Ruby Monstas

Session 29

Agenda

Recap Databases

Introduction to databases, Part 2

Exercises

SQL Part 1 Recap

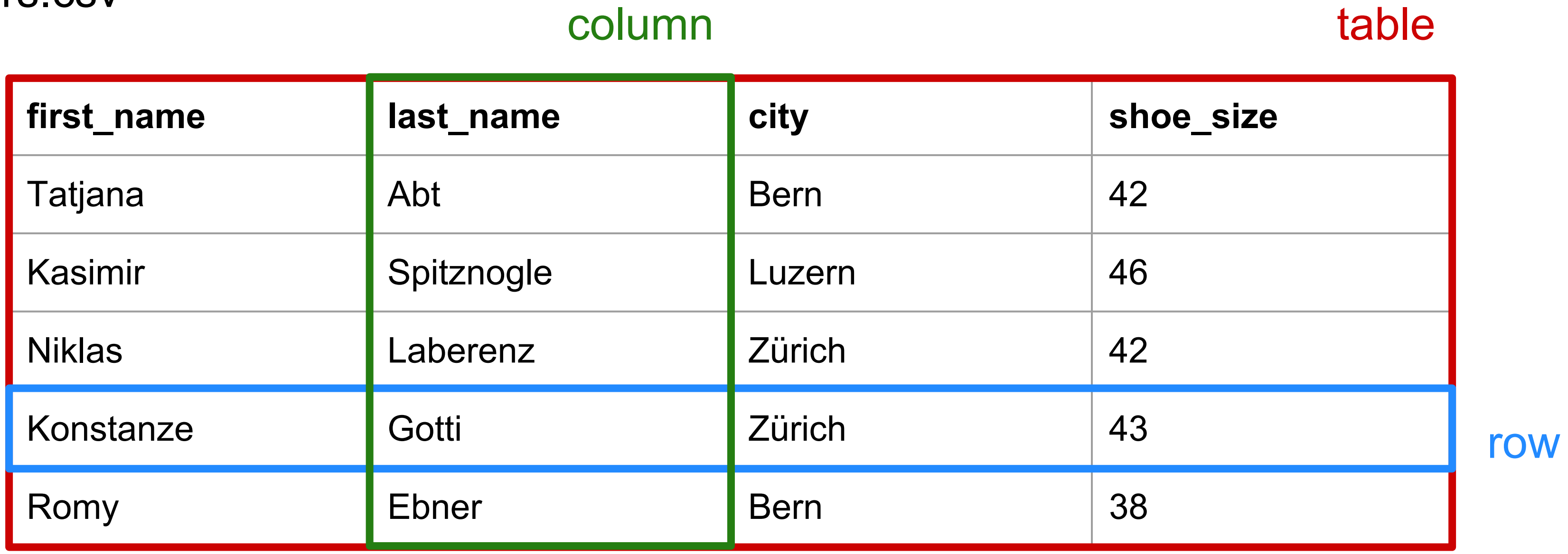
Before SQL: CSV

users.csv

```
first_name,last_name,city,shoe_size
Tatjana,Abt,Bern,42
Kasimir,Spitznogle,Luzern,46
Niklas,Laberenz,Zürich,42
Konstanze,Gotti,Zürich,43
Romy,Ebner,Bern,38
```

Before SQL: CSV

users.csv


column
table
row

first_name	last_name	city	shoe_size
Tatjana	Abt	Bern	42
Kasimir	Spitznogle	Luzern	46
Niklas	Laberenz	Zürich	42
Konstanze	Gotti	Zürich	43
Romy	Ebner	Bern	38

SQL / SQLite

SQL (Structured Query Language)

SQLite (SQL implementation)

SQL Basics

```sql
CREATE TABLE 'users' (first_name string, last_name string, city
string, shoe_size integer);

INSERT INTO 'users' VALUES ('Tatjana', 'Abt', 'Bern', 42);

INSERT INTO 'users' VALUES ('Kasimir', 'Spitznogle', 'Luzern', 46);
INSERT INTO 'users' VALUES ('Niklas', 'Laberenz', 'Zürich', 42);
INSERT INTO 'users' VALUES ('Konstanze', 'Gotti', 'Zürich', 43);
INSERT INTO 'users' VALUES ('Romy', 'Ebner', 'Bern', 38);
```

