

Pregunta 1. Para empezar, vamos a crear algunos usuarios y unas vistas (3 puntos):

1. Crea las siguientes cuentas de usuario. (0,5 puntos)
 - a. Usuario: alan | Contraseña: turing
 - b. Usuario: steve | Contraseña: wozniak

469

470 • `CREATE USER alan@localhost IDENTIFIED BY 'turing';`

471

472 • `CREATE USER steve@localhost IDENTIFIED BY 'wozniak';`

473

Aquí dejo la captura de los comandos que he usado para crear los usuarios.

✓	17 09:55:02	CREATE USER steve@localhost IDENTIFIED BY 'wozniak'	0 row(s) affected
✓	18 09:55:51	CREATE USER alan@localhost IDENTIFIED BY 'turing'	0 row(s) affected

Este es el resultado de ejecutar esos comandos que me devuelve la consola

2. Crea las siguientes vistas. (2 puntos)
 - a. Crea una vista llamada pobreza que muestra el nombre, apellidos y salario del empleado mejor pagado en ese momento.

```
74  -- Crea una vista del empleado mejor pagado en ese momento
75 • select e.first_name, e.last_name, max(s.salary) as salario
76  from
77      employees e
78  join
79      salaries s
80  on e.emp_no = s.emp_no
81  where s.to_date = '9999-01-01'
82  group by e.emp_no
83  having max(s.salary) = (select max(salary) from salaries);
84
```

Lo primero esta es la consulta que uso para generar después la vista

	first_name	last_name	salario
▶	Charlene	Brattka	113229

Este es el resultado que me devuelve la vista.

Ahora sólo tengo que crear el resto del código para la vista.

```

75 • create or replace view pobreza as
76 select e.first_name, e.last_name, max(s.salary) as salario
77 from
78     employees e
79 join
80     salaries s
81 on e.emp_no = s.emp_no
82 where s.to_date = '9999-01-01'
83 group by e.emp_no
84 having max(s.salary) = (select max(salary) from salaries);
85
86

```

Para comprobar este resultado lo que tengo que hacer es un select a la vista

1485


1486

1487 • select * from pobreza;

<

Result Grid

Filter Rows:

Export:  Wrap

	first_name	last_name	salario
▶	Charlene	Brattka	113229

Aquí tenemos el resultado de la operación.

- b. Crea una vista llamada RRHH que muestre el nombre y apellido de todos los empleados que actualmente trabajen en dicho departamento.

Lo primero, como en el caso anterior voy a crear la consulta que generará la vista.

```
10
11 • select e.emp_no, e.first_name, e.last_name, d.dept_name
12 from
13     employees e
14 join
15     dept_emp de
16 on e.emp_no = de.emp_no
17 join
18     departments d
19 on de.dept_no = d.dept_no
20 where de.to_date = '9999-01-01' and d.dept_name like 'Human Resources'
21 order by emp_no;
```

Esta es el código de la consulta que genero. El order by está simplemente ahí para hacer la comprobación y ver que no tengo ningún empleado repetido bajo ningún concepto, debido a que puede que un empleado haya cambiado de departamento varias veces, es decir, haya estado en este departamento, se le haya cambiado, y haya vuelto al primero.

	emp_no	first_name	last_name	dept_name
►	10005	Kyoichi	Maliniak	Human Resources
	10013	Eberhardt	Terkki	Human Resources
	10036	Adamantios	Portugali	Human Resources
	10039	Alejandro	Brender	Human Resources
	10054	Mayumi	Schueller	Human Resources
	10071	Hisao	Lipner	Human Resources
	10077	Mona	Azuma	Human Resources
	10080	Premal	Baek	Human Resources
	10086	Somnath	Foote	Human Resources
	10100	Hironobu	Haraldson	Human Resources

Si ejecuto la consulta este es el resultado. Ahora sólo tengo que terminar de formar la vista y comprobar que se ejecuta correctamente.

```

30 • create or replace view RRHH as
31 select e.emp_no, e.first_name, e.last_name, d.dept_name
32 from
33     employees e
34 join
35     dept_emp de
36 on e.emp_no = de.emp_no
37 join
38     departments d
39 on de.dept_no = d.dept_no
40 where de.to_date = '9999-01-01' and d.dept_name like 'Human Resources'
41 order by emp_no;
42

```

Código completo. Lanzo un select a la vista.

