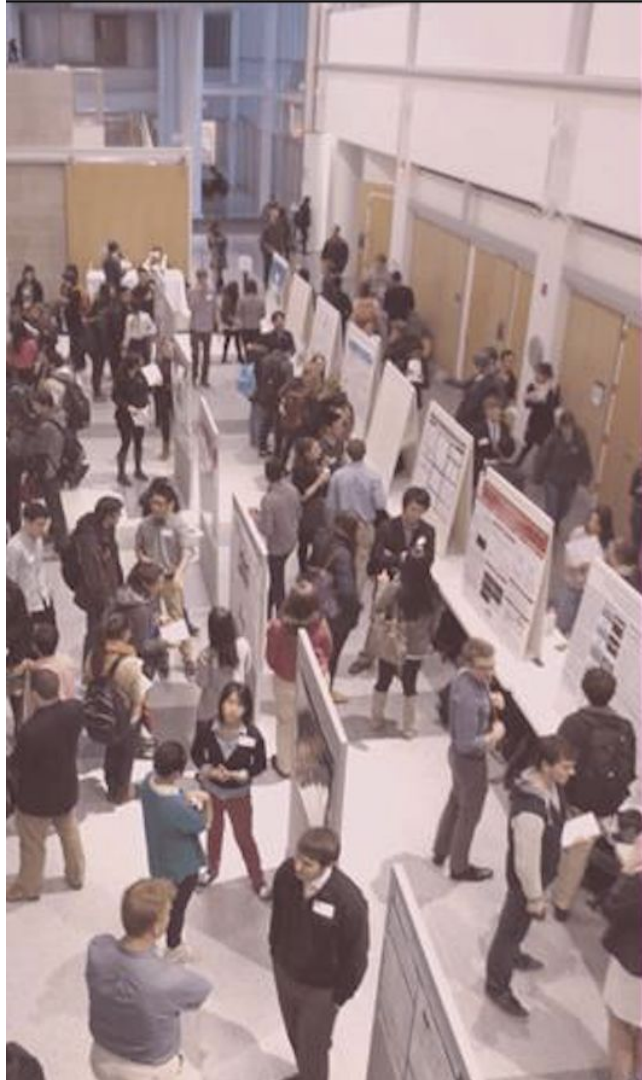


ORIE 3120

Lecture 3: SQL #2

[Basic queries (SELECT, WHERE, ORDER BY, ...),
schema design, DDL, DML]



DO YOU WANT TO PARTAKE IN
UNDERGRADUATE RESEARCH?

APPLY TO BECOME A MENTEE

Research Fields include:

Molecular Biology
Engineering
Computer Science
Humanities
Social Sciences
Economics, and many more!



APPLY BY FEBRUARY 14, 11:59 PM

<https://tinyurl.com/pmp20mentee>

**AS A MENTEE, YOU WILL HAVE THE
OPPORTUNITY TO:**

- Biweekly meetings and small group presentations
- Lab tours around campus
- Personalized mentorship by experienced upperclassmen

