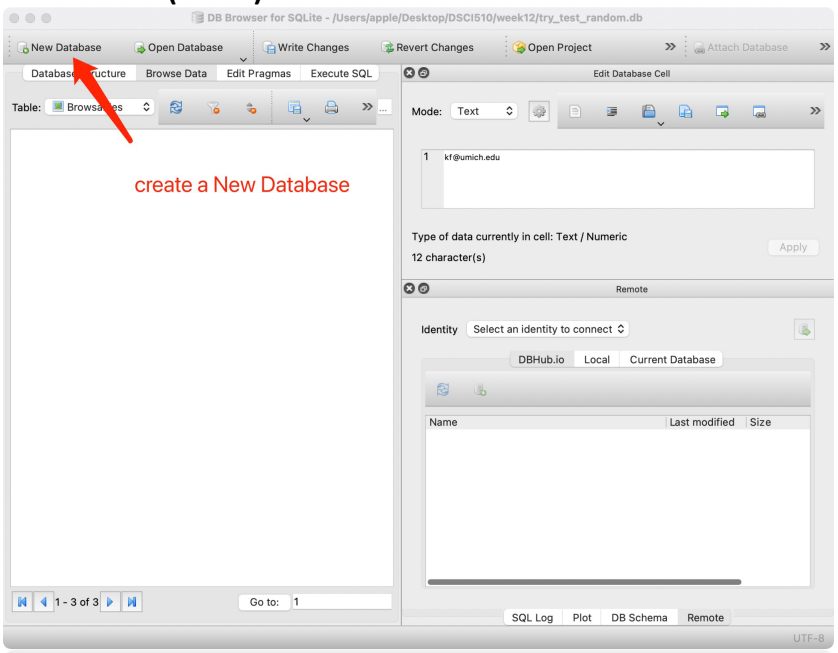
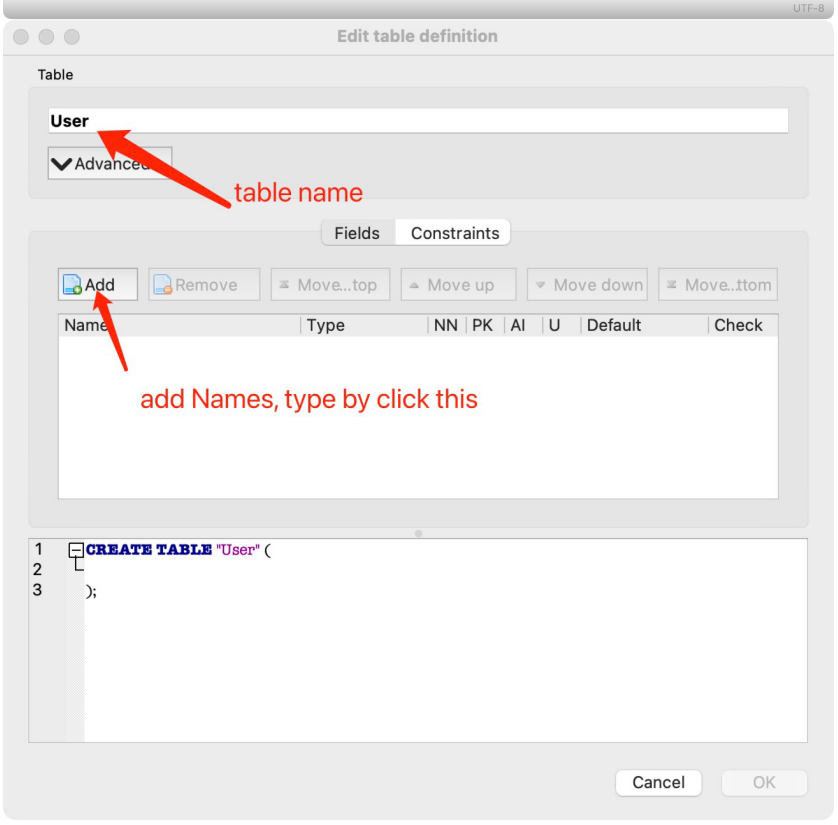
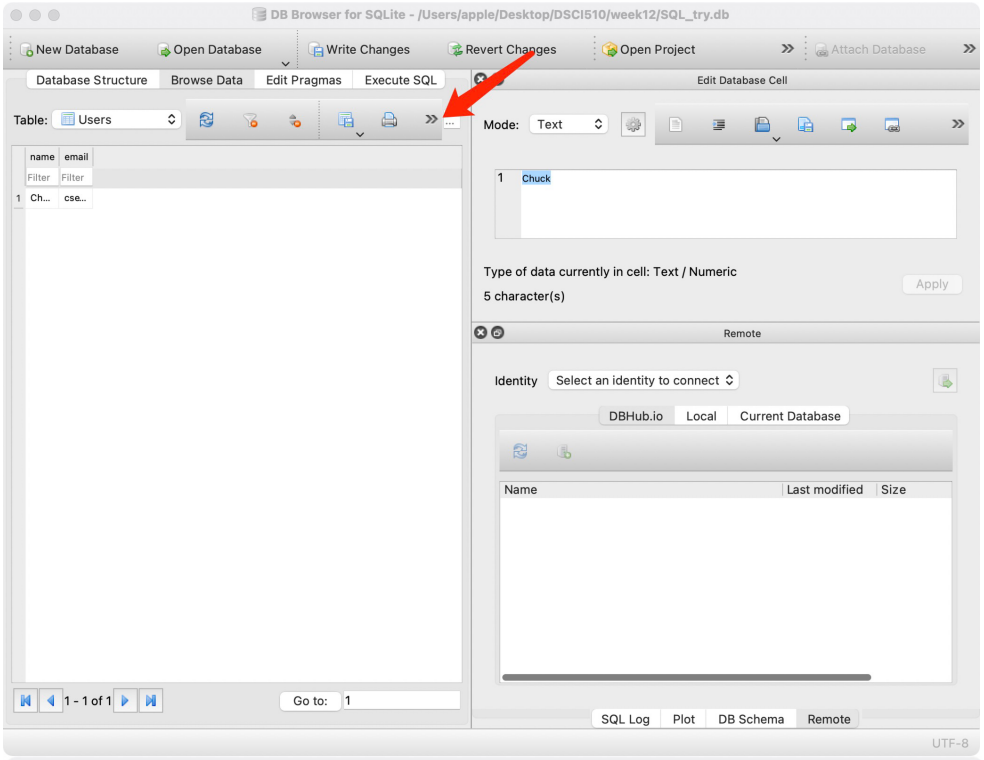
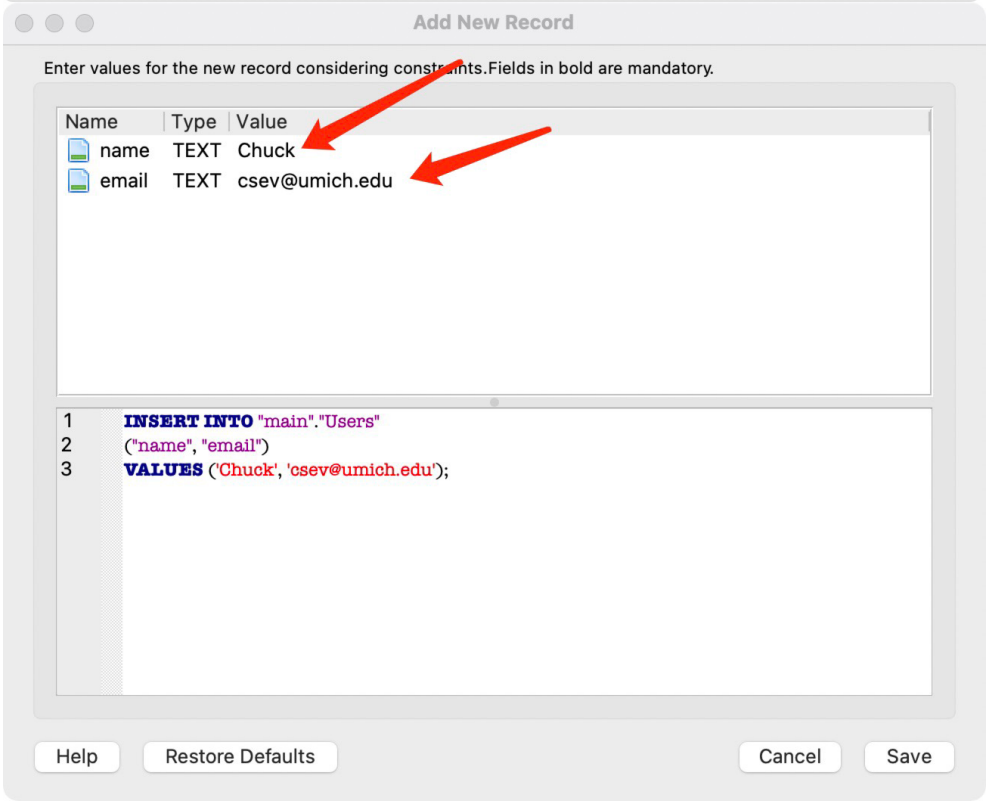
SQL: (use sqlite, Download and Install: http://sqlitebrowser.org/)
Structured Query Language is the language we use to issue commands to the database
-Create data(insert):





