

# BOLETÍN DE EJERCICIOS TEMA 2

PROFESOR: MARTÍN GARCÍA FIGUEIRA  
martin@ciclosmontecastelo.com

---

## EJERCICIOS BASE DE DATOS WORLD

**NOTA:** Para hacer estos ejercicios debemos importar el script **BD\_World.sql**

La base de datos WORLD consta de tres tablas:

- **country:** Información sobre países del mundo
- **city:** Información sobre algunas ciudades de esos países
- **countrylanguage:** Lengua hablada en cada país

### Ejercicio 1

Averigua el nombre de los países que, según esta base de datos, no tienen como forma de gobierno (campo **GovernmentForm**) la República (**Republic**) o cualquier otra forma que tenga que ver con la República. El resultado debe mostrarse en dos columnas encabezadas por **País** y **Forma de Gobierno**.

```
SELECT c.Nombre as 'País', c.GovernmentForm as 'Forma de gobierno'
FROM country c
WHERE c.GovernmentForm NOT LIKE '%Republic%'
```

	País	Forma de gobierno
1	Aruba	Nonmetropolitan Territory of The Netherlands
2	Afghanistan	Islamic Emirate
3	Anguilla	Dependent Territory of the UK
4	Andorra	Parliamentary Coprincipality
5	Netherlands Antilles	Nonmetropolitan Territory of The Netherlands
6	United Arab Emirates	Emirate Federation
7	American Samoa	US Territory
8	Antarctica	Co-administrated
9	French Southern territories	Nonmetropolitan Territory of France
10	Antigua and Barbuda	Constitutional Monarchy
11	Australia	Constitutional Monarchy, Federation
12	Belgium	Constitutional Monarchy, Federation
13	Bahrain	Monarchy (Emirate)
14	Bahamas	Constitutional Monarchy
15	Belize	Constitutional Monarchy
16	Bermuda	Dependent Territory of the UK
17	Barbados	Constitutional Monarchy
18	Brunei	Monarchy (Sultanate)

## Ejercicio 2

Obtén un listado de todos los continentes que aparecen en esta base de datos especificando para cada uno de ellos cuántos países le pertenecen.

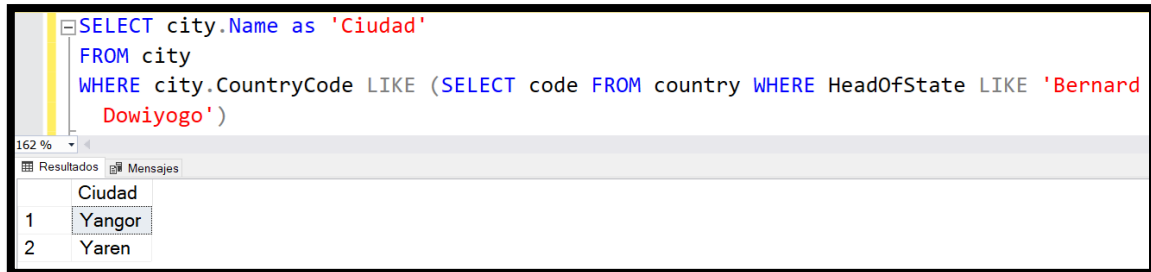
El resultado se debe mostrar en dos columnas, **Continente**, **Nº de Países**, ordenados por orden alfabético de continente.

```
SELECT continent as 'Continente', COUNT(nombre) as 'Nº de países'
FROM country
GROUP BY Continent
ORDER BY Continent
```

	Continente	Nº de países
1	Africa	58
2	Antarctica	5
3	Asia	50
4	Europe	46
5	North America	37
6	Oceania	28
7	South America	14

## Ejercicio 3

Averigua el nombre ciudades que pertenezcan a un país cuyo jefe de estado (**HeadofState**) tenga por nombre '**Bernard Dowiyogo**'. El resultado simplemente debe constar de la columna **Ciudad**.



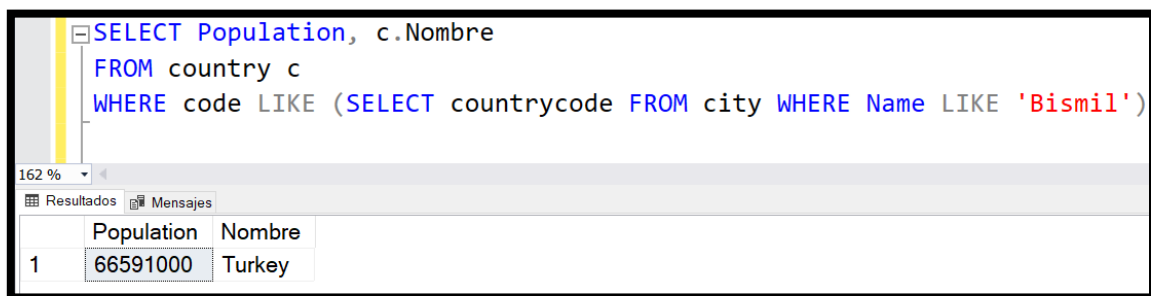
The screenshot shows a SQL query in a text editor and its results in a table. The query is: `SELECT city.Name as 'Ciudad' FROM city WHERE city.CountryCode LIKE (SELECT code FROM country WHERE HeadOfState LIKE 'Bernard Dowiyogo')`. The results table has one column, 'Ciudad', and two rows: 'Yangor' and 'Yaren'.

```
SELECT city.Name as 'Ciudad'
FROM city
WHERE city.CountryCode LIKE (SELECT code FROM country WHERE HeadOfState LIKE 'Bernard Dowiyogo')
```

	Ciudad
1	Yangor
2	Yaren

#### Ejercicio 4

Averigua la población (**Population**) del País al que pertenece la ciudad 'Bismil'. El resultado deben ser dos columnas, **Población y País**.



The screenshot shows a SQL query in a text editor and its results in a table. The query is: `SELECT Population, c.Nombre FROM country c WHERE code LIKE (SELECT countrycode FROM city WHERE Name LIKE 'Bismil')`. The results table has two columns, 'Population' and 'Nombre', and one row: '66591000' and 'Turkey'.

```
SELECT Population, c.Nombre
FROM country c
WHERE code LIKE (SELECT countrycode FROM city WHERE Name LIKE 'Bismil')
```

