <Execute> to create an SQL editor and run the call:

p_genero_id INT

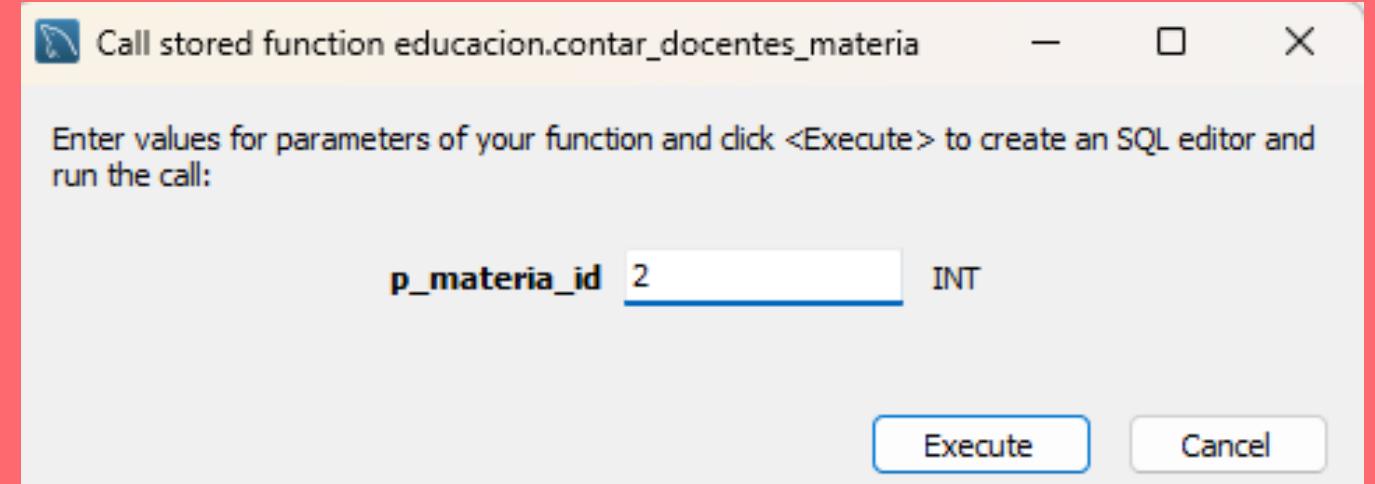
Execute Cancel

Result Grid | Filter Rows:

	educación.contar_estudiantes_genero(1)
▶	110

07. FUNCIONES

```
-- 4. funcion poara obtener docentes asignados a una meteria en especifico  
  
DELIMITER //  
  
CREATE FUNCTION contar_docentes_materia(p_materia_id INT)  
RETURNS INT  
DETERMINISTIC  
BEGIN  
  
    DECLARE total INT;  
  
    SELECT COUNT(*) INTO total  
    FROM asignacion_materias  
    WHERE materia_id = p_materia_id;  
    -- Devolvemos el total  
  
    RETURN total;  
END //  
  
DELIMITER ;
```



Result Grid	
	educacion.contar_docentes_materia(2)
▶	8

07. FUNCIONES

```
-- 5. funcion para verificar si un docente tiene un correo registrado  
DELIMITER //  
  
CREATE FUNCTION tiene_correo_docente(docente_id INT)  
RETURNS INT  
DETERMINISTIC  
BEGIN  
    DECLARE tiene_correo INT;  
  
    -- Verificamos si el docente tiene correo registrado  
    SELECT COUNT(*) INTO tiene_correo  
    FROM docentes  
    WHERE docente_id = docente_id AND correo IS NOT NULL;  
  
    -- Devolvemos 1 si tiene correo, 0 si no tiene  
    RETURN IF(tiene_correo > 0, 1, 0);  
END //  
  
DELIMITER ;
```