Then insert the values:





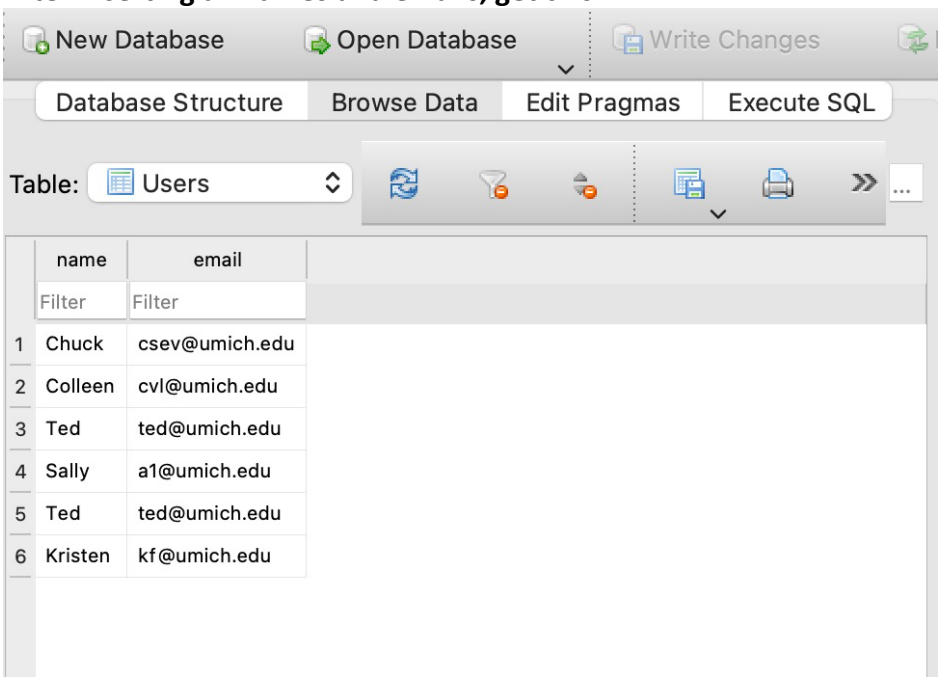
Some Steps to Create SQL Tables

CREATE TABLE "Users" ("name" TEXT, "email" TEXT)

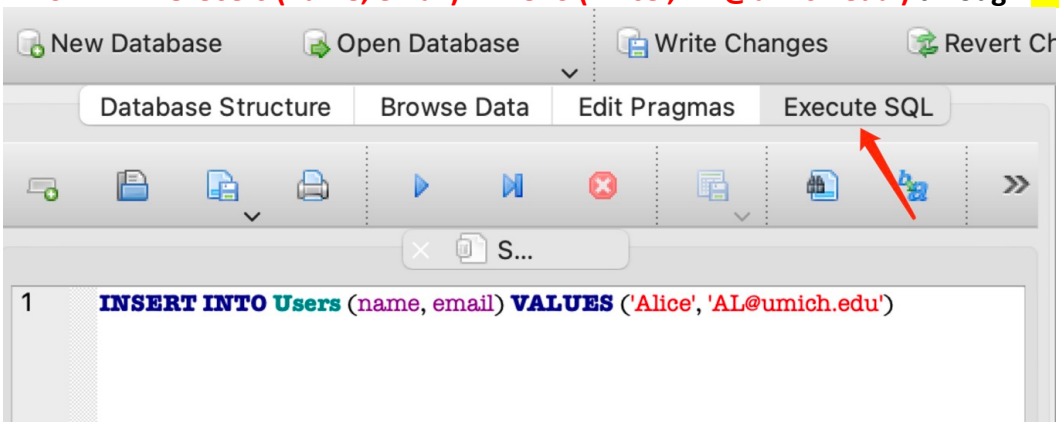
INSERT INTO Users (name, email) VALUES ('Chuck', 'csev@umich.edu')

INSERT INTO Users (name, email) VALUES ('Colleen', 'cvl@umich.edu')
INSERT INTO Users (name, email) VALUES ('Ted', 'ted@umich.edu')
INSERT INTO Users (name, email) VALUES ('Sally', 'a1@umich.edu')
INSERT INTO Users (name, email) VALUES ('Ted', 'ted@umich.edu')
INSERT INTO Users (name, email) VALUES ('Kristen', 'kf@umich.edu')

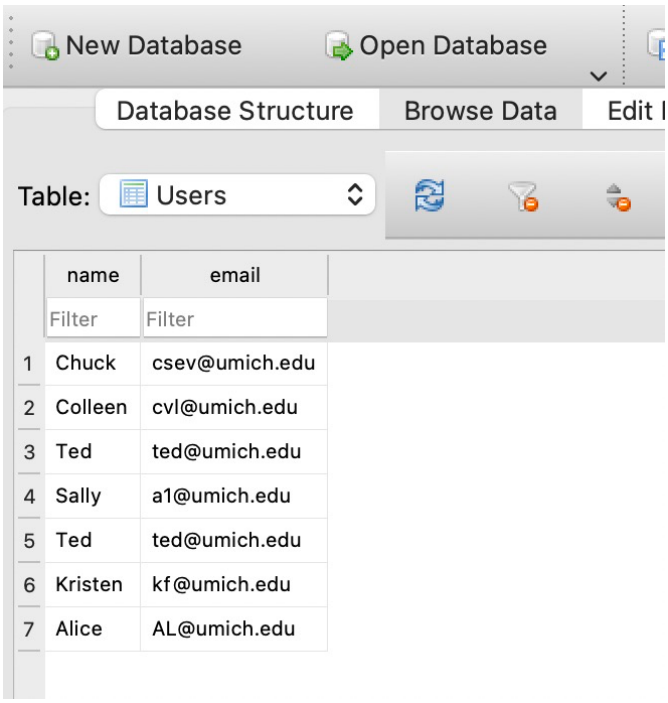
After inserting all names and emails, get this:



Also, we could insert values through execute SQL by typing
: INSERT INTO Users (name, email) VALUES ('Alice', 'AL@umich.edu') through Execute SQL

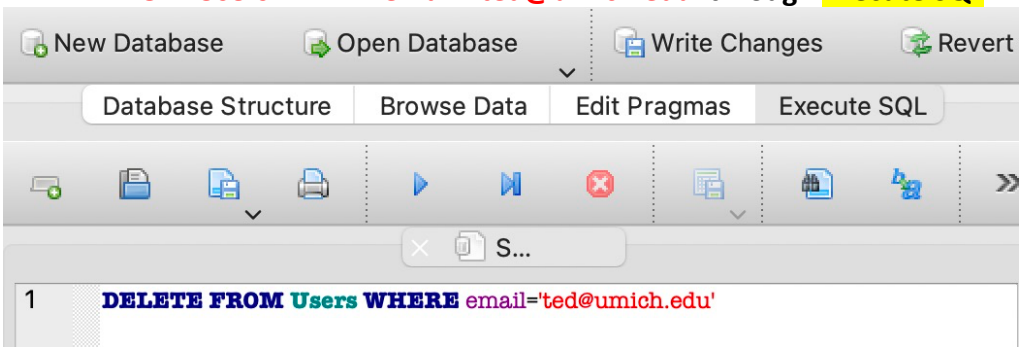


Now the table looks like(from Browse Data):

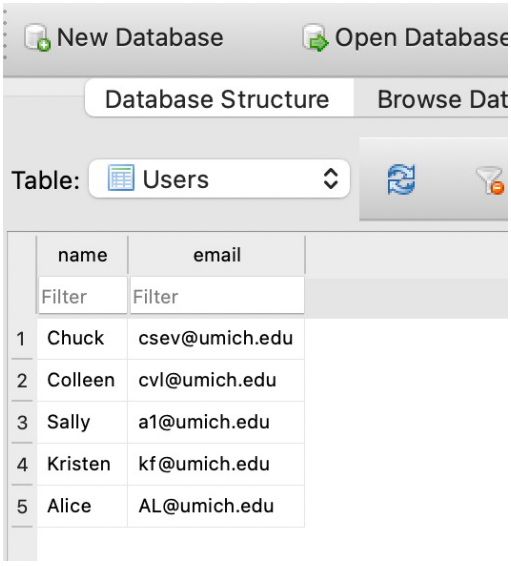


	name	email
	Filter	Filter
1	Chuck	csev@umich.edu
2	Colleen	cvl@umich.edu
3	Ted	ted@umich.edu
4	Sally	a1@umich.edu
5	Ted	ted@umich.edu
6	Kristen	kf@umich.edu
7	Alice	AL@umich.edu

Delete: we could delete a row in a table based on selection criteria by typing:
DELETE FROM Users WHERE email='ted@umich.edu' through Execute SQL

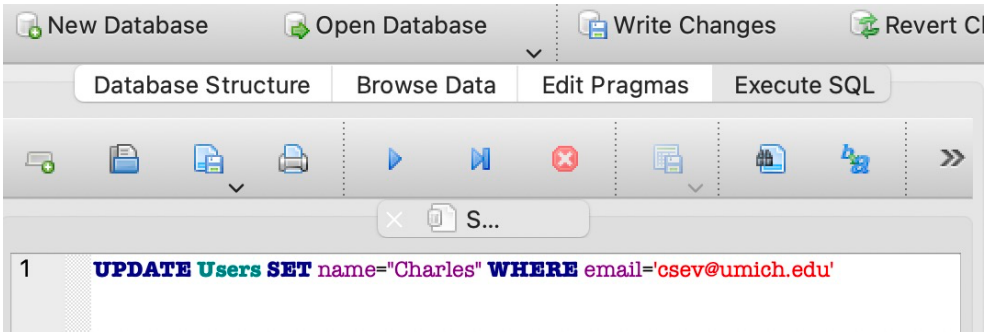


Now the table looks like:

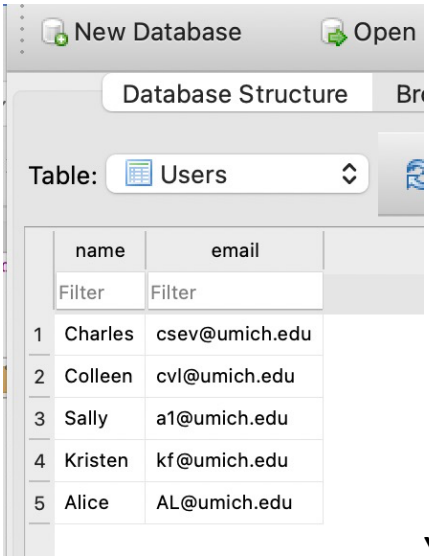


	name	email
	Filter	Filter
1	Chuck	csev@umich.edu
2	Colleen	cvl@umich.edu
3	Sally	a1@umich.edu
4	Kristen	kf@umich.edu
5	Alice	AL@umich.edu

