Proyecto Base de datos

D. Andrea Vieira Hernández
IES Alixar
Curso 2021-2022

Atentos
APOYO INTEGRAL AL ESTUDIANTE

MySQL Workbench

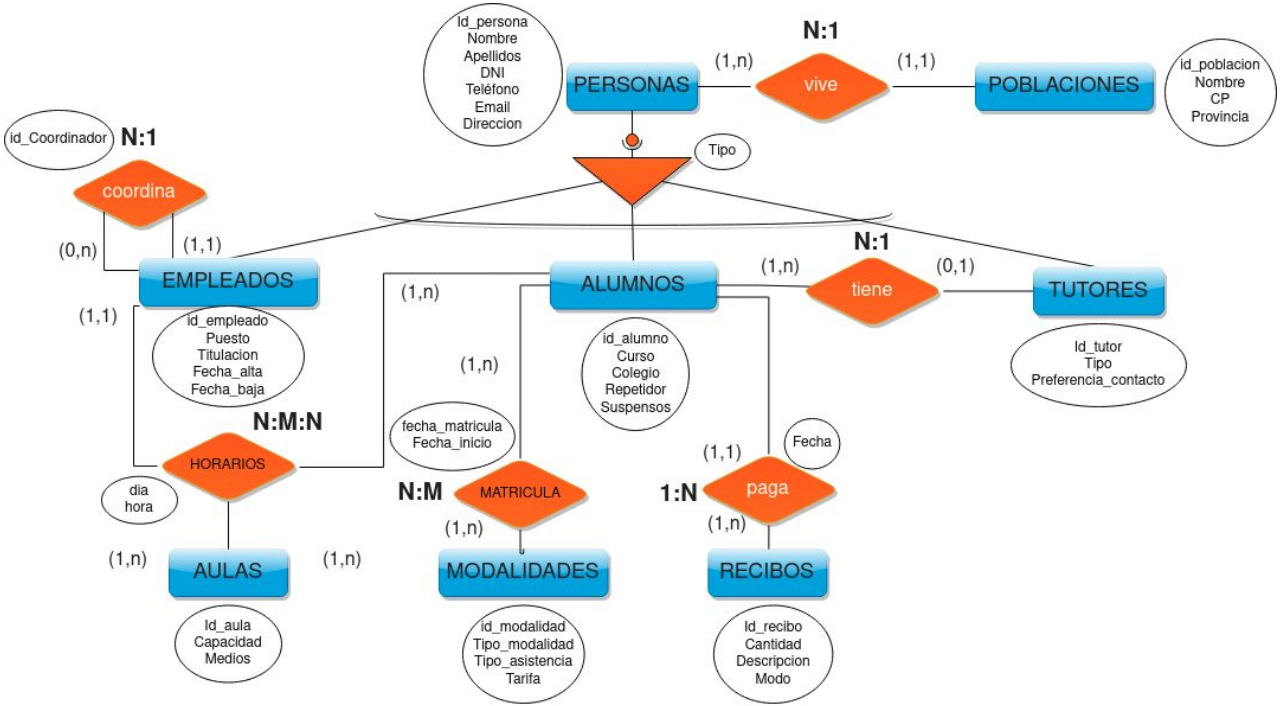
Google Sheets

draw.io

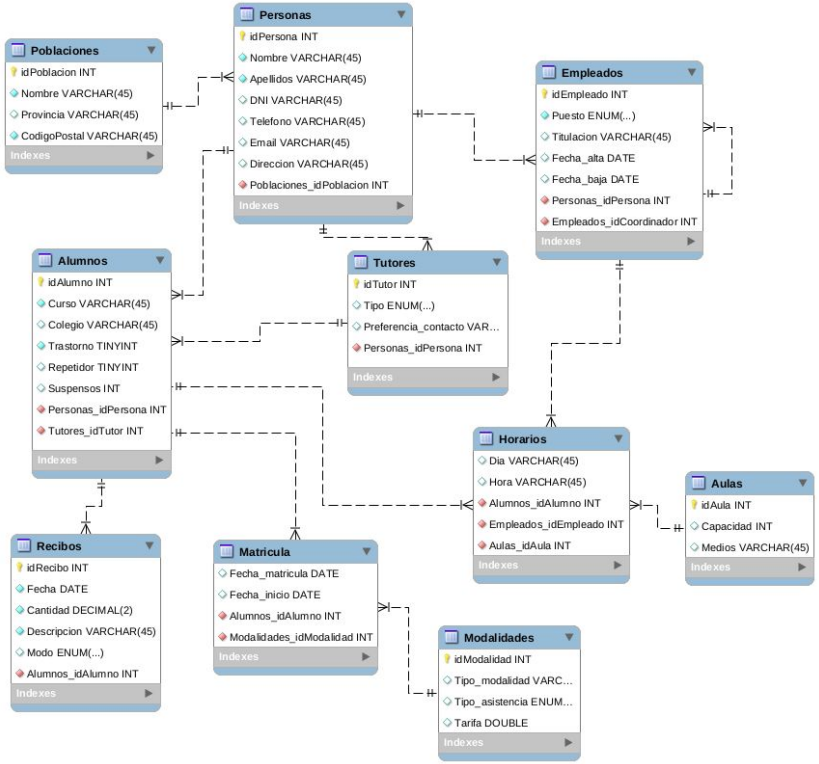
DBeaver

mockaroo

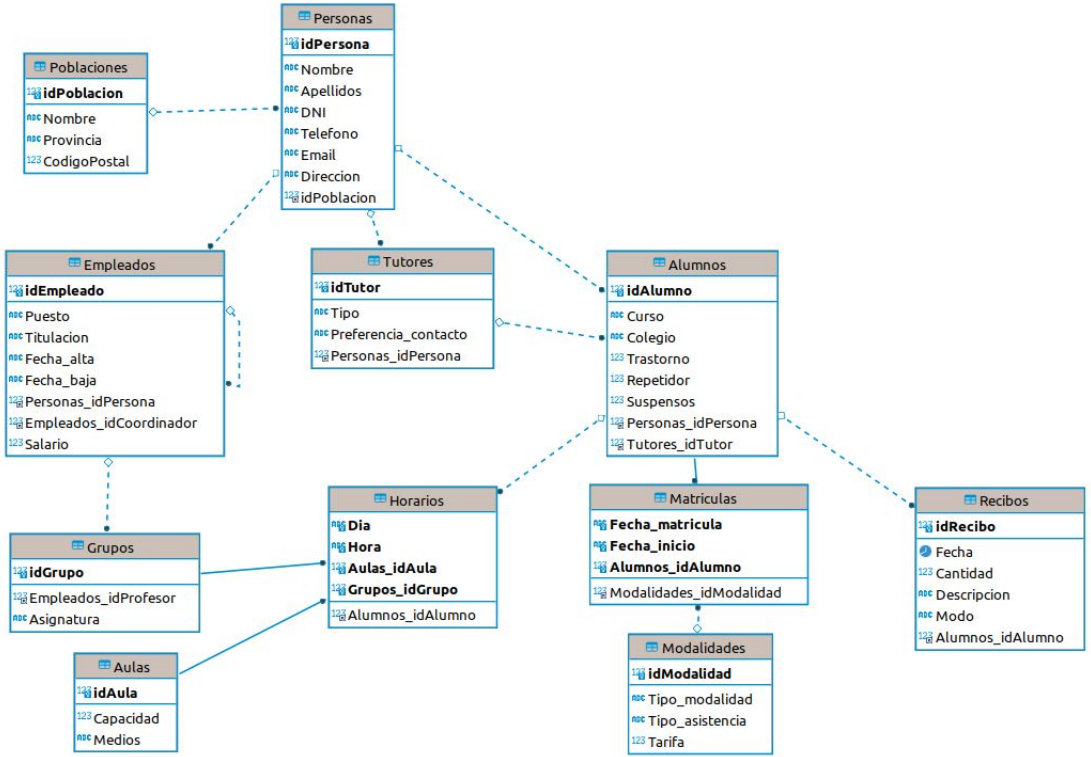
El reto…


Id_persona
Nombre
Apellidos
DNI
Teléfono
Email
Direccion
PERSONAS
N:1
(1,n)
vive
(1,1)
POBLACIONES
id_poblacion
Nombre
CP
Provincia
Tipo
id_Coordinador
N:1
coordina
(0,n)
(1,1)
EMPLEADOS
id_empleado
Puesto
Titulacion
Fecha_alta
Fecha_baja
(1,1)
N:M:N
HORARIOS
dia
hora
(1,n)
AULAS
Id_aula
Capacidad
Medios
(1,n)
(1,n)
fecha_matricula
Fecha_inicio
N:M
MATRICULA
(1,n)
MODALIDADES
id_modalidad
Tipo_modalidad
Tipo_asistencia
Tarifa
(1,n)
ALUMNOS
id_alumno
Curso
Colegio
Repetidor
Suspensos
N:1
(1,n)
tiene
(0,1)
TUTORES
Id_tutor
Tipo
Preferencia_contacto
(1,1)
Fecha
1:N
paga
(1,n)
RECIBOS
Id_recibo
Cantidad
Descripcion
Modo

