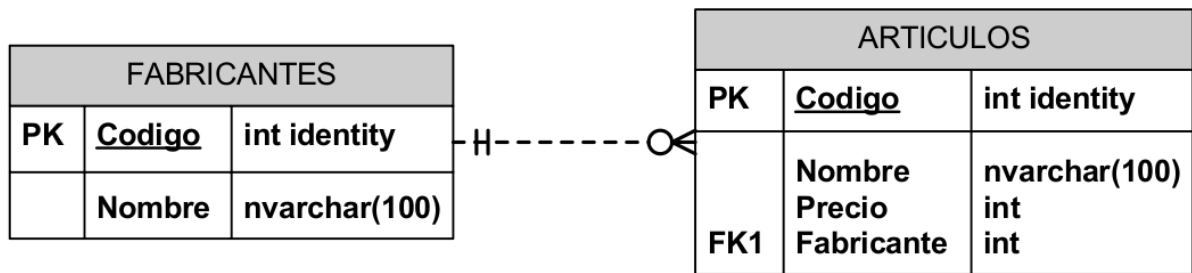


Ejercicios SQL - Manipulación de Datos - Solución

1. La Tienda de Informática



1.1. Obtener los nombres y los precios de los productos de la tienda.

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

1.2. Obtener el nombre de los productos cuyo precio sea mayor o igual a 200 €.

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

1.3. 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
```

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

```
SELECT Nombre, Precio * 166.386 AS PrecioPtas FROM ARTICULOS
```

1.5. Seleccionar el precio medio de todos los productos.

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

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

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

1.7. Obtener el número de artículos cuyo precio sea mayor o igual a 180 €.

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

1.8. 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
```

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

```
SELECT *
FROM ARTICULOS, FABRICANTES
WHERE ARTICULOS.Fabricante = FABRICANTES.Codigo
```

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

```
SELECT ARTICULOS.Nombre, Precio, FABRICANTES.Nombre
FROM ARTICULOS, FABRICANTES
WHERE ARTICULOS.Fabricante = FABRICANTES.Codigo
```

1.11. Obtener el precio medio de los productos de cada fabricante, mostrando solo los códigos de fabricante.

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

1.12. Obtener el precio medio de los productos de cada fabricante, mostrando el nombre del fabricante.

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

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

```
SELECT AVG(Precio), FABRICANTES.Nombre
FROM ARTICULOS, FABRICANTES
WHERE ARTICULOS.Fabricante = FABRICANTES.Codigo
GROUP BY FABRICANTES.Nombre
HAVING AVG(Precio) >= 150
```

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

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

1.15. 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, FABRICANTES F
WHERE A.Fabricante = F.Codigo
AND A.Precio = (
    SELECT MAX(A2.Precio)
    FROM ARTICULOS A2
    WHERE A2.Fabricante = F.Codigo)
```

1.16. Añadir un nuevo producto: Altavoces de 70 € (del fabricante 2).

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

1.17. Cambiar el nombre del producto 8 a 'Impresora Laser'.

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

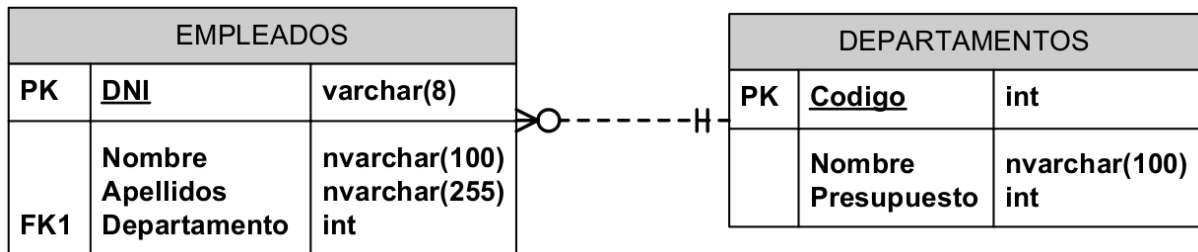
1.18. Aplicar un descuento del 10 % (multiplicar el precio por 0'9) a todos los productos.

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

1.19. 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



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' ó '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, DEPARTAMENTOS
WHERE 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.

