



# Boletín 2. Ejercicios SQL avanzados.

---

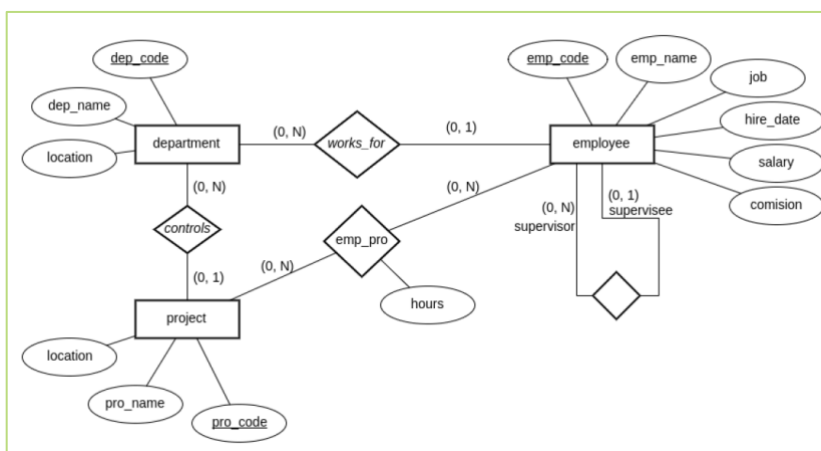
## Introducción a las Bases de Datos.

Pedro Sánchez García

PROFESOR: DR. ANTONIO FARIÑA MARTÍNEZ

## 1. DESCRIPCIÓN DEL MODELO DE DATOS

La base de datos que se usará para las consultas está reflejada en el siguiente modelo Entidad-Relación.



Transformado a relacional, la base de datos almacena cuatro tablas: *department*, *employee*, *project* y *emp\_proc*, cuyos campos se describen a continuación.

Tabla *department*

campo	tipo	descripción
dep_code	NUMERIC(2,0)	Número o código del departamento. Es la clave primaria de la tabla.
dep_name	VARCHAR(14)	Nombre del departamento. Debe ser único.
location	VARCHAR(13)	Localidad donde el departamento está ubicado. No puede tener valores nulos.

Tabla *employee*

campo	tipo	descripción
emp_code	NUMERIC(4,0)	Número o código de la persona empleada. Es la clave primaria de la tabla.
emp_name	VARCHAR(10)	Nombre.
job	VARCHAR(10)	Puesto de trabajo. No puede tener valores nulos.
manager	NUMERIC(4,0)	Código de la persona responsable. Clave foránea que referencia a la tabla <i>employee</i> .
hire_date	DATE	Fecha de contratación.
salary	NUMERIC(7,2)	Salario mensual.
commission	NUMERIC(7,2)	Comisión.
dep_code	NUMERIC(2,0)	Código del departamento al que el empleado o empleada está adscrito. Clave foránea que referencia a la tabla <i>department</i> .

Tabla *project*

campo	tipo	descripción
pro_code	NUMERIC(4,0)	Número o código del proyecto. Es la clave primaria de la tabla.
pro_name	VARCHAR(10)	Nombre del proyecto.
location	VARCHAR(13)	Ciudad donde se realiza el proyecto.
dep_code	NUMERIC(2,0)	Número del departamento controlador del proyecto. Clave foránea que referencia la tabla <i>department</i> .

Tabla *emp\_pro*

campo	tipo	descripción
emp_code	NUMERIC(4,0)	Código del empleado o empleada. Clave foránea que referencia la tabla <i>employee</i> .
pro_code	VARCHAR(10)	Código del proyecto. Clave foránea que referencia la tabla <i>project</i> .
		Estos dos atributos son la clave de la tabla.
hours	NUMERIC(2,0)	Horas que ha trabajado un empleado o empleada en un proyecto.

## Notas

- Todos los valores alfanuméricos están descritos en mayúsculas y sin acentos gráficos.
- En la columna *commission*, si tiene un valor nulo significa que el empleado o empleada no tiene comisión.
- En la columna *manager*, si tiene un valor nulo significa que el empleado o empleada no tiene responsable.

### 1.1 DATOS ALMACENADOS EN LAS TABLAS

Tabla *department*

dep_code	dep_name	location
10	"ACCOUNTING"	"MADRID"
20	"RESEARCH"	"BARCELONA"
30	"SALES"	"LUGO"
40	"OPERATIONS"	"SEVILLA"

4 filas.

Tabla *employee*

emp_code	emp_name	job	manager	hire_date	salary	commi ssion	dep_code
7839	"ABAD"	"PRESIDENT"		"1981-11-17"	5000.00		10
7566	"LOPEZ"	"MANAGER"	7839	"1981-04-02"	2975.00		20
7902	"GONZALEZ"	"ANALYST"	7566	"1981-12-03"	3000.00		20
7369	"RAMOS"	"CLERK"	7902	"1980-12-17"	800.00		20
7698	"IGLESIAS"	"MANAGER"	7839	"1981-05-01"	2850.00		30
7499	"SANCHEZ"	"SALESMAN"	7698	"1981-02-20"	1600.00	300.00	30
7521	"ALVAREZ"	"SALESMAN"	7698	"1981-02-22"	1250.00	500.00	30
7654	"ENRIQUEZ"	"SALESMAN"	7698	"1981-09-28"	1250.00	1400.00	30
7782	"GARCIA"	"MANAGER"	7839	"1981-06-09"	2450.00		10
7788	"MANZANO"	"ANALYST"	7566	"1982-12-09"	3000.00		20
7844	"NADAL"	"SALESMAN"	7698	"1981-09-08"	1500.00	0.00	30
7876	"SAAVEDRA"	"CLERK"	7788	"1983-01-12"	1100.00		20
7900	"TORRES"	"CLERK"	7698	"1981-12-03"	950.00		30
7934	"VIDAL"	"CLERK"	7782	"1982-01-23"	1300.00		10

14 filas.

