DATABASE MANAGERS

(Data for this tutorial at www.peteraldhous.com/Data)

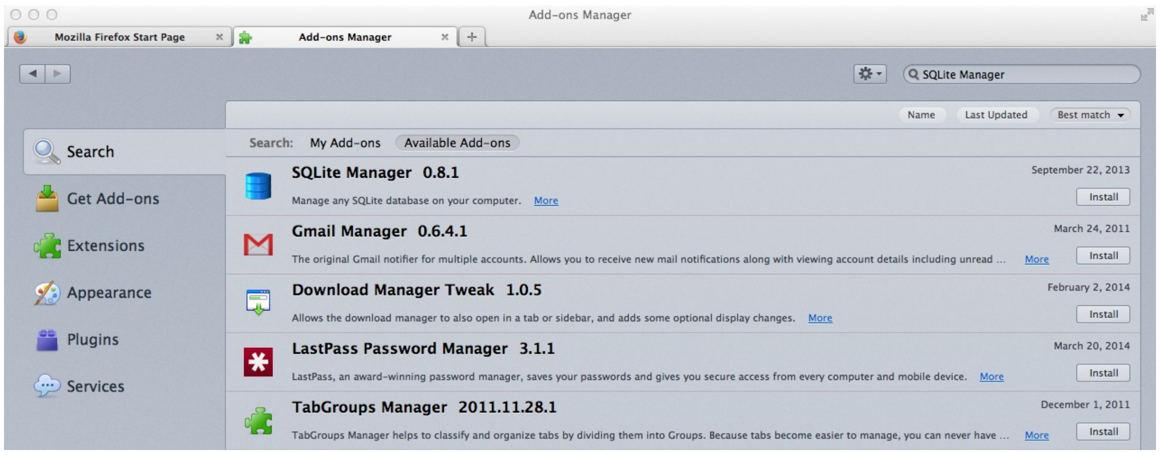
We've seen how spreadsheets can filter data and calculate subtotals. Database managers can handle larger datasets, and with practice are more flexible and nimble for filtering, grouping and aggregating data.

They also allow you to join multiple data tables into one, or match records across different datasets, if they have common fields – which can be a powerful tool. Again, we'll work with data used in reporting this story, about the drug company Pfizer's payments to doctors.

We will work with SQLite, database software that can be managed using a free add-on to the Firefox browser.

(Firefox uses SQLite to store information including your bookmarks; using the SQLite Manager add-on, you can manage any SQLite database.)

First, download and install SQLite Manager. In Firefox, select **Tools>Add-ons** and type **SQLite Manager** in the search box at top right. You should now see the add-on under the **Available Add-ons** tab:



SQLite Manager 0.8.1 September 22, 2013
Manage any SQLite database on your computer. More Install

Gmail Manager 0.6.4.1 March 24, 2011
The original Gmail notifier for multiple accounts. Allows you to receive new mail notifications along with viewing account details including unread ... More Install

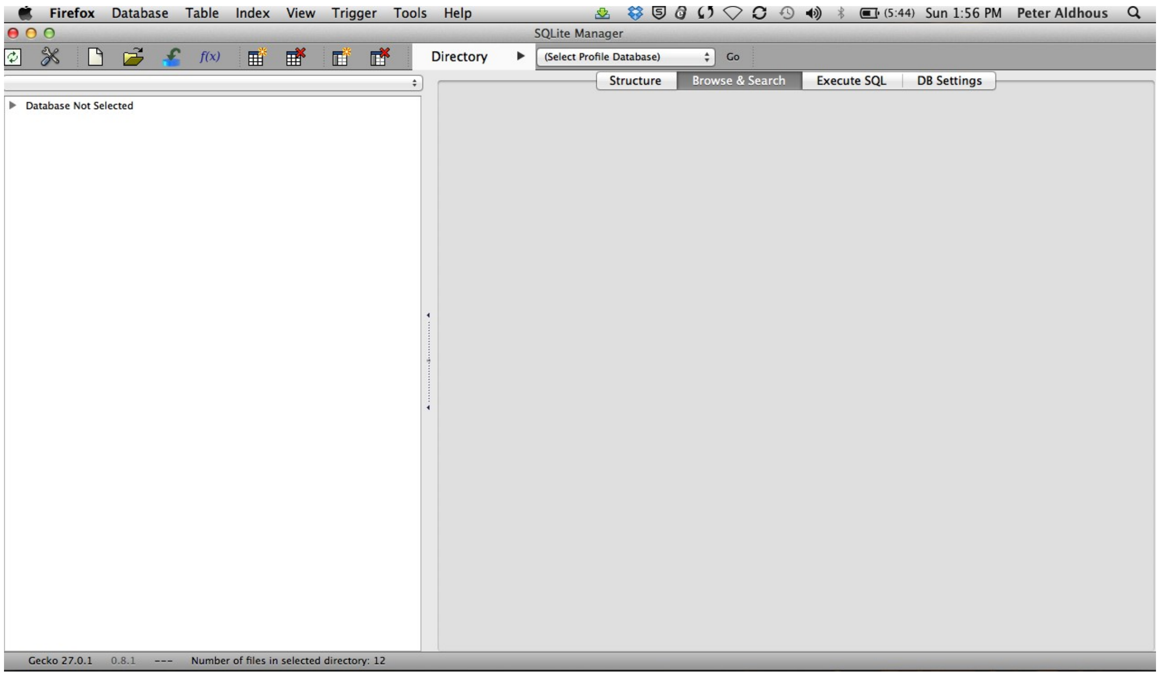
Download Manager Tweak 1.0.5 February 2, 2014
Allows the download manager to also open in a tab or sidebar, and adds some optional display changes. More Install

LastPass Password Manager 3.1.1 March 20, 2014
LastPass, an award-winning password manager, saves your passwords and gives you secure access from every computer and mobile device. More Install

TabGroups Manager 2011.11.28.1 December 1, 2011
TabGroups Manager helps to classify and organize tabs by dividing them into Groups. Because tabs become easier to manage, you can never have ... More Install

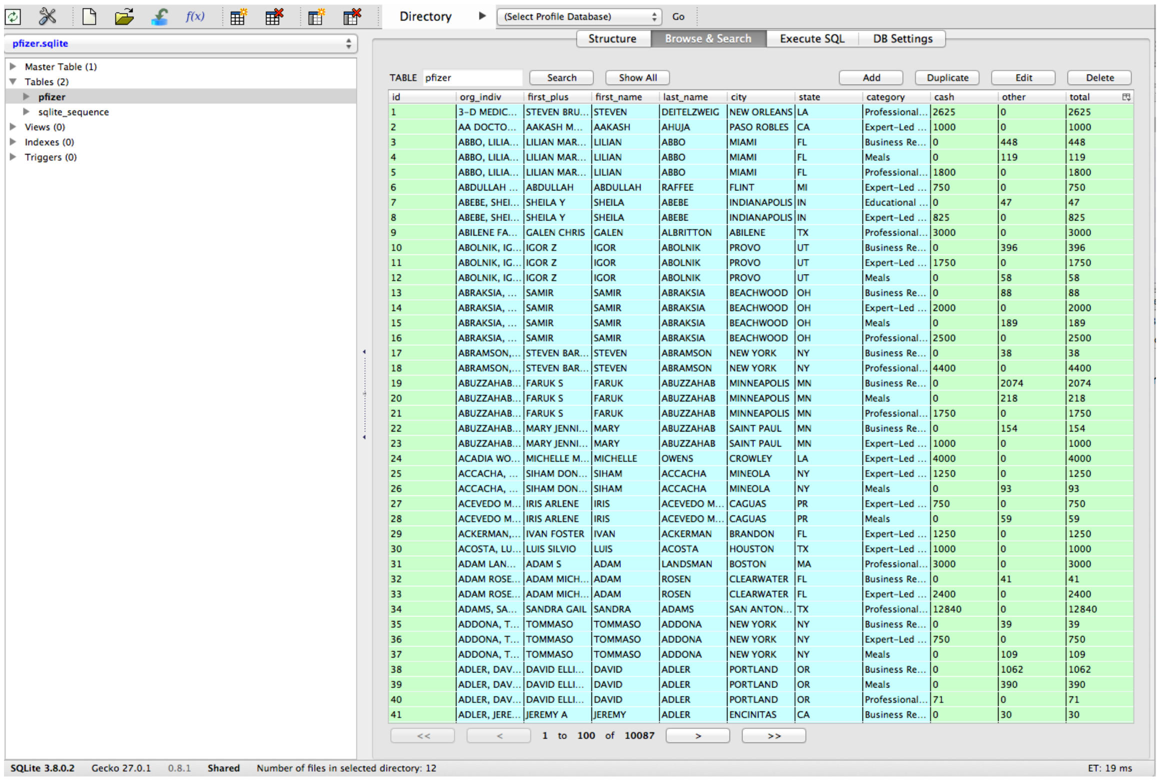
Click **Install** and restart Firefox.