	Population	Nombre
1	66591000	Turkey

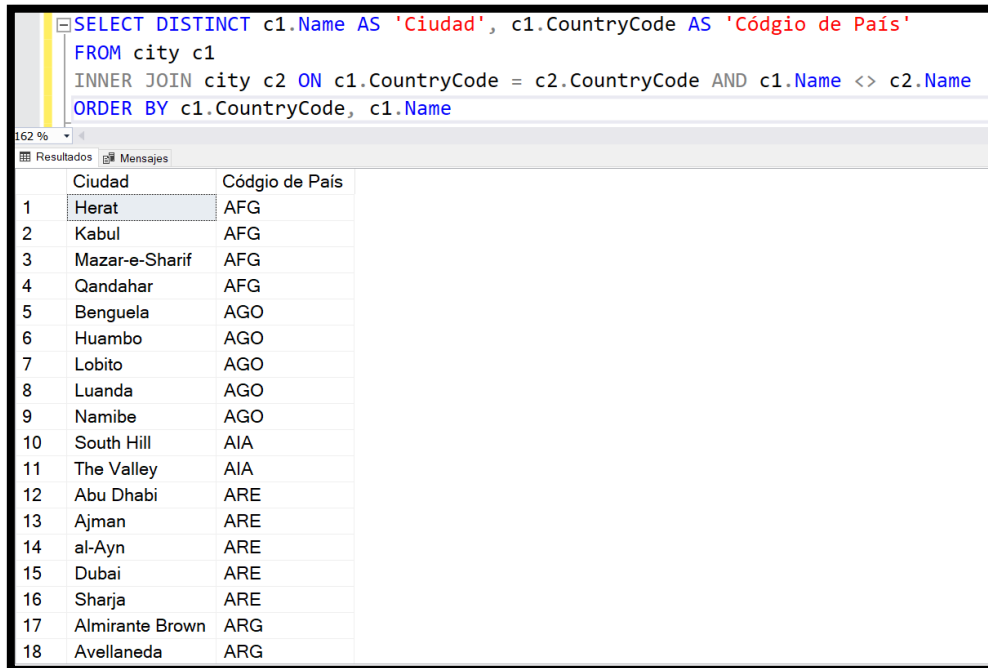
#### Ejercicio 5

Teniendo en cuenta los datos proporcionados en nuestra base de datos **World**, averigua el nombre de todos los países cuya esperanza de vida, (**LifeExpectancy**), sea superior a la media mundial. En el resultado se debe mostrar los nombres de los países, en una columna **País**, junto a su esperanza de vida, **Esperanza de Vida**. Ordena el resultado según la columna **Esperanza de Vida** en orden descendente.

<pre>SELECT c.Nombre as 'Pais', LifeExpectancy as 'Esperanza de vida' FROM country c WHERE c.LifeExpectancy &gt; (SELECT AVG(LifeExpectancy) FROM country) ORDER BY LifeExpectancy DESC</pre>		
162 %		
Resultados Mensajes		
	Pais	Esperanza de vida
1	Andorra	83,5
2	Macao	81,6
3	San Marino	81,1
4	Japan	80,7
5	Singapore	80,1
6	Australia	79,8
7	Switzerland	79,6
8	Sweden	79,6
9	Hong Kong	79,5
10	Iceland	79,4
11	Canada	79,4
12	Italy	79
13	Gibraltar	79
14	Cayman Islands	78,9
15	France	78,8
16	Spain	78,8
17	Liechtenstein	78,8
18	Monaco	78,8

## Ejercicio 6

Obtén un listado en el que aparezcan las ciudades que pertenecen al mismo país en dos columnas, **Ciudad** y **País**, sin repeticiones y ordenadas por código de país y nombre de ciudad.



The screenshot shows a SQL query editor with the following query:

```
SELECT DISTINCT c1.Name AS 'Ciudad', c1.CountryCode AS 'Código de País'
FROM city c1
INNER JOIN city c2 ON c1.CountryCode = c2.CountryCode AND c1.Name <> c2.Name
ORDER BY c1.CountryCode, c1.Name
```

Below the query, the results are displayed in a table with two columns: 'Ciudad' and 'Código de País'. The results are ordered by country code and then by city name.

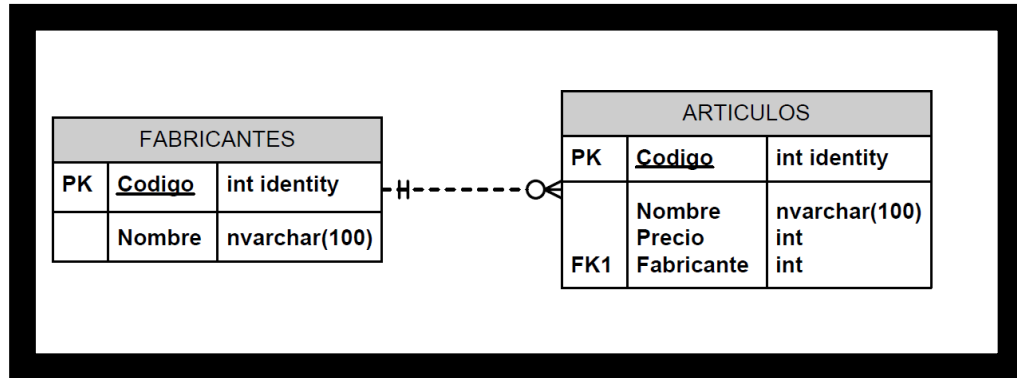
	Ciudad	Código de País
1	Herat	AFG
2	Kabul	AFG
3	Mazar-e-Sharif	AFG
4	Qandahar	AFG
5	Benguela	AGO
6	Huambo	AGO
7	Lobito	AGO
8	Luanda	AGO
9	Namibe	AGO
10	South Hill	AIA
11	The Valley	AIA
12	Abu Dhabi	ARE
13	Ajman	ARE
14	al-Ayn	ARE
15	Dubai	ARE
16	Sharja	ARE
17	Almirante Brown	ARG
18	Avellaneda	ARG

## EJERCICIOS BASE DE DATOS BDMONTECASTELO

**NOTA:** Para hacer estos ejercicios debemos importar el script **BDMontecastelo.sql**

### 1. La Tienda de Informática

Haz las siguientes consultas:



1.1. Obtener los nombres de los artículos de la tienda.

```
SELECT Nombre FROM ARTICULOS
```

1.2. Obtener los nombres y los precios de los artículos de la tienda.

```
SELECT Nombre, Precio FROM ARTICULOS
```

1.3. Obtener el nombre de los artículos cuyo precio sea menor o igual a 200 €.

```
SELECT Nombre FROM ARTICULOS WHERE Precio > 200
```

1.4. Obtener todos los datos de los artículos cuyo precio esté entre los 60 € y los 120 € (ambas cantidades incluidas).

```
/* Con AND */
SELECT * FROM ARTICULOS
  WHERE Precio >= 60 AND Precio <= 120

/* Con BETWEEN */
SELECT * FROM ARTICULOS
  WHERE Precio BETWEEN 60 AND 120
```

1.5. Obtener el nombre y el precio en pesetas (es decir, el precio en euros multiplicado por 166'386)

```
/* Sin AS */  
SELECT Nombre, Precio * 166.386 FROM ARTICULOS  
  
/* Con AS */  
SELECT Nombre, Precio * 166.386 AS PrecioPtas FROM ARTICULOS
```

1.6. Seleccionar el precio medio de todos los artículos.

```
SELECT AVG(Precio) FROM ARTICULOS
```

1.7. Obtener el precio medio de los artículos cuyo código de fabricante sea 2.

```
SELECT AVG(Precio) FROM ARTICULOS WHERE Fabricante=2
```

1.8. Obtener el número de artículos cuyo precio sea mayor o igual a 180 ¤.

```
SELECT COUNT(*) FROM ARTICULOS WHERE Precio >= 180
```

1.9. Obtener el nombre y precio de los artículos cuyo precio sea mayor o igual a 180 € y ordenarlos descendientemente por precio, y luego ascendientemente por nombre.

```
SELECT Nombre, Precio FROM ARTICULOS  
WHERE Precio >= 180  
ORDER BY Precio DESC, Nombre
```

