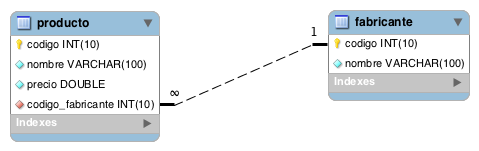
Práctica 2 Consultas SQL

Objetivo

Realizar algunas consultas mediante sentencias SQL

1. Abrimos MySQL WorkBench y creamos una conexión o trabajamos en una ya creada.
2. Creamos una Base de Datos (BD) llamada ***tienda.***
3. Creamos dos tablas ***fabricante*** y ***producto***
4. Agregamos información a dichas tablas.

Modelo Entidad Relación de la BD **tienda**



Sentencias para la creación de la BD y sus tablas con los diferentes registros



Consultas a realizar

1. Cláusula **SELECT**

Nos permite indicar cuáles serán las columnas que tendrá la tabla de resultados de la consulta que estamos realizando. Las opciones que podemos indicar son las siguientes:

* El nombre de una columna de la tabla sobre la que estamos realizando la consulta. Será una columna de la tabla que aparece en la cláusula FROM.
* Una constante que aparecerá en todas las filas de la tabla resultado.
* Una expresión que nos permite calcular nuevos valores.

Ejercicio. Lista el nombre de todos los productos que hay en la tabla producto.

**select nombre from producto;**



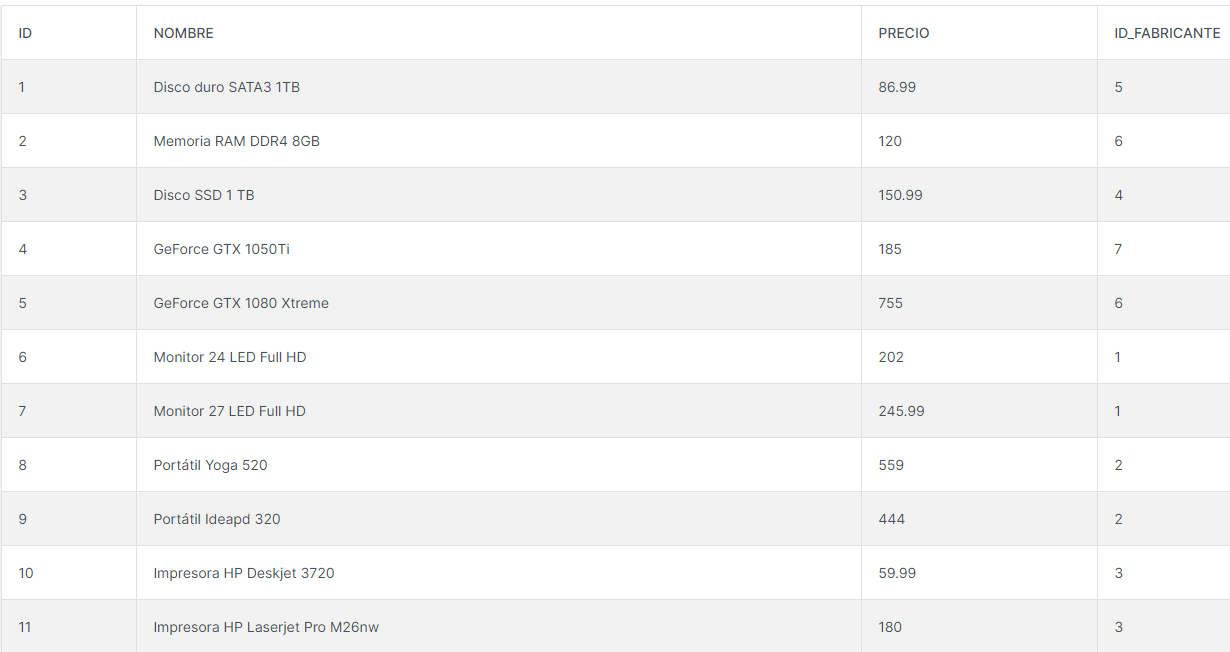
2. Lista los nombres y los precios de todos los productos de la tabla producto.