Tabla *project*

pro_code	pro_name	location	dep_code
1001	P1	SEVILLA	20
1004	P4	LUGO	30
1005	P5	LUGO	30
1006	P6	SANTANDER	30
1008	P8	MADRID	30

5 filas.

Tabla *emp\_pro*

emp_code	pro_code	hours
7499	1004	15
7499	1005	12
7521	1004	10
7521	1008	8
7654	1001	16
7654	1006	15
7654	1008	5
7844	1005	6
7934	1001	4

9 filas.

## 2. CONSULTAS A DESARROLLAR

1) Halla los empleados y empleadas que tienen una comisión superior a la mitad de su salario.

1	SELECT	emp_name
2	FROM	employee
3	WHERE	commission > (salary/2)

Data Output		Explain	Messages	Notific
	emp_name			
	character varying (10)			
1	ENRIQUEZ			

2) Halla los empleados y empleadas que no tienen comisión, o que la tengan menor o igual que el 25% de su salario.

Query Editor		Query History
1	SELECT	emp_name
2	FROM	employee
3	WHERE	commission IS null OR commission <= 0.25*salary

Data Output		Explain	Messages	Notifications
	emp_name			
	character varying (10)			
1	ABAD			
2	LOPEZ			
3	GONZALEZ			
4	RAMOS			
5	IGLESIAS			
6	SANCHEZ			
7	GARCIA			
8	MANZANO			
9	NADAL			
10	SAAVEDRA			
11	TORRES			
12	VIDAL			

3) Obtén los datos del personal que no es supervisado por nadie.

1	SELECT	emp_name
2	FROM	employee
3	WHERE	manager IS null

Data Output		Explain	Message
	emp_name		
	character varying (10)		
1	ABAD		

4) Para el personal con comisión, obtén sus nombres y el cociente entre su salario y su comisión (excepto cuando la comisión sea cero), ordenando el resultado por nombre.

```
1 SELECT emp_name, round(salary/commission,2) AS ratio_sal_comm
2 FROM employee
3 WHERE commission >0
4 ORDER BY emp_name;
```

	emp_name character varying (10)	ratio_sal_comm numeric
1	ALVAREZ	2.50
2	ENRIQUEZ	0.89
3	SANCHEZ	5.33

5) Para los empleados y empleadas que tengan como responsable a alguien con un código mayor que el suyo, obtén los que reciben de salario más de 1000 y menos de 2000, o que están en el departamento 30.

```
1 SELECT emp_name
2 FROM employee
3 WHERE manager > emp_code AND salary BETWEEN 1000 AND 2000
4 OR dep_code = 30
```

	emp_name character varying (10)
1	IGLESIAS
2	SANCHEZ
3	ALVAREZ
4	ENRIQUEZ
5	NADAL
6	TORRES

6) Obtén el nombre, salario, comisión y salario total (salario+comisión, si tiene comisión) del personal con salario total superior a 2300

```
1 SELECT emp_name, salary, commission, salary+commission AS salario_total
2 FROM employee
3 WHERE commission IS NOT null AND (salary+commission)>2300
```

	emp_name character varying (10)	salary numeric (7,2)	commission numeric (7,2)	salario_total numeric
1	ENRIQUEZ	1250.00	1400.00	2650.00

7) Obtén los puestos de trabajo que hay en cada departamento, de forma que no se repitan filas.

Este ejercicio implica JOIN y posterior DISTINCT en job para que no se repitan filas:

```
1 SELECT DISTINCT job
2 FROM employee JOIN department
3 ON employee.dep_code = department.dep_code
```

Data Output	Explain	Messages	Notifications
<b>job</b> character varying (10)			
1 CLERK			
2 PRESIDENT			
3 MANAGER			
4 SALESMAN			
5 ANALYST			

```
1 SELECT DISTINCT job, employee.dep_code, dep_name
2 FROM employee JOIN department
3 ON employee.dep_code = department.dep_code
```

Data Output	Explain	Messages	Notifications
<b>job</b> character varying (10)	<b>dep_code</b> numeric (2)	<b>dep_name</b> character varying (14)	
1 MANAGER	10	ACCOUNTING	
2 MANAGER	30	SALES	
3 MANAGER	20	RESEARCH	
4 CLERK	10	ACCOUNTING	
5 SALESMAN	30	SALES	
6 ANALYST	20	RESEARCH	
7 CLERK	30	SALES	
8 CLERK	20	RESEARCH	
9 PRESIDENT	10	ACCOUNTING	

8) Obtén el salario más alto de la empresa, el total destinado a comisiones y el número de personas trabajando en la empresa.

```
1 SELECT MAX(salary) AS salario_maximo, SUM(commission) AS total_comisiones, COUNT(*) AS personas_totales
2 FROM employee
```

Data Output	Explain	Messages	Notifications
<b>salario_maximo</b> numeric	<b>total_comisiones</b> numeric	<b>personas_totales</b> bigint	
1 5000.00	2200.00	14	