Call stored function educación.tiene_correo_docente

Enter values for parameters of your function and click <Execute> to create an SQL editor and run the call:

docente_id INT

Result Grid		Filter Rows:
education.tiene_correo_docente(2)	1	

08. JOINS

```
-- JOINS --  
-- 1. Listar todos los estudiantes con el nombre de la materia, docente y horario  
SELECT  
    e.nombres AS estudiante,  
    e.apellidos AS apellido_estudiante,  
    m.nombre AS materia,  
    d.nombre AS docente,  
    d.apellidos AS apellido_docente,  
    h.dia AS dia_clase,  
    h.hora_inicio,  
    h.hora_fin  
FROM matriculas ma  
JOIN estudiantes e ON ma.estudiante_id = e.estudiante_id  
JOIN paralelos p ON ma.paralelo_id = p.paralelo_id  
JOIN asignacion_materias am ON p.paralelo_id = am.paralelo_id  
JOIN materias m ON am.materia_id = m.materia_id  
JOIN docentes d ON am.docente_id = d.docente_id  
JOIN horarios h ON am.horario_id = h.horario_id;
```

	estudiante	apellido_estudiante	materia	docente	apellido_docente	dia_clase	hora_inicio	hora_fin
▶	Hernández	Alonso	Lengua y Literatura	Andrés	López	Lunes	07:00:00	09:00:00
	Isabella	Nieto	Lengua y Literatura	Andrés	López	Lunes	07:00:00	09:00:00
	Rita	Cortes	Lengua y Literatura	Andrés	López	Lunes	07:00:00	09:00:00
	Marta	Casado	Lengua y Literatura	Andrés	López	Lunes	07:00:00	09:00:00
	Leonardo	Ibañez	Lengua y Literatura	Andrés	López	Lunes	07:00:00	09:00:00
	Jacqueline	Hidalgo	Lengua y Literatura	Andrés	López	Lunes	07:00:00	09:00:00
	Patricio	Leon	Lengua y Literatura	Andrés	López	Lunes	07:00:00	09:00:00
	Carlito	Delgado	Lengua y Literatura	Andrés	López	Lunes	07:00:00	09:00:00
	Carlos	Diez	Lengua y Literatura	Andrés	López	Lunes	07:00:00	09:00:00
	Rodrigo	Muñoz	Inglés	Isabel	Flores	Lunes	07:00:00	09:00:00
	Xavi	Lopez	Inglés	Isabel	Flores	Lunes	07:00:00	09:00:00
	Fernanda	Flores	Inglés	Isabel	Flores	Lunes	07:00:00	09:00:00

08. JOINS

```
-- 2. listar a los estudiantes con su calificacion del primer bimestre
SELECT
    e.nombres AS estudiante,
    e.apellidos AS apellido_estudiante,
    m.nombre AS materia,
    c.trimestre,
    c.nota
FROM calificaciones c
JOIN estudiantes e ON c.estudiante_id = e.estudiante_id
JOIN asignacion_materias am ON c.asignacion_id = am.asignacion_id
JOIN materias m ON am.materia_id = m.materia_id
WHERE c.trimestre = 1;
```

	estudiante	apellido_estudiante	materia	trimestre	nota
▶	Xavi	Lopez	Matemáticas	1	7.00
	Hernández	Alonso	Estudios Sociales	1	5.00
	Mariano	Cabrera	Ciencias Naturales	1	10.00
	Bianca	Sánchez	Lengua y Literatura	1	6.00
	Concepción	Vidal	Inglés	1	4.00
	Roberto	Martín	Ciencias Naturales	1	5.00
	Cuauhtémoc	Soto	Ciencias Naturales	1	8.00
	Jacín	Rodríguez	Lengua y Literatura	1	8.00
	Lucio	Vicente	Inglés	1	4.00
	Xavi	Esteban	Matemáticas	1	7.00
	Conchita	Pérez	Ciencias Naturales	1	6.00
	Luisa	Díaz	Matemáticas	1	10.00