**select nombre, precio from producto;**



3. Lista todas las columnas de la tabla producto.

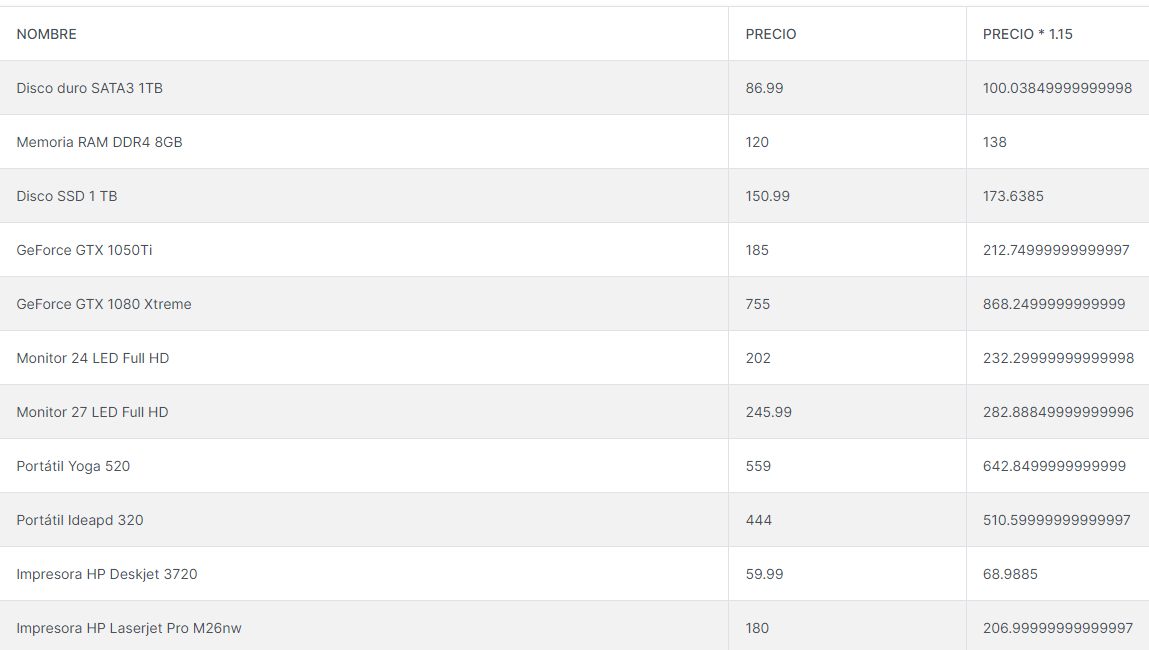
**select \* from producto**



4. Es posible realizar cálculos aritméticos entre columnas para calcular nuevos valores. En el resultado se ve como se multiplica la columna PRECIO por 1.15 para calcular un equivalente. El resultado es una nueva columna llamada PRECIO\*1.15.

Ejercicio. Lista el nombre de los productos, el precio en euros y el precio en dólares estadounidenses (USD).

**select nombre, precio, precio\*1.15 from producto;**



5. Con la palabra reservada AS podemos crear alias para las columnas. Esto puede ser útil cuando estamos calculando nuevas columnas a partir de valores de las columnas actuales. Si el nuevo nombre que estamos creando para el alias contiene espacios en blanco es necesario usar comillas simples.

Ejercicio. Lista el nombre de los productos, el precio en euros y el precio en dólares estadounidenses (USD). Utiliza los siguientes alias para las columnas: nombre de producto, euros, dólares.

**select nombre as 'NOMBRE DE PRODUCTO', precio AS EUROS, precio\*1.15 AS DÓLARES from producto;**



6. A diferencia de la función [LOWER()](https://estradawebgroup.com/Post/Como-convertir-texto-a-minusculas-con-LOWER-en-SQL-Server/20498) que convierte en minúsculas una cadena de texto, la función UPPER() convierte una cadena de texto a mayúsculas. Ejemplo UPPER(input\_string). En esta sintaxis, input\_string puede ser una cadena de caracteres literales, una variable, una expresión de cadena de caracteres o una columna de una tabla.

Ejercicio. Lista los nombres y los precios de todos los productos de la tabla producto, convirtiendo los nombres a mayúscula.