## 9) Halla el nombre del último empleado o empleada por orden alfabético.

```
1 SELECT MAX(emp_name) AS nombre_último_empleado
2 FROM employee
```

	nombre_último_empleado
1	VIDAL

## 10) Halla el salario más alto, el más bajo, y la diferencia entre ellos.

```
1 SELECT MAX(salary) AS salario_máximo, MIN(salary) AS salario_mínimo, (MAX(salary)-MIN(salary)) AS diferencia
2 FROM employee
```

	salario_máximo	salario_mínimo	diferencia
1	5000.00	800.00	4200.00

Si fuese por departamento, para ver esto:

```
1 SELECT dep_code, max(salary), min(salary), max(salary)-min(salary)
2 from employee
3 group by dep_code
```

	max	min	?column?
1	5000.00	1300.00	3700.00
2	2850.00	950.00	1900.00
3	3000.00	800.00	2200.00

## 11) ¿Cuántos empleos diferentes, cuántos empleados y empleadas, y cuántos salarios diferentes encontramos en el departamento 30, y a cuánto asciende la suma de salarios de dicho departamento?


ATENCIÓN CON DETALLE DE REPETIDOS: Se pone DISTINCT en algunos campos:

```
1 SELECT COUNT(DISTINCT job) AS empleos_diferentes, COUNT(*) AS empleados, COUNT(DISTINCT salary) AS salarios, SUM(salary) AS suma_
2 FROM employee JOIN department
3 ON employee.dep_code = department.dep_code
4 WHERE employee.dep_code = 30
5
```

	empleos_diferentes	empleados	salarios	suma_salarios
1	3	6	5	9400.00


## 12) ¿Cuántos empleados y empleadas tiene el departamento 20?

```
1 SELECT COUNT(*) AS empleados_dep30
2 FROM employee
3 WHERE dep_code = 30
```

Data Output	Explain	Messages	Notifications
 empleados_dep30 bigint			
1	6		



## 13) ¿Cuántos empleados y empleadas tienen comisión?

```
1 SELECT COUNT(*) AS empleados_con_comisión
2 FROM employee
3 WHERE commission IS NOT null
```

Data Output	Explain	Messages	Notifications
 empleados_con_comisión bigint			
1	4		



## 14) ¿Qué empleos distintos encontramos en la empresa, y cuánta gente desempeñan cada uno de ellos?

```
1 SELECT COUNT(*), job
2 FROM employee
3 GROUP BY job
```

Data Output	Explain	Messages
 count bigint	 job character varying (10)	
1	4	CLERK
2	1	PRESIDENT
3	3	MANAGER
4	4	SALESMAN
5	2	ANALYST

## 15) Halla la suma de salarios de cada departamento, junto con el código de departamento.

```
1 SELECT SUM(salary), dep_code
2 FROM employee
3 GROUP BY dep_code
```

Data Output	Explain	Messages	Notifi
 sum numeric	 dep_code numeric (2)		
1	8750.00	10	
2	9400.00	30	
3	10875.00	20	



16) Para cada departamento muestra cuántos proyectos controla, junto con el código del departamento.

```
1 SELECT COUNT(*), dep_code
2 FROM project
3 GROUP BY dep_code
```

	Data Output	Explain	Messages	N
	count bigint	dep_code numeric (2)		
1	4	30		
2	1	20		

17) Muestra los proyectos en los que trabajan al menos tres personas y cuántas horas trabajan en dichos proyectos.

Resuelto en clase: Realmente, sobraría el COUNT(\*), aunque no está mal

```
1 SELECT COUNT(*), pro_code, SUM(hours)
2 FROM emp_pro
3 GROUP BY pro_code
4 HAVING COUNT(*) > 0
```

	Data Output	Explain	Messages	Notifications
	count bigint	pro_code numeric (4)	sum numeric	
1	2	1004	25	
2	2	1008	13	
3	2	1001	20	
4	2	1005	18	
5	1	1006	15	

18) Para cada departamento muestra cuántos proyectos controla en cada ciudad.

```
1 SELECT DISTINCT location, COUNT(*), dep_code
2 FROM project
3 GROUP BY dep_code, location
```

	Data Output	Explain	Messages	Notifications
	location character varying (13)	count bigint	dep_code numeric (2)	
1	MADRID	1	30	
2	LUGO	2	30	
3	SANTANDER	1	30	
4	SEVILLA	1	20	

19) Para cada departamento muestra cuántos empleados y empleadas tiene que ganen más de 1500.

```
1 SELECT COUNT(emp_code), dep_code
2 FROM employee
3 WHERE salary > 1500
4 GROUP BY dep_code
```

	Data Output	Explain	Messages	Notification
	count bigint	dep_code numeric (2)		
1	2	10		
2	2	30		
3	3	20		

20) Muestra los departamentos que tienen un salario mínimo mayor o igual a 1000. Muestra su código y cuánta gente trabaja en ellos.

```
1 SELECT dep_code, count(emp_code) AS número_personas
2 FROM employee
3 GROUP BY dep_code
4 HAVING min(salary) >= 1000
```

	Data Output	Explain	Messages	Notifications
	dep_code numeric (2)	número_personas bigint		
1	10	3		

21) Muestra los departamentos y los trabajos donde hay por lo menos dos trabajadores con ese puesto de trabajo.

```
1 SELECT dep_code, job
2 FROM employee
3 GROUP BY dep_code, job
4 HAVING count(*) > 2
```

	Data Output	Explain	Messages	Nc
	dep_code numeric (2)	job character varying (10)		
1	30	SALESMAN		

## 22) Halla los datos de los empleados y empleadas cuyo salario es mayor que el del empleado o empleada de código 7934, ordenando por el salario.