Open SQLite Manager by selecting **Tools>SQLite Manager** in Firefox. You should see a screen like this:



Now open the database **pfizer.sqlite**, available online here, by selecting **Database>Connect Database**. Navigate to the database file, and click **Open**.

After the database opens, select the table **pfizer** in the panel to the left, and click the **Browse and Search** tab in the right-hand panel. You should now be able to see the first few rows of the data in the table:



id	org_indiv	first_plus	first_name	last_name	city	state	category	cash	other	total
1	3-D MEDIC...	STEVEN BRU...	STEVEN	DEITELZWEIG	NEW ORLEANS	LA	Professional...	2625	0	2625
2	AA DOCTO...	AAKASH M...	AAKASH	AHUJA	PASO ROBLES	CA	Expert-Led ...	1000	0	1000
3	ABBO, LILIA...	LILIAN MAR...	LILIAN	ABBO	MIAMI	FL	Business Re...	0	448	448
4	ABBO, LILIA...	LILIAN MAR...	LILIAN	ABBO	MIAMI	FL	Meals	0	119	119
5	ABBO, LILIA...	LILIAN MAR...	LILIAN	ABBO	MIAMI	FL	Professional...	1800	0	1800
6	ABDULLAH ...	ABDULLAH	ABDULLAH	RAFFEE	FLINT	MI	Expert-Led ...	750	0	750
7	ABEBE, SHEI...	SHEILA Y	SHEILA	ABEBE	INDIANAPOLIS	IN	Educational ...	0	47	47
8	ABEBE, SHEI...	SHEILA Y	SHEILA	ABEBE	INDIANAPOLIS	IN	Expert-Led ...	825	0	825
9	ABILENE FA...	GALEN CHRIS	GALEN	ALBRITTON	ABILENE	TX	Professional...	3000	0	3000
10	ABOLNIK, IG...	IGOR Z	IGOR	ABOLNIK	PROVO	UT	Business Re...	0	396	396
11	ABOLNIK, IG...	IGOR Z	IGOR	ABOLNIK	PROVO	UT	Expert-Led ...	1750	0	1750
12	ABOLNIK, IG...	IGOR Z	IGOR	ABOLNIK	PROVO	UT	Meals	0	58	58
13	ABRAKSIA, ...	SAMIR	SAMIR	ABRAKSIA	BEACHWOOD	OH	Business Re...	0	88	88
14	ABRAKSIA, ...	SAMIR	SAMIR	ABRAKSIA	BEACHWOOD	OH	Expert-Led ...	2000	0	2000
15	ABRAKSIA, ...	SAMIR	SAMIR	ABRAKSIA	BEACHWOOD	OH	Meals	0	189	189
16	ABRAKSIA, ...	SAMIR	SAMIR	ABRAKSIA	BEACHWOOD	OH	Professional...	2500	0	2500
17	ABRAMSON,...	STEVEN BAR...	STEVEN	ABRAMSON	NEW YORK	NY	Business Re...	0	38	38
18	ABRAMSON,...	STEVEN BAR...	STEVEN	ABRAMSON	NEW YORK	NY	Professional...	4400	0	4400
19	ABUZZAHAB...	FARUK S	FARUK	ABUZZAHAB	MINNEAPOLIS	MN	Business Re...	0	2074	2074
20	ABUZZAHAB...	FARUK S	FARUK	ABUZZAHAB	MINNEAPOLIS	MN	Meals	0	218	218
21	ABUZZAHAB...	FARUK S	FARUK	ABUZZAHAB	MINNEAPOLIS	MN	Professional...	1750	0	1750
22	ABUZZAHAB...	MARY JENNI...	MARY	ABUZZAHAB	SAINT PAUL	MN	Business Re...	0	154	154
23	ABUZZAHAB...	MARY JENNI...	MARY	ABUZZAHAB	SAINT PAUL	MN	Expert-Led ...	1000	0	1000
24	ACADIA WO...	MICHELLE M...	MICHELLE	OWENS	CROWLEY	LA	Expert-Led ...	4000	0	4000
25	ACCACHA, ...	SIHAM DON...	SIHAM	ACCACHA	MINEOLA	NY	Expert-Led ...	1250	0	1250
26	ACCACHA, ...	SIHAM DON...	SIHAM	ACCACHA	MINEOLA	NY	Meals	0	93	93
27	ACEVEDO M...	IRIS ARLENE	IRIS	ACEVEDO M...	CAGUAS	PR	Expert-Led ...	750	0	750
28	ACEVEDO M...	IRIS ARLENE	IRIS	ACEVEDO M...	CAGUAS	PR	Meals	0	59	59
29	ACKERMAN,...	IVAN FOSTER	IVAN	ACKERMAN	BRANDON	FL	Expert-Led ...	1250	0	1250
30	ACOSTA, LU...	LUIS SILVIO	LUIS	ACOSTA	HOUSTON	TX	Expert-Led ...	1000	0	1000
31	ADAM LAN...	ADAM S	ADAM	LANDSMAN	BOSTON	MA	Professional...	3000	0	3000
32	ADAM ROSE...	ADAM MICH...	ADAM	ROSEN	CLEARWATER	FL	Business Re...	0	41	41
33	ADAM ROSE...	ADAM MICH...	ADAM	ROSEN	CLEARWATER	FL	Expert-Led ...	2400	0	2400
34	ADAMS, SA...	SANDRA GAIL	SANDRA	ADAMS	SAN ANTON...	TX	Professional...	12840	0	12840
35	ADDONA, T...	TOMMASO	TOMMASO	ADDONA	NEW YORK	NY	Business Re...	0	39	39
36	ADDONA, T...	TOMMASO	TOMMASO	ADDONA	NEW YORK	NY	Expert-Led ...	750	0	750
37	ADDONA, T...	TOMMASO	TOMMASO	ADDONA	NEW YORK	NY	Meals	0	109	109
38	ADLER, DAV...	DAVID ELLI...	DAVID	ADLER	PORTLAND	OR	Business Re...	0	1062	1062
39	ADLER, DAV...	DAVID ELLI...	DAVID	ADLER	PORTLAND	OR	Meals	0	390	390
40	ADLER, DAV...	DAVID ELLI...	DAVID	ADLER	PORTLAND	OR	Professional...	71	0	71
41	ADLER, JERE...	JEREMY A	JEREMY	ADLER	ENCINITAS	CA	Business Re...	0	30	30

Notice that it looks much like a spreadsheet, except there is no co-ordinate system. Instead, the column names, called “fields” in a database, are fixed, and each row or “record” has a unique ID number, created by SQLite as a “Primary Key” when the data was imported. (We’ll do this with a new table in a few minutes.)

Notice also that the field names are simplified and have no spaces. This will keep things succinct when we write database queries. SQLite Manager also color-codes the fields by the type of data they contain: here numbers have a light green background and text fields are light blue.

Database queries

1. Filtering and sorting data

To extract information from our database, we need to ask for it in the language that databases understand: Structured Query Language, or SQL. Don't panic: the logic of SQL is very easy to follow – it's the closest that computer code comes to plain English.

