



Tarea Bases de Datos Relaciones

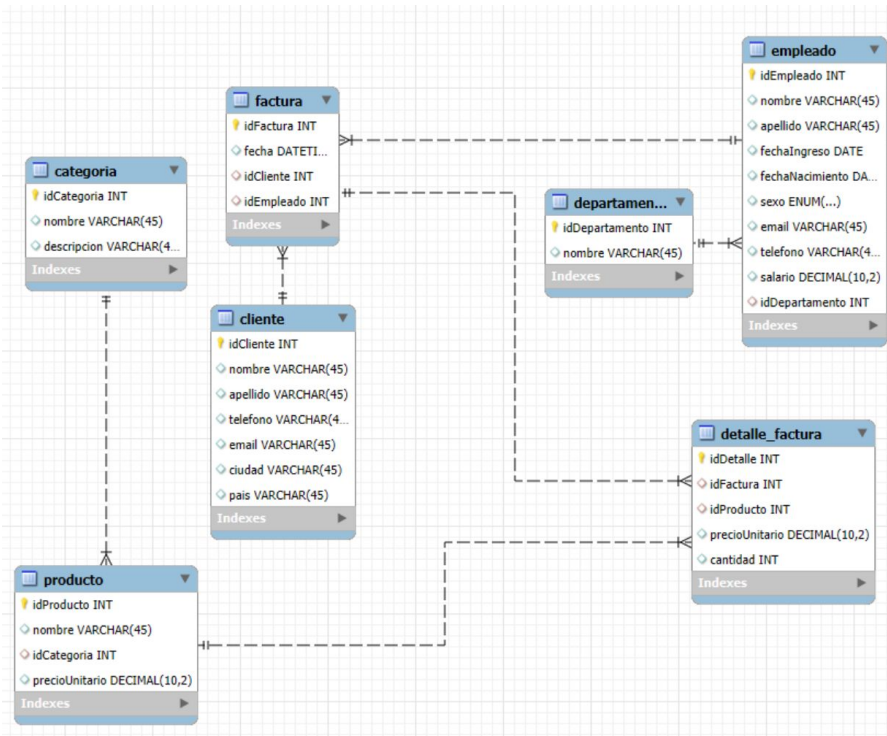
Andrés Vega Zamora



Parte 1

Base de datos - COMPRAS

Modelo Relacional - Database Compras



En este modelo relacional se identifican las tablas: categoría, factura, empleado, cliente, departamento, producto y detalle_factura.

Las descripciones de los Primary Keys y Foreign Keys de dichas tablas son las siguientes:

categoria: idCategoria (PK, FK)

producto: idProducto (PK, FK), idCategoria (FK)

detalle_factura: idDetalle (PK), idFactura (FK), idProducto (FK)

factura: idFactura (PK, FK), idCliente (FK), idEmpleado (FK)

cliente: idCliente (PK, FK)

empleado: idEmpleado (PK, FK), idDepartamento (FK)

departamento: idDepartamento (PK,FK)

Modelo Entidad Relación



Entidades:

- **Cliente:** Con atributos como nombre, apellido, teléfono, etc.
- **Producto:** Con atributos como nombre, precio unitario y categoría.
- **Factura:** Con atributos como fecha y cliente asociado.
- **Empleado:** Con atributos como nombre, apellido, departamento, etc.
- **Departamento:** Con atributos como nombre.
- **Detalle_factura:** Con atributos como cantidad y producto asociado a cada línea de factura.
- **Categoría:** Con atributos como descripción y nombre..

