# Ejercicios prácticos

|  |
| --- |
| **Asignatura**: Bases de datos |
| **Nombre**: Actividad práctica 2 |
| **Unidades**: 4 y 5 |

## Guía

* Se plantean diferentes problemas que tratan aspectos de las unidades 4 y 5.
* Se tiene que trabajar sobre un SGBD de MySQL
* Hay que emplear lenguaje SQL.
* Aquellos ejercicios que se pida realizar una consulta se tienen que mostrar la sentencia empleada y una captura con los resultados obtenidos.
* Se debe entregar este mismo documento en Word o PDF con las soluciones de cada ejercicio.

## Ejemplo

A partir de la siguiente tabla y sus datos.

|  |
| --- |
| CREATE TABLE contactos (  id INT AUTO\_INCREMENT PRIMARY KEY,  nombre VARCHAR(255),  apellido1 VARCHAR(255),  apellido2 VARCHAR(255),  telefono VARCHAR(10),  email VARCHAR(255)  );  INSERT INTO contactos (nombre, apellido1, apellido2, telefono, email)  VALUES ('Antonio', 'Gonzalez', 'García', '700800900', 'ant@email.com'),  ('Laura', 'Miró', 'Carreño', '600600600', 'laura@email.com'),  ('Carolina', 'Miras', 'Miras', '666677685', 'caro@email.com'),  ('Raúl', 'Miranda', 'Gomez', '808606606', 'ramigo@email.com'); |

Obtener todos los datos de los contactos en el que su número de teléfono empiece por 6.

Sentencia SQL.

|  |
| --- |
| SELECT \*  FROM personas  WHERE telefono LIKE '6%'; |

Captura con los resultados.

Tabla

Descripción generada automáticamente

## Enunciado de los ejercicios

## Ejercicio 1.

Utilizando un SGDB de MySQL y el cliente que quieras. Cargar la siguiente estructura de la base de datos de employees .

|  |
| --- |
| DROP DATABASE IF EXISTS employees;  CREATE DATABASE IF NOT EXISTS employees;  USE employees;  CREATE TABLE employees (  emp\_no INT NOT NULL,  birth\_date DATE NOT NULL,  first\_name VARCHAR(14) NOT NULL,  last\_name VARCHAR(16) NOT NULL,  gender ENUM ('M','F') NOT NULL,  hire\_date DATE NOT NULL,  PRIMARY KEY (emp\_no)  );  CREATE TABLE departments (  dept\_no CHAR(5) NOT NULL,  dept\_name VARCHAR(40) NOT NULL,  PRIMARY KEY (dept\_no),  UNIQUE KEY (dept\_name)  );  CREATE TABLE dept\_manager (  emp\_no INT NOT NULL,  dept\_no CHAR(5) NOT NULL,  from\_date DATE NOT NULL,  to\_date DATE NOT NULL,  FOREIGN KEY (emp\_no) REFERENCES employees (emp\_no) ON DELETE CASCADE,  FOREIGN KEY (dept\_no) REFERENCES departments (dept\_no) ON DELETE CASCADE,  PRIMARY KEY (emp\_no,dept\_no)  );  CREATE TABLE dept\_emp (  emp\_no INT NOT NULL,  dept\_no CHAR(5) NOT NULL,  from\_date DATE NOT NULL,  to\_date DATE NOT NULL,  FOREIGN KEY (emp\_no) REFERENCES employees (emp\_no) ON DELETE CASCADE,  FOREIGN KEY (dept\_no) REFERENCES departments (dept\_no) ON DELETE CASCADE,  PRIMARY KEY (emp\_no,dept\_no)  );  CREATE TABLE titles (  emp\_no INT NOT NULL,  title VARCHAR(50) NOT NULL,  from\_date DATE NOT NULL,  to\_date DATE,  FOREIGN KEY (emp\_no) REFERENCES employees (emp\_no) ON DELETE CASCADE,  PRIMARY KEY (emp\_no,title, from\_date)  );  CREATE TABLE salaries (  emp\_no INT NOT NULL,  salary INT NOT NULL,  from\_date DATE NOT NULL,  to\_date DATE NOT NULL,  FOREIGN KEY (emp\_no) REFERENCES employees (emp\_no) ON DELETE CASCADE,  PRIMARY KEY (emp\_no, from\_date)  ); |