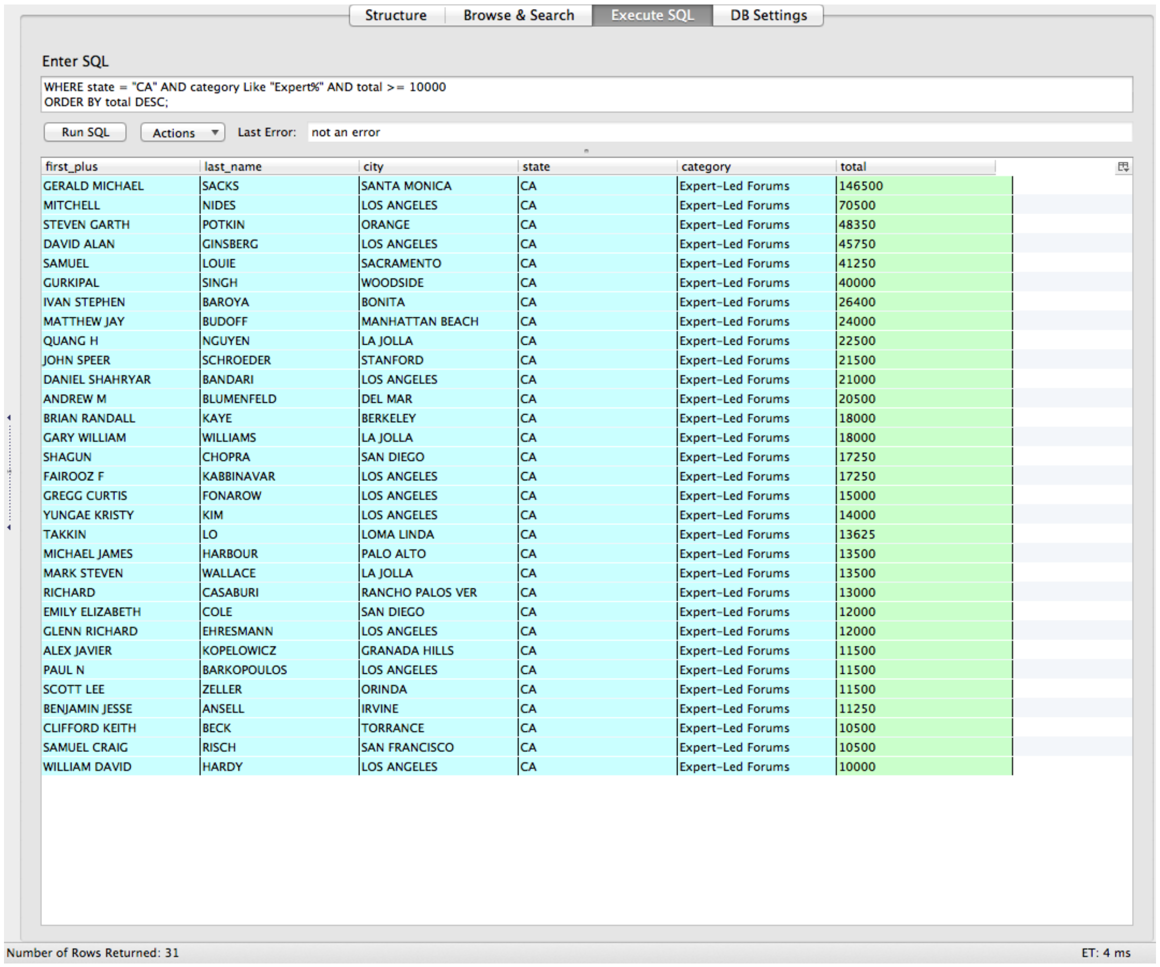
Learning SQL is very useful, because (with small variations in syntax), all databases use the same language. So in this tutorial, you won't just be learning how to use SQLite, but also starting to acquire skills that can be transferred to other database software, including Microsoft Access, PostgreSQL and MySQL.

Click on the **Execute SQL** tab and notice that **Enter SQL** box contains the statement **SELECT * FROM tablename**. Replace tablename with **pfizer**, and click **Run SQL**. That should return the entire table, because ***** is a wildcard that tells SQLite to return information from every field in a table. The query will return all 10,087 records, because we haven't asked for the data to be filtered in any way.

OK, now let's run a more useful query, filtering the data to repeat our spreadsheet task of making a list of all doctors in California who were paid $10,000 or more by Pfizer to run expert-led forums. Paste or type this query into the **Enter SQL** box:

```
SELECT first_plus, last_name, city, state, category, total
FROM pfizer
WHERE state = 'CA' AND category Like 'Expert%' AND total >=
10000
ORDER BY total DESC;
```

Click **Run SQL** and you should see the following results:



```
WHERE state = "CA" AND category Like "Expert%" AND total >= 10000
ORDER BY total DESC;
```

Last Error: not an error

first_plus	last_name	city	state	category	total
GERALD MICHAEL	SACKS	SANTA MONICA	CA	Expert-Led Forums	146500
MITCHELL	NIDES	LOS ANGELES	CA	Expert-Led Forums	70500
STEVEN GARTH	POTKIN	ORANGE	CA	Expert-Led Forums	48350
DAVID ALAN	GINSBERG	LOS ANGELES	CA	Expert-Led Forums	45750
SAMUEL	LOUIE	SACRAMENTO	CA	Expert-Led Forums	41250
GURKIPAL	SINGH	WOODSIDE	CA	Expert-Led Forums	40000
IVAN STEPHEN	BAROYA	BONITA	CA	Expert-Led Forums	26400
MATTHEW JAY	BUDOFF	MANHATTAN BEACH	CA	Expert-Led Forums	24000
QUANG H	NGUYEN	LA JOLLA	CA	Expert-Led Forums	22500
JOHN SPEER	SCHROEDER	STANFORD	CA	Expert-Led Forums	21500
DANIEL SHAHRYAR	BANDARI	LOS ANGELES	CA	Expert-Led Forums	21000
ANDREW M	BLUMENFELD	DEL MAR	CA	Expert-Led Forums	20500
BRIAN RANDALL	KAYE	BERKELEY	CA	Expert-Led Forums	18000
GARY WILLIAM	WILLIAMS	LA JOLLA	CA	Expert-Led Forums	18000
SHAGUN	CHOPRA	SAN DIEGO	CA	Expert-Led Forums	17250
FAIROOZ F	KABBINAVAR	LOS ANGELES	CA	Expert-Led Forums	17250
GREGG CURTIS	FONAROW	LOS ANGELES	CA	Expert-Led Forums	15000
YUNGAE KRISTY	KIM	LOS ANGELES	CA	Expert-Led Forums	14000
TAKKIN	LO	LOMA LINDA	CA	Expert-Led Forums	13625
MICHAEL JAMES	HARBOUR	PALO ALTO	CA	Expert-Led Forums	13500
MARK STEVEN	WALLACE	LA JOLLA	CA	Expert-Led Forums	13500
RICHARD	CASABURI	RANCHO PALOS VER	CA	Expert-Led Forums	13000
EMILY ELIZABETH	COLE	SAN DIEGO	CA	Expert-Led Forums	12000
GLENN RICHARD	EHRESMANN	LOS ANGELES	CA	Expert-Led Forums	12000
ALEX JAVIER	KOPELOWICZ	GRANADA HILLS	CA	Expert-Led Forums	11500
PAUL N	BARKOPOULOS	LOS ANGELES	CA	Expert-Led Forums	11500
SCOTT LEE	ZELLER	ORINDA	CA	Expert-Led Forums	11500
BENJAMIN JESSE	ANSELL	IRVINE	CA	Expert-Led Forums	11250
CLIFFORD KEITH	BECK	TORRANCE	CA	Expert-Led Forums	10500
SAMUEL CRAIG	RISCH	SAN FRANCISCO	CA	Expert-Led Forums	10500
WILLIAM DAVID	HARDY	LOS ANGELES	CA	Expert-Led Forums	10000

Number of Rows Returned: 31

ET: 4 ms

Let's break this query down:

SELECT first_plus, last_name, city, state, category, total
FROM pfizer
WHERE state = 'CA' AND category Like 'Expert%' AND total >= 10000
ORDER BY total DESC;

The first two lines tell SQLite to select the named fields from the pfizer table, with each field separated by a comma.

```
SELECT first_plus, last_name, city, state, category, total
FROM pfizer
WHERE state = 'CA' AND category Like 'Expert%' AND total >=
10000
ORDER BY total DESC;
```

The **WHERE** clause applies a filter to select only certain records from the table.

When filtering text fields, the search string should be put in quote marks. The second text field filter uses the operator **LIKE** to perform a fuzzy match, and is used with wildcard characters: the **%** wildcard takes the place of any number of characters, while the _ wildcard is used to represent single characters only. Here the **%** wildcard is simply saving us from having to type **Expert-Led Forums** in full, but such queries can be very useful to return data entered in slightly different ways. (**LIKE** also matches irrespective of case, whereas = requires the case to be exactly as typed.)

Our query also includes a number filter, here telling SQLite to return records only when the total is greater or equal to 10,000. Try experimenting with different operators, such as =, < (less than), and <> (not equal to).

In this query, each part of the **WHERE** statement is linked by **AND**, which ensures that records will only be returned if all the stated criteria are met. **WHERE** statements obey the same Boolean logic we used to filter in the spreadsheet tutorial; again, see what happens if you replace the first **AND** with **OR**.

```
SELECT first_plus, last_name, city, state, category, total
FROM pfizer
WHERE state = 'CA' AND category Like 'Expert%' AND total >=
10000
ORDER BY total DESC;
```

The final line of the query sorts the results in descending order by the total paid. See what happens if you remove **DESC**. The semi-colon simply marks the end of the query. See what happens if you change the end of the query to the following:

```
ORDER BY total DESC
LIMIT 20;
```

Now let's run the following query, which extends the search for doctors paid $10,000 or more for running Expert-led forums to New York, as well as California:

```
SELECT first_plus, last_name, city, state, category, total
FROM pfizer
WHERE (state = 'CA' OR state = 'NY') AND category Like 'Expert%'
AND total >= 10000
ORDER BY total DESC;
```

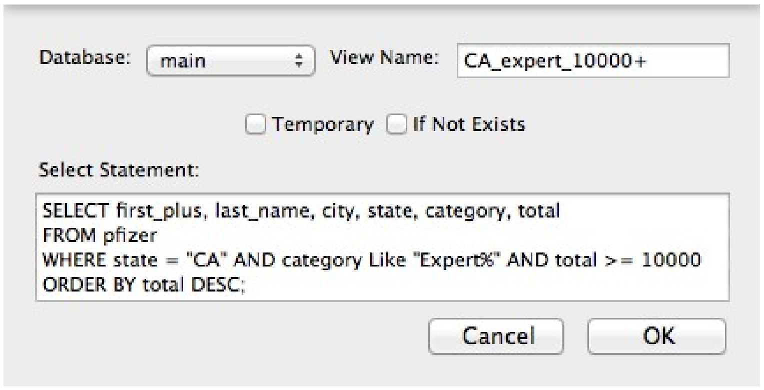
Now remove the brackets surrounding the first part of the **WHERE** clause and see if you can work out what's going on. Hint: think algebra!

