Lab 1 - SQL

Objective: to practice writing SQL queries.

To run this lab as a `jupyter` notebook, you can download it here (the zip-file contains the notebook and the database).

Background

We have a database to handles the academic achievements of students at LTH – in it we have three tables:

- `students` – contains student data:
  - `ssn` – social security number ('personnummer')
  - `first_name`
  - `last_name`
- `courses` – describes the courses:
  - `course_code`
  - `course_name`
  - `level` ("G1", "G2", or "A")
  - `credits`
- `taken_courses` – keeps track of which courses the students have taken, once a student has passed a course, we add a row in this table:
  - `ssn` – the social security number of the student
  - `course_code` – what course has been taken
  - `grade`

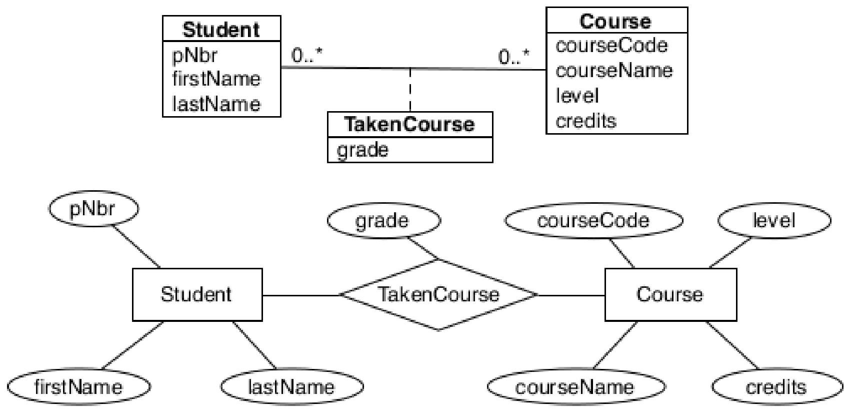

Figure 1: There should be an image here

Some sample data:

```
ssn           first_name   last_name
```

```
---           ----------   ---------
861103-2438   Bo           Ek
911212-1746   Eva          Alm
950829-1848   Anna         Nyström
...           ...          ...

course_code   course_name                 level    credits
-----------   -----------                 -----    -------
EDA016        Programmeringsteknik        G1       7.5
EDAA01        Programmeringsteknik - FK   G1       7.5
EDA230        Optimerande kompilatorer    A        7.5
...           ...                         ...      ...

ssn           course_code   grade
---           -----------   -----
861103-2438   EDA016        4
861103-2438   EDAA01        3
911212-1746   EDA016        3
...           ...           ...
```

The tables have been created with the following SQL statements:

```sql
CREATE TABLE students (
  ssn          CHAR(11),
  first_name   TEXT NOT NULL,
  last_name    TEXT NOT NULL,
  PRIMARY KEY  (ssn)
);

CREATE TABLE courses (
  course_code   CHAR(6),
  course_name   TEXT NOT NULL,
  level         CHAR(2),
  credits       DOUBLE NOT NULL CHECK (credits > 0),
  PRIMARY KEY   (course_code)
);

CREATE TABLE taken_courses (
  ssn           CHAR(11),
  course_code   CHAR(6),
  grade         INTEGER NOT NULL CHECK (grade >= 3 AND grade <= 5),
  PRIMARY KEY   (ssn, course_code),
  FOREIGN KEY   (ssn) REFERENCES students(ssn),
  FOREIGN KEY   (course_code) REFERENCES courses(course_code)
);
```

All courses offered at the “Computer Science and Engineering” program at LTH during the academic year 2013/14 are in the table 'courses'. Also, the database has been filled with made up data. SQL statements like the following have been used to insert the data:

```
INTO   students (ssn, first_name, last_name)
VALUES ('950705-2308', 'Anna', 'Johansson'),
       ('930702-3582', 'Anna', 'Johansson'),
       ('911212-1746', 'Eva', 'Alm'),
       ('910707-3787', 'Eva', 'Nilsson'),
       ...
```

Assignments

```
%load_ext sql
```

```
%sql sqlite:///lab1.sqlite
```

The tables `students`, `courses` and `taken_courses` already exist in your database. If you change the contents of the tables, you can always recreate the tables with the following command (at the mysql prompt):

```
sqlite3 lab1.db < setup-lab1-db.sql
```

After some of the questions there is a number in brackets. This is the number of rows generated by the question. For instance, [72] after question a) means that there are 72 students in the database.

a) What are the names (first name, last name) of all the students? [72]

```
%%sql
```

b) Same as question a) but produce a sorted listing. Sort first by last name and then by first name.

```
%%sql
```

c) What are the names of the students who were born in 1985? [4]

```
%%sql
```

d) The next-to-last digit in the social security number is even for females, and odd for males. List the names of all female students in our database. Hint: the `SUBSTR` function can be useful. [26]

```
%%sql
```

e) How many students are registered in the database?

```
%%sql
```

f) Which courses are offered by the department of Mathematics (their course codes have the form `FMAxxx`)? [22]

```
%%sql
```

g) Which courses give more than 7.5 credits? [16]

```
%%sql
```

h) How may courses are there for each level (`G1`, `G2`, and `A`)?

```
%%sql
```

i) Which courses (course codes only) have been taken by the student with social security number 910101–1234? [35]

```
%%sql
```

j) What are the names of these courses, and how many credits do they give?

```
%%sql
```

k) How many credits has the student taken?

```
%%sql
```

l) Which is the student's grade average?

```
%%sql
```

m) Which students have taken 0 credits? [11]

```
%%sql
```

n) List the names and average grades of the 10 students with the highest grade average?

```
%%sql
```

o) List the social security number and total number of credits for all students. Students with no credits should be included with 0 credits, not null. If you do this with an outer join you might want to use the function `COALESCE(v1, v2, ...)`; it returns the first value which is not `NULL`. (It is a little bit tricky to get this query right, if you're missing the students with 0 credits, don't worry, your TA will help you get it right). [72]

```
%%sql
```

p) Is there more than one student with the same name? If so, who are these students and what are their social security numbers? [7]

```
%%sql
```

q) What 5 courses have the highest grade average?

```
%%sql
```

r) (Not required) What are the 'best' three first initial letters of the last names, i.e., if you take the average grades for each first letter of the last name, which three initials have the highest averages?

```
%%sql
```