1.10. Obtener un listado completo de artículos, incluyendo por cada artículo los datos del artículo y de su fabricante.

```
SELECT *  
FROM ARTICULOS INNER JOIN FABRICANTES  
ON ARTICULOS.Fabricante = FABRICANTES.Codigo
```

1.11. Obtener un listado de artículos, incluyendo el nombre del artículo, su precio, y el nombre de su fabricante.

```
/* Con INNER JOIN */  
SELECT ARTICULOS.Nombre, Precio, FABRICANTES.Nombre  
FROM ARTICULOS INNER JOIN FABRICANTES  
ON ARTICULOS.Fabricante = FABRICANTES.Codigo
```

1.12. Obtener el precio medio de los artículos de cada fabricante, mostrando solo los códigos de fabricante.

```
SELECT AVG(Precio), Fabricante FROM ARTICULOS  
GROUP BY Fabricante
```

1.13. Obtener el precio medio de los artículos de cada fabricante, mostrando el nombre del fabricante.

```
SELECT AVG(Precio), FABRICANTES.Nombre  
FROM ARTICULOS INNER JOIN FABRICANTES  
ON ARTICULOS.Fabricante = FABRICANTES.Codigo  
GROUP BY FABRICANTES.Nombre
```

1.14. Obtener los nombres de los fabricantes que ofrezcan productos cuyo precio medio sea mayor o igual a 150

```
* Con INNER JOIN */  
SELECT AVG(Precio), FABRICANTES.Nombre  
FROM ARTICULOS INNER JOIN FABRICANTES  
ON ARTICULOS.Fabricante = FABRICANTES.Codigo  
GROUP BY FABRICANTES.Nombre  
HAVING AVG(Precio) >= 150
```

1.15. Obtener el nombre y precio del artículo más barato.

```
SELECT Nombre, Precio  
FROM ARTICULOS  
WHERE Precio = (SELECT MIN(Precio) FROM ARTICULOS)
```



1.16. Obtener una lista con el nombre y precio de los artículos más caros de cada proveedor (incluyendo el nombre del proveedor).

```
SELECT A.Nombre, A.Precio, F.Nombre
FROM ARTICULOS A INNER JOIN FABRICANTES F
ON A.Fabricante = F.Codigo
AND A.Precio =
(
    SELECT MAX(A.Precio)
    FROM ARTICULOS A
    WHERE A.Fabricante = F.Codigo
)
```

1.17. Añadir un nuevo artículo: Altavoces de 70 € (del fabricante 2)

```
INSERT INTO ARTICULOS( Nombre , Precio , Fabricante)
VALUES ( 'Altavoces' , 70 , 2 )
```

1.18. Cambiar el nombre del artículo 8 a 'Impresora Laser'

```
UPDATE ARTICULOS
SET Nombre = 'Impresora Laser'
WHERE Codigo = 8
```

1.19. Aplicar un descuento del 10% (multiplicar el precio por 0'9) a todos los productos.

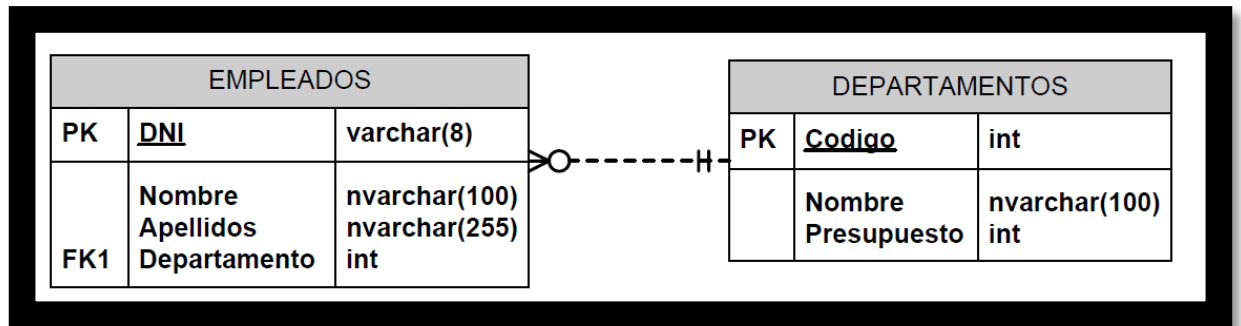
```
UPDATE ARTICULOS
SET Precio = Precio * 0.9
```

1.20. Aplicar un descuento de 10 € a todos los productos cuyo precio sea mayor o igual a 120 €

```
UPDATE ARTICULOS
SET Precio = Precio - 10
WHERE Precio >= 120
```

## 2. Empleados

Haz las siguientes consultas:



2.1. Obtener los apellidos de los empleados.

```
SELECT Apellidos FROM EMPLEADOS
```

2.2. Obtener los apellidos de los empleados sin repeticiones.

```
SELECT DISTINCT Apellidos FROM EMPLEADOS
```

2.3. Obtener todos los datos de los empleados que se apellidan 'López'.

```
SELECT * FROM EMPLEADOS WHERE Apellidos = 'López'
```

2.4. Obtener todos los datos de los empleados que se apellidan 'López' y los que se apellidan 'Pérez'.

```

/* Con OR */
SELECT * FROM EMPLEADOS
  WHERE Apellidos = 'López' OR Apellidos = 'Pérez'

/* Con IN */
SELECT * FROM EMPLEADOS
  WHERE Apellidos IN ('López', 'Pérez')

```

2.5. Obtener todos los datos de los empleados que trabajan para el departamento 14.

```
SELECT * FROM EMPLEADOS WHERE Departamento = 14
```

2.6. Obtener todos los datos de los empleados que trabajan para el departamento 37 y para el departamento 77.

```

/* Con OR */
SELECT * FROM EMPLEADOS
  WHERE Departamento = 37 OR Departamento = 77

/* Con IN */
SELECT * FROM EMPLEADOS
  WHERE Departamento IN (37,77)

```

2.7. Obtener todos los datos de los empleados cuyo apellido comience por 'P'.

```

SELECT * FROM EMPLEADOS
  WHERE Apellidos LIKE 'P%'

```

2.8. Obtener el presupuesto total de todos los departamentos.

```
SELECT SUM(Presupuesto) FROM DEPARTAMENTOS
```

2.9. Obtener el número de empleados en cada departamento.

```

SELECT Departamento, COUNT(*)
  FROM EMPLEADOS
 GROUP BY Departamento

```

2.10. Obtener un listado completo de empleados, incluyendo por cada empleado los datos del empleado y de su departamento.

```

SELECT *
  FROM EMPLEADOS INNER JOIN DEPARTAMENTOS
    ON EMPLEADOS.Departamento = DEPARTAMENTOS.Codigo

```

- 2.11. Obtener un listado completo de empleados, incluyendo el nombre y apellidos del empleado junto al nombre y presupuesto de su departamento.

```
/* Sin etiquetas */
SELECT EMPLEADOS.Nombre, Apellidos, DEPARTAMENTOS.Nombre, Presupuesto
FROM EMPLEADOS
INNER JOIN DEPARTAMENTOS
ON EMPLEADOS.Departamento = DEPARTAMENTOS.Codigo

/* Con etiquetas */
SELECT E.Nombre, Apellidos, D.Nombre, Presupuesto
FROM EMPLEADOS E
INNER JOIN DEPARTAMENTOS D
ON E.Departamento = D.Codigo
```