El reto…


Poblaciones
idPoblacion INT
Nombre VARCHAR(45)
Provincia VARCHAR(45)
CodigoPostal VARCHAR(45)
Indexes
Personas
idPersona INT
Nombre VARCHAR(45)
Apellidos VARCHAR(45)
DNI VARCHAR(45)
Telefono VARCHAR(45)
Email VARCHAR(45)
Direccion VARCHAR(45)
Poblaciones_idPoblacion INT
Indexes
Empleados
idEmpleado INT
Puesto ENUM(...)
Titulacion VARCHAR(45)
Fecha_alta DATE
Fecha_baja DATE
Personas_idPersona INT
Empleados_idCoordinador INT
Indexes
Alumnos
idAlumno INT
Curso VARCHAR(45)
Colegio VARCHAR(45)
Trastorno TINYINT
Repetidor TINYINT
Suspensos INT
Personas_idPersona INT
Tutores_idTutor INT
Indexes
Tutores
idTutor INT
Tipo ENUM(...)
Preferencia_contacto VAR...
Personas_idPersona INT
Indexes
Horarios
Dia VARCHAR(45)
Hora VARCHAR(45)
Alumnos_idAlumno INT
Empleados_idEmpleado INT
Aulas_idAula INT
Indexes
Aulas
idAula INT
Capacidad INT
Medios VARCHAR(45)
Indexes
Recibos
idRecibo INT
Fecha DATE
Cantidad DECIMAL(2)
Descripcion VARCHAR(45)
Modo ENUM(...)
Alumnos_idAlumno INT
Indexes
Matricula
Fecha_matricula DATE
Fecha_inicio DATE
Alumnos_idAlumno INT
Modalidades_idModalidad INT
Indexes
Modalidades
idModalidad INT
Tipo_modalidad VARC...
Tipo_asistencia ENUM...
Tarifa DOUBLE
Indexes

La puesta en marcha



Consultas:

El empleado que sea coordinador de todos, porque es el dueño de la academia, debe aparecer toda la información en una sola columna, donde se recoja el nombre completo, la ciudad y provincia donde vive y cuánto cobra.

```
SELECT DISTINCT CONCAT_WS (' ', p.Nombre, p.Apellidos, ', que vive en', p2.Nombre,'-', p2.Provincia, ', es el dueño de Atentos y cobra',
                           e.Salario, '€') 'Datos del propietario'
FROM Personas p INNER JOIN Empleados e INNER JOIN Empleados e2 INNER JOIN Poblaciones p2
    ON p.idPersona = e.Personas_idPersona
    AND e.idEmpleado = e2.Empleados_idCoordinador
    AND p2.idPoblacion = p.idPoblacion
WHERE e.Empleados_idCoordinador is NULL;
```

Results 1 ×

SELECT DISTINCT CONCAT_WS (', p.Nombre, Enter a SQL expression to filter results (use Ctrl+Space)

	Datos del propietario
1	Johnnie Overbury , que vive en Villanueva del Trabuco - Málaga , es el dueño de Atentos y cobra 3500 €

Consultas:

Los alumnos de Castilleja de la Cuesta matriculados en Apoyo Escolar que asisten de manera presencial.

```sql
SELECT p.Nombre, p.Apellidos
FROM Personas p INNER JOIN Alumnos a INNER JOIN Matriculas m INNER JOIN Modalidades m2 INNER JOIN Poblaciones p2
    ON p.idPersona = a.Personas_idPersona
    AND a.idAlumno = m.Alumnos_idAlumno
    AND m.Modalidades_idModalidad = m2.idModalidad
    AND p.idPoblacion = p2.idPoblacion
WHERE m2.Tipo_modalidad = 'Apoyo escolar'
AND m2.Tipo_asistencia = 'Presencial'
AND p2.Nombre = 'Castilleja de la Cuesta'
ORDER BY p.Nombre, p.Apellidos;
```

Personas 1 ×

SELECT p.Nombre, p.Apellidos FROM Persona Enter a SQL expression to filter results (use Ctrl+Space)

	Nombre	Apellidos
1	Alane	Clatworthy
2	Alisun	Cullin
3	Alvy	Romeo
4	Angus	Humbee
5	Aviva	Shearer
6	Batholomew	McFadyen
7	Blake	Downie
8	Carl	Hourican

Consultas:

Los alumnos que tienen clase los lunes a las 20:00 en cualquier aula y de cualquier asignatura.

```
SELECT CONCAT_WS(' ',p.Nombre, p.Apellidos) Alumno, g.Asignatura,  CONCAT('Aula: ', a2.idAula) Aula
FROM Personas p INNER JOIN Alumnos a INNER JOIN Horarios h INNER JOIN Aulas a2 INNER JOIN Grupos g
    ON p.idPersona = a.Personas_idPersona
    AND h.Alumnos_idAlumno = a.idAlumno
    AND h.Aulas_idAula = a2.idAula
    AND h.Grupos_idGrupo = g.idGrupo
WHERE h.Dia = 'Lunes' AND h.Hora = '20:00'
ORDER BY a2.idAula;
```

Grupos 1 ×

SELECT CONCAT_WS(' ',p.Nombre, p.Apell Enter a SQL expression to filter results (use Ctrl+Space)