A continuación, Haz la carga de datos con las sentencias que tienes a disposición:

|  |
| --- |
| INSERT INTO employees (emp\_no, birth\_date, first\_name, last\_name, gender, hire\_date)  VALUES (1, '1980-01-15', 'John', 'Doe', 'M', '2005-05-10'),  (2, '1985-03-22', 'Jane', 'Smith', 'F', '2007-08-15'),  (3, '1977-07-05', 'Michael', 'Johnson', 'M', '2010-02-20'),  (4, '1990-09-12', 'Emily', 'Brown', 'F', '2012-11-30'),  (5, '1982-04-30', 'William', 'Davis', 'M', '2014-06-25'),  (6, '1988-12-18', 'Olivia', 'Lee', 'F', '2016-09-05'),  (7, '1975-11-03', 'James', 'Wilson', 'M', '2018-03-12'),  (8, '1993-02-08', 'Sophia', 'Anderson', 'F', '2020-07-17'),  (9, '1984-06-21', 'Daniel', 'Martinez', 'M', '2022-01-22'),  (10, '1989-08-07', 'Ava', 'Taylor', 'F', '2023-04-09'),  (11, '1991-03-18', 'Robert', 'Jones', 'M', '2010-08-15'),  (12, '1986-05-29', 'Ella', 'Williams', 'F', '2012-02-10'),  (13, '1993-09-08', 'David', 'Moore', 'M', '2014-05-20'),  (14, '1987-12-14', 'Emma', 'Johnson', 'F', '2016-07-05'),  (15, '1990-02-25', 'Matthew', 'Smith', 'M', '2017-11-30'),  (16, '1984-06-01', 'Sophia', 'Garcia', 'F', '2019-04-15'),  (17, '1992-11-11', 'James', 'Brown', 'M', '2020-09-20'),  (18, '1989-08-03', 'Olivia', 'Taylor', 'F', '2022-01-10'),  (19, '1994-01-22', 'William', 'Anderson', 'M', '2023-05-01'),  (20, '1983-04-09', 'Ava', 'Martinez', 'F', '2023-09-10'),  (21, '1992-07-14', 'Daniel', 'Davis', 'M', '2010-04-12'),  (22, '1986-10-05', 'Mia', 'Wilson', 'F', '2012-08-22');  INSERT INTO departments (dept\_no, dept\_name)  VALUES ('HR', 'Human Resources'),  ('IT', 'Information Technology'),  ('FIN', 'Finance'),  ('SALES', 'Sales'),  ('MKTG', 'Marketing'),  ('ENG', 'Engineering'),  ('ADM', 'Administration'),  ('OPS', 'Operations'),  ('LEGAL', 'Legal'),  ('PRD', 'Production');  INSERT INTO dept\_manager (emp\_no, dept\_no, from\_date, to\_date)  VALUES (1, 'HR', '2010-01-01', '9999-01-01'),  (3, 'FIN', '2013-04-15', '9999-01-01'),  (5, 'MKTG', '2014-06-25', '9999-01-01'),  (7, 'ADM', '2018-10-01', '9999-01-01'),  (9, 'LEGAL', '2023-01-22', '9999-01-01'),  (10, 'PRD', '2023-05-10', '9999-01-01'),  (11, 'SALES', '2020-11-12', '9999-01-01'),  (12, 'OPS', '2014-06-01', '9999-01-01'),  (13, 'ENG', '2018-10-15', '9999-01-01'),  (14, 'IT', '2016-07-05', '9999-01-01');  INSERT INTO dept\_emp (emp\_no, dept\_no, from\_date, to\_date)  VALUES (1, 'HR', '2005-05-10', '9999-01-01'),  (2, 'IT', '2007-08-15', '9999-01-01'),  (3, 'FIN', '2010-02-20', '9999-01-01'),  (4, 'SALES', '2012-11-30', '2021-10-03'),  (4, 'MKTG', '2021-10-04', '9999-01-01'),  (5, 'MKTG', '2014-06-25', '9999-01-01'),  (6, 'ENG', '2016-09-05', '9999-01-01'),  (7, 'ADM', '2018-03-12', '9999-01-01'),  (8, 'OPS', '2020-07-17', '9999-01-01'),  (9, 'LEGAL', '2022-01-22', '9999-01-01'),  (10, 'LEGAL', '2023-04-09', '2023-05-09'),  (10, 'PRD', '2023-05-10', '9999-01-01'),  (11, 'SALES', '2010-08-15', '9999-01-01'),  (12, 'OPS', '2012-02-10', '9999-01-01'),  (13, 'ENG', '2014-05-20', '9999-01-01'),  (14, 'IT', '2016-07-05', '9999-01-01'),  (15, 'OPS', '2017-11-30', '9999-01-01'),  (16, 'ADM', '2019-04-15', '2022-09-01'),  (16, 'SALES', '2022-09-02', '9999-01-01'),  (17, 'HR', '2020-09-20', '9999-01-01'),  (18, 'LEGAL', '2022-01-10', '9999-01-01'),  (19, 'PRD', '2023-05-01', '9999-01-01'),  (20, 'SALES', '2023-09-10', '9999-01-01');  INSERT INTO titles (emp\_no, title, from\_date, to\_date)  VALUES (1, 'Manager', '2005-05-10', '2010-01-01'),  (1, 'Director', '2010-01-02', '9999-01-01'),  (2, 'Engineer', '2007-08-15', '2013-04-14'),  (2, 'Senior Engineer', '2013-04-15', '9999-01-01'),  (3, 'Analyst', '2010-02-20', '9999-01-01'),  (4, 'Sales Representative', '2012-11-30', '9999-01-01'),  (5, 'Marketing Coordinator', '2014-06-25', '9999-01-01'),  (6, 'Software Developer', '2016-09-05', '9999-01-01'),  (7, 'Administrator', '2018-03-12', '9999-01-01'),  (8, 'Operations Manager', '2020-07-17', '9999-01-01'),  (9, 'Legal Advisor', '2022-01-22', '9999-01-01'),  (10, 'Operations Manager', '2023-04-09', '9999-01-01'),  (11, 'Sales Representative', '2010-08-15', '9999-01-01'),  (12, 'Marketing Coordinator', '2012-02-10', '9999-01-01'),  (13, 'Software Developer', '2014-05-20', '9999-01-01'),  (14, 'Engineer', '2016-07-05', '9999-01-01'),  (15, 'Operations Manager', '2017-11-30', '9999-01-01'),  (16, 'Administrator', '2019-04-15', '9999-01-01'),  (17, 'Manager', '2020-09-20', '9999-01-01'),  (18, 'Legal Advisor', '2022-01-10', '9999-01-01'),  (19, 'Production Worker', '2023-05-01', '2023-05-24'),  (19, 'Supervisor Worker', '2023-05-25', '9999-01-01'),  (20, 'Sales Representative', '2023-09-10', '9999-01-01');  INSERT INTO salaries (emp\_no, salary, from\_date, to\_date)  VALUES (1, 80000, '2005-05-10', '2010-01-01'),  (1, 100000, '2010-01-02', '9999-01-01'),  (2, 75000, '2007-08-15', '2013-04-14'),  (2, 90000, '2013-04-15', '9999-01-01'),  (3, 60000, '2010-02-20', '9999-01-01'),  (4, 55000, '2012-11-30', '9999-01-01'),  (5, 70000, '2014-06-25', '9999-01-01'),  (6, 85000, '2016-09-05', '9999-01-01'),  (7, 65000, '2018-03-12', '9999-01-01'),  (8, 90000, '2020-07-17', '9999-01-01'),  (9, 75990, '2022-01-22', '9999-01-01'),  (10, 85000, '2023-04-09', '9999-01-01'),  (11, 60000, '2010-08-15', '9999-01-01'),  (12, 60000, '2012-02-10', '2022-02-09'),  (12, 65000, '2022-02-10', '9999-01-01'),  (13, 70000, '2014-05-20', '9999-01-01'),  (14, 75000, '2016-07-05', '9999-01-01'),  (15, 80000, '2017-11-30', '9999-01-01'),  (16, 85000, '2019-04-15', '9999-01-01'),  (17, 70000, '2020-09-20', '2021-02-20'),  (17, 90000, '2021-02-21', '9999-01-01'),  (18, 95000, '2022-01-10', '9999-01-01'),  (19, 60000, '2023-05-25', '9999-01-01'),  (20, 65000, '2023-09-10', '9999-01-01'); |