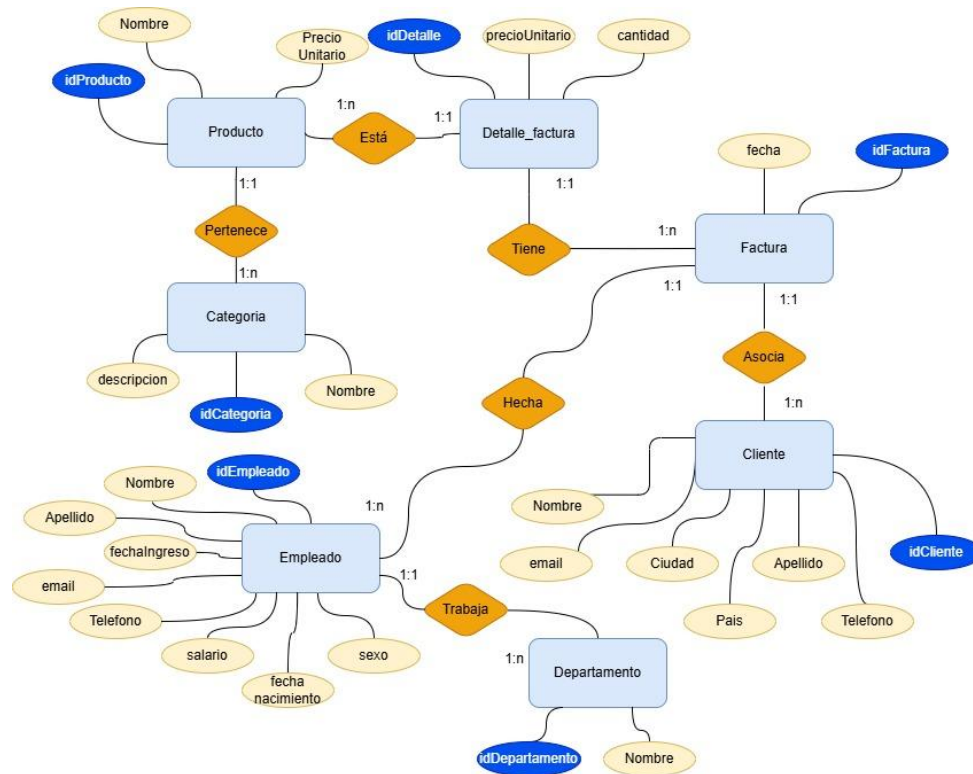
Modelo Entidad Relación



Relaciones:

- **Cliente - Factura:** Un cliente puede tener muchas facturas, pero una factura solo pertenece a un cliente (relación uno a muchos).
- **Empleado - Factura:** Un empleado puede generar muchas facturas, pero una factura solo puede ser generada por un empleado (relación uno a muchos).
- **Empleado - Departamento:** Un empleado pertenece a un solo departamento, pero un departamento puede tener muchos empleados (relación uno a muchos).
- **Producto - Categoría:** Un producto pertenece a una sola categoría, pero una categoría puede tener muchos productos (relación uno a muchos).
- **Detalle_factura - Factura:** Cada detalle de una factura pertenece a una única factura, y una factura puede tener muchos detalles (relación uno a muchos).
- **Detalle_factura - Producto:** Cada detalle de una factura corresponde a un único producto, y un producto puede aparecer en muchos detalles de factura (relación uno a muchos).

Modelo Entidad Relación



¿Qué es el error 1451 y en qué circunstancias se produce?



El error 1451 indica que no es posible eliminar o modificar una fila padre.

En el ejemplo se presenta si se intenta eliminar las filas de la tabla “empleado” donde el “idEmpleado” sea 5.

Sin embargo, no se puede eliminar porque en la tabla “Factura”, existen filas hijas que hacen referencia a ese “idEmpleado”. Eliminar las filas haría que dichas referencias sean inválidas.

¿Que operadores lógicos podemos utilizar en una consulta?



Los operadores lógicos disponibles en SQL son:

- AND: Devuelve verdadero si ambas expresiones son verdaderas
- OR: Devuelve verdadero si al menos una de las expresiones es verdadera
- NOT: Devuelve el valor invertido de verdad
- XOR: Devuelve verdadero si solamente una de dos condiciones es verdadera

Resultados totales en una consulta: Total de una factura



```
SELECT IDFACTURA, SUM(PRECIOUNITARIO *  
CANTIDAD) AS TOTAL
```

```
FROM DETALLE_FACTURA GROUP BY IDFACTURA
```

Esta consulta multiplica el precio unitario por la cantidad de elementos de cada línea del detalle de la factura y luego las suma y agrupa por el campo IDFactura

	IDFACTURA	TOTAL
▶	1	19.76
	2	4.70
	3	20.00
	4	1.80
	5	26.12
	6	22.20
	7	24.20
	8	18.40
	9	12.10
	10	10.50
	11	9.72
	12	12.80
	13	14.00
	14	14.60
	15	7.20
	16	14.62
	17	0.50
	18	17.00
	19	13.30
	20	6.30

¿Para qué sirve la sentencia GROUP BY?

