ORIE 3120

Lecture 3: SQL #2
[Basic queries (SELECT, WHERE, ORDER BY, …), schema design, DDL, DML]

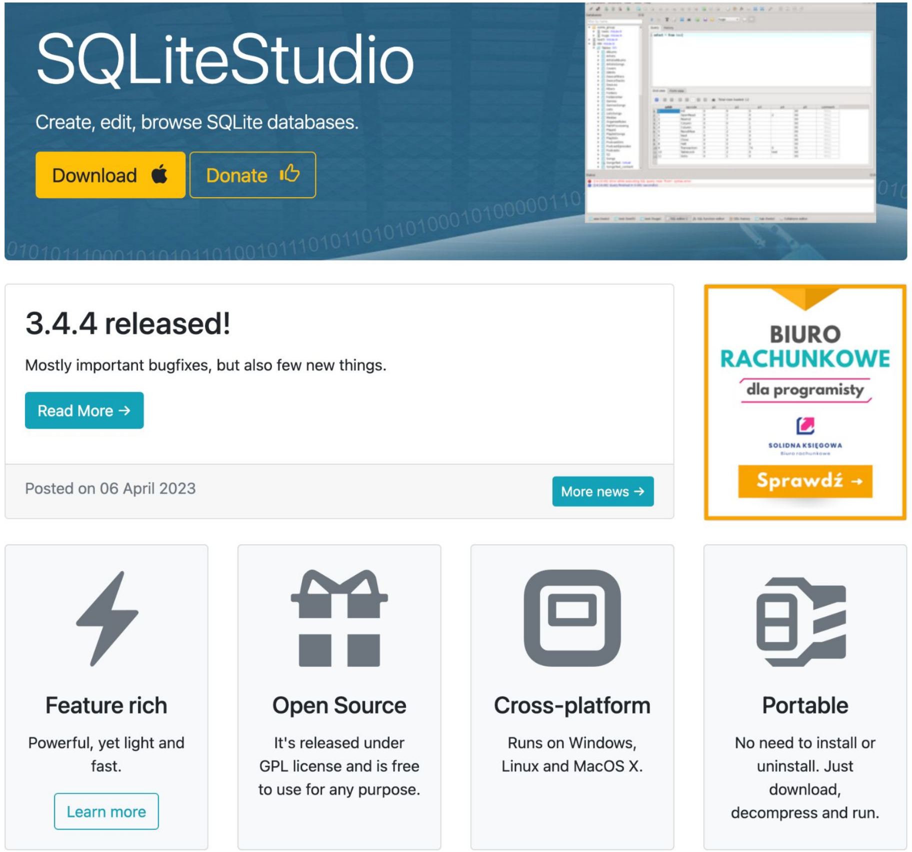
Logistics

- Office Hours start next week. Check google calendar (linked in course syllabus on Canvas) for times & locations.
- HW1 (on SQL) posted by Sat Jan 27, due 11:59pm eastern on Mon Feb 5
- I will also post some questions on prerequisite ENGRD 2700 material
    - These will be on a *future* HW
    - If they seem difficult, that's a signal to deepen your understanding of the prerequisite material
- Recitations start next week, first one on Tues Jan 30
- Supplemental reading from Canvas:
    - 01-HoffmanSQL.pdf, pages 1-5
    - 02-SQLiteFunctionsfor3120.pdf
    - Topics in the reading but not covered in the slides, like IN and BETWEEN, won't be on the exam or HW

Logistics

- Waiting list —
  - ORIE course staff will be issuing enrollment codes this morning
  - I will make a canvas announcement this evening about the # of people left on the list
  - If you don't get an enrollment code, some people will drop the course when the HW comes out. HW1 is due Feb 5 & the add deadline is Feb 5
  - If you are planning to drop the course, please do so by Feb 3
- Repeat from Tuesday:
  - If you are not registered and don't have access to Canvas & want to try to get a seat in the class, email Yuheng Wang, yw634@cornell.edu
  - If you have questions about the waiting list, email ORIE-UG-Support@cornell.edu
  - Please don't email me — I won't be able to respond via email

Please install SQLite Studio



SQLiteStudio

Create, edit, browse SQLite databases.

3.4.4 released!

Mostly important bugfixes, but also few new things.

Posted on 06 April 2023

Feature rich

Powerful, yet light and fast.

Open Source

It's released under GPL license and is free to use for any purpose.

Cross-platform

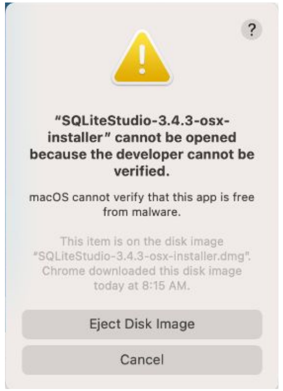
Runs on Windows, Linux and MacOS X.

Portable

No need to install or uninstall. Just download, decompress and run.

https://sqlitestudio.pl/

If you get this error message on a Mac

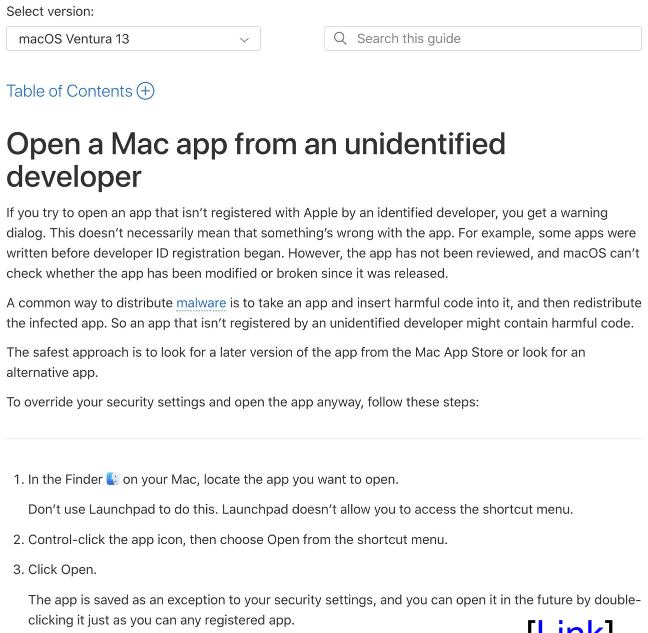


"SQLiteStudio-3.4.3-osx-installer" cannot be opened because the developer cannot be verified.

macOS cannot verify that this app is free from malware.

This item is on the disk image "SQLiteStudio-3.4.3-osx-installer.dmg". Chrome downloaded this disk image today at 8:15 AM.

Then follow these instructions



Open a Mac app from an unidentified developer

If you try to open an app that isn't registered with Apple by an identified developer, you get a warning dialog. This doesn't necessarily mean that something's wrong with the app. For example, some apps were written before developer ID registration began. However, the app has not been reviewed, and macOS can't check whether the app has been modified or broken since it was released.

A common way to distribute malware is to take an app and insert harmful code into it, and then redistribute the infected app. So an app that isn't registered by an unidentified developer might contain harmful code.

The safest approach is to look for a later version of the app from the Mac App Store or look for an alternative app.

To override your security settings and open the app anyway, follow these steps:

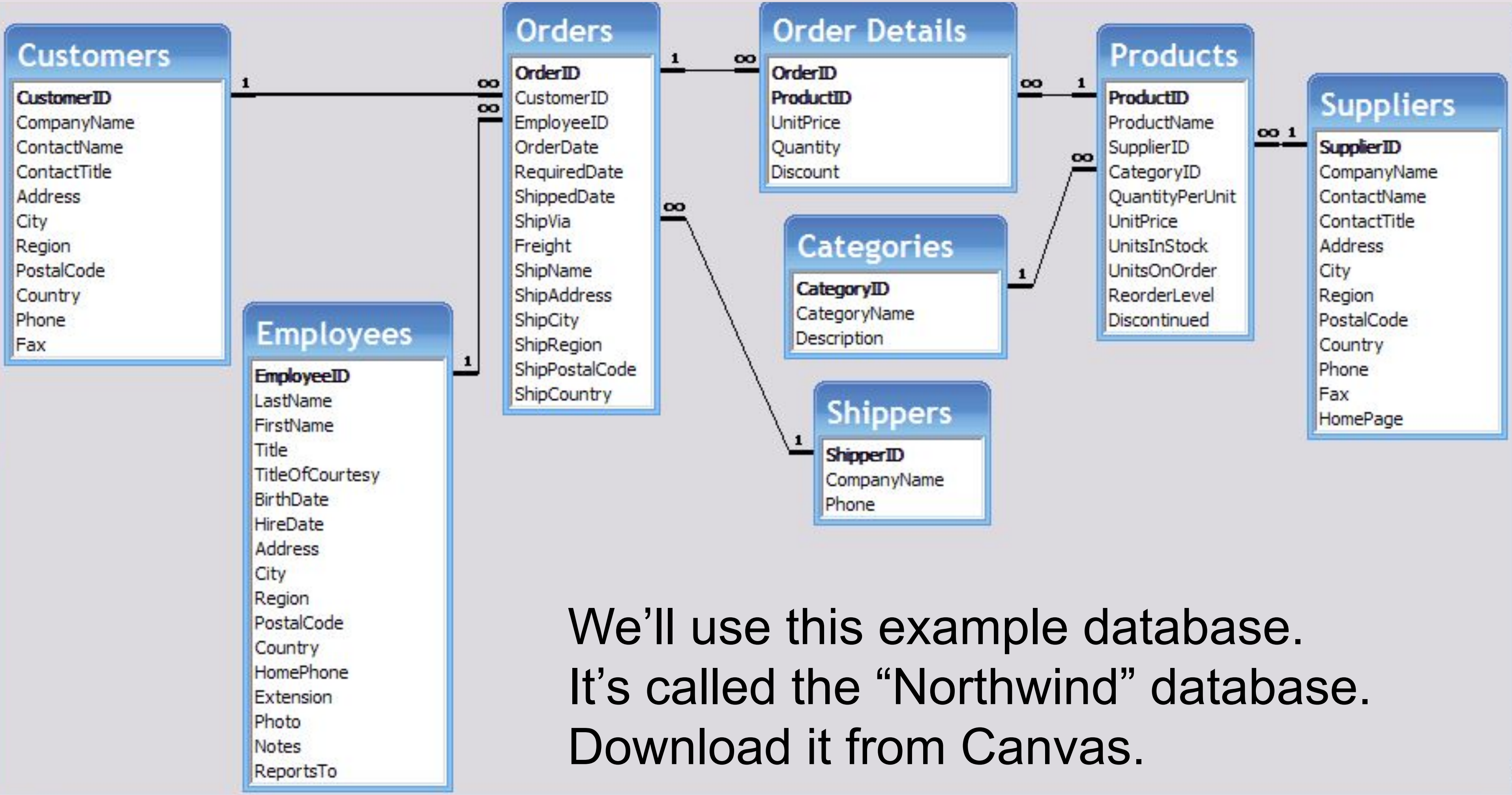
1. In the Finder on your Mac, locate the app you want to open.

   Don't use Launchpad to do this. Launchpad doesn't allow you to access the shortcut menu.
2. Control-click the app icon, then choose Open from the shortcut menu.
3. Click Open.

   The app is saved as an exception to your security settings, and you can open it in the future by double-clicking it just as you can any registered app.

[Link]

You may need to do this twice, once for the installer, and once for SQLite Studio itself



We’ll use this example database.
It’s called the “Northwind” database.
Download it from Canvas.

Queries

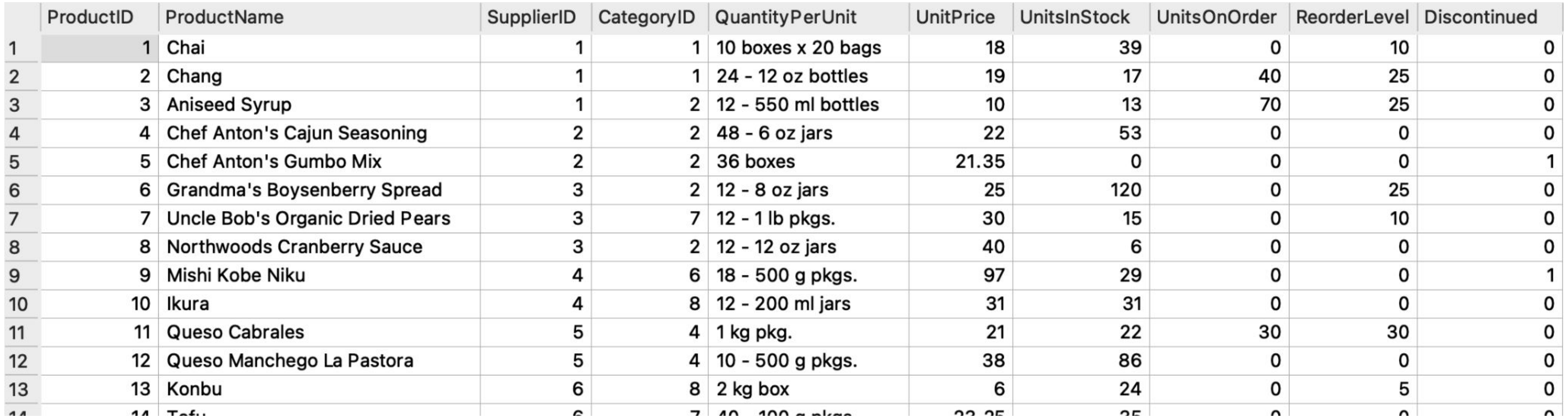
Queries

- A query is a statement describing a data request.
- There are a small set of keywords
- By convention, we capitalize them (SELECT, AS, WHERE, etc.)
- There is a prescribed syntax

Here’s a query

```
SELECT * FROM Products
```

Here’s that query’s result



	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
1	1	Chai	1	1	10 boxes x 20 bags	18	39	0	10	0
2	2	Chang	1	1	24 - 12 oz bottles	19	17	40	25	0
3	3	Aniseed Syrup	1	2	12 - 550 ml bottles	10	13	70	25	0
4	4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22	53	0	0	0
5	5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35	0	0	0	1
6	6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25	120	0	25	0
7	7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30	15	0	10	0
8	8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40	6	0	0	0
9	9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97	29	0	0	1
10	10	Ikura	4	8	12 - 200 ml jars	31	31	0	0	0
11	11	Queso Cabrales	5	4	1 kg pkg.	21	22	30	30	0
12	12	Queso Manchego La Pastora	5	4	10 - 500 g pkgs.	38	86	0	0	0
13	13	Konbu	6	8	2 kg box	6	24	0	5	0

- It looks like a table, and can be stored as one.
- When we store a query’s result, we call it a “view”

How did we get this?

```
SELECT * FROM Products
```

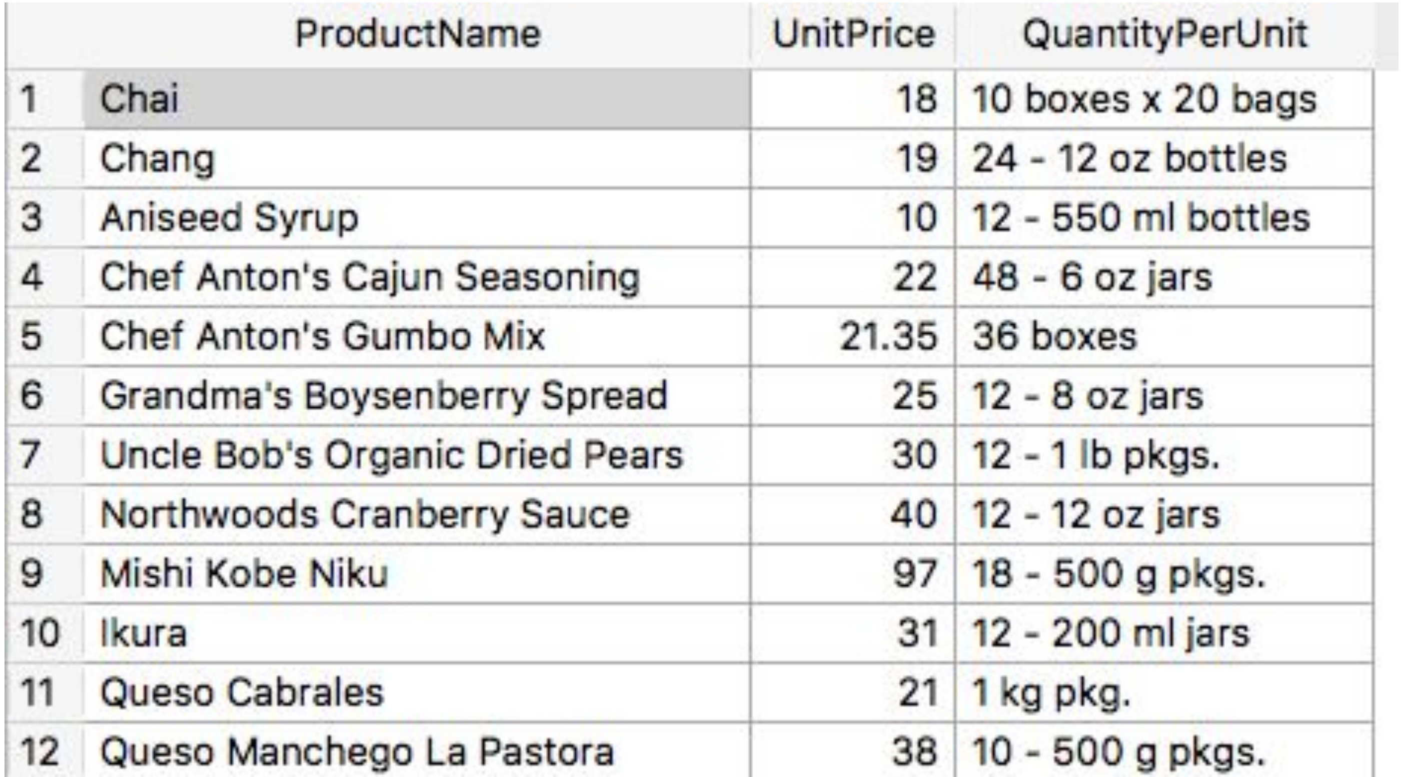
- “*” means “all of the fields”
- “FROM Products” means “get it from the table Products”
- We got all of the records.
- We can be selective and only get some of them.

