**IS 456 IT Database Systems Management**

**HOP01A Basic SQLite Queries**

4/13/2021 Developed by Farzin Bahadori

5/13/2021 Developed by Smita Dutta

School of Technology & Computing @ City University of Seattle (CityU)

**Before You Start**

* The directory path shown in screenshots may be different from yours.
* Some steps are not explained in the tutorial**.** If you are not sure what to do:
  1. Consult the resources listed below.
  2. If you cannot solve the problem after a few tries, ask a TA for help.

**Learning Outcomes**

Students will be able to:

* Understand the SQLite queries.
* Run queries in SQLite.
* Build a table

**Execute the following statement:**

-- 03 The SELECT statement

-- world.db

SELECT 'Hello, World';

SELECT 'Hello, World' AS Result;

SELECT \* FROM Country;

SELECT \* FROM Country ORDER BY Name;

SELECT Name, LifeExpectancy FROM Country ORDER BY Name;

SELECT Name, LifeExpectancy AS "Life Expectancy" FROM Country ORDER BY Name;

-- 04 Selecting Rows

-- world.db

SELECT Name, Continent, Region FROM Country;

SELECT Name, Continent, Region FROM Country WHERE Continent = 'Europe';

SELECT Name, Continent, Region FROM Country WHERE Continent = 'Europe' ORDER BY Name;

SELECT Name, Continent, Region FROM Country WHERE Continent = 'Europe' ORDER BY Name LIMIT 5;

SELECT Name, Continent, Region FROM Country WHERE Continent = 'Europe' ORDER BY Name LIMIT 5 OFFSET 5;

-- 05 Selecting Columns

-- world.db

SELECT \* from Country;

SELECT Name, Continent, Region from Country;

SELECT Name AS Country, Continent, Region from Country;

SELECT Name AS Country, Region, Continent from Country;

-- 06 Counting Rows

-- world.db

SELECT COUNT(\*) FROM Country;

SELECT COUNT(\*) FROM Country WHERE Population > 1000000;

SELECT COUNT(\*) FROM Country WHERE Population > 100000000;

SELECT COUNT(\*) FROM Country WHERE Population > 100000000 AND Continent = 'Europe' ;

SELECT COUNT(\*) FROM Country;

SELECT COUNT(LifeExpectancy) FROM Country;

-- 07 Inserting Data

-- test.db

SELECT \* FROM customer;

INSERT INTO customer (name, address, city, state, zip)

VALUES ('Fred Flintstone', '123 Cobblestone Way', 'Bedrock', 'CA', '91234');

INSERT INTO customer (name, city, state)

VALUES ('Jimi Hendrix', 'Renton', 'WA');

-- 08 Updating Data

-- test.db

SELECT \* FROM customer;

UPDATE customer SET address = '123 Music Avenue', zip = '98056' WHERE id = 5;

UPDATE customer SET address = '2603 S Washington St', zip = '98056' WHERE id = 5;

UPDATE customer SET address = NULL, zip = NULL WHERE id = 5;

-- 09 Deleting Data

-- test.db

SELECT \* FROM customer WHERE id = 4;

DELETE FROM customer WHERE id = 4;

SELECT \* FROM customer;

DELETE FROM customer WHERE id = 5;

SELECT \* FROM customer;

# Screenshots

Provide at least 3 screenshots as part of HOP submission.



# Summary

Write a 150-word summary to explain your understandings and findings from this lab assignment.

What I learned from this Hands-On Practice was that I do not like using SQLite. Some of the main differences between SQLite and T-SQL are SQLite uses LIMIT whereas TSQL uses TOP to limit the amount of results returned. Using an editor to view the queries is nice somewhat but is very tedious as SQLite only processes one query at a time instead of displaying multiple queries as TSQL does. The way databases are used is strange as the explicit nature of calling the database is not readily apparent in SQLite.

In the CLI SQLite is only apparent at the initial call and can quickly become lost in the mix if the user doesn’t pay attention. However, it seems that the appeal of SQLite is how it processes queries from outside sources as interfacing with it is easy to learn.