Update: Allow the updating of a field with a where clause by typing:
UPDATE Users SET name="Charles" WHERE email='csev@umich.edu' through Execute SQL

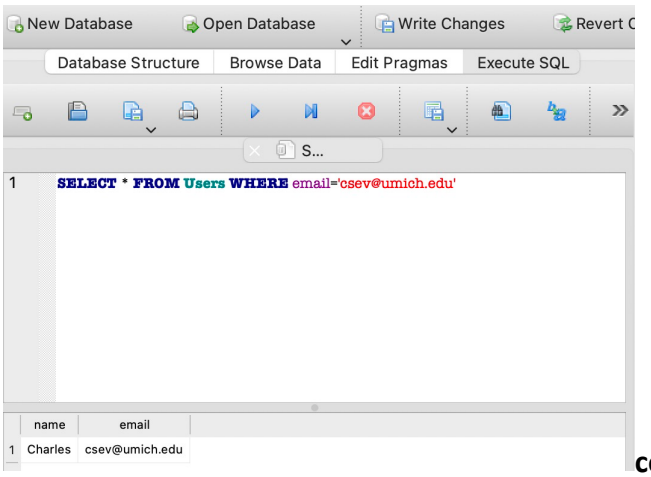


Now the table looks like:



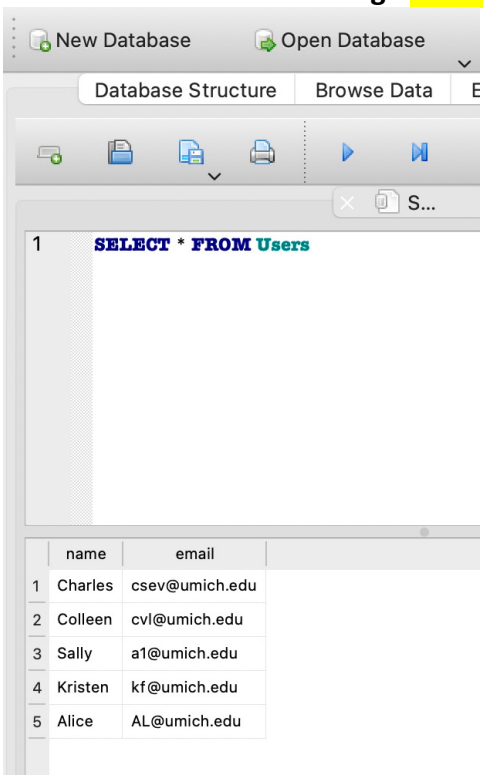
You can see that Chuck changed into Charles

Retrieving Records: Select
The select statement retrieves a group of records -you can either retrieve all the records or a subset of the records with a WHERE clause by typing:
SELECT * FROM Users WHERE email='csev@umich.edu' through Execute SQL:

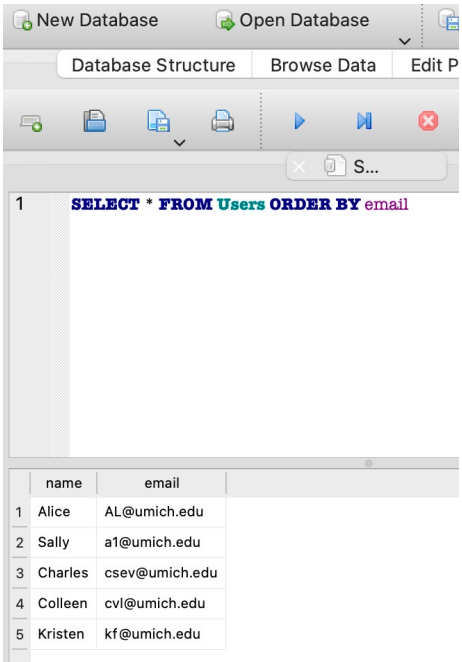


could see under there is a table that selected

SELECT * FROM Users through Execute SQL:



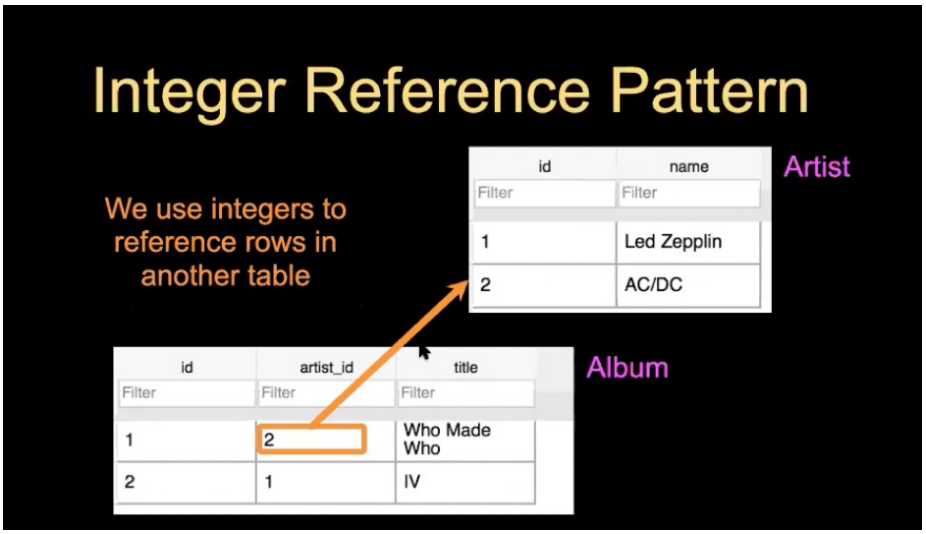
SELECT * FROM Users ORDER BY email through Execute SQL:



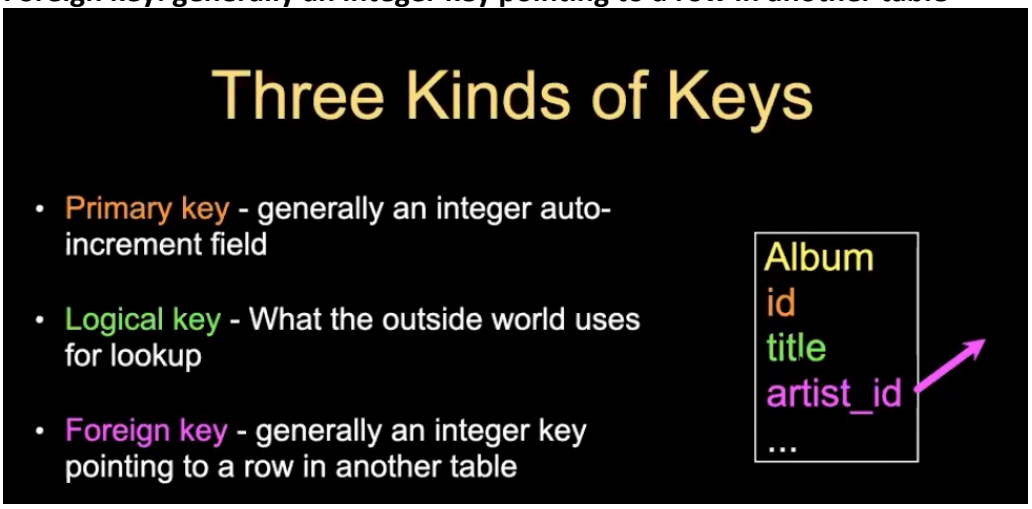
SELECT * FROM Users ORDER BY name DESC through Execute SQL:

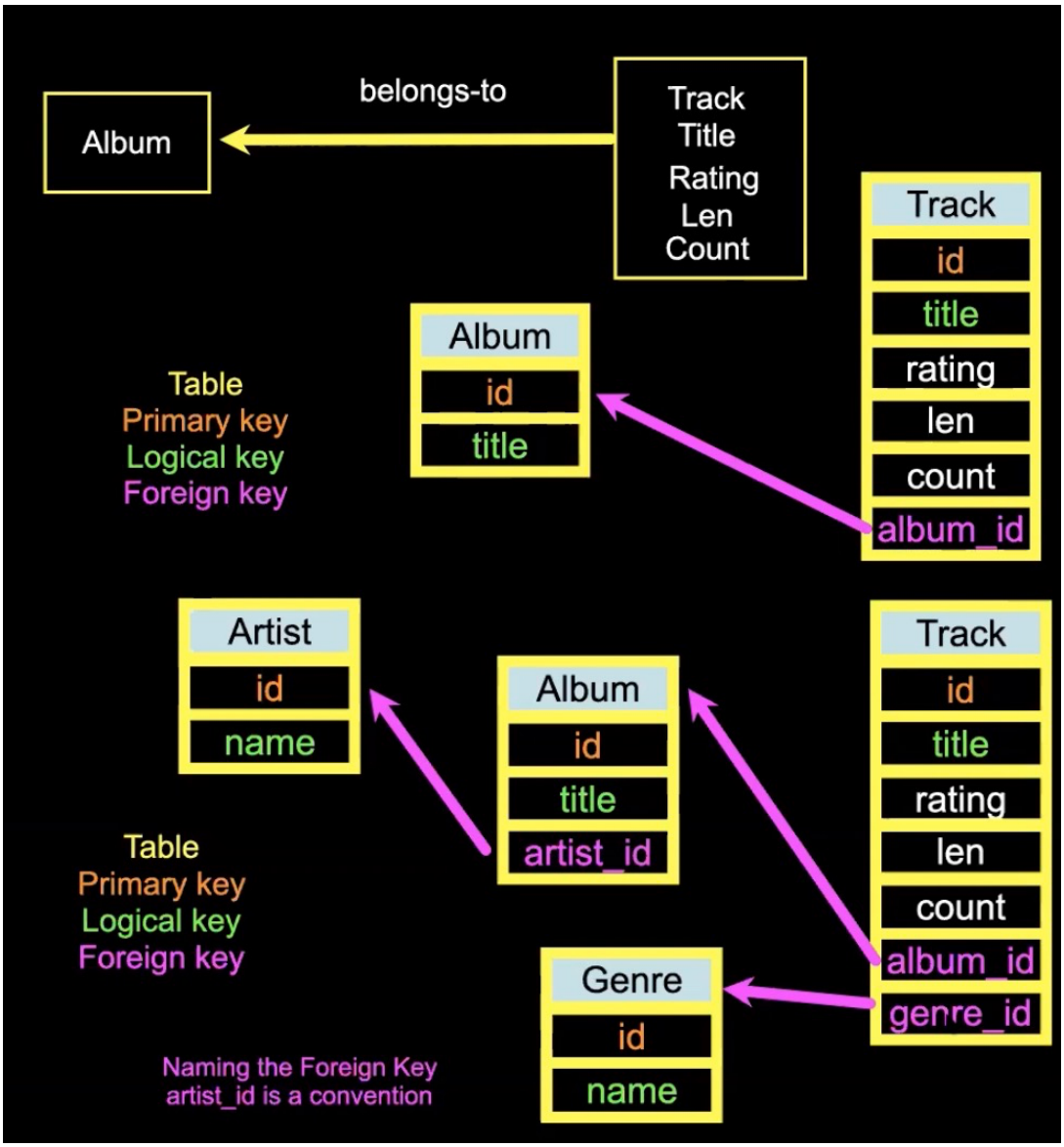
```
1  SELECT * FROM Users ORDER BY name DESC
```