```
SELECT E.Nombre, Apellidos, D.Nombre, Presupuesto
FROM EMPLEADOS E, DEPARTAMENTOS D
WHERE 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 €.

```
SELECT EMPLEADOS.Nombre, Apellidos
FROM EMPLEADOS, DEPARTAMENTOS
WHERE EMPLEADOS.Departamento = DEPARTAMENTOS.Codigo
AND DEPARTAMENTOS.Presupuesto > 60000
```

```
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.

```
SELECT Nombre
FROM DEPARTAMENTOS
WHERE Codigo IN (
    SELECT Departamento
    FROM EMPLEADOS
    GROUP BY Departamento
    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 a (código 14).

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

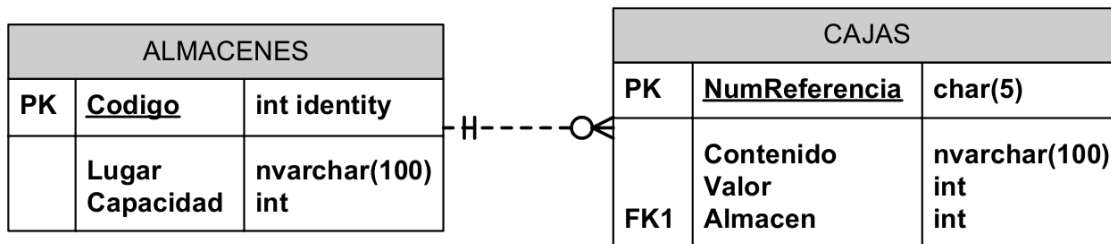
2.19. Despedir a todos los empleados que trabajen para departamentos cuyo presupuesto sea superior a los 60.000 €.

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

2.20. Despedir a todos los empleados.

```
DELETE FROM EMPLEADOS
```

3. Los Almacenes



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
```

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, CAJAS
WHERE 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 estén saturados (los almacenes donde el número 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.

```
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)
```

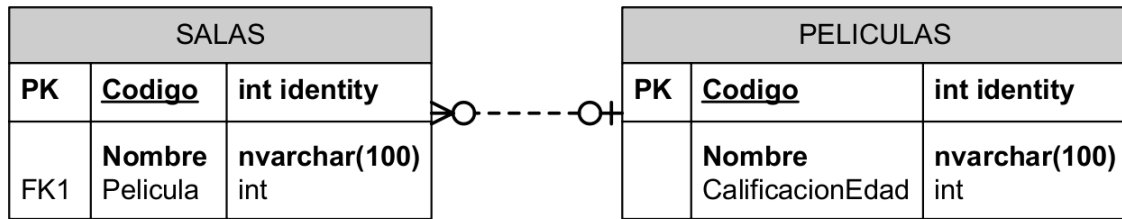
3.15. Eliminar todas las cajas cuyo valor sea inferior a 100 €.

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

3.16. Vaciar el contenido de los almacenes que estén saturados.

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

4. Películas y Salas



4.1. Mostrar las distintas calificaciones de edad que existen.

```
SELECT DISTINCT CalificacionEdad FROM PELICULAS
```

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

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

4.3. 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

LEFT → MUESTRA TODOS LOS PRIMEROS AUNQUE NO ESTÉN EN EL SEGUNDO
```

4.4. 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

RIGHT → MUESTRA TODOS LOS SEGUNDOS AUNQUE NO ESTÉN EN EL PRIMERO
```

4.5. 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 )
```