We can choose which fields to get

```
SELECT ProductName, UnitPrice, QuantityPerUnit
FROM Products
```

- Here we only look at 3 fields from the table Products
- We look at all the rows

Here’s that query’s result



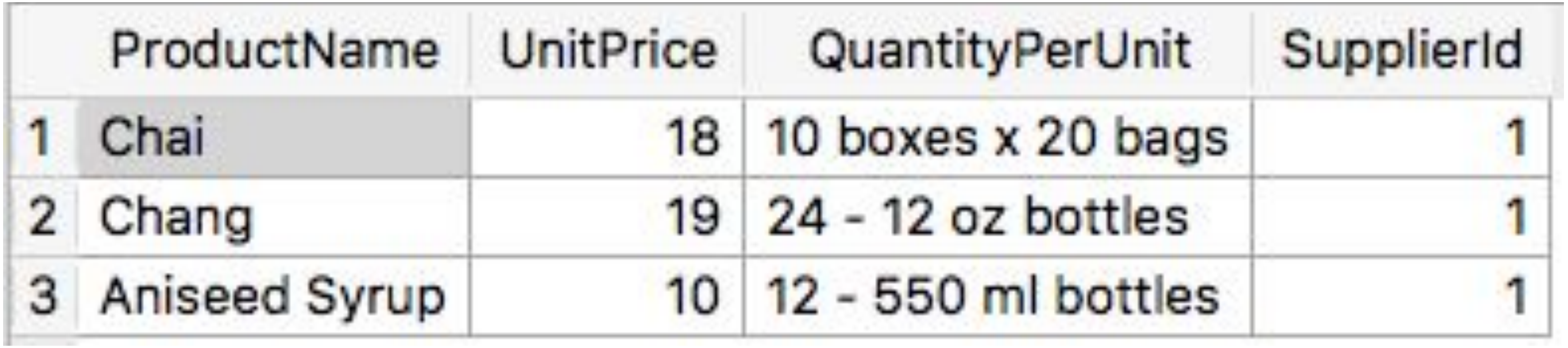
	ProductName	UnitPrice	QuantityPerUnit
1	Chai	18	10 boxes x 20 bags
2	Chang	19	24 - 12 oz bottles
3	Aniseed Syrup	10	12 - 550 ml bottles
4	Chef Anton's Cajun Seasoning	22	48 - 6 oz jars
5	Chef Anton's Gumbo Mix	21.35	36 boxes
6	Grandma's Boysenberry Spread	25	12 - 8 oz jars
7	Uncle Bob's Organic Dried Pears	30	12 - 1 lb pkgs.
8	Northwoods Cranberry Sauce	40	12 - 12 oz jars
9	Mishi Kobe Niku	97	18 - 500 g pkgs.
10	Ikura	31	12 - 200 ml jars
11	Queso Cabrales	21	1 kg pkg.
12	Queso Manchego La Pastora	38	10 - 500 g pkgs.

WHERE

WHERE selects some of the rows

```
SELECT ProductName, UnitPrice, QuantityPerUnit, SupplierId
FROM Products
WHERE SupplierId=1
```

- We selected the same 3 columns from the table Products, plus the column SupplierId
- We got only the products from Supplier #1

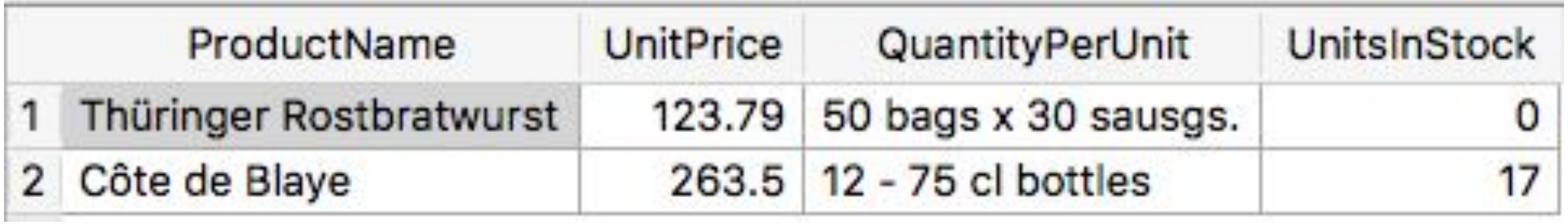


	ProductName	UnitPrice	QuantityPerUnit	SupplierId
1	Chai	18	10 boxes x 20 bags	1
2	Chang	19	24 - 12 oz bottles	1
3	Aniseed Syrup	10	12 - 550 ml bottles	1

WHERE selects some of the rows

```
SELECT ProductName, UnitPrice, QuantityPerUnit, UnitsInStock
FROM Products
WHERE UnitPrice > 100
```

- Here we only see products that cost more than $100 per unit

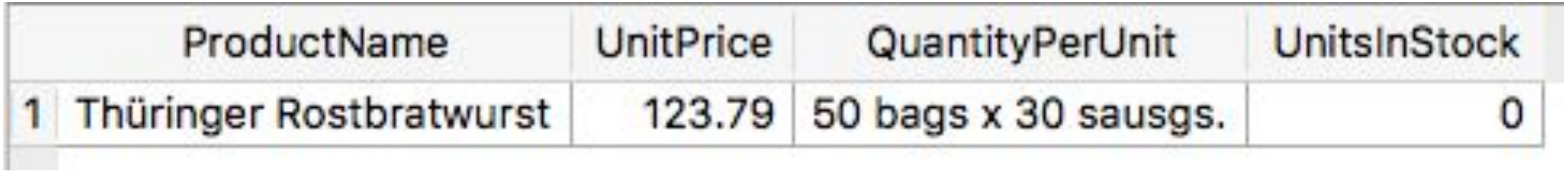


	ProductName	UnitPrice	QuantityPerUnit	UnitsInStock
1	Thüringer Rostbratwurst	123.79	50 bags x 30 sausgs.	0
2	Côte de Blaye	263.5	12 - 75 cl bottles	17

AND lets you filter on multiple conditions

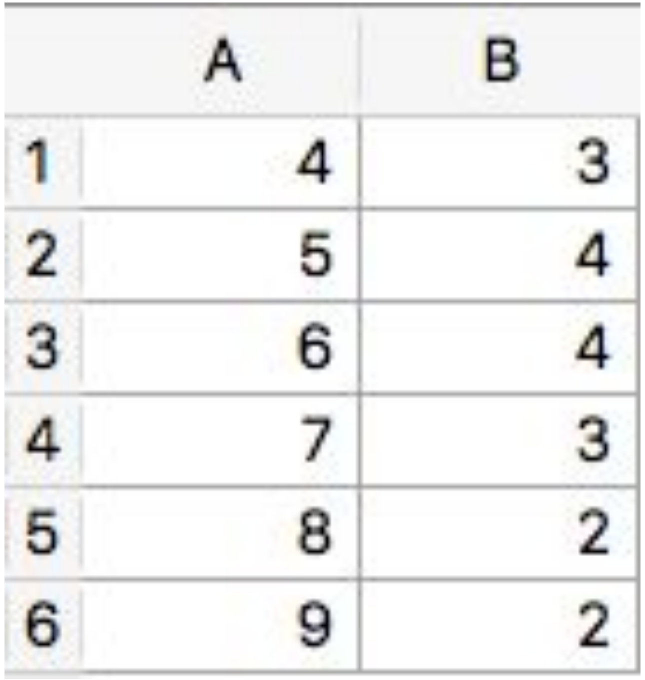
```
SELECT ProductName, UnitPrice, QuantityPerUnit, UnitsInStock
FROM Products
WHERE UnitPrice > 100
AND UnitsInStock = 0
```

- Here we only at products that cost more than $100 per unit and that have no units in stock



	ProductName	UnitPrice	QuantityPerUnit	UnitsInStock
1	Thüringer Rostbratwurst	123.79	50 bags x 30 sausgs.	0

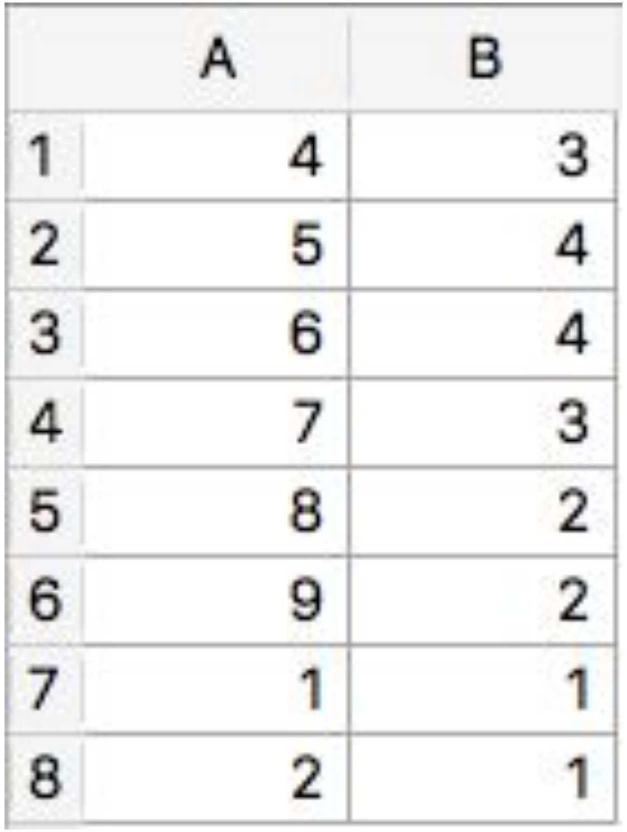
Lec 3, Q1: What WHERE clause could have generated this result?



	A	B
1	4	3
2	5	4
3	6	4
4	7	3
5	8	2
6	9	2

The query is:
SELECT A,B FROM T WHERE ….

(a) WHERE A>3 AND B<5
(b) WHERE A>3
(c) WHERE B>5
(d) WHERE A>3 AND B>5
(e) two or more of the above



	A	B
1	4	3
2	5	4
3	6	4
4	7	3
5	8	2
6	9	2
7	1	1
8	2	1

Lec 3, Q2: What WHERE clause could have generated this result?

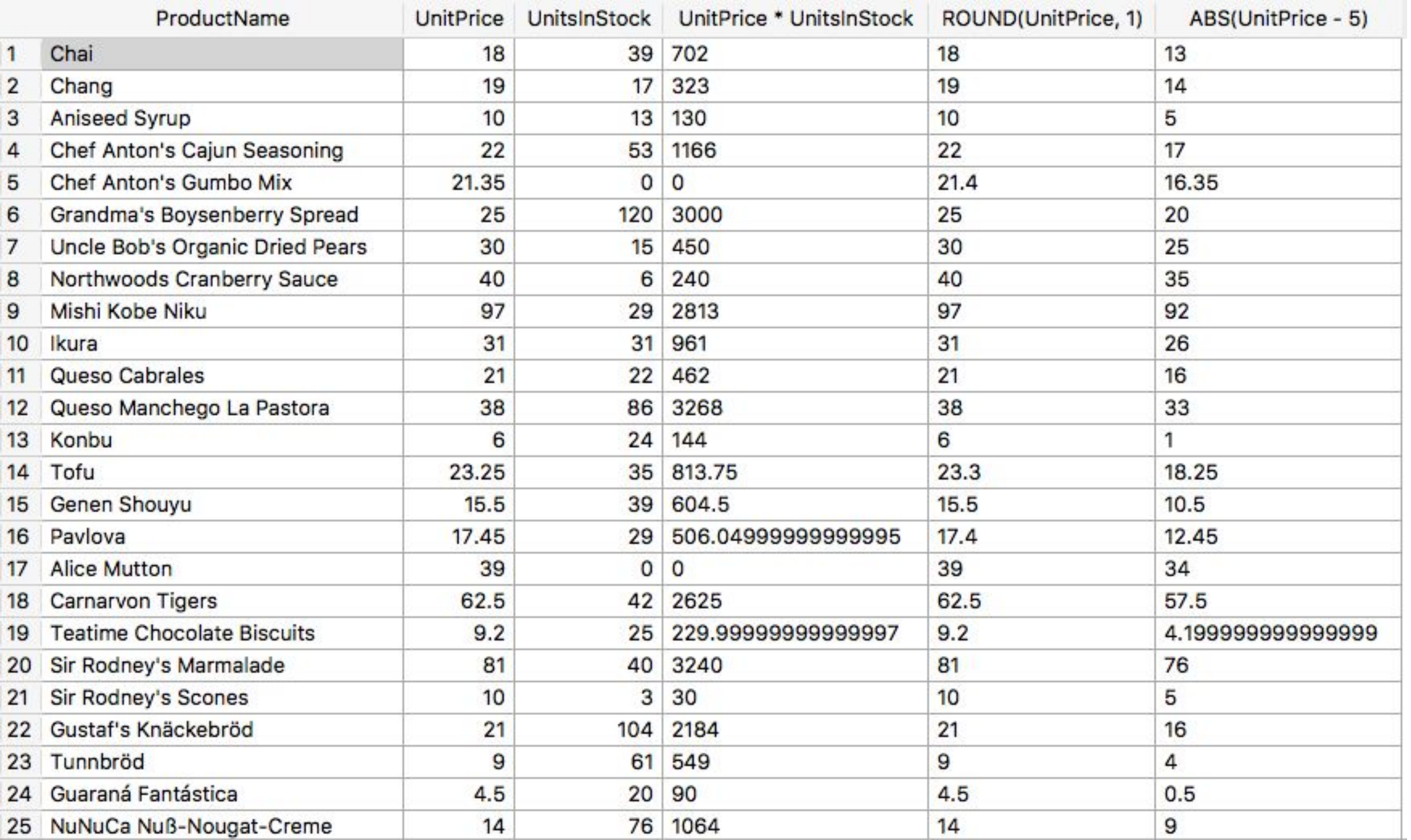
The query is:
SELECT A,B FROM T WHERE ….

