

SQL con SQLite

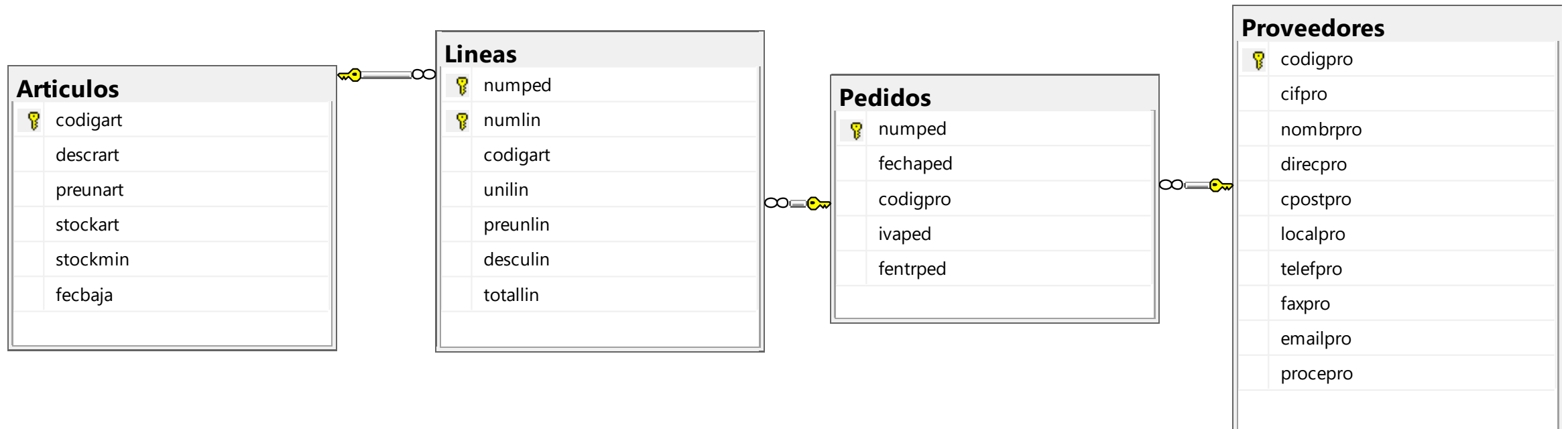
Select, Update, Delete (básico)