SQL Queries

```sql
SELECT first_name, last_name FROM users WHERE city == 'Bern';

SELECT max(shoe_size) FROM users;

SELECT city FROM users WHERE first_name LIKE 'K%';


SELECT shoe_size FROM users WHERE first_name LIKE '%z%' OR
last_name LIKE '%z%';
```

Introduction to databases

Part 2

SQL: UPDATE

first_name	last_name	city	shoe_size
Tatjana	Abt	Bern	42
Kasimir	Spitznogle	Luzern	46
Niklas	Laberenz	Zürich	42
Konstanze	Gotti	Zürich	43
Romy	Ebner	Bern	38

Our task: Romy moved to Luzern, we need to update her record.

SQL: UPDATE

The UPDATE query looks like the following:

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

So we provide a table and set a certain column to a value for every row that matches the where condition

SQL: UPDATE

What does the UPDATE query have to look like in our case?

first_name	last_name	city	shoe_size
Tatjana	Abt	Bern	42
Kasimir	Spitznogle	Luzern	46
Niklas	Laberenz	Zürich	42
Konstanze	Gotti	Zürich	43
Romy	Ebner	Bern	38

SQL: UPDATE

“Update the city to Luzern for the row of Romy”

```
UPDATE users SET city='Luzern' WHERE first_name == 'Romy';
```

But what if we have multiple Romys in our database?

Unique Rows

What does make each row unique?

first_name	**last_name**	**city**	**shoe_size**
Tatjana	Abt	Bern	42
Kasimir	Spitznogle	Luzern	46
Niklas	Laberenz	Zürich	42
Konstanze	Gotti	Zürich	43
Romy	Ebner	Bern	38

Row IDs

We need a column/attribute that is data-independent to reliably address a row.

id	first_name	last_name	city	shoe_size
1	Tatjana	Abt	Bern	42
2	Kasimir	Spitznogle	Luzern	46
3	Niklas	Laberenz	Zürich	42
4	Konstanze	Gotti	Zürich	43
5	Romy	Ebner	Bern	38

Row IDs

How do we update Romys record now?

id	first_name	last_name	city	shoe_size
1	Tatjana	Abt	Bern	42
2	Kasimir	Spitznogle	Luzern	46
3	Niklas	Laberenz	Zürich	42
4	Konstanze	Gotti	Zürich	43
5	Romy	Ebner	Bern	38

Row IDs

How do we update Romys record now?

```
UPDATE users SET city='Luzern' WHERE id == 5;
```

id	first_name	last_name	city	shoe_size
1	Tatjana	Abt	Bern	42
2	Kasimir	Spitznogle	Luzern	46
3	Niklas	Laberenz	Zürich	42
4	Konstanze	Gotti	Zürich	43
5	Romy	Ebner	Bern	38

Primary Key (Row ID)

This ID concept is called a primary key.

It is used to uniquely identify a row and reference it in our SQL queries.

SQL: DELETE

The DELETE query looks like the following:

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

So we provide a table and delete every row that matches the where condition

SQL: DELETE

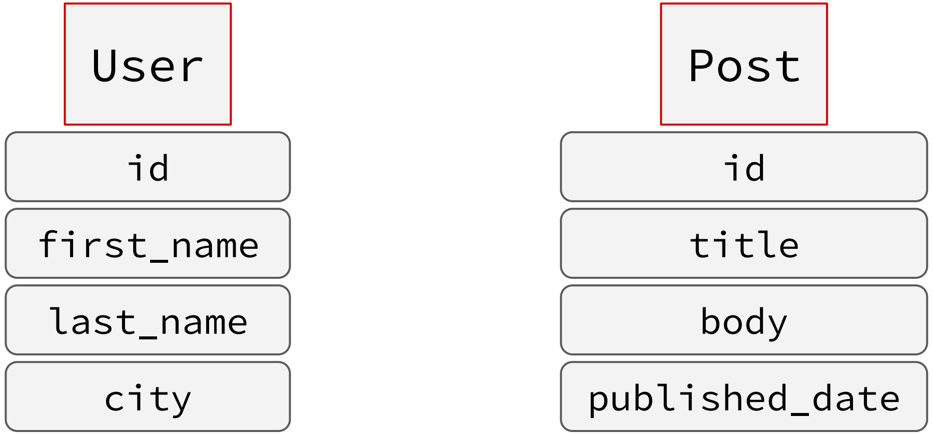
So how do we delete Romy?

```
DELETE FROM users WHERE id == 5;
```

What happens if we leave off the where clause?