(a) WHERE A>3 AND B<5
(b) WHERE A>3
(c) WHERE A<3
(d) WHERE (A>3 AND B<5) OR A<3
(e) two or more of the above

Calculated columns

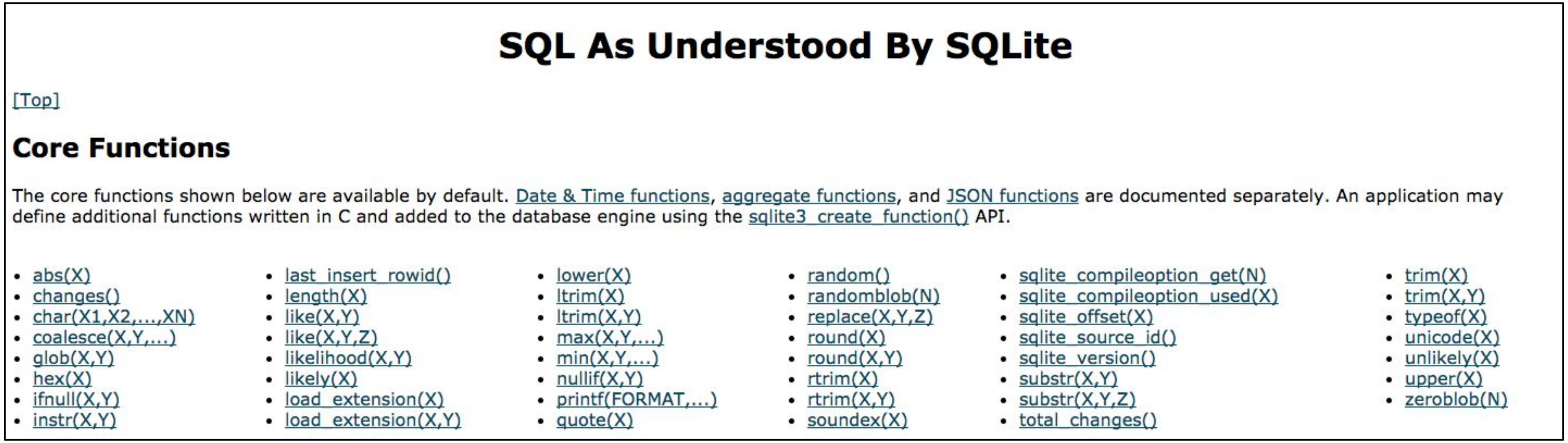
You can do some math with your fields

```
SELECT ProductName,
       UnitPrice,
       UnitsInStock,
       UnitPrice*UnitsInStock,
       ROUND(UnitPrice,1),
       ABS(UnitPrice-5)
FROM Products
```



	ProductName	UnitPrice	UnitsInStock	UnitPrice * UnitsInStock	ROUND(UnitPrice, 1)	ABS(UnitPrice - 5)
1	Chai	18	39	702	18	13
2	Chang	19	17	323	19	14
3	Aniseed Syrup	10	13	130	10	5
4	Chef Anton's Cajun Seasoning	22	53	1166	22	17
5	Chef Anton's Gumbo Mix	21.35	0	0	21.4	16.35
6	Grandma's Boysenberry Spread	25	120	3000	25	20
7	Uncle Bob's Organic Dried Pears	30	15	450	30	25
8	Northwoods Cranberry Sauce	40	6	240	40	35
9	Mishi Kobe Niku	97	29	2813	97	92
10	Ikura	31	31	961	31	26
11	Queso Cabrales	21	22	462	21	16
12	Queso Manchego La Pastora	38	86	3268	38	33
13	Konbu	6	24	144	6	1
14	Tofu	23.25	35	813.75	23.3	18.25
15	Genen Shouyu	15.5	39	604.5	15.5	10.5
16	Pavlova	17.45	29	506.04999999999995	17.4	12.45
17	Alice Mutton	39	0	0	39	34
18	Carnarvon Tigers	62.5	42	2625	62.5	57.5
19	Teatime Chocolate Biscuits	9.2	25	229.99999999999997	9.2	4.199999999999999
20	Sir Rodney's Marmalade	81	40	3240	81	76
21	Sir Rodney's Scones	10	3	30	10	5
22	Gustaf's Knäckebröd	21	104	2184	21	16
23	Tunnbröd	9	61	549	9	4
24	Guaraná Fantástica	4.5	20	90	4.5	0.5
25	NuNuCa Nuß-Nougat-Creme	14	76	1064	14	9

But not very much math



SQL As Understood By SQLite

[Top]

Core Functions

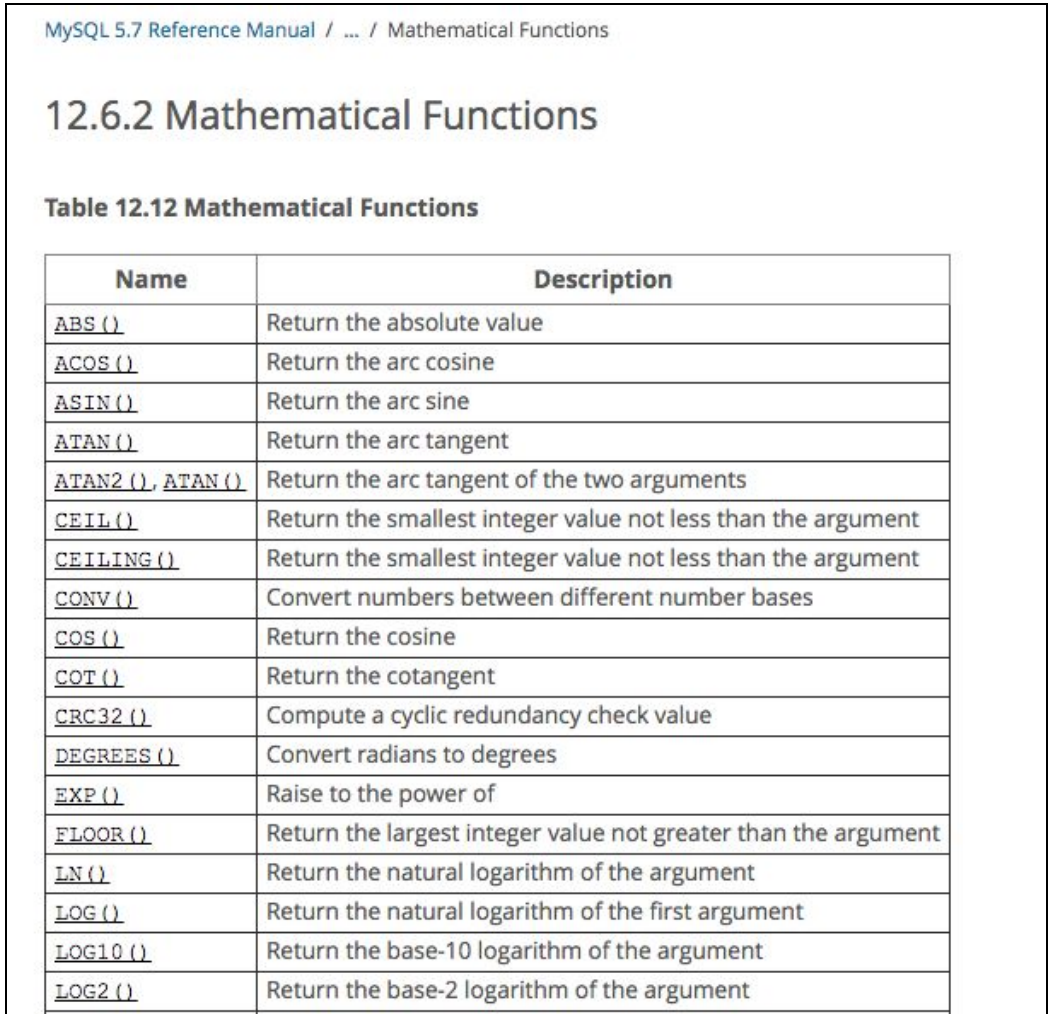
The core functions shown below are available by default. Date & Time functions, aggregate functions, and JSON functions are documented separately. An application may define additional functions written in C and added to the database engine using the sqlite3_create_function() API.

- 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)

Figure: SQLite documentation, from https://www.sqlite.org/lang_corefunc.html

SQLite supports these math functions: abs, max, min, random, round.
We'll talk about other functions in a bit.

Figure: MySQL documentation



MySQL 5.7 Reference Manual / ... / Mathematical Functions

12.6.2 Mathematical Functions

Table 12.12 Mathematical Functions

Name	Description
ABS()	Return the absolute value
ACOS()	Return the arc cosine
ASIN()	Return the arc sine
ATAN()	Return the arc tangent
ATAN2(), ATAN()	Return the arc tangent of the two arguments
CEIL()	Return the smallest integer value not less than the argument
CEILING()	Return the smallest integer value not less than the argument
CONV()	Convert numbers between different number bases
COS()	Return the cosine
COT()	Return the cotangent
CRC32()	Compute a cyclic redundancy check value
DEGREES()	Convert radians to degrees
EXP()	Raise to the power of
FLOOR()	Return the largest integer value not greater than the argument
LN()	Return the natural logarithm of the argument
LOG()	Return the natural logarithm of the first argument
LOG10()	Return the base-10 logarithm of the argument
LOG2()	Return the base-2 logarithm of the argument

Other variants of SQL let you do more math

SQLite lets you manipulate strings

```
SELECT QuantityPerUnit,
    LTRIM(QuantityPerUnit,'0123456789'),
    SUBSTR(QuantityPerUnit,2,8),
    SUBSTR(QuantityPerUnit,-2,2),
    LENGTH(QuantityPerUnit),
    UPPER(QuantityPerUnit)
FROM Products
```

SQLite lets you manipulate strings

upper and **lower**: converts to upper and lower case

length: returns the length of the string in characters

Examples:

- LENGTH('orie 3120') returns 9
- UPPER('orie 3120') returns 'ORIE 3120'

SQLite lets you manipulate strings

ltrim(*X*,*Y*): removes any and all characters that appear in Y from the left side of X.
Note that the *order* of the characters in Y does not matter.

ltrim(*X*): removes spaces from the left side of X

rtrim: like ltrim, but removes from the right side

trim: like ltrim, but removes from both sides

Examples:

- LTRIM('ORIE 3120','O') returns 'RIE 3120'
- LTRIM('ORIE 3120','RO') returns 'IE 3120'
- LTRIM('ORIE 3120','3120') returns 'ORIE 3120'

SQLite lets you manipulate strings

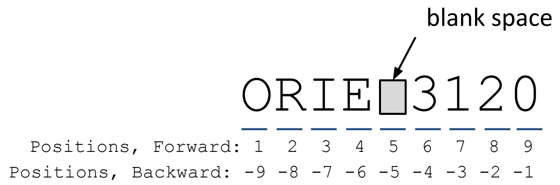
substr(*X*,*Y*,*Z*):

- returns a substring of X starting from character Y and returning Z characters.
- left-most character is Y=1
- substr(X,Y) returns all characters in the string starting from character Y
- If Y<0, the first character in the substring is found by counting from the end of the string
- If Z<0, abs(Z) characters <u>preceding</u> character Y are returned

Examples:

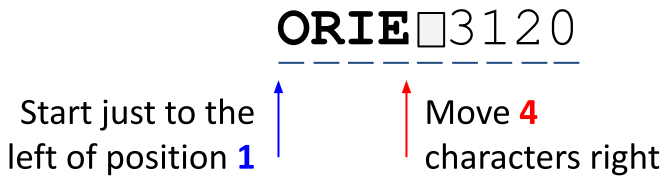
- SUBSTR('ORIE 3120',1,4) returns 'ORIE'
- SUBSTR('ORIE 3120',-1,-4) returns ' 312' (note the space)
- SUBSTR('ORIE 3120',5) returns ' 3120'
- SUBSTR('ORIE 3120',-1,4) returns '0'

SUBSTR Examples In Detail


blank space
ORIE□3120
Positions, Forward: 1 2 3 4 5 6 7 8 9
Positions, Backward: -9 -8 -7 -6 -5 -4 -3 -2 -1

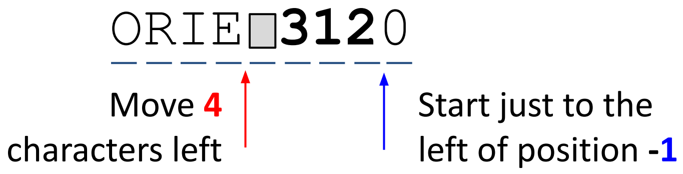
Example 1: positive X & Y

SUBSTR('ORIE 3120',1,4) → '**ORIE**'

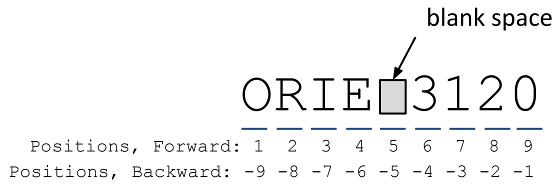

ORIE□3120
Start just to the left of position **1**
Move **4** characters right

Example 2: negative X & Y

SUBSTR('ORIE 3120',-1,-4) → ' **312**' (starts with a blank space)


ORIE□**312**0
Move **4** characters left
Start just to the left of position **-1**

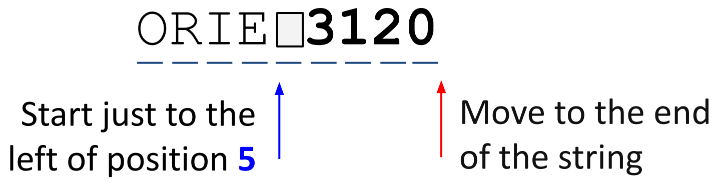
SUBSTR Examples In Detail



```
Positions, Forward:  1  2  3  4  5  6  7  8  9
Positions, Backward: -9 -8 -7 -6 -5 -4 -3 -2 -1
```

Example 3: positive X, no Y

SUBSTR('ORIE 3120',5) → ' **3120**'

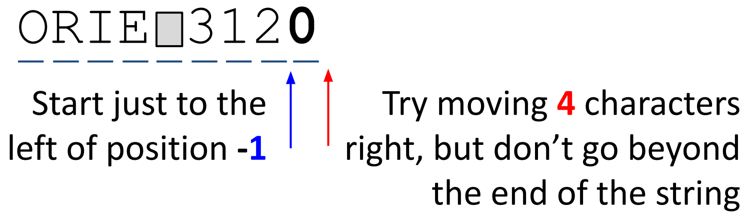


Start just to the left of position **5**

Move to the end of the string

Example 4: negative X, positive Y

SUBSTR('ORIE 3120',-1,4) → '**0**' (starts with a blank space)



Start just to the left of position **-1**

Try moving **4** characters right, but don't go beyond the end of the string

SQLite lets you manipulate strings

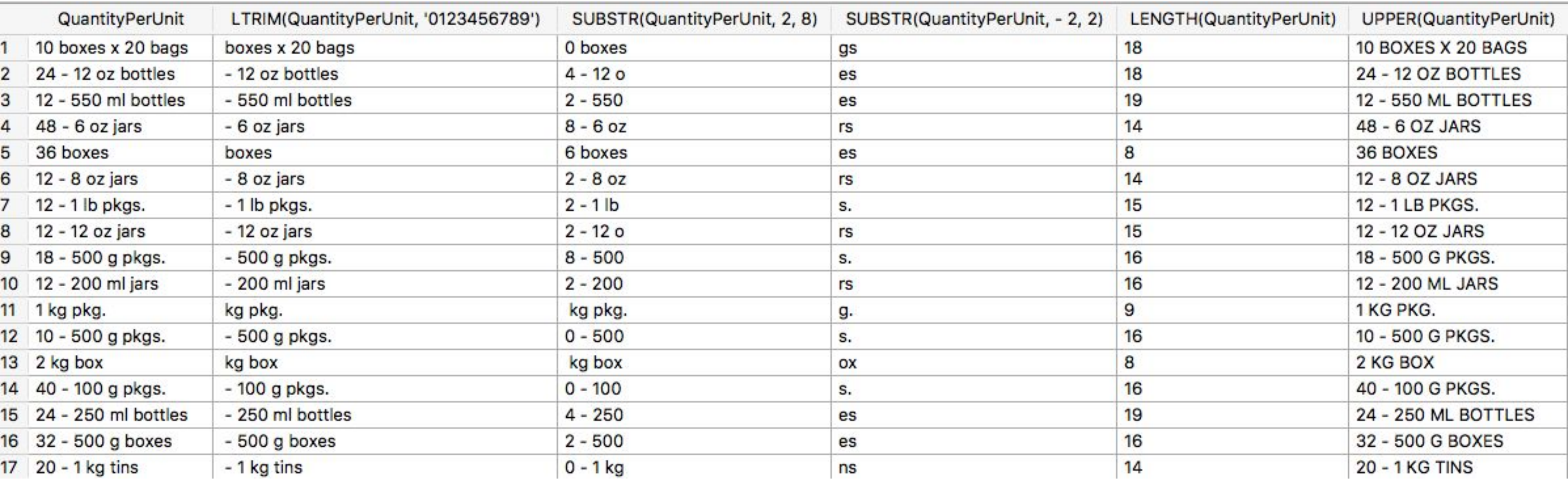
replace(*X*,*Y*,*Z*): substitutes string Z for every occurrence of string Y in string X

There is a bug in SQLite Studio 3.4.3 that prevents REPLACE from working. We won't use this command during homework or exams. If you really want to use it, it does work with the sqlite3 client distributed by Apache [link] and also worked with previous versions of SQLite Studio. It also looks like the SQLite Studio developer is working to fix it [see this issue on github].

