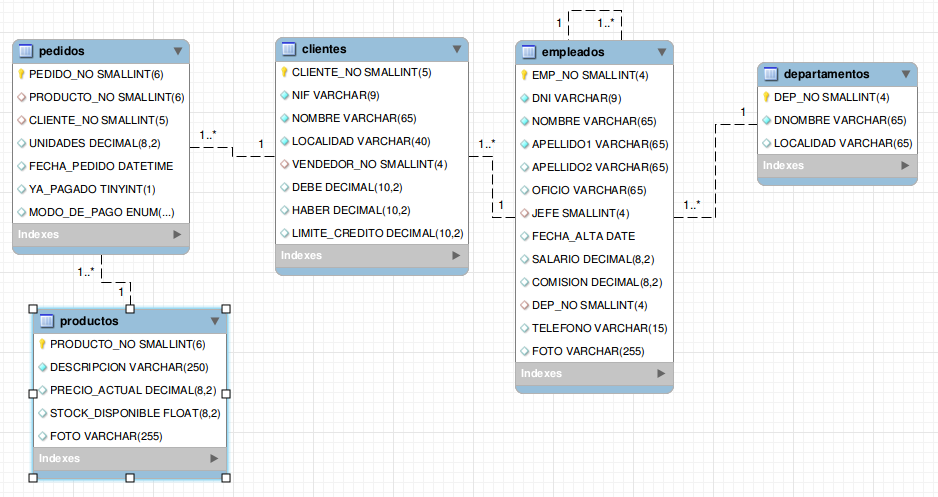
Índice:

**No se encontraron elementos de tabla de contenido.**



1. Listar los 5 pedidos más recientes (incluyendo los datos del producto)

SELECT \*

FROM pedidos NATURAL JOIN productos

ORDER BY fecha\_pedido DESC

LIMIT 5

;

1. Mostrar el salario más alto y el más bajo de la empresa

SELECT MAX(salario), MIN(salario)

FROM empleados;

1. Mostrar el salario más alto de la empresa, junto con el nombre de los empleados que tienen ese salario (puede haber varios empleados con ese salario)

SELECT nombre, salario

FROM empleados

WHERE salario = (

SELECT MAX(salario)

FROM empleados);

1. Mostrar el salario más alto y el más bajo de la empresa, junto con el nombre de los empleados que tienen esos salarios

SELECT nombre, salario

FROM empleados

WHERE salario = (

SELECT MAX(salario)

FROM empleados)

OR

salario = (

SELECT MIN(salario)

FROM empleados)

;

1. Mostrar a los empleados que tienen asignado un departamento inexistente (es posible que no devuelva ninguna fila porque todos estén bien asignados).

SELECT \*

FROM empleados

WHERE dep\_no NOT IN(

SELECT dep\_no

FROM departamentos);

1. Indicar cuántos empleados tiene a su cargo cada empleado que sea jefe de otros. Sólo mostrar los que son jefes de 2 personas exactamente. Se debe de mostrar el nombre y apellidos de esos jefes.

SELECT jefe.emp\_no, jefe.apellido1, jefe.apellido2, jefe.nombre, COUNT(\*)

FROM empleados subordinado INNER JOIN empleados jefe ON subordinado.jefe = jefe.emp\_no

GROUP BY jefe.emp\_no

HAVING COUNT(\*)=2;

1. Mostrar el salario más alto y el más bajo de la empresa, junto con el nombre de los empleados que tienen esos salarios

SELECT nombre, salario

FROM empleados

WHERE salario = (

SELECT MAX(salario)

FROM empleados)

OR

salario = (

SELECT MIN(salario)

FROM empleados)

;

1. Actualizar el stock del producto\_no 40, reduciéndolo en 2 unidades y rebajar el precio un 5%, todo ello mediante un único comando.

UPDATE productos

SET STOCK\_DISPONIBLE = STOCK\_DISPONIBLE - 2, PRECIO\_ACTUAL = PRECIO\_ACTUAL - PRECIO\_ACTUAL\*5/100

WHERE PRODUCTO\_NO = 40

;

1. Mostrar a todos los empleados, al lado de cada empleado mostrar los datos de su departamento. Si un empleado no tiene asignado departamento, también se muestra al empleado (los datos del departamento aparecerían en blanco).