Como su nombre lo indica, el Group By sirve para agrupar los resultados de un query a una base de datos según algún campo de elección.

Por ejemplo, el query:

```
select idDepartamento, count(idEmpleado) from empleado group by idDepartamento;
```

Cuenta la cantidad de empleados y los agrupa por departamento.

	idDepartamento	count(idEmpleado)
▶	1	2
	2	2
	3	1

¿Cuál es la diferencia entre INNER JOIN, LEFT JOIN, RIGHT JOIN, CROSS JOIN, NATURAL JOIN?



- **INNER JOIN:** Solo las filas coincidentes.
- **LEFT JOIN:** Todas las filas de la izquierda, incluyendo las que no tienen coincidencias en la derecha.
- **RIGHT JOIN:** Todas las filas de la derecha, incluyendo las que no tienen coincidencias en la izquierda.
- **CROSS JOIN:** Producto cartesiano de todas las filas.
- **NATURAL JOIN:** INNER JOIN basado en nombres de columna comunes.

¿Para qué sirven las SUBQUERIES?

Funcionan para filtrar los resultados de una consulta al compararlos con otra consulta, y así lograr consultas más complejas

Ejemplo:

```
SELECT * FROM FACTURA WHERE IDCLIENTE IN (SELECT IDCLIENTE FROM CLIENTE WHERE ciudad = 'Quebec');
```

Este ejemplo devuelve las filas de la tabla factura de los clientes cuya ciudad es quebec

	idFactura	fecha	idCliente	idEmpleado
▶	5	2017-03-07 00:00:00	7	3
	9	2017-11-09 00:00:00	7	3
	15	2019-01-22 00:00:00	7	3
*	NULL	NULL	NULL	NULL

TRIGGER de respaldo para las facturas y líneas de factura



```
DELIMITER $$
```

```
CREATE TRIGGER facturas_backup_trigger AFTER INSERT ON  
factura
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    INSERT INTO factura_backup (idFactura, fecha,  
                                idCliente, idEmpleado)
```

```
    VALUES (NEW.idFactura, NEW.fecha, NEW.idCliente,  
            NEW.idEmpleado);
```

```
END;
```

```
$$
```

```
DELIMITER ;
```

```
DELIMITER $$
```

```
CREATE TRIGGER detalle_factura_backup_trigger AFTER  
INSERT ON detalle_factura
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    INSERT INTO detalle_factura_backup (idDetalle,  
                                         idFactura, idProducto, precioUnitario, cantidad)
```

```
    VALUES (NEW.idDetalle, NEW.idFactura,  
            NEW.idProducto, NEW.precioUnitario,  
            NEW.cantidad);
```

```
END;
```

```
$$
```


```
DELIMITER ;
```

VIEW: Precio medio por categorías



```
CREATE VIEW precio_medio_categoria AS
SELECT
    c.nombre AS categoria, AVG(p.precioUnitario) AS precio_medio
FROM
    producto p
INNER JOIN
    categoria c ON p.idCategoria = c.idCategoria
GROUP BY
    c.nombre;
```

PROCEDURE: 3, más de 3 o menos de 3 productos en venta



```
DELIMITER $$
CREATE PROCEDURE productos_por_venta(IN id_factura INT)
BEGIN
    DECLARE cantidad_productos INT;
    SELECT SUM(cantidad) INTO cantidad_productos
    FROM detalle_factura
    WHERE idFactura = id_factura;
    IF cantidad_productos = 3 THEN
        SELECT 'Se vendieron exactamente 3 productos';
    ELSEIF cantidad_productos > 3 THEN
        SELECT 'Se vendieron más de 3 productos';
    ELSE
        SELECT 'Se vendieron menos de 3 productos';
    END IF;
END $$
DELIMITER ;
```



Parte 2

Base de datos - EMPRESA

1.Creación de la base de datos



```
DROP DATABASE IF EXISTS EMPRESA;  
  
CREATE DATABASE EMPRESA;  
  
USE EMPRESA;  
  
CREATE TABLE departamento (  
    codDepto VARCHAR(4) PRIMARY  
    KEY,  
    nombreDpto VARCHAR(20) NOT  
    NULL,  
    Ciudad VARCHAR(15),  
    codDirector VARCHAR(12)  
);
```

```
CREATE TABLE empleado (  
    nDIEmp VARCHAR(12) PRIMARY KEY,  
    nomEmp VARCHAR(30),  
    sexEmp CHAR(1),  
    fecNac DATE,  
    fecIncorporacion DATE,  
    salEmp FLOAT,  
    comisionE FLOAT,  
    cargoE VARCHAR(15),  
    jefeID VARCHAR(12),  
    codDepto VARCHAR(4) NOT NULL,  
    FOREIGN KEY (codDepto) REFERENCES  
    departamento(codDepto),  
    FOREIGN KEY (jefeID) REFERENCES  
    empleado(nDIEmp)  
);
```