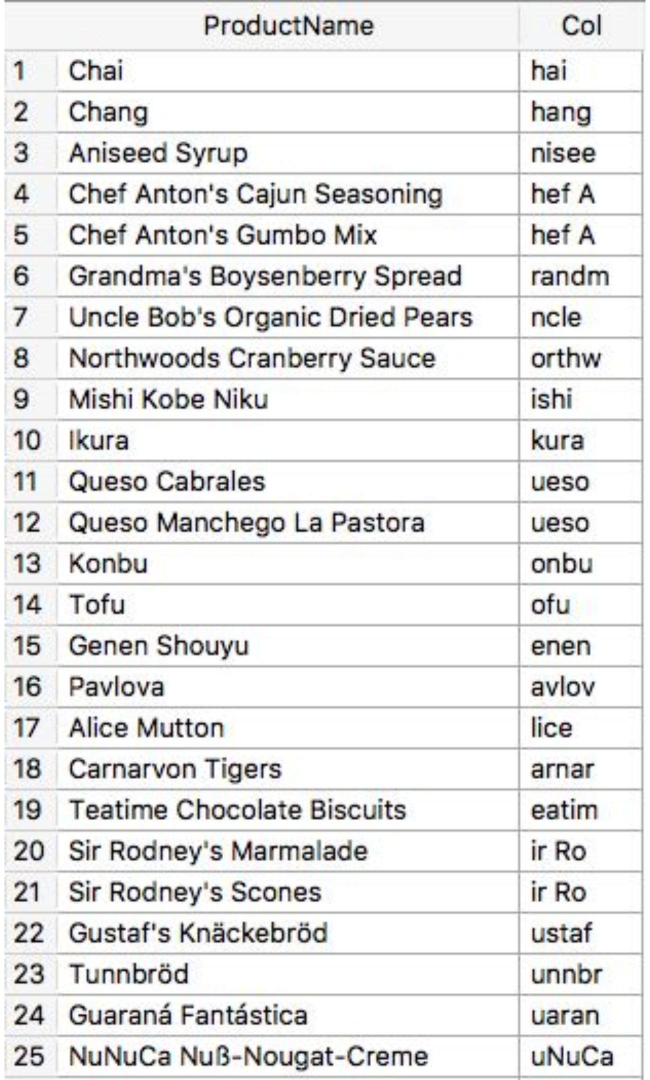
Examples:

- REPLACE('ORIE 3120','ORIE','ENGRC') returns 'ENGRC 3120'
- REPLACE('ORIE 3120','ORIE','') returns ' 3120'

```
SELECT QuantityPerUnit,
       LTRIM(QuantityPerUnit,'0123456789'),
       SUBSTR(QuantityPerUnit,2,8),
       SUBSTR(QuantityPerUnit,-2,2),
       LENGTH(QuantityPerUnit),
       UPPER(QuantityPerUnit)
FROM Products
```



	QuantityPerUnit	LTRIM(QuantityPerUnit, '0123456789')	SUBSTR(QuantityPerUnit, 2, 8)	SUBSTR(QuantityPerUnit, - 2, 2)	LENGTH(QuantityPerUnit)	UPPER(QuantityPerUnit)
1	10 boxes x 20 bags	boxes x 20 bags	0 boxes	gs	18	10 BOXES X 20 BAGS
2	24 - 12 oz bottles	- 12 oz bottles	4 - 12 o	es	18	24 - 12 OZ BOTTLES
3	12 - 550 ml bottles	- 550 ml bottles	2 - 550	es	19	12 - 550 ML BOTTLES
4	48 - 6 oz jars	- 6 oz jars	8 - 6 oz	rs	14	48 - 6 OZ JARS
5	36 boxes	boxes	6 boxes	es	8	36 BOXES
6	12 - 8 oz jars	- 8 oz jars	2 - 8 oz	rs	14	12 - 8 OZ JARS
7	12 - 1 lb pkgs.	- 1 lb pkgs.	2 - 1 lb	s.	15	12 - 1 LB PKGS.
8	12 - 12 oz jars	- 12 oz jars	2 - 12 o	rs	15	12 - 12 OZ JARS
9	18 - 500 g pkgs.	- 500 g pkgs.	8 - 500	s.	16	18 - 500 G PKGS.
10	12 - 200 ml jars	- 200 ml jars	2 - 200	rs	16	12 - 200 ML JARS
11	1 kg pkg.	kg pkg.	kg pkg.	g.	9	1 KG PKG.
12	10 - 500 g pkgs.	- 500 g pkgs.	0 - 500	s.	16	10 - 500 G PKGS.
13	2 kg box	kg box	kg box	ox	8	2 KG BOX
14	40 - 100 g pkgs.	- 100 g pkgs.	0 - 100	s.	16	40 - 100 G PKGS.
15	24 - 250 ml bottles	- 250 ml bottles	4 - 250	es	19	24 - 250 ML BOTTLES
16	32 - 500 g boxes	- 500 g boxes	2 - 500	es	16	32 - 500 G BOXES
17	20 - 1 kg tins	- 1 kg tins	0 - 1 kg	ns	14	20 - 1 KG TINS



	ProductName	Col
1	Chai	hai
2	Chang	hang
3	Aniseed Syrup	nisee
4	Chef Anton's Cajun Seasoning	hef A
5	Chef Anton's Gumbo Mix	hef A
6	Grandma's Boysenberry Spread	randm
7	Uncle Bob's Organic Dried Pears	ncle
8	Northwoods Cranberry Sauce	orthw
9	Mishi Kobe Niku	ishi
10	Ikura	kura
11	Queso Cabrales	ueso
12	Queso Manchego La Pastora	ueso
13	Konbu	onbu
14	Tofu	ofu
15	Genen Shouyu	enen
16	Pavlova	avlov
17	Alice Mutton	lice
18	Carnarvon Tigers	arnar
19	Teatime Chocolate Biscuits	eatim
20	Sir Rodney's Marmalade	ir Ro
21	Sir Rodney's Scones	ir Ro
22	Gustaf's Knäckebröd	ustaf
23	Tunnbröd	unnbr
24	Guaraná Fantástica	uaran
25	NuNuCa Nuß-Nougat-Creme	uNuCa

Lec 3, Q3: What command generated the “Col” column?

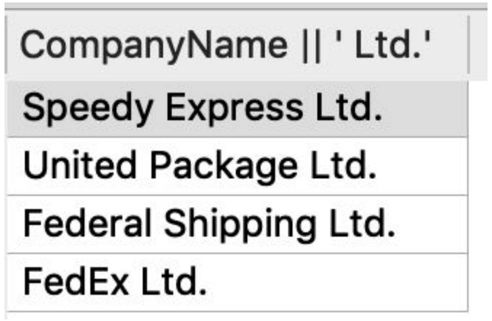
(a) SUBSTR(ProductName,5,2)
(b) SUBSTR(ProductName,1,5)
(c) LTRIM(ProductName,’abc’)
(d) SUBSTR(ProductName,5,-2)
(e) SUBSTR(ProductName,2,5)

Lec 3, Q4: What command(s) could have generated the string 'ab'?

(a) SUBSTR('abcd',1,2)
(b) REPLACE('abcd','cd','')
(c) RTRIM('abcd','dc')
(d) a and b
(d) a and c
(e) b and c
(f) a, b and c

Concatenation

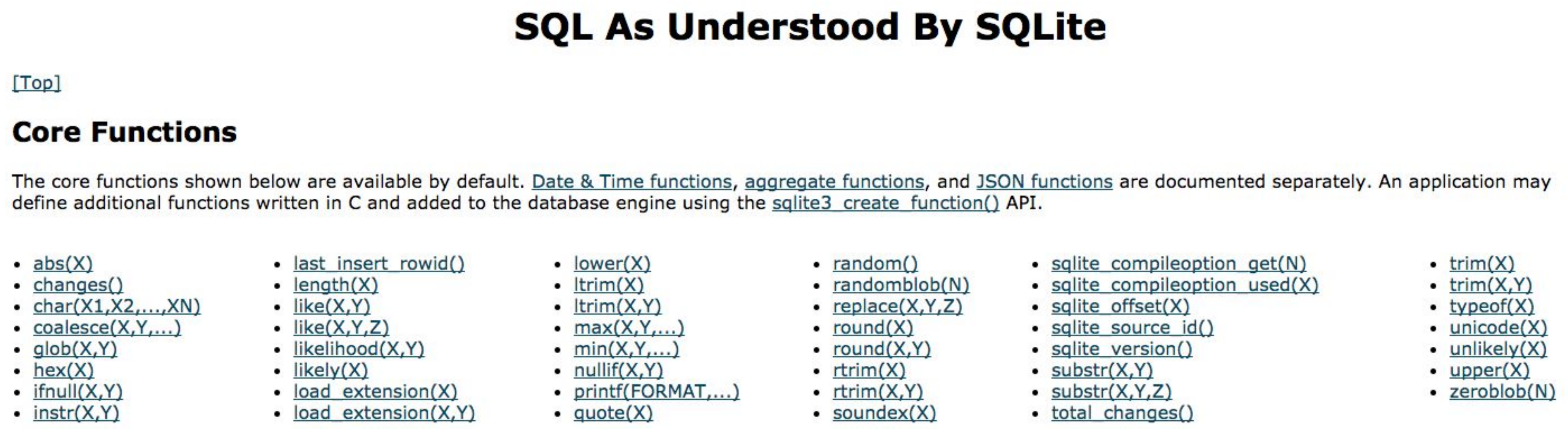
- The double-pipe operator || concatenates two strings.
- It does the same thing as the CONCAT function.
- Example: SELECT CompanyName || ' Ltd.' FROM Shippers



CompanyName || ' Ltd.'
Speedy Express Ltd.
United Package Ltd.
Federal Shipping Ltd.
FedEx Ltd.

- This query returns the same records as:
  SELECT CONCAT(CompanyName, ' Ltd.') FROM Shippers

Descriptions of these string commands are available in the SQLite documentation



SQL As Understood By SQLite

[Top]

Core Functions

The core functions shown below are available by default. Date & Time functions, aggregate functions, and JSON functions are documented separately. An application may define additional functions written in C and added to the database engine using the sqlite3_create_function() API.

- 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)

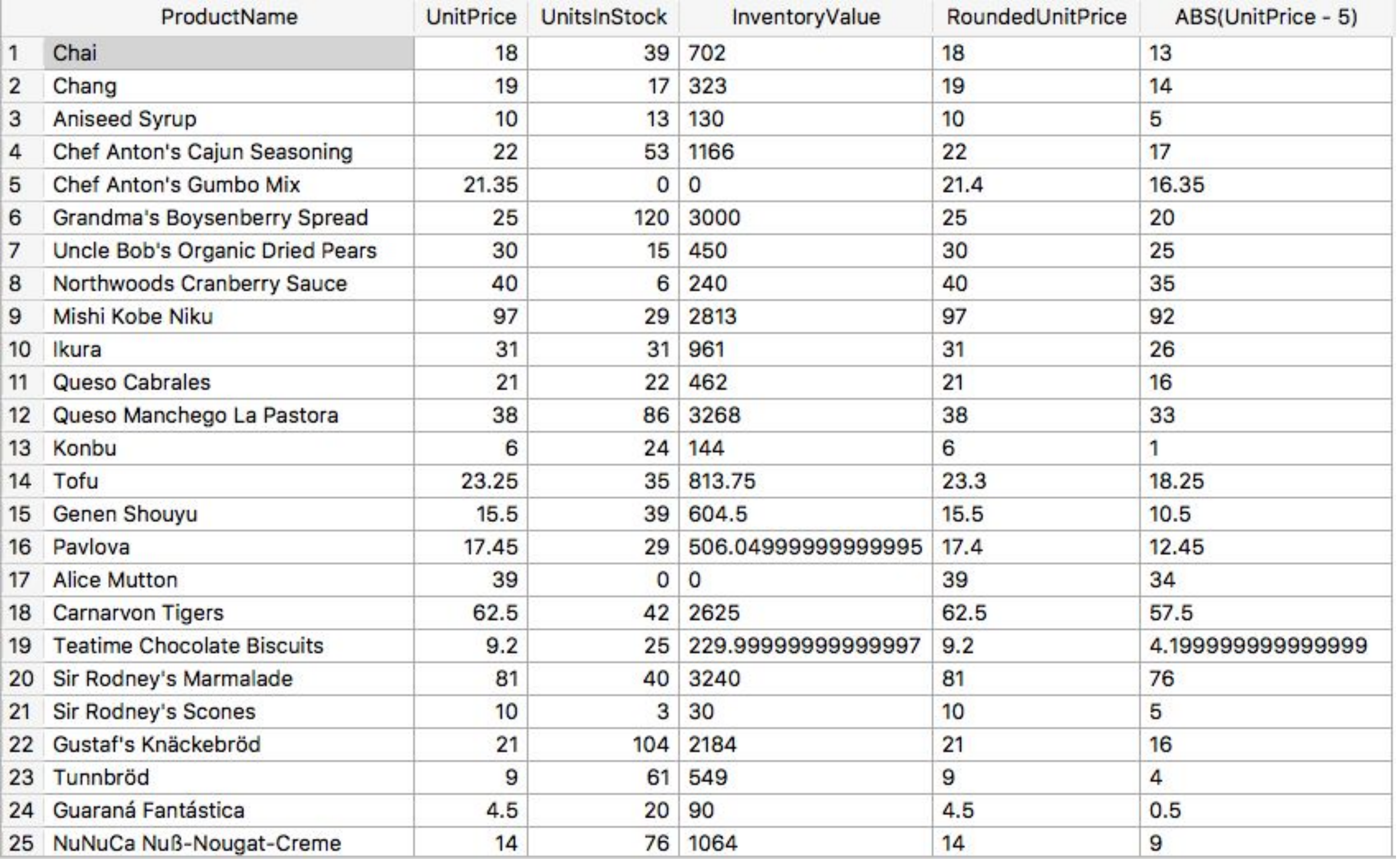
https://www.sqlite.org/lang_corefunc.html

The reading on the website, "SQLiteFunctionsfor3120.pdf", has simplified documentation for the functions we are covering in the course.

AS keyword
CASE statements

You can rename your fields using AS

```
SELECT ProductName,
       UnitPrice,
       UnitsInStock,
       UnitPrice*UnitsInStock AS InventoryValue,
       ROUND(UnitPrice,1) AS RoundedUnitPrice,
       ABS(UnitPrice-5)
FROM Products
```



	ProductName	UnitPrice	UnitsInStock	InventoryValue	RoundedUnitPrice	ABS(UnitPrice - 5)
1	Chai	18	39	702	18	13
2	Chang	19	17	323	19	14
3	Aniseed Syrup	10	13	130	10	5
4	Chef Anton's Cajun Seasoning	22	53	1166	22	17
5	Chef Anton's Gumbo Mix	21.35	0	0	21.4	16.35
6	Grandma's Boysenberry Spread	25	120	3000	25	20
7	Uncle Bob's Organic Dried Pears	30	15	450	30	25
8	Northwoods Cranberry Sauce	40	6	240	40	35
9	Mishi Kobe Niku	97	29	2813	97	92
10	Ikura	31	31	961	31	26
11	Queso Cabrales	21	22	462	21	16
12	Queso Manchego La Pastora	38	86	3268	38	33
13	Konbu	6	24	144	6	1
14	Tofu	23.25	35	813.75	23.3	18.25
15	Genen Shouyu	15.5	39	604.5	15.5	10.5
16	Pavlova	17.45	29	506.04999999999995	17.4	12.45
17	Alice Mutton	39	0	0	39	34
18	Carnarvon Tigers	62.5	42	2625	62.5	57.5
19	Teatime Chocolate Biscuits	9.2	25	229.99999999999997	9.2	4.199999999999999
20	Sir Rodney's Marmalade	81	40	3240	81	76
21	Sir Rodney's Scones	10	3	30	10	5
22	Gustaf's Knäckebröd	21	104	2184	21	16
23	Tunnbröd	9	61	549	9	4
24	Guaraná Fantástica	4.5	20	90	4.5	0.5
25	NuNuCa Nuß-Nougat-Creme	14	76	1064	14	9