SELECT \*

FROM empleados LEFT JOIN clientes ON empleados.emp\_no = clientes.vendedor\_no

;

1. Borrar al empleado 9013

DELETE

FROM empleados

WHERE EMP\_NO = 9003;

1. Insertar los siguientes departamentos mediante un único comando INSERT:

70 MANTENIMIENTO FERROL

80 REPARACIONES NARÓN

INSERT INTO departamentos VALUES

(70, 'MANTENIMIENTO', 'FERROL'),

(80, 'REPARACIONES', 'NARÓN')

;

1. Cuánto dinero en total ha facturado la empresa en 1999

SELECT SUM(precio\_actual \* unidades) AS 'Facturación en 1999'

FROM pedidos INNER JOIN productos ON pedidos.producto\_no = productos.producto\_no

WHERE fecha\_pedido >= '1999-01-01' AND fecha\_pedido < '2000-01-01'

;

SELECT SUM(precio\_actual \* unidades) AS 'Facturación en 1999'

FROM pedidos INNER JOIN productos ON pedidos.producto\_no = productos.producto\_no

WHERE fecha\_pedido LIKE'1999-%'

;

1. Listar aquellos departamentos en los que todos sus empleados carezcan de información sobre su comisión.

**SELECT \* FROM departamentos**

**WHERE dep\_no NOT IN (**

**SELECT dep\_no FROM empleados**

**WHERE comision IS NOT NULL AND dep\_no IS NOT NULL**

**);**

**#esta sin embargo es incorrecta, AVERIGUA POR QUÉ:**

**SELECT \* FROM departamentos**

**WHERE dep\_no NOT IN (**

**SELECT dep\_no FROM empleados**

**WHERE comision IS NOT NULL**

**);**

1. Crear una tabla temporal llamada pedidos\_ano\_2000 que incluya los pedidos del año 2000, la tabla temporal tendrá los mismos atributos de la tabla original, pero sin KEYs.

CREATE TEMPORARY TABLE IF NOT EXISTS pedidos\_ano\_2000

(SELECT \*

FROM pedidos

WHERE YEAR(FECHA\_PEDIDO) = '2000'

);

¿Quién puede acceder a esa tabla?

Sólo puede acceder a esa tabla la sesión que la ha creado.

¿Esa tabla existe indefinidamente?

La tabla temporal se crea en RAM y desaparece cuando se cierra la sesión (o si el usuario la borra explícitamente con DROP TABLE).

1. Borrar el salario del empleado 7499

#Lo que nos piden no se realiza con el comando DELETE (borraría la fila completa.

#La solución es guardar el valor NULL en el salario del empleado:

UPDATE empleados

SET salario = NULL

WHERE emp\_no = 7499;

1. Media de ganancias (*salario* +*comisión*) de los empleados con oficio *vendedor*

SELECT AVG(IFNULL(salario,0) + IFNULL(comision,0))

FROM empleados

WHERE oficio LIKE 'vendedor';

La siguiente es una solución ERRÓNEA:

SELECT AVG(salario+ comision)

FROM empleados

WHERE oficio LIKE 'vendedor';

1. Insertar los siguientes empleados mediante comandos INSERT individuales para cada fila. Averigua por qué se produce un error y utiliza 2 posibles soluciones al problema.

INSERT INTO `bdempleados`.`empleados` (`EMP\_NO`, `DNI`, `NOMBRE`, `APELLIDO1`, `APELLIDO2`, `OFICIO`, `JEFE`, `FECHA\_ALTA`, `SALARIO`, `COMISION`, `DEP\_NO`, `TELEFONO`)

VALUES ('9001', '14256821Y', 'Luís', 'Yañez', 'Rioboo', 'ADMINISTRATIVO', '9002', '2016-04-05', '150000', '0', '10', '652986532');

INSERT INTO `bdempleados`.`empleados` (`EMP\_NO`, `DNI`, `NOMBRE`, `APELLIDO1`, `APELLIDO2`, `OFICIO`, `JEFE`, `FECHA\_ALTA`, `SALARIO`, `COMISION`, `DEP\_NO`, `TELEFONO`)

VALUES ('9002', '45325698P', 'Ana', 'Rinlo', 'Rois', 'INGENIERA', '8904', '2016-04-05', '150000', '0', '10', '547852146');

#El problema es que la primera fila a insertar incumple la FOREIGN KEY en el atributo JEFE: intentamos insertar un empleado(9001) cuyo jefe (9002) todavía no existe: no existe el empleado 9002 cuando intentamos insertar al empleado 9001

#Manera de evitarlo 1:

#deshabilitar chequeo de las FOREIGN KEY

SET FOREIGN\_KEY\_CHECKS=0;