Máster en Data Science

M1967 - Modelos de Datos y Sistemas de Información 2018-2019

Consultas, actualizaciones y borrados en SQL

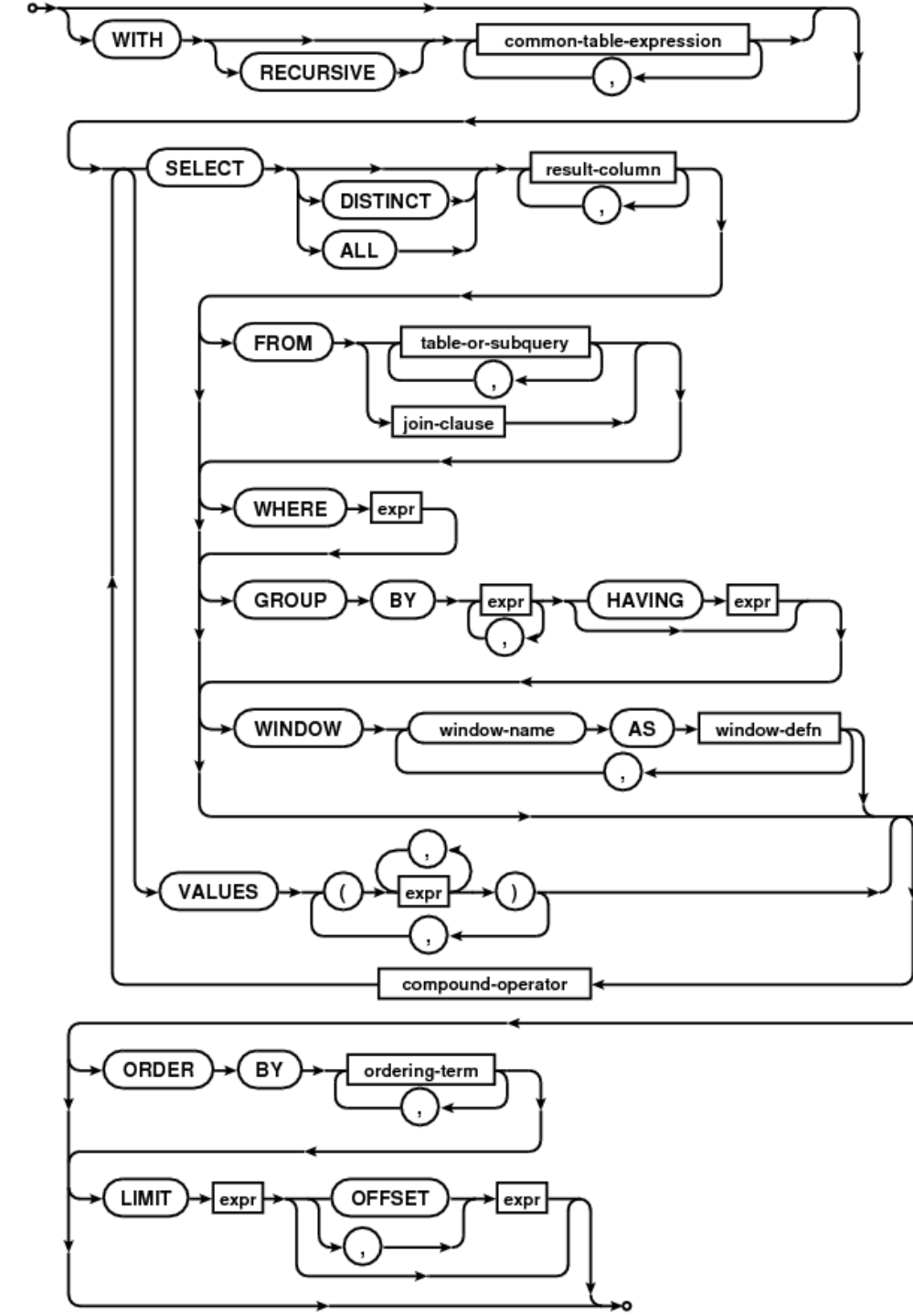
- En este tema, veremos como realizar consultas, actualizaciones y borrados de datos sobre una sola tabla.
- Trabajaremos con la siguiente base de datos, cuyos ficheros .sql para su creación e inserción de datos se encuentran colgados en Moodle:



SECCIÓN 1: CONSULTAS BÁSICAS

Consultas en SQL: SELECT

- Las consultas sobre tablas se realizan con la instrucción SELECT.
- Una consulta SELECT en SQLite tiene la siguiente estructura:



Consultas en SQL: SELECT

- La consulta más básica consistiría en consultar todos los datos de todas las columnas y filas de una tabla.
- La siguiente consulta devuelve todos los datos de todos los Proveedores:

	codiapro	cifpro	nombrpro	direcpro	cpostr	localpro
1	P001	A39184215	Bau Pi, Pablo	Alta 3, 2D	39390	Santander
2	P002	A48162311	Zar Luna, Ana	Ercilla 22, 1A	48002	Bilbao
3	P003	B28373212	Gras Leon, Luz	Pez 14, 5C dcha.	28119	Madrid
4	P004	B85392314	Gil Laso, Luis	Uria 18, 2F	85223	Oviedo

...



```
SELECT * FROM Proveedores;
```

*Tablas que
consultamos.
En este caso,
solo una tabla.*



*Proyección. Indica las columnas que queremos devolver. Un * en la proyección indica que se quiere devolver todas las columnas de la tabla*

Consultas en SQL: SELECT

- Las siguientes consultas devuelven lo mismo. Se recomienda usar la última propuesta, que da un alias a la tabla del FROM.
 - Si bien cuando se realizan consultas sobre una sola tabla esto no parece ser de mucha utilidad, veremos que facilita las cosas cuando trabajemos sobre varias tablas a la vez.

	codiapro	cifpro	nombrpro	direcpro	cpostr	localpro	
1	P001	A39184215	Bau Pi, Pablo	Alta 3, 2D	39390	Santander	(
2	P002	A48162311	Zar Luna, Ana	Ercilla 22, 1A	48002	Bilbao	(
3	P003	B28373212	Gras Leon, Luz	Pez 14, 5C dcha.	28119	Madrid	(
4	P004	B85392314	Gil Laso, Luis	Uria 18, 2F	85223	Oviedo	(

...

