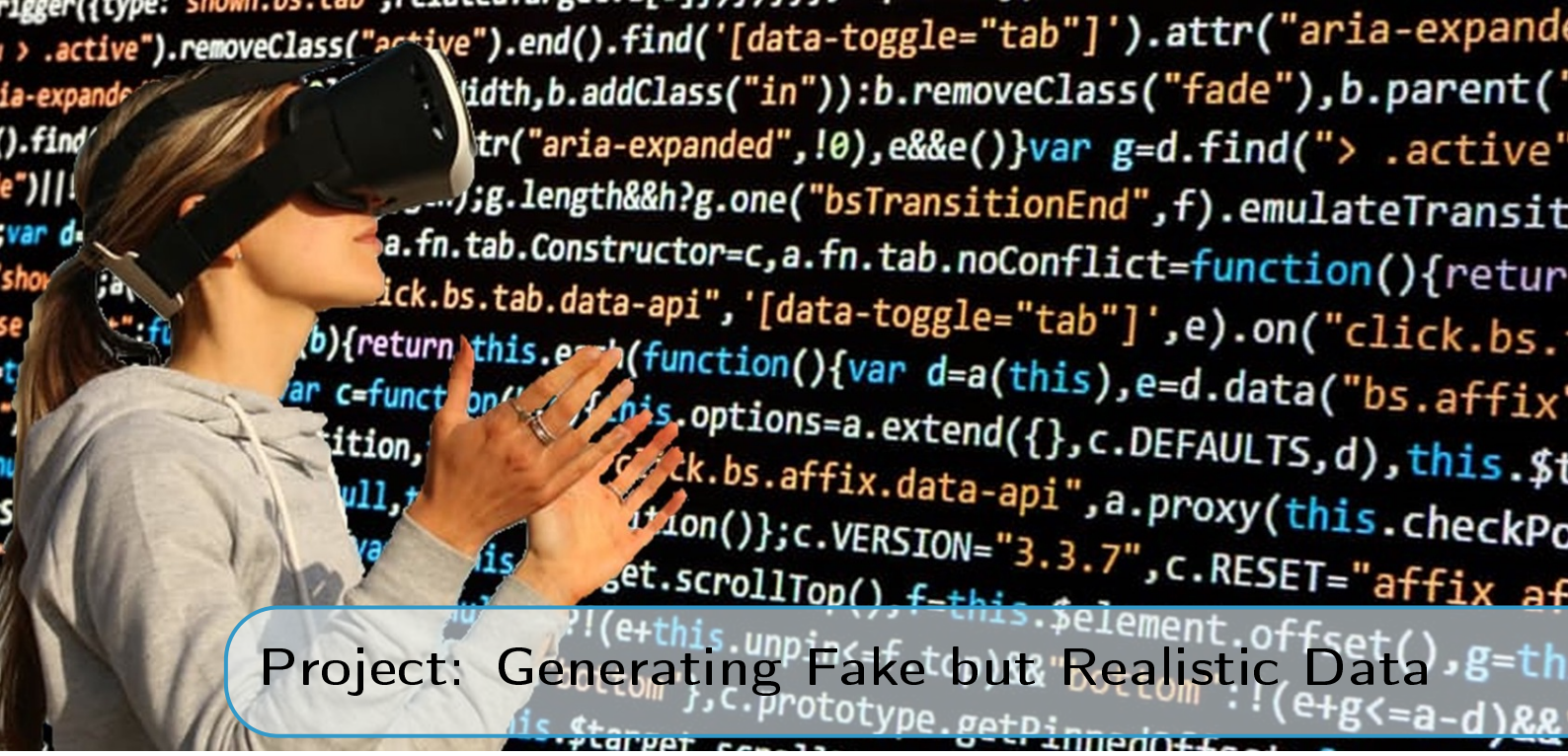


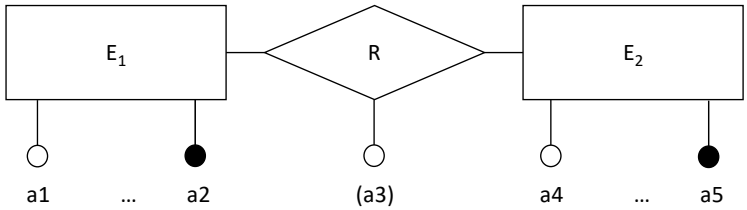
Project: Generating Fake but Realistic Data

Instructions

- This project is an individual project.
- Submit your answers by **Friday 27 August 2021, 17:00** to Luminus.
- There is strictly no possibility of late submission.
- Download the template answer file "`answers.sql`" from
  "`Files > Projects > Generating Fake but Realistic Data`".
- Write your answers in the indicated sections of the file "`answers.sql`" .
- Submit the completed file "`answers.sql`" to
  "`Files > Projects > Generating Fake but Realistic Data > Submissions`".
- Do not submit other files.
- Follow the naming requirements, include your student number and write your answers the file "`answers.sql`" as instructed.

The generation of fake but realistic database instances is important for both performance evaluation and demonstration purposes. In this project we use `mockaroo.com` to create such a database instance.

The following entity-relationship diagram translates into three tables.



Explore the types of data that `mockaroo.com` can generate and the available options.

Create a case. Choose an original example application domain to instantiate the entity-relationship diagram. For instance, you could have chosen (not any more) the entity set E_1 to be `person`, the entity set E_2 to be `company`, and the optional many-to-many relationship set R to be `worksfor`. You may add attributes to make the example more realistic. Design the corresponding three tables. For instance, create a table `person` (name the tables and their attributes according to the example you chose) for persons with their emails and names, a table `company` for companies, with their names and addresses, and a table `worksfor` associating the emails of the persons to the names of the companies for which they are working. Each entity should have at least one primary key and one other attribute. The relationship needs not have attributes but may if it makes the example more interesting (it makes Question 1.e accordingly more difficult).

The SQL code that you submit should run without error in PostgreSQL version 13 or SQLite version 3 or above.

1. (4 points) Present and create the case.
   (a) Indicate for which of PostgreSQL or SQLite your code for the following questions is written and tested.
   (b) Introduce the chosen example and describe the content of the three tables in English. Indicate which of PostgreSQL or SQLite your code for the following questions is written for.
   (c) Create the three tables with appropriate domains and integrity constraints. Give the corresponding three SQL DDL code (`CREATE TABLE` statements.)
   (d) Populate the two tables corresponding to the two entity sets. Each of the two tables should contain approximately 100 rows. Use `mockaroo.com` to create fake but realistic data according to the application domain that you have chosen. Give the corresponding SQL DML code (`INSERT` statements - notice that `mockaroo.com` offers the option to generate the `INSERT` statements.)
   (e) Populate the table corresponding to the relationship. The table should contain approximately 1000 rows (or 10%, chosen at random, of the possible relationships.) Use one single SQL `INSERT` statement to do so (the statement inserts the result of a query). Consider using the `random()` function. Note that the function behaves differently in PostgreSQL and SQLite. Give the corresponding SQL DML code (`INSERT` statement.)
2. (1 point) The most interesting original cases get one mark at the discretion of the markers.

– END OF PAPER –