**Exercise 0**

$ sqlite3 test.db

SQLite version 3.7.8 2011-09-19 14:49:19

Enter ".help" for instructions

Enter SQL statements terminated with a ";"

sqlite> create table test (id);

sqlite> .quit

**Exercise 1 Creating Tables**

**CREATE** **TABLE** person **(**

id INTEGER **PRIMARY** **KEY,**

first\_name TEXT**,**

last\_name TEXT**,**

age INTEGER

**);**

Run a script from the command line

sqlite3 ex1.db < ex1.sql

Above creates ex1.db

**Exercise 2 Creating a Multi-table database**

**CREATE** **TABLE** person **(**

id INTEGER **PRIMARY** **KEY,**

first\_name TEXT**,**

last\_name TEXT**,**

age INTEGER

**);**

**CREATE** **TABLE** pet **(**

id INTEGER **PRIMARY** **KEY,**

name TEXT**,**

breed TEXT**,**

age INTEGER**,**

dead INTEGER

**);**

**CREATE** **TABLE** person\_pet **(**

person\_id INTEGER**,**

pet\_id INTEGER

**);**

sqlite3 ex2.db < ex2.sql

Above creates ex1.db

Open the database and use the .schema command to dump it:

**Extra Credit**

* In these tables I made a 3rd relation table to link them. How would you get rid of this relation table person\_pet and put that information right into person?   
  Answer : add “pet\_id” column to person and delete “person\_pet” table  
  What's the implication of this change?  
  Answer : a person can only have one pet.
* If you can put one row into person\_pet, can you put more than one?   
  Answer : Yes

How would you record a crazy cat lady with 50 cats? Answer: insert 50 rows

* Create another table for the cars people might own, and create its corresponding relation table.

**CREATE** **TABLE** cars **(**

id INTEGER **PRIMARY** **KEY,**

make TEXT**,**

model TEXT**,**

age INTEGER

**);**

**CREATE** **TABLE** person\_car **(**

person\_id INTEGER**,**

car\_id INTEGER

**);**

* Search for "sqlite3 datatypes" in your favorite search engine and go read the "Datatypes In SQLite Version 3" document. Take notes on what types you can use and other things that seem important. We'll cover more later.

# Exercise 3: Inserting Data

**INSERT** **INTO** person **(**id**,** first\_name**,** last\_name**,** age**)**

**VALUES** **(0,** "Zed"**,** "Shaw"**,** **37);**

**INSERT** **INTO** pet **(**id**,** name**,** breed**,** age**,** dead**)**

**VALUES** **(0,** "Fluffy"**,** "Unicorn"**,** **1000,** **0);**

**INSERT** **INTO** pet **VALUES** **(1,** "Gigantor"**,** "Robot"**,** **1,** **1);**

$ sqlite3 ex3.db < ex2.sql

$ sqlite3 -echo ex3.db < ex3.sql

*INSERT INTO person (id, first\_name, last\_name, age)*

*VALUES (0, "Zed", "Shaw", 37);*

*INSERT INTO pet (id, name, breed, age, dead)*

*VALUES (0, "Fluffy", "Unicorn", 1000, 0);*

*INSERT INTO pet VALUES (1, "Gigantor", "Robot", 1, 1);*

$

**Extra Credit**

* Insert yourself and your pets (or imaginary pets like I have).

INSERT INTO person (id, first\_name, last\_name, age)

VALUES (1, "Steph", "Smith", 37);

INSERT INTO pet (id, name, breed, age, dead)

VALUES (2, "Betsey", "Spaniel", 9, 1);

INSERT INTO pet VALUES (2, "Iggie", "Iguana", 1, 1);

* If you changed the database in the last exercise to not have the person\_pet table then make a new database with that schema, and insert the same information into it.