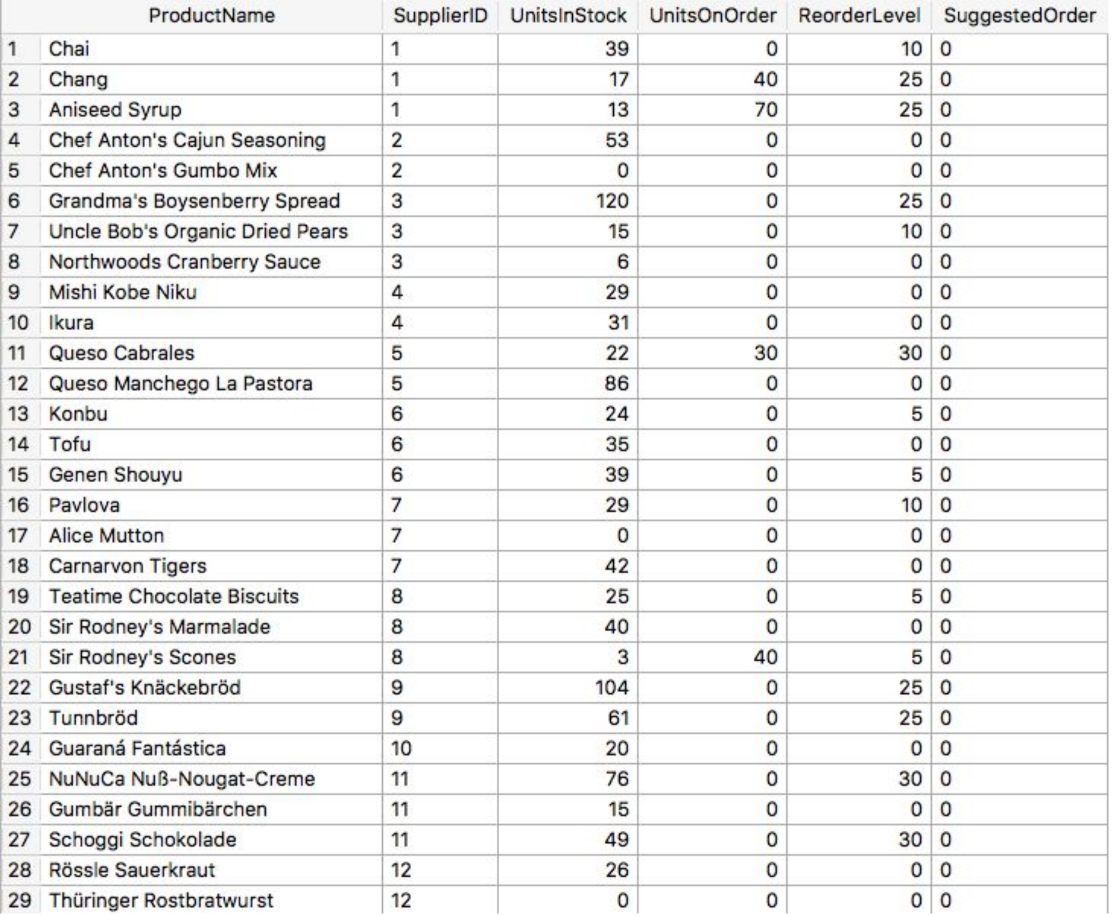
You **can’t** refer to a renamed field within another field, only in the things that come after FROM

```
SELECT ProductName,
        UnitPrice,
        UnitsInStock,
        UnitPrice*UnitsInStock AS InventoryValue,
        InventoryValue*0.88 As InventoryValueInEuros
FROM Products
```

(This won’t work)

You can use CASE statements

```
SELECT ProductName, SupplierID,
UnitsInStock,UnitsOnOrder,ReorderLevel,
CASE WHEN ReorderLevel>UnitsInStock+UnitsOnOrder
      THEN ReorderLevel-UnitsInStock-UnitsOnOrder
      ELSE 0
END AS SuggestedOrder
FROM Products
```



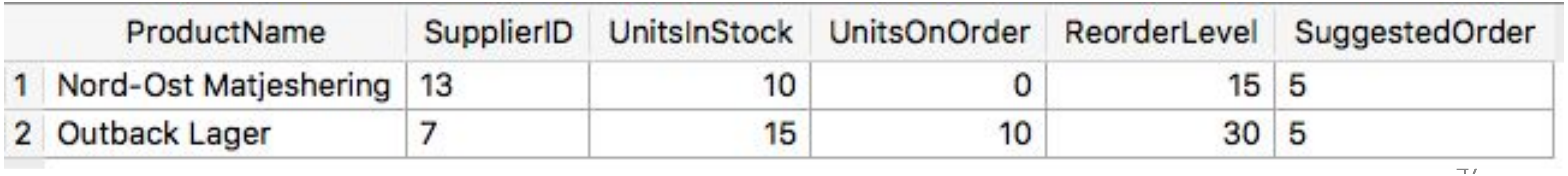
	ProductName	SupplierID	UnitsInStock	UnitsOnOrder	ReorderLevel	SuggestedOrder
1	Chai	1	39	0	10	0
2	Chang	1	17	40	25	0
3	Aniseed Syrup	1	13	70	25	0
4	Chef Anton's Cajun Seasoning	2	53	0	0	0
5	Chef Anton's Gumbo Mix	2	0	0	0	0
6	Grandma's Boysenberry Spread	3	120	0	25	0
7	Uncle Bob's Organic Dried Pears	3	15	0	10	0
8	Northwoods Cranberry Sauce	3	6	0	0	0
9	Mishi Kobe Niku	4	29	0	0	0
10	Ikura	4	31	0	0	0
11	Queso Cabrales	5	22	30	30	0
12	Queso Manchego La Pastora	5	86	0	0	0
13	Konbu	6	24	0	5	0
14	Tofu	6	35	0	0	0
15	Genen Shouyu	6	39	0	5	0
16	Pavlova	7	29	0	10	0
17	Alice Mutton	7	0	0	0	0
18	Carnarvon Tigers	7	42	0	0	0
19	Teatime Chocolate Biscuits	8	25	0	5	0
20	Sir Rodney's Marmalade	8	40	0	0	0
21	Sir Rodney's Scones	8	3	40	5	0
22	Gustaf's Knäckebröd	9	104	0	25	0
23	Tunnbröd	9	61	0	25	0
24	Guaraná Fantástica	10	20	0	0	0
25	NuNuCa Nuß-Nougat-Creme	11	76	0	30	0
26	Gumbär Gummibärchen	11	15	0	0	0
27	Schoggi Schokolade	11	49	0	30	0
28	Rössle Sauerkraut	12	26	0	0	0
29	Thüringer Rostbratwurst	12	0	0	0	0

```
SELECT ProductName, SupplierID,
UnitsInStock,UnitsOnOrder,ReorderLevel,
CASE WHEN ReorderLevel>UnitsInStock+UnitsOnOrder
     THEN ReorderLevel-UnitsInStock-UnitsOnOrder
     ELSE 0
END AS SuggestedOrder
FROM Products
```

You can refer to renamed fields in WHERE clauses

```
SELECT ProductName, SupplierID,
        UnitsInStock,UnitsOnOrder,ReorderLevel,
        CASE WHEN ReorderLevel>UnitsInStock+UnitsOnOrder
            THEN ReorderLevel-UnitsInStock-UnitsOnOrder
            ELSE 0
        END AS SuggestedOrder
FROM Products
WHERE SuggestedOrder > 0
```

```
SELECT ProductName, SupplierID,
        UnitsInStock,UnitsOnOrder,ReorderLevel,
        CASE WHEN ReorderLevel>UnitsInStock+UnitsOnOrder
            THEN ReorderLevel-UnitsInStock-UnitsOnOrder
            ELSE 0
        END AS SuggestedOrder
FROM Products
WHERE SuggestedOrder > 0
```



	ProductName	SupplierID	UnitsInStock	UnitsOnOrder	ReorderLevel	SuggestedOrder
1	Nord-Ost Matjeshering	13	10	0	15	5
2	Outback Lager	7	15	10	30	5

NULL

Null

- A null represents a missing value in a record in a specific field
- It is not zero
- It is not a space
- It is nothing
- A field with a null value has been left blank during record creation
- Sometimes this is fine, sometimes it is a problem

Keep in mind for WHERE/ON/CASE statements: NULL has tricky behavior in comparisons

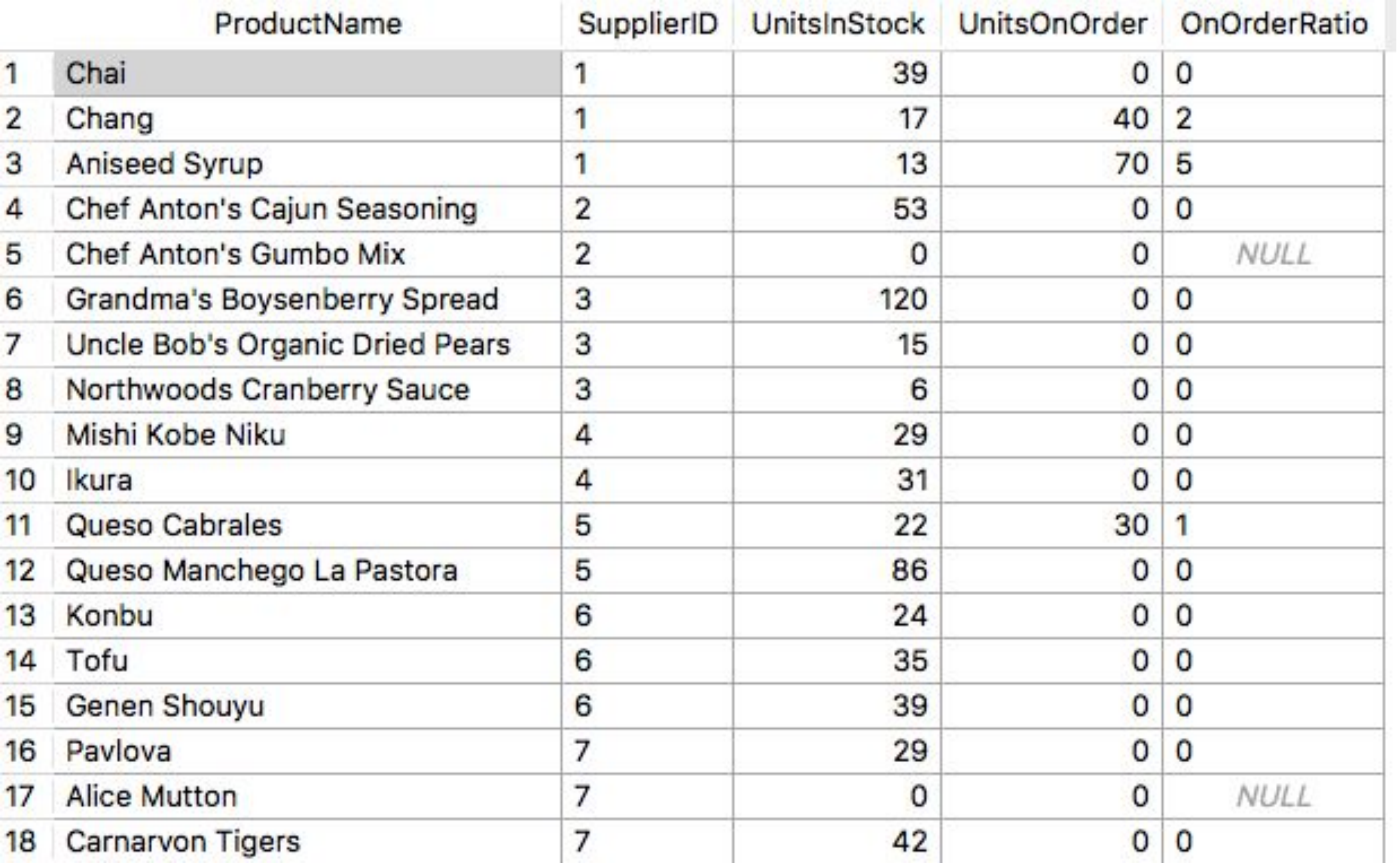
- Think of NULL as “Unknown”
- NULL = NULL is false
- NULL <> NULL is false (!= is the same as <>)
- To check whether something is NULL or not, use IS NULL and IS NOT NULL

You can look up NULL values

```
SELECT *
FROM Orders
WHERE ShippedDate IS NULL
```

You can return NULL as a value

```
SELECT ProductName, SupplierID,
        UnitsInStock,UnitsOnOrder,
        CASE WHEN UnitsInStock>0
            THEN UnitsOnOrder / UnitsInStock
            ELSE NULL
        END AS OnOrderRatio
FROM Products
```

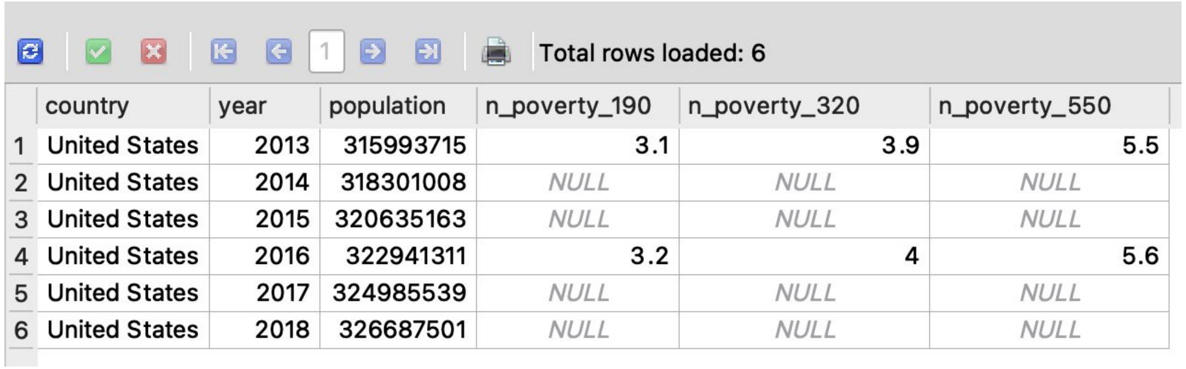


	ProductName	SupplierID	UnitsInStock	UnitsOnOrder	OnOrderRatio
1	Chai	1	39	0	0
2	Chang	1	17	40	2
3	Aniseed Syrup	1	13	70	5
4	Chef Anton's Cajun Seasoning	2	53	0	0
5	Chef Anton's Gumbo Mix	2	0	0	NULL
6	Grandma's Boysenberry Spread	3	120	0	0
7	Uncle Bob's Organic Dried Pears	3	15	0	0
8	Northwoods Cranberry Sauce	3	6	0	0
9	Mishi Kobe Niku	4	29	0	0
10	Ikura	4	31	0	0
11	Queso Cabrales	5	22	30	1
12	Queso Manchego La Pastora	5	86	0	0
13	Konbu	6	24	0	0
14	Tofu	6	35	0	0
15	Genen Shouyu	6	39	0	0
16	Pavlova	7	29	0	0
17	Alice Mutton	7	0	0	NULL
18	Carnarvon Tigers	7	42	0	0

Lec 3, Q5

We ran this query against the Poverty table from the HW:

```
SELECT * FROM poverty
WHERE country = 'United States' AND year > 2012
```



Total rows loaded: 6

	country	year	population	n_poverty_190	n_poverty_320	n_poverty_550
1	United States	2013	315993715	3.1	3.9	5.5
2	United States	2014	318301008	NULL	NULL	NULL
3	United States	2015	320635163	NULL	NULL	NULL
4	United States	2016	322941311	3.2	4	5.6
5	United States	2017	324985539	NULL	NULL	NULL
6	United States	2018	326687501	NULL	NULL	NULL

(a) 6
(b) 4
(c) 2
(d) 0

How many records will be returned by this query?

```
SELECT * FROM poverty
WHERE country = 'United States' AND year > 2012
AND n_poverty_550 = NULL
```

Logical Operators

NOT

Think of NULL as “Unknown”

If X is…	Then NOT X is…
TRUE	FALSE
FALSE	TRUE
NULL	**NULL**

AND

Think of NULL as “Unknown”

Cells with a white background show X AND Y		Y		
		TRUE	FALSE	NULL
X	TRUE	TRUE	FALSE	**NULL**
	FALSE	FALSE	FALSE	**FALSE**
	NULL	**NULL**	**FALSE**	**NULL**

OR

Think of NULL as "Unknown"

Cells with a white background show X OR Y		Y		
		TRUE	FALSE	NULL
X	TRUE	TRUE	TRUE	**TRUE**
	FALSE	TRUE	FALSE	**NULL**
	NULL	**TRUE**	**NULL**	**NULL**

Course Logistics, Jan 30

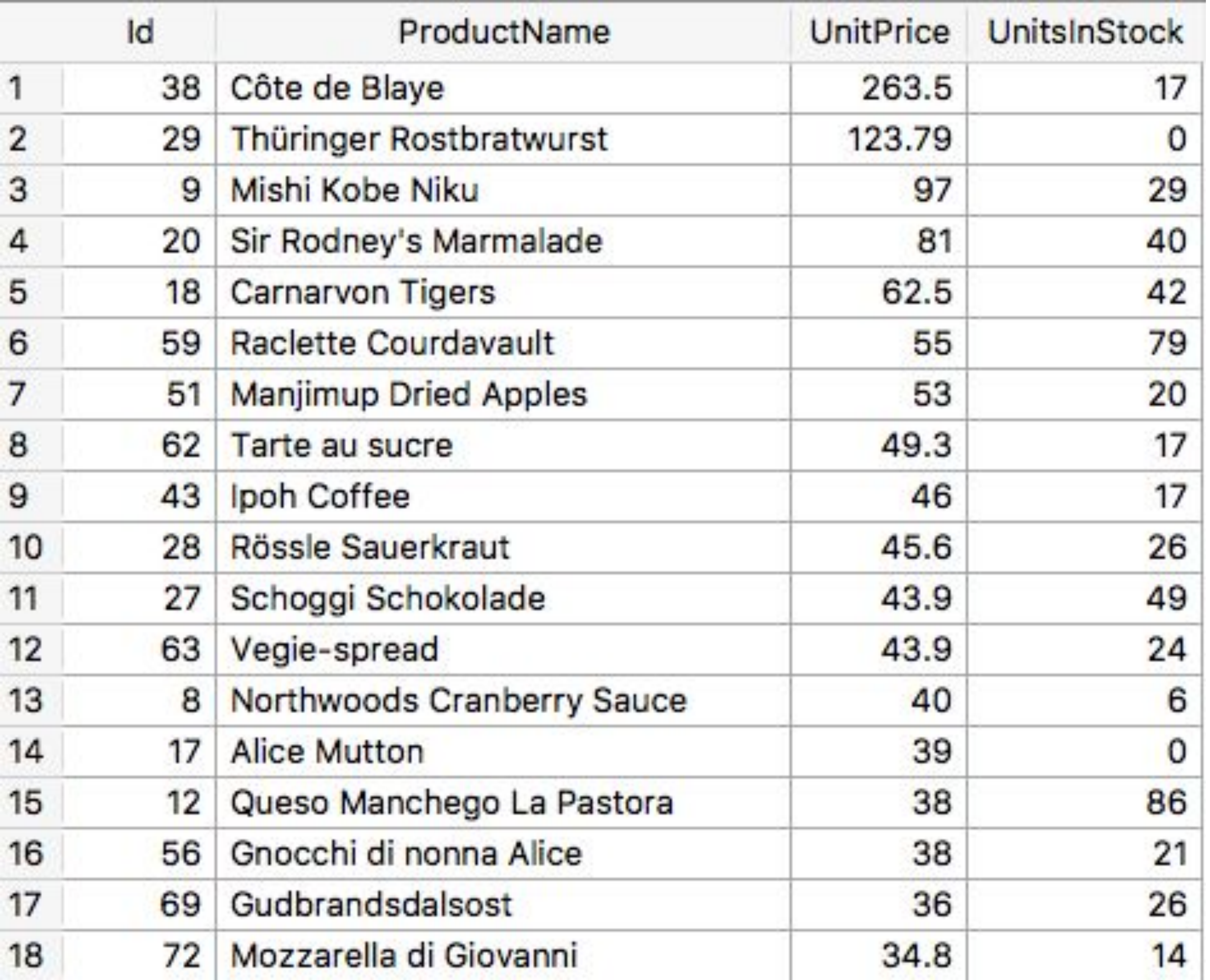
- Recitation 1 due on Gradescope by the end of the last recitation on Friday (5:59pm)
- HW1 due Mon Feb 5 at 11:59pm
- Office hours posted on the course google calendar
- If you are on Canvas, you should also be on Gradescope. If not, the entry code is JKW75X
- CATME link is still being created — we will extend the deadline to enter your schedule into CATME to this Sunday (Feb 4 @ 11:59pm). When the link is available, there will be a Canvas announcement.