**select UPPER(nombre) AS 'UPPER(NOMBRE)' , precio from producto;**



7. Lista los nombres y los precios de todos los productos de la tabla producto, convirtiendo los nombres a minúscula.

**select LOWER(nombre) AS 'LOWER(NOMBRE)' , precio from producto;**

****

8. La función SUBSTR en SQL se utiliza con la siguiente sintaxis básica:

SUBSTRING(cadena, inicio, longitud)

* cadena: La cadena de texto de la que deseas extraer la subcadena.
* inicio: La posición inicial desde la cual se extraerán los caracteres.
* longitud: La cantidad de caracteres a extraer.

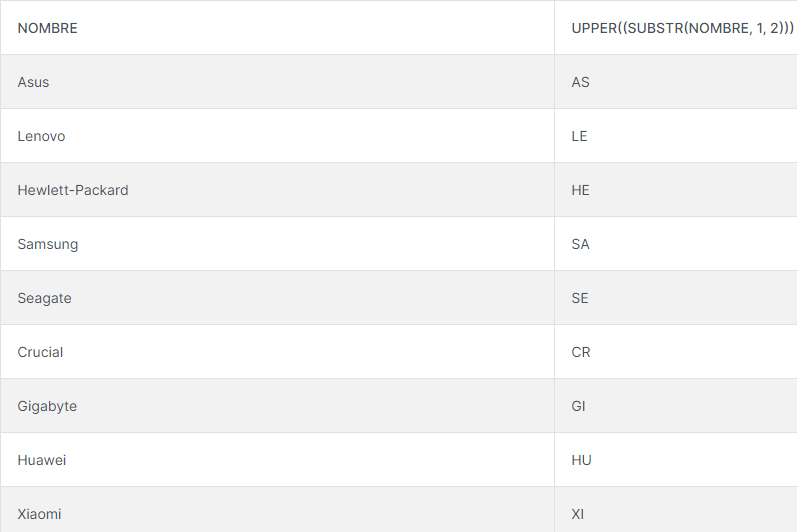
Por ejemplo, si tienes una columna llamada nombre en una tabla y deseas extraer los primeros tres caracteres de cada nombre, puedes usar la función SUBSTR de la siguiente manera:

SELECT SUBSTR(nombre, 1, 3) AS primeros\_tres\_caracteres

FROM tabla;

Ejercicio. Lista el nombre de todos los fabricantes en una columna, y en otra columna obtenga en mayúsculas los dos primeros caracteres del nombre del fabricante.

**select nombre, UPPER((SUBSTR(NOMBRE, 1, 2))) from fabricante;**



9. La función ROUND en SQL se utiliza para redondear un número dado al entero más cercano o a un determinado decimal. La función ROUND(valor,precision) recibe dos parámetros. El primero es el valor numérico a redondear, el segundo se llama precision e indica el número de dígitos decimales que queremos mantener en el número redondeado. Ejemplo, si deseamos redondear el valor 1.25 a un dolo decimal escribimos Round(1.25 , 1) obteniendo 1.3

Ejercicio. Lista los nombres y los precios de todos los productos de la tabla producto, redondeando el valor del precio sin decimales.

**select nombre, ROUND(precio, 0) as 'Precio Redondeado' from producto;**



10. La función Truncate en SQL se utiliza para truncar o eliminar las cifras decimales de un número. Su sintaxis es la siguiente: TRUNCATE (número, cantidad\_de\_decimales). Donde «número» es el valor que se desea truncar y «cantidad\_de\_decimales» es el número de decimales que se desean eliminar. Por ejemplo, si queremos truncar el número 3.145 a dos decimales, la consulta sería la siguiente:

TRUNCATE(3.145, 2), obteniendo 3.14

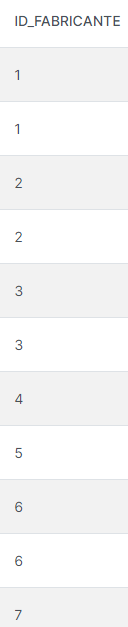
Ejercicio. Lista los nombres y los precios de todos los productos de la tabla producto, truncando el valor del precio para mostrarlo sin ninguna cifra decimal.

**select nombre, truncate(precio,0) as 'Precio truncado' from producto;**



11. Lista el código de los fabricantes que tienen productos en la tabla producto.

**select id\_fabricante from producto as pd join fabricante AS fab on fab.id = pd.id\_fabricante;**



12. Modificadores ALL, DISTINCT y DISTINCTROW

Los modificadores ALL y DISTINCT indican si se deben incluir o no filas repetidas en el resultado de la consulta.

