Get information on all table names in the current database, while limiting your query to the 'public' table\_schema.

Now have a look at the columns in university\_professors by selecting all entries in information\_schema.columns that correspond to that table.

Finally, print the first five rows of the university\_professors table.

-- Query the right table in information\_schema

SELECT table\_name

FROM information\_schema.tables

-- Specify the correct table\_schema value

WHERE table\_schema = 'public';

-- Query the right table in information\_schema to get columns

SELECT column\_name, data\_type

FROM information\_schema.columns

WHERE table\_name = 'university\_professors' AND table\_schema = 'public';

-- Query the first five rows of our table

SELECT \*

FROM university\_professors

LIMIT 5;

-- Create a table for the professors entity type

Create a table universities with three text columns: university\_shortname, university, and university\_city.

CREATE TABLE professors (

firstname text,

lastname text

);

-- Print the contents of this table

SELECT \*

FROM professors

-- Create a table for the universities entity type

CREATE TABLE universities(

university\_shortname text,

university text,

university\_city text);

-- Print the contents of this table

SELECT \*

FROM universities

Alter professors to add the text column university\_shortname.

-- Add the university\_shortname column

ALTER TABLE professors

ADD COLUMN university\_shortname text;

-- Print the contents of this table

SELECT \*

FROM professors

Rename the organisation column to organization in affiliations.

Delete the university\_shortname column in affiliations.

-- Rename the organisation column

ALTER TABLE affiliations

RENAME COLUMN organisation TO organization;

-- Delete the university\_shortname column

ALTER TABLE affiliations

DROP COLUMN university\_shortname;

* Insert all DISTINCT professors from university\_professors into professors.
* Print all the rows in professors.

-- Insert unique professors into the new table

INSERT INTO professors

SELECT DISTINCT firstname, lastname, university\_shortname

FROM university\_professors;

-- Doublecheck the contents of professors

SELECT \*

FROM professors;

Insert all DISTINCT affiliations into affiliations from university\_professors.

-- Insert unique affiliations into the new table

INSERT INTO affiliations

SELECT DISTINCT firstname, lastname, function, organization

FROM university\_professors;

-- Doublecheck the contents of affiliations

SELECT \*

FROM affiliations;

Delete the university\_professors table.

-- Delete the university\_professors table

DROP TABLE university\_professors;

* Execute the given sample code.
* As it doesn't work, have a look at the error message and correct the statement accordingly – then execute it again.

-- Let's add a record to the table

INSERT INTO transactions (transaction\_date, amount, fee)

VALUES ('2018-09-24', 5454, '30');

-- Doublecheck the contents

SELECT \*

FROM transactions;

* Execute the given sample code.
* As it doesn't work, add an integer type cast at the right place and execute it again.

-- Calculate the net amount as amount + fee

SELECT transaction\_date, amount + CAST(fee AS integer) AS net\_amount

FROM transactions;

Have a look at the distinct university\_shortname values in the professors table and take note of the length of the strings.

Now specify a fixed-length character type with the correct length for university\_shortname.

-- Select the university\_shortname column

SELECT DISTINCT(university\_shortname)

FROM professors;

-- Specify the correct fixed-length character type

ALTER TABLE professors

ALTER COLUMN university\_shortname

TYPE char(3);

Change the type of the firstname column to varchar(64).

-- Change the type of firstname

ALTER TABLE professors

ALTER COLUMN firstname

TYPE varchar(64);

* Run the sample code as is and take note of the error.
* Now use SUBSTRING() to reduce firstname to 16 characters so its type can be altered to varchar(16).

-- Convert the values in firstname to a max. of 16 characters

ALTER TABLE professors

ALTER COLUMN firstname

TYPE varchar(16)

USING SUBSTRING(firstname FROM 1 FOR 16)

Add a not-null constraint for the firstname column.

-- Disallow NULL values in firstname

ALTER TABLE professors

ALTER COLUMN firstname SET NOT NULL;

Add a not-null constraint for the lastname column.

-- Disallow NULL values in lastname

ALTER TABLE professors

ALTER COLUMN lastname SET NOT NULL;

Add a unique constraint to the university\_shortname column in universities. Give it the name university\_shortname\_unq.