- 2.12. Obtener los nombres y apellidos de los empleados que trabajen en departamentos cuyo presupuesto sea mayor de 60.000 €.

```
/* Sin subconsulta */
SELECT EMPLEADOS.Nombre, Apellidos
FROM EMPLEADOS
INNER JOIN DEPARTAMENTOS
ON EMPLEADOS.Departamento = DEPARTAMENTOS.Codigo AND
DEPARTAMENTOS.Presupuesto > 60000

/* Con subconsulta */
SELECT Nombre, Apellidos FROM EMPLEADOS
WHERE Departamento IN
(SELECT Codigo FROM DEPARTAMENTOS WHERE Presupuesto > 60000)
```

- 2.13. Obtener los datos de los departamentos cuyo presupuesto es superior al presupuesto medio de todos los departamentos.

```
SELECT *
FROM DEPARTAMENTOS
WHERE Presupuesto >
(
    SELECT AVG(Presupuesto)
    FROM DEPARTAMENTOS
)
```

- 2.14. Obtener los nombres (únicamente los nombres) de los departamentos que tienen más de dos empleados.

```

/* Con subconsulta */
SELECT Nombre FROM DEPARTAMENTOS
WHERE Codigo IN
(
    SELECT Departamento
    FROM EMPLEADOS
    GROUP BY Departamento
    HAVING COUNT(*) > 2
)

/* Con UNION. No funciona si dos departamentos tienen el
mismo nombre */
SELECT DEPARTAMENTOS.Nombre
FROM EMPLEADOS INNER JOIN DEPARTAMENTOS
ON Departamento = Codigo
GROUP BY DEPARTAMENTOS.Nombre
HAVING COUNT(*) > 2

```

- 2.15. Añadir un nuevo departamento: 'Calidad', con presupuesto de 40.000 € y código 11. Añadir un empleado vinculado al departamento recién creado: Esther Vázquez, DNI: 89267109

```

INSERT INTO DEPARTAMENTOS
VALUES ( 11 , 'Calidad' , 40000)

INSERT INTO EMPLEADOS
VALUES ( '89267109' , 'Esther' , 'Vázquez' , 11)

```

- 2.16. Aplicar un recorte presupuestario del 10% a todos los departamentos.

```
UPDATE DEPARTAMENTOS SET Presupuesto = Presupuesto * 0.9
```

- 2.17. Reasignar a los empleados del departamento de investigación (código 77) al departamento de informática (código 14).

```
UPDATE EMPLEADOS SET Departamento = 14 WHERE Departamento = 77
```

- 2.18. Despedir a todos los empleados que trabajan para el departamento de informática (código 14).

```
DELETE FROM EMPLEADOS
WHERE Departamento = 14
```

- 2.19. Despedir a todos los empleados que trabajen para departamentos cuyo presupuesto sea superior a los 60.000 ¤.

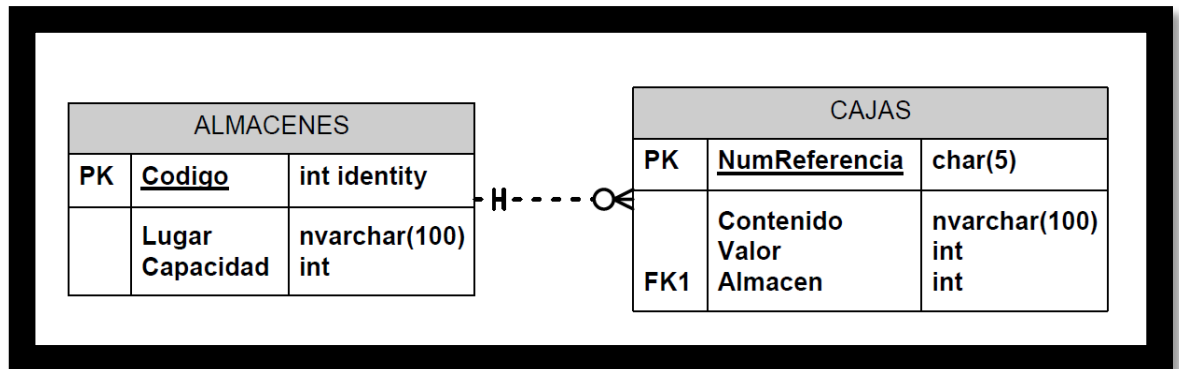
```
DELETE FROM EMPLEADOS
WHERE Departamento IN
(
    SELECT Codigo FROM DEPARTAMENTO
    WHERE Presupuesto >= 60000
)
```

2.20. Despedir a todos los empleados.

```
DELETE FROM EMPLEADOS
```

### 3. Almacenes

Haz las siguientes consultas:



3.1. Obtener todos los almacenes

```
SELECT * FROM ALMACENES
```

3.2. Obtener todas las cajas cuyo contenido tenga un valor superior a 150 ¤.

```
SELECT * FROM CAJAS WHERE Valor > 150
```

3.3. Obtener los tipos de contenidos de las cajas.

```
SELECT DISTINCT Contenido FROM CAJAS
```

3.4. Obtener el valor medio de todas las cajas.

```
SELECT AVG(Valor) FROM CAJAS
```

3.5. Obtener el valor medio de las cajas de cada almacén.

```
SELECT Almacen, AVG(Valor)
FROM CAJAS
GROUP BY Almacen
```

13

3.6. Obtener los códigos de los almacenes en los cuales el valor medio de las cajas sea superior a 150 ¤.

```
SELECT Almacen, AVG(Valor)
FROM CAJAS
GROUP BY Almacen
HAVING AVG(Valor) > 150
```

- 3.7. Obtener el número de referencia de cada caja junto con el nombre de la ciudad en el que se encuentra.

```
SELECT NumReferencia, Lugar
FROM ALMACENES INNER JOIN CAJAS
ON ALMACENES.Codigo = CAJAS.Almacen
```

- 3.8. Obtener el número de cajas que hay en cada almacén.

```
/* Esta consulta no tiene en cuenta los almacenes vacíos */
SELECT Almacen, COUNT(*)
FROM CAJAS
GROUP BY Almacen

/* Esta consulta tiene en cuenta los almacenes vacíos */
SELECT Codigo, COUNT(NumReferencia)
FROM ALMACENES
LEFT JOIN CAJAS
ON ALMACENES.Codigo = CAJAS.Almacen
GROUP BY Codigo
```

- 3.9. Obtener los códigos de los almacenes que están saturados (los almacenes donde el numero de cajas es superior a la capacidad).

```
SELECT Codigo
FROM ALMACENES
WHERE Capacidad < (
    SELECT COUNT(*)
    FROM CAJAS
    WHERE Almacen = Codigo
)
```

- 3.10. Obtener los números de referencia de las cajas que están en Bilbao.



```

/* Sin subconsultas */
SELECT NumReferencia
  FROM ALMACENES LEFT JOIN CAJAS
    ON ALMACENES.Codigo = CAJAS.Almacen
 WHERE Lugar = 'Bilbao'

/* Con subconsultas */
SELECT NumReferencia
  FROM CAJAS
 WHERE Almacen IN (
    SELECT Codigo
      FROM ALMACENES
     WHERE Lugar = 'Bilbao'
  )

```

- 3.11. Insertar un nuevo almacén en Barcelona con capacidad para 3 cajas.

```
INSERT INTO ALMACENES(Lugar,Capacidad) VALUES('Barcelona',3)
```

- 3.12. Insertar una nueva caja, con número de referencia 'H5RT', con contenido 'Papel', valor 200, y situada en el almacén 2.

```
INSERT INTO CAJAS
VALUES('H5RT','Papel',200,2)
```

- 3.13. Rebajar el valor de todas las cajas un 15%.

```
UPDATE CAJAS SET Valor = Valor * 0.85
```

- 3.14. Rebajar un 20% el valor de todas las cajas cuyo valor sea superior al valor medio de todas las cajas.

```
UPDATE CAJAS SET Valor = Valor * 0.80
WHERE Valor > (SELECT AVG(Valor) FROM CAJAS)
```

- 3.15. Eliminar todas las cajas cuyo valor sea inferior a 100 ¤.