Implica hacer join de tabla consigo misma, tal y como tienes en las anotaciones tomadas en clase:

```
1 SELECT e.*
2 FROM employee e JOIN employee j ON e.salary > j.salary
3 WHERE j.emp_code = 7934
4 ORDER BY e.salary
```

	emp_code	emp_name	job	manager	hire_date	salary	commission	dep_code
	(PK) numeric (4)	character varying (10)	character varying (10)	numeric (4)	date	numeric (7,2)	numeric (7,2)	numeric (2)
1	7844	NADAL	SALESMAN	7698	1981-09-08	1500.00	0.00	30
2	7499	SANCHEZ	SALESMAN	7698	1981-02-20	1600.00	300.00	30
3	7782	GARCIA	MANAGER	7839	1981-06-09	2450.00	[null]	10
4	7698	IGLESIAS	MANAGER	7839	1981-05-01	2850.00	[null]	30
5	7566	LOPEZ	MANAGER	7839	1981-04-02	2975.00	[null]	20
6	7902	GONZALEZ	ANALYST	7566	1981-12-03	3000.00	[null]	20
7	7788	MANZANO	ANALYST	7566	1982-12-09	3000.00	[null]	20
8	7839	ABAD	PRESIDENT	[null]	1981-11-17	5000.00	[null]	10

La otra posibilidad es plantear una subconsulta:

```
--22 como subconsulta
SELECT *
FROM employee
WHERE salary > (
    SELECT salary
    FROM employee
    WHERE emp_code = 7934
)
ORDER BY salary;
```

## 23) Obtén los datos de la gente que trabaja en Barcelona o Madrid.

```
1 SELECT *
2 FROM employee JOIN department
3 ON employee.dep_code=department.dep_code
4 WHERE location = 'MADRID' OR location = 'BARCELONA'
```

p_name	job	manager	hire_date	salary	commission	dep_code	dep_code	dep_name	location
character varying (10)	character varying (10)	numeric (4)	date	numeric (7,2)	numeric (7,2)	numeric (2)	numeric (2)	character varying (14)	character varying (13)
AD	PRESIDENT	[null]	1981-11-17	5000.00	[null]	10	10	ACCOUNTING	MADRID
PEZ	MANAGER	7839	1981-04-02	2975.00	[null]	20	20	RESEARCH	BARCELONA
NZALEZ	ANALYST	7566	1981-12-03	3000.00	[null]	20	20	RESEARCH	BARCELONA
MOS	CLERK	7902	1980-12-17	800.00	[null]	20	20	RESEARCH	BARCELONA
RCIA	MANAGER	7839	1981-06-09	2450.00	[null]	10	10	ACCOUNTING	MADRID
NZANO	ANALYST	7566	1982-12-09	3000.00	[null]	20	20	RESEARCH	BARCELONA
AVEDRA	CLERK	7788	1983-01-12	1100.00	[null]	20	20	RESEARCH	BARCELONA
AL	CLERK	7782	1982-01-23	1300.00	[null]	10	10	ACCOUNTING	MADRID

24) Halla los empleados y empleadas cuyo salario supera o coincide con la media del salario de la empresa.

```

1 SELECT emp_name
2 FROM employee
3 WHERE salary >= (SELECT AVG(salary)
4                  FROM employee)

```

	emp_name	character varying (10)
1	ABAD	
2	LOPEZ	
3	GONZALEZ	
4	IGLESIAS	
5	GARCIA	
6	MANZANO	

25) Obtén los empleados y empleadas del departamento 10 que tienen el mismo empleo que alguien del departamento de ventas. Desconocemos el código de dicho departamento.

```

1 SELECT emp_name
2 FROM employee
3 WHERE dep_code = 10 AND job IN (SELECT job
4                                FROM employee e JOIN department d ON e.dep_code=d.dep_code
5                                WHERE d.dep_name = 'SALES')

```

	emp_name	character varying (10)
1	GARCIA	
2	VIDAL	

26) Halla el conjunto de personas responsables de la empresa, es decir, aquellos que tengan al menos a una persona a su mando, ordenados inversamente por nombre.

He planteado la siguiente propuesta:

```

1 SELECT manager AS responsables_empresa, count(*) AS personas_a_su_mando
2 FROM employee
3 WHERE manager IS NOT null
4 GROUP BY manager
5 HAVING count(*) >= 1
6

```

	responsables_empresa	numeric (4)	personas_a_su_mando	bigint
1	7566		2	
2	7782		1	
3	7902		1	
4	7788		1	
5	7839		3	
6	7698		5	

```

1 SELECT emp_name
2 FROM employee
3 WHERE emp_code IN (SELECT manager
4                     FROM employee)

```

	emp_name
	character varying (10)
1	ABAD
2	LOPEZ
3	GONZALEZ
4	IGLESIAS
5	GARCIA
6	MANZANO

## 27) Halla los empleados y empleadas que no tienen a ninguna persona a su mando

Nos cargamos nulos:

```

1 SELECT emp_name
2 FROM employee
3 WHERE emp_code NOT IN (SELECT manager
4                       FROM employee
5                       WHERE manager IS NOT null)

```

	emp_name
	character varying (10)
1	RAMOS
2	SANCHEZ
3	ALVAREZ
4	ENRIQUEZ
5	NADAL
6	SAAVEDRA
7	TORRES
8	VIDAL

Mediante JOIN:

```

1 SELECT distinct j.emp_name
2 FROM employee e right join employee j ON e.manager=j.emp_code
3 WHERE e.emp_code IS null