Questions? Email curbpeermentorship@gmail.com

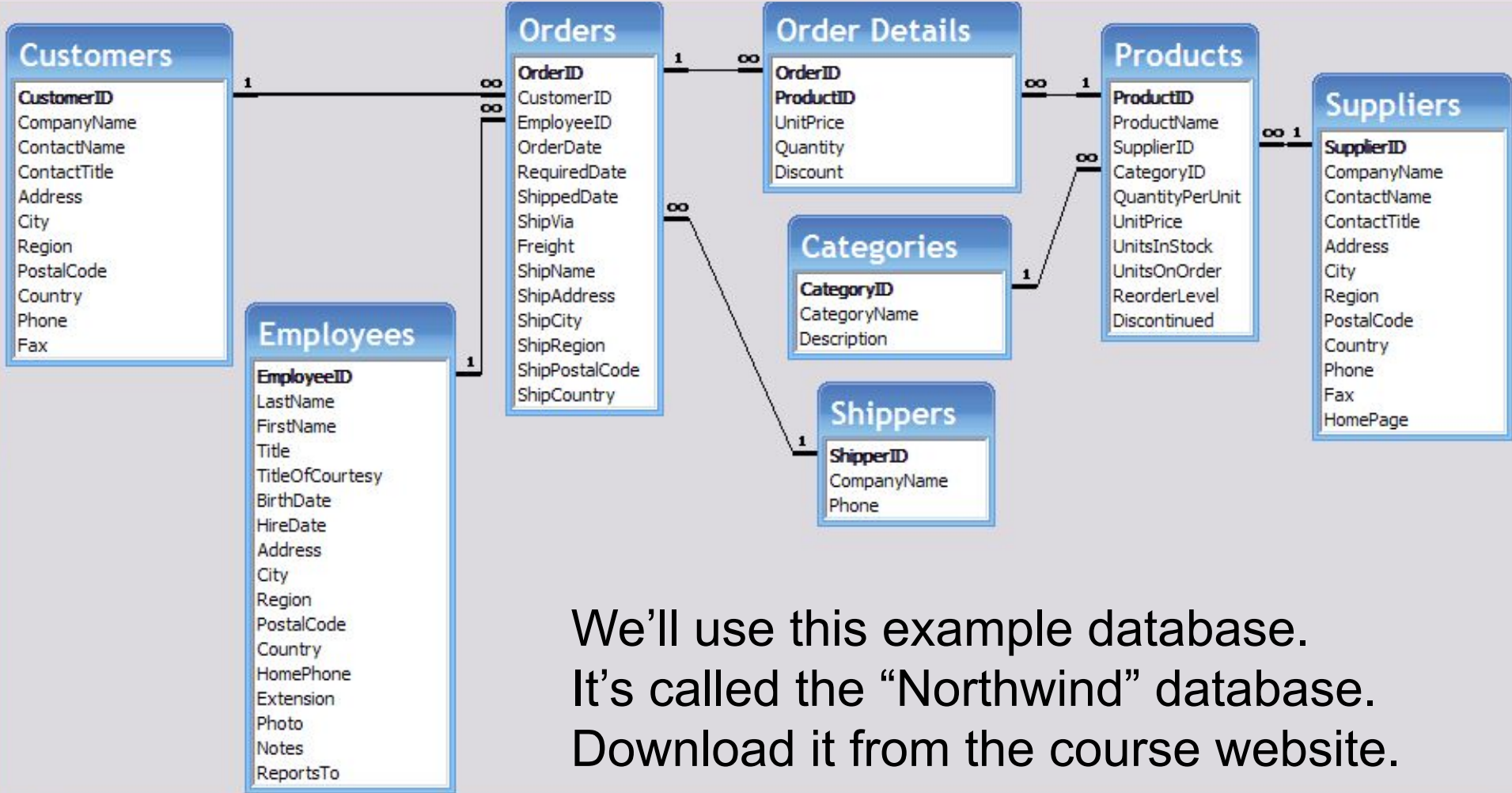


Questions from Piazza

- Format for hw?
 - SQL command + screenshot
- Where to find recitation, hw, data files, ...?
 - On the website under “Resources” tab
- When are TA office hours?
 - On the website under “Calendar” tab
- iClicker?
 - Ok to use clicker or phone app; recommend buying a clicker (wifi issues!)
 - We'll begin counting iClicker participation after add deadline

Install SQLiteStudio 3.2.1 for recitation & homework

- Download from <http://sqlitestudio.pl>
- Has versions for Windows, Linux, and Mac OSX.
- The Mac and Windows versions are a bit different.
- The recitations use screenshots from the Windows version.
If you want to use the Mac version that is mostly ok, but you may need to be patient while we help you work through issues from time to time.



We'll use this example database. It's called the "Northwind" database. Download it from the course website.

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
```

Note: the Northwind database slide I showed you has plural table names, “Products”, but the demo database has singular names, “Product”.

I'll write these queries using the singular names.

Here's that query's result

	Id	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 Product” means “get it from the table Product”
- 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 Product
```

- 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 Product  
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

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

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
7	1	1
8	2	1

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	4.1999999999999999
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\(N\)](#)

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.

12.6.2 Mathematical Functions

Table 12.12 Mathematical Functions

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

	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

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)

Concatenation

- The double-pipe operator `||` concatenates two strings

```
SELECT CompanyName || ' Ltd.' FROM Shippers
```

- Results:

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

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

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	4.1999999999999999
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