	Alumno	Asignatura	Aula
6	Lothario Filan	Geología	Aula: 41
7	Lamar Sambals	Naturales	Aula: 42
8	Erhart Aldred	Música	Aula: 46
9	Clotilda Dombrell	Música	Aula: 52
10	Sutherlan Pechell	Eduacación física	Aula: 53
11	Ernaline Paslow	Biología	Aula: 62
12	Tabbatha Abrahms	Latín	Aula: 64
13	Dyna Sandeson	Filosofía	Aula: 75

Consultas:

Poblaciones de Sevilla con más de una persona relacionada con la academia.

```
SELECT p2.Nombre, COUNT(*) Cantidad
FROM Personas p INNER JOIN Poblaciones p2
    ON p.IdPoblacion = p2.IdPoblacion
WHERE p2.Provincia = 'Sevilla'
GROUP BY p2.Nombre
HAVING COUNT(Cantidad) > 1;
```

ersonas 1 | Poblaciones 3 | Poblaciones 3 ×

ELECT p2.Nombre, COUNT(*) Cantidad FI | Enter a SQL e.

	Nombre	Cantidad
1	Bormujos	65
2	Castilleja de la Cuesta	105
3	Tomares	281

Consultas:

Nombre y apellidos de los alumnos en una misma columna que pagaron más que la media durante el año 2020 en Tomares. Se debe indicar la cantidad pagada y la fecha del recibo.

```
SELECT CONCAT_WS(' ',p.Nombre, p.Apellidos) 'Alumno', DATE_FORMAT(r.Fecha, "%d %M %Y") 'Fecha de Pago', CONCAT(r.Cantidad, '€') Cantidad
FROM Personas p INNER JOIN Alumnos a INNER JOIN Recibos r INNER JOIN Poblaciones p2
                ON p.idPersona = a.Personas_idPersona
                AND r.Alumnos_idAlumno = a.idAlumno
                AND p2.idPoblacion = p.idPoblacion
WHERE r.Cantidad > (SELECT AVG(r2.Cantidad) FROM Recibos r2)
                   AND YEAR(r.Fecha) = 2020
                   AND p2.Nombre = 'Tomares'
ORDER BY p.Nombre;
```

Results 1 ×

SELECT CONCAT_WS('',p.Nombre, p | Enter a SQL expression to filter results (use Ctrl+Space)

	Alumno	Fecha de Pago	Cantidad
4	Arlena Calwell	02 May 2020	199€
5	Arlena Calwell	24 November 2020	293€
6	Art Iwaszkiewicz	07 September 2020	297€
7	Art Iwaszkiewicz	22 June 2020	225€
8	Basilio Yewdall	12 October 2020	221€
9	Basilio Langelay	10 April 2020	228€
10	Berte Sawl	12 November 2020	327€
11	Clarine Edie	10 May 2020	266€
12	Danyelle McKeemar	14 June 2020	267€
13	Darell Yanele	16 March 2020	206€
14	Denice Minot	03 April 2020	201€
15	Ernaline Paslow	25 January 2020	204€
16	Franklin Portwain	05 August 2020	295€

Consultas:

El profesor que haya estado en activo durante más tiempo, pero que ya no trabaje en la academia.

```
SELECT CONCAT_WS (' ', p.Nombre, p.Apellidos) Profesor, e.Fecha_alta , e.Fecha_baja, DATEDIFF(e.Fecha_baja, e.Fecha_alta) Dias_trabajados
FROM Personas p INNER JOIN Empleados e
    ON p.idPersona = e.Personas_idPersona
WHERE e.Puesto = 'Profesor'
    AND DATEDIFF(e.Fecha_alta, e.Fecha_baja) = (SELECT MIN(DATEDIFF(e2.Fecha_alta, e2.Fecha_baja))
                                                FROM  Empleados e2
                                                WHERE e2.Puesto = 'Profesor');
```

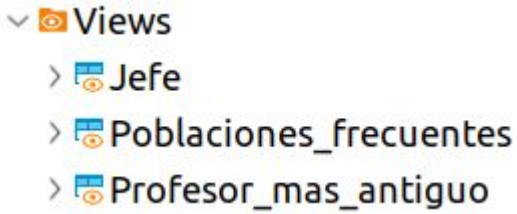
Empleados 1 ×

SELECT CONCAT_WS (' ', p.Nombre, p. Enter a SQL expression to filter results (use Ctrl+Space)

	Profesor	Fecha_alta	Fecha_baja	Dias_trabajados
1	Tiertza Tillett	2011-07-07	2020-01-06	3,105

Vistas:

```
CREATE VIEW Jefe AS (SELECT DISTINCT CONCAT_WS (' ', p.Nombre, p.Apellidos, ', que vive en', p2.Nombre,'-', p2.Provincia, ', es el dueño de Aten
                            e.Salario, '€') 'Datos del propietario'
FROM Personas p INNER JOIN Empleados e INNER JOIN Empleados e2 INNER JOIN Poblaciones p2
    ON p.idPersona = e.Personas_idPersona
    AND e.idEmpleado = e2.Empleados_idCoordinador
    AND p2.idPoblacion = p.idPoblacion
WHERE e.Empleados_idCoordinador is NULL);
```