08. JOINS

```
-- 3. Obtener los docentes que imparten clases en un paralelo específico
```

```
SELECT
```

```
    d.nombre AS docente,  
    d.apellidos AS apellido_docente,  
    m.nombre AS materia,  
    p.nombre_paralelo  
FROM asignacion_materias am  
JOIN docentes d ON am.docente_id = d.docente_id  
JOIN materias m ON am.materia_id = m.materia_id  
JOIN paralelos p ON am.paralelo_id = p.paralelo_id  
WHERE p.nombre_paralelo = 'A';
```

	docente	apellido_docente	materia	nombre_paralelo
▶	Sofía	Castro	Ciencias Naturales	A
	Raúl	Ortega	Estudios Sociales	A
	Ana	Mendoza	Estudios Sociales	A
	Luis	Mendoza	Ciencias Naturales	A
	Luis	Cordero	Inglés	A
	Elena	Pérez	Lengua y Literatura	A
	Laura	Gómez	Matemáticas	A
	Isabel	Flores	Inglés	A
	Carlos	Rodríguez	Ciencias Naturales	A
	Elena	Pérez	Lengua y Literatura	A
	Ana	Gómez	Inglés	A
	Maria	Ortega	Estudios Sociales	A

08. JOINS

-- 4. Obtener la cantidad de estudiantes por estado de matrícula

```
SELECT  
    em.descripcion AS estado_matricula,  
    COUNT(e.estudiante_id) AS total_estudiantes  
FROM matriculas ma  
JOIN estudiantes e ON ma.estudiante_id = e.estudiante_id  
JOIN estados_matricula em ON ma.estado_id = em.estado_id  
GROUP BY em.descripcion;
```

Result Grid | Filter Rows:

	estado_matricula	total_estudiantes
▶	Activa	52
▶	Retirada	102
▶	Suspendida	46

08. JOINS

-- 5. Obtener el total de horas semanales por materia en cada paralelo

```
SELECT
    p.nombre_paralelo,
    m.nombre AS materia,
    SUM(h.hora_fin - h.hora_inicio) AS total_horas_semanales
FROM asignacion_materias am
JOIN materias m ON am.materia_id = m.materia_id
JOIN paralelos p ON am.paralelo_id = p.paralelo_id
JOIN horarios h ON am.horario_id = h.horario_id
GROUP BY p.nombre_paralelo, m.nombre;
```

	nombre_paralelo	materia	total_horas_semanales
B		Ciencias Naturales	80000
	A	Ciencias Naturales	60000
B		Estudios Sociales	60000
	A	Estudios Sociales	60000
B		Inglés	100000
A		Inglés	60000
B		Lengua y Literatura	40000
A		Lengua y Literatura	40000
B		Matemáticas	80000
A		Matemáticas	40000

09. TRIGGERS

```
-- Tabla Única de logs para auditoria y control
CREATE TABLE log_acciones (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    tipo_accion VARCHAR(50),
    tabla_afectada VARCHAR(50),
    registro_id INT,
    descripcion TEXT,
    usuario VARCHAR(100),
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```
DELIMITER //
-- Trigger para INSERT en estudiantes
CREATE TRIGGER trg_insert_estudiante_log
AFTER INSERT ON estudiantes
FOR EACH ROW
BEGIN
    INSERT INTO log_acciones (tipo_accion, tabla_afectada, registro_id, descripcion, usuario)
    VALUES ('INSERT', 'estudiantes', NEW.estudiante_id, CONCAT('Estudiante ', NEW.nombres, ' ', NEW.apellidos, ' creado.'), USER());
END;
//

DELIMITER //
-- Trigger para UPDATE en calificaciones
CREATE TRIGGER trg_update_calificacion_log
AFTER UPDATE ON calificaciones
FOR EACH ROW
BEGIN
    INSERT INTO log_acciones (tipo_accion, tabla_afectada, registro_id, descripcion, usuario)
    VALUES ('UPDATE', 'calificaciones', NEW.calificacion_id,
            CONCAT('Nota cambiada de ', OLD.nota, ' a ', NEW.nota), USER());
END;
//

DELIMITER //
-- Trigger para DELETE en estudiantes
CREATE TRIGGER trg_delete_estudiante_log
BEFORE DELETE ON estudiantes
FOR EACH ROW
BEGIN
    INSERT INTO log_acciones (tipo_accion, tabla_afectada, registro_id, descripcion, usuario)
    VALUES ('DELETE', 'estudiantes', OLD.estudiante_id, CONCAT('Estudiante ', OLD.nombres, ' ', OLD.apellidos, ' eliminado.'), USER());
END;
//
```