#insertar datos

INSERT INTO `bdempleados`.`empleados` (`EMP\_NO`, `DNI`, `NOMBRE`, `APELLIDO1`, `APELLIDO2`, `OFICIO`, `JEFE`, `FECHA\_ALTA`, `SALARIO`, `COMISION`, `DEP\_NO`, `TELEFONO`)

VALUES ('9001', '14256821Y', 'Luís', 'Yañez', 'Rioboo', 'ADMINISTRATIVO', '9002', '2016-04-05', '150000', '0', '10', '652986532');

INSERT INTO `bdempleados`.`empleados` (`EMP\_NO`, `DNI`, `NOMBRE`, `APELLIDO1`, `APELLIDO2`, `OFICIO`, `JEFE`, `FECHA\_ALTA`, `SALARIO`, `COMISION`, `DEP\_NO`, `TELEFONO`)

VALUES ('9002', '45325698P', 'Ana', 'Rinlo', 'Rois', 'INGENIERA', '8904', '2016-04-05', '150000', '0', '10', '547852146');

#habilitar chequeo de las FOREIGN KEY

SET FOREIGN\_KEY\_CHECKS=1;

#el problema es que al volver a habilitar el chequeo de las FOREIGN KEY no se comprueba si las filas introducidas previamente tienen algún error en sus FOREIGN KEY.

#Tenemos que comprobar si alguna clave ajena es errónea, para ello buscamos esas posibles filas erróneas mediante esta consulta: buscar aquellos empleados cuyo jefe no existe.

SELECT \*

FROM empleados

WHERE jefe NOT IN (

SELECT emp\_no

FROM empleados

);

#Manera de evitarlo 2:

#ordenar los INSERT de modo que no se infrinjan las FOREIGN KEY (si fuesen muchas filas sería muy complicado ordenarlas todas, y en algunos casos sería imposible si hubiese referencias cruzadas)

INSERT INTO `bdempleados`.`empleados` (`EMP\_NO`, `DNI`, `NOMBRE`, `APELLIDO1`, `APELLIDO2`, `OFICIO`, `JEFE`, `FECHA\_ALTA`, `SALARIO`, `COMISION`, `DEP\_NO`, `TELEFONO`) VALUES ('9002', '45325698P', 'Ana', 'Rinlo', 'Rois', 'INGENIERA', '8904', '2016-04-05', '150000', '0', '10', '547852146');

INSERT INTO `bdempleados`.`empleados` (`EMP\_NO`, `DNI`, `NOMBRE`, `APELLIDO1`, `APELLIDO2`, `OFICIO`, `JEFE`, `FECHA\_ALTA`, `SALARIO`, `COMISION`, `DEP\_NO`, `TELEFONO`) VALUES ('9001', '14256821Y', 'Luís', 'Yañez', 'Rioboo', 'ADMINISTRATIVO', '9002', '2016-04-05', '150000', '0', '10', '652986532');

1. Indicar el número de empleados que ganan más de 160.000 en cada departamento; mostrar solamente aquellos departamentos que tienen, al menos, 2 empleados que ganen más de 160.000. El título de las columnas será:

* Departamento
* Número de empleados

SELECT dnombre AS 'Departamento', COUNT(\*) AS 'Número de empleados'

FROM empleados NATURAL JOIN departamentos

WHERE salario > 160000

GROUP BY dep\_no

HAVING COUNT(\*) >= 2

;

1. Obtener el nombre y salario de los empleados que ganan menos que la media salarial de su departamento.

SELECT apellido1, apellido2, nombre, salario, oficio, dnombre

FROM empleados e1 NATURAL JOIN departamentos

WHERE e1.salario < (

SELECT AVG(salario)

FROM empleados e2

WHERE e2.dep\_no = e1.dep\_no

);

1. Mostrar a todos los empleados, al lado de cada empleado mostrar los datos de cada cliente que atiende. Si un empleado no atiende a clientes SÍ se muestra al empleado

SELECT \*

FROM empleados LEFT JOIN clientes ON empleados.emp\_no = clientes.vendedor\_no

;

1. Visualizar el departamento con más empleados.

**SELECT dnombre**

**FROM departamentos**

**WHERE dep\_no IN**

**(**

**SELECT dep\_no**

**FROM empleados**

**GROUP BY dep\_no**

**HAVING COUNT (\*) >= ALL**

**(**

**SELECT COUNT(\*)**

**FROM empleados**

**GROUP BY dep\_no**

**)**

**);**

#otra forma

SELECT COUNT(\*), dnombre

FROM empleados NATURAL JOIN departamentos

GROUP BY departamentos.dep\_no

HAVING COUNT(\*) >= (

SELECT COUNT(\*)