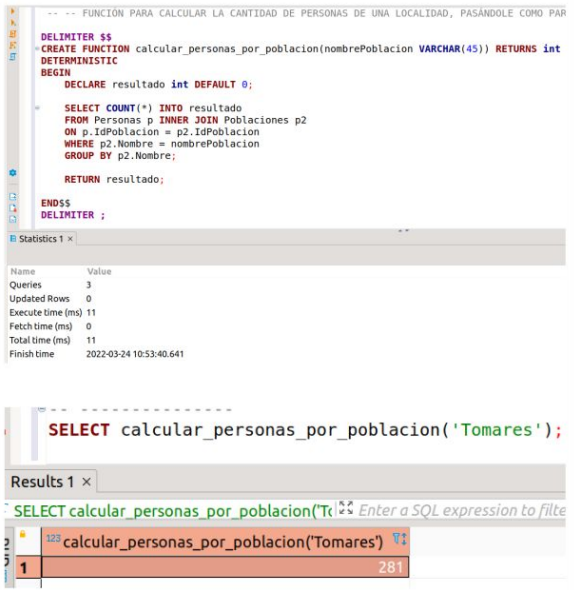



```
CREATE VIEW Poblaciones_frecuentes AS (SELECT p2.Nombre, COUNT(*) Cantidad
                                       FROM Personas p INNER JOIN Poblaciones p2
                                           ON p.IdPoblacion = p2.IdPoblacion
                                       WHERE p2.Provincia = 'Sevilla'
                                       GROUP BY p2.Nombre
                                       HAVING COUNT(Cantidad) > 1);
```

```
CREATE VIEW Profesor_mas_antiguo AS (SELECT CONCAT_WS (' ', p.Nombre, p.Apellidos) Profesor, e.Fecha_alta , e.Fecha_baja, DATEDIFF(e.Fecha_baja,
                                     FROM Personas p INNER JOIN Empleados e
                                         ON p.idPersona = e.Personas_idPersona
                                     WHERE e.Puesto = 'Profesor'
                                     AND DATEDIFF(e.Fecha_alta, e.Fecha_baja) = (SELECT MIN(DATEDIFF(e2.Fecha_alta, e2.Fecha_baja))
                                                                                 FROM  Empleados e2
                                                                                 WHERE e2.Puesto = 'Profesor'));
```

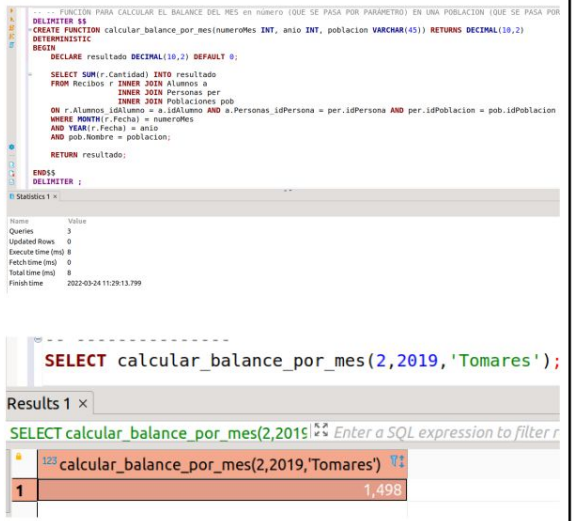
Funciones:

Se crea una función para calcular la cantidad de personas relacionadas con la academia cuya dirección pertenezca a una localidad,
pasándole como parámetro el nombre de dicha población.

Consulta	Captura
DELIMITER $$ CREATE FUNCTION calcular_personas_por_poblacion(nombrePoblacion VARCHAR(45)) RETURNS int DETERMINISTIC BEGIN DECLARE resultado int DEFAULT 0; SELECT COUNT(*) INTO resultado FROM Personas p INNER JOIN Poblaciones p2 ON p.IdPoblacion = p2.IdPoblacion WHERE p2.Nombre = nombrePoblacion GROUP BY p2.Nombre; RETURN resultado; END$$ DELIMITER ; SELECT calcular_personas_por_poblacion('Tomares');	

Funciones:

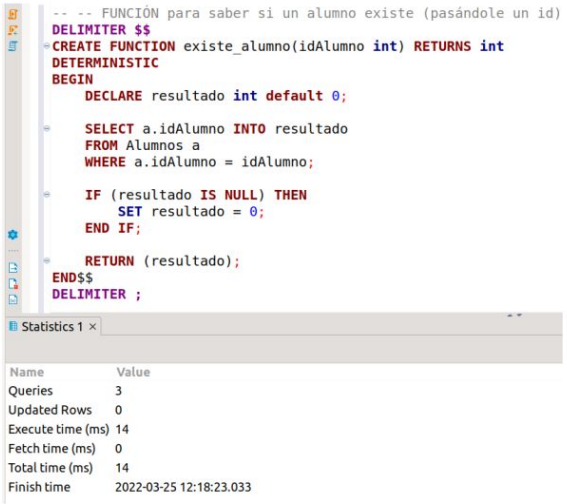
Se crea un función para calcular el balance del mes (que se pasa su número por parámetro) con los usuarios de una población (que se pasa por parámetro).