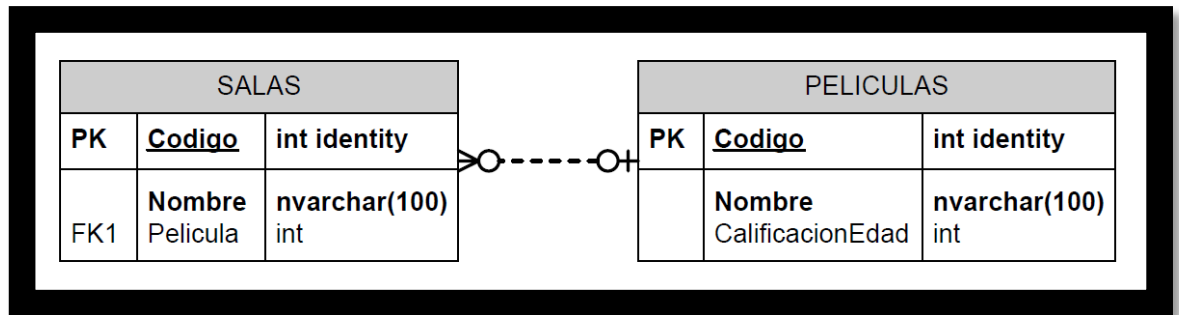
```
DELETE FROM CAJAS WHERE Valor < 100
```

- 3.16. Vaciar el contenido de los almacenes que están saturados.

```
DELETE FROM CAJAS WHERE Almacen IN
(
  SELECT Codigo
  FROM ALMACENES
  WHERE Capacidad < (
    SELECT COUNT(*)
    FROM CAJAS
    WHERE Almacen = Codigo
  )
)
```

#### 4. Películas y Salas

Haz las siguientes consultas:



4.1. Mostrar el nombre de todas las películas.

```
SELECT Nombre FROM PELICULAS
```

4.2. Mostrar las distintas calificaciones de edad que existen.

```
SELECT DISTINCT CalificacionEdad FROM PELICULAS
```

4.3. Mostrar todas las películas que no han sido calificadas.

```
SELECT * FROM PELICULAS WHERE CalificacionEdad IS NULL
```

4.4. Mostrar todas las salas que no proyectan ninguna película.

```
SELECT * FROM SALAS WHERE Pelicula IS NULL
```

4.5. Mostrar la información de *todas* las salas y, si se proyecta alguna película en la sala, mostrar también la información de la película.

```
SELECT *  
FROM SALAS LEFT JOIN PELICULAS  
ON SALAS.Pelicula = PELICULAS.Codigo
```

4.6. Mostrar la información de *todas* las películas y, si se proyecta en alguna sala, mostrar también la información de la sala.

```
SELECT *  
FROM SALAS RIGHT JOIN PELICULAS  
ON SALAS.Pelicula = PELICULAS.Codigo
```

4.7. Mostrar los nombres de las películas que no se proyectan en ninguna sala.

```
/* Con JOIN */  
SELECT PELICULAS.Nombre  
FROM SALAS RIGHT JOIN PELICULAS  
ON SALAS.Pelicula = PELICULAS.Codigo  
WHERE SALAS.Pelicula IS NULL  
  
/* Con Subconsulta */  
SELECT Nombre FROM PELICULAS  
WHERE Codigo NOT IN  
(  
    SELECT Pelicula FROM SALAS  
    WHERE Pelicula IS NOT NULL  
)
```

4.8. Añadir una nueva película 'Uno, Dos, Tres', para mayores de 7 años.

```
INSERT INTO PELICULAS(Nombre,CalificacionEdad) VALUES('Uno, Dos, Tres',7)
```

4.9. Hacer constar que todas las películas no calificadas han sido calificadas 'no recomendables para menores de 13 años'.

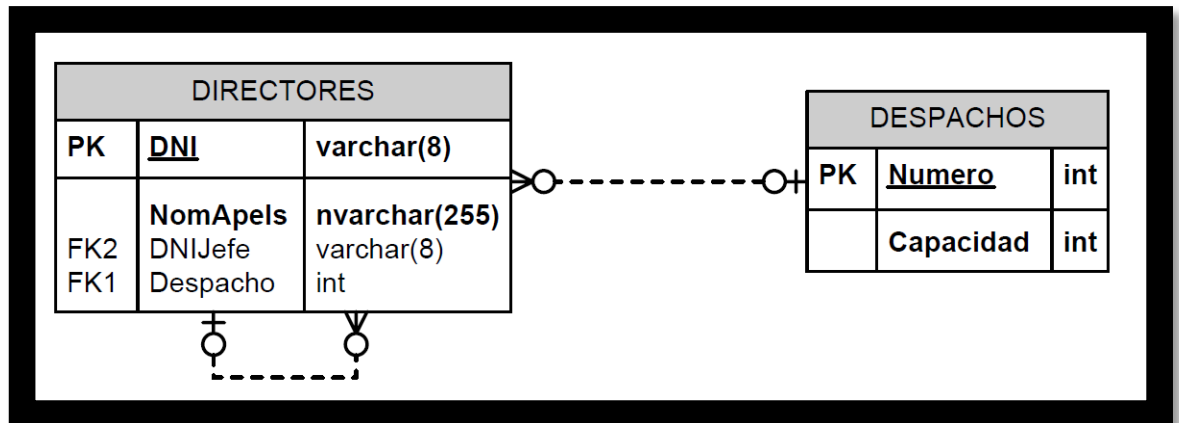
```
UPDATE PELICULAS SET CalificacionEdad=13 WHERE CalificacionEdad IS NULL
```

4.10. Eliminar todas las salas que proyectan películas recomendadas para todos los públicos.

```
DELETE FROM SALAS WHERE Pelicula IN  
(SELECT Codigo FROM PELICULAS WHERE CalificacionEdad = 0)
```

## 5. Directores

Haz las siguientes consultas:



5.1. Mostrar el DNI, nombre y apellidos de todos los directores.

```
SELECT DNI, NomApels FROM DIRECTORES
```

5.2. Mostrar los datos de los directores que no tienen jefes.

```
SELECT * FROM DIRECTORES WHERE DNIJefe IS NULL
```

5.3. Mostrar el nombre y apellidos de cada director, junto con la capacidad del despacho en el que se encuentra.

```
SELECT NomApels, Despacho, Capacidad  
FROM DIRECTORES INNER JOIN DESPACHOS  
ON DIRECTORES.Despacho = DESPACHOS.Numero
```