```

	emp_name
	character varying (10)
1	ALVAREZ
2	ENRIQUEZ
3	NADAL
4	RAMOS
5	SAAVEDRA
6	SANCHEZ
7	TORRES
8	VIDAL

## 28) Obtén todos los departamentos sin personal

El único que no tiene personal es el 40:

```
1 SELECT dep_code, dep_name
2 FROM department
3 WHERE dep_code NOT IN (SELECT dep_code
4                        FROM employee)
5
```

Data Output	Explain	Messages	Notifications
dep_code [PK] numeric (2)	dep_name character varying (14)		
1	40 OPERATIONS		

## 29) Muestra el código de las personas que más horas trabajan en cada proyecto

Sería la siguiente estructura

```
1 SELECT e.emp_code, pro_code, hours
2 FROM employee e JOIN emp_pro emp ON e.emp_code=emp.emp_code
3 WHERE hours IN (SELECT MAX(hours)
4                 FROM emp_pro
5                 GROUP BY pro_code)
```

Data Output	Explain	Messages	Notifications
emp_code numeric (4)	pro_code numeric (4)	hours numeric (2)	
1	7499	1004	15
2	7499	1005	12
3	7521	1008	8
4	7654	1001	16
5	7654	1006	15

30) Obtén los empleados y empleadas cuyo salario supera al de sus compañeros de departamento. Si hay algún departamento donde dos o más tienen el salario más alto, entonces nadie supera a sus compañeros

---




Se trata de subconsulta correlacionada

```
1 SELECT emp_name, dep_code --,*
2 FROM employee e
3 WHERE salary >
4     (SELECT max(salary)
5      FROM employee
6      WHERE dep_code = e.dep_code
7      AND e.emp_code <> emp_code
8      );
```

Data Output	Explain	Messages	Notifications
<div><div></div><div><div>emp_name</div><div>character varying (10)</div></div></div>	<div><div></div><div><div>dep_code</div><div>numeric (2)</div></div></div>		
1	ABAD	10	
2	IGLESIAS	30	

Posee un problema LIBRETA:

```
1 SELECT emp_name, dep_code --,*
2 FROM employee e
3 WHERE (SELECT count (*)
4        FROM employee
5        WHERE salary >= e.salary AND e.dep_code = dep_code
6        GROUP BY dep_code
7        HAVING max(salary) = e.salary
8        ) =1;
```

Data Output		Explain	Messages	Notifications
	<b>emp_name</b> character varying (10) 	<b>dep_code</b> numeric (2) 		
1	ABAD	10		
2	IGLESIAS	30		
3	PEPE	40		

### 31) Obtén los datos de las personas que más ganan para cada puesto de trabajo

```

1 SELECT *
2 FROM employee
3 WHERE salary IN (SELECT MAX(salary)
4                  FROM employee
5                  GROUP BY job)
6 GROUP BY employee.emp_code, job

```

	emp_code [PK] numeric (4)	emp_name character varying (10)	job character varying (10)	manager numeric (4)	hire_date date	salary numeric (7,2)	commission numeric (7,2)	dep_code numeric (2)
1	7566	LOPEZ	MANAGER	7839	1981-04-02	2975.00	[null]	20
2	7934	VIDAL	CLERK	7782	1982-01-23	1300.00	[null]	10
3	7839	ABAD	PRESIDENT	[null]	1981-11-17	5000.00	[null]	10
4	7902	GONZALEZ	ANALYST	7566	1981-12-03	3000.00	[null]	20
5	7788	MANZANO	ANALYST	7566	1982-12-09	3000.00	[null]	20
6	7499	SANCHEZ	SALESMAN	7698	1981-02-20	1600.00	300.00	30

### 32) ¿Quién trabaja en ciudades de más de cinco letras? Ordena el resultado inversamente por ciudades y normalmente por los nombres

Planteamiento con join:

```

1 SELECT *
2 FROM employee e JOIN department d ON e.dep_code=d.dep_code
3 WHERE length(location) > 5
4 ORDER BY location DESC, emp_name

```

emp_name character varying (10)	job character varying (10)	manager numeric (4)	hire_date date	salary numeric (7,2)	commission numeric (7,2)	dep_code numeric (2)	dep_code numeric (2)	dep_name character varying (14)	location character varying (10)
ABAD	PRESIDENT	[null]	1981-11-17	5000.00	[null]	10	10	ACCOUNTING	MADRID
GARCIA	MANAGER	7839	1981-06-09	2450.00	[null]	10	10	ACCOUNTING	MADRID
VIDAL	CLERK	7782	1982-01-23	1300.00	[null]	10	10	ACCOUNTING	MADRID
GONZALEZ	ANALYST	7566	1981-12-03	3000.00	[null]	20	20	RESEARCH	BARCELONA
LOPEZ	MANAGER	7839	1981-04-02	2975.00	[null]	20	20	RESEARCH	BARCELONA
MANZANO	ANALYST	7566	1982-12-09	3000.00	[null]	20	20	RESEARCH	BARCELONA
RAMOS	CLERK	7902	1980-12-17	800.00	[null]	20	20	RESEARCH	BARCELONA
SAAVEDRA	CLERK	7788	1983-01-12	1100.00	[null]	20	20	RESEARCH	BARCELONA