ORDER BY

You can order your results

```
SELECT ProductID, ProductName, UnitPrice, UnitsInStock
FROM Products
ORDER BY UnitPrice DESC
```

- Here we look at all fields and records
- But, they are now sorted
- DESC sorts in descending order, ASC sorts in ascending order
- The default is ascending order

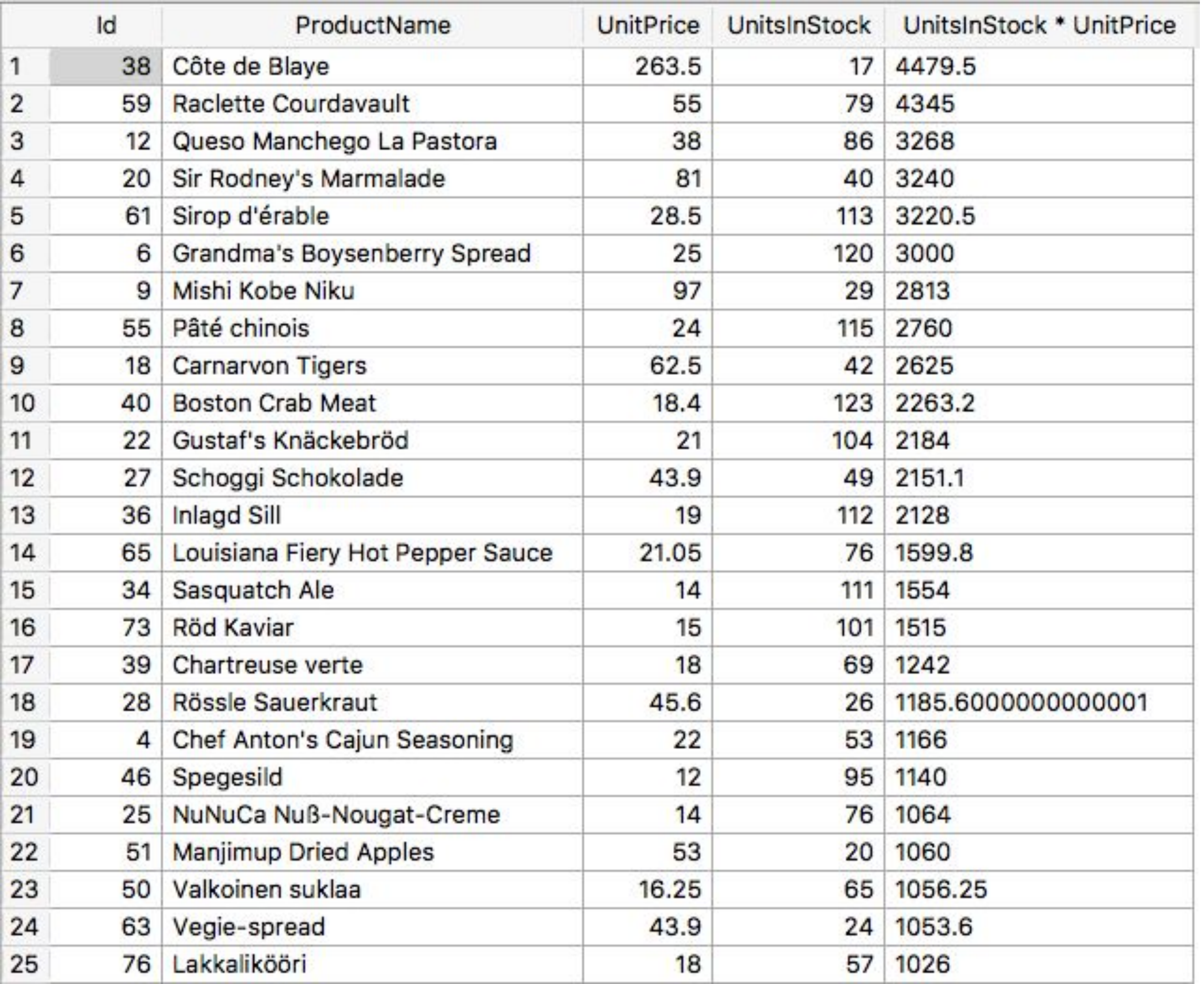


	Id	ProductName	UnitPrice	UnitsInStock
1	38	Côte de Blaye	263.5	17
2	29	Thüringer Rostbratwurst	123.79	0
3	9	Mishi Kobe Niku	97	29
4	20	Sir Rodney's Marmalade	81	40
5	18	Carnarvon Tigers	62.5	42
6	59	Raclette Courdavault	55	79
7	51	Manjimup Dried Apples	53	20
8	62	Tarte au sucre	49.3	17
9	43	Ipoh Coffee	46	17
10	28	Rössle Sauerkraut	45.6	26
11	27	Schoggi Schokolade	43.9	49
12	63	Vegie-spread	43.9	24
13	8	Northwoods Cranberry Sauce	40	6
14	17	Alice Mutton	39	0
15	12	Queso Manchego La Pastora	38	86
16	56	Gnocchi di nonna Alice	38	21
17	69	Gudbrandsdalsost	36	26
18	72	Mozzarella di Giovanni	34.8	14

```
SELECT ProductID,
       ProductName,
       UnitPrice,
       UnitsInStock
FROM Products
ORDER BY UnitPrice DESC
```

You can order by calculated columns

```
SELECT ProductID, ProductName, UnitPrice, UnitsInStock,
        UnitsInStock*UnitPrice
FROM Products
ORDER BY UnitsInStock*UnitPrice DESC
```



	Id	ProductName	UnitPrice	UnitsInStock	UnitsInStock * UnitPrice
1	38	Côte de Blaye	263.5	17	4479.5
2	59	Raclette Courdavault	55	79	4345
3	12	Queso Manchego La Pastora	38	86	3268
4	20	Sir Rodney's Marmalade	81	40	3240
5	61	Sirop d'érable	28.5	113	3220.5
6	6	Grandma's Boysenberry Spread	25	120	3000
7	9	Mishi Kobe Niku	97	29	2813
8	55	Pâté chinois	24	115	2760
9	18	Carnarvon Tigers	62.5	42	2625
10	40	Boston Crab Meat	18.4	123	2263.2
11	22	Gustaf's Knäckebröd	21	104	2184
12	27	Schoggi Schokolade	43.9	49	2151.1
13	36	Inlagd Sill	19	112	2128
14	65	Louisiana Fiery Hot Pepper Sauce	21.05	76	1599.8
15	34	Sasquatch Ale	14	111	1554
16	73	Röd Kaviar	15	101	1515
17	39	Chartreuse verte	18	69	1242
18	28	Rössle Sauerkraut	45.6	26	1185.6000000000001
19	4	Chef Anton's Cajun Seasoning	22	53	1166
20	46	Spegesild	12	95	1140
21	25	NuNuCa Nuß-Nougat-Creme	14	76	1064
22	51	Manjimup Dried Apples	53	20	1060
23	50	Valkoinen suklaa	16.25	65	1056.25
24	63	Vegie-spread	43.9	24	1053.6
25	76	Lakkalikööri	18	57	1026

You can refer to columns by their column number

(for when you don’t want to type out the full calculation again)

```
SELECT ProductID, ProductName, UnitPrice, UnitsInStock,
        UnitsInStock*UnitPrice
FROM Products
ORDER BY 5 DESC
```

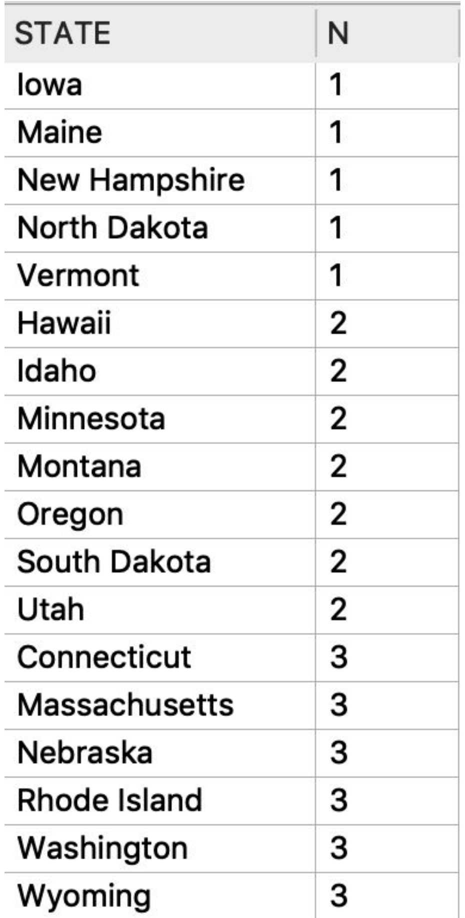
- UnitsInStock*UnitPrice is the 5th column

You can also give the column a name with AS and refer to that

```
SELECT ProductID, ProductName, UnitPrice, UnitsInStock,
        UnitsInStock*UnitPrice AS InventoryValue
FROM Products
ORDER BY InventoryValue DESC
```

- The results will be the same as before.

You can sort by 2 or more columns



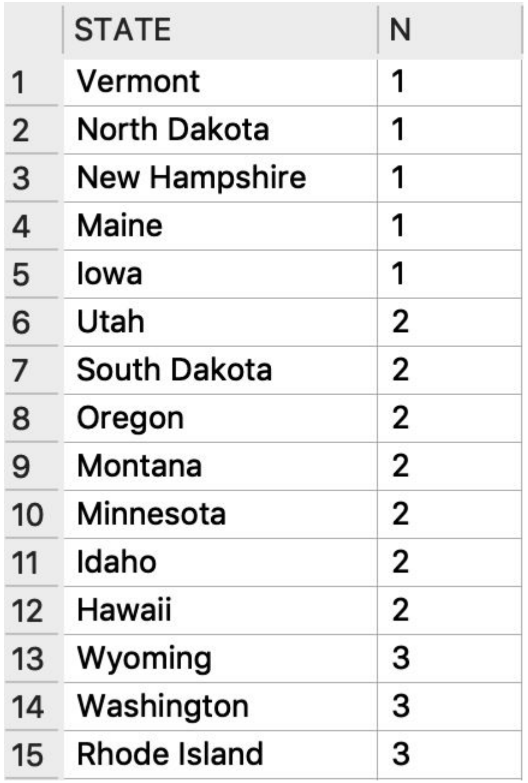
STATE	N
Iowa	1
Maine	1
New Hampshire	1
North Dakota	1
Vermont	1
Hawaii	2
Idaho	2
Minnesota	2
Montana	2
Oregon	2
South Dakota	2
Utah	2
Connecticut	3
Massachusetts	3
Nebraska	3
Rhode Island	3
Washington	3
Wyoming	3

```
SELECT STATE, ROUND(Murder,0) AS N
FROM CrimeRatesByState2005
ORDER BY 2,1
```

Sorts by this column first (ascending order is the default)

Then it sorts by this column (again, in ascending order)

You can specify different sort orders for the columns



	STATE	N
1	Vermont	1
2	North Dakota	1
3	New Hampshire	1
4	Maine	1
5	Iowa	1
6	Utah	2
7	South Dakota	2
8	Oregon	2
9	Montana	2
10	Minnesota	2
11	Idaho	2
12	Hawaii	2
13	Wyoming	3
14	Washington	3
15	Rhode Island	3

```
SELECT STATE, ROUND(Murder,0) AS N
FROM CrimeRatesByState2005
ORDER BY 2 ASC,1 DESC
```

If you need the results in a certain order, you must specify ORDER BY

SQLite is free to return the results in any order it likes, as long as it matches the ORDER BY you asked for.

If you need the results in a certain order, make sure to specify this in the ORDER BY.

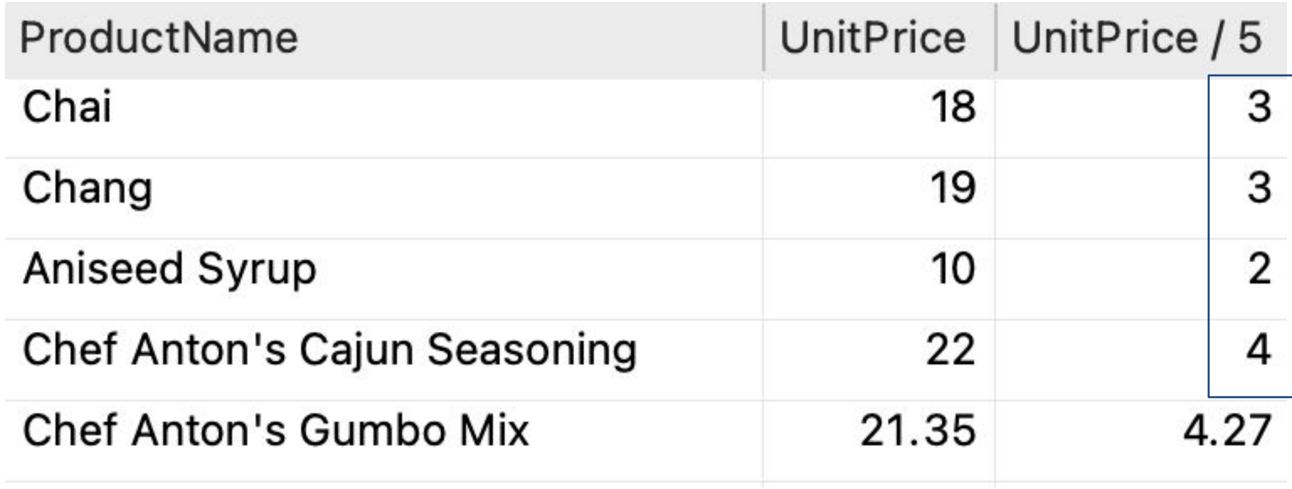
Otherwise the order can change unpredictably, e.g., if SQLite decides that returning in a different order would be more efficient.