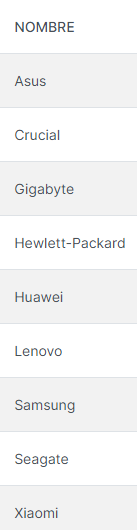
* ALL indica que se deben incluir todas las filas, incluidas las repetidas. Es la opción por defecto, por lo tanto no es necesario indicarla.
  + DISTINCT elimina las filas repetidas en el resultado de la consulta.
  + DISTINCTROW es un sinónimo de DISTINCT.

Ejercicio. Lista el código de los fabricantes que tienen productos en la tabla producto, eliminando los códigos que aparecen repetidos.

**select DISTINCT id\_fabricante from producto as pd join fabricante AS fab on fab.id = pd.id\_fabricante;**

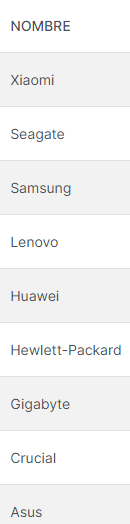
13. Si no indicamos nada en la cláusula ORDER BY se ordenará por defecto de forma ascendente. Ejercicio. Lista los nombres de los fabricantes ordenados de forma ascendente.

**select nombre from fabricante Order by nombre ASC;**

****

14. Ejercicio. Lista los nombres de los fabricantes ordenados de forma descendente.

**select nombre from fabricante Order by nombre DESC;**



15. Ejercicio. Lista los nombres de los productos ordenados en primer lugar por el nombre de forma ascendente y en segundo lugar por el precio de forma descendente.

**select nombre, precio from producto order by nombre ASC ,precio DESC;**

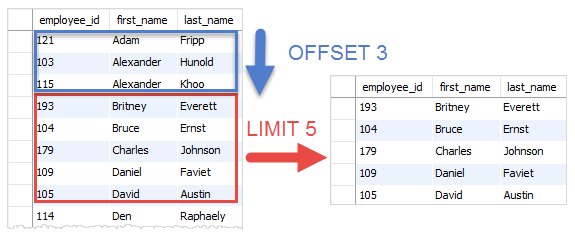


16. Cláusula LIMIT.

LIMIT permite limitar el número de filas que se incluyen en el resultado de la consulta. La sintaxis de MySQL es la siguiente

[LIMIT {[offset,] row\_COUNT | row\_COUNT OFFSET offset}]

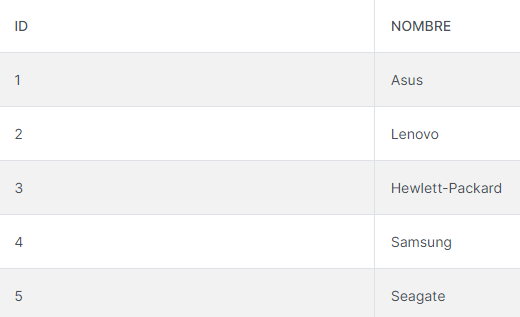
donde row\_COUNT es el número de filas que queremos obtener y offset el número de filas que nos saltamos antes de empezar a contar. Es decir, la primera fila que se obtiene como resultado es la que está situada en la posición offset + 1.



Ejercicio. Devuelve una lista con las 5 primeras filas de la tabla fabricante.

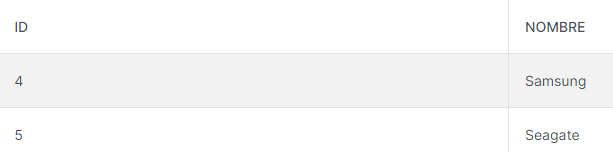
**select \* from fabricante LIMIT 5 OFFSET 0;**

**select \* from fabricante LIMIT 5;**

****

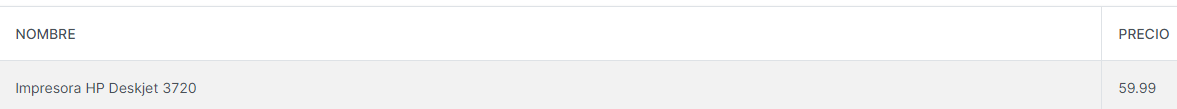
17.Ejercicio. Devuelve una lista con 2 filas a partir de la cuarta fila de la tabla fabricante. La cuarta fila también se debe incluir en la respuesta.