33) Para cada persona muestra los proyectos en los que trabaja. Muestra el nombre de la persona y el nombre del proyecto

```
1 SELECT e.emp_name, p.pro_name
2 FROM employee e JOIN emp_pro ep ON e.emp_code = ep.emp_code JOIN project p ON ep.pro_code = p.pro_code;
```

	emp_name character varying (10)	pro_name character varying (10)
1	SANCHEZ	P4
2	SANCHEZ	P5
3	ALVAREZ	P4
4	ALVAREZ	P8
5	ENRIQUEZ	P1
6	ENRIQUEZ	P6
7	ENRIQUEZ	P8
8	NADAL	P5
9	VIDAL	P1

34) Muestra los proyectos controlados por cada departamento. muestra el nombre del departamento y el nombre del proyecto. Deben aparecer todos los departamentos, incluso si no controla ningún proyecto.

```
1 SELECT dep_name, pro_name
2 FROM department d LEFT JOIN project p ON d.dep_code = p.dep_code
```

	dep_name character varying (14)	pro_name character varying (10)
1	RESEARCH	P1
2	SALES	P4
3	SALES	P5
4	SALES	P6
5	SALES	P8
6	OPERATIONS	[null]
7	ACCOUNTING	[null]

35) Obtén un listado en el que se reflejen los empleados y empleadas y los nombres de sus responsables. En el listado debe aparecer todo el mundo, aunque no tengan responsable.

Tal y como vimos en clase, aplicando **left join** se obtiene el siguiente resultado:

```
1 SELECT e.emp_name as empleado, j.emp_name as jefe_correspondiente
2 FROM employee e LEFT JOIN employee j ON e.manager = j.emp_code
```

	empleado character varying (10)	jefe_correspondiente character varying (10)
1	ABAD	[null]
2	LOPEZ	ABAD
3	GONZALEZ	LOPEZ
4	RAMOS	GONZALEZ
5	IGLESIAS	ABAD
6	SANCHEZ	IGLESIAS
7	ALVAREZ	IGLESIAS
8	ENRIQUEZ	IGLESIAS
9	GARCIA	ABAD
10	MANZANO	LOPEZ
11	NADAL	IGLESIAS
12	SAAVEDRA	MANZANO
13	TORRES	IGLESIAS
14	VIDAL	GARCIA

### 36) Los nombres de empleados y empleadas contratados antes que su responsable

Aquí apliqué join clásico (inner join) y la condición porque no aparece ninguna aclaración en el enunciado:

```
1 SELECT e.emp_name, e.hire_date, j.emp_name, j.hire_date
2 FROM employee e JOIN employee j ON e.manager=j.emp_code
3 WHERE e.hire_date < j.hire_date
```

	emp_name character varying (10)	hire_date date	emp_name character varying (10)	hire_date date
1	LOPEZ	1981-04-02	ABAD	1981-11-17
2	RAMOS	1980-12-17	GONZALEZ	1981-12-03
3	IGLESIAS	1981-05-01	ABAD	1981-11-17
4	SANCHEZ	1981-02-20	IGLESIAS	1981-05-01
5	ALVAREZ	1981-02-22	IGLESIAS	1981-05-01
6	GARCIA	1981-06-09	ABAD	1981-11-17

37) Para cada departamento, muestra los empleados y empleadas que trabajan en proyectos controlados por el departamento. Muestra el nombre del departamento y el código de los empleados y empleadas.

Me resultó complicado observar al principio, el orden de join que precisaba:

```
1 SELECT DISTINCT d.dep_name, ep.emp_code
2 FROM department d JOIN project p ON p.dep_code = d.dep_code
3 JOIN emp_pro ep ON ep.pro_code = p.pro_code ;
```

	dep_name character varying (14)	emp_code numeric (4)
1	SALES	7499
2	SALES	7654
3	SALES	7521
4	RESEARCH	7654
5	RESEARCH	7934
6	SALES	7844

38) Obtén el código del empleado o empleada, el nombre, el salario, el código del proyecto y las horas que le dedica cada persona vinculada a algún proyecto.

```
1 SELECT emp.emp_code, e.emp_name, e.salary, emp.pro_code, emp.hours
2 FROM emp_pro emp JOIN employee e ON emp.emp_code=e.emp_code
```

	emp_code numeric (4)	emp_name character varying (10)	salary numeric (7,2)	pro_code numeric (4)	hours numeric (2)
1	7499	SANCHEZ	1600.00	1004	15
2	7499	SANCHEZ	1600.00	1005	12
3	7521	ALVAREZ	1250.00	1004	10
4	7521	ALVAREZ	1250.00	1008	8
5	7654	ENRIQUEZ	1250.00	1001	16
6	7654	ENRIQUEZ	1250.00	1006	15
7	7654	ENRIQUEZ	1250.00	1008	5
8	7844	NADAL	1500.00	1005	6
9	7934	VIDAL	1300.00	1001	4

39) ¿Cuántas personas hay en cada departamento, y cuál es la media del salario de cada uno? Indique el nombre del departamento para clarificar el resultado.

