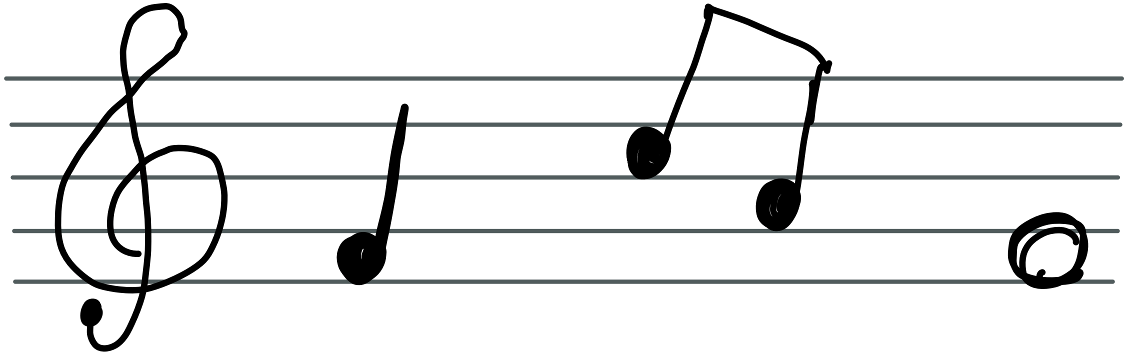
CS50 Course-wide Supersection

carter@cs50.harvard.edu

cs50.ly/question



TEXT	NUMERIC	INTEGER	REAL	BLOB
name	volume	length		
		tempo		

ALBUM TITLE

Structured Query Language

cs50.ly/question

directory software database

```
songs/ $ sqlite3 songs.db
```

```
sqlite> SELECT
```

songs

id	name	tempo	...
1	Neighborhood	77	...
2	SAD!	75	...
3	rockstar	160	...
...	...	...	...

everything

```
SELECT  *
FROM songs;
```

table

songs

id	**name**	**tempo**	**...**
1	Neighborhood	77	...
2	SAD!	75	...
3	rockstar	160	...
...	...	...	...

column name

```
SELECT name
FROM songs;
```

table name

songs

id	name	tempo	...
1	Neighborhood	77	...
2	SAD!	75	...
3	rockstar	160	...
...	...	...	...

```
SELECT name
FROM songs
WHERE tempo < 100;
```

name column

songs table

condition

songs

id	name	tempo	...
1	Neighborhood	77	...
2	SAD!	75	...
3	rockstar	160	...
...	...	...	...

```
SELECT name
FROM songs
WHERE tempo < 100
AND danceability > 0.5;
```

adding on to our condition

sqlite3

```
.schema
```

```
cd
cd songs        sqlite3 songs.db
                                  sqlite>
```

3-Minute Exercise

Write a SELECT query to search for songs that are highly danceable and energetic, based on the columns in the **songs** table.

cs50.ly/question

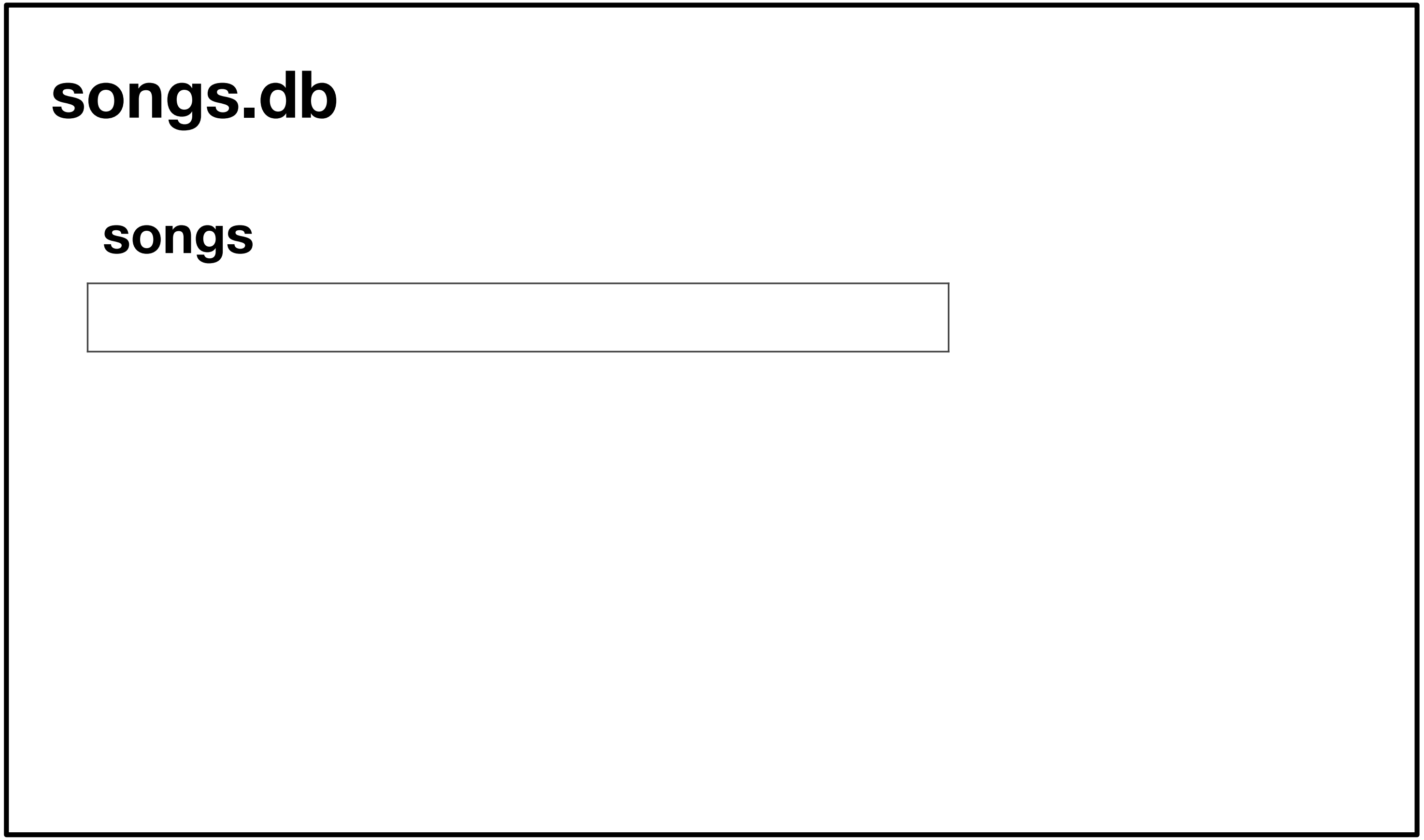
```
SELECT name
FROM songs
WHERE danceability >0.8
```