-- Make universities.university\_shortname unique

ALTER TABLE universities

ADD CONSTRAINT university\_shortname\_unq UNIQUE(university\_shortname);

Add a unique constraint to the organization column in organizations. Give it the name organization\_unq.

-- Make organizations.organization unique

ALTER TABLE organizations

ADD CONSTRAINT organization\_unq UNIQUE(organization);

First, find out the number of rows in universities.

-- Count the number of rows in universities

SELECT COUNT(\*)

FROM universities;

Then, find out how many unique values there are in the university\_city column.

Using the above steps, identify the *candidate key* by trying out different combination of columns.

-- Try out different combinations

SELECT COUNT(DISTINCT(firstname, lastname))

FROM professors;

* Rename the organization column to id in organizations.
* Make id a primary key and name it organization\_pk.

-- Rename the organization column to id

ALTER TABLE organizations

RENAME COLUMN organization TO id;

-- Make id a primary key

ALTER TABLE organizations

ADD CONSTRAINT organization\_pk PRIMARY KEY (id);

* Rename the university\_shortname column to id in universities.
* Make id a primary key and name it university\_pk.

-- Rename the university\_shortname column to id

ALTER TABLE universities

RENAME COLUMN university\_shortname TO id;

-- Make id a primary key

ALTER TABLE universities

ADD CONSTRAINT university\_pk PRIMARY KEY(id);

Add a new column id with data type serial to the professors table.Make id a primary key and name it professors\_pkey.

Write a query that returns all the columns and 10 rows from professors.

-- Add the new column to the table

ALTER TABLE professors

ADD COLUMN id serial;

-- Make id a primary key

ALTER TABLE professors

ADD CONSTRAINT professors\_pkey PRIMARY KEY (id);

-- Have a look at the first 10 rows of professors

SELECT \* FROM professors

LIMIT 10;

Count the number of distinct rows with a combination of the make and model columns.Add a new column id with the data type varchar(128).

Concatenate make and model into id using an UPDATE table\_name SET column\_name = ... query and the CONCAT() function.

Make id a primary key and name it id\_pk.

-- Count the number of distinct rows with columns make, model

SELECT COUNT(DISTINCT(make, model))

FROM cars;

-- Add the id column

ALTER TABLE cars

ADD COLUMN id varchar(128);

-- Update id with make + model

UPDATE cars

SET id = CONCAT(make, model);

-- Make id a primary key

ALTER TABLE cars

ADD CONSTRAINT id\_pk PRIMARY KEY (id);

-- Have a look at the table

SELECT \* FROM cars;

* Given the above description of a student entity, create a table students with the correct column types.
* Add a PRIMARY KEY for the social security number ssn.

*Note that there is no formal length requirement for the integer column. The application would have to make sure it's a correct SSN!*

-- Create the table

CREATE TABLE students (

last\_name varchar(128) NOT NULL,

ssn integer PRIMARY KEY,

phone\_no char(12)

);

Rename the university\_shortname column to university\_id in professors.

-- Rename the university\_shortname column

ALTER TABLE professors

RENAME COLUMN university\_shortname TO university\_id;

* Add a foreign key on university\_id column in professors that references the id column in universities.
* Name this foreign key professors\_fkey.

-- Rename the university\_shortname column

ALTER TABLE professors

RENAME COLUMN university\_shortname TO university\_id;

-- Add a foreign key on professors referencing universities

ALTER TABLE professors

ADD CONSTRAINT professors\_fkey FOREIGN KEY (university\_id) REFERENCES universities (id);

-- Try to insert a new professor

INSERT INTO professors (firstname, lastname, university\_id)

VALUES ('Albert', 'Einstein', 'UZH');

* JOIN professors with universities on professors.university\_id = universities.id, i.e., retain all records where the foreign key of professors is equal to the primary key of universities.
* Filter for university\_city = 'Zurich'.