5.4. Mostrar el número de directores que hay en cada despacho.

```

/* Sin tener en cuenta despachos vacios */
SELECT Despacho, COUNT(*)
  FROM DIRECTORES
  GROUP BY Despacho

/* Teniendo en cuenta despachos vacios */
SELECT Numero, COUNT(DNI)
  FROM DESPACHOS LEFT JOIN DIRECTORES
    ON DESPACHOS.Numero = DIRECTORES.Despacho
  GROUP BY Numero

```

5.5. Mostrar los datos de los directores cuyos jefes no tienen jefes.

```

SELECT * FROM DIRECTORES
  WHERE DNIJefe IN
    (SELECT DNI FROM DIRECTORES WHERE DNIJefe IS NULL)

```

5.6. Mostrar los nombres y apellidos de los directores junto con los de su jefe.

```

/* Con INNER JOIN. No muestra directores que no tienen jefes. */
SELECT d1.NomApels, d2.NomApels
  FROM DIRECTORES d1 INNER JOIN DIRECTORES d2 ON
    d1.DNIJefe = d2.DNI

/* Con LEFT JOIN. Si muestra directores sin jefe. */
SELECT d1.NomApels, d2.NomApels
  FROM DIRECTORES d1 LEFT JOIN DIRECTORES d2
    ON d1.DNIJefe = d2.DNI

```

5.7. Mostrar el número de despachos que están sobre utilizados.

```

SELECT Numero
  FROM DESPACHOS
  WHERE Capacidad < (
    SELECT COUNT(*)
      FROM DIRECTORES
      WHERE Despacho = Numero
  )

```

5.8. Añadir un nuevo director llamado Paco Pérez, DNI 28301700, sin jefe, y situado en el despacho 124.

```

INSERT INTO DIRECTORES VALUES('28301700', 'Paco P´erez', NULL, 124)

```

5.9. Asignar a todos los empleados apellidados Pérez un nuevo jefe con DNI 74568521.

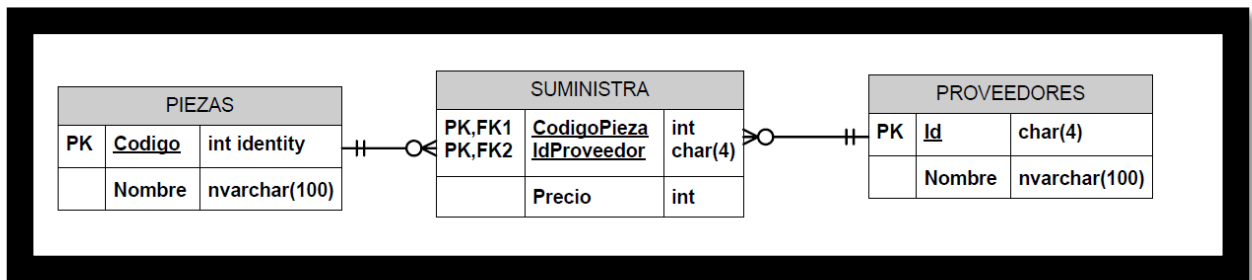
```
UPDATE DIRECTORES SET DNIJefe = '74568521' WHERE NomApels LIKE '%Pérez%'
```

5.10. Despedir a todos los directores, excepto a los que no tienen jefe.

```
DELETE FROM DIRECTORES WHERE DNIJefe IS NOT NULL
```

## 6. Piezas y Proveedores

Haz las siguientes consultas:



6.1. Obtener los nombres de todas las piezas.

```
SELECT Nombre FROM PIEZAS
```

6.2. Obtener todos los datos de todos los proveedores.

```
SELECT * FROM PROVEEDORES
```

6.3. Obtener el precio medio al que se nos suministran las piezas.

```
SELECT CodigoPieza, AVG(Precio)
FROM SUMINISTRA
GROUP BY CodigoPieza
```

6.4. Obtener los nombres de los proveedores que suministran la pieza 1.



```

/* Sin subconsulta */
SELECT PROVEEDORES.Nombre
FROM PROVEEDORES
INNER JOIN SUMINISTRA
ON PROVEEDORES.Id = SUMINISTRA.IdProveedor AND
SUMINISTRA.CodigoPieza = 1

/* Con subconsulta */ SELECT
Nombre
FROM PROVEEDORES
WHERE Id IN
(SELECT IdProveedor FROM SUMINISTRA WHERE CodigoPieza = 1)

```

6.5. Obtener los nombres de las piezas suministradas por el proveedor cuyo código es HAL.

```

/* Sin subconsulta */
SELECT PIEZAS.Nombre
FROM PIEZAS
INNER JOIN SUMINISTRA
ON PIEZAS.Codigo = SUMINISTRA.CodigoPieza AND
SUMINISTRA.IdProveedor = 'HAL'

/* Con subconsulta IN */
SELECT Nombre
FROM PIEZAS
WHERE Codigo IN
(SELECT CodigoPieza FROM SUMINISTRA WHERE IdProveedor = 'HAL')

/* Con subconsulta EXISTS */
SELECT Nombre
FROM PIEZAS
WHERE EXISTS
(
SELECT * FROM SUMINISTRA
WHERE IdProveedor = 'HAL'
AND CodigoPieza = PIEZAS.Codigo
)

```

6.6. Obtener los nombres de los proveedores que suministran las piezas más caras, indicando el nombre de la pieza y el precio al que la suministran.

```

SELECT p1.Nombre, pr1.Nombre, Precio
FROM PIEZAS p1
INNER JOIN
    (SUMINISTRA s1 INNER JOIN PROVEEDORES pr1 ON
    s1.IdProveedor = pr1.Id) ON p1.Codigo =
    s1.CodigoPieza
WHERE Precio IN
(
    SELECT MAX(Precio) FROM SUMINISTRA s2
    GROUP BY s2.CodigoPieza
    HAVING s2.CodigoPieza = p1.Codigo
)

```

6.7. Hacer constar en la base de datos que la empresa "Skellington Supplies"(codigo TNBC) va a empezar a suministrarnos tuercas (código 1) a 7 pesetas cada tuerca.

```

INSERT INTO SUMINISTRA VALUES ( 'TNBC'
, 1 , 7 )

```

6.8. Aumentar los precios en una unidad.

```

UPDATE SUMINISTRA SET Precio = Precio + 1

```

6.9. Hacer constar en la base de datos que la empresa "Susan Calvin Corp."(RBT) no va a suministrarnos ninguna pieza (aunque la empresa en sí va a seguir constando en nuestra base de datos).

```

DELETE FROM SUMINISTRA
WHERE IdProveedor = 'RBT'