2. Insertar datos



```
INSERT INTO departamento (codDepto, nombreDpto,  
Ciudad, codDirector)
```

VALUES

```
('1000', 'VENTAS', 'MADRID', 1),
```

```
('1500', 'INVESTIGACIÓN', 'MADRID', 2),
```

...

```
('5500', 'FINANZAS', 'BARCELONA', 10);
```

```
INSERT INTO empleado (nDIEmp, nomEmp, sexEmp, fecNac,  
fecIncorporacion, salEmp, comisionE, cargoE, jefeID, codDepto)
```

VALUES

```
('31.840.269', 'Homer Simpson', 'M', '1982-05-17',  
'2021-03-17', 52345, 12345, 'Jefe', NULL, 1000),
```

```
('31.840.270', 'Marge Bouvier', 'F', '1985-11-03',  
'2023-11-09', 68912, 28912, 'Jefe', NULL, 1500),
```

...

```
('31.840.308', 'Homer Simpson', 'M', '1994-09-11',  
'2022-09-10', 43543, 28765, 'Senior', '31.840.278',  
5500);
```

3. Datos completos de los empleados

SELECT * FROM empleado

	nDIEmp	nomEmp	sexEmp	fecNac	fecIncorporacion	salEmp	comisionE	cargoE	jefeID	codDepto
▶	31.840.269	Homer Simpson	M	1982-05-17	2021-03-17	52345	12345	Jefe	NULL	1000
	31.840.270	Marge Bouvier	F	1985-11-03	2023-11-09	68912	28912	Jefe	NULL	1500
	31.840.271	Bart Simpson	M	1992-07-29	2019-07-25	57567	17567	Jefe	NULL	2000
	31.840.272	Lisa Simpson	F	1988-02-14	2022-02-11	64123	24123	Jefe	NULL	2500
	31.840.273	Maggie Simpson	F	1995-12-25	2020-12-22	59876	19876	Jefe	NULL	3000
	31.840.274	Ned Flanders	M	1981-09-21	2018-09-20	62098	22098	Jefe	NULL	3500
	31.840.275	Seymour Skinner	M	1998-04-06	2024-04-04	55432	15432	Jefe	NULL	4000
	31.840.276	Montgomery Burns	M	1987-03-19	2021-03-18	66789	26789	Jefe	NULL	4500
	31.840.277	Waylon Smithers	M	1990-10-12	2019-10-10	50123	18123	Jefe	NULL	5000
	31.840.278	Moe Szyslak	M	1989-06-05	2022-06-03	69543	29543	Jefe	NULL	5500
	31.840.279	Apu Nahasapeem...	M	1993-01-31	2020-01-30	18567	21789	Secre...	31.8...	1000
	31.840.280	Barney Gumble	M	1984-08-28	2018-08-26	23912	16543	Secre...	31.8...	1500
	31.840.281	Krusty el Payaso	M	1997-11-15	2023-11-13	19567	23123	Secre...	31.8...	2000
	31.840.282	Milhouse Van Hou...	M	1986-04-10	2021-04-09	21123	19876	Secre...	31.8...	2500
	31.840.283	Nelson Muntz	M	1999-09-02	2019-09-01	20876	25432	Secre...	31.8...	3000
	31.840.284	Otto Mann	M	1983-07-04	2022-07-02	22098	20123	Secre...	31.8...	3500
	31.840.285	Comic Book Guy	M	1996-02-29	2020-02-28	17432	27890	Secre...	31.8...	4000
	31.840.286	Dr. Hibbert	M	1980-12-31	2018-12-30	24789	13567	Secre...	31.8...	4500
	31.840.287	Clancy Wiggum	M	1991-05-05	2023-05-04	18123	22456	Secre...	31.8...	5000
	31.840.288	Abraham Simpson	M	1994-08-18	2021-08-17	23543	18901	Secre...	31.8...	5500
	31.840.289	Mona Simpson	F	1982-11-22	2019-11-21	28567	29345	Junior	31.8...	1000
	31.840.290	Jacqueline Bouvier	F	1985-03-07	2022-03-06	31912	17654	Junior	31.8...	1500
	31.840.291	Patty Bouvier	F	1992-12-19	2020-12-18	29567	24734	Junior	31.8...	2000