INSERT INTO person\_pet(person\_id, pet\_id)

VALUES (1, 2);

INSERT INTO person\_pet(person\_id, pet\_id)

VALUES (1, 3);

* Go back to the list of data types and take notes on what format you need for the different types. For example, how many ways can you write TEXT data.
* CHARACTER(20)  
  VARCHAR(255)  
  VARYING CHARACTER(255)  
  NCHAR(55)  
  NATIVE CHARACTER(70)  
  NVARCHAR(100)  
  TEXT  
  CLOB

<https://www.sqlite.org/datatype3.html>

# Exercise 4: Insert Referential Data

**INSERT** **INTO** person\_pet **(**person\_id**,** pet\_id**)** **VALUES** **(0,** **0);**

**INSERT** **INTO** person\_pet **VALUES** **(0,** **1);**

$ sqlite3 -echo ex3.db < ex4.sql

*INSERT INTO person\_pet (person\_id, pet\_id) VALUES (0, 0);*

*INSERT INTO person\_pet VALUES (0, 1);* $

**Extra Credit**

* Add the relationships for you and your pets.

Answer – done in Exercise 3

* Using this table, could a pet be owned by more than one person?

Answer - yes

Is that logically possible? Answer - no

What about the family dog? Wouldn't everyone in the family technically own it?

* Given the above, and given that you have an alternative design that puts the pet\_id in the person table, which design is better for this situation?  
  Answer - separate relation table with person\_id and pet\_id is better

# Exercise 5: Selecting Data

**SELECT** **\*** **FROM** person**;**

**SELECT** name**,** age **FROM** pet**;**

**SELECT** name**,** age **FROM** pet **WHERE** dead **=** **0;**

**SELECT** **\*** **FROM** person **WHERE** first\_name **!=** "Zed"**;**

$ sqlite3 -echo ex3.db < ex5.sql

*SELECT \* FROM person;*

*0|Zed|Shaw|37*

*SELECT name, age FROM pet;*

*Fluffy|1000*

*Gigantor|1*

*SELECT name, age FROM pet WHERE dead = 0;*

*Fluffy|1000*

*SELECT \* FROM person WHERE first\_name != "Zed";*

$

**Extra Credit**

* Write a query that finds all pets older than 10 years.

Answer – SELECT \* FROM pets WHERE age > 10

* Write a query to find all people younger than you.

Answer – SELECT \* FROM person WHERE age < 37

* Do one that's older.

Answer – SELECT \* FROM person WHERE age > 37

* Write a query that uses more than one test in the WHERE clause using the AND to write it. For example, WHERE first\_name = "Zed" AND age > 30.

Answer - SELECT \* FROM person WHERE first\_name = "Zed" AND age > 30

* Do another query that searches for rows using 3 columns and uses both AND and OR operators.  
  Answer – SELECT \* person WHERE first\_name = ‘Steph’ AND age = 37 OR id = 1

# Exercise 6: Select Across Many Tables

**SELECT** pet**.**id**,** pet**.**name**,** pet**.**age**,** pet**.**dead

**FROM** pet**,** person\_pet**,** person

**WHERE**

pet**.**id **=** person\_pet**.**pet\_id **AND**

person\_pet**.**person\_id **=** person**.**id **AND**

person**.**first\_name **=** "Zed"**;**

$ sqlite3 -column -header ex3.db < ex6.sql

*id name age dead*

*---------- ---------- ---------- ----------*

*0 Fluffy 1000 0*

*1 Gigantor 1 1*

$

**Extra Credit**

* This may be a mind blowing weird way to look at data if you already know a language like Python or Ruby. Take the time to model the same relationships using classes and objects then map it to this setup.

Answer -

* Do a query that finds your pets you've added thus far.

Answer –

**SELECT** pet**.**id**,** pet**.**name**,** pet**.**age**,** pet**.**dead

**FROM** pet**,** person\_pet**,** person

**WHERE**

pet**.**id **=** person\_pet**.**pet\_id **AND**

person\_pet**.**person\_id **=** person**.**id **AND**

person**.**first\_name **=** "Steph"**;**

* Change the queries to use your person.id intead of the person.name like I've been doing.