## Nos comunican que ha habido un error y se han creado dos trabajadores pero no se ha podido cargar los datos del departamento en el que pertenece, los cargos que tienen en la empresa y el salario que les corresponde. A continuación se muestra la información que deben tener:

|  |  |  |
| --- | --- | --- |
| **Número de empleado** | 21 | 22 |
| **Trabajador** | Daniel Davis | Mia Wilson |
| **Departamento** | IT | OPS |
| **Cargos** | Software Developer | Production Worker |
| **Salario** | 75000 | 55000 |
| El departamento, cargo y salario les pertenecen desde la fecha en el que fueron contratados hasta ahora. No han tenido ningún cambio. | | |

Genera las sentencias necesarias para que la base de datos tenga correctamente los datos de estos trabajadores.

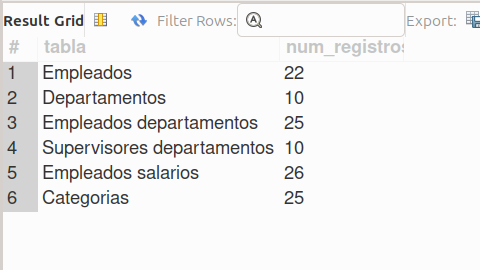
|  |
| --- |
| INSERT INTO salaries (emp\_no, salary, from\_date, to\_date)  VALUES (21, 75000, (select hire\_date from employees where emp\_no = 21), "9999-01-01"),  (22, 55000, (select hire\_date from employees where emp\_no = 22), "9999-01-01");  insert into dept\_emp (emp\_no, dept\_no, from\_date, to\_date)  values (21, "IT", (select hire\_date from employees where emp\_no = 21), "9999-01-01"),  (22, "OPS", (select hire\_date from employees where emp\_no = 22), "9999-01-01");  insert into titles (emp\_no, title, from\_date, to\_date)  values (21, "Software Developer", (select hire\_date from employees where emp\_no = 21), "9999-01-01"),  (22, "Production Worker", (select hire\_date from employees where emp\_no = 22), "9999-01-01"); |

Finalmente, ejecuta esta sentencia y pon una captura del resultado obtenido.

|  |
| --- |
| SELECT 'Empleados' AS tabla, COUNT(\*) AS num\_registros FROM employees  UNION ALL  SELECT 'Departamentos', COUNT(\*) FROM departments  UNION ALL  SELECT 'Empleados departamentos', COUNT(\*) FROM dept\_emp  UNION ALL  SELECT 'Supervisores departamentos', COUNT(\*) FROM dept\_manager  UNION ALL  SELECT 'Empleados salarios', COUNT(\*) FROM salaries  UNION ALL  SELECT 'Categorias', COUNT(\*) FROM titles; |

Comprueba que la información obtenida es igual a los resultado que aparecen en esta tabla, si no coinciden, busca donde puede estar el error y soluciónalo.

|  |  |
| --- | --- |
| **Tabla** | **Número de registros** |
| Empleados | 22 |
| Departamentos | 10 |
| Empleados departamentos | 25 |
| Supervisores departamentos | 10 |
| Empleados salarios | 26 |
| Categorías | 25 |



## Ejercicio 2.

Realizar las siguientes consultas en SQL:

1. Obtener el nombre, apellidos y fecha de nacimiento de todos los empleados que sean mujeres.

|  |
| --- |
| select emp\_no, first\_name, last\_name, birth\_date from employees where gender = "F"; |

1. Obtener toda la información de todos los departamentos que existen ordenador por su nombre de forma descendente.

|  |
| --- |
| SELECT \* FROM employees.departments  order by dept\_name desc; |

1. Obtener todos los salarios que estén entre 60000 y 80000 inclusive ordenados en forma ascendente. Los salarios no se pueden repetir y solo se deben mostrar aquellos que están activos. Se considera activo aquellos que tiene una fecha igual a 01/01/9999.

|  |
| --- |
| SELECT DISTINCT salary FROM employees.salaries  WHERE (salary BETWEEN 60000 AND 80000) AND (to\_date = "9999-01-01")  ORDER BY salary ASC; |

1. Muestra todos los datos de los empleados que tengan más de cuarenta años.

Nota. Se puede utilizar la función *YEAR(fecha)* para extraer el año de una fecha y *CURDATE()* para obtener la fecha actual.

|  |
| --- |
| SELECT \*  FROM employees.employees  WHERE (YEAR(curdate())-YEAR(birth\_date)) > 40; |

1. Obtener el número, nombre y apellido de los empleados que su nombre empiecen por la letra J, además se quiere conocer el cargo que tienen actualmente.

|  |
| --- |
| SELECT  emp.emp\_no, emp.first\_name, emp.last\_name, t.title  FROM  employees emp  JOIN  titles t ON emp.emp\_no = t.emp\_no  WHERE  emp.first\_name LIKE 'J%'; |

1. Obtener todos los datos del trabajador con código 16 y mostrar el nombre de los departamentos que ha pertenecido.

|  |
| --- |
| SELECT  employees.\*, dept\_emp.dept\_no  FROM  employees,  dept\_emp  WHERE  (employees.emp\_no = dept\_emp.emp\_no)  AND (employees.emp\_no = 16); |