By now you should be starting to get the hang of SQL, so see if you can write queries to return the same fields from the **pfizer** table, applying these filters:

1. Find the 10 highest paid doctors in California or New York, based on payments for professional advice.
2. Find the doctor from any state who was paid the most for meals.

2. Saving and exporting queries

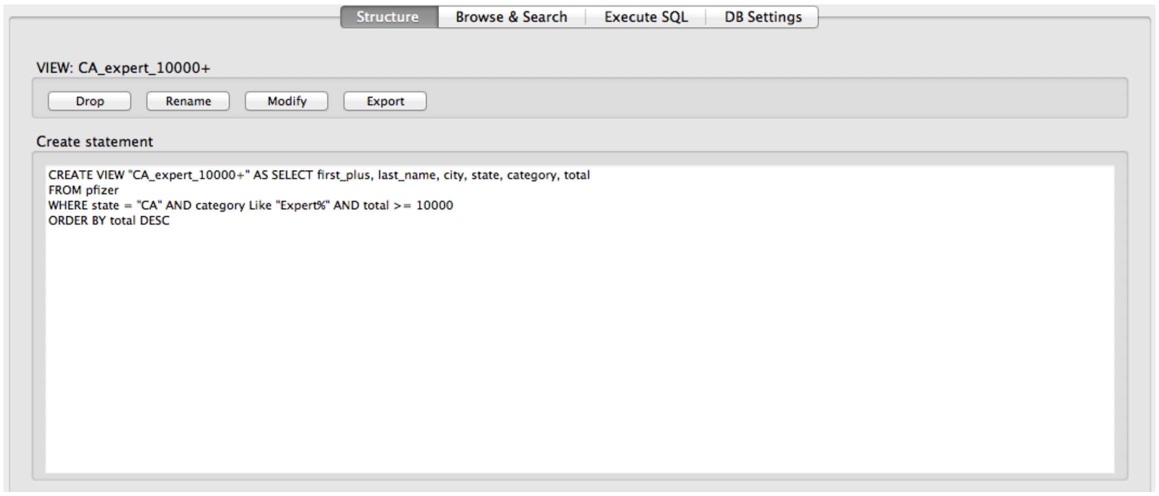
OK, let's return to our query about doctors in California paid $10,000 or more for running Expert-led forums, and save it for later use. Select **View>Create View** from the top menu, give the view a suitable name, and paste the SQL for the query into the box:



```
SELECT first_plus, last_name, city, state, category, total
FROM pfizer
WHERE state = "CA" AND category Like "Expert%" AND total >= 10000
ORDER BY total DESC;
```

Click **OK,** and at the next dialog box click **Yes**. Double click on **Views** in the left panel and select the newly created view. The results of the query appear in the **Browse & Search** tab.

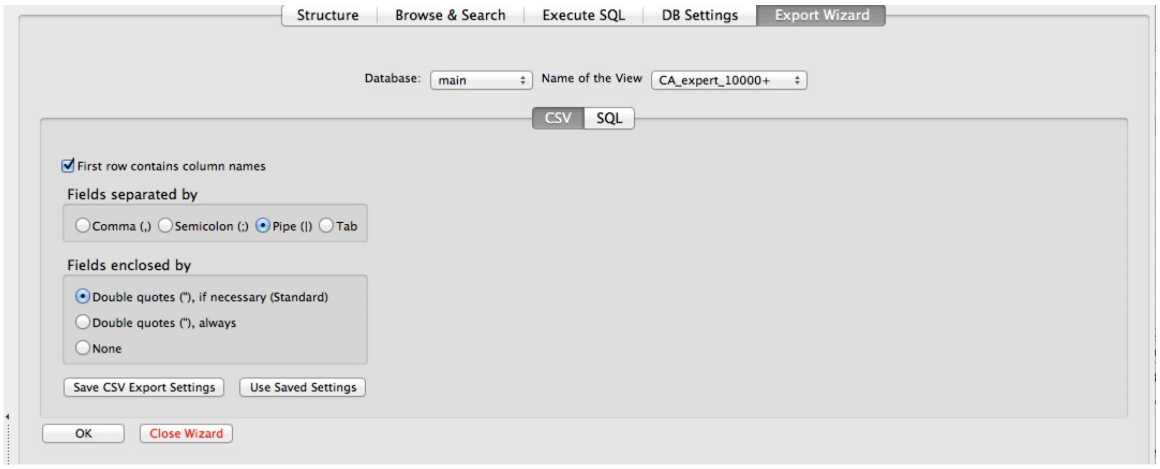
Now click on the **Structure** tab, which should look like this:



```
CREATE VIEW "CA_expert_10000+" AS SELECT first_plus, last_name, city, state, category, total
FROM pfizer
WHERE state = "CA" AND category Like "Expert%" AND total >= 10000
ORDER BY total DESC
```

By creating views, you can keep a record of the queries you have run, which is good practice in data journalism.

You may also want to export the results of your queries, so now click **Export**, and fill in the options in the wizard as follows:



Database: main
Name of the View: CA_expert_10000+

[✓] First row contains column names

Fields separated by: Comma (,) Semicolon (;) [✓ Pipe (|)] Tab

Fields enclosed by: [✓ Double quotes ("), if necessary (Standard)] Double quotes ("), always None

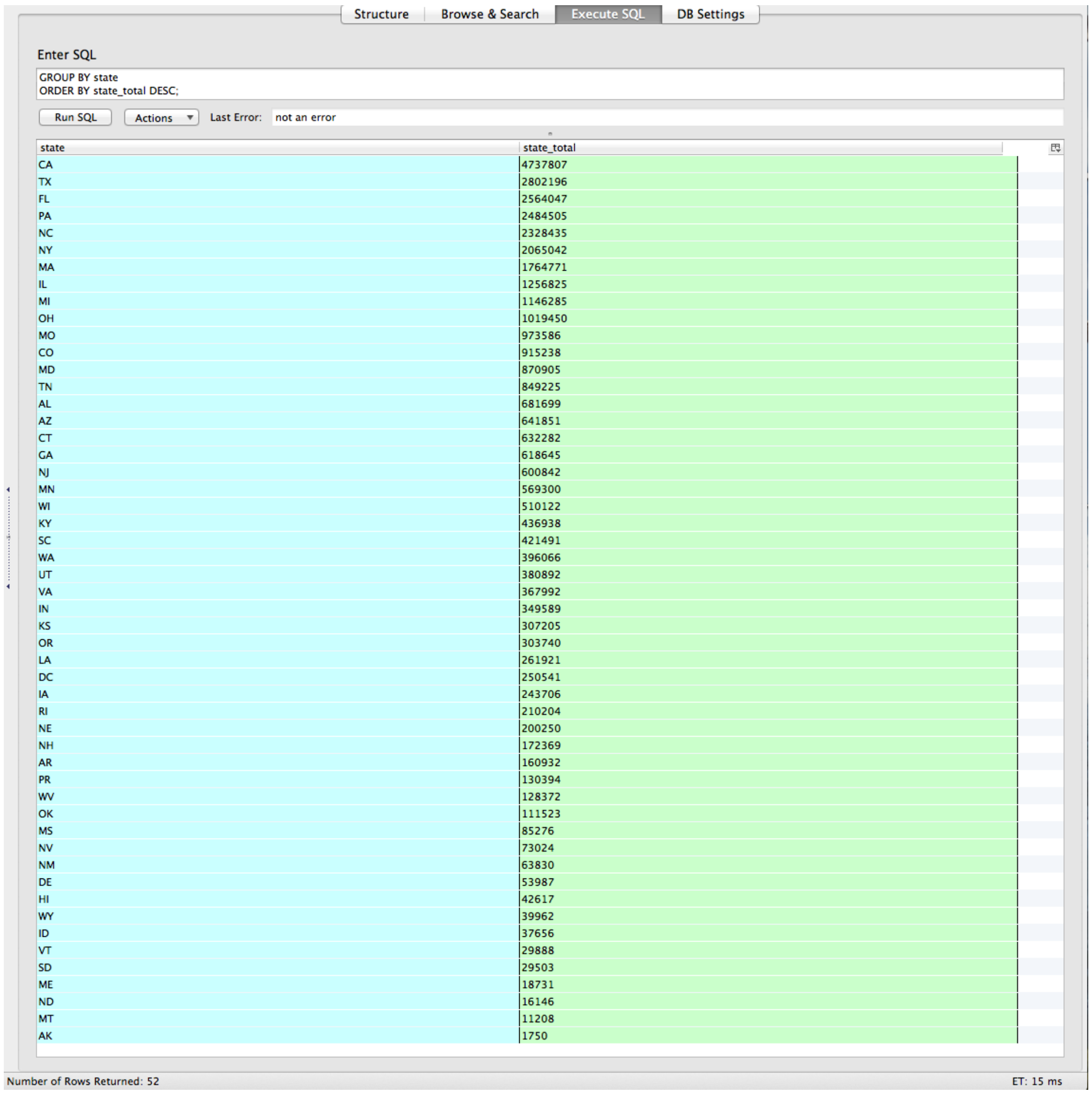
I'd recommend using the Pipe symbol (|) to separate the fields in the exported data, as it is unlikely to appear in the data itself. Click OK, and you will save the data in CSV format, a simple text file that can easily be imported into spreadsheets and other data analysis software.