`SELECT * FROM Proveedores;`

Se indica explícitamente de que tabla se quieren los datos

`SELECT Proveedores.* FROM Proveedores;`

Se le da un alias a la tabla. Se recomienda que sea corto.

`SELECT pr.* FROM Proveedores pr;`

Consultas en SQL: SELECT

- Si sólo queremos un conjunto concreto de columnas, lo indicamos en la proyección:

	codigpro	cifpro	nombrpro
1	P001	A39184215	Bau Pi, Pablo
2	P002	A48162311	Zar Luna, Ana
3	P003	B28373212	Gras Leon, Luz
4	P004	B85392314	Gil Laso, Luis



```
SELECT pr.codigpro, pr.cifpro, pr.nombrpro  
FROM Proveedores pr;
```

Consultas en SQL: SELECT

- En la clausula WHERE, podemos poner condiciones que deben cumplir las filas retornadas. Sólo aquellas que las cumplan serán retornadas.
- Pueden utilizarse operadores de igualdad (=) para comprobar si el valor de la columna coincide con un valor literal.
 - Ejemplo: devolver todos los proveedores que residen en Santander.

	codiapro	cifpro	nombrpro	direcpro	cpostr	localpro	ti
1	P001	A39184215	Bau Pi, Pablo	Alta 3, 2D	39390	Santander	(:




```
SELECT pr.*  
FROM Proveedores pr  
WHERE pr.localpro = 'Santander'
```


Consultas en SQL: SELECT

- Se puede también usar el operador de desigualdad, <>, para buscar todas las filas cuyo valor en la columna no coincida con el dado.
 - Ejemplo: devolver todos los proveedores que no sea de Santander

	codiapr	cifpro	nombrpro	direcpro	cpoctor	localpro	te
1	P002	A48162311	Zar Luna, Ana	Ercilla 22, 1A	48002	Bilbao	(3
2	P003	B28373212	Gras Leon, Luz	Pez 14, 5C dcha.	28119	Madrid	(3
3	P004	B85392314	Gil Laso, Luis	Uria 18, 2F	85223	Oviedo	(3



```
SELECT pr.*  
FROM Proveedores pr  
WHERE pr.localpro <> 'Santander'
```

Consultas en SQL: SELECT

- Las fechas pueden compararse con = (igual), > (mayor), < (menor), >= (mayor o igual), y <= (menor o igual).
 - Ejemplo: devolver todos los Pedidos cuya fecha de entrada sea posterior a agosto de 2010.

	numpec	fechaped	codiapro	ivaped	fentroped
1	3	2010-10-15	P003	21	2010-12-15
2	4	2010-08-13	P001	21	2010-09-10



```
SELECT pe.*  
FROM Pedidos pe  
WHERE pe.fentroped > '2010-08-31'
```

```
SELECT pe.*  
FROM Pedidos pe  
WHERE pe.fentroped >= '2010-09-01'
```

Consultas en SQL: SELECT

- Las columnas con valor numérico pueden compararse con = (igual), > (mayor), < (menor), >= (mayor o igual), y <= (menor o igual).
 - Ejemplo: devolver todas las líneas de pedido con un descuento menor del 10%.

	numbec	numlin	codiaart	unilin	preunlin	desculin
1	1	1	0001	1	220	0
2	1	2	0003	2	295	0
3	2	1	0002	3	120	2
4	2	3	0002	5	120	0
5	3	1	0002	1	110	0
6	4	1	0002	4	120	0
7	4	2	0004	10	180	0



```
SELECT li.*  
FROM Lineas li  
WHERE li.desculin < 10
```

Consultas en SQL: SELECT

- Se pueden usar expresiones AND y OR para concatenar condiciones. También las instrucciones IN y BETWEEN, entre otras.
 - Ejemplo: devolver los datos de todos los proveedores que residen en Santander u Oviedo.

	cifpro	nombrpro	localpro
1	A39184215	Bau Pi, Pablo	Santander
2	B85392314	Gil Laso, Luis	Oviedo



```
SELECT pr.*  
FROM Proveedores pr  
WHERE pr.localpro = 'Santander'  
       OR pr.localpro = 'Oviedo'
```

```
SELECT pr.*  
FROM Proveedores pr  
WHERE pr.localpro IN ('Santander','Oviedo')
```

Consultas en SQL: SELECT

- La expresión NOT se puede usar para devolver todas aquellas filas que no cumplan con determinadas características
 - Ejemplo: devolver los datos de todos los proveedores que no residen en Santander ni Oviedo.