**select \* from fabricante LIMIT 2 OFFSET 3;**



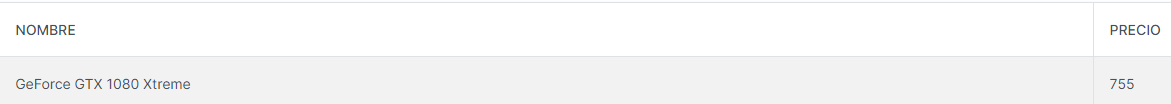
18. Ejercicio. Lista el nombre y el precio del producto más barato. (Utilice solamente las cláusulas ORDER BY y LIMIT).

**select nombre, precio from producto Order BY precio ASC limit 1;**



19. Ejercicio. Lista el nombre y el precio del producto más caro. (Utilice solamente las cláusulas ORDER BY y LIMIT).

**select nombre, precio from producto Order BY precio DESC limit 1;**

****

20. La cláusula WHERE nos permite añadir filtros a nuestras consultas para seleccionar sólo aquellas filas que cumplen una determinada condición. Estas condiciones se denominan predicados y el resultado de estas condiciones puede ser **verdadero**, **falso** o **desconocido**.

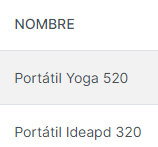
Una condición tendrá un resultado **desconocido** cuando alguno de los valores utilizados tiene el valor NULL.

Podemos diferenciar cinco tipos de condiciones que pueden aparecer en la cláusula WHERE:

* Condiciones para comparar valores o expresiones.
  + Condiciones para comprobar si un valor está dentro de un rango de valores.
  + Condiciones para comprobar si un valor está dentro de un conjunto de valores.
  + Condiciones para comparar cadenas con patrones.
  + Condiciones para comprobar si una columna tiene valores a NULL.

Los operandos usados en las condiciones pueden ser nombres de columnas, constantes o expresiones. Los operadores que podemos usar en las condiciones pueden ser aritméticos, de comparación, lógicos, etc.

**select nombre from producto where id\_fabricante = 2;**



21. Ejercicio. Lista el nombre de los productos que tienen un precio menor o igual a 120€.

**select nombre from producto where precio <= 120;**



22. Ejercicio. Lista el nombre de los productos que tienen un precio mayor o igual a 400€.

**select nombre from producto where precio >= 400;**



23. Ejercicio. Lista el nombre de los productos que **no tienen** un precio mayor o igual a 400€.

24. Ejercicio. Lista todos los productos que tengan un precio entre 80€ y 300€. Sin utilizar el operador BETWEEN.





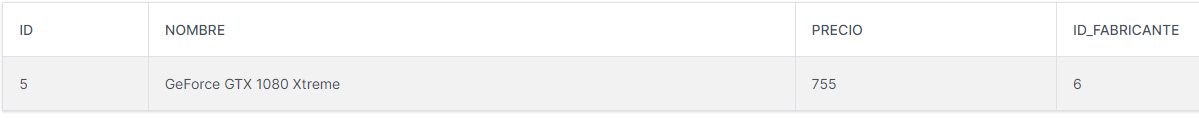
25. Ejercicio. Lista todos los productos que tengan un precio entre 60€ y 250€. Utilizando el operador BETWEEN.





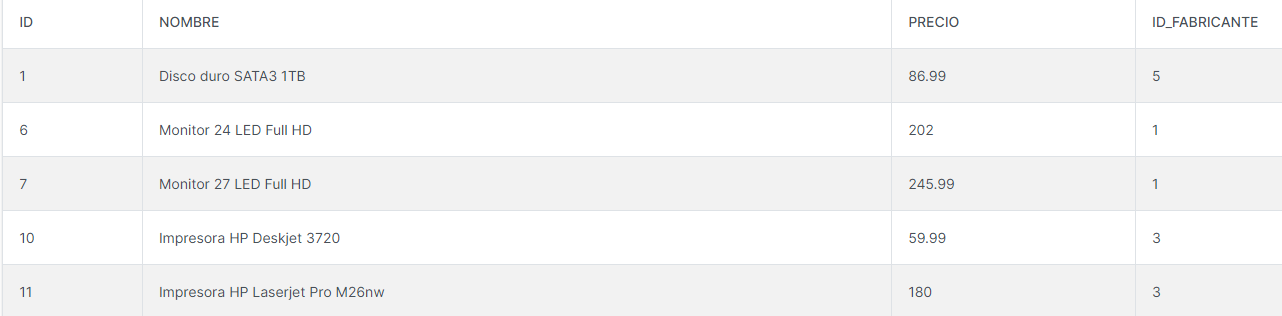
26. Ejercicio. Lista todos los productos que tengan un precio mayor que 200€ y que el identificador de fabricante sea igual a 6.

