-- SQLite3 Tutorial 1 --

SLIDE What is SQLite3

- **Relational Database**
- **Packaged with your Application**
- **Structured Query Language**
- **Queries**

SQLite3 is a relational database that is packaged inside your application. A relational database is a database that has strictly defined tables with specific rows and columns. You define, retrieve and alter that data using the Structured Query Language (SQL). You issue commands called queries using SQL.

SLIDE 2 Understanding Tables

Primary keys are unique numbers that identify unique piece of data stored in a table. Rows contain all the data that is specific to each entity. Columns represent the data that each entity has.

SLIDE 3 Primary & Foreign Keys

We join data stored in different tables by referring to the unique Primary Keys that identify each entity. When you store a primary key from another table that key is referred to as a foreign key.

```
-- Create a database

sqlite3 studentdb.db

-- Create the sex table because we want to define that it can only have either the value 'F' or 'M'
-- When you want to have a list of options other databases define you should use an enumerated type which is a list of values. SQLite doesn't have enums so instead we will create a table for sex

-- Marking data as a primary key means that an incrementing value will be added if no value is provided

CREATE TABLE sex(
id TEXT PRIMARY KEY NOT NULL,
sex_type INTEGER);

-- You insert values into the sex table with insert

INSERT INTO sex(id, sex_type) VALUES ('M', 1);
INSERT INTO sex(id, sex_type) VALUES ('F', 2);

-- Display the values added
-- Format the output to show each record on a separate line with data aligned in columns
.mode column

-- Display the column names as headers
.headers on
```

```
-- Change our table column widths
.width 15 20

-- Display everything from the table sex
SELECT * FROM sex;

-- Display every value on its own line
.mode line
SELECT * FROM sex;

-- Display statement used to create the table
.schema sex

-- Create the student table
-- AUTOINCREMENT increments itself starting from 1 each time a new entity is entered
-- You define a foreign key by referencing the table and name for the id where the foreign key resides.

CREATE TABLE student(
f_name VARCHAR(23) NOT NULL,
l_name VARCHAR(23) NOT NULL,
sex CHARACTER(1) NOT NULL,
id INTEGER PRIMARY KEY AUTOINCREMENT,
FOREIGN KEY (sex) REFERENCES sex(id));
```

SLIDE 4 Data Types

- **Dynamic Type System**
- **Integer**
- **Real**
- **Text**
- **Blob**
- **Numeric**

SQLite uses a dynamic type system in which the data type is defined based on the type of data entered.

There are 5 data types being Integer, Real, Numeric, Text, Blob and Null. Integers are numbers without decimal places. Reals are numbers with decimal places. Text can store an unlimited number of characters. Blobs which stands for Binary Large OBject store any kind of data. A Null is anything without a value.

Numerics can contain values from any of the other types. It converts the data entered to different types as long as the data is stored and no data is lost. For example if you insert '123' it will convert it into an Integer. If a real type is entered as long as it is less than 15 decimal digits it will be stored as a Real and if larger it will be stored as a Text.

SLIDE 5 Data Type Conversion

- **Integer : INT, INTEGER, TINYINT, SMALLINT, …**
- **Real : REAL, DOUBLE, FLOAT, …**

- **Text : VARCHAR, CHARACTER, TEXT, …**
- **Blob : BLOB**
- **Numeric : DECIMAL, BOOL, DATE, …**

Any type containing INT is stored as an Integer. Any type containing Char or Text is a Text. Blobs are Blobs. Reals are any of the data types listed as well as Double Precision. Everything else except for Null is stored as a Numeric.

```
-- Create an enum that represents the type of test
CREATE TABLE test_type(
id INTEGER PRIMARY KEY NOT NULL,
type TEXT NOT NULL);

-- Insert values
INSERT INTO test_type(id, type) VALUES (1, 'Q');
INSERT INTO test_type(id, type) VALUES (2, 'T');

-- Create test table
CREATE TABLE test(
date DATE NOT NULL, -- DATE is stored as a NUMERIC type
test_type INT NOT NULL,
id INTEGER PRIMARY KEY AUTOINCREMENT,
FOREIGN KEY (test_type) REFERENCES test_type(id));

-- Table for test scores
-- I use a composite primary key here only to show you how to define them
-- They are typically not used because the more complicated an index is the more inefficient
the database will be

CREATE TABLE test_score(
student_id INTEGER NOT NULL,
test_id INTEGER NOT NULL,
score INTEGER NOT NULL,
FOREIGN KEY (test_id) REFERENCES test(id),
FOREIGN KEY (student_id) REFERENCES student(id),
PRIMARY KEY (test_id, student_id)); -- Composite Primary Key

-- Table for absences
CREATE TABLE absence(
student_id INTEGER NOT NULL,
date DATE NOT NULL,
PRIMARY KEY (student_id, date),
FOREIGN KEY (student_id) REFERENCES student(id));

-- Display all tables & views
.tables

-- Exit SQLite
.exit
```

That's all for now. In the next video I'll cover how to generate random data using Python to populate our databases. In the videos that follow that I'll show how to do just about anything with Python and SQLite3.