	ciforo	nombroro	localorc
1	A48162311	Zar Luna, Ana	Bilbao
2	B28373212	Gras Leon, Luz	Madrid



```
SELECT pr.*  
FROM Proveedores pr  
WHERE pr.localpro <> 'Santander'  
      AND pr.localpro <> 'Oviedo'
```

```
SELECT pr.*  
FROM Proveedores pr  
WHERE pr.localpro NOT IN  
      ('Santander','Oviedo')
```

Consultas en SQL: SELECT

- Se pueden usar expresiones AND y OR para concatenar condiciones. También las instrucciones IN y BETWEEN, entre otras.
 - Ejemplo: devolver los datos de todos los pedidos realizados entre junio y septiembre de 2010

	numpec	fechaped	codiapro	ivaped	fentrped
1	1	2010-05-22	P001	21	2010-06-16
2	2	2010-06-10	P002	21	2010-08-15
3	4	2010-08-13	P001	21	2010-09-10



```
SELECT pe.*  
FROM Pedidos pe  
WHERE pe.fentrped >= '2010-06-01'  
      AND pe.fentrped <='2010-09-30'
```

```
SELECT pe.*  
FROM Pedidos pe  
WHERE pe.fentrped BETWEEN '2010-06-01'  
      AND '2010-09-30'
```

Consultas en SQL: SELECT

- Se pueden usar expresiones AND y OR para concatenar condiciones. También las instrucciones IN y BETWEEN, entre otras.
 - Ejemplo: seleccionar los datos de todas las líneas con un descuento menor de 10 y mayor a 0.

	numbec	numlin	codiaart	unilin	preunlin	desculin
1	2	1	0002	3	120	2
2	2	2	0003	2	300	3



```
SELECT 1.* FROM Lineas l  
WHERE l.desculin < 10 AND l.desculin >0
```

```
SELECT 1.* FROM Lineas l  
WHERE l.desculin < 10 AND l.desculin <>0
```

```
SELECT 1.* FROM Lineas l  
WHERE l.desculin BETWEEN 0 AND 10
```



Consultas en SQL: SELECT

- Se pueden comprar valores de dos columnas.
 - Ejemplo 1: devolver los datos de todos los artículos cuyo stock es mayor al stock mínimo

	codiaart	descart	preunart	stockart	stockmir	fecbaia
1	0001	MESA OFICINA 90x1,80	225	100	1	NULL
2	0002	SILLA ERGONOMIC MOD. MX	120	25	1	NULL
3	0003	ARMARIO DIPLOMATIC	300	2	1	NULL
4	0004	ARCHIVADOR MOD. TR	180	3	1	NULL

`SELECT a.* FROM Articulos a
WHERE a.stockart > a.stockmin`



- Ejemplo 2: devolver todos los pedidos que se hayan enviado el mismo día en que se realizaron.

	numdec	fechaped	codiaopr	ivaped	fentroped
1	5	2010-11-13	P004	18	2010-11-13

`SELECT pe.* FROM Pedidos pe
WHERE pe.fechaped = pe.fentroped`



Consultas en SQL: SELECT

- Cuando queremos comprobar si un valor es o no nulo, se utiliza el operador IS:
 - Ejemplo 1: devolver los datos de todos los proveedores que tienen correo electrónico.

```
SELECT pr.* FROM Proveedores pr  
WHERE pr.emailpro IS NULL
```

	codiapi	ciforo	nombrpro	direcpro	cpostor	localpro	telepro	faxpro	emailpro	procepi
1	P003	B28373212	Gras Leon, Luz	Pez 14, 5C dcha.	28119	Madrid	(34) 916 677 829	(34) 916 677 889	NULL	UE

- Ejemplo 2: devolver los datos de todos los proveedores que no tienen correo electrónico.

```
SELECT pr.* FROM Proveedores pr  
WHERE pr.emailpro IS NOT NULL
```

	codiapi	ciforo	nombrpro	direcpro	cpostor	localpro	telepro	faxpro	emailpro	procepi
1	P001	A39184215	Bau Pi, Pablo	Alta 3, 2D	39390	Santander	(34) 942 223 345	(34) 942 223 344	mailto:baupi@eresmas.es	UE
2	P002	A48162311	Zar Luna, Ana	Ercilla 22, 1A	48002	Bilbao	(34) 947 865 413	(34) 947 865 434	mailto:zarana@yahoo.es	UE
3	P004	B85392314	Gil Laso, Luis	Uria 18, 2F	85223	Oviedo	(34) 952 345 6632	(34) 952 345 678	mailto:luzgras@kmail.com	UE