Consulta	Captura
DELIMITER $$ CREATE FUNCTION calcular_balance_por_mes(numeroMes INT, anio INT, poblacion VARCHAR(45)) RETURNS DECIMAL(10,2) DETERMINISTIC BEGIN DECLARE resultado DECIMAL(10,2) DEFAULT 0; SELECT SUM(r.Cantidad) INTO resultado FROM Recibos r INNER JOIN Alumnos a INNER JOIN Personas per INNER JOIN Poblaciones pob ON r.Alumnos_idAlumno = a.idAlumno AND a.Personas_idPersona = per.idPersona AND per.idPoblacion = pob.idPoblacion WHERE MONTH(r.Fecha) = numeroMes AND YEAR(r.Fecha) = anio AND pob.Nombre = poblacion; RETURN resultado; END$$ DELIMITER ; SELECT calcular_balance_por_mes(2,2019,'Tomares');	

Funciones:

Se crea una función para comprobar si existe un alumno en los registros. Esta función será tremendamente útil en procedimientos, pues nos ayuda a capturar posibles excepciones.

Consulta	Captura

DELIMITER $$
CREATE FUNCTION existe_alumno(idAlumno int) RETURNS int
DETERMINISTIC
BEGIN
 	
DECLARE resultado int default 0;
 	
SELECT a.idAlumno INTO resultado
FROM Alumnos a
WHERE a.idAlumno = idAlumno;
 	
IF (resultado IS NULL) THEN
SET resultado = 0;
END IF;
 	
RETURN (resultado);
 	
END$$
DELIMITER ;
 	
SELECT existe_alumno(5);
	

Procedimientos:

Genera un pequeño informe sobre las inscripciones que haya podido realizar un alumno en el centro, de manera que se pueda evaluar el cambio de curso, las tarificaciones y las distintas fechas de inicio de curso, además controla la excepción en el caso que se introduzca un código de alumno que no exista.

Consulta	Captura	

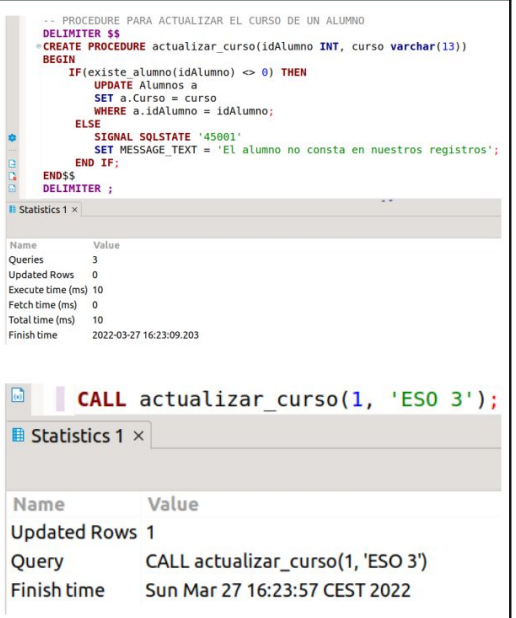
```
DELIMITER $$
CREATE PROCEDURE informe_alumno(IN idAlumno INT)
BEGIN
    IF(existe_alumno(idAlumno) <> 0) THEN
        SELECT CONCAT('+++++++++ INFORME ++++++++++', '\n',
'\n', 'Alumno: ',p.Nombre, ' ', p.Apellidos, '\n', '----------', '\n','Colegio:
', a.Colegio, ' Curso: ', a.Curso, '\n', '----------', '\n','Modalidad: ',
m2.Tipo_modalidad, ' Asistencia: ', m2.Tipo_asistencia, '\n', '----------',
'\n', 'Fecha de inicio: ', m.Fecha_inicio , ' Tarifa: ', m2.Tarifa, '€','\n',
'++++++++++++++++++++++++++')
        FROM Personas p INNER JOIN Alumnos a
            INNER JOIN Matriculas m
            INNER JOIN Modalidades m2
            ON p.idPersona = a.Personas_idPersona
            AND a.idAlumno = m.Alumnos_idAlumno
            AND m.Modalidades_idModalidad = m2.idModalidad
        WHERE a.idAlumno = idAlumno
        ORDER BY p.Apellidos, p.Nombre, m.Fecha_inicio;
    ELSE
        SIGNAL SQLSTATE '45001'
        SET MESSAGE_TEXT = 'El alumno no consta en nuestros
registros';
    END IF;
END $$
DELIMITER ;
```


```
"+++++++++ INFORME ++++++++++

Alumno: Morgen Aitken
----------
Colegio: Murainagrass School Curso: ESO
----------
Modalidad: Asignatura específica Asisten
----------
Fecha de inicio: 2010-05-05 Tarifa: 174€
 ++++++++++++++++++++++++++"
"+++++++++ INFORME ++++++++++

Alumno: Morgen Aitken
----------
Colegio: Murainagrass School Curso: ESO
----------
Modalidad: Apoyo escolar Asistencia: Onl
----------
Fecha de inicio: 2017-04-06 Tarifa: 138€
 ++++++++++++++++++++++++++"
```

Procedimientos:

Actualiza el curso de los alumnos solo si el alumno existe, además controla la excepción en el caso que se introduzca un código de alumno que no exista.