3. Grouping and aggregating data

Now let's repeat our spreadsheet exercise of subtotaling all of the payments grouped by state. Select the **pfizer** table, click on the **Execute SQL** tab, and run the following query:

SELECT state, SUM(total) AS state_total
FROM pfizer
GROUP BY state
ORDER BY state_total DESC;

Click **Run SQL** and you should see the following results:



```
GROUP BY state
ORDER BY state_total DESC;
```

Last Error: not an error

state	state_total
CA	4737807
TX	2802196
FL	2564047
PA	2484505
NC	2328435
NY	2065042
MA	1764771
IL	1256825
MI	1146285
OH	1019450
MO	973586
CO	915238
MD	870905
TN	849225
AL	681699
AZ	641851
CT	632282
GA	618645
NJ	600842
MN	569300
WI	510122
KY	436938
SC	421491
WA	396066
UT	380892
VA	367992
IN	349589
KS	307205
OR	303740
LA	261921
DC	250541
IA	243706
RI	210204
NE	200250
NH	172369
AR	160932
PR	130394
WV	128372
OK	111523
MS	85276
NV	73024
NM	63830
DE	53987
HI	42617
WY	39962
ID	37656
VT	29888
SD	29503
ME	18731
ND	16146
MT	11208
AK	1750

Number of Rows Returned: 52

ET: 15 ms

Again, let's break this query down:

```
SELECT state, SUM(total) AS state_total
FROM pfizer
GROUP BY state
ORDER BY state_total DESC;
```

The first two lines return data for state and total, with the totals added up using the function **SUM** and the field renamed **AS** state_total. See what happens if you replace **SUM** with **AVG**, **MAX**, **MIN** or **COUNT**.

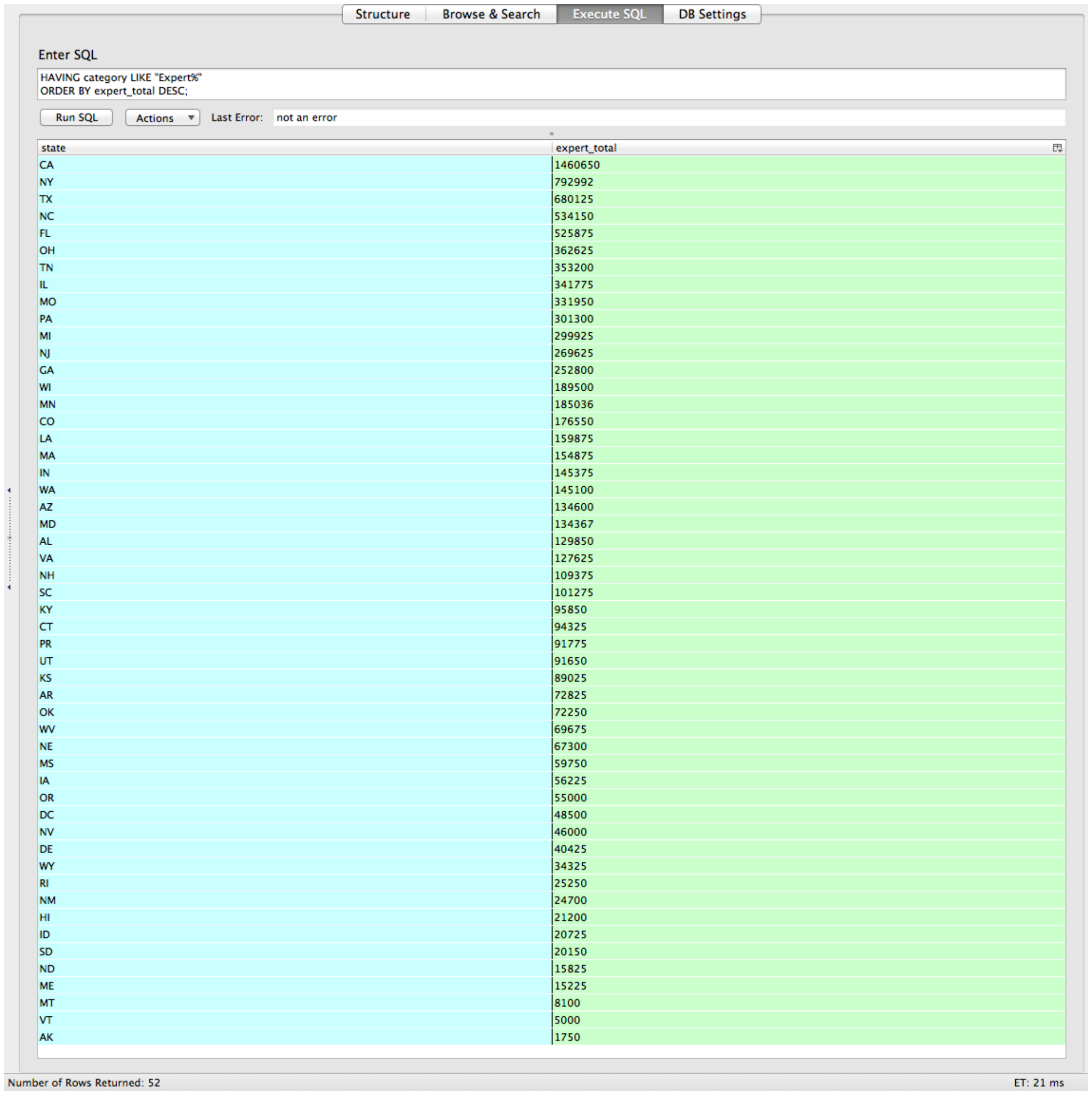
```
SELECT state, SUM(total) AS state_total
FROM pfizer
GROUP BY state
ORDER BY state_total DESC;
```

The third line is crucial, telling SQL how to group the data to calculate the subtotals. In **GROUP BY** queries like this, fields that are selected but aren't being aggregated (using **SUM**, **AVG** etc) must also appear in the **GROUP BY** clause.

Now let's total by state just for payments made for Expert-led forums, using this query:

```
SELECT state, SUM(total) AS expert_total
FROM pfizer
GROUP BY state, category
HAVING category LIKE 'Expert%'
ORDER BY expert_total DESC;
```

Click **Run SQL** and you should see the following results:



```
HAVING category LIKE "Expert%"
ORDER BY expert_total DESC;
```

Last Error: not an error

state	expert_total
CA	1460650
NY	792992
TX	680125
NC	534150
FL	525875
OH	362625
TN	353200
IL	341775
MO	331950
PA	301300
MI	299925
NJ	269625
GA	252800
WI	189500
MN	185036
CO	176550
LA	159875
MA	154875
IN	145375
WA	145100
AZ	134600
MD	134367
AL	129850
VA	127625
NH	109375
SC	101275
KY	95850
CT	94325
PR	91775
UT	91650
KS	89025
AR	72825
OK	72250
WV	69675
NE	67300
MS	59750
IA	56225
OR	55000
DC	48500
NV	46000
DE	40425
WY	34325
RI	25250
NM	24700
HI	21200
ID	20725
SD	20150
ND	15825
ME	15225
MT	8100
VT	5000
AK	1750

Number of Rows Returned: 52

ET: 21 ms

This query introduces the **HAVING** clause:

```
SELECT state, SUM(total) AS expert_total
FROM pfizer
GROUP BY state, category
HAVING category LIKE 'Expert%'
ORDER BY expert_total DESC;
```

HAVING does the same filtering job as **WHERE** for a **GROUP BY** query; fields that appear in the **HAVING** clause must also appear under **GROUP BY**.