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
```

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

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

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

	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

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

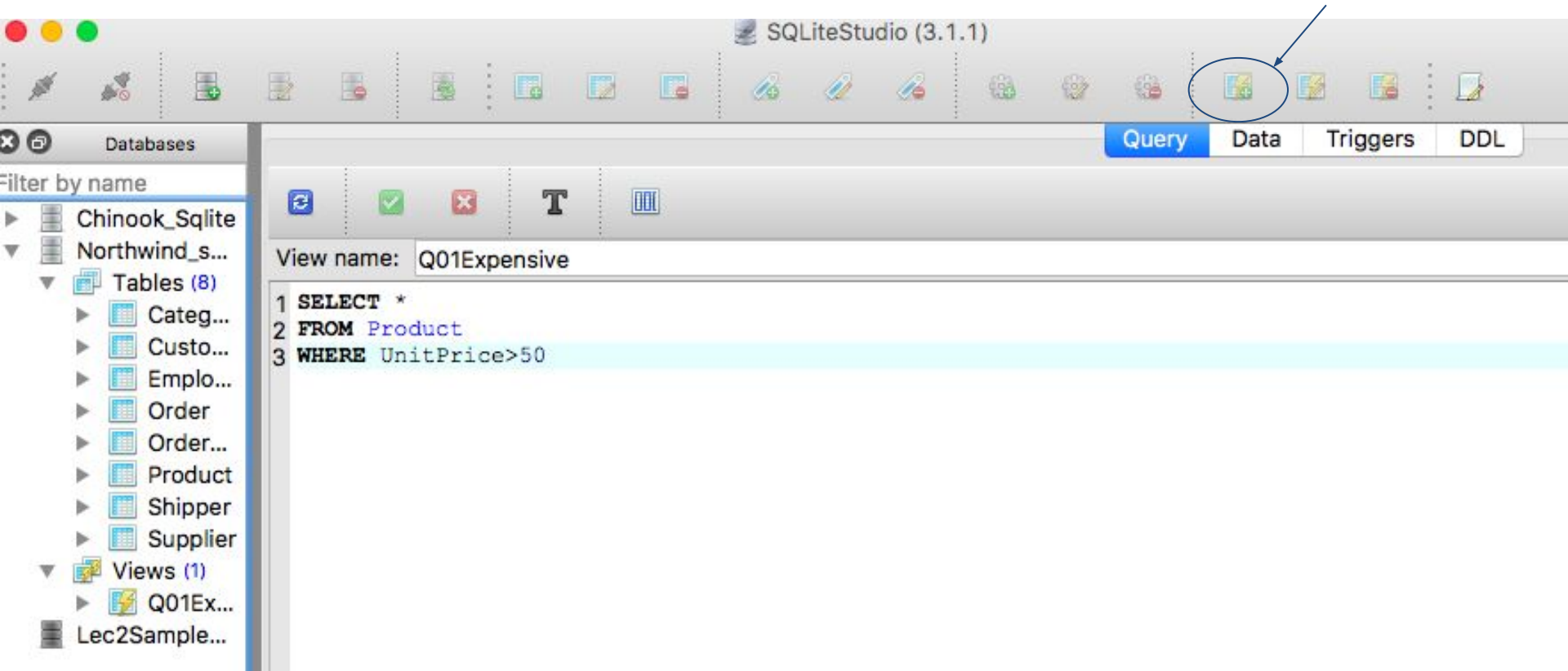
	STATE	N
1	Vermont	1
2	North Dakota	1
3	New Hampshire	1
4	Maine	1
5	Iowa	1
6	Utah	2
7	South Dakota	2
8	Oregon	2
9	Montana	2
10	Minnesota	2
11	Idaho	2
12	Hawaii	2
13	Wyoming	3
14	Washington	3
15	Rhode Island	3

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

Views

Views are saved queries

Create them by clicking this button



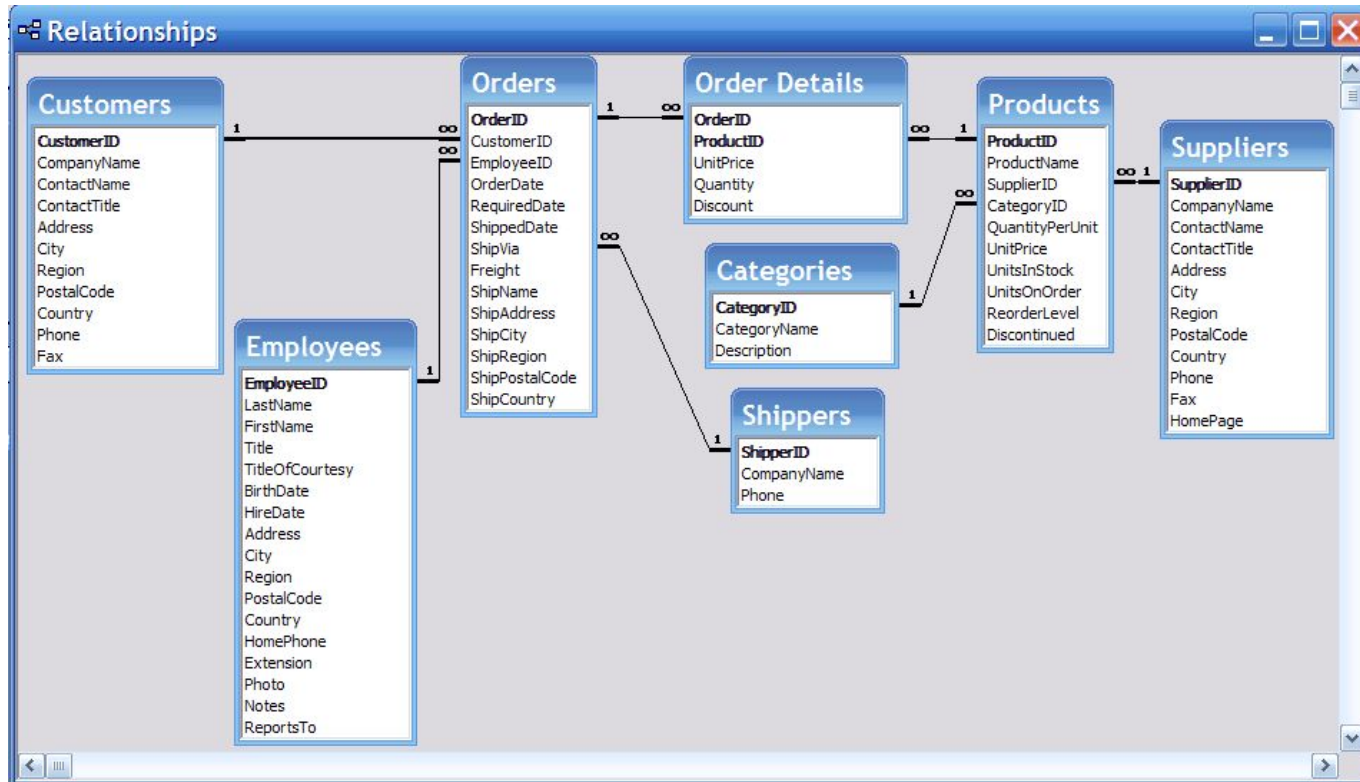
```
1 SELECT ProductName, UnitsInStock, UnitPrice FROM Q01Expensive
2 WHERE UnitsInStock>0
```

	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

