SQLite3 - Data Definition Language (DDL)

Creating/loading a database (From a Linux terminal)

```
>>> sqlite3 user_database.db
```

This will either load the existing database if the file has already been created, or it will create a new database.

Listing Database Contents

```
.table;
```

Show the names of all tables in the database

```
PRAGMA table_info(table_name);
```

List all of the column names and data types for a table

Creating a table

```
CREATE TABLE users (user_ID Integer PRIMARY KEY AUTOINCREMENT, name text, departmentID number  REFERENCES departments(department_ID));
```

Here we created a new users table, setting the primary key to the user_ID column and added a foreign key reference to another table

Data Types Available

- Null, Integer, Real, Text, Blob
- Boolean : Just use integer 1/0
- Date: Use either text/integer (see below)

Date Data Type

No date type so use either

- Text : ISO8601 - "YYYY-MM-DD HH:MM:SS.SSS"
- Integer: Epoch time 41323332

Altering a table

```
ALTER TABLE users ADD COLUMN email_address text;
```

```
ALTER TABLE users DROP COLUMN email;
```

```
ALTER TABLE users RENAME email_address TO email;
```

Deleting a table

```
DROP TABLE users;
```

SQLite3 - Data Manipulation Language (DML)

SELECT

SELECT * FROM users;
This gets all columns from the table

SELECT username,email FROM users WHERE first_name LIKE 'bob';
This query retrieves columns from the database that match a filter.

SELECT email FROM users WHERE age > 21 LIMIT 5 ORDER BY age;
Get the usernames of the first 5 users with the age over 5, sort by age.

INSERT

INSERT INTO users (username,email) VALUES ('Bob1', 'bob@bob.com');

UPDATE

UPDATE users SET email = 'bob@bobby.com' WHERE username = 'Bob1';
This function will update ALL Records where the username is a match!

DELETE

DELETE FROM users WHERE username = 'Bob1';
This function will delete ALL Records where the username is a match!

ORDER BY

SELECT * FROM users ORDER BY first_name ASC;
Orders the query results either ascending or descending.

COUNT

SELECT COUNT(userID) FROM users;
#Counts up how many records in the users table

GROUP BY

SELECT count(userID) FROM users GROUP BY departmentID;
This query aggregates rows creating summary rows from which you can use count / sum / avg / min / max functions

---

SUM / AVG / MIN / MAX

SELECT SUM(Quantity) FROM invoice;
This aggregation function adds up the combined totals for all values in the query column.

INNER JOIN

SELECT users.email, departments.name FROM users
INNER JOIN departments ON users.department_ID =
departments.department_ID;
This query combines results from multiple tables and returns a joined table.