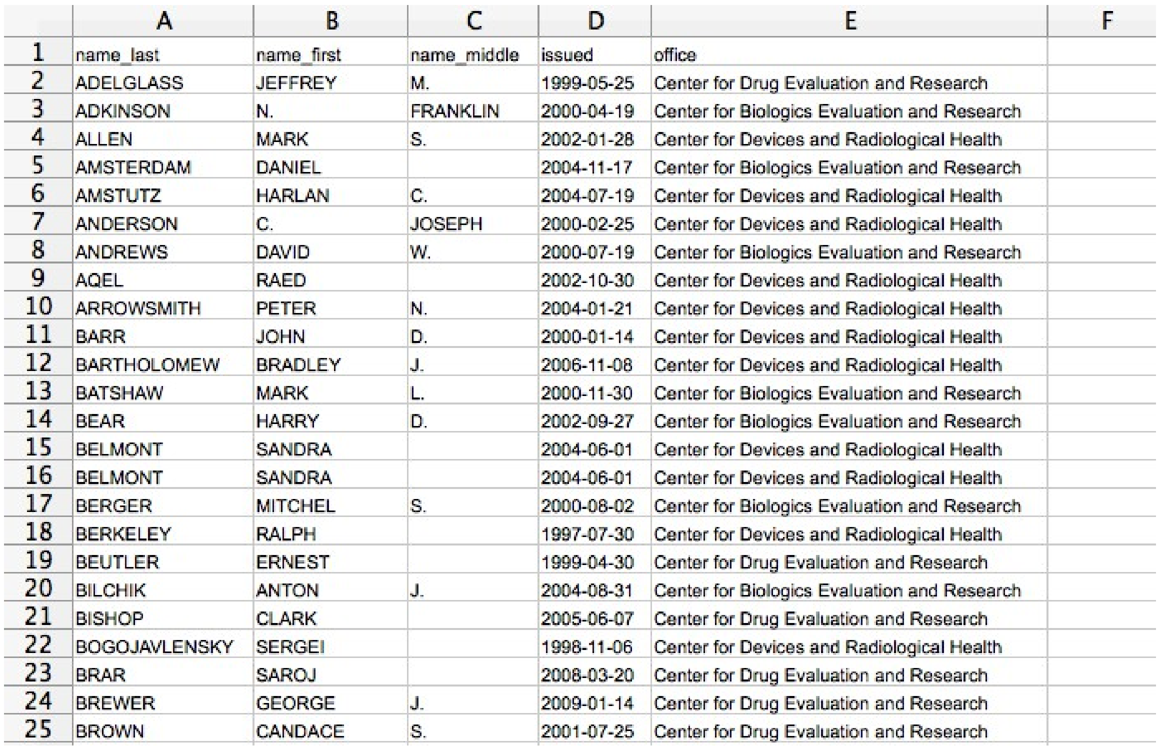
We can also aggregate data by more than one field at a time. For example, this query performs the same aggregation as in the pivot table example from the spreadsheet tutorial, although it does not return a pivot table view:

```
SELECT state, category, SUM(total) AS subtotal
FROM Pfizer
GROUP BY state, category;
```

Using multiple data tables

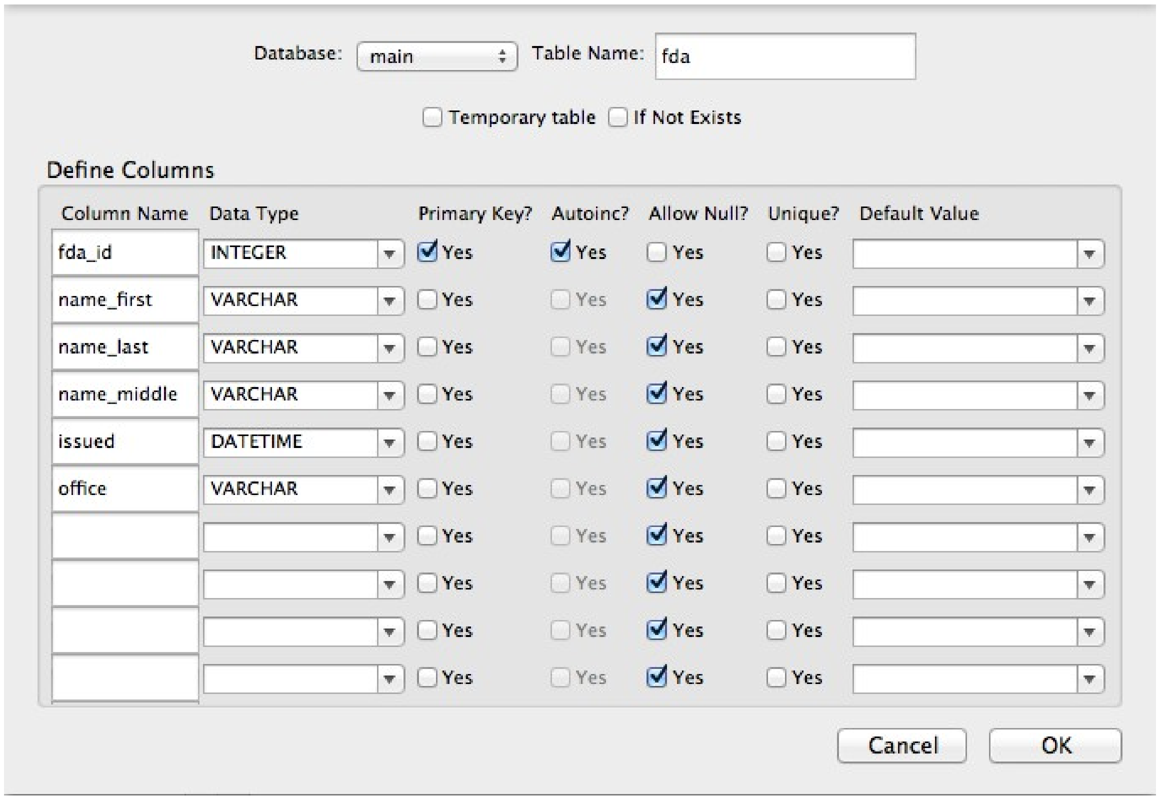
1. Creating a new data table

We're going to import the data in the file **fda_warning.csv,** available online here, which details warning letters sent by the Food and Drug Administration to doctors because of problems with their conduct of clinical research. The data is in a CSV file with Pipe separators. The first few rows look like this when imported into a spreadsheet:



	A	B	C	D	E	F
1	name_last	name_first	name_middle	issued	office	
2	ADELGLASS	JEFFREY	M.	1999-05-25	Center for Drug Evaluation and Research	
3	ADKINSON	N.	FRANKLIN	2000-04-19	Center for Biologics Evaluation and Research	
4	ALLEN	MARK	S.	2002-01-28	Center for Devices and Radiological Health	
5	AMSTERDAM	DANIEL		2004-11-17	Center for Biologics Evaluation and Research	
6	AMSTUTZ	HARLAN	C.	2004-07-19	Center for Devices and Radiological Health	
7	ANDERSON	C.	JOSEPH	2000-02-25	Center for Devices and Radiological Health	
8	ANDREWS	DAVID	W.	2000-07-19	Center for Biologics Evaluation and Research	
9	AQEL	RAED		2002-10-30	Center for Devices and Radiological Health	
10	ARROWSMITH	PETER	N.	2004-01-21	Center for Devices and Radiological Health	
11	BARR	JOHN	D.	2000-01-14	Center for Devices and Radiological Health	
12	BARTHOLOMEW	BRADLEY	J.	2006-11-08	Center for Devices and Radiological Health	
13	BATSHAW	MARK	L.	2000-11-30	Center for Biologics Evaluation and Research	
14	BEAR	HARRY	D.	2002-09-27	Center for Biologics Evaluation and Research	
15	BELMONT	SANDRA		2004-06-01	Center for Devices and Radiological Health	
16	BELMONT	SANDRA		2004-06-01	Center for Devices and Radiological Health	
17	BERGER	MITCHEL	S.	2000-08-02	Center for Biologics Evaluation and Research	
18	BERKELEY	RALPH		1997-07-30	Center for Devices and Radiological Health	
19	BEUTLER	ERNEST		1999-04-30	Center for Drug Evaluation and Research	
20	BILCHIK	ANTON	J.	2004-08-31	Center for Biologics Evaluation and Research	
21	BISHOP	CLARK		2005-06-07	Center for Drug Evaluation and Research	
22	BOGOJAVLENSKY	SERGEI		1998-11-06	Center for Devices and Radiological Health	
23	BRAR	SAROJ		2008-03-20	Center for Drug Evaluation and Research	
24	BREWER	GEORGE	J.	2009-01-14	Center for Drug Evaluation and Research	
25	BROWN	CANDACE	S.	2001-07-25	Center for Drug Evaluation and Research	

First we need to create a table into which to import the data. Select **Table>Create Table**, and fill in the dialog box as follows:



Database: main Table Name: fda

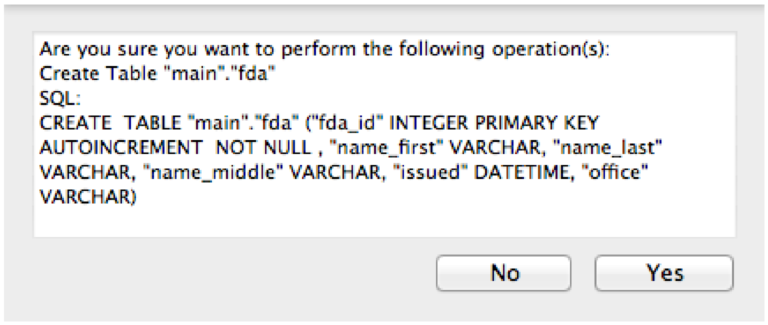
Temporary table If Not Exists

Define Columns

Column Name	Data Type	Primary Key?	Autoinc?	Allow Null?	Unique?	Default Value
fda_id	INTEGER	[✓ Yes]	[✓ Yes]	Yes	Yes	
name_first	VARCHAR	Yes	Yes	[✓ Yes]	Yes	
name_last	VARCHAR	Yes	Yes	[✓ Yes]	Yes	
name_middle	VARCHAR	Yes	Yes	[✓ Yes]	Yes	
issued	DATETIME	Yes	Yes	[✓ Yes]	Yes	
office	VARCHAR	Yes	Yes	[✓ Yes]	Yes	
		Yes	Yes	[✓ Yes]	Yes	
		Yes	Yes	[✓ Yes]	Yes	
		Yes	Yes	[✓ Yes]	Yes	
		Yes	Yes	[✓ Yes]	Yes	

The first field will be automatically created when the data is imported, giving a unique ID number to each record. For this field, make sure to select **INTEGER** for **Data Type**, and to check the **Primary Key** and **Autoinc** boxes. The other column names match those in the data; **VARCHAR** means a text field of varying length; **DATETIME** is used for the issued date.

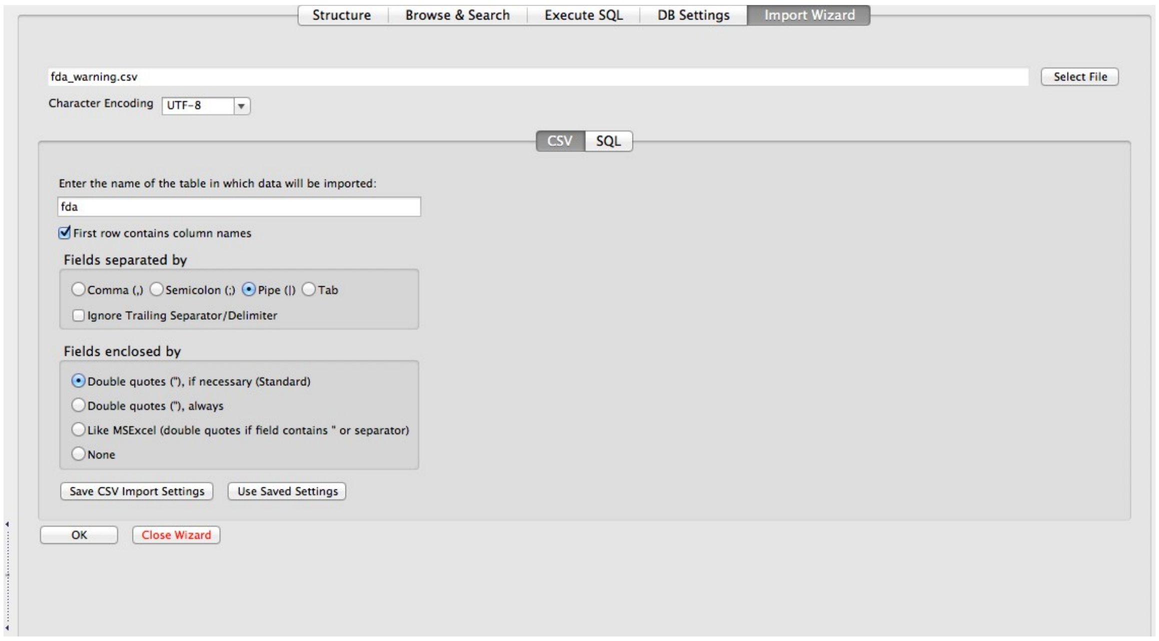
Click **Yes** at the next dialog box, which shows the SQL code being used to create the table:



```
Are you sure you want to perform the following operation(s):
Create Table "main"."fda"
SQL:
CREATE  TABLE "main"."fda" ("fda_id" INTEGER PRIMARY KEY
AUTOINCREMENT  NOT NULL , "name_first" VARCHAR, "name_last"
VARCHAR, "name_middle" VARCHAR, "issued" DATETIME, "office"
VARCHAR)
```

Now we can import the data, by clicking the **Import** icon:

Fill in the **Import Wizard** as follows, and select **OK** at the subsequent prompts:



fda_warning.csv

Character Encoding: UTF-8

Enter the name of the table in which data will be imported: fda

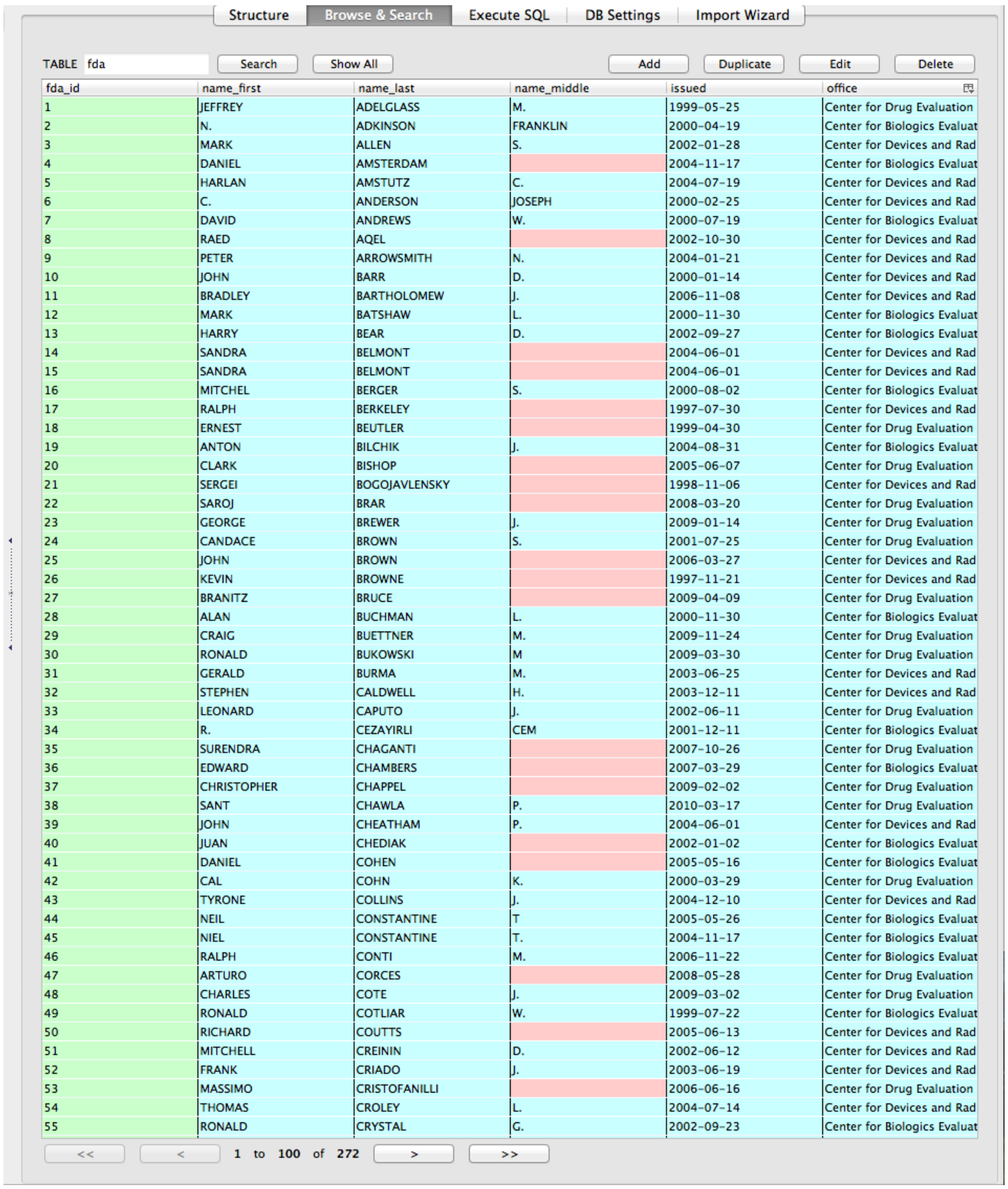
[✓] First row contains column names

Fields separated by: Comma (,) Semicolon (;) [✓ Pipe (|)] Tab

[ ] Ignore Trailing Separator/Delimiter

Fields enclosed by:
- [✓ Double quotes ("), if necessary (Standard)]
- Double quotes ("), always
- Like MSExcel (double quotes if field contains " or separator)
- None