1503 • select * from RRHH;

<

Result Grid | Filter Rows: | Export: | Wr

	emp_no	first_name	last_name	dept_name
▶	10005	Kyoichi	Maliniak	Human Resources
	10013	Eberhardt	Terkki	Human Resources
	10036	Adamantios	Portugali	Human Resources
	10039	Alejandro	Brender	Human Resources
	10054	Mayumi	Schueller	Human Resources
	10071	Hisao	Lipner	Human Resources
	10077	Mona	Azuma	Human Resources
	10080	Premal	Baek	Human Resources
	10086	Somnath	Foote	Human Resources
	10100	Hironobu	Haraldson	Human Resources

Aquí vemos el select y su ejecución que es idéntica a la ejecución de la consulta.

3. Otorga a los usuarios los siguientes privilegios. (0,5 puntos)
 - a. alan – todos los privilegios para ambas vistas.

```

15 • GRANT ALL PRIVILEGES ON pobreza TO alan@localhost;
16 • GRANT ALL PRIVILEGES ON RRHH TO alan@localhost;

```

Con estos dos comandos doy los privilegios a alan sobre las vistas.

Vamos a comprobar la ejecución de estos dos comandos.

✓	31	10:15:55	GRANT ALL PRIVILEGES ON RRHH TO alan@localhost	0 row(s) affected
✓	32	10:15:59	GRANT ALL PRIVILEGES ON RRHH TO alan@localhost	0 row(s) affected

Aquí vemos que la ejecución va sin problemas.

- b. steve – sólo se le permite consultar sobre la tabla de empleados, sin embargo, debe poder modificar la columna last_name.

```
38 • GRANT SELECT ON employees TO steve@localhost;
```

Con este comando garantizo los privilegios de select sobre employees a Steve.

Por ahora no funciona lo de modificar una sola columna así que lo voy a dejar por ahora.

Pregunta 2. Sigamos. A continuación, encontrarás 5 enunciados de scripts diferentes para implementar. **Elige sólo 3 de ellos** e implementa esos 3. (4,5 puntos)

1. **Enchufados.** Crea una función que busque el primer salario de un empleado dado su emp_no y retorne un mensaje de error si dicha cuantía es inferior a \$100.000. En caso contrario, debe devolver el primer salario del empleado. (1,5 puntos)
2. **Reducción de jornada.** Crea un procedimiento que baje el salario un 40% a un trabajador dado su emp_no. El procedimiento debe asegurarse de cerrar correctamente el registro del salario anterior y abrir un nuevo registro para el nuevo salario. (1,5 puntos)

```

1511
1512 DELIMITER //
1513 • CREATE PROCEDURE reduccion_jornada(in cod_empleado int)
1514 BEGIN
1515
1516     -- Voy a obtener el valor del sueldo actual de un trabajador mediante consulta y lo almaceno
1517     -- en una variable
1518
1519     DECLARE sueldoActual INT;
1520     SELECT salary into sueldoActual
1521     from salaries
1522     where salaries.emp_no = cod_empleado and salaries.to_date like '9999-01-01';
1523
1524     -- Cierro ese registro de ese trabajador
1525     update salaries set to_date = current_date where salaries.emp_no = cod_empleado and to_date like '
1526
1527     -- Creo un nuevo registro de ese trabajador con el importe del nuevo salario
1528     insert into salaries values(cod_empleado, (sueldoActual * 0.60), current_date(), '9999-01-01');
1529
1530 END//

```

Output:

#	Time	Action	Message
34	10:18:37	GRANT UPDATE ON employees.last_name TO steve@localhost	Error Code: 1146. Table 'employees.last_name' doesn't exist
35	10:19:03	GRANT UPDATE ON empresa.employees.last_name TO steve@localhost	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'last_name' at line 1
36	10:29:08	CREATE PROCEDURE reduccion_jornada(in cod_empleado int) BEGIN -- Voy a obt...	0 row(s) affected

Para ahorrar algo de tiempo las explicaciones las doy como comentarios en el código.

```

1532 • select * from salaries where to_date like '9999-01-01';