Trigger	Event	Table	Statement	Timing	Created	sql_mode	Definer	character_set_client	collation_connection	Database Collation
trg_update_calificacion_log	UPDATE	calificaciones	BEGIN INSERT INTO log_eventos (tipo_accio...	AFTER	2025-08-03 23:27:41.98	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai
trg_insert_estudiante_log	INSERT	estudiantes	BEGIN INSERT INTO log_eventos (tipo_accio...	AFTER	2025-08-03 23:27:41.95	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai
trg_delete_estudiante_log	DELETE	estudiantes	BEGIN INSERT INTO log_eventos (tipo_accio...	BEFORE	2025-08-03 23:27:42.00	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai

log_id	tipo_accion	tabla_afectada	registro_id	descripcion	usuario	fecha
4	INSERT	estudiantes	203	Estudiante Mateo Barba creado.	root@localhost	2025-08-03 23:12:00
5	UPDATE	calificaciones	2	Nota cambiada de 3.00 a 8.00	root@localhost	2025-08-03 23:17:02
6	DELETE	estudiantes	203	Estudiante Mateo Barba eliminado.	root@localhost	2025-08-03 23:26:53

10. INDEX

```
-- INDICES
-- índices simples en claves foráneas
CREATE INDEX idx_estudiante_id ON calificaciones(estudiante_id);
CREATE INDEX idx_asignacion_id ON calificaciones(asignacion_id);

CREATE INDEX idx_matricula_estudiante ON matriculas(estudiante_id);
CREATE INDEX idx_matricula_paralelo ON matriculas(paralelo_id);

-- Índice simple para búsquedas por correo
CREATE INDEX idxRepresentante_correo ON representantes(correo);

-- Índice para ordenamiento por apellido
CREATE INDEX idx_estudiante_apellidos ON estudiantes(apellidos);

-- Consultas frecuentes por estudiante y asignatura
CREATE INDEX idx_calificaciones_est_asig ON calificaciones(estudiante_id, asignacion_id);

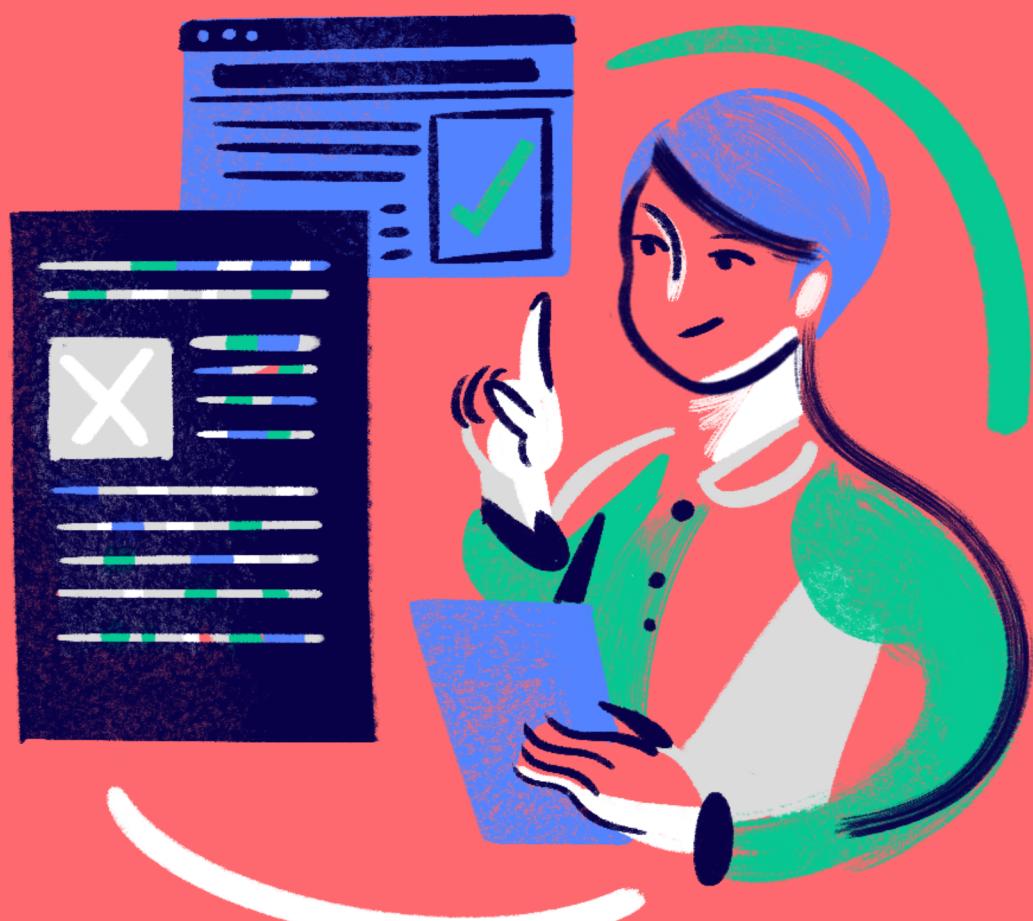
-- Consultas por paralelo y estado de matrícula
CREATE INDEX idx_matriculas_paralelo_estado ON matriculas(paralelo_id, estado_id);
```