AND energy > 0.8 ;

```
songs/ $ sqlite3 songs.db
```

songs.db

```
CREATE TABLE songs;
```


songs.db
songs

```
CREATE TABLE songs (
    id INTEGER,
name ~~song~~ TEXT,
    tempo INTEGER,
    PRIMARY KEY(id)
                 ↑
);
```

songs.db

songs

id	name	tempo

table

column name

```
INSERT INTO songs
(id, name, tempo)
VALUES
(1, 'Drive', 142);
```

id name tempo

songs.db

songs

id	name	tempo
1	Drive	142

```
CREATE TABLE artists (
    id INTEGER,
    name TEXT,
    PRIMARY KEY(id),
);
```

songs.db

songs

id	name	tempo
1	Drive	142

artists

id	name
1	Oh wonder

```
INSERT INTO songs
(id, name)
VALUES
(1, 'Oh Wonder');
```

```
cd songs
sqlite3 mysongs.db
```

5-minute Exercise

Create a new database, **mysongs.db**, with two tables, one for songs and one for artists. Insert your favorite song and artist.

```
.schema CREATE TABLE songs(
                    name TEXT,
                    born INTEGER
```

[illegible]

);

table

new value

column

```
UPDATE songs
SET tempo = 71
WHERE name = 'Drive';
```

```
UPDATE <table>
SET <column> = <value>
WHERE <predicate>;
```

songs.db

songs

id	name	tempo
1	Drive	71

SELECT *
FROM songs;

.schema

artists

id	name
1	Oh Wonder

3-minute Exercise

Update your new database by
changing the value of a certain column
in a given row.

table

```
DELETE FROM songs
WHERE name = 'Drive';
```

column

```
DELETE FROM <table>
WHERE <predicate>;
```

songs.db

songs

id	name	tempo

artists

id	name
1	Oh Wonder

cs50.ly/question

songs.db

songs

id	name	tempo	duration	artist_id
1	Something Comforting	144	282	23
2	Drive	142	196	45

artists

id	name	age	label
23	Porter Robinson	29	Mom+Pop
45	Oh Wonder	31	Republic

```sql
SELECT name
FROM artists
WHERE duration < 240;
```

songs.db

songs

id	name	tempo	duration	artist_id
1	Something Comforting	144	282	23
2	Drive	142	196	45

artists

id	name	age	label
23	Porter Robinson	29	Mom+Pop
45	Oh Wonder	31	Republic

```
SELECT artists.name
FROM artists
JOIN songs
ON songs.artist_id =
artists.id
WHERE duration < 240;
```

artists JOIN songs

id	songs.name	tempo	duration	artist_id	artists.name	age	label
1	Something Comforting	144	282	23	Porter Robinson	29	Mom+ Pop
2	Drive	142	196	45	Oh Wonder	31	Republic

```
SELECT name
FROM artists
WHERE id IN
(
                        45
    SELECT artist_id
    FROM songs
    WHERE duration < 240
);
```

songs.db

songs

id	name	tempo	duration	artist_id
1	Something Comforting	144	282	23
2	Drive	142	196	45

```
sqlite> SELECT * FROM songs;
```

artists

id	name	age	label
23	Porter Robinson	29	Mom+Pop
45	Oh Wonder	31	Republic

```
SELECT name
FROM artists
WHERE id IN =
(
    45
);
```

ORDER BY

Will order results by given column

`ORDER BY title`	Order results alphabetically by title
`ORDER BY rating ASC`	Order results by rating, starting with lowest and **ASC**ending.
`ORDER BY rating DESC`	Order results by rating, starting with highest and **DESC**ending.

COUNT

Will count results of SELECT statements

`SELECT COUNT(title)`	Return count of selected titles, not titles themselves.
`SELECT COUNT(*)`	Return count of selected rows, not rows themselves

LIKE

% indicates wildcard characters in relative location of string:

`WHERE title LIKE "%Harry Potter"`	All titles with Harry Potter at the end.
`WHERE title LIKE "Harry Potter%"`	All titles with Harry Potter at the beginning.
`WHERE title LIKE "%Harry Potter%"`	All titles with Harry Potter somewhere in the string.

LIMIT

Prints first number of values from query

`LIMIT 5`	Print first 5 rows from query

`ORDER BY rating DESC`
`LIMIT 5`
	Print first 5 highest ratings

`ORDER BY rating ASC`
`LIMIT 3`
	Print first 3 lowest ratings

AND

Can find intersection of WHERE queries

`WHERE id IN [101, 102]`
`AND`
`id IN [102]`
	Returns 102

```
sqlite > SELECT name
   ...> FROM artists;
sqlite>
```

Lab: Songs

Use what you've learned to query a database of songs!

```
-- Comment
```

```
cd songs
code 1.sql
```