Count en emp\_code como detalle a destacar, pues obtenía error con count(\*). Si quisiese todos los departamentos, haría right join, de forma que tendría 0 empleados y salario medio nulo:

```
1 SELECT dep_name, count(emp_code), round(avg(salary))
2 FROM employee e JOIN department d ON e.dep_code=d.dep_code
3 GROUP BY e.dep_code, dep_name
```

	dep_name	count	round
	character varying (14)	bigint	numeric
1	ACCOUNTING	3	2917
2	SALES	6	1567
3	RESEARCH	5	2175

```
1 SELECT d.dep_code, d.dep_name, count(emp_code), avg(salary) --NO SIRVE count(*)
2 FROM employee e RIGHT JOIN department d ON d.dep_code = e.dep_code
3 GROUP BY d.dep_code, d.dep_name;
```

	dep_code	dep_name	count	avg
	[PK] numeric (2)	character varying (14)	bigint	numeric
1	10	ACCOUNTING	3	2916.6666666666666667
2	40	OPERATIONS	0	[null]
3	20	RESEARCH	5	2175.0000000000000000
4	30	SALES	6	1566.6666666666666667

40) Muestra la suma de salarios de los empleados y empleadas de cada departamento que tienen un salario superior al salario medio de la empresa. Muestra el nombre del departamento.

```
1 SELECT dep_name, sum(salary)
2 FROM employee e JOIN department d ON e.dep_code=d.dep_code
3 WHERE salary > (SELECT avg(salary)
4 FROM employee)
5 GROUP BY e.dep_code, dep_name
```

	dep_name	sum
	character varying (14)	numeric
1	SALES	2850.00
2	ACCOUNTING	7450.00
3	RESEARCH	8975.00

41) Para cada proyecto controlado por el departamento 30, indica su número, ciudad, número de personas participantes en el proyecto, las horas dedicadas por el empleado o empleada que más ha trabajado, y las horas dedicadas por el que menos ha trabajado, y la diferencia entre ellas.

En principio es adecuado, pero sobra el join con department. Sólo es necesario entre emp y p:


```
1 SELECT emp.pro_code, p.location, count(emp.emp_code), MAX(hours), MIN(hours), MAX(hours)-MIN(hours)
2 FROM emp_pro emp JOIN project p ON emp.pro_code=p.pro_code
3 JOIN department d ON p.dep_code=d.dep_code
4 WHERE p.dep_code = 30
5 GROUP BY emp.pro_code, p.location
```

Data Output		Explain	Messages	Notifications		
	pro_code numeric (4)	location character varying (13)	count bigint	max numeric	min numeric	?column? numeric
1	1004	LUGO	2	15	10	5
2	1005	LUGO	2	12	6	6
3	1006	SANTANDER	1	15	15	0
4	1008	MADRID	2	8	5	3

42) Por cada departamento muestra el total de horas trabajadas en proyectos por los empleados y empleadas de cada puesto de trabajo. Muestra el código de departamento y el nombre del puesto de trabajo.

Planteamiento:

```
1 SELECT e.dep_code, job, sum (hours)
2 FROM employee e JOIN emp_pro ep ON e.emp_code = ep.emp_code
3 GROUP BY e.dep_code, job;
```

Data Output		Explain	Messages	Notifications
 dep_code <small>numeric (2)</small>	job <small>character varying (10)</small>	sum <small>numeric</small>		
1	30 SALESMAN	87		
2	10 CLERK	4		

43) Para cada empleado o empleada muestra su nombre y el número de personas a la que supervisa para cada puesto de trabajo

```
1 SELECT j.emp_name as jefe, e.job, count(e.emp_code)
2 FROM employee j JOIN employee e ON j.emp_code=e.manager
3 GROUP BY j.emp_name, e.job
```

	jefe	job	count
	character varying (10)	character varying (10)	bigint
1	IGLESIAS	SALESMAN	4
2	IGLESIAS	CLERK	1
3	ABAD	MANAGER	3
4	GONZALEZ	CLERK	1
5	LOPEZ	ANALYST	2
6	GARCIA	CLERK	1
7	MANZANO	CLERK	1

Aquí lo peculiar es que pongo los jefes a izquierda y empleados a la derecha mediante join clásico

44) Muestra, para cada proyecto, cuántas horas trabajan en total todos los empleados y empleadas que tienen un salario superior al salario medio de la empresa. Muestra el nombre del proyecto.

```
1 SELECT pro_name, sum(hours)
2 FROM employee e JOIN emp_pro emp ON e.emp_code=emp.emp_code
3 JOIN project p ON emp.pro_code=p.pro_code
4 WHERE salary > (SELECT avg(salary)
5                 FROM employee)
6 GROUP BY pro_name
```

	pro_name	sum
	character varying (10)	numeric

45) Para cada departamento, muestra su código, su nombre, el salario más alto y más bajo que cobran sus empleados y empleadas, la diferencia entre estos dos salarios, y el número de proyectos a cargo del departamento

```
1 SELECT d.dep_code, d.dep_name, max(salary), min(salary), max(salary) - min(salary), count(distinct p.pro_code)
2 FROM project p JOIN department d ON p.dep_code = d.dep_code JOIN employee e ON e.dep_code = d.dep_code
3 GROUP BY d.dep_code, d.dep_name;
```

	dep_code	dep_name	max	min	?column?	count
	[PK] numeric (2)	character varying (14)	numeric	numeric	numeric	bigint
1	20	RESEARCH	3000.00	800.00	2200.00	1
2	30	SALES	2850.00	950.00	1900.00	4

46) Considerando únicamente los empleados y empleadas con salario menor de 5000, halla la media de los salarios de los departamentos cuyo salario mínimo supera a 900. Muestra también el código y el nombre de los departamentos

```

1 SELECT d.dep_code, d.dep_name, avg(salary)
2 FROM employee e JOIN department d ON e.dep_code=d.dep_code
3 WHERE salary < 5000
4 GROUP BY d.dep_code, d.dep_name
5 HAVING MIN(salary)>900

```

	dep_code [PK] numeric (2)	dep_name character varying (14)	avg numeric
1	10	ACCOUNTING	1875.0000000000000000
2	30	SALES	1566.6666666666666667