FROM empleados

GROUP BY dep\_no

ORDER BY 1 DESC

LIMIT 1

);

1. mostrar el número de pedido y la fecha del pedido de aquellos pedidos realizados en el año 1999 fuera del mes de noviembre

SELECT pedido\_no, fecha\_pedido

FROM bdempleados.pedidos

WHERE fecha\_pedido NOT LIKE '1999-11%';

SELECT pedido\_no, date\_format(fecha\_pedido, '%d/%m/%Y') AS fecha

FROM bdempleados.pedidos

WHERE YEAR(fecha\_pedido) = 1999

AND

NOT (MONTH(fecha\_pedido) = 11) ;

SELECT pedido\_no, date\_format(fecha\_pedido, '%d/%m/%Y') AS fecha

FROM bdempleados.pedidos

WHERE YEAR(fecha\_pedido) = 1999

AND

MONTH(fecha\_pedido) <> 11 ;

SELECT pedido\_no, date\_format(fecha\_pedido, '%d/%m/%Y') AS fecha

FROM bdempleados.pedidos

WHERE YEAR(fecha\_pedido) = 1999

AND

MONTH(fecha\_pedido) NOT IN (11) ;

1. Importe del pedido más caro y datos de la persona que realiza ese pedido

**SELECT clientes.nombre, pedido\_no, descripcion, unidades, precio\_actual, unidades \* precio\_actual**

**FROM clientes NATURAL JOIN pedidos NATURAL JOIN productos**

**WHERE unidades \* precio\_actual =**

**(SELECT MAX(unidades \* precio\_actual)**

**FROM pedidos NATURAL JOIN productos**

**)**

**;**

1. Apellido, localidad y oficio de aquellos empleados de Madrid o Barcelona que ganan menos de 200.000

SELECT apellido1, localidad, oficio, salario

FROM empleados INNER JOIN departamentos ON empleados.dep\_no = departamentos.DEP\_NO

WHERE (localidad = 'Madrid' OR localidad = 'Barcelona' ) AND salario < 200000

;

SELECT apellido, localidad, oficio, salario

FROM empleados INNER JOIN departamentos ON empleados.dep\_no = departamentos.DEP\_NO

WHERE localidad IN ('Madrid','Barcelona') AND salario < 200000

;

1. Mostrar aquellos departamentos que SÍ tienen empleados

SELECT \*

FROM departamentos WHERE dep\_no IN (

SELECT DISTINCT(dep\_no)

FROM empleados

);

1. Crear una nueva tabla llamada empleados\_antiguos\_reducida a partir de la tabla empleados (debe crearse utilizando el resultado de un SELECT) que incluya sólo estos atributos (emp\_no, apellido1, apellido2 y nombre) y sin KEYs, e insertar en ella al empleado 8904

CREATE TABLE IF NOT EXISTS empleados\_antiguos\_reducida

(SELECT emp\_no, apellido1, apellido2, nombre

FROM empleados

WHERE emp\_no = 8904

);

1. Mostrar el apellido y salario de los 5 empleados con **menor** salario (si hay empate y por ello hay más de 5, deben mostrarse todos ellos)

SELECT apellido1, nombre, salario

FROM empleados

WHERE salario <=

(SELECT salario

FROM empleados WEHRE salario IS NOT NULL

ORDER BY salario ASC

LIMIT 4,1)

ORDER BY 3 ASC

;

#esta otra no funcionaría si el 5º salario más bajo lo ganasen varios empleados, porque dejaría de mostrarse alguno de ellos

SELECT apellido, salario

FROM empleados

ORDER BY salario ASC

LIMIT 5

;

#Esta otra no funciona porque MySQL se queja de que estamos utilizando LIMIT en una subconsulta, pero eso también lo hacemos en la primera solución y allí no da problema, así que el problema interno achacable a MySQL debe ser un poco más complejo que el indicado en el mensaje de error:

SELECT apellido1, nombre, salario

FROM empleados

WHERE salario IN

(SELECT salario

FROM empleados

WHERE salario IS NOT NULL

ORDER BY salario ASC

LIMIT 5)

ORDER BY 3 ASC

;

1. mostrar el número de pedido y la fecha del pedido de aquellos pedidos realizados en noviembre del año 1999