1. Obtener todos los datos de los empleados que pertenecen al departamento "Administration"

|  |
| --- |
| SELECT  emp.\*, d.dept\_no  FROM  employees emp  JOIN  dept\_emp d ON emp.emp\_no = d.emp\_no  WHERE  d.dept\_no = 'ADM'; |

1. Obtener todos los datos de los trabajadores y el departamento en el que están asignados actualmente.

|  |
| --- |
| SELECT  emp.\*, d.dept\_no  FROM  employees emp  JOIN  dept\_manager d ON emp.emp\_no = d.emp\_no  WHERE  d.to\_date > YEAR(CURDATE()); |

1. Obtener todos los datos de los trabajadores e indicar si son supervisores de un departamento o no. No hay que mostrar el departamento en el que son supervisores.

Nota. Se puede utilizar la sentencia CASE para mostrar un resultado según una condición. La estructura es: CASE WHEN condición THEN resultado ELSE resultado contrario.

|  |
| --- |
| SELECT  t.\*,  CASE  WHEN  EXISTS( SELECT  1  FROM  dept\_manager m  WHERE  t.emp\_no = m.emp\_no)  THEN  'Supervisor'  ELSE 'No Supervisor'  END AS es\_supervisor  FROM  employees t; |

1. Obtener el nombre y apellidos de aquellos trabajadores que han cambiado alguna vez de departamento. Es necesario que aparezca el nombre del departamento anterior y el departamento nuevo. Mostrar los resultado ordenados alfabéticamente por el nombre y luego por los apellidos de los trabajadores.

|  |
| --- |
| SELECT  e.first\_name,  e.last\_name,  old\_dept.dept\_name AS departamento\_anterior,  new\_dept.dept\_name AS departamento\_nuevo  FROM  employees e  JOIN  dept\_emp old\_dept\_emp ON e.emp\_no = old\_dept\_emp.emp\_no  AND old\_dept\_emp.to\_date = (SELECT  MAX(to\_date)  FROM  dept\_emp  WHERE  emp\_no = e.emp\_no  AND to\_date < CURDATE())  JOIN  departments old\_dept ON old\_dept\_emp.dept\_no = old\_dept.dept\_no  JOIN  dept\_emp new\_dept\_emp ON e.emp\_no = new\_dept\_emp.emp\_no  AND new\_dept\_emp.from\_date = (SELECT  MIN(from\_date)  FROM  dept\_emp  WHERE  emp\_no = e.emp\_no  AND from\_date > old\_dept\_emp.to\_date)  JOIN  departments new\_dept ON new\_dept\_emp.dept\_no = new\_dept.dept\_no  ORDER BY e.first\_name , e.last\_name; |

## Ejercicio 3.

Ahora se plantean otro tipo de consultas para sacar estadísticas y conjuntos de resultados. A continuación se indican los enunciados:

1. Obtener cuantos trabajadores son hombres y cuantos son mujeres.

|  |
| --- |
| SELECT  (SELECT  COUNT(\*)  FROM  employees  WHERE  gender = 'M') AS total\_Hombres,  (SELECT  COUNT(\*)  FROM  employees  WHERE  gender = 'F') AS total\_Mujeres; |

1. Muestra el total de trabajadores que hay actualmente en cada departamento.

|  |
| --- |
| SELECT  d.dept\_no, d.dept\_name, COUNT(emp\_no) AS total\_Trabajadores  FROM  departments d  JOIN  dept\_emp dep ON d.dept\_no = dep.dept\_no  WHERE  dep.to\_date > CURDATE()  OR dep.to\_date = '9999-01-01'  GROUP BY d.dept\_no , d.dept\_name; |

1. Obtener el total de salario que ganan los trabajadores que pertenecen actualmente en el departamento de Sales

|  |
| --- |
| SELECT  SUM(s.salary)  FROM  salaries s  JOIN  dept\_emp dep ON s.emp\_no = dep.emp\_no  WHERE  dept\_no = 'SALES'  AND (dep.to\_date > CURDATE()  OR dep.to\_date = '9999-01-01') |

1. Obtener una lista de departamentos junto con el promedio de salario de los empleados en cada departamento. Solo se tienen que mostrar aquellos departamentos cuyo promedio de salario sea superior a 60.000 al año.