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

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

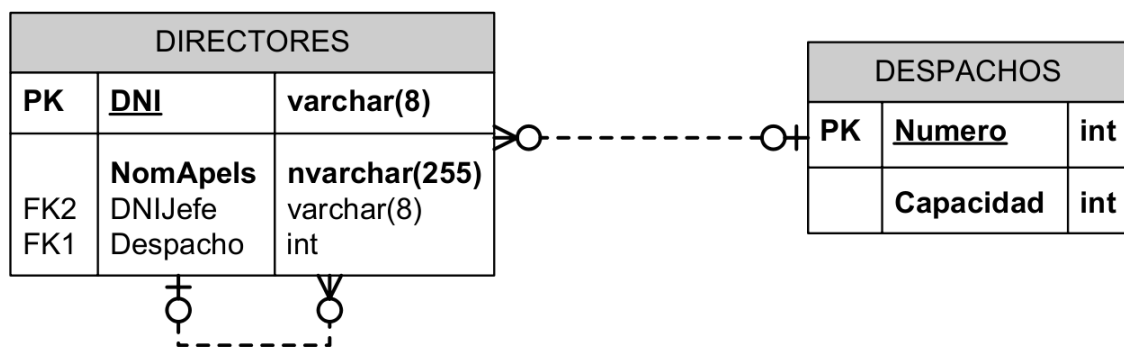
4.7. Hacer constar que todas las películas no calificadas han sido calificadas 'no recomendables para menores de 13 años'.

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

4.8. 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. Los Directores



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, DESPACHOS
WHERE DIRECTORES.Despacho = DESPACHOS.Numero
```

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

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

```
/* Teniendo en cuenta despachos vacíos */
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.

```
/* No muestra los directores que no tienen jefes. */
SELECT d1.NomApels, d2.NomApels
FROM DIRECTORES d1, DIRECTORES d2
WHERE d1.DNIJefe = d2.DNI
```

```
/* 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 estén sobreutilizados.

```
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 Pérez', NULL, 124)
```

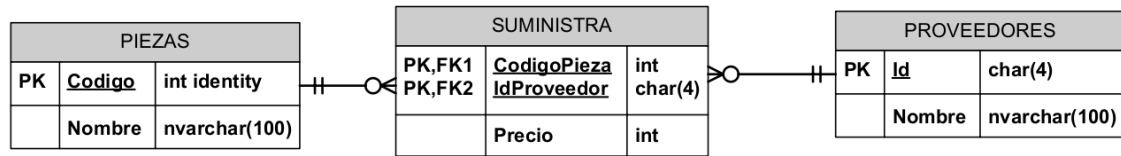
5.9. Asignar a todos los empleados apellidados Pérez un nuevo jefe con DNI 74568521.

```
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



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, SUMINISTRA
WHERE 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, SUMINISTRA
WHERE 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. Hacer constar en la base de datos que la empresa “Skellington Supplies” (código TNBC) va a empezar a suministrarnos tuercas (código 1) a 7 pesetas cada tuerca.

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

6.7. Aumentar los precios en una unidad.

```
UPDATE SUMINISTRA SET Precio = Precio + 1
```

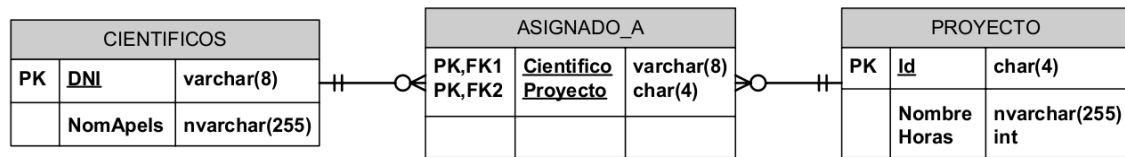
6.8. Hacer constar en la base de datos que la empresa “Susan Calvin Corp.”(código RBT) no va a suministrarnos ninguna pieza más.

```
DELETE FROM SUMINISTRA WHERE IdProveedor = 'RBT'
```

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

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

7. Los científicos



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, ASIGNADO_A A, PROYECTO P
WHERE C.DNI = A.Cientifico
AND A.Proyecto = P.Id
```

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
```

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.