Schema Design

From first lecture: A database schema is a collection of tables related by keys



Primary Key

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

It must satisfy these properties:

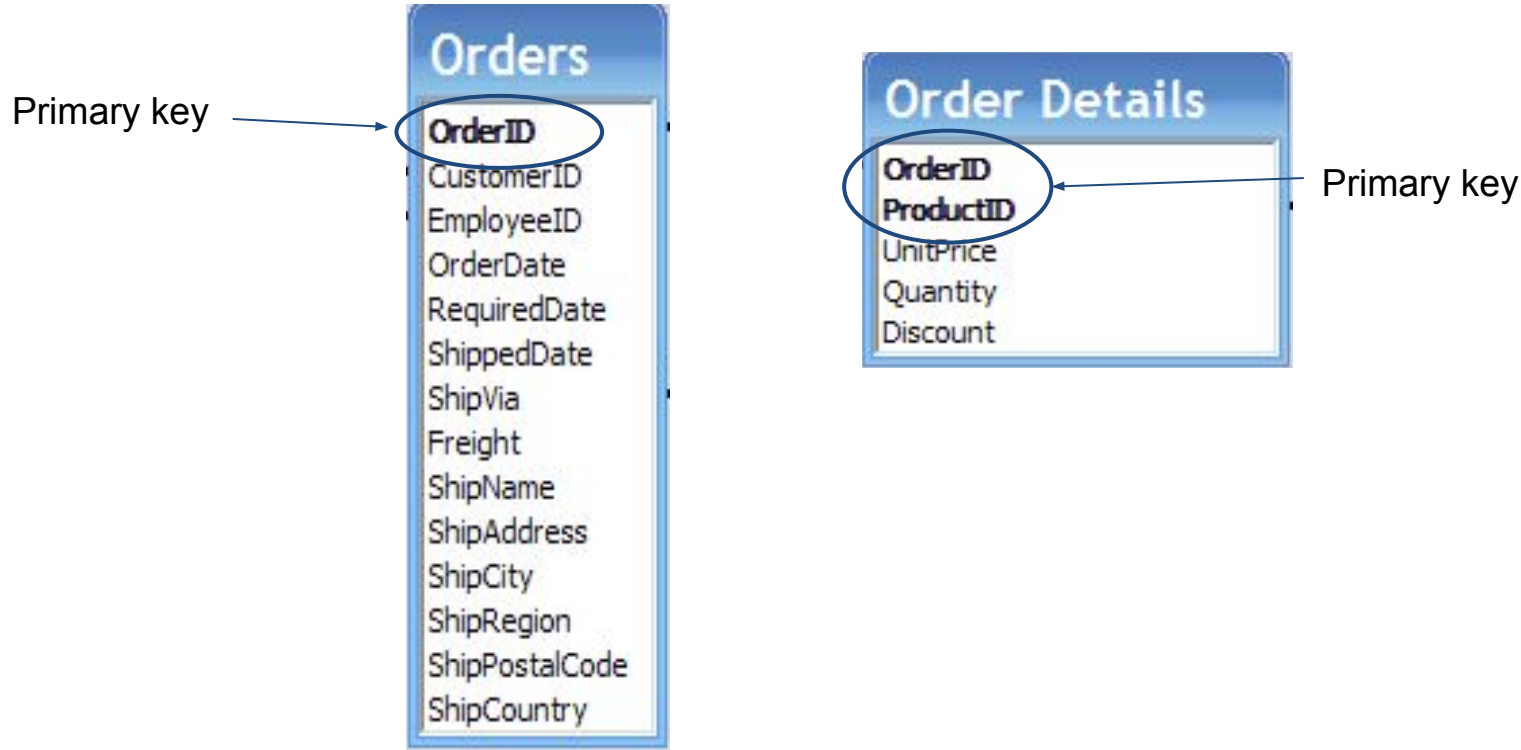
1. Each record has a unique value
2. No record has a NULL value

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”

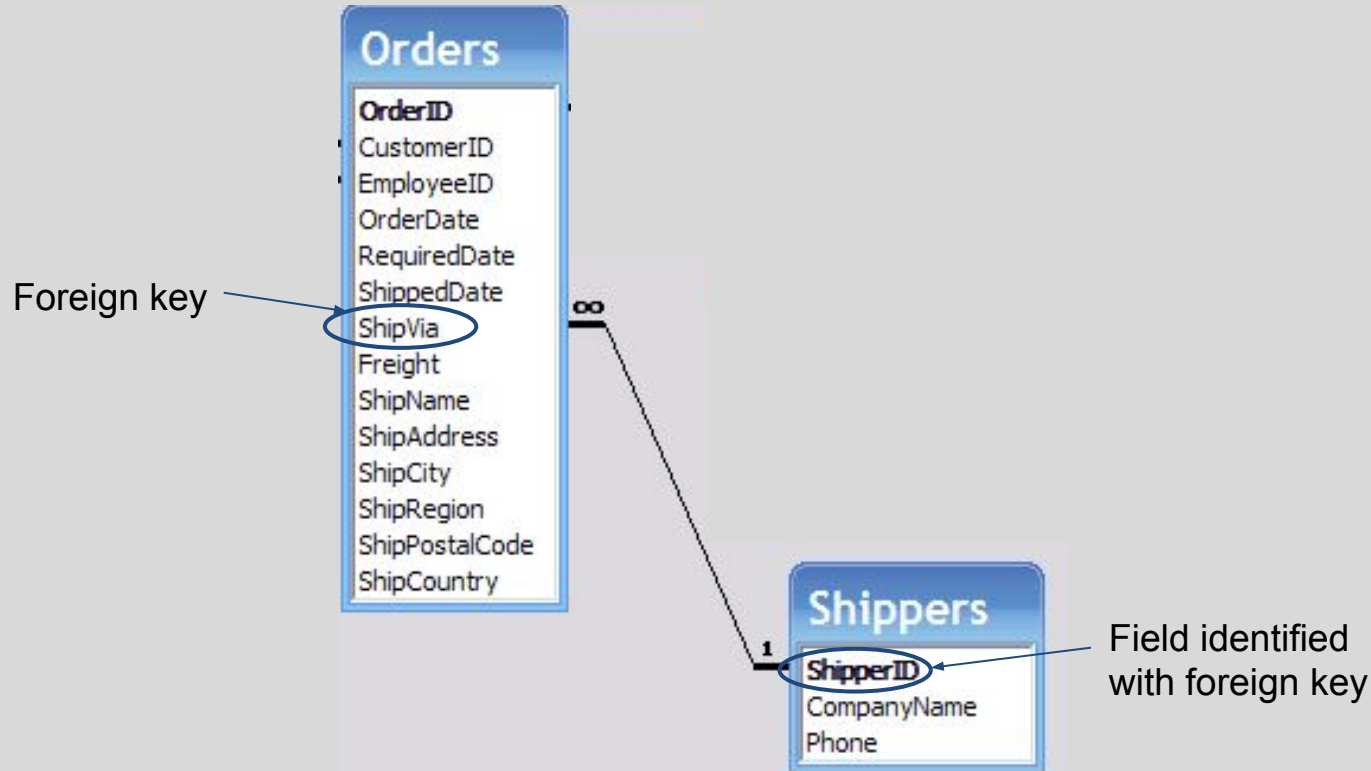
In our diagrams, we indicate primary keys with **boldface**



Definition: Foreign Key

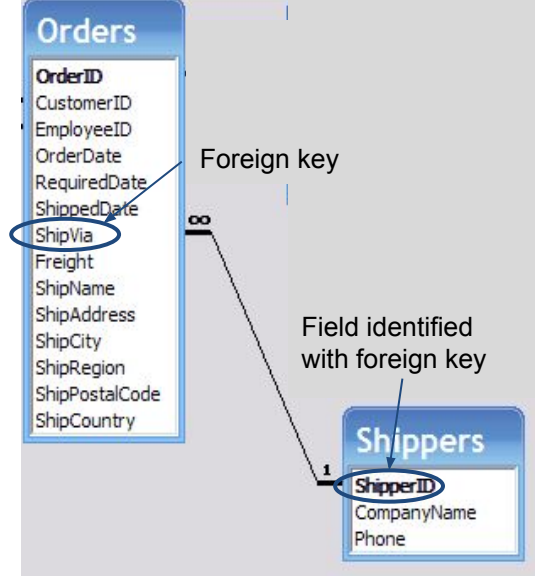
- A foreign key is a field (or collection of fields) in one table that references another field (or collection of fields).
- Values in the referenced field(s) must be unique.
- The referenced field(s) is/are usually in a different table, but can be in the same table.

Foreign Key Example



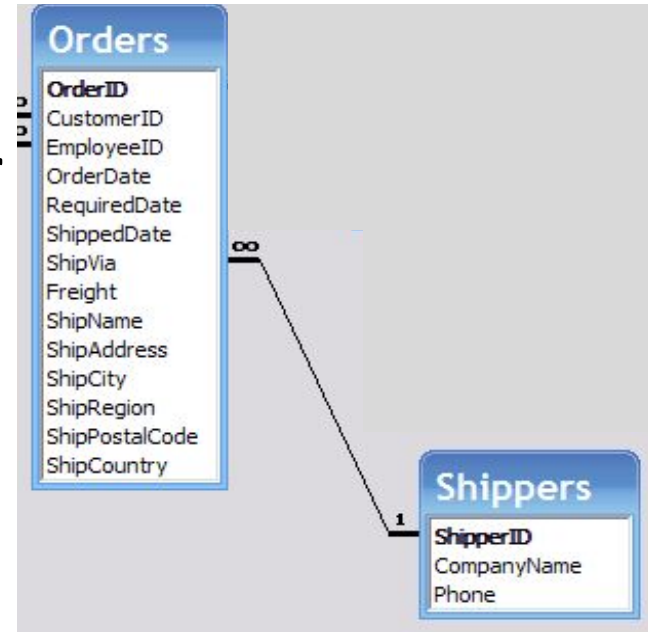
Foreign Key Example

OrderID	...	ShippedDate	ShipVia	...