4. Datos completos de los departamentos

SELECT * FROM departamento

	codDepto	nombreDpto	Ciudad	codDirector
▶	1000	VENTAS	MADRID	1
	1500	INVESTIGACIÓN	MADRID	2
	2000	MANTENIMIENTO	MADRID	3
	2500	PRODUCCIÓN	MADRID	4
	3000	MARKETING	VALENCIA	5
	3500	CONTABILIDAD	VALENCIA	6
	4000	RRHH	VALENCIA	7
	4500	IT	VALENCIA	8
	5000	LOGÍSTICA	BARCELONA	9
	5500	FINANZAS	BARCELONA	10
✱	NULL	NULL	NULL	NULL

5. Datos de los empleados con cargo 'Secretario'

```
SELECT * FROM empleado WHERE cargoE like 'Secretari_'
```

[illegible]

6. Nombre y salario de los empleados

SELECT nomEmp, salEmp FROM empleado

	nomEmp	salEmp
▶	Homero Simpson	52345
	Marge Bouvier	68912
	Bart Simpson	57567
	Lisa Simpson	64123
	Maggie Simpson	59876
	Ned Flanders	62098
	Seymour Skinner	55432
	Montgomery Burns	66789
	Waylon Smithers	50123
	Moe Szyslak	69543
	Apu Nahasapeem...	18567
	Barney Gumble	23912
	Krusty el Payaso	19567
	Millhouse Van Hou...	21123
	Nelson Muntz	20876
	Otto Mann	22098
	Comic Book Guy	17432
	Dr. Hibbert	24789
	Clancy Wiggum	18123
	Abraham Simpson	23543
	Mona Simpson	28567
	Jacqueline Bouvier	31912

7. Datos de los vendedores, por nombre

```
SELECT * FROM empleado WHERE
```

```
(codDepto = (SELECT codDepto FROM departamento WHERE nombreDpto = 'VENTAS'))
```

AND (cargoE = 'Junior' OR cargoE= 'Senior')

ORDER BY nomEmp;

[illegible]

8. Listar departamentos por nombre y ciudad, ascendente y descendente

`SELECT nombreDpto, Ciudad FROM departamento ORDER BY nombreDpto ASC, Ciudad DESC;`

	nombreDpto	Ciudad
►	CONTABILIDAD	VALENCIA
	FINANZAS	BARCELONA
	INVESTIGACIÓN	MADRID
	IT	VALENCIA
	LOGÍSTICA	BARCELONA
	MANTENIMIENTO	MADRID
	MARKETING	VALENCIA
	PRODUCCIÓN	MADRID
	RRHH	VALENCIA
	VENTAS	MADRID

9. Nombre y cargo de empleados, ordenado por cargo y salario


```
SELECT nomEmp, cargoE FROM empleado ORDER BY cargoE, SalEmp;
```

	nomEmp	cargoE
►	Waylon Smithers	Jefe
	Homero Simpson	Jefe
	Seymour Skinner	Jefe
	Bart Simpson	Jefe
	Maggie Simpson	Jefe
	Ned Flanders	Jefe
	Lisa Simpson	Jefe
	Montgomery Burns	Jefe
	Marge Bouvier	Jefe
	Moe Szyslak	Jefe
	Jimbo Jones	Junior
	Duffman	Junior

	Duffman	Junior
	Mona Simpson	Junior
	Waylon Smithers Jr.	Junior
	Patty Bouvier	Junior
	Selma Bouvier	Junior
	Mayor Quimby	Junior
	Dolph Starbeam	Junior
	Jacqueline Bouvier	Junior
	Kearney Zyzwicz	Junior
	Comic Book Guy	Secre...
	Clancy Wiggum	Secre...
	Apu Nahasapeem...	Secre...
	Krusty el Payaso	Secre...
	Nelson Muntz	Secre...
	Milhouse Van Hou...	Secre...
	Otto Mann	Secre...