Consultas en SQL: SELECT

- Se puede usar el LIKE para evaluar expresiones regulares sobre columnas de tipo texto.
 - Ejemplo: devolver los códigos de los proveedores cuyo código postal comience por 4.

```
SELECT p.* FROM proveedores p
WHERE p.cpostpro LIKE ('4%')
```

	codiapro	cifpro	nombrpro	direcpro	cpostr	localpro	telefono	faxpro	emailpro	procedi
1	P002	A48162311	Zar Luna, Ana	Ercilla 22, 1A	48002	Bilbao	(34) 947 865 413	(34) 947 865 434	mailto:zarana@yahoo.es	UE


%: cualquier cadena de caracteres

_: cualquier carácter.

Consultas en SQL: SELECT

- Operadores de agrupación. Sirven para agrupar los valores en base a un criterio. Por ejemplo: AVG (media), MAX (máximo), MIN (mínimo), COUNT (número de coincidencias)...

- Ejemplo 1: obtener el precio del artículo más caro



	maximo
1	300

```
SELECT MAX(a.preunart) maximo  
FROM Articulos a
```


- Ejemplo 2: obtener el pedido más antiguo



	minimo
1	2010-05-22

```
SELECT MIN(pe.fechaped) minimo  
FROM Pedidos pe
```


- Ejemplo 3: obtener el iva medio de los pedidos al proveedor 'P001'



	media
1	19.5

```
SELECT AVG(pe.ivaped) media  
FROM Pedidos pe WHERE pe.codigpro = 'P001'
```

- Ejemplo 4: número de proveedores en la base de datos



	numero
1	4


```
SELECT COUNT(pr.*) num  
FROM Proveedores pr
```

Consultas en SQL: SELECT

- Los resultados de las consultas se pueden ordenar con la clausula ORDER BY.
 - Ejemplo 1: obtener los datos de los pedidos ordenados de forma ascendente por fecha de pedido y de forma descendente por fecha de entrega

	numdec	fechaped	codiaorc	ivaped	fentroed
1	1	2010-05-22	P001	21	2010-06-16
2	2	2010-06-10	P002	18	2010-08-15
3	4	2010-08-13	P001	18	2010-09-10
4	3	2010-10-15	P003	18	2010-12-15
5	5	2010-11-13	P004	21	2010-11-13


```
SELECT pe.*  
FROM Pedidos pe  
ORDER BY pe.fechaped ASC,  
         pe.fecentrped ASC
```



- Ejemplo 2: obtener los datos de los artículos que no se hayan dado de baja ordenados por precio, de menor a mayor. En caso de empate en el precio, ordenarlos por stock disponible, también de menor a mayor.

	codiaart	descart	preunart	stockart	stockmir	fecbaia
1	0002	SILLA ERGONOMIC MOD. MX	120	25	1	NULL
2	0004	ARCHIVADOR MOD. TR	180	3	1	NULL
3	0001	MESA OFICINA 90x1,80	225	100	1	NULL
4	0003	ARMARIO DIPLOMATIC	300	2	1	NULL

```
SELECT a.*  
FROM Articulos a  
WHERE a.fecbaja IS NOT NULL  
ORDER BY a.preunart ASC, a.stockart ASC
```



Consultas en SQL: SELECT

- También se pueden realizar operaciones con los valores de una o varias columnas.
 - Ejemplo 1: devolver el precio total de cada línea (sin aplicar el descuento), junto con el resto de datos de la línea.

	numpec	numlin	codiaar	unilin	preunlin	desculir	precioT
1	1	1	0001	1	220	0	220
2	1	2	0003	2	295	0	590
3	2	1	0002	3	120	2	360
4	2	2	0003	2	300	3	600

Alias para la nueva columna

```
SELECT l.*,  
       (l.unilin*l.preunlin) precioTotalSinDesc  
FROM Lineas l
```

- Ejemplo 2: devolver los datos de todos los proveedores que no tienen correo electrónico.