SQL: Foreign Key

Usually tables don’t stand on their own. You have data spread out to multiple tables. Let’s take this example:



SQL: Foreign Key

users

id	first_name	last_name	city
1	Janet	Doe	Chicago
2	John	Doe	Denver

posts

id	title	body	published_date
1	Title 1	Text 1	2016-01-20
2	Title 2	Text 2	2016-01-11
3	Title 3	Text 3	2016-01-14
4	Title 4	Text 4	2016-01-06
5	Title 5	Text 5	2016-01-19

SQL: Foreign Key

users

id	first_name	last_name	city
1	Janet	Doe	Chicago
2	John	Doe	Denver

posts

id	title	body	published_date
1	Title 1	Text 1	2016-01-20
2	Title 2	Text 2	2016-01-11
3	Title 3	Text 3	2016-01-14
4	Title 4	Text 4	2016-01-06
5	Title 5	Text 5	2016-01-19

SQL: Foreign Key

users

id	first_name	...
1	Janet	...
2	John	...

posts

id	title	body	published_date	user_id
1	Title 1	Text 1	2016-01-20	
2	Title 2	Text 2	2016-01-11	
3	Title 3	Text 3	2016-01-14	
4	Title 4	Text 4	2016-01-06	
5	Title 5	Text 5	2016-01-19	

SQL: Foreign Key

users

id	first_name	...
1	Janet	...
2	John	...

posts

id	title	body	published_date	user_id
1	Title 1	Text 1	2016-01-20	**1**
2	Title 2	Text 2	2016-01-11	**2**
3	Title 3	Text 3	2016-01-14	**1**
4	Title 4	Text 4	2016-01-06	**2**
5	Title 5	Text 5	2016-01-19	**1**

SQL: Foreign Key

users

id	first_name	...
1	Janet	...
2	John	...

posts

id	title	body	published_date	user_id
1	Title 1	Text 1	2016-01-20	1
2	Title 2	Text 2	2016-01-11	2
3	Title 3	Text 3	2016-01-14	1
4	Title 4	Text 4	2016-01-06	2
5	Title 5	Text 5	2016-01-19	1

Query: All of Janets posts

SQL: Foreign Key

All of Janets posts:

```
SELECT * FROM posts WHERE user_id == 1;
```

id	first_name	...
1	Janet	...
2	John	...

id	title	body	published_date	user_id
1	Title 1	Text 1	2016-01-20	1
2	Title 2	Text 2	2016-01-11	2
3	Title 3	Text 3	2016-01-14	1
4	Title 4	Text 4	2016-01-06	2
5	Title 5	Text 5	2016-01-19	1

SQL: Foreign Key

```
SELECT * FROM posts WHERE user_id == 1;
```

Result:

id	title	body	published_date	user_id
1	Title 1	Text 1	2016-01-20	1
3	Title 3	Text 3	2016-01-14	1
5	Title 5	Text 5	2016-01-19	1

SQL: Foreign Key

Foreign keys are columns that reference rows in another table.
User ID in our case is a foreign key on posts that references a user row.
With this concept, we are able to connect records.