1		1/24/2019	4	
2		1/24/2019	1	
3		1/25/2019	2	
4		1/26/2019	4	
...



ShipperID	CompanyName	Phone
1	UPS	888-123-4567
2	FedEx	888-314-1592
3	USPS	888-271-8281
4	DHL	888-141-4213

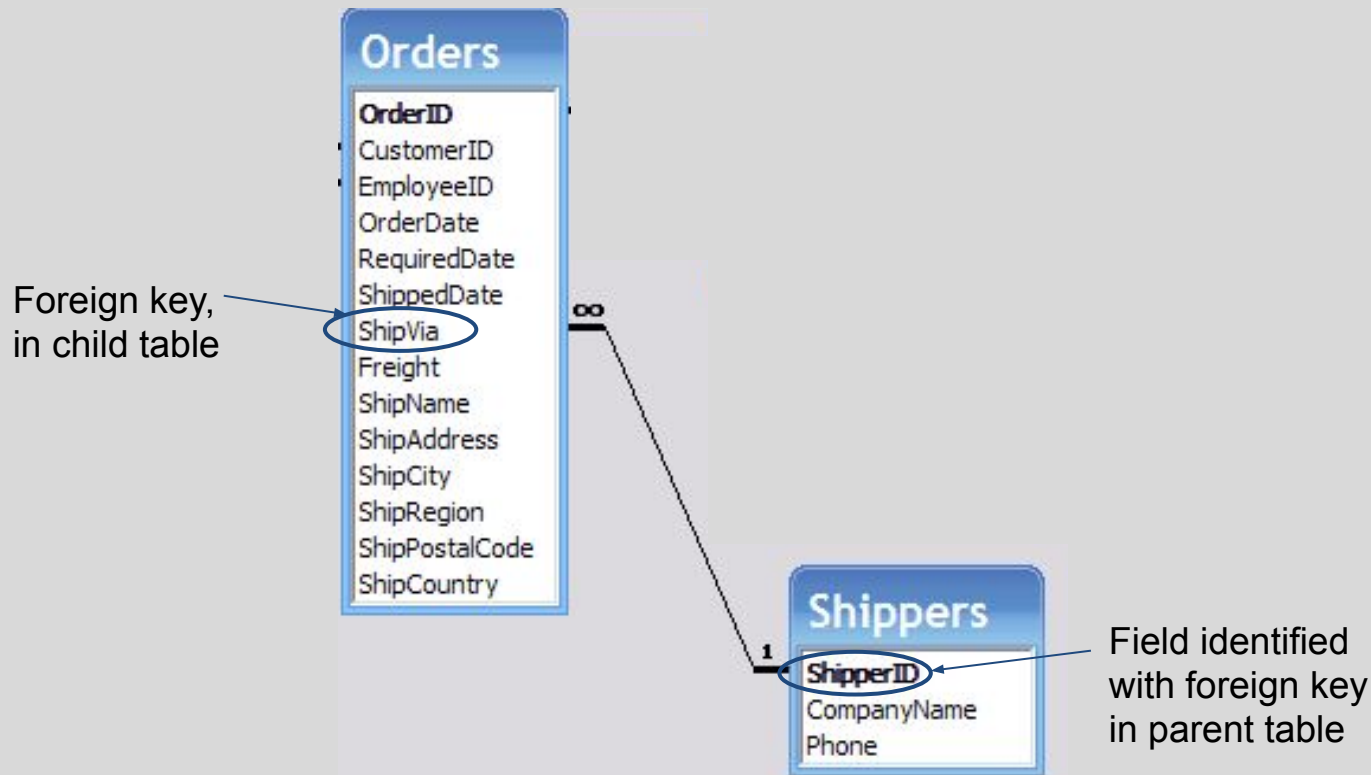
- The ShipVia field is a foreign key for the Orders table.
- This foreign key references the ShipperID field in the Shippers table.
- The foreign key ensures that for every ShipVia in the Orders table, there is a corresponding ShipperID in the Shippers table.



Foreign Key

- The table referencing is called the “child”
- The table being referenced is called the “parent”
- A foreign key indicates a one-to-many relationship:
 - A record in the parent table may be referenced by many records in the child table
 - A record in the child table references at most one parent

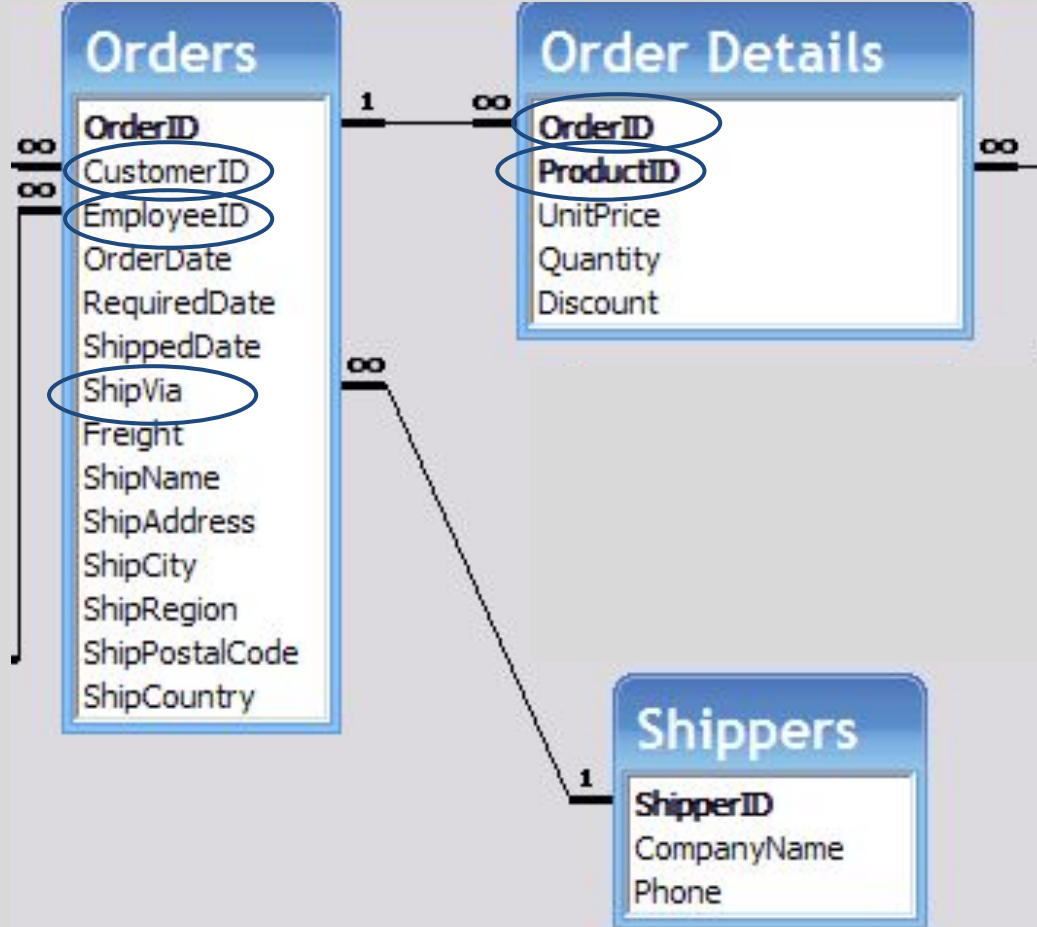
In our diagrams, we indicate foreign keys with a line labeled by ∞