**SELECT pedido\_no, date\_format(fecha\_pedido, '%d/%m/%Y') AS fecha**

**FROM bdempleados.pedidos**

**WHERE fecha\_pedido like '1999-11-%';**

**SELECT pedido\_no, date\_format(fecha\_pedido, '%d/%m/%Y') AS fecha**

**FROM bdempleados.pedidos**

**WHERE fecha\_pedido BETWEEN '1999-11-01' AND '1999-11-30';**

**SELECT pedido\_no, date\_format(fecha\_pedido, '%d/%m/%Y') AS fecha**

**FROM bdempleados.pedidos**

**WHERE YEAR(fecha\_pedido) = 1999**

**AND**

**MONTH(fecha\_pedido) = 11;**

1. Para cada pedido indicar el nombre del cliente y el nombre del producto comprado

SELECT clientes.nombre AS 'Nombre del cliente ', productos.descripcion AS 'Nombre del producto'

FROM (clientes INNER JOIN pedidos

ON clientes.cliente\_no = pedidos.cliente\_no) INNER JOIN productos

ON pedidos.producto\_no = productos.producto\_no

;

1. Mostrar el **nombre** de todos los productos que se han vendido alguna vez. No se permite mostrar duplicados

SELECT DISTINCT descripcion

FROM pedidos NATURAL JOIN productos;

1. Media de *salario* de los empleados con oficio *vendedor*

SELECT AVG(salario)

FROM empleados

WHERE oficio = 'vendedor';

1. Mostrar aquellos departamentos que NO tienen empleados

SELECT \*

FROM departamentos

WHERE dep\_no NOT IN (

SELECT DISTINCT(dep\_no)

FROM empleados

WHERE dep\_no IS NOT NULL

);

#otra válida, que no elimina duplicados en la subconsulta, pero no es necesario hacerlo:

SELECT \*

FROM departamentos

WHERE dep\_no NOT IN (

SELECT dep\_no

FROM empleados

WHERE dep\_no IS NOT NULL

);

#aunque aparentemente esta otra es correcta, falla, porque hay empleados que tienen NULL en su departamento:

SELECT \*

FROM departamentos

WHERE dep\_no NOT IN (

SELECT DISTINCT(dep\_no)

FROM empleados

);

1. mostrar el nombre y localidad de cada cliente ordenados de modo ascendente por localidad, y para aquellos clientes de la misma localidad debe ordenarse por el nombre del cliente en orden **descendente**

SELECT NOMBRE,LOCALIDAD

FROM clientes

ORDER BY LOCALIDAD, NOMBRE DESC

;

1. Mostar a los Empleados de oficio vendedor (todos los vendedores) y mostrar también a los Empleados de oficio director (pero sólo a los directores que gane menos de 300000).

**SELECT\***

**FROM empleados**

**WHERE oficio = 'vendedor'**

**OR**

**(oficio = 'director' AND salario < 300000)**

**;**

1. productos cuya descripción incluye la palabra *mesa*

SELECT \*

FROM productos

WHERE descripcion LIKE '%mesa%'

1. Apellido, oficio, salario, comisión y ganancias (salario + comisión) de todos los empleados

**SELECT apellido, oficio, salario, comision ,**

**IFNULL(salario,0) + IFNULL(comision,0) AS 'ganacias'**

**FROM empleados IFNULL(salario + comision,0)**

**;**

**Este estaría mal porque si salario = 30 y comisión = NULL el resultado es 0:**

**SELECT apellido, oficio, salario, comision ,**

**IFNULL(salario + comision,0) AS 'ganacias'**

**FROM empleados**

**;**

1. Para cada empleado indicar el apellido del empleado y el apellido de su jefe. El nombre de las columnas respectivas será: subordinado, jefe

SELECT subordinado.apellido AS 'subordinado', jefe.apellido AS 'jefe'

FROM empleados subordinado INNER JOIN empleados jefe ON subordinado.director = jefe.emp\_no

;

1. Indicar cuántos clientes tenemos en cada localidad. Cada columna debe de mostrar una etiqueta significativa, ordenar de mayor a menor número de empleados (y los que empaten desempatan por el nombre de localidad en ascendente).

SELECT localidad, COUNT(\*) AS 'Cantidad de empleados que trabajan en la localidad'

FROM clientes

GROUP BY localidad

;

1. Obtener el nombre y salario del empleado (o empleados si varios empatan) que mayor salario tiene dentro de cada oficio, excluyendo al presidente.