	codiaar	descart	preunar	stockart	stockmi	fecbaia	stockRe
1	0001	MESA OFICINA 90x1,80	225	100	1	NULL	99
2	0002	SILLA ERGONOMIC MOD. MX	120	25	1	NULL	24
3	0003	ARMARIO DIPLOMATIC	300	2	1	NULL	1
4	0004	ARCHIVADOR MOD. TR	180	3	1	NULL	2

```
SELECT a.*,  
       (a.stockart-a.stockmin) stockRestante  
FROM Articulos a;
```

Consultas en SQL: SELECT

- Las operaciones aritméticas también se pueden usar en el WHERE o incluso el ORDER BY, entre otros:
 - Ejemplo: devolver los datos de las líneas cuyo precio total sin descuento sea mayor a 300:

	numbec	numlin	codigaar	unilin	preunlin	desculir	precioT
1	1	2	0003	2	295	0	590
2	2	1	0002	3	120	2	360
3	2	2	0003	2	300	3	600
4	2	3	0002	5	120	0	600
5	4	1	0002	4	120	0	480
6	4	2	0004	10	180	0	1800



```
SELECT l.*,  
       (l.unilin*l.preunlin) precioTotalSinDesc  
FROM Lineas l  
WHERE precioTotalSinDesc > 300
```

```
SELECT l.*,  
       (l.unilin*l.preunlin) precioTotalSinDesc  
FROM Lineas l  
WHERE (l.unilin*l.preunlin) > 300
```

AHORA TOCA HACER EL EJERCICIO 3

SECCIÓN 2: ACTUALIZADOS Y BORRADOS BÁSICOS

Actualización de datos en SQL: UPDATE

- La clausula UPDATE sirve para actualizar datos en las filas.
 - Ejemplo: actualizar al 21 el iva de todos los pedidos

```
UPDATE pedidos SET ivaped = 21;
```

Actualización de datos en SQL: UPDATE

- Se puede actualizar varios campos a la vez, y sólo para las filas que cumplan las condiciones del WHERE.

- Ejemplo 1: actualiza el email del proveedor con código 'P003' a 'mailto:luzgras@kmail.com'

```
UPDATE proveedores SET emailpro = 'mailto:luzgras@kmail.com'  
WHERE codigpro = 'P004';
```

- Ejemplo 2: poner a 0 el stock y stock mínimo de todos los artículos con fecha de baja no nula

```
UPDATE articulos SET stockart = 0, stockmin = 0  
WHERE fecbaja IS NOT null
```

Borrado de datos en SQL: DELETE

- El borrado de datos se ejecuta con DELETE.
 - Ejemplo 1: borrar todos los datos de la tabla líneas.

```
DELETE FROM lineas;
```

- Ejemplo 2: borrar los pedidos del proveedor 'P004'

```
DELETE FROM pedidos WHERE codigpro = 'P004';
```

¡OJO! Si hay filas en otras tablas referenciado a las filas de la tabla en donde realizamos el DELETE, la instrucción dará error

AHORA TOCA HACER EL EJERCICIO 5

SECCIÓN 3: FUNCIONES ÚTILES (SIN AGREGACIONES) Y ESTRUCTURAS CONDICIONALES (CASE)

Funciones útiles en SQLite

- Sqlite ofrece las siguientes funciones:

- [abs\(X\)](#)
- [changes\(\)](#)
- [char\(X1,X2,...,XN\)](#)
- [coalesce\(X,Y,...\)](#)
- [glob\(X,Y\)](#)
- [hex\(X\)](#)
- [ifnull\(X,Y\)](#)
- [instr\(X,Y\)](#)
- [last insert rowid\(\)](#)
- [length\(X\)](#)
- [like\(X,Y\)](#)
- [like\(X,Y,Z\)](#)

- [likelihood\(X,Y\)](#)
- [likely\(X\)](#)
- [load extension\(X\)](#)
- [load extension\(X,Y\)](#)
- [lower\(X\)](#)
- [ltrim\(X\)](#)
- [ltrim\(X,Y\)](#)
- [max\(X,Y,...\)](#)
- [min\(X,Y,...\)](#)
- [nullif\(X,Y\)](#)
- [printf\(FORMAT,...\)](#)
- [quote\(X\)](#)