-- Select all professors working for universities in the city of Zurich

SELECT professors.lastname, universities.id, universities.university\_city

FROM professors

JOIN universities

ON professors.university\_id = universities.id

WHERE universities.university\_city = 'Zurich';

Add a professor\_id column with integer data type to affiliations, and declare it to be a foreign key that references the id column in professors.

Rename the organization column in affiliations to organization\_id.

Add a foreign key constraint on organization\_id so that it references the id column in organizations.

-- Add a professor\_id column

ALTER TABLE affiliations

ADD COLUMN professor\_id integer REFERENCES professors (id);

-- Rename the organization column to organization\_id

ALTER TABLE affiliations

RENAME organization TO organization\_id;

-- Add a foreign key on organization\_id

ALTER TABLE affiliations

ADD CONSTRAINT affiliations\_organization\_fkey FOREIGN KEY (organization\_id) REFERENCES organizations (id);

First, have a look at the current state of affiliations by fetching 10 rows and all columns.

-- Have a look at the 10 first rows of affiliations

SELECT \* FROM affiliations

LIMIT 10;

Update the professor\_id column with the corresponding value of the id column in professors.  
*"Corresponding" means rows in professors where the firstname and lastname are identical to the ones in affiliations.*

0Check out the first 10 rows and all columns of affiliations again. Have the professor\_ids been correctly matched?

-- Update professor\_id to professors.id where firstname, lastname correspond to rows in professors

UPDATE affiliations

SET professor\_id = professors.id

FROM professors

WHERE affiliations.firstname = professors.firstname AND affiliations.lastname = professors.lastname;

-- Have a look at the 10 first rows of affiliations again

SELECT \* FROM affiliations

LIMIT 10;

Drop the firstname and lastname columns from the affiliations table.

-- Drop the firstname column

ALTER TABLE affiliations

DROP COLUMN firstname;

-- Drop the lastname column

ALTER TABLE affiliations

DROP COLUMN lastname;

Have a look at the existing foreign key constraints by querying table\_constraints in information\_schema.

Delete the affiliations\_organization\_id\_fkey foreign key constraint in affiliations.Run the DELETE and SELECT queries to double check that the deletion cascade actually works.

-- Identify the correct constraint name

SELECT constraint\_name, table\_name, constraint\_type

FROM information\_schema.table\_constraints

WHERE constraint\_type = 'FOREIGN KEY';

-- Drop the right foreign key constraint

ALTER TABLE affiliations

DROP CONSTRAINT affiliations\_organization\_id\_fkey;

-- Add a new foreign key constraint from affiliations to organizations which cascades deletion

ALTER TABLE affiliations

ADD CONSTRAINT affiliations\_organization\_id\_fkey FOREIGN KEY (organization\_id) REFERENCES organizations (id) ON DELETE CASCADE;

-- Delete an organization

DELETE FROM organizations

WHERE id = 'CUREM';

-- Check that no more affiliations with this organization exist

SELECT \* FROM affiliations

WHERE organization\_id = 'CUREM';

* Count the number of total affiliations by university.
* Sort the result by that count, in descending order.

-- Count the total number of affiliations per university

SELECT COUNT(\*), professors.university\_id

FROM affiliations

JOIN professors

ON affiliations.professor\_id = professors.id

-- Group by the ids of professors

GROUP BY professors.university\_id

ORDER BY count DESC;

Join all tables in the database (starting with affiliations, professors, organizations, and universities) and look at the result.

-- Join all tables

SELECT \*

FROM affiliations

JOIN professors

ON affiliations.professor\_id = professors.id

JOIN organizations

ON affiliations.organization\_id = organizations.id

JOIN universities

ON professors.university\_id = universities.id;

* Now group the result by organization sector, professor, and university city.
* Count the resulting number of rows.

-- Filter the table and sort it

SELECT COUNT(\*), organizations.organization\_sector,

professors.id, universities.university\_city

FROM affiliations

JOIN professors

ON affiliations.professor\_id = professors.id

JOIN organizations

ON affiliations.organization\_id = organizations.id

JOIN universities

ON professors.university\_id = universities.id

WHERE organizations.organization\_sector = 'Media & communication'

GROUP BY organizations.organization\_sector,

professors.id, universities.university\_city

ORDER BY count DESC;