```

Result Grid

	emp_no	salary	from_date	to_date
▶	10001	88958	2002-06-22	9999-01-01
	10002	72527	2001-08-02	9999-01-01
	10003	43311	2001-12-01	9999-01-01
	10004	74057	2001-11-27	9999-01-01
	10005	94692	2001-09-09	9999-01-01
	10006	59755	2001-08-02	9999-01-01
	10007	88070	2002-02-07	9999-01-01
	10009	94409	2002-02-14	9999-01-01
	10010	80324	2001-11-23	9999-01-01
	10012	54423	2001-12-16	9999-01-01

salaries 7 x

Con este comando busco un trabajador que sea la victima. Ahora debo de ejecutar el procedimiento sobre ese trabajador y ver el resultado.

```
4 call reduccion_jornada(10001);
```

Ejecuto la llamada a la función. Busco ahora todos los registro de este trabajador para ver si hay cambios.

```
1535
1536 select * from salaries where emp_no = 10001 order by from_date desc;
```

	emp_no	salary	from_date	to_date
▶	10001	53375	2025-05-23	9999-01-01
	10001	88958	2002-06-22	2025-05-23
	10001	85097	2001-06-22	2002-06-22

Con el comando de arriba, gracias a la función de ordenar por fecha descendente, vemos todos los cambios. El registro que vimos al principio esta cerrado a fecha de hoy y se ha creado uno nuevo con el salario modificado.

3. **La extra.** Crea un trigger que evite que un empleado tenga dos salarios superpuestos en salaries. Si un nuevo salario se solapa en fechas con otro salario existente para el mismo empleado, muestra un mensaje de error. (1,5 puntos)
4. **Migajas.** Crea un trigger que evite que un empleado pueda recibir un nuevo salario que sea un 5% más alto que su último salario, mostrando un mensaje de error en tal caso. (1,5 puntos)

Vamos con este ejercicio, lo que haremos será crear un trigger, declaramos una variable que almacene el valor del salario actual, lo compare con el nuevo valor de salario que se quiere insertar y en caso de no cumplir con los requisitos devuelva mensaje de error.

Tendremos que crear un trigger cuya condición sea before insert, para que se pueda cancelar el insert.

```

600 delimiter //
601 • create trigger no_tan_rapido
602 before insert on salaries
603 for each row
604 begin
605     -- declaramos una variable para almacenar el último salario de un trabajador
606     declare salarioActual int;
607     -- Obtenemos con un select el valor de el salario actual de ese trabajador
608     select s.salary into salarioActual from salaries s where s.emp_no = new.emp_no and s.to_date like '9999-01-01';
609
610     -- Ahora debemos de realizar la comparación para ver si se pasa de alto el sueldo
611     if new.salary > (salarioActual * 1.05) then
612         signal sqlstate '50001' set message_text = 'No quieras cobrar tanto que no nos sobra el dinero';
613     end if;
614
615 end //
616

```

Aquí tenemos el código del trigger. Vemos que la condición es before insert on salaries.

Declaro una variable del tipo int llamada salarioActual para almacenar el valor del salario actual.

Obtengo el valor de salarioActual mediante una consulta.



Mediante un if evaluo el valor de new.salary con el valor de salarioActual más un 5% con (salarioActual * 1.05).

Si esta condición se cumple devuelvo un mensaje. No se si 50001 es correcto o si sería más correcto 45000.

```

1618 • select * from salaries where emp_no = 10001 and to_date like '9999-01-01';
1619

```

Result Grid				
Filter Rows: <input type="text"/>				
Export:  Wrap Cell Content: 				
emp_no	salary	from_date	to_date	
▶ 10001	53375	2025-05-23	9999-01-01	

Como no me he querido complicar busco un conejillo de indias, el trabajador 10001 que es un sufridor. Compruebo el valor de su salario actual. Y ahora voy a intentar hacer un nuevo insert en salaries para este trabajador, ya que queremos ver que pasa.


```

1619
1620 insert into salaries values (10001, 100000, current_date(), '9999-01-01');

```

Output

#	Time	Action	Message
66	11:24:24	insert into salaries values (10001, 100000, current_date(), '9999-01-01')	Error Code: 1644. No quieras cobrar tanto que no nos sobra el dinero
67	11:27:21	select * from salaries where emp_no = 10001 and to_date like '9999-01-01' LIMIT 0, ...	1 row(s) returned
68	11:28:47	insert into salaries values (10001, 100000, current_date(), '9999-01-01')	Error Code: 1644. No quieras cobrar tanto que no nos sobra el dinero

Al ejecutar el insert que vemos en la pantalla, con un valor un tanto abultado para asegurarnos que supere el 5% del límite que queremos, la consola de workbench nos devuelve el mensaje que le hemos indicado. Ahora debo de intentar comprobar en la tabla salaries para el trabajador 10001 si se ha insertado algo nuevo.

```

1617
1618 • select * from salaries where emp_no = 10001 and to_date like '9999-01-01';
1619

```

Result Grid

emp_no	salary	from_date	to_date
10001	53375	2025-05-23	9999-01-01

Vemos que todo sigue igual que como lo teníamos antes, ya que de haberse producido el inserte tendríamos como resultado del select dos registros con distintos valores de salario, current_date como from_date y '9999-01-01' como to date.

