Introduction to Data Management with Databases & SQL

Databases are invaluable tools for organization and are better than a spreadsheet for working with multiple data sets, asking questions, and adding structure to your data. SQL is a programming language for working with databases. This workshop will introduce you to the basics of SQL, and will include hands-on practice creating databases and tables, importing data, and querying the database. The topics covered in this workshop are applicable across different SQL implementations, including MySQL, SQLite, and PostgreSQL. No previous experience with SQL is necessary.

****

Software/Materials (materials available on GitHub repo for SQL workshop):

1. SQL cheat sheet
2. Modified Chinook database
3. SQLiteStudio

Objectives:

1. Develop a conceptual understanding of databases and SQL and learn some basic SQL commands for creating databases and executing queries

Activities:

1. Distribute laptops

2. Introductions
    a. Hi my name is ____
    b. What's your name and why are you interested in databases?
    c. Everyone please sign in!

3. Conceptual conversation
    a. The concept of relational databases
        i. Databases are an organized collection of data
        ii. Structured Query Language is a programming language for managing data in a RDBMS
        iii. Database vs. client
            1. Database holds the data, you need a client to see it
            2. We're using SQLite Studio, you can also use sqlite3 in the command line
    b. Terminology
        i. Database
            1. Contains all of your tables
        ii. Table

1. Each table contains one type of entity
2. Each row is a record
3. Each column is a field

iii. Query

1. Any operation on the database

4. Download SQLiteStudio

5. Using SQL to create a database with your data
    a. This database will have two tables, one for each type of entity; our entities are academic programs and students

```
--CREATE A DATABASE ON THE DESKTOP USING THE SQLite Studio GUI
--CLICK ON Database >> Add a database

--CREATE A TABLE FOR PROGRAMS
CREATE TABLE programs (
id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
program VARCHAR
);

--INSERT SOME DATA INTO THE PROGRAMS TABLE
INSERT INTO programs(program) VALUES ('Anthropology');
INSERT INTO programs(program) VALUES ('Linguistics');
INSERT INTO programs(program) VALUES ('Biology');

--ADD ANOTHER FIELD TO THE PROGRAMS TABLE
ALTER TABLE programs
ADD program_level VARCHAR;

--UPDATE THE PROGRAM_LEVEL FIELD FOR EACH PROGRAM
UPDATE programs
SET program_level = 'Ph.D.' WHERE program = 'Anthropology';
SET program_level = 'Master''s' WHERE program = 'Linguistics';
SET program_level = 'Ph.D.' WHERE program = 'Biology';


--CREATE A TABLE FOR STUDENTS (PAY ATTENTION TO THE FOREIGN
KEY!)
CREATE TABLE students (
id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
student VARCHAR(255),
id_program INTEGER,
FOREIGN KEY (id_program) REFERENCES programs(id)
```

```
);

--INSERT SOME DATA INTO THE STUDENTS TABLE (REMEMBER THAT ID
WILL AUTOINCREMENT)
INSERT INTO students(student, id_program) VALUES ('Josefina',
3);
INSERT INTO students(student, id_program) VALUES ('Cecilia',
2);
INSERT INTO students(student, id_program) VALUES ('Nico', 2);
INSERT INTO students(student, id_program) VALUES ('Sarah', 1);
```

6. Using SQL to query databases
    a. We can execute queries to manipulate the data in our example database

```
--SHOW ALL THE RECORDS IN THE TABLE STUDENTS
SELECT *
FROM students;

--SHOW THE VALUE FOR THE FIELD STUDENT FOR ALL RECORDS IN THE TABLE
STUDENTS
SELECT student
FROM students;

--SHOW THE VALUES FOR THE FIELDS STUDENT AND ID FOR ALL THE RECORDS
IN THE TABLE STUDENTS
SELECT student, id
FROM students;

--SHOW ALL THE RECORDS IN THE TABLE STUDENTS WHERE THE VALUE OF THE
FIELD ID IS EQUAL TO '3'
SELECT *
FROM students
WHERE id = '3';

--SHOW ALL THE RECORDS FOR STUDENT WITH THE INFORMATION ABOUT THEIR
RESPECTIVE PROGRAMS
SELECT *
FROM students INNER JOIN programs
WHERE students.id_program = programs.id;
```

7. More data!
    a. Download the modified Chinook database (full db available here)

```
--SELECT THE 1ST 5 TRACKS BY ANY ARTIST WHOSE NAME STARTS WITH
--"aero" ORDERED BY TRACK NAME
SELECT A.name FROM track AS A INNER JOIN album AS B ON
A.AlbumId=B.AlbumId INNER JOIN artist AS C ON
B.ArtistId=C.ArtistId WHERE C.Name LIKE "aero%" ORDER BY A.name
LIMIT 5;


--SELECT THE AVERAGE TRACK LENGTH IN MINUTES GROUPED BY GENRE
SELECT B.Name,AVG(A.milliseconds/1000/60) minutes FROM track A
INNER JOIN genre B ON A.GenreId=B.GenreId GROUP BY B.GenreId
ORDER BY minutes DESC;
```

8. SQL clients
    a. Command line
    b. GUIs

9. Different SQL implementations
    a. They all handle the same basic operations, but offer different query operations at a higher level
        i. PostgreSQL, MySQL

10. Integration with other programs
    a. You can create a Python script that interacts with a SQL database using the Psycopg2 adapter

11. Resources!
    a. For practice with the basic SQL queries, W3schools has a nice tutorial.

    b. For designing your database, check out this SQL designer. Once you set up your database structure using the GUI tools, you can export the SQL code.

    c. For practice integrating SQL with Python, check out my git repo *spending*. This is a python script for keeping track of shared expenses. Download the code and play around with it from the terminal. Just FYI, this uses PostgreSQL and the Psycopg2 adapter, so you will need to do some research to figure out how to connect to a SQLite database and check the documentation to see how psycopg works.