calificaciones	idx_estudiante_id	No	BTREE	estudiante_id	1	A	199
calificaciones	idx_asignacion_id	No	BTREE	asignacion_id	1	A	30
calificaciones	idx_calificaciones_est_asig	No	BTREE	estudiante_id	1	A	199
calificaciones	idx_calificaciones_est_asig	No	BTREE	asignacion_id	2	A	199

11. RENDIMIENTO-ANTES

```
SELECT  
    e.estudiante_id,  
    CONCAT(e.nombres, ' ', e.apellidos) AS estudiante,  
    p.nombre_paralelo,  
    m.nombre AS materia,  
    c.trimestre,  
    c.nota  
FROM calificaciones c  
JOIN estudiantes e ON e.estudiante_id = c.estudiante_id  
JOIN asignacion_materias am ON am.asignacion_id = c.asignacion_id  
JOIN materias m ON m.materia_id = am.materia_id  
JOIN paralelos p ON p.paralelo_id = am.paralelo_id  
WHERE e.nombres = 'Tina' AND e.apellidos = 'Medina'  
    AND p.nombre_paralelo = 'B' AND p.grado = 'Quinto'  
ORDER BY m.nombre, c.trimestre;
```

	estudiante_id	estudiante	nombre_paralelo	materia	trimestre	nota
▶	1	Tina Medina	B	Ciencias Naturales	1	9.00