- For a value to be inserted for ShipVia in the child table, a value for ShipperID in the parent table must exist.
- For a value to be removed for ShipperId in the parent table, all corresponding values for ShipVia must be removed from the child table.
- This is called “referential integrity”

- You can also have NULL values for a foreign key.
- This means that record in the child table does not refer to the parent table.

Here are some more foreign keys



Which of these are legal (foreign key, referenced field) pairs?

T1

	A	B
1	1	A
2	1	C
3	1	B
4	2	D
5	3	E

T2

	C	D
1	5	A
2	4	B
3	3	C
4	2	D
5	1	E

- (a) T1.A references T2.C
- (b) T1.A references T2.D
- (c) T2.C references T1.A
- (d) T2.D references T1.A
- (e) None of the above

*do not include keys that reference themselves;
only include keys that are a single field

Which of these are legal (foreign key, referenced field) pairs?

T1

	A	B
1	1	A
2	1	C
3	1	B
4	2	D
5	3	E

T2

	C	D
1	5	A
2	4	B
3	3	C
4	2	D
5	1	E

- (a) T1.B references T2.C
- (b) T1.B references T2.D
- (c) T2.C references T1.B
- (d) T2.D references T1.B
- (e) None of the above

*do not include keys that reference themselves;
only include keys that are a single field

How many legal
(foreign key, referenced field) pairs can
you identify in these two tables*?

T1

	A	B
1	1	A
2	1	C
3	1	B
4	2	D
5	3	E

T2

	C	D
1	5	A
2	4	B
3	3	C
4	2	D
5	1	E

(a) 0

(b) 1

(c) 2

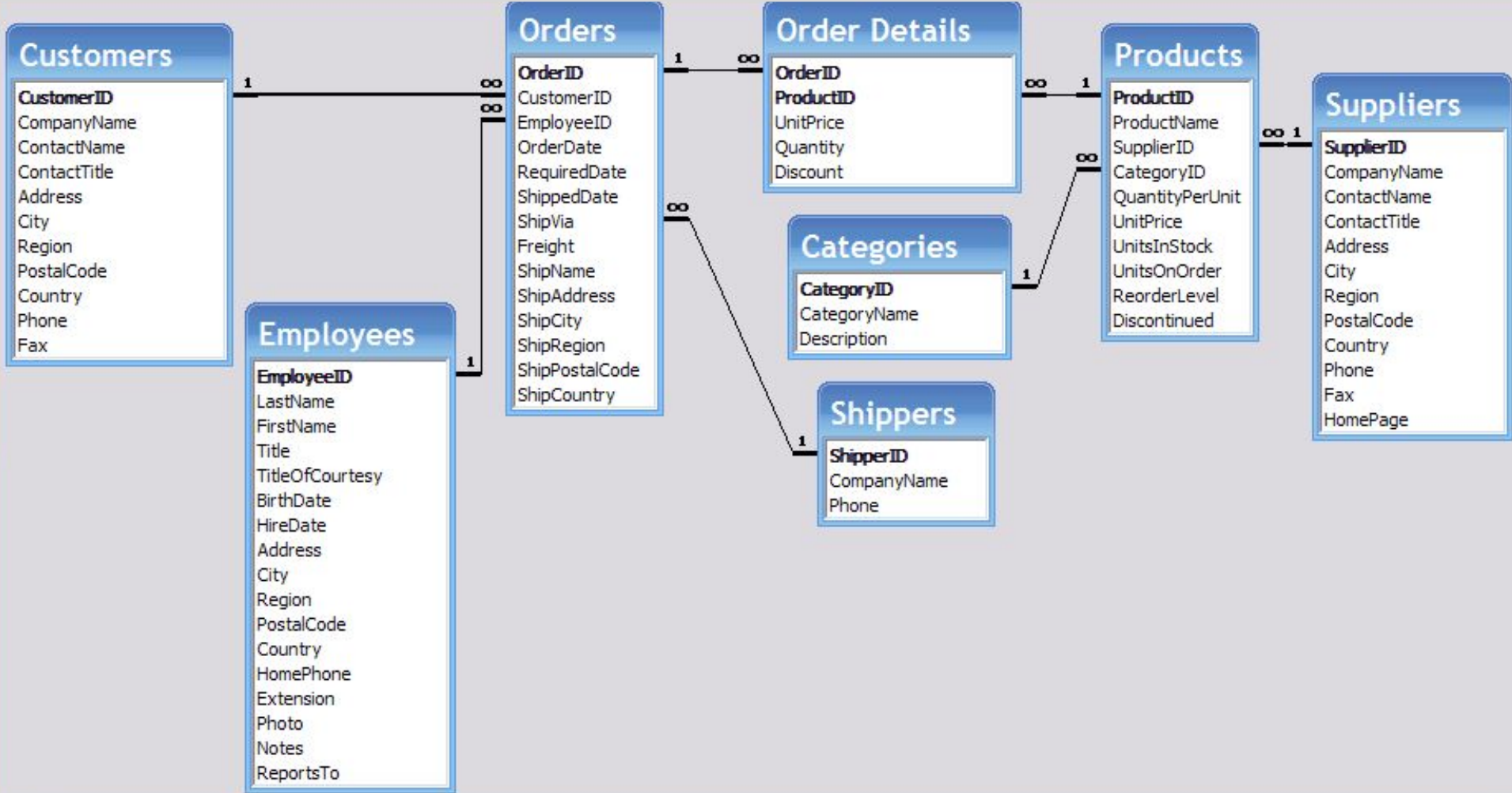
(d) 3

(e) 4

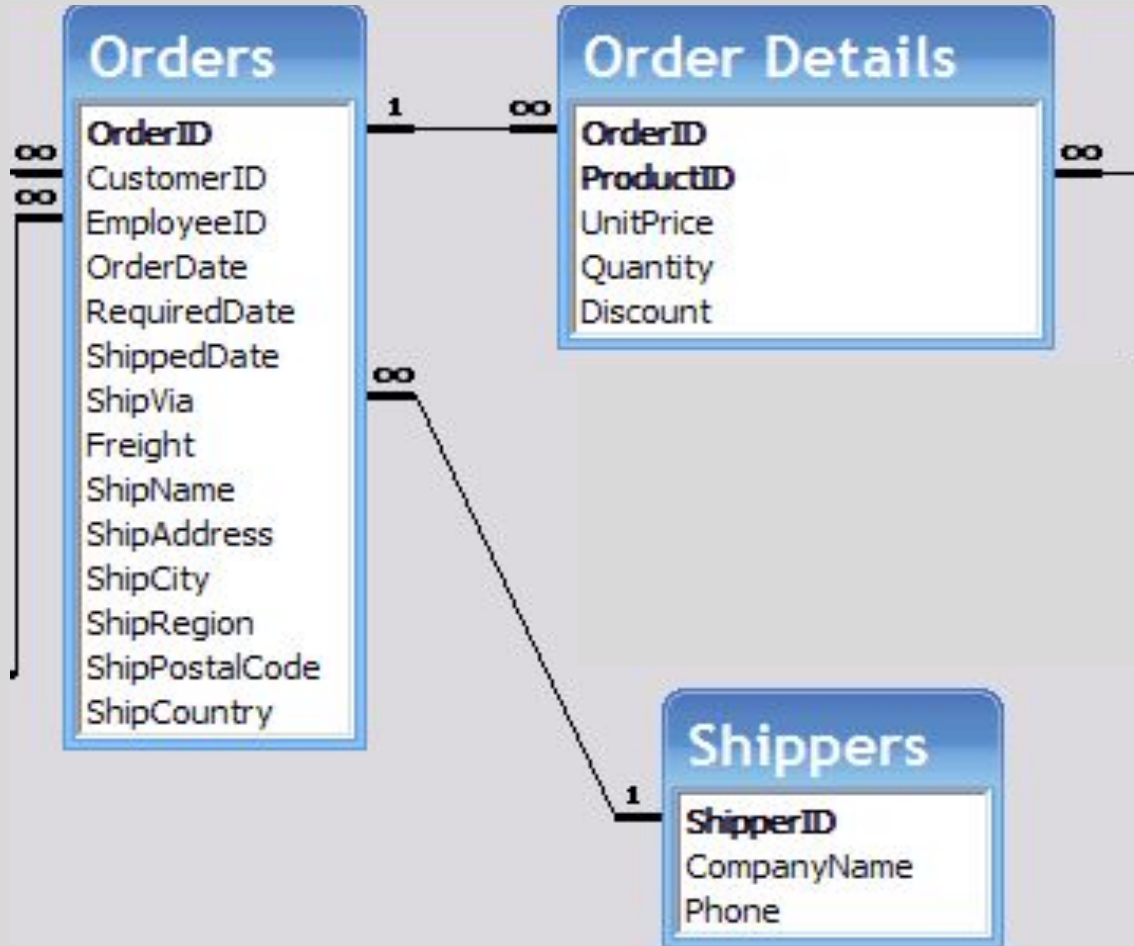
*do not include keys that reference themselves;
only include keys that are a single field

Let's do an exercise in schema design

- Suppose our orders can go out in multiple shipments
- How should we change our schema?



How can we change our schema to handle multiple shipments per order?





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: WITHOUT ROWID

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Default value
1	Column1	INTEGER							<i>NULL</i>
2	Column2	VARCHAR (50)							<i>NULL</i>
3	Column3	DATE							<i>NULL</i>

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

```
DROP TABLE TestTable
```

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!
- We put single quotes around most data types, but not 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

Fancier INSERT

```
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

- If we wanted to raise 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 GUI if you have a lot of data to delete

Example

- Delete that 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
```

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.

Result of first command

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

Next lecture: GROUP BY