	Otto Mann	Secre...
	Abraham Simpson	Secre...
	Barney Gumble	Secre...
	Dr. Hibbert	Secre...
	Disco Stu	Senior
	Sideshow Bob	Senior
	Hans Moleman	Senior
	Agnes Skinner	Senior
	Bumblebee Man	Senior
	Krusty's Sidekick	Senior
	Reverendo Lovejoy	Senior
	Homero Simpson	Senior
	Captain McCallister	Senior
	Groundskeeper W...	Senior

10. Departamento cuya suma de salarios se la más alta



```
SELECT d.nombreDpto, SUM(e.salEmp) AS total_salario  
FROM departamento d INNER JOIN empleado e ON d.codDepto = e.codDepto  
GROUP BY d.nombreDpto  
ORDER BY total_salario DESC LIMIT 1;
```

	nombreDpto	total_salario
►	INVESTIGACIÓN	170648

11. Salarios y comisiones de los empleados del departamento 2000, ordenado por comisión

```
SELECT nomEmp, salEmp, comisionE FROM empleado WHERE codDepto = 2000 ORDER BY comisionE;
```

	nomEmp	salEmp	comisionE
►	Bart Simpson	57567	17567
	Agnes Skinner	39567	18901
	Krusty el Payaso	19567	23123
	Patty Bouvier	29567	24234

12. Listar todas las comisiones que sean diferentes, ordenada por valor.

```
SELECT DISTINCT comisionE FROM empleado  
ORDER BY comisionE;
```

comisionE	
11234	21678
12345	21789
13567	22098
14567	22456
15432	23123
16543	23456
17567	24123
17654	24234
18123	25345
18901	25432
19012	26543
19876	26789
20123	27890
20987	28765
	28912
	29345
	29543

13. Listar los diferentes salarios

`SELECT DISTINCT salEmp FROM empleado;`

	salEmp
▶	52345
	68912
	57567
	64123
	59876
	62098
	55432
	66789
	50123
	69543
	18567
	23912
	19567
	21123
	20876
	22098
	17432
	24789

	18123
	23543
	28567
	31912
	29567
	30123
	30876
	32098
	27432
	31789
	28123
	29543
	38567
	45912
	39567
	42123
	40876
	41098
	37432
	44789
	38123
	43543

14. Valor total de suma de salario + 5000 euros a empleados de departamento 3000 ordenado por nombre

```
SELECT nomEMP, (salEmp + 5.000) AS Salario_bonificado FROM empleado WHERE codDepto = 3000  
ORDER BY nomEmp;
```

	nomEMP	Salario_bonificado
►	Bumblebee Man	45876
	Maggie Simpson	64876
	Mayor Quimby	35876
	Nelson Muntz	25876

15. Empleados que ganan una comisión superior a su sueldo.

`SELECT nomEmp, salEmp, comisionE FROM empleado WHERE comisionE > salEmp;`

	nomEmp	salEmp	comisionE
►	Apu Nahasapeemapetilon	18567	21789
	Krusty el Payaso	19567	23123
	Nelson Muntz	20876	25432
	Comic Book Guy	17432	27890
	Clancy Wiggum	18123	22456
	Mona Simpson	28567	29345
	Jimbo Jones	27432	28765

16. Empleados cuya comisión es menor o igual que el 30% de su sueldo.

`SELECT nomEmp, salEmp, comisionE FROM empleado WHERE comisionE <= salEmp*0.3;`

	nomEmp	salEmp	comisionE
▶	Homero Simpson	52345	12345
	Seymour Skinner	55432	15432
	Sideshow Bob	38123	11234

17. Empleados cuyo salario es menor o igual que el 40% de su comisión.

`SELECT nomEmp, salEmp, comisionE FROM empleado WHERE comisionE*0.4 >= salEmp;`

	nomEmp	salEmp	comisionE

18. Salario, comisión, el salario total, id y nombre, de empleados con comisión superior a 10.000 €, ordenar por id.

```
SELECT nomEmp, nDIEmp, salEmp, comisionE, (salEmp + comisionE) as Salario_total FROM empleado  
WHERE comisionE > 10000 ORDER BY nDIEmp;
```