Consulta	Captura

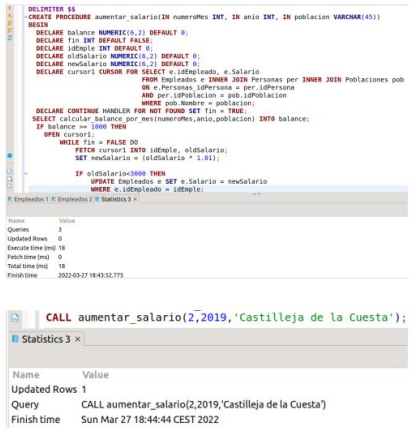
DELIMITER $$
CREATE PROCEDURE actualizar_curso(idAlumno INT, curso varchar(13))
BEGIN
IF(existe_alumno(idAlumno) <> 0) THEN
UPDATE Alumnos a
SET a.Curso = curso
WHERE a.idAlumno = idAlumno;
ELSE
SIGNAL SQLSTATE '45001'
SET MESSAGE_TEXT = 'El alumno no consta en nuestros registros';
END IF;
END$$
DELIMITER ;
 	
CALL actualizar_curso(1, 'ESO 3');
	

Procedimientos:

Procedimiento con cursor: evalúa si en una localidad en concreto, un mes ha tenido un balance superior a 1.000€, a todos los empleados que sean de ese lugar, se les aumentará el salario un 1% siempre y cuando su salario no sea superior a 3.000€.

Consulta	Captura

```
DELIMITER $$
CREATE PROCEDURE aumentar_salario(IN numeroMes INT, IN anio
INT, IN poblacion VARCHAR(45))
BEGIN
  DECLARE balance NUMERIC(6,2) DEFAULT 0;
  DECLARE fin INT DEFAULT FALSE;
  DECLARE idEmple INT DEFAULT 0;
  DECLARE oldSalario NUMERIC(6,2) DEFAULT 0;
  DECLARE newSalario NUMERIC(6,2) DEFAULT 0;
  DECLARE cursor1 CURSOR FOR
        SELECT e.idEmpleado, e.Salario
        FROM Empleados e INNER JOIN Personas per
          INNER JOIN Poblaciones pob
          ON e.Personas_idPersona = per.idPersona
          AND per.idPoblacion = pob.idPoblacion
        WHERE pob.Nombre = poblacion;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET fin = TRUE;

  SELECT
 calcular_balance_por_mes(numeroMes,anio,poblacion)
  INTO balance;

  IF balance >= 1000 THEN
    OPEN cursor1;
      WHILE fin = FALSE DO
        FETCH cursor1 INTO idEmple, oldSalario;
        SET newSalario = (oldSalario * 1.01);
        IF oldSalario<3000 THEN
          UPDATE Empleados e
          SET e.Salario = newSalario
          WHERE e.idEmpleado = idEmple;
        ELSE
          SIGNAL SQLSTATE '45003'
          SET MESSAGE_TEXT = 'No se puede aumentar el salario';
        END IF;
      END WHILE;
    CLOSE cursor1;
  ELSE
    SIGNAL SQLSTATE '45003'
    SET MESSAGE_TEXT = 'No se puede aumentar el salario';
  END IF;
END $$
DELIMITER ;

CALL aumentar_salario(2,2019,'Castilleja de la Cuesta');
```

