

# Table Relations and Normalization: Takeaways



by Dataquest Labs, Inc. - All rights reserved © 2020

## Syntax

- Launching the SQLite shell:

```
sqlite3 chinook.db
```

- Switching column headers on:

```
.headers on
```

- Switching to column mode:

```
.mode column
```

- Displaying help text:

```
.help
```

- Displaying a list of all tables and views:

```
.tables
```

- Running BASH shell commands:

```
.shell [command]
```

- Viewing table schema.

```
.schema [table_name]
```

- Quitting the SQLite Shell:

```
.quit
```

- Creating a table:

```
CREATE TABLE [table_name] (  
  [column1_name] [column1_type],  
  [column2_name] [column2_type],  
  [column3_name] [column3_type],  
  [...]  
);
```

- Creating a table with a primary and a foreign key:

```
CREATE TABLE purchase (  
    purchase_id INTEGER PRIMARY KEY,  
    user_id INTEGER,  
    purchase_date TEXT,  
    total NUMERIC,  
    FOREIGN KEY (user_id) REFERENCES user(user_id)  
);
```

- Creating a compound primary key:

```
CREATE TABLE [table_name] (  
    [column_one_name] [column_one_type],  
    [column_two_name] [column_two_type],  
    [column_three_name] [column_three_type],  
    [column_four_name] [column_four_type],  
    PRIMARY KEY (column_one_name, column_two_name)  
);
```

- Inserting values into a table:

```
INSERT INTO [table_name] (  
    [column1_name],  
    [column2_name],  
    [column3_name]  
) VALUES (  
    [value1],  
    [value2],  
    [value3]  
);
```

OR

```
INSERT INTO [table_name] VALUES ([value1], [value2], [value3]);
```

- Deleting selected rows from a table:

```
DELETE FROM [table_name]
WHERE [expression];
```

- Adding a column:

```
ALTER TABLE [table_name]
ADD COLUMN [column_name] [column_type];
```

- Changing values for existing rows:

```
UPDATE [table_name]
SET [column_name] = [expression]
WHERE [expression]
```

## Concepts

- A semicolon is necessary to end your queries in the SQLite shell.
- SQLite comes with a number of dot commands to work with databases.
- You run dot commands are ran within SQLite shell.
- SQLite uses **TEXT**, **INTEGER**, **REAL**, **NUMERIC**, **BLOB** data types behind the scenes.
- A breakdown of SQLite data types and equivalent data types from other types of SQL.

Type	Commonly Used For	Equivalent Types
<b>TEXT</b>	Names Email Addresses Dates and Times Phone Numbers	<b>CHARACTER</b> <b>VARCHAR</b> <b>NCHAR</b> <b>NVARCHAR</b> <b>DATETIME</b>
<b>INTEGER</b>	IDs Quantities	<b>INT</b> <b>SMALLINT</b> <b>BIGINT</b> <b>INT8</b>
<b>REAL</b>	Weights Averages	<b>DOUBLE</b> <b>FLOAT</b>
<b>NUMERIC</b>	Prices Statuses	<b>DECIMAL</b> <b>BOOLEAN</b>
<b>BLOB</b>	Binary Data	<b>BLOB</b>



A primary key is a unique identifier for each row



A foreign key describes how the column is related to a foreign table



Database normalization optimizes the design of databases, allowing for stronger data integrity. For example, it helps you avoid data duplication if a record being stored multiple times, and it helps avoid data modification if you need to update several rows after removing duplicate records.



A compound primary key is when two or more columns combine to form a primary key.



## Resources



[SQLite Shell](#)



[Database Normalization](#)



Takeaways by Dataquest Labs, Inc. - All rights reserved © 2020