	name	email
1	Sally	a1@umich.edu
2	Kristen	kf@umich.edu
3	Colleen	cvl@umich.edu
4	Charles	csev@umich.edu
5	Alice	AL@umich.edu



Primary key: generally an integer auto-increment field
Logical key: what the outside world uses for lookup
Foreign key: generally an integer key pointing to a row in another table





Create multiple tables:

Some Steps to Create multiple SQL Tables:

Multi-Table SQL:

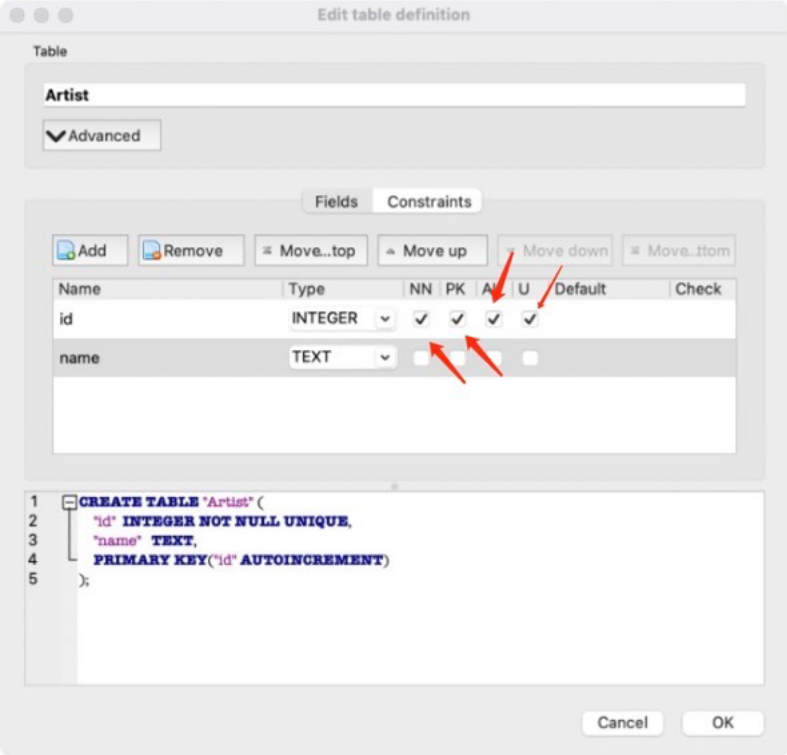
```
CREATE TABLE "Artist" (
    "id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL UNIQUE,
    "name" TEXT)

CREATE TABLE "Album" (
    "id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL UNIQUE,
    artist_id INTEGER,
```

"title" TEXT)

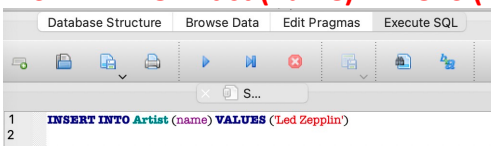
CREATE TABLE "Genre" (
"id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL UNIQUE,
"name" TEXT)

CREATE TABLE "Track" (
"id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL UNIQUE,
album_id INTEGER, genre_id INTEGER, len INTEGER, rating INTEGER,
"title" TEXT, "count" INTEGER)

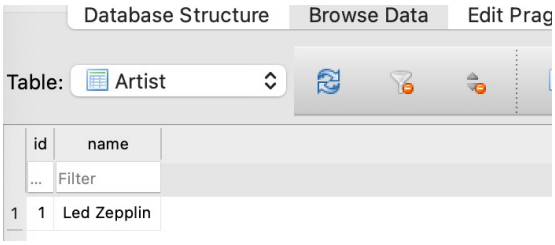
CREATE TABLE "Artist" (
"id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL UNIQUE,
"name" TEXT)