5. **Las mujeres facturan.** Escribe un procedimiento almacenado que devuelva una lista de las empleadas (mujeres) que actualmente cobran más una determinada cuantía, la cual se pasa como parámetro. (1,5 puntos)

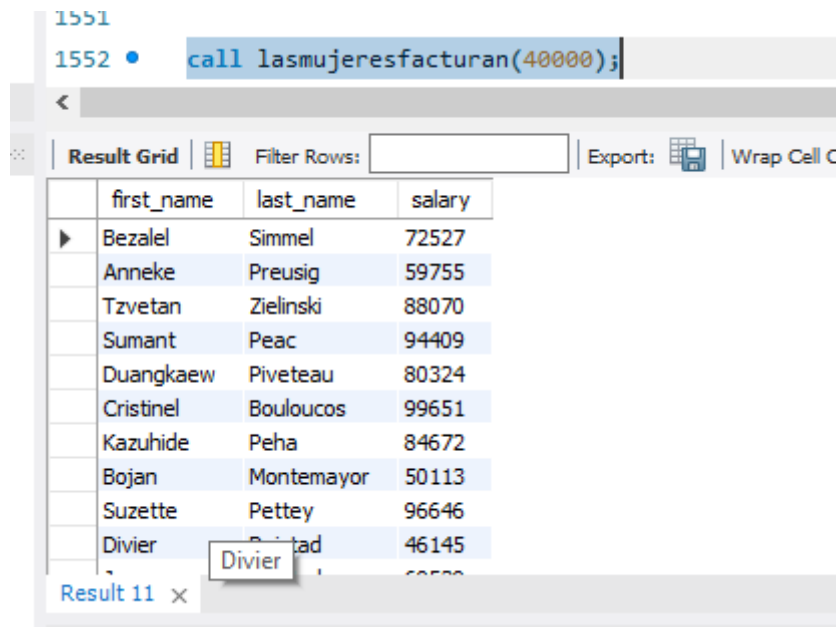
```

---
540 DELIMITER //
541 • CREATE PROCEDURE lasmujeresfacturan(in cuantia int)
542 BEGIN
543
544     -- Aquí lo que tengo que hacer es obtener los resultados con una consulta
545     select e.first_name, e.last_name, s.salary
546     from employees e
547     join salaries s on e.emp_no = s.emp_no
548     where e.gender = 'F' and s.salary > cuantia and s.to_date like '9999-01-01';
549
550 END//
551

```

Creamos el procedimiento, definimos un parámetro de entrada que será la cuantía a comparar, desarrollamos la consulta dentro del procedimiento y finalizamos el procedimiento.

Ahora lo llamamos y buscamos el resultado.



The screenshot shows a database interface with a SQL query editor at the top and a result grid below. The query is `call lasmujeresfacturan(40000);`. The result grid displays a table with three columns: `first_name`, `last_name`, and `salary`. The table contains 11 rows of employee data. A tooltip is visible over the row for 'Divier', showing the value '46145' for the salary column.

first_name	last_name	salary
Bezalel	Simmel	72527
Anneke	Preusig	59755
Tzvetan	Zielinski	88070
Sumant	Peac	94409
Duangkaew	Piveteau	80324
Cristinel	Bouloucos	99651
Kazuhide	Peha	84672
Bojan	Montemayor	50113
Suzette	Pettey	96646
Divier	Peatad	46145
		66666

Al llamar al procedimiento indicándole un parámetro obtenemos el resultado deseado.

Pregunta 3. Nuestro superintendente quiere que cada vez que un empleado reciba un nuevo título, se registren el emp_no, la fecha del nombramiento, el nuevo título y, especialmente, el emp_no del manager del departamento en una tabla llamada "titulines". Considera las siguientes preguntas. (2,5 puntos)

1. Para lograr la funcionalidad requerida, ¿qué herramienta sería la más apropiada en este caso? ¿Trigger, función o procedimiento? Justifica tu respuesta. (1 punto)

En este caso yo creo que el elemento más adecuado sería un trigger.

