This is CS50

Week 7

Yuliia Zhukovets

Preceptor

yuliia@cs50.harvard.edu

Agenda

- Design Principles
- Database Design Practice
- Songs

Design Principles

Quick Technical Design Principles

- Each table should be a collection of a **single entity**.
  - For example, we should have a different table for each of employees, employee *relationships*, songs, and artists.

youtunes.db

employees

name	role
Alice	IT Staff
Laura	General Manager

employee_relationships

manager	employee
Laura	Alice

Quick Technical Design Principles

- Each table should be a collection of a **single entity.**
  - For example, we should have a different table for each of employees, employee *relationships*, songs, and artists.

- Each piece of data should be stored in a **single location**, and thereafter referred to via its ID, aka primary key.
  - For example, we should ensure every employee has an ID, and use that ID in the employee *relationships* table.

youtunes.db

employees

id	name	role
1	Alice	IT Staff
2	Laura	General Manager

employee_relationships

manager_id	employee_id
2	1

Database Design Practice

Create a Database

1. In your terminal, use **`sqlite3 friends.db`** to create a new database, **`friends.db`**.

Create a Database

1. In your terminal, use **`sqlite3 friends.db`** to create a new database, **`friends.db`**.
2. Create a table, also named **`friends`**, with at least two columns, **id**, **name**, and at least one additional friend attribute you'd like to store. Denote **id** as the primary key.

```
CREATE TABLE "friend" (
    "id",
    "first_name",
    "last_name",
    PRIMARY KEY("id")
);
```

```
CREATE TABLE "friend" (
    "id" INTEGER,
    "first_name" TEXT,
    "last_name" TEXT,
    PRIMARY KEY("id")
);
```

```
CREATE TABLE "friend" (
    "id" INTEGER NOT NULL,
    "first_name" TEXT NOT NULL,
    "last_name" TEXT NOT NULL,
    PRIMARY KEY("id")
);
```

NOT NULL

"required"

Insert into a Database

Ensure you're still in your sqlite prompt by looking at your terminal prefix.

1. Use `INSERT INTO` to add at least 3 friends to your table.
2. Use `SELECT` to ensure that your friends have been added.

```
INSERT INTO tablename (column1, column2)
VALUES (value1, value2);
```

```
SELECT * FROM tablename;
```

Problem Set: Songs

songs.db

songs

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

artists

id	name	birthyear	label
23	Porter Robinson	1992	Mom+Pop
45	Oh Wonder	1990	Republic

1–5

Selecting

Ordering

Limiting

Aggregating

Selecting

```
SELECT name
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

artists

id	name	birthyear	label
23	Porter Robinson	1992	Mom+Pop
45	Oh Wonder	1990	Republic

LIKE

% indicates wildcard characters in relative location of string:

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

`1.sql`

List the names of all songs in the database.

1.sql

List the names of all songs in the database.

```
SELECT name
FROM songs;
```

Ordering

```
SELECT *
FROM songs
WHERE tempo > 100
ORDER BY tempo;
```

```
SELECT *
FROM songs
WHERE tempo > 100
ORDER BY tempo DESC;
```

`2.sql`

List the names of all songs in increasing order of tempo.

2.sql

List the names of all songs in increasing order of tempo.

```
SELECT name
FROM songs
ORDER BY tempo;
```

Limiting

```
SELECT *
FROM songs
WHERE tempo > 100
ORDER BY tempo ASC
LIMIT 1;
```

`3.sql`

List the names of the top 5 longest songs,
in descending order of length.

3.sql

List the names of the top 5 longest songs,
in descending order of length.

```sql
SELECT name
FROM songs
ORDER BY duration_ms DESC
LIMIT 5;
```

`4.sql`

List the names of any songs that have *danceability*, *energy*, and *valence* greater than *0.75*.

4.sql

List the names of any songs that have *danceability*, *energy*, and *valence* greater than *0.75*.

```
SELECT name FROM songs
WHERE danceability > 0.75
AND energy > 0.75
AND valence > 0.75;
```

Aggregating

COUNT()
MIN()
MAX()
AVG()
SUM()

```
SELECT COUNT(*)
FROM songs
WHERE tempo > 100;
```

```
SELECT AVG(tempo)
FROM songs
WHERE tempo > 100;
```

`5.sql`

Find the average energy of all the songs.

5.sql

Find the average energy of all the songs.

```
SELECT AVG(energy)
FROM songs;
```

SELECTs
JOINs

SELECTs

JOINs

songs.db

songs

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

artists

id	name	birthyear	label
23	Porter Robinson	1992	Mom+Pop
45	Oh Wonder	1990	Republic

```
SELECT *
FROM songs
WHERE artist_id =
(
    SELECT id
    FROM artists
    WHERE name = "Oh Wonder"
);
```

```
SELECT *
FROM songs
WHERE artist_id =
(
    SELECT id
    FROM artists
    WHERE name = "Oh Wonder"
);
```

songs.db

songs

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

artists

id	name	birthyear	label
23	Porter Robinson	1992	Mom+Pop
45	Oh Wonder	1990	Republic

```
SELECT *
FROM songs
WHERE artist_id =
(
    45
);
```

```
SELECT *
FROM songs
WHERE artist_id =
(
    45
);
```

songs.db

songs

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

artists

id	name	birthyear	label
23	Porter Robinson	1992	Mom+Pop
45	Oh Wonder	1990	Republic

SELECTs

JOINs

songs.db

songs

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

artists

id	name	birthyear	label
23	Porter Robinson	1992	Mom+Pop
45	Oh Wonder	1990	Republic

songs JOIN artists

songs | artists

id	name	tempo	duration	artist_id	id	name	birthyear	label
1	Something Comforting	144	282	23	23	Porter Robinson	1992	Mom+Pop
2	Drive	142	196	45	45	Oh Wonder	1990	Republic

```
SELECT *
FROM songs
JOIN artists
ON songs.artist_id = artists.id;
```

```
SELECT *
FROM songs
JOIN artists
ON songs.artist_id = artists.id;
```

```
SELECT *
FROM songs
JOIN artists
ON songs.artist_id = artists.id
WHERE artists.name = "Oh Wonder";
```

6–8

`6.sql`

List the names of songs that are by Post Malone.

6.sql

List the names of songs that are by Post Malone.

```
SELECT name FROM songs WHERE
artist_id =
(SELECT id FROM artists
WHERE name = 'Post Malone');
```

7.sql

Find the average energy of songs that are by Drake.

7.sql

Find the average energy of songs that are by Drake.

```
SELECT AVG(energy)
FROM songs
WHERE artist_id IN (SELECT id FROM
artists WHERE name = 'Drake');
```

`8.sql`

List the names of the songs that feature other artists.

8.sql

List the names of the songs that feature other artists.

```
SELECT name
FROM songs
WHERE name LIKE '%feat.%';
```

This is CS50

Week 7