```

6.10. Hacer constar en la base de datos que la empresa "Susan Calvin Corp."(RBT) ya no va a suministrarnos clavos (código 4)

```

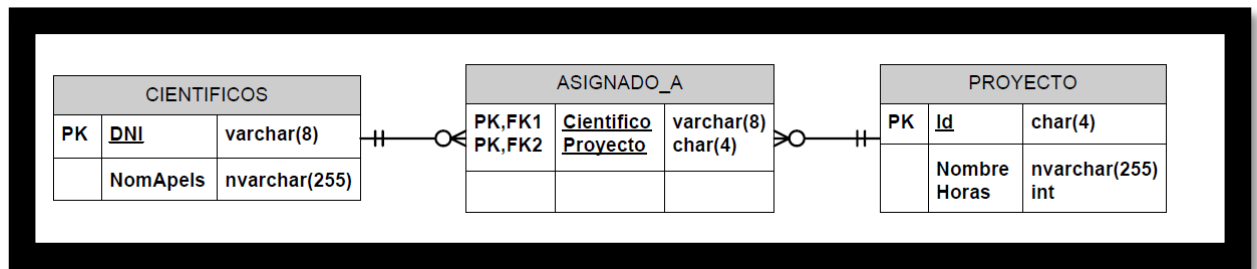
DELETE FROM SUMINISTRA
WHERE IdProveedor = 'RBT'
AND CodigoPieza = 4

```



## 7. Científicos

Haz las siguientes consultas:



7.1. Sacar una relación completa de los científicos asignados a cada proyecto. Mostrar DNI, Nombre del científico, Identificador del proyecto y nombre del proyecto.

```
SELECT DNI, NomApels, Id, Nombre
FROM CIENTIFICOS C
INNER JOIN
(ASIGNADO_A A INNER JOIN PROYECTO P
ON A.Proyecto = P.Id)
ON C.DNI = A.Cientifico
```

7.2. Obtener el número de proyectos al que está asignado cada científico (mostrar el DNI y el nombre).

```
SELECT DNI, NomApels, COUNT(Proyecto)
FROM CIENTIFICOS LEFT JOIN ASIGNADO_A
ON CIENTIFICOS.DNI = ASIGNADO_A.Cientifico
GROUP BY DNI, NomApels
```

27

7.3. Obtener el número de científicos asignados a cada proyecto (mostrar el identificador de proyecto y el nombre del proyecto).

```
SELECT Id, Nombre, COUNT(Proyecto)
FROM PROYECTO LEFT JOIN ASIGNADO_A
ON PROYECTO.Id = ASIGNADO_A.Proyecto
GROUP BY Id, Nombre
```

7.4. Obtener el número de horas de dedicación de cada científico.

```

SELECT DNI, NomApels, SUM(Horas)
FROM CIENTIFICOS C LEFT JOIN
(ASIGNADO_A A INNER JOIN PROYECTO P
ON A.Proyecto = P.Id)
ON C.DNI = A.Cientifico
GROUP BY DNI, NomApels

```

7.5. Obtener el DNI y nombre de los científicos que se dedican a más de un proyecto y cuya dedicación media a cada proyecto sea superior a las 80 horas.

```

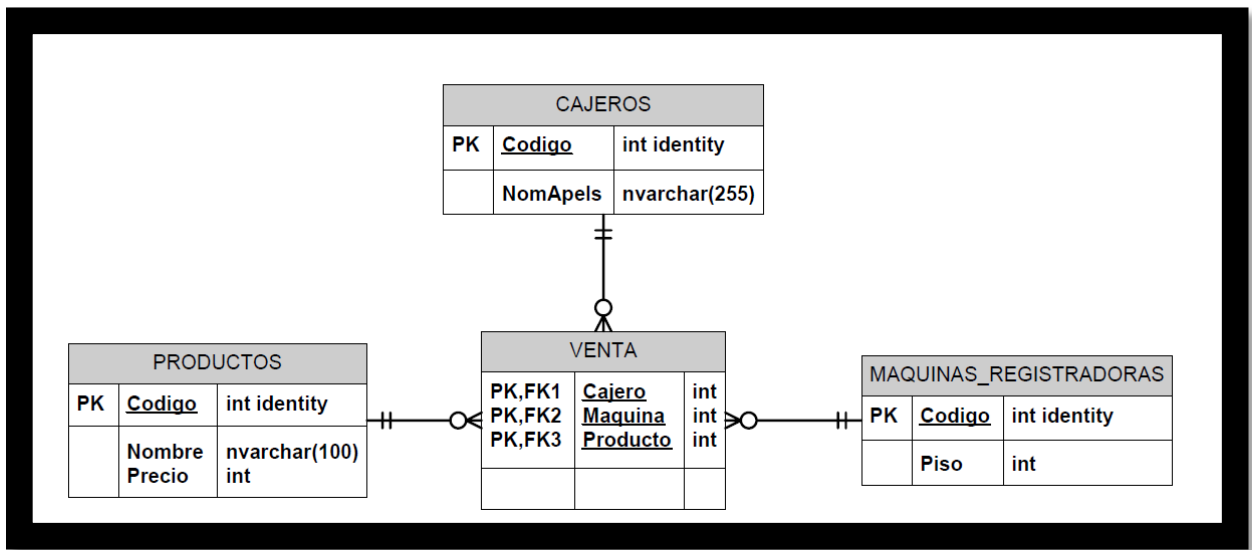
/* Con dos subconsultas */
SELECT DNI, NomApels FROM
CIENTIFICOS C
WHERE 1 <
(
SELECT COUNT(*) FROM ASIGNADO_A
WHERE Cientifico = C.DNI
)
AND
80 <
(
SELECT AVG(Horas)
FROM PROYECTO INNER JOIN ASIGNADO_A
ON PROYECTO.Id = ASIGNADO_A.Proyecto
WHERE Cientifico = C.DNI
)

/* Juntando tablas y con HAVING */
SELECT DNI, NomApels
FROM CIENTIFICOS C
INNER JOIN ASIGNADO_A A ON C.DNI = A.Cientifico
INNER JOIN PROYECTO P ON A.Proyecto = P.Id
GROUP BY DNI, NomApels
HAVING COUNT(Proyecto) > 1 AND AVG(Horas) > 80

```

## 8. Grandes almacenes

Haz las siguientes consultas:



8.1. Mostrar el número de ventas de cada producto, ordenado de más a menos ventas.

```
SELECT Codigo, Nombre, COUNT(VENTA.Producto)
FROM PRODUCTOS
LEFT JOIN VENTA
ON PRODUCTOS.Codigo = VENTA.Producto
GROUP BY Codigo, Nombre
ORDER BY COUNT(VENTA.Producto) DESC
```

8.2. Obtener un informe completo de ventas, indicando el nombre del cajero que realizó la venta, nombre y precios de los productos vendidos, y piso en el que se encuentra la máquina registradora donde se realizó la venta.

```
SELECT NomApels, Nombre, Precio, Piso
FROM CAJEROS C
INNER JOIN
  (PRODUCTOS P INNER JOIN
    (MAQUINAS_REGISTRADORAS M INNER JOIN VENTA V ON
      V.Maquina = M.Codigo)
    ON V.Producto = P.Codigo)
  ON V.Cajero = C.Codigo
```

8.3. Obtener las ventas totales realizadas en cada piso.

```
SELECT Piso, SUM(Precio)
FROM VENTA V
INNER JOIN PRODUCTOS P ON V.Producto = P.Codigo
INNER JOIN MAQUINAS_REGISTRADORAS M ON V.Maquina =
M.Codigo
GROUP BY Piso
```

8.4. Obtener el código y nombre de cada empleado junto con el importe total de sus ventas.

```
SELECT C.Codigo, C.NomApels, SUM(Precio)
FROM PRODUCTOS P
INNER JOIN
  (CAJEROS C LEFT JOIN VENTA V ON
    V.Cajero = C.Codigo)
  ON V.Producto = P.Codigo
GROUP BY C.Codigo, NomApels
```

8.5. Obtener el código y nombre de aquellos cajeros que hayan realizado ventas en pisos cuyas ventas totales sean inferiores a los 500 €.