Create others follow above text…
Then insert
INSERT INTO Artist (name) VALUES ('Led Zepplin') through Execute SQL:

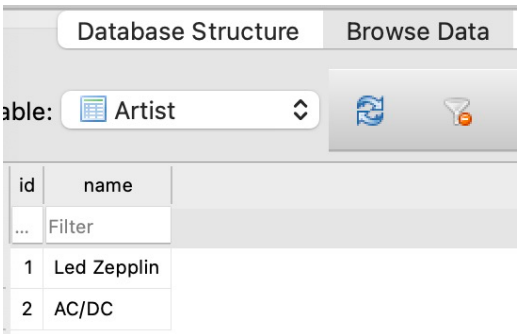


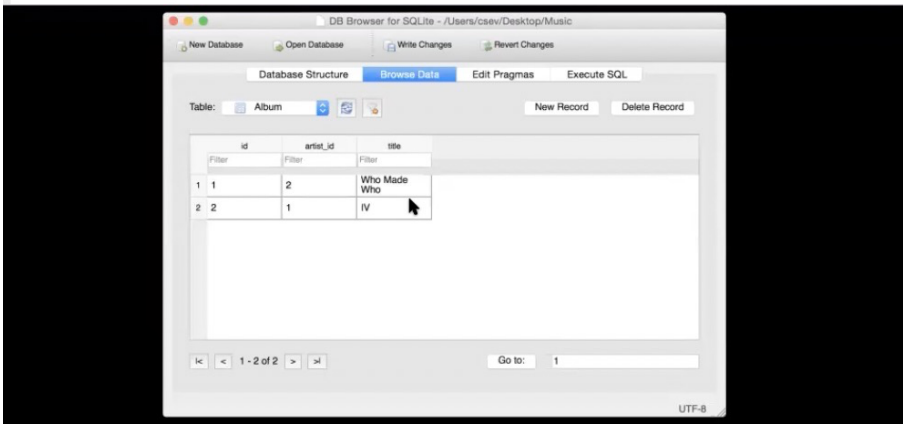
Browse Data:



INSERT INTO Artist (name) VALUES ('AC/DC')

Browse data:

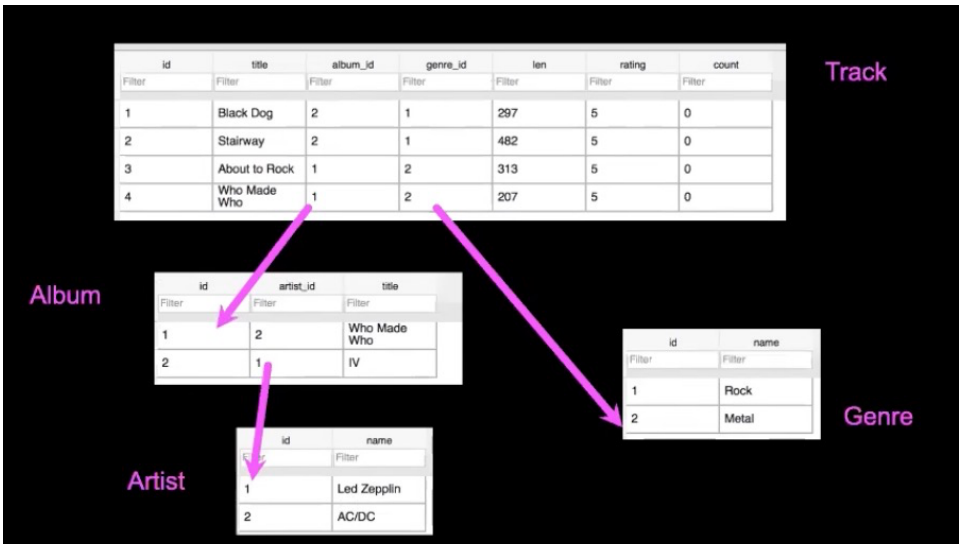




insert into Album (title, artist_id) values ('Who Made Who', 2)
insert into Album (title, artist_id) values ('IV', 1)

```
insert into Track (title, rating, len, count, album_id, genre_id)
     values ('Black Dog', 5, 297, 0, 2, 1)
insert into Track (title, rating, len, count, album_id, genre_id)
     values ('Stairway', 5, 482, 0, 2, 1)
insert into Track (title, rating, len, count, album_id, genre_id)
     values ('About to Rock', 5, 313, 0, 1, 2)
insert into Track (title, rating, len, count, album_id, genre_id)
     values ('Who Made Who', 5, 207, 0, 1, 2)
```

	id	title	album_id	genre_id	len	rating	count
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Black Dog	2	1	297	5	0
2	2	Stairway	2	1	482	5	0
3	3	About to Rock	1	2	313	5	0
4	4	Who Made Who	1	2	207	5	0



Summary

- Relational databases allow us to scale to very large amounts of data
- The key is to have one copy of any data element and use relations and joins to link the data to multiple places
- This greatly reduces the amount of data which much be scanned when doing complex operations across large amounts of data
- Database and SQL design is a bit of an art form