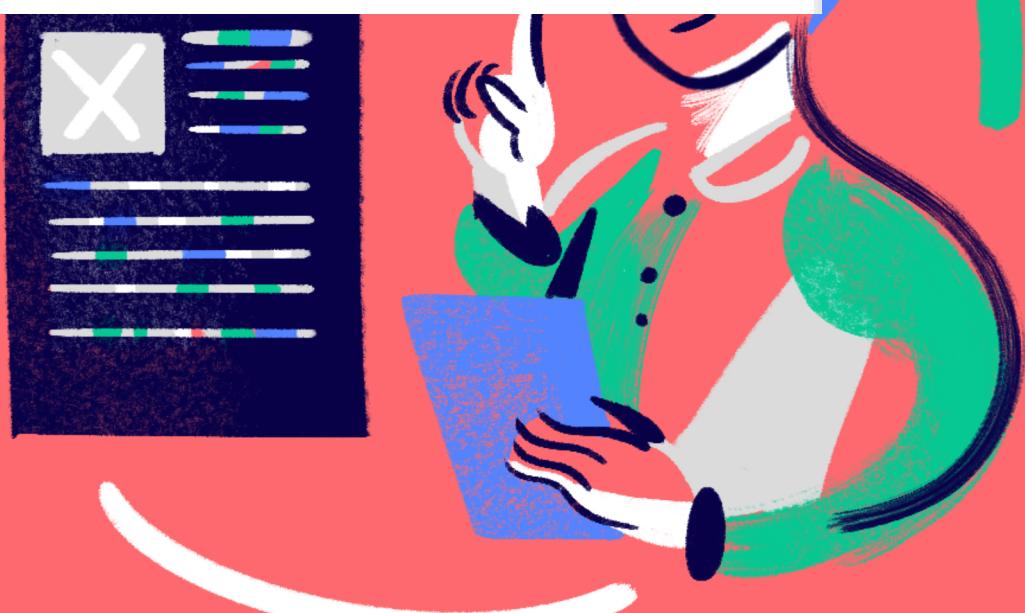


11. RENDIMIENTO-ANTES

```
24 17:01:55 SELECT e.estudiante_id, CONCAT(e.nombres, ' ', e.apellidos) AS estudiante, p.nombre_paralelo, m.nombre AS mat... 1 row(s) returned 0.015 sec / 0.000 sec
```

EXPLAIN

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	p	NULL	ALL	PRIMARY	NULL	NULL	NULL	14	7.14	Using where; Using temporary; Using
1	SIMPLE	am	NULL	ref	PRIMARY,materia_id,paralelo_id	paralelo_id	4	educacion.p.paralelo_id	2	100.00	NULL
1	SIMPLE	m	NULL	eq_ref	PRIMARY	PRIMARY	4	educacion.am.materia_id	1	100.00	NULL
1	SIMPLE	c	NULL	ref	estudiante_id,asignacion_id	asignacion_id	4	educacion.am.asignacion_id	6	100.00	NULL
1	SIMPLE	e	NULL	eq_ref	PRIMARY	PRIMARY	4	educacion.c.estudiante_id	1	5.00	Using where



11. RENDIMIENTO-DESPUES

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	e	HULL	ref	PRIMARY, idx_estudiante_apellidos	idx_estudiante_apellidos	402	const	6	10.00	Using where
1	SIMPLE	c	HULL	ref	idx_estudiante_id, idx_asignacion_id, idx_califica...	idx_estudiante_id	4	educacion.e.estudiante_id	1	100.00	HULL
1	SIMPLE	am	HULL	eq_ref	PRIMARY, materia_id, paralelo_id	PRIMARY	4	educacion.c.asignacion_id	1	100.00	HULL
1	SIMPLE	m	HULL	eq_ref	PRIMARY	PRIMARY	4	educacion.am.materia_id	1	100.00	HULL
1	SIMPLE	p	HULL	eq_ref	PRIMARY	PRIMARY	4	educacion.am.paralelo_id	1	7.14	Using where

34 17:08:40 SELECT e.estudiante_id, CONCAT(e.nombres, ' ', e.apellidos) AS estudiante, p.nombre_paralelo, m.nombre AS mat... 1 row(s) returned

0.000 sec / 0.000 sec

table: qué tabla se consulta

type: tipo de acceso (ALL es malo, ref, const, eq_ref es mejor)

key: qué índice se está usando (si no hay índice, estará en NULL)

rows: cuántas filas MySQL estima recorrer