With the new **fda** table selected in the left panel, select the **Browse & Search** tab to view the imported data:



fda_id	name_first	name_last	name_middle	issued	office
1	JEFFREY	ADELGLASS	M.	1999-05-25	Center for Drug Evaluation
2	N.	ADKINSON	FRANKLIN	2000-04-19	Center for Biologics Evaluat
3	MARK	ALLEN	S.	2002-01-28	Center for Devices and Rad
4	DANIEL	AMSTERDAM	🟥	2004-11-17	Center for Biologics Evaluat
5	HARLAN	AMSTUTZ	C.	2004-07-19	Center for Devices and Rad
6	C.	ANDERSON	JOSEPH	2000-02-25	Center for Devices and Rad
7	DAVID	ANDREWS	W.	2000-07-19	Center for Biologics Evaluat
8	RAED	AQEL	🟥	2002-10-30	Center for Devices and Rad
9	PETER	ARROWSMITH	N.	2004-01-21	Center for Devices and Rad
10	JOHN	BARR	D.	2000-01-14	Center for Devices and Rad
11	BRADLEY	BARTHOLOMEW	J.	2006-11-08	Center for Devices and Rad
12	MARK	BATSHAW	L.	2000-11-30	Center for Biologics Evaluat
13	HARRY	BEAR	D.	2002-09-27	Center for Biologics Evaluat
14	SANDRA	BELMONT	🟥	2004-06-01	Center for Devices and Rad
15	SANDRA	BELMONT	🟥	2004-06-01	Center for Devices and Rad
16	MITCHEL	BERGER	S.	2000-08-02	Center for Biologics Evaluat
17	RALPH	BERKELEY	🟥	1997-07-30	Center for Devices and Rad
18	ERNEST	BEUTLER	🟥	1999-04-30	Center for Drug Evaluation
19	ANTON	BILCHIK	J.	2004-08-31	Center for Biologics Evaluat
20	CLARK	BISHOP	🟥	2005-06-07	Center for Drug Evaluation
21	SERGEI	BOGOJAVLENSKY	🟥	1998-11-06	Center for Devices and Rad
22	SAROJ	BRAR	🟥	2008-03-20	Center for Drug Evaluation
23	GEORGE	BREWER	J.	2009-01-14	Center for Drug Evaluation
24	CANDACE	BROWN	S.	2001-07-25	Center for Drug Evaluation
25	JOHN	BROWN	🟥	2006-03-27	Center for Devices and Rad
26	KEVIN	BROWNE	🟥	1997-11-21	Center for Devices and Rad
27	BRANITZ	BRUCE	🟥	2009-04-09	Center for Drug Evaluation
28	ALAN	BUCHMAN	L.	2000-11-30	Center for Biologics Evaluat
29	CRAIG	BUETTNER	M.	2009-11-24	Center for Drug Evaluation
30	RONALD	BUKOWSKI	M	2009-03-30	Center for Drug Evaluation
31	GERALD	BURMA	M.	2003-06-25	Center for Devices and Rad
32	STEPHEN	CALDWELL	H.	2003-12-11	Center for Devices and Rad
33	LEONARD	CAPUTO	J.	2002-06-11	Center for Drug Evaluation
34	R.	CEZAYIRLI	CEM	2001-12-11	Center for Biologics Evaluat
35	SURENDRA	CHAGANTI	🟥	2007-10-26	Center for Drug Evaluation
36	EDWARD	CHAMBERS	🟥	2007-03-29	Center for Biologics Evaluat
37	CHRISTOPHER	CHAPPEL	🟥	2009-02-02	Center for Drug Evaluation
38	SANT	CHAWLA	P.	2010-03-17	Center for Drug Evaluation
39	JOHN	CHEATHAM	P.	2004-06-01	Center for Devices and Rad
40	JUAN	CHEDIAK	🟥	2002-01-02	Center for Biologics Evaluat
41	DANIEL	COHEN	🟥	2005-05-16	Center for Biologics Evaluat
42	CAL	COHN	K.	2000-03-29	Center for Drug Evaluation
43	TYRONE	COLLINS	J.	2004-12-10	Center for Devices and Rad
44	NEIL	CONSTANTINE	T	2005-05-26	Center for Biologics Evaluat
45	NIEL	CONSTANTINE	T.	2004-11-17	Center for Biologics Evaluat
46	RALPH	CONTI	M.	2006-11-22	Center for Biologics Evaluat
47	ARTURO	CORCES	🟥	2008-05-28	Center for Drug Evaluation
48	CHARLES	COTE	J.	2009-03-02	Center for Drug Evaluation
49	RONALD	COTLIAR	W.	1999-07-22	Center for Biologics Evaluat
50	RICHARD	COUTTS	🟥	2005-06-13	Center for Devices and Rad
51	MITCHELL	CREININ	D.	2002-06-12	Center for Devices and Rad
52	FRANK	CRIADO	J.	2003-06-19	Center for Devices and Rad
53	MASSIMO	CRISTOFANILLI	🟥	2006-06-16	Center for Drug Evaluation
54	THOMAS	CROLEY	L.	2004-07-14	Center for Devices and Rad
55	RONALD	CRYSTAL	G.	2002-09-23	Center for Biologics Evaluat

<< < 1 to 100 of 272 > >>

Notice that empty values, called **NULLS**, are color-coded in pink. The date values are colored the same as text, and when used in queries should be put in quote marks, as for text.

For instance, this query returns all records from the **fda** table with issue dates from 2005 onwards:

```
SELECT *
FROM fda
WHERE issued >= '2005-01-01'
ORDER BY issued;
```

2. Querying across joined data tables

Now we're going to create a query across the two data tables, so we select doctors paid by Pfizer to run Expert-led forums who had also received a warning letter from the FDA for problems with their conduct of clinical research.

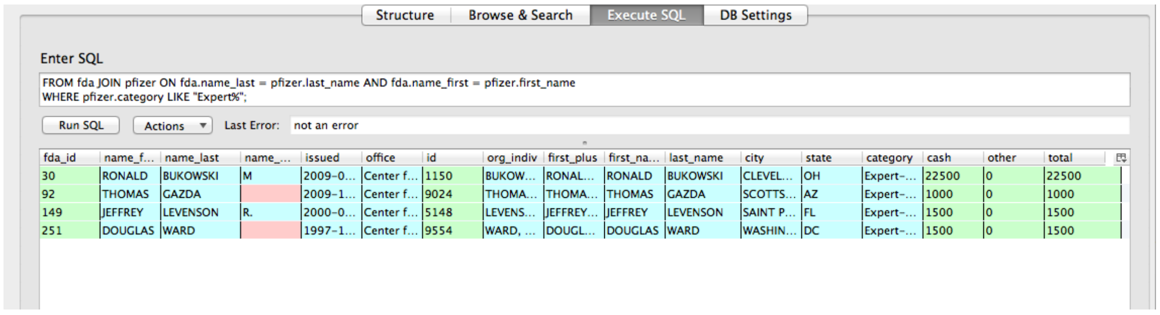
To find doctors who may be the same individual, we need to match them by both first and last name. Here is how to achieve that using SQL:

```
SELECT *
FROM fda JOIN pfizer ON fda.name_last = pfizer.last_name AND
fda.name_first = pfizer.first_name
WHERE pfizer.category LIKE 'Expert%';
```

Notice that where there is more than one table, both the table and the field should be specified, separated by a period.

Take some time to understand the logic of the highlighted **FROM** clause, which performs a **JOIN** linking the two tables, **ON** the fields specified.

This query should return the following results:



```
FROM fda JOIN pfizer ON fda.name_last = pfizer.last_name AND fda.name_first = pfizer.first_name
WHERE pfizer.category LIKE "Expert%";
```

Last Error: not an error

fda_id	name_f...	name_last	name_...	issued	office	id	org_indiv	first_plus	first_na...	last_name	city	state	category	cash	other	total
30	RONALD	BUKOWSKI	M	2009-0...	Center f...	1150	BUKOW...	RONAL...	RONALD	BUKOWSKI	CLEVEL...	OH	Expert-...	22500	0	22500
92	THOMAS	GAZDA		2009-1...	Center f...	9024	THOMA...	THOMA...	THOMAS	GAZDA	SCOTTS...	AZ	Expert-...	1000	0	1000
149	JEFFREY	LEVENSON	R.	2000-0...	Center f...	5148	LEVENS...	JEFFREY...	JEFFREY	LEVENSON	SAINT P...	FL	Expert-...	1500	0	1500
251	DOUGLAS	WARD		1997-1...	Center f...	9554	WARD, ...	DOUGL...	DOUGLAS	WARD	WASHIN...	DC	Expert-...	1500	0	1500

This tutorial will get you started with SQL and SQLite, but there is much more to learn. Here is a reference for SQL, as understood by SQLite.