	nomEmp	nDIEmp	salEmp	comisionE	Salario_total
▶	Homer Simpson	31.840.269	52345	12345	64690
	Marge Bouvier	31.840.270	68912	28912	97824
	Bart Simpson	31.840.271	57567	17567	75134
	Lisa Simpson	31.840.272	64123	24123	88246
	Maggie Simpson	31.840.273	59876	19876	79752
	Ned Flanders	31.840.274	62098	22098	84196
	Seymour Skinner	31.840.275	55432	15432	70864
	Montgomery Burns	31.840.276	66789	26789	93578
	Waylon Smithers	31.840.277	50123	18123	68246
	Moe Szyslak	31.840.278	69543	29543	99086
	Apu Nahasapeem...	31.840.279	18567	21789	40356
	Barney Gumble	31.840.280	23912	16543	40455
	Krusty el Payaso	31.840.281	19567	23123	42690
	Milhouse Van Hou...	31.840.282	21123	19876	40999
	Nelson Muntz	31.840.283	20876	25432	46308
	Otto Mann	31.840.284	22098	20123	42221
	Comic Book Guy	31.840.285	17432	27890	45322
	Dr. Hibbert	31.840.286	24789	13567	38356
	Clancy Wiggum	31.840.287	18123	22456	40579
	Abraham Simpson	31.840.288	23543	18901	42444
	Mona Simpson	31.840.289	28567	29345	57912

	Jacqueline Bouvier	31.840.290	31912	17654	49566
	Patty Bouvier	31.840.291	29567	24234	53801
	Selma Bouvier	31.840.292	30123	20987	51110
	Mayor Quimby	31.840.293	30876	26543	57419
	Kearney Zyzwicz	31.840.294	32098	11234	43332
	Jimbo Jones	31.840.295	27432	28765	56197
	Dolph Starbeam	31.840.296	31789	19012	50801
	Duffman	31.840.297	28123	25345	53468
	Waylon Smithers Jr.	31.840.298	29543	21678	51221
	Hans Moleman	31.840.299	38567	14567	53134
	Groundskeeper W...	31.840.300	45912	23456	69368
	Agnes Skinner	31.840.301	39567	18901	58468
	Reverendo Lovejoy	31.840.302	42123	29345	71468
	Bumblebee Man	31.840.303	40876	17654	58530
	Krusty's Sidekick	31.840.304	41098	24234	65332
	Disco Stu	31.840.305	37432	20987	58419
	Captain McCallister	31.840.306	44789	26543	71332
	Sideshow Bob	31.840.307	38123	11234	49357
	Homer Simpson	31.840.308	43543	28765	72308

19. Empleados que tienen un salario superior a 50.000 €, y tienen como jefe al empleado con documento de identidad '31.840.269'.

```
SELECT nomEmp, salEmp, jefeID FROM empleado WHERE (salEmp > 50000 AND jefeID = '31.940.269')
```

	nomEmp	salEmp	jefeID
--	--------	--------	--------

20. Nombres de los departamentos que no sean 'VENTAS', 'INVESTIGACIÓN', ni 'MANTENIMIENTO', ordenados por ciudad.

```
SELECT nombreDpto, Ciudad FROM departamento WHERE nombreDpto NOT IN ('VENTAS',  
INVESTIGACIÓN', 'MANTENIMIENTO') ORDER BY Ciudad;
```

	nombreDpto	Ciudad
►	LOGÍSTICA	BARCELONA
	FINANZAS	BARCELONA
	PRODUCCIÓN	MADRID
	MARKETING	VALENCIA
	CONTABILIDAD	VALENCIA
	RRHH	VALENCIA
	IT	VALENCIA

21. Datos de los empleados cuyo nombre inicia por la letra 'M', su salario es mayor a 40.000 o reciben comisión y trabajan en 'VENTAS'.

```
SELECT * FROM empleado WHERE
```

AND codDepto =

[illegible]

22. Nombre, salario y comisión de los empleados que reciben un salario entre la mitad de la comisión la propia comisión.

```
SELECT nomEmp, salEmp, comisionE FROM empleado WHERE salEmp BETWEEN comisionE/2 AND comisionE;
```

	nomEmp	salEmp	comisionE
►	Apu Nahasapeemapetilon	18567	21789
	Krusty el Payaso	19567	23123
	Nelson Muntz	20876	25432
	Comic Book Guy	17432	27890
	Clancy Wiggum	18123	22456
	Mona Simpson	28567	29345
	Jimbo Jones	27432	28765