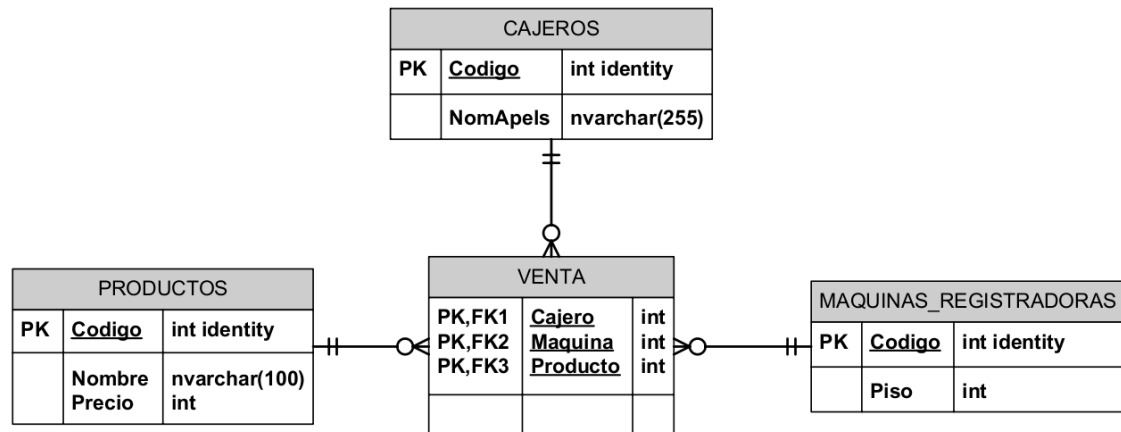
```
SELECT DNI, NomApels, SUM(Horas)
FROM CIENTIFICOS C, ASIGNADO_A A, PROYECTO P
WHERE C.DNI = A.Cientifico
AND A.Proyecto = P.Id
GROUP BY DNI, NomApels
```

7.5. Obtener el DNI y nombre de los científicos que se dedican a más de un proyecto y a 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, ASIGNADO_A
    WHERE PROYECTO.Id = ASIGNADO_A.Proyecto
    AND Cientifico = C.DNI )

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

8. Los Grandes Almacenes



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 VENTA V, CAJEROS C, PRODUCTOS P, MAQUINAS_REGISTRADORAS M
WHERE V.Cajero = C.Codigo
AND V.Producto = P.Codigo
AND V.Maquina = M.Codigo
```

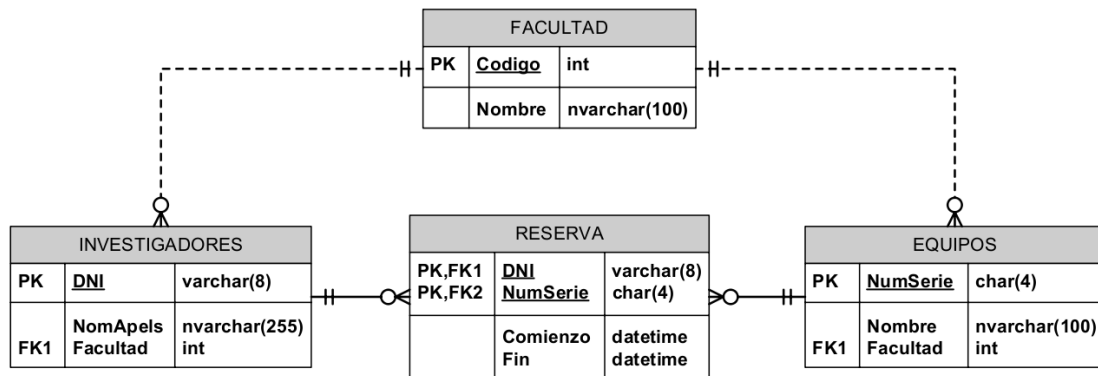
8.3. Obtener las ventas totales realizadas en cada piso.

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

8.4. 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. Los Investigadores



9.1. Obtener el DNI y nombre de aquellos investigadores que han realizado más de una a 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. Numero 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,
    INVESTIGADORES I,
    EQUIPOS E,
    FACULTAD F_INV,
    FACULTAD F_EQUIP
WHERE R.DNI = I.DNI
    AND R.NumSerie = E.NumSerie
    AND I.Facultad = F_INV.Codigo
    AND 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, INVESTIGADORES I, EQUIPOS E
WHERE R.DNI = I.DNI
    AND R.NumSerie = E.NumSerie
    AND I.Facultad <> E.Facultad

/* Con EXISTS */
SELECT DNI, NomApels
FROM INVESTIGADORES I
WHERE EXISTS (
    SELECT *
    FROM RESERVA R, EQUIPOS E
    WHERE R.NumSerie = E.NumSerie
        AND 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 )
```