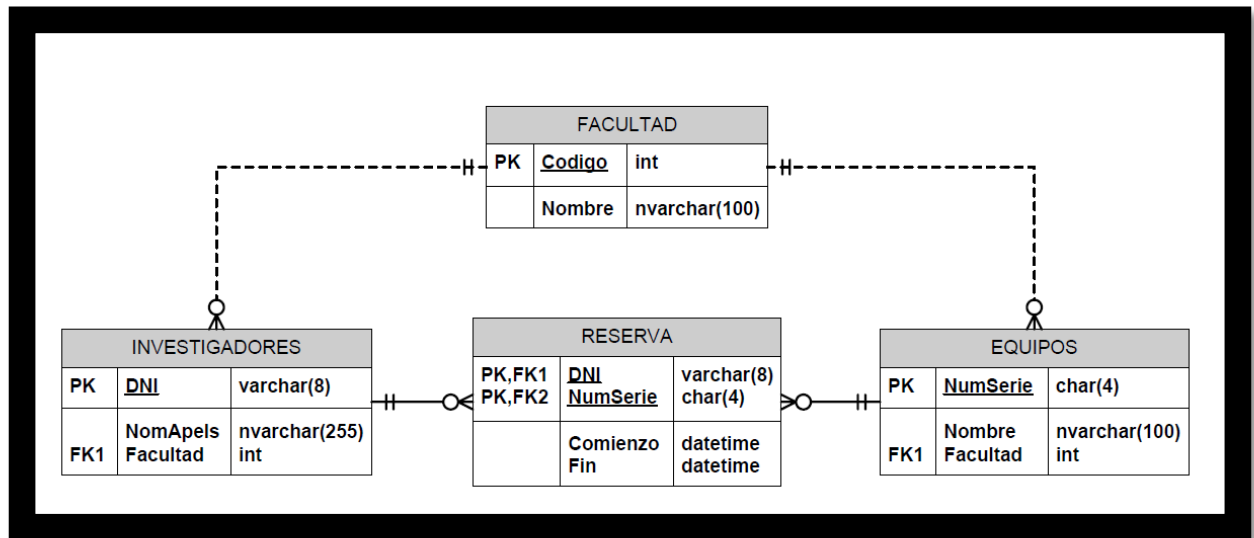
```
SELECT Codigo, NomApels
FROM CAJEROS
WHERE Codigo IN
(
    SELECT Cajero FROM VENTA
    WHERE Maquina IN
    (
        SELECT Codigo FROM MAQUINAS_REGISTRADORAS
        WHERE Piso IN (
            SELECT Piso
            FROM VENTA V, PRODUCTOS P, MAQUINAS_REGISTRADORAS M
            WHERE V.Producto = P.Codigo
            AND V.Maquina = M.Codigo GROUP
            BY Piso
            HAVING SUM(Precio) < 500
        )
    )
)
```





## 9. Investigadores

Haz las siguientes consultas:



9.1. Obtener el DNI y nombre de aquellos investigadores que han realizado m´as de una reserva.

```
/* Juntando tablas */
SELECT I.DNI, NomApels
  FROM INVESTIGADORES I
 LEFT JOIN RESERVA R
    ON R.DNI = I.DNI
 GROUP BY I.DNI, NomApels
 HAVING COUNT(R.DNI) > 1

/* Con subconsulta */
SELECT DNI, NomApels
  FROM INVESTIGADORES
 WHERE DNI IN
   (
     SELECT DNI FROM RESERVA
    GROUP BY DNI
   HAVING COUNT(*) > 1
   )
```



9.2. Obtener un listado completa de reservas, incluyendo los siguientes datos:

- DNI y nombre del investigador, junto con el nombre de su facultad.
- Número de serie y nombre del equipo reservado, junto con el nombre de la facultad a la que pertenece.
- Fecha de comienzo y fin de la reserva.

```
SELECT I.DNI, NomApels, F_INV.Nombre,  
E.NumSerie, E.Nombre, F_EQUIP.Nombre,  
Comienzo, Fin  
FROM RESERVA R  
INNER JOIN INVESTIGADORES I ON R.DNI = I.DNI  
INNER JOIN EQUIPOS E ON R.NumSerie = E.NumSerie  
INNER JOIN FACULTAD F_INV ON I.Facultad = F_INV.Codigo  
INNER JOIN FACULTAD F_EQUIP ON E.Facultad = F_EQUIP.Codigo
```

9.3. Obtener el DNI y el nombre de los investigadores que han reservado equipos que no son de su facultad.

```
/* Juntando tablas */  
SELECT DISTINCT I.DNI, NomApels  
FROM RESERVA R  
INNER JOIN INVESTIGADORES I ON R.DNI = I.DNI  
INNER JOIN EQUIPOS E ON R.NumSerie = E.NumSerie  
AND I.Facultad <> E.Facultad  
  
/* Con EXISTS */  
SELECT DNI, NomApels  
FROM INVESTIGADORES I  
WHERE EXISTS (  
    SELECT *  
    FROM RESERVA R INNER JOIN EQUIPOS E  
    ON R.NumSerie = E.NumSerie  
    WHERE R.DNI = I.DNI  
    AND I.Facultad <> E.Facultad  
)
```

9.4. Obtener los nombres de las facultades en las que ningún investigador ha realizado una reserva.



```
SELECT Nombre FROM FACULTAD
WHERE Codigo IN (
    SELECT Facultad
    FROM INVESTIGADORES I LEFT JOIN RESERVA R
    ON I.DNI = R.DNI
    GROUP BY Facultad
    HAVING COUNT(R.DNI)=0
)
```

9.5. Obtener los nombres de las facultades con investigadores 'ociosos' (investigadores que no han realizado ninguna reserva).

```
SELECT Nombre FROM FACULTAD
WHERE Codigo IN
(
    SELECT Facultad FROM INVESTIGADORES
    WHERE DNI NOT IN
    (
        SELECT DNI FROM RESERVA
    )
)
```

9.6. Obtener el número de serie y nombre de los equipos que nunca han sido reservados.

```
/* Juntando tablas */
SELECT E.NumSerie, Nombre
FROM EQUIPOS E
LEFT JOIN RESERVA R
ON R.NumSerie = E.NumSerie
GROUP BY E.NumSerie,Nombre
HAVING COUNT(R.NumSerie)=0

/* Con subconsulta IN */
SELECT NumSerie, Nombre FROM EQUIPOS
WHERE NumSerie NOT IN
(
    SELECT NumSerie FROM RESERVA )

/* Con EXISTS */
SELECT NumSerie, Nombre
FROM EQUIPOS E
WHERE NOT EXISTS
(
    SELECT * FROM RESERVA R
    WHERE R.NumSerie = E.NumSerie
)
```