23. Entregar el salario más alto de la empresa.



```
SELECT MAX(salEmp) FROM empleado;
```


	MAX(salEmp)
▶	69543

24. Entregar el total a pagar por comisiones, y el número de empleados que las reciben.

```
SELECT SUM(comisionE) AS Total_comisiones, count(*) AS Empleados_con_comision FROM empleado  
WHERE comisionE > 0;
```

	Total_comisiones	Empleados_con_comision
▶	864991	40

25. Hallar el salario más alto, el más bajo y la diferencia entre ellos.



```
SELECT MAX(salEmp) AS Salario_mas_alto, MIN(salEmp) AS Salario_mas_bajo,  
(MAX(salEmp) - MIN(salEmp)) AS Diferencia FROM empleado;
```


	Salario_mas_alto	Salario_mas_bajo	Diferencia
▶	69543	17432	52111

26. Número de empleados de sexo femenino y de sexo masculino, por departamento

```
SELECT d.nombreDpto AS Departamento,  
       COUNT(CASE WHEN e.sexEmp = 'F' THEN 1 ELSE NULL END) AS Femenino,  
       COUNT(CASE WHEN e.sexEmp = 'M' THEN 1 ELSE NULL END) AS Masculino  
FROM  
       empleado e  
       INNER JOIN departamento d ON e.codDepto = d.codDepto  
GROUP BY d.nombreDpto;
```

	Departamento	Femenino	Masculino
▶	VENTAS	1	3
	INVESTIGACIÓN	2	2
	MANTENIMIENTO	2	2
	PRODUCCIÓN	2	2
	MARKETING	1	3
	CONTABILIDAD	1	3
	RRHH	0	4
	IT	0	4
	LOGÍSTICA	0	4
	FINANZAS	0	4

27. Hallar el salario promedio por departamento.



```
SELECT AVG(e.salEmp) AS Salario_Promedio, d.nombreDpto AS Departamento
FROM
    empleado e
    INNER JOIN departamento d ON e.codDepto = d.codDepto
GROUP BY d.nombreDpto;
```

	Salario_Promedio	Departamento
▶	34511.5	VENTAS
	42662	INVESTIGACIÓN
	36567	MANTENIMIENTO
	39373	PRODUCCIÓN
	38126	MARKETING
	39348	CONTABILIDAD
	34432	RRHH
	42039	IT
	33623	LOGÍSTICA
	41543	FINANZAS

28. Número de cargos en cada departamento y cual es el promedio de salario de cada uno. Indicar el nombre del departamento en el resultado.

SELECT

AVG(e.salEmp) AS Salario_Promedio,

d.nombreDpto AS Departamento,

COUNT(*) AS Cargos_por_departamento

FROM

empleado e

INNER JOIN departamento d ON e.codDepto = d.codDepto

GROUP BY d.nombreDpto;

	Salario_Promedio	Departamento	Cargos_por_departamento
►	34511.5	VENTAS	4
	42662	INVESTIGACIÓN	4
	36567	MANTENIMIENTO	4
	39373	PRODUCCIÓN	4
	38126	MARKETING	4
	39348	CONTABILIDAD	4
	34432	RRHH	4
	42039	IT	4
	33623	LOGÍSTICA	4
	41543	FINANZAS	4

29. Calcular el total de salarios por departamento.

SELECT

d.nombreDpto AS Departamento,

SUM(e.salEmp) AS Total_Salarios

FROM

empleado e

INNER JOIN departamento d ON e.codDepto = d.codDepto

GROUP BY d.nombreDpto;

	Departamento	Total_Salarios
▶	VENTAS	138046
	INVESTIGACIÓN	170648
	MANTENIMIENTO	146268
	PRODUCCIÓN	157492
	MARKETING	152504
	CONTABILIDAD	157392
	RRHH	137728
	IT	168156
	LOGÍSTICA	134492
	FINANZAS	166172

30. Hallar la suma de salarios más alta, crear para ello una vista.

CREATE VIEW departamento_mayor_salario AS

SELECT

d.nombreDpto,

SUM(e.salEmp) AS Total_Salarios

FROM

empleado e

INNER JOIN departamento d ON e.codDepto = d.codDepto

GROUP BY d.nombreDpto

ORDER BY Total_salario DESC

LIMIT 1;

SELECT * FROM departamento_mayor_salario;

	nombreDpto	Total_salario
▶	INVESTIGACIÓN	170648