La razón es que para introducir datos en una tabla externa después de insertar nuevos datos en otra tabla, con un after insert en un trigger indicando como objetivo la tabla afectada podemos automatizar y realizar un insert en la tabla auxiliar. Si bien es cierto que tendremos que realizar una query dentro del trigger para obtener ciertos datos y eso nos puede provocar problemas de concurrencia severos, sobre todo en BD muy grandes, como tampoco es

aconsejable llamar a funciones y procedimientos dentro de un trigger por la misma razón lo haremos dentro del trigger. Habrá que definir varias variables posiblemente dentro del mismo, pero creo que aún así es nuestra mejor opción.

2. Desarrolla la funcionalidad solicitada y demuestra su funcionamiento. (1,5 puntos)

```
3  
4 create table titulines(  
5     codigoEmp int,  
6     fNombra date,  
7     titulo varchar(40),  
8     codigoMan int  
9 );
```

Lo primero que tengo que hacer es crear la tabla. Ya se que no le he dado primary key pero dado que este ejercicio es para el examen no lo veo indispensable.

```

564 DELIMITER //
565 • CREATE TRIGGER titulitis
566 AFTER INSERT ON titles
567 FOR EACH ROW
568 BEGIN
569
570     -- Vamos a hacerlo paso por paso aunque sea más largo
571     -- Primero obtengo el departamento en el que trabaja este empleado
572
573     declare departamento char(4);
574     declare jefe int;
575
576     select dept_no into departamento from dept_emp where emp_no = new.emp_no and dept_emp.to_date like '9999-01-01';
577     -- Una vez tenemos el código de departamento lo que hacemos es obtener el código del jefe de
578     -- departamento
579     select emp_no into jefe from dept_manager where dept_no = departamento and to_date like '9999-01-01';
580
581     -- Después de esto sólo tenemos que realizar el insert en titulines
582
583     insert into titulines values(new.emp_no, new.from_date, new.title, jefe);
584
585
586 END //
587

```

Este es el código del trigger. Se podría hacer más corto seguro, pero he decidido hacerlo en pasos más pequeños para asegurar. Declaramos que salta after inserts on titles. Eso lo tenemos claro. Podría hacerse en una consulta más grande, pero a lo seguro.

Después lo que hago es declarar dos variables, una departamento que almacenará el código de departamento en el que trabaja el trabajador y otra jefe que almacenará el emp_no del jefe de ese departamento en caso de que tenga jefe ese departamento.

Con la primera consulta obtengo el código de departamento y lo almaceno en departamento.

Con la segunda consulta obtengo el emp_no de ese departamento.

Luego lo importante, después de hacer un inserte en titles lo que hago es un insert en titulines con los siguientes valores:

New.emp_no : Concuerda con el del empleado que ha recibido un nuevo título

New.from_date: Concuerda con la fecha en la que le es otorgado el título

New.title : concuerda con el nuevo valor del título que se ha insertado en la tabla titles

Jefe : El emp_no del jefe de departamento en el que trabaja el sujeto en cuestión.

```

589
590     select * from titles where to_date like '9999-01-01';
591

```

Uso este comando, pero simplemente para seleccionar la víctima del nuevo título.

```

1
2 insert into titles values (10001, 'Titulazo', current_date(), '9999-01-01');
3

```

Hago un nuevo insert en la tabla titles. Como fecha desde uso current_date(), fecha hasta la del fin de los tiempos, y como titulo uno que se vea bien de lejos al comprobar el resultado.

```

1593
1594 select * from titles where emp_no = 10001 and to_date like '9999-01-01';

```

emp_no	title	from_date	to_date
10001	Senior Engineer	1986-06-26	9999-01-01
10001	Titulazo	2025-05-23	9999-01-01

Ejecuto una comprobación en la tabla titles para ver que se ha ejecutado correctamente el insert. Se puede dar el caso de que hubiese un error.

Ahora solo queda hacer un select a la tabla titulines para comprobar si se ha realizado el insert allí de forma correcta.

```

1596 select * from titulines;
1597

```

codigoEmp	fNombra	titulo	codigoMan
10001	2025-05-23	Titulazo	10008

Aquí vemos el resultado. Se han insertado exactamente los datos que indicábamos en el problema y en el trigger.