SELECT apellido1, apellido2, nombre, salario, oficio

FROM empleados e1

WHERE e1.oficio <> 'presidente' AND salario = (

SELECT MAX(salario)

FROM empleados e2

WHERE e2.oficio = e1.oficio

);

1. Para cada empleado indicar el nombre de los productos que ha vendido

SELECT DISTINCT clientes.nombre AS 'Nombre del cliente ', productos.descripcion AS 'Nombre del producto'

FROM ((clientes INNER JOIN pedidos

ON clientes.cliente\_no = pedidos.cliente\_no) INNER JOIN productos

ON pedidos.producto\_no = productos.producto\_no) INNER JOIN empleados

ON clientes.vendedor\_no = empleados.emp\_no

;

1. Para cada cliente mostrar cada uno de los pedidos realizados, indicando el nombre del producto y la cantidad de producto comprada en ese pedido

**SELECT nombre, DESCRIPCION, UNIDADES**

**FROM (clientes INNER JOIN pedidos ON clientes.cliente\_no = pedidos.cliente\_no)**

**INNER JOIN productos ON productos.producto\_no = pedidos.producto\_no**

**;**

**SELECT nombre, DESCRIPCION, UNIDADES**

**FROM (clientes NATURAL JOIN pedidos)**

**NATURAL JOIN productos**

**;**

1. Apellido, oficio, salario, comisión y ganancias (salario + comisión) de aquellos empleados que ganan (salario + comisión) más de 150.000. La columna de ganancias se llamará ganancias.

SELECT apellido1, apellido2, oficio, salario, comision,

IFNULL(comision,0) + IFNULL(salario,0) AS 'ganacias'

FROM empleados

WHERE IFNULL(salario, 0) + IFNULL(comision,0) > 150000

;

1. Para cada cliente indicar cuál es el producto del que más unidades ha comprado (si hay empate entre varios productos, indicarlos todos).

SELECT c1.nombre AS 'Cliente', pr1.descripcion AS 'Producto', SUM(pe1.unidades) AS 'Unidades vendidas'

FROM (clientes c1 NATURAL JOIN pedidos pe1) NATURAL JOIN productos pr1

GROUP BY c1.cliente\_no, pe1.producto\_no

HAVING SUM(pe1.unidades) >= ALL (

(SELECT SUM(pe2.unidades)

FROM (clientes c2 NATURAL JOIN pedidos pe2) NATURAL JOIN productos pr2

WHERE c1.cliente\_no = c2.cliente\_no

GROUP BY c2.cliente\_no, pe2.producto\_no)

);

1. Utiliza transacciones para borrar todas las filas de la tabla empleados, comprobar que se han borrado y deshacer el borrado

START TRANSACTION;

DELTE \* FROM empleados;

SELECT \* FROM empleados;

ROLLBACK;

1. Dada la siguiente consulta averigua por qué no se puede ejecutar:

USE bdempleados;

SELECT \* FROM empleados WHERE salario IS NOT NULL

ORDER BY salario DESC LIMIT 1

UNION

SELECT \* FROM empleados WHERE salario IS NOT NULL

ORDER BY salario ASC LIMIT 1

;

No se puede ejecutar porque hay un solo comando (un solo ;), y MySQL sólo permite utilizar una vez por comando la opción ORDER BY (y además debe ser al final del comando.

Busca una alternativa para poder conseguir el mismo resultado que se pretende conseguir con la consulta anterior. Pista: usa una tabla temporal y guarda en ella las dos consultas parciales anteriores:

DROP TABLE IF EXISTS empleados\_extremos;

CREATE TEMPORARY TABLE IF NOT EXISTS empleados\_extremos

(SELECT \* FROM empleados WHERE salario IS NOT NULL

ORDER BY salario DESC LIMIT 1

);

INSERT INTO empleados\_extremos

(SELECT \* FROM empleados WHERE salario IS NOT NULL

ORDER BY salario ASC LIMIT 1

);

SELECT \* FROM empleados\_extremos;

## Preguntas

1. ¿Es correcta esta solución para mostrar la información de los empleados, junto con la del departamento al que pertenecen?:

SELECT apellido, empleados.dep\_no, localidad

FROM empleados, departamentos;

1. ¿Con qué comandos se puede confirmar una transacción y qué diferencia existe entre esas opciones?

COMMIT;

Confirma y finaliza una transacción, pero no comienza una nueva transacción.

START TRANSACTION ; # equivale a BEGIN

Confirma y finaliza una transacción, Y comienza una nueva transacción.