47) Lista los empleados y empleadas que tengan el mayor salario de su departamento, mostrando su nombre, su salario y el nombre del departamento.

```

1 SELECT e.emp_name, e.salary, d.dep_name
2 FROM employee e JOIN department d ON e.dep_code=d.dep_code
3 WHERE salary IN (SELECT MAX(salary)
4 FROM employee
5 GROUP BY dep_code)
6 GROUP BY e.emp_name, e.salary, d.dep_name

```

	emp_name character varying (10)	salary numeric (7,2)	dep_name character varying (14)
1	GONZALEZ	3000.00	RESEARCH
2	ABAD	5000.00	ACCOUNTING
3	IGLESIAS	2850.00	SALES
4	MANZANO	3000.00	RESEARCH

La otra posibilidad es plantearlo como una subconsulta correlacionada:

```

1 SELECT e.emp_name, e.salary, d.dep_name
2 FROM employee e JOIN department d ON d.dep_code = e.dep_code
3 WHERE e.salary = (
4 SELECT max(salary)
5 FROM employee
6 WHERE dep_code = e.dep_code
7 );

```

	emp_name character varying (10)	salary numeric (7,2)	dep_name character varying (14)
1	ABAD	5000.00	ACCOUNTING
2	GONZALEZ	3000.00	RESEARCH
3	MANZANO	3000.00	RESEARCH
4	IGLESIAS	2850.00	SALES

#### 48) El puesto de trabajo con el salario medio más alto. (Se pone HAVING)

```
1 SELECT job, MAX(salary)
2 FROM employee
3 WHERE salary IN (SELECT avg(salary)
4                  FROM employee
5                  GROUP BY job)
6 GROUP BY job
```

	job character varying (10)	max numeric
1	PRESIDENT	5000.00
2	ANALYST	3000.00

Era preciso incluir HAVING como en la consulta 51:

```
1 SELECT job
2 FROM employee
3 GROUP BY job
4 HAVING avg(salary) >= ALL (
5     SELECT avg(salary)
6     FROM employee
7     GROUP BY job
8 )
```

job character varying (10)
1 PRESIDENT

#### 49) Para cada supervisor, muestra los datos del subordinado(s) que más gana.

MAL:

```
1 SELECT j.emp_name, e.emp_name
2 FROM employee j JOIN employee e ON j.emp_code=j.manager
3 WHERE e.salary IN (SELECT MAX(salary)
4                  FROM employee e)
5 GROUP BY j.emp_name, e.emp_name
```

emp_name character varying (10)	emp_name character varying (10)
------------------------------------	------------------------------------

Precisaba incluir where j.emp\_code=manager al seleccionar salario:

```
1 SELECT j.emp_code, j.emp_name, e.emp_name, e.salary
2 FROM employee j JOIN employee e ON j.emp_code = e.manager
3 WHERE e.salary = (SELECT max(salary)
4                  FROM employee
5                  WHERE j.emp_code = manager)
6
```

emp_code [PK] numeric (4)	emp_name character varying (10)	emp_name character varying (10)	salary numeric (7,2)
1 7839	ABAD	LOPEZ	2975.00
2 7566	LOPEZ	GONZALEZ	3000.00
3 7902	GONZALEZ	RAMOS	800.00
4 7698	IGLESIAS	SANCHEZ	1600.00
5 7566	LOPEZ	MANZANO	3000.00
6 7788	MANZANO	SAAVEDRA	1100.00
7 7782	GARCIA	VIDAL	1300.00



50) Deseamos saber a cuántos empleados y empleadas supervisa cada responsable. Para ello, obtén un listado en el que se reflejen el código y el nombre de cada responsable, junto al número de personas a las que supervisa directamente. Como puede haber empleados y empleadas sin responsable, para estos se indicará sólo el número de ellos, y los valores restantes (código y nombre del responsable) se dejarán como nulos.

Tendría el siguiente aspecto, donde agrupo por código y nombre del jefe haciendo posteriormente el recuento:

```

1 SELECT j.emp_code, j.emp_name, count (e.emp_code)
2 FROM employee e LEFT JOIN employee j ON e.manager=j.emp_code
3 GROUP BY j.emp_code, j.emp_name

```

	emp_code [PK] numeric (4)	emp_name character varying (10)	count bigint
1	[null]	[null]	1
2	7566	LOPEZ	2
3	7839	ABAD	3
4	7698	IGLESIAS	5
5	7782	GARCIA	1
6	7902	GONZALEZ	1
7	7788	MANZANO	1

51) Hallar el/los departamento(s) cuya suma de salarios sea la más alta, mostrando esta suma de salarios y el nombre del departamento.

```

1 SELECT dep_name, SUM(salary)
2 FROM employee e JOIN department d ON e.dep_code=d.dep_code
3 GROUP BY dep_name
4 ORDER BY SUM(salary) desc

```

	dep_name character varying (14)	sum numeric
1	RESEARCH	10875.00
2	SALES	9400.00
3	ACCOUNTING	8750.00

Obtenía el resultado anterior, por lo que incluyo la resolución a continuación, donde era preciso poner un having para que fuese superior al general:

```

1 SELECT dep_name, sum(salary)
2 FROM department d JOIN employee e ON d.dep_code = e.dep_code
3 GROUP BY d.dep_code, d.dep_name
4 HAVING sum(salary) >= ALL (
5     SELECT sum(salary)
6     FROM employee
7     GROUP BY dep_code
8 )

```

	dep_name character varying (14)	sum numeric
1	RESEARCH	10875.00