Division

Be careful when dividing integers

- If SQLite thinks that you are dividing two integers, its answer will be an integer.
- This will round the answer.
- This is often not what you want.

```
SELECT ProductName, UnitPrice, UnitPrice / 5 FROM Products
```



ProductName	UnitPrice	UnitPrice / 5
Chai	18	3
Chang	19	3
Aniseed Syrup	10	2
Chef Anton's Cajun Seasoning	22	4
Chef Anton's Gumbo Mix	21.35	4.27

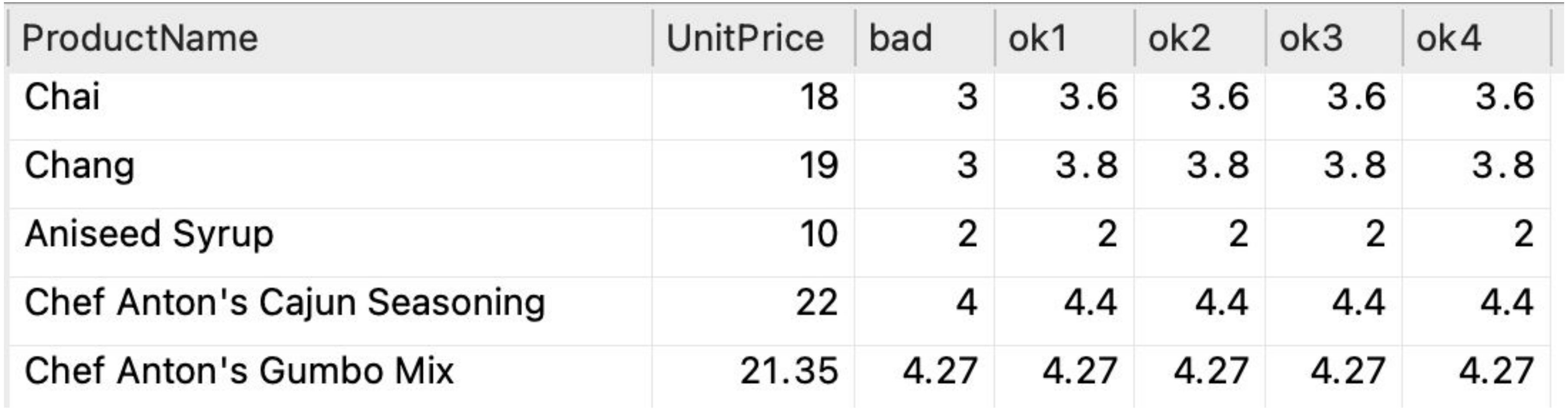
Rounded

Not Rounded

Be careful when dividing integers

To avoid this, we need to tell SQLite that either the numerator or the denominator is a real number

```
SELECT ProductName, UnitPrice, UnitPrice / 5 AS bad,
UnitPrice / 5.0 AS ok1,
CAST(UnitPrice AS REAL)/5 AS ok2,
UnitPrice/CAST(5 AS REAL) AS ok3,
CAST(UnitPrice AS REAL)/ CAST(5 AS REAL) AS ok4
FROM Products
```

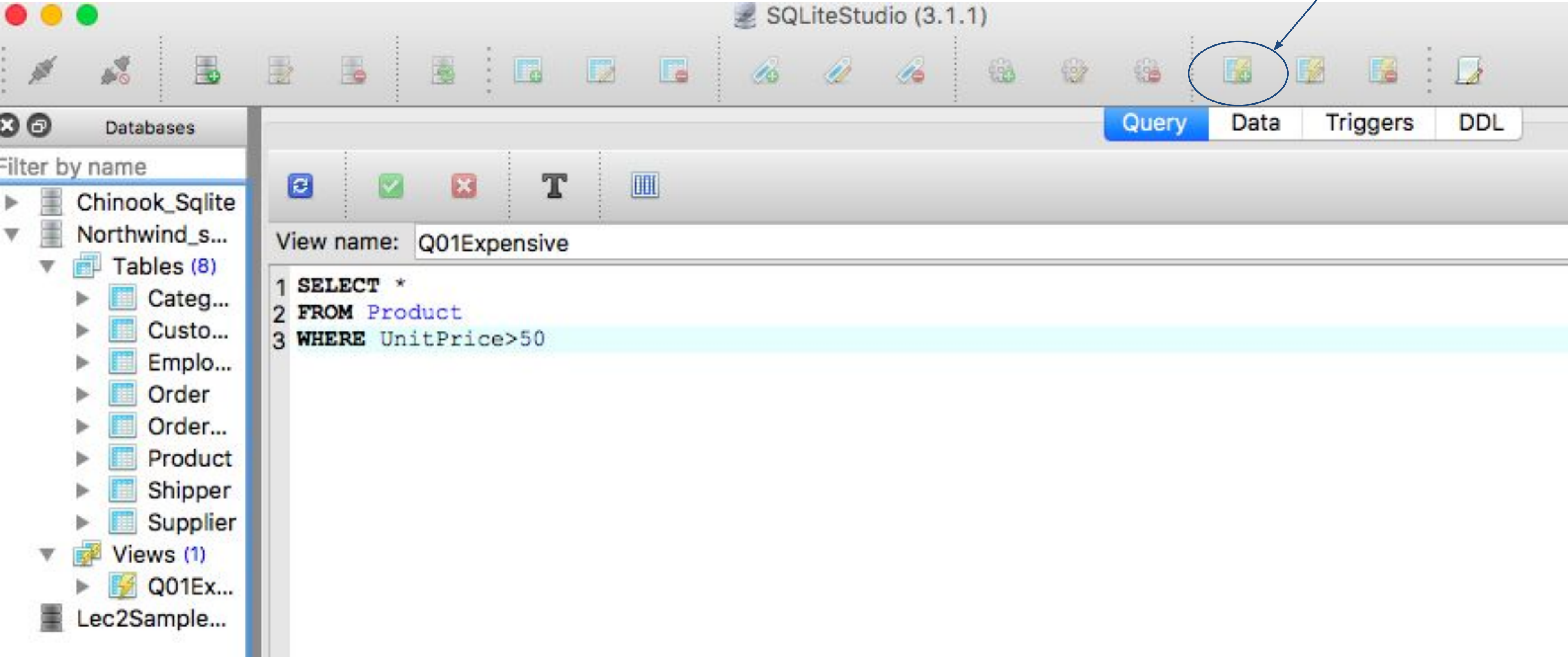


ProductName	UnitPrice	bad	ok1	ok2	ok3	ok4
Chai	18	3	3.6	3.6	3.6	3.6
Chang	19	3	3.8	3.8	3.8	3.8
Aniseed Syrup	10	2	2	2	2	2
Chef Anton's Cajun Seasoning	22	4	4.4	4.4	4.4	4.4
Chef Anton's Gumbo Mix	21.35	4.27	4.27	4.27	4.27	4.27

Views

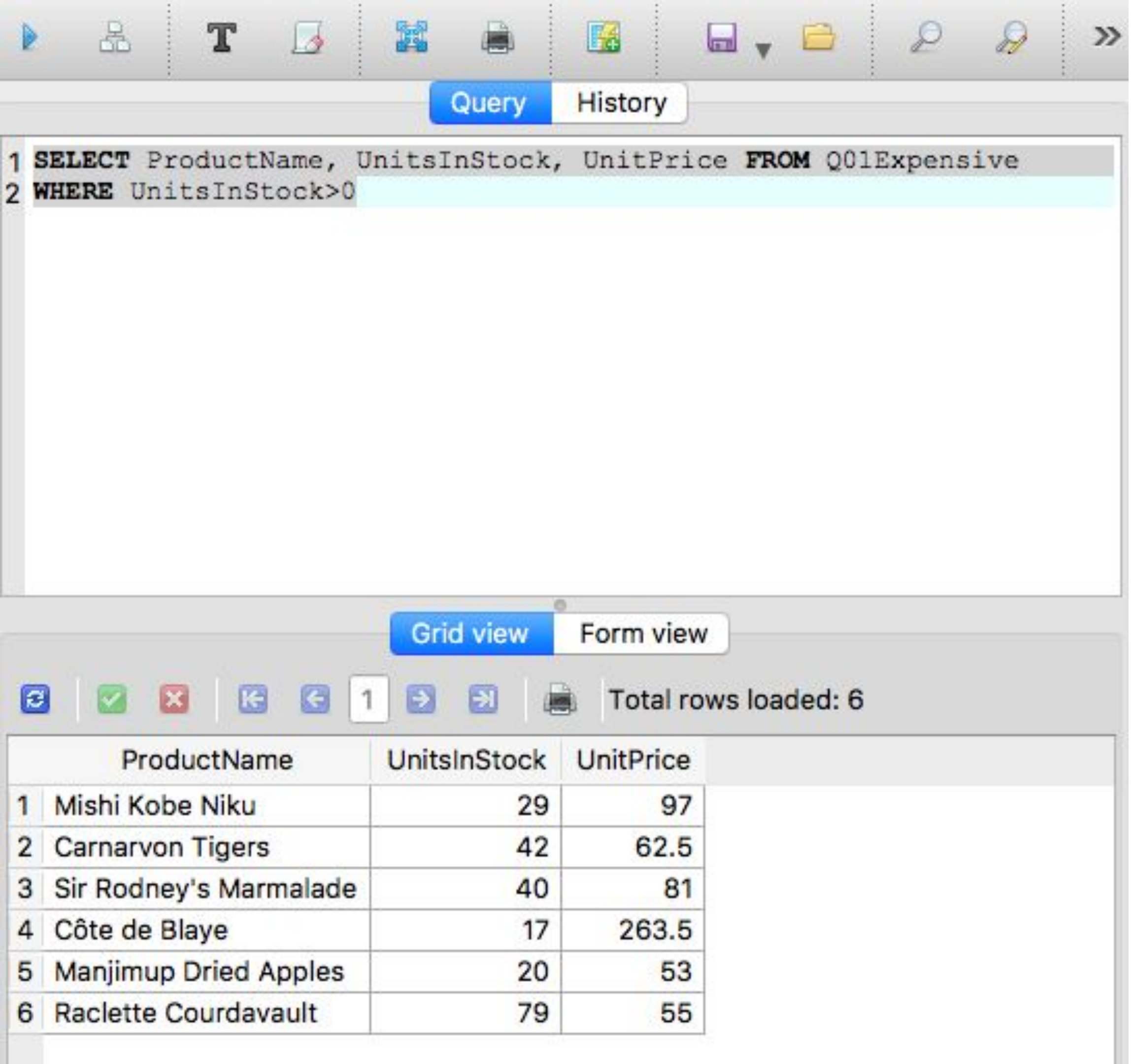
Views are saved queries

Create them by clicking this button



View name: Q01Expensive

```
SELECT *
FROM Product
WHERE UnitPrice>50
```



```
SELECT ProductName, UnitsInStock, UnitPrice FROM Q01Expensive
WHERE UnitsInStock>0
```

Total rows loaded: 6

	ProductName	UnitsInStock	UnitPrice
1	Mishi Kobe Niku	29	97
2	Carnarvon Tigers	42	62.5
3	Sir Rodney's Marmalade	40	81
4	Côte de Blaye	17	263.5
5	Manjimup Dried Apples	20	53
6	Raclette Courdavault	79	55

You can refer to them in other queries or views

Creating, Altering, and Deleting Tables & Views

SQL commands for creating, altering, and deleting table and view schema

- Data Definition Language (DDL) is the part of SQL that enables a database user to create and restructure database objects, such as the creation or deletion of a table
- Commands:
  – CREATE [creates a new table or view]
  – ALTER [alters an existing table or view]
  – DROP [gets rid of a table or view]
- In this class we'll use SQLiteStudio's GUI instead of these commands, so their syntax won't be on HW or exams.

To give you a flavor for how they work, here is a table and the corresponding CREATE TABLE command

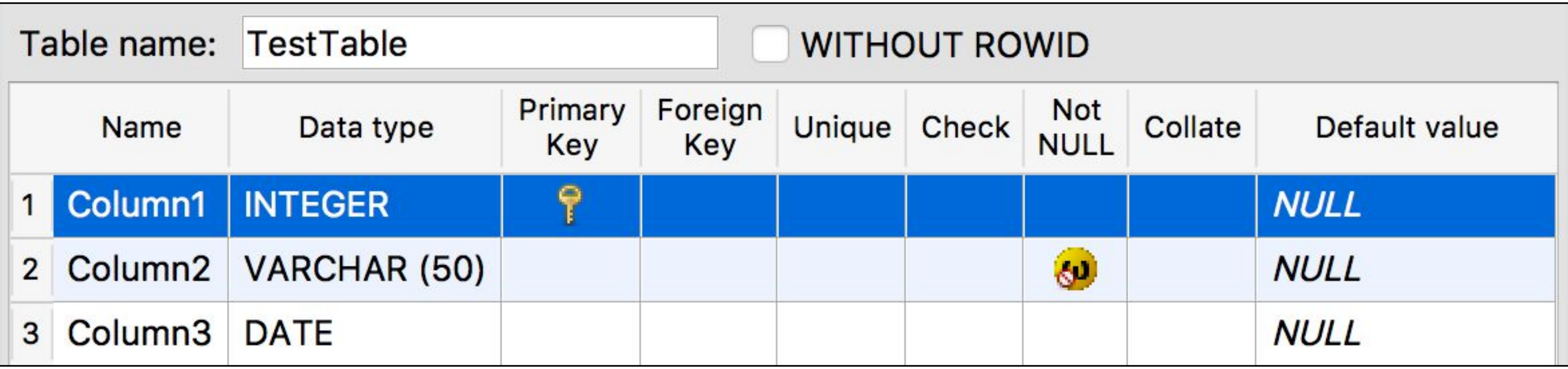


Table name: TestTable ☐ WITHOUT ROWID

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Default value
1	Column1	INTEGER	🔑						*NULL*
2	Column2	VARCHAR (50)					●		*NULL*
3	Column3	DATE							*NULL*

```
CREATE TABLE TestTable (
    Column1 INTEGER PRIMARY KEY,
    Column2 VARCHAR (50) NOT NULL,
    Column3 DATE)
```

```
DROP TABLE TestTable
```

Primary Key

A primary key is a field (or collection of fields) in a table.

It must satisfy these properties:

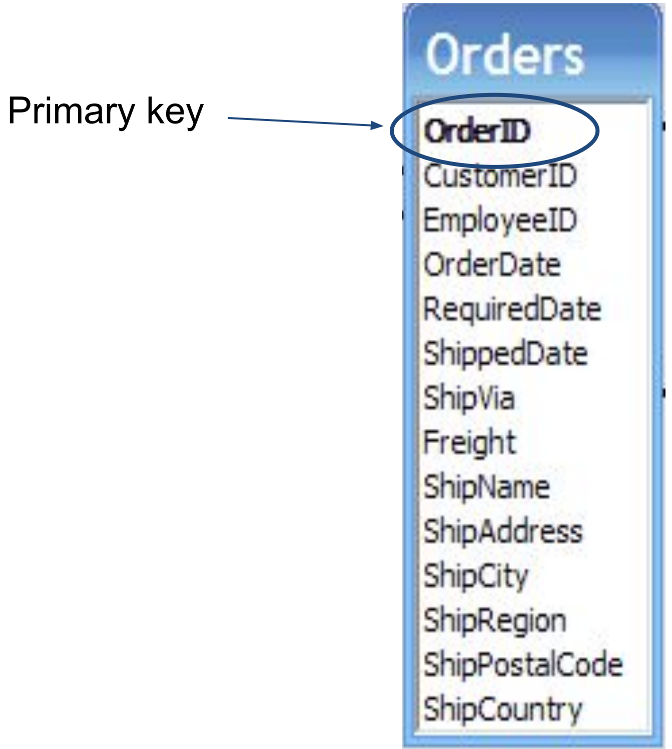
1. Each record has a unique value
2. No record has a NULL value

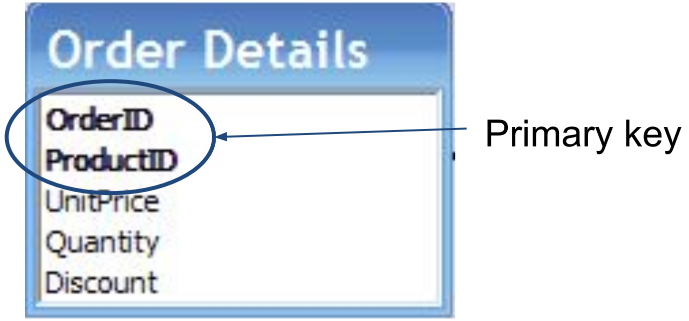
It helps the database identify a record uniquely.

If you try to add two records with the same value for a primary key, the database will give you an error.