R/.



27. Ejercicio. Lista todos los productos donde el identificador de fabricante sea 1, 3 o 5.

R/.



28. Operador IN

Este operador nos permite comprobar si el valor de una determinada columna está incluido en una lista de valores.

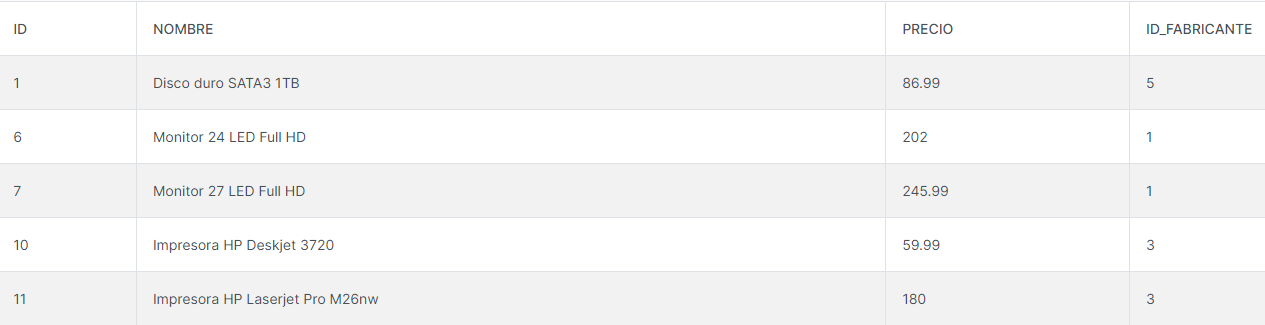
**Ejemplo:**

Obtener todos los datos de los alumnos que tengan como primer apellido Sánchez, Martínez o Domínguez.

SELECT \* FROM alumno WHERE apellido1 IN (`Sánchez`, `Martínez`, `Domínguez`);

Ejercicio.Lista todos los productos donde el identificador de fabricante sea 1, 3 o 5. Utilizando el operador IN.

R/.



29. Ejercicio. Lista el nombre y el precio de los productos en céntimos (Habrá que multiplicar por 100 el valor del precio). Cree un alias para la columna que contiene el precio que se llame céntimos.

R/.



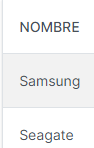
30. Operador LIKE. Se utiliza para comparar si una cadena de caracteres coincide con un patrón. En el patrón podemos utilizar cualquier carácter alfanumérico, pero hay dos caracteres que tienen un significado especial, el símbolo del porcentaje (%) y el carácter de subrayado (\_).

%: Este carácter equivale a cualquier conjunto de caracteres.

\_: Este carácter equivale a cualquier carácter.

Ejercicio. Lista los nombres de los fabricantes cuyo nombre empiece por la letra s.

R/.



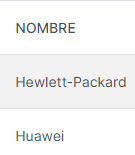
31. Ejercicio. Lista los nombres de los fabricantes cuyo nombre termine por la vocal e.

R/.



32. Ejercicio. Lista los nombres de los fabricantes cuyo nombre contenga el carácter w.

R/.



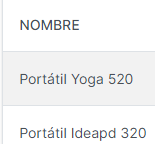
33. Ejercicio. Lista los nombres de los fabricantes cuyo nombre sea de 4 caracteres.

R/.



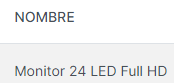
34. Ejercicio. Devuelve una lista con el nombre de todos los productos que contienen la cadena Portátil en el nombre.

R/.



35. Ejercicio. Devuelve una lista con el nombre de todos los productos que contienen la cadena Monitor en el nombre y tienen un precio inferior a 215 €.

R/.



36. Ejercicio. Lista el nombre y el precio de todos los productos que tengan un precio mayor o igual a 180€. Ordene el resultado en primer lugar por el precio (en orden descendente) y en segundo lugar por el nombre (en orden ascendente).

R/.