- [random\(\)](#)
- [randomblob\(N\)](#)
- [replace\(X,Y,Z\)](#)
- [round\(X\)](#)
- [round\(X,Y\)](#)
- [rtrim\(X\)](#)
- [rtrim\(X,Y\)](#)
- [soundex\(X\)](#)
- [sqlite compileoption get\(N\)](#)
- [sqlite compileoption used\(X\)](#)
- [sqlite offset\(X\)](#)
- [sqlite source id\(\)](#)
- [sqlite version\(\)](#)

- [substr\(X,Y\)](#)
- [substr\(X,Y,Z\)](#)
- [total changes\(\)](#)
- [trim\(X\)](#)
- [trim\(X,Y\)](#)
- [typeof\(X\)](#)
- [unicode\(X\)](#)
- [unlikely\(X\)](#)
- [upper\(X\)](#)
- [zeroblob\(N\)](#)

Funciones útiles en SQLite

- Algunos ejemplos de uso de funciones SQLite:
 - Ejemplo 1: Devolver el código de los proveedores junto con su email, y en una nueva columna, el mismo email eliminando el 'mailto:'

	codigpro	emailpro	email
1	P001	mailto:baupi@eresmas.es	baupi@eresmas.es
2	P002	mailto:zarana@yahoo.es	zarana@yahoo.es
3	P003	NULL	NULL
4	P004	NULL	NULL



```
SELECT pr.*,  
replace(pr.emailpro,'mailto:','') as email  
FROM Proveedores pr;
```

- Ejemplo 2: Devolver el tipo de dato, en cada fila, de cada columna de la tabla artículos

	typeof(a.c	typeof(a.c	typeof(a.p	typeof(a.s	typeof(a.s	typeof(a.f
1	text	text	real	integer	integer	text
2	text	text	integer	integer	integer	text
3	text	text	integer	integer	integer	null
4	text	text	real	integer	integer	null



```
SELECT typeof(a.codigart), typeof(a.descrart),  
typeof(a.preunart), typeof(a.stockart),  
typeof(a.stockmin), typeof(a.fecbaja)  
FROM Articulos a;
```

Funciones útiles en SQLite

- Además de las funciones, existen otros comandos útiles:
 - Ejemplo: Devolver, concatenados en una columna y separados por comas, la dirección, código postal y localidad de los proveedores.

	codigpro	direccionCompleta
1	P001	Calle Alta 3, 2D, Santander, 39390
2	P002	Calle Ercilla 22, 1A, Bilbao, 48002
3	P003	Calle Pez 14, 5C dcha., Madrid, 28119
4	P004	Calle Uria 18, 2F, Oviedo, 85223



```
SELECT pr.codigpro, 'Calle ' || pr.direcpro  
|| ', ' || pr.localpro || ', ' ||  
pr.cpostpro as direccionCompleta  
FROM Proveedores pr;
```


Funciones útiles en SQLite

- Se puede usar "CASE" para implementar estructuras condicionales en una consulta:
 - Ejemplo 1: Devolver el código de los proveedores, junto con el mensaje "no tiene correo" si su email es nulo. En caso contrario, devolver su email.

	codigpro	email
1	P001	mailto:baupi@eresmas.es
2	P002	mailto:zarana@yahoo.es
3	P003	no tiene correo
4	P004	no tiene correo



```
SELECT pr.codigpro,  
CASE WHEN pr.emailpro IS NULL  
      THEN 'no tiene correo' ELSE pr.emailpro END as  
email  
FROM Proveedores pr;
```

- Ejemplo 2: Devolver los artículos junto con una columna que indique "precioAlto" cuando el precio sea mayor a 300, "precioMedio" cuando el precio esté entre 150 y 300, y "precioBajo" cuando este sea menor de 150

	codigart	descart	preunart	stockart	stockmin	fecbaja	precio
1	0001	MESA OFICINA 90x1,80	247.500000000000003	100	1	2018-11-15	precioMedio
2	0002	SILLA ERGONOMIC MOD. MX	132	1	1	2018-11-15	precioBajo
3	0003	ARMARIO DIPLOMATIC	330	2	1	NULL	precioAlto
4	0004	ARCHIVADOR MOD. TR	198.000000000000003	3	1	NULL	precioMedio

```
SELECT a.*,  
CASE WHEN a.preunart > 300 THEN 'precioAlto' WHEN a.preunart  
      >= 150 AND a.preunart <= 300 THEN 'precioMedio' ELSE  
      'precioBajo' END precio FROM Articulos a;
```

AHORA TOCA HACER EL EJERCICIO 6