If we just say “key”, we usually mean “primary key”

In our diagrams, we indicate primary keys with **boldface**





Inserting, Updating, and Deleting Data

SQL commands for altering data in tables

- Data Manipulation Language (DML) is the part of SQL used to manipulate the data within objects of a database
- Commands:
  - INSERT
  - UPDATE
  - DELETE
- For changing just a few rows, you can use SQLite Studio's GUI
- For changing lots of rows, either with SQL only or by calling SQL commands from R or Python, the commands are useful

INSERT

- The basic syntax is:

```
INSERT INTO Tablename
VALUES ('value1','value2','value3')
```

- Note the single quotes
- Use single quotes around strings or dates
- Don’t use single quotes around numeric data

Example

- Let’s create a table Clothing with the structure

ProductId	INTEGER
ProductDescrip	VARCHAR(25)
Cost	NUMBER(6,2)

…we would use

```
CREATE TABLE Clothing (
    ProductID INTEGER PRIMARY KEY,
    ProductDescrip VARCHAR (25),
    Cost NUMBER(6,2))
```

Example

- Now to insert values into the table Clothing with the structure

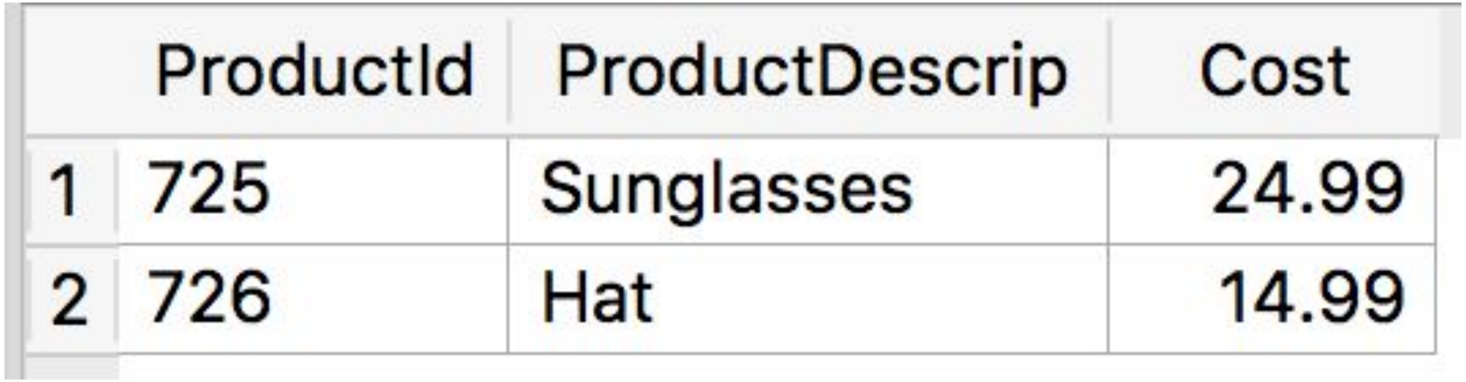
ProductId	INTEGER
ProductDescrip	VARCHAR(25)
Cost	NUMBER(6,2)

…we would use

```
INSERT INTO Clothing VALUES(725, 'Sunglasses' , 24.99);
INSERT INTO Clothing VALUES(726, 'Hat' , 14.99)
```

- Notice: separate queries with semicolon ;

Here’s Clothing after this INSERT

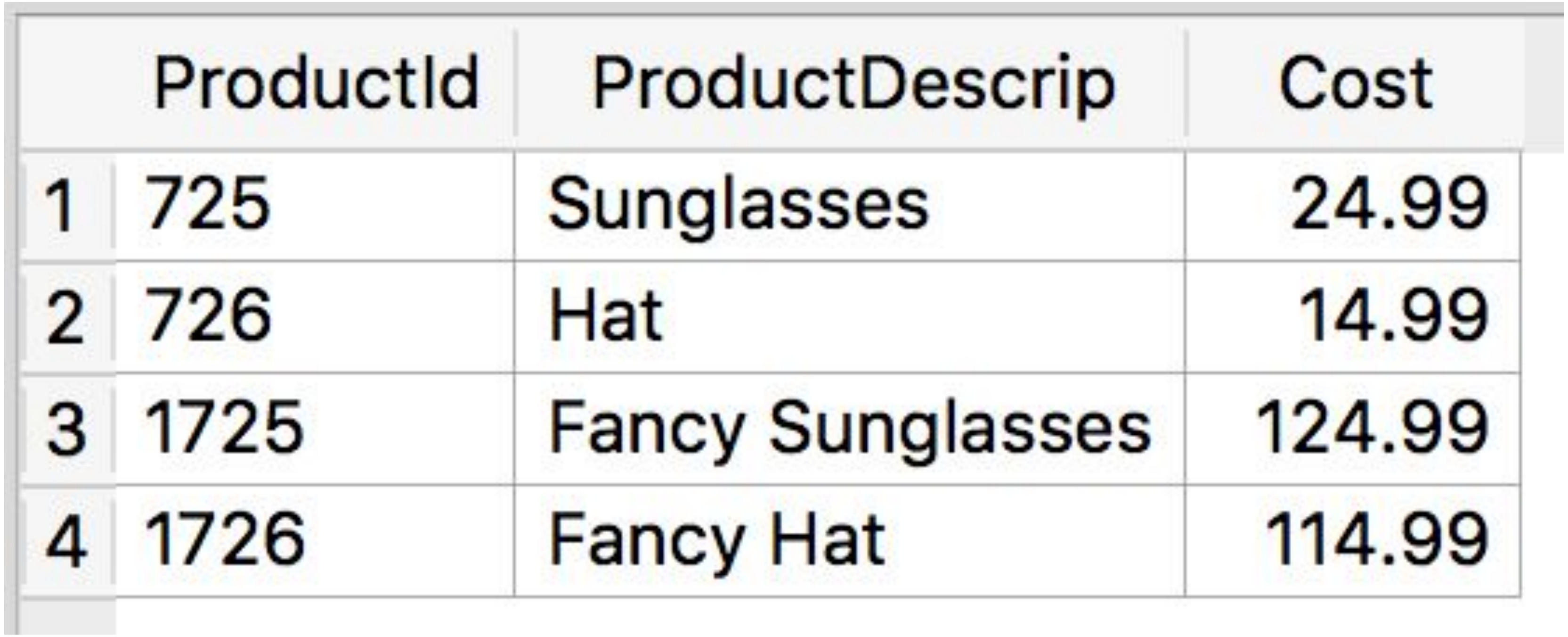


	ProductId	ProductDescrip	Cost
1	725	Sunglasses	24.99
2	726	Hat	14.99

You can insert the results of a query

```
INSERT INTO Clothing
    SELECT ProductId+1000,
        'Fancy ' || ProductDescrip,
        Cost+100
    FROM Clothing
```

There are 2 new rows in Clothing



	ProductId	ProductDescrip	Cost
1	725	Sunglasses	24.99
2	726	Hat	14.99
3	1725	Fancy Sunglasses	124.99
4	1726	Fancy Hat	114.99

UPDATE

- The simplest use of UPDATE is to update the value of a single column for a single record in a table
- The syntax is

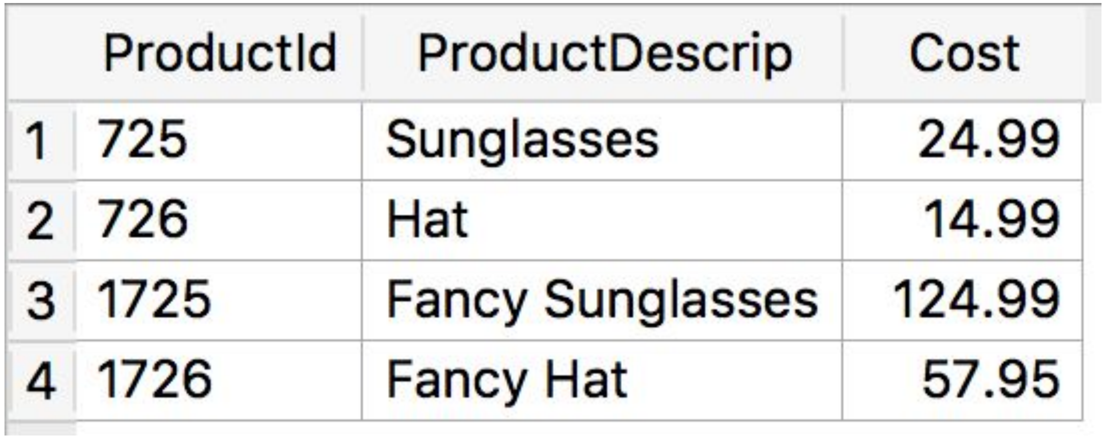
```
UPDATE TableName
SET ColumnName = 'value'
WHERE condition
```

Example

- If we want to lower the price for our fancy hat:

```
UPDATE Clothing
SET Cost = 57.95
WHERE ProductId = 1726
```

Now the fancy hat is $57.95



	ProductId	ProductDescrip	Cost
1	725	Sunglasses	24.99
2	726	Hat	14.99
3	1725	Fancy Sunglasses	124.99
4	1726	Fancy Hat	57.95

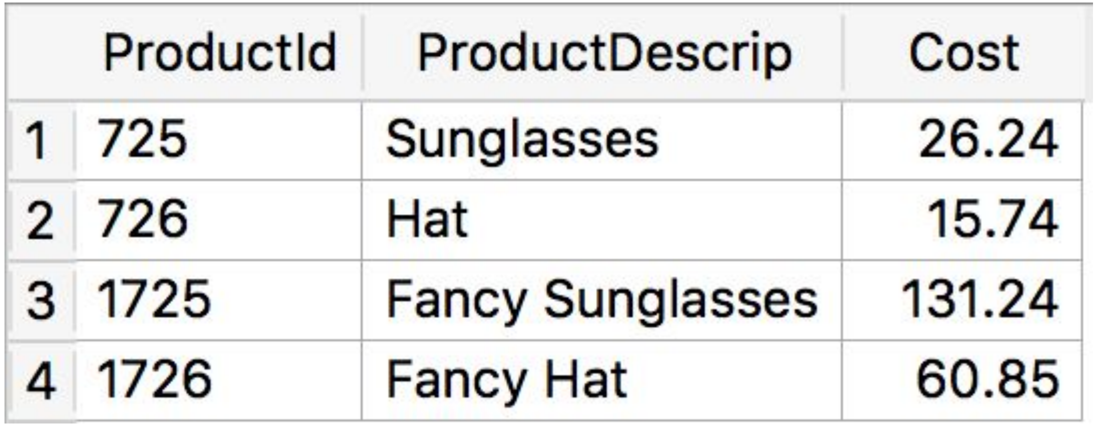
Another Example

- This command raises prices by 5%:

```
UPDATE Clothing
SET Cost = ROUND(Cost*1.05,2)
```

- This affects all of the rows in the table, and would take a long time to do manually on a table with a 10,000 records
- You could add a WHERE clause if you only wanted to raise the prices for some products

Now prices are 5% higher



	ProductId	ProductDescrip	Cost
1	725	Sunglasses	26.24
2	726	Hat	15.74
3	1725	Fancy Sunglasses	131.24
4	1726	Fancy Hat	60.85

DELETE

- Be careful with this command, you do not want to delete useful data by mistake!
- It removes an entire row of data from the table.
- It could be incorrect data, duplicate data, or a discontinued product, for example.

DELETE

The syntax is:

```
DELETE FROM TableName
WHERE condition
```

This is much better than the SQLiteStudio GUI if you have a lot of data to delete

Example

- Delete the cheap hat

```
DELETE FROM Clothing
WHERE ProductId = 726
```

- The following command deletes all the data in the table!

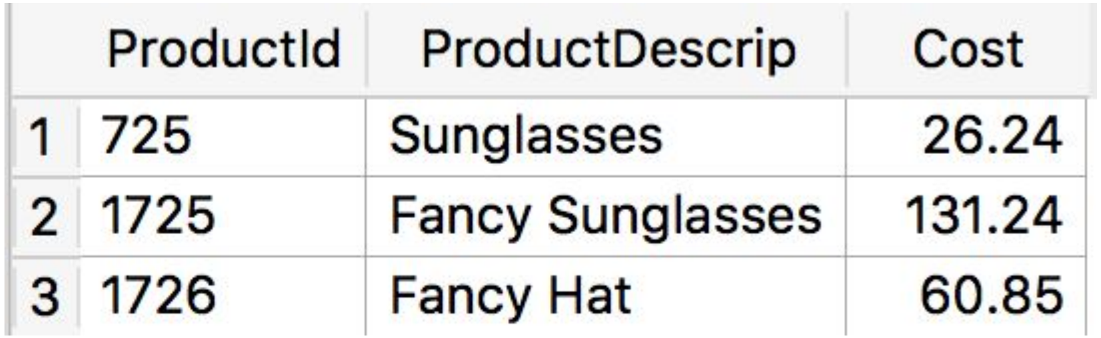
```
DELETE FROM Clothing
```

- That’s different from deleting the table itself:

```
DROP TABLE Clothing
```

Example

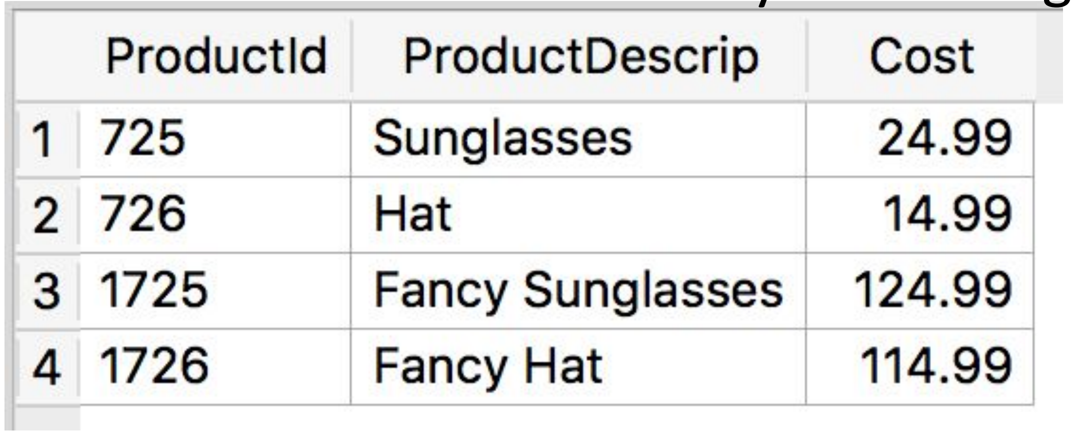
Here is the table after running:
DELETE FROM Clothing WHERE ProductId = 726



	ProductId	ProductDescrip	Cost
1	725	Sunglasses	26.24
2	1725	Fancy Sunglasses	131.24
3	1726	Fancy Hat	60.85

Lec3, Q6

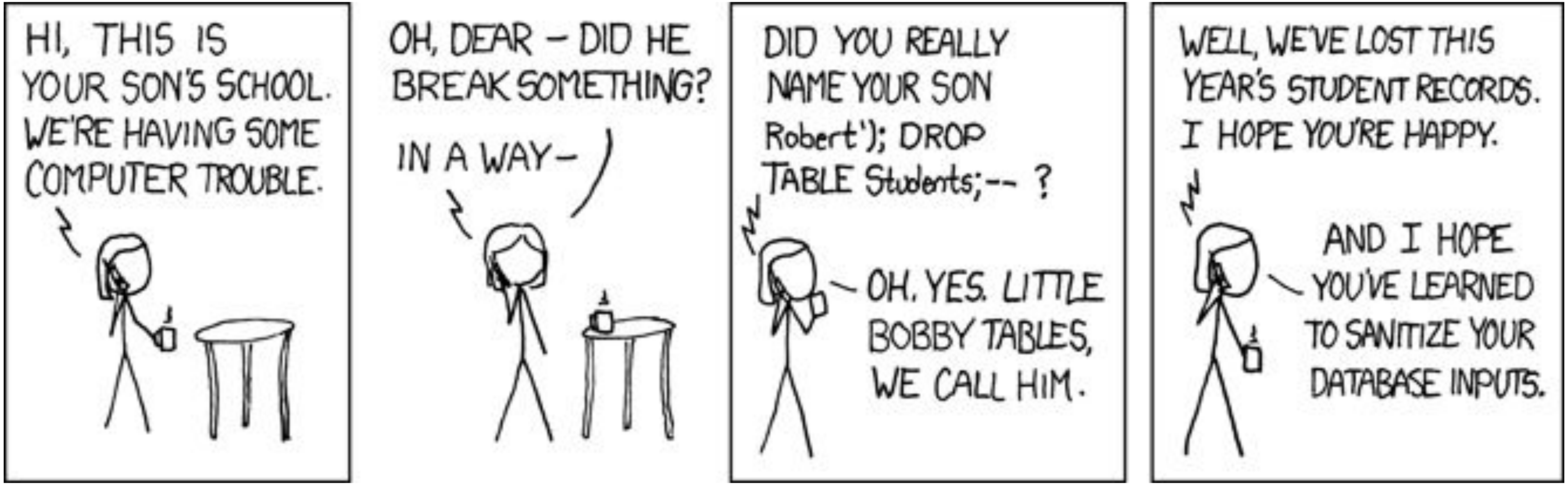
Here are the records currently in Clothing



	ProductId	ProductDescrip	Cost
1	725	Sunglasses	24.99
2	726	Hat	14.99
3	1725	Fancy Sunglasses	124.99
4	1726	Fancy Hat	114.99

(a) 4
(b) 3
(c) 2
(d) 1
(e) 0

How many records will it have after we run
DELETE FROM Clothing WHERE Cost > 100


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Tip for the Recitation

SQLite Studio Bugs

- When exporting results: The interface for selecting the database doesn’t always work on SQLite.
  Fix: disconnect from all of the databases except the one you want to export from.

Next lecture:
GROUP BY