SQL: Foreign Key

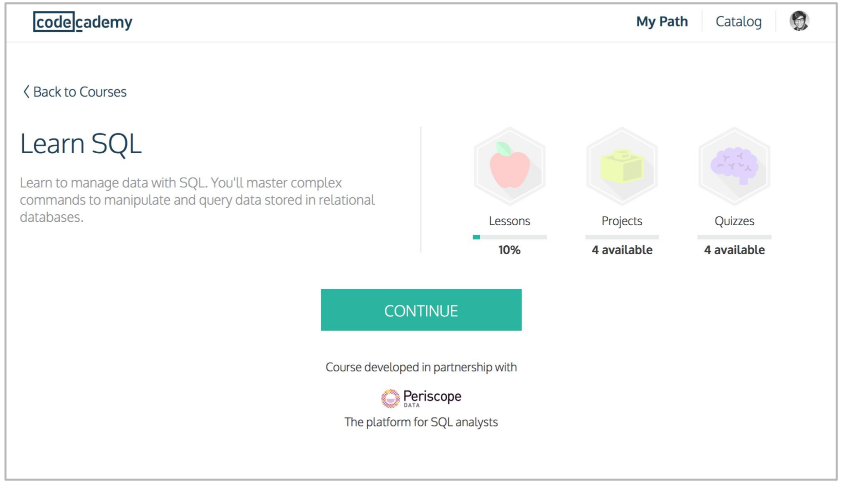
We can also put rules on this, for example:

“When I delete a user row, also delete all their posts”

We don’t cover how to do this just yet.

Exercises

Experiment: Codecademy SQL course



codecademy My Path Catalog

Back to Courses

Learn SQL

Learn to manage data with SQL. You'll master complex commands to manipulate and query data stored in relational databases.

Lessons 10%

Projects 4 available

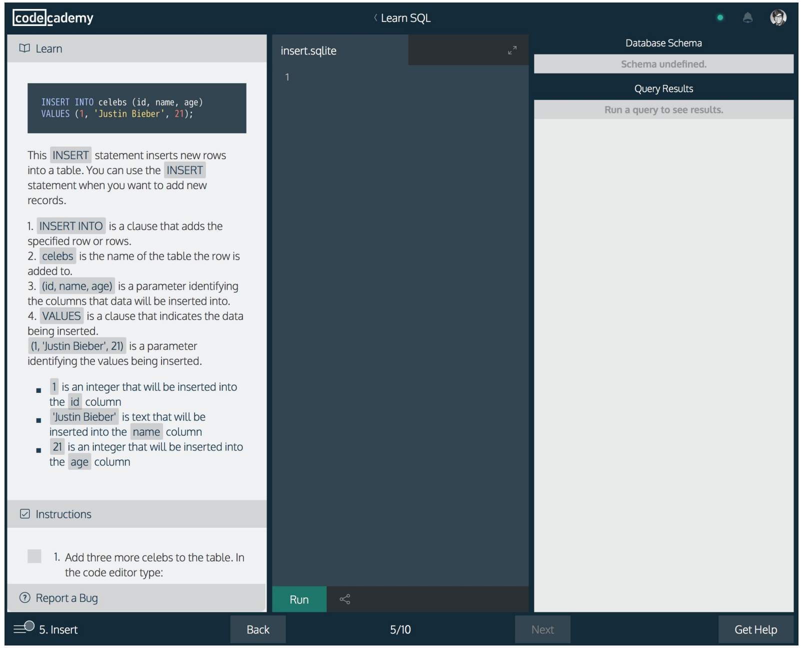
Quizzes 4 available

CONTINUE

Course developed in partnership with

Periscope DATA

The platform for SQL analysts



codecademy ‹ Learn SQL

Learn

```
INSERT INTO celebs (id, name, age)
VALUES (1, 'Justin Bieber', 21);
```

This INSERT statement inserts new rows into a table. You can use the INSERT statement when you want to add new records.

1. INSERT INTO is a clause that adds the specified row or rows.
2. celebs is the name of the table the row is added to.
3. (id, name, age) is a parameter identifying the columns that data will be inserted into.
4. VALUES is a clause that indicates the data being inserted.
(1, 'Justin Bieber', 21) is a parameter identifying the values being inserted.

- 1 is an integer that will be inserted into the id column
- 'Justin Bieber' is text that will be inserted into the name column
- 21 is an integer that will be inserted into the age column

Instructions

1. Add three more celebs to the table. In the code editor type:

Report a Bug

insert.sqlite

Run

Database Schema

Schema undefined.

Query Results

Run a query to see results.

5. Insert Back 5/10 Next Get Help

Your feedback, please?

http://goo.gl/forms/rUrZqOPNq6 (Session 28)

Time to practice