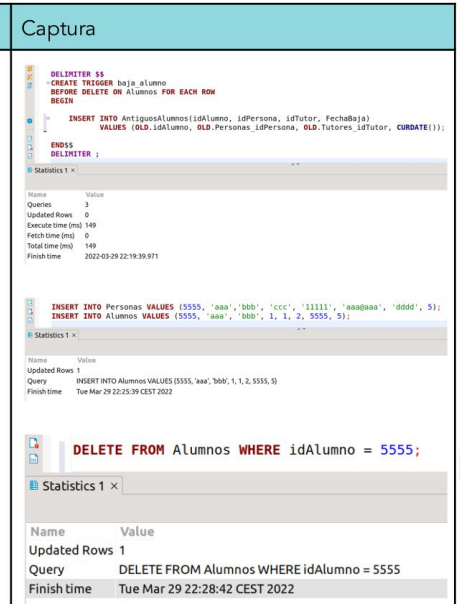


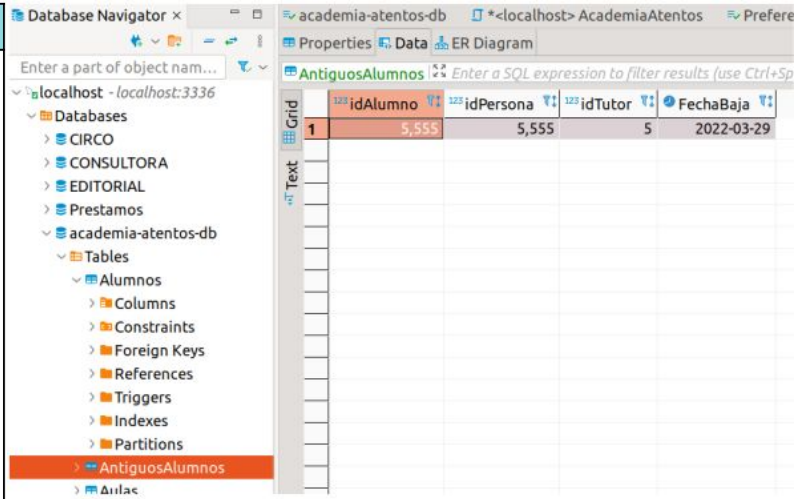
Disparadores:

Se inserta en una tabla (AntiguosAlumnos) que funciona a modo de almacén, donde se guardan los datos fundamentales de los alumnos, cuando estos se eliminan de la tabla Alumnos.

Consulta	Captura

```
CREATE TABLE `academia-atentos-db`.AntiguosAlumnos (
    idAlumno INTEGER NULL,
    idPersona INTEGER NULL,
    idTutor INTEGER NULL,
    FechaBaja DATE NULL);

DELIMITER $$
CREATE TRIGGER baja_alumno
BEFORE DELETE ON Alumnos FOR EACH ROW
BEGIN
    INSERT INTO AntiguosAlumnos(idAlumno, idPersona, idTutor,
FechaBaja)
    VALUES (OLD.idAlumno, OLD.Personas_idPersona,
OLD.Tutores_idTutor, CURDATE());
END$$
DELIMITER ;

INSERT INTO Personas VALUES (5555, 'aaa','bbb', 'ccc', '11111',
'aaa@aaa', 'dddd', 5);
INSERT INTO Alumnos VALUES (5555, 'aaa', 'bbb', 1, 1, 2, 5555, 5);

DELETE FROM Alumnos WHERE idAlumno = 5555;
```



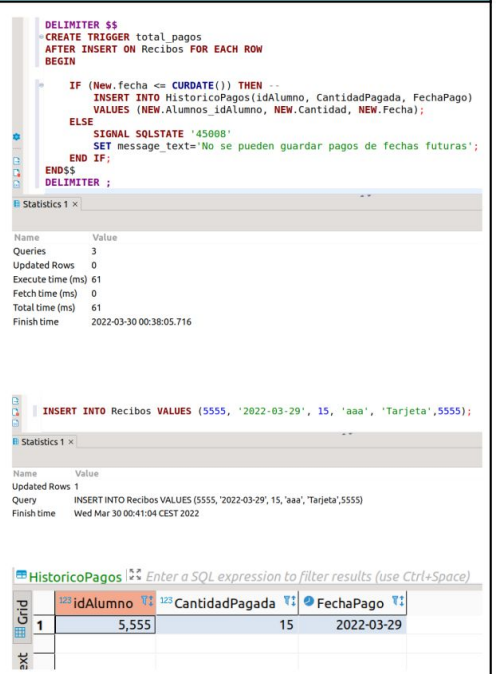


Disparadores:

Se dispara una inserción en una tabla de seguridad para el histórico de pagos, con los datos fundamentales del alumno y del pago. Añade una excepción si se añade una fecha posterior a la fecha actual.

Consulta	Captura

```
CREATE TABLE `academia-atentos-db`.HistoricoPagos (
    idAlumno INTEGER NULL,
    CantidadPagada DECIMAL(10,2) NULL,
    FechaPago DATE);

DELIMITER $$
CREATE TRIGGER total_pagos
AFTER INSERT ON Recibos FOR EACH ROW
BEGIN
  IF (New.fecha <= CURDATE()) THEN
    INSERT INTO HistoricoPagos(idAlumno, CantidadPagada,
FechaPago)
    VALUES (NEW.Alumnos_idAlumno, NEW.Cantidad, NEW.Fecha);
  ELSE
    SIGNAL SQLSTATE '45008'
    SET message_text='No se pueden guardar pagos de fechas
futuras';
  END IF;

END$$
DELIMITER ;

INSERT INTO Recibos VALUES (5555, '2022-03-29', 15, 'aaa',
'Tarjeta',5555);
```



Conclusiones y valoración personal