|  |
| --- |
| SELECT  d.\*, AVG(s.salary) AS salario\_Promedio  FROM  departments d  JOIN  dept\_emp demp ON d.dept\_no = demp.dept\_no  JOIN  salaries s ON demp.emp\_no = s.emp\_no  WHERE  (demp.to\_date > CURDATE()  OR demp.to\_date = '9999-01-01')  GROUP BY d.dept\_no , d.dept\_name  HAVING salario\_Promedio > 60000; |

1. Obtener el nombre de los cargos de trabajo junto con el número total de empleados que tienen asignado ese cargo. Solo se deben de mostrar aquellos cargos de trabajo que tienen más de 2 empleados.

|  |
| --- |
| SELECT  t.title, COUNT(e.emp\_no) AS numero\_Trabajadores  FROM  titles t  JOIN  employees e ON t.emp\_no = e.emp\_no  WHERE  t.from\_date <= CURDATE()  AND (t.to\_date >= CURDATE()  OR t.to\_date = '9999-01-01')  GROUP BY t.title  HAVING numero\_Trabajadores > 2 |

1. Obtener una lista de los códigos de todos los departamentos que tienen al menos tres empleados y todos los departamentos que no tienen ningún empleado.

|  |
| --- |
| SELECT  dept\_no, COUNT(dept\_no) AS numero\_Empleados  FROM  dept\_emp  WHERE  from\_date <= CURDATE()  AND (to\_date >= CURDATE()  OR to\_date = '9999-01-01')  GROUP BY dept\_no  HAVING numero\_Empleados >= 3  OR numero\_Empleados = 0; |

## Ejercicio 4.

Se ha detectado que el departamento de Recursos humanos realiza constantes consultas para obtener siempre la misma información de los trabajadores:

* El nombre, apellidos y años de antigüedad en la empresa.
* El nombre del cargo que tienen actualmente.
* El nombre del departamento al que pertenecen actualmente
* Su salario actual.

Para evitar hacer varias consultas se pide crear una vista denominada informacionEmpleados que permita visualizar los datos que se han detallado. Prestar atención que no se muestra la fecha de contratación, si no, los años de antigüedad que lleva el trabajador.

|  |
| --- |
| CREATE VIEW informacionEmpleados AS  SELECT  emp.emp\_no,  emp.first\_name,  emp.last\_name,  TIMESTAMPDIFF(YEAR,  demp.from\_date,  NOW()) AS emp\_antiquity\_inYears,  t.title,  dep.dept\_name,  s.salary  FROM  employees emp  JOIN  dept\_emp demp ON emp.emp\_no = demp.emp\_no  JOIN  titles t ON emp.emp\_no = t.emp\_no  JOIN  departments dep ON demp.dept\_no = dep.dept\_no  JOIN  salaries s ON emp.emp\_no = s.emp\_no  WHERE  (((t.to\_date > NOW()  OR t.to\_date = '9999-01-01')  AND NOW() >= t.from\_date)  AND ((demp.to\_date > NOW()  OR demp.to\_date = '9999-01-01')  AND NOW() >= demp.from\_date)  AND ((s.to\_date > NOW()  OR s.to\_date = '9999-01-01')  AND NOW() >= s.from\_date)  AND (s.to\_date > NOW())); |

Por último se solicita la posibilidad de filtrar la información de acuerdo con el nombre de un empleado. Realiza una consulta utilizando la vista que has creado y los requisitos indicados. Muestra una captura del resultado obtenido buscando por los empleados que tenga el nombre ‘Daniel’.

|  |
| --- |
| SELECT  \*  FROM  informacionEmpleados  WHERE  first\_name = 'Daniel'; |

## Ejercicio 5.

Para mejorar la seguridad de nuestra base de datos, se decide que el departamento de RRHH no tenga acceso directo a las tablas de la base de dato, solo puedan visualizar la vista informacionEmpleados.

Crea un nuevo usuario en la base de datos denominado departamentoRRHH y contraseña rrhh. Aplica los permisos necesarios para que solo puedan utilizar la vista creada en el ejercicio anterior.

Describe todos los pasos que has realizado, incluyendo las sentencias necesarias y una captura donde se evidencie que no se puede visualizar las tablas de la base de datos y si muestre la vista informacionEmpleados.

|  |
| --- |
| **SENTENCIAS SQL:**  CREATE USER 'departamentoRRHH'@'localhost' IDENTIFIED BY 'rrhh';  GRANT SELECT ON employees.informacionEmpleados TO 'departamentoRRHH'@'localhost';  SHOW GRANTS FOR 'departamentoRRHH'@'localhost'; -- Verificar la correcta concesión de permisos |