Exercise 3

Publication: 15.03.2021
Publication of Solutions: 22.03.2021

The exercise is based on the *Mondial* database.

Connection to PostgreSQL server

Connect to your PostgreSQL account, e.g., via pgAdmin version 4 (any version should work) and import the *Mondial* database. A tutorial for pgAdmin4 can be found online.[1]

1 DRL - SQL to natural language

Consider the following SQL queries and write down in natural language what the queries return:

1.
```
select name
from country join encompasses on code = country
where continent = 'Europe';
```

2.
```
select C.name, C.area
from country C
where not exists (
        select *
        from religion R
        where R.country = C.code and
              R.percentage between 50 and 100);
```

3.
```
select sum(C.population)
from country C join encompasses E on C.code = E.country
where E.continent = 'Asia';
```

4.
```
select C.capital
from country C join economy E on C.code = E.country
where C.name like '%a%'
    and E.agriculture between 50 and 80
    and E.inflation is not NULL;
```

5. The next query is split into three parts to make it easier to understand. For (a) and (b) replace the view names with meaningful names. For (c) write down in natural language what the query returns similar to the previous exercises.

   (a)
   ```
   create view a as
   select EC.country
   from economy EC
   where EC.gdp > (
       select avg(E.gdp)
       from economy E);
   ```

[1] https://files.ifi.uzh.ch/dbtg/dbs/pgAdmin/tutorial_pgAdmin4.pdf

(b)
```
create view b as
select country
from ethnicGroup
group by country
having count(name) = 3;
```

(c)
```
select count(b.country)
from a join b on a.country = b.country;
```

2 DRL - Writing SQL Queries

For the following queries, (a) formulate and write down the query in SQL, and (b) execute it on your *Mondial* database. Make sure your expressions are correct for all Mondial database instances.

1. The average population of all countries which contain a desert partially or fully. (The SQL query returns exactly 85263969.461538461538 when executed on the *Mondial* database)
2. The area of the second largest island. (The SQL query returns exactly 786000 when executed on the *Mondial* database)
3. The codes and the areas of all countries that have a major religion (the percentage of the religion is greater than 50%) and a major ethnic group (the percentage of the ethnic group is greater than 50%). (The SQL query returns exactly 88 rows when executed on the *Mondial* database)
4. The names of all seas that merge into a deeper sea. (The SQL query returns exactly 31 rows when executed on the *Mondial* database)
5. The names and abbreviations of all organizations whose headquarter is not located in a capital city. (The SQL query returns exactly 68 rows when executed on the *Mondial* database)
6. The names of all provinces with at least one million inhabitants through which exactly five rivers flow. (The SQL query returns exactly 3 rows when executed on the *Mondial* database)
7. The codes of all countries that have at least one neighboring country and for each of these country codes the number of neighboring countries whose country codes' start with an 'A' and number of neighboring countries whose country codes' start with a 'Z'. (The SQL query returns exactly 164 rows when executed on the *Mondial* database)

3 DRL - Understanding SQL Queries

Consider the following query **Q** expressed in natural language:

Q: The names of all countries exactly once that border to at least one country and at most two countries.

1. Consider the following incomplete SQL statement to answer query **Q**. Replace the placeholders such that the complete SQL query returns the correct result.

```
with symBorders(country1, country2) as (
  select country1, country2 from borders
  union
  select country2, country1 from borders
)
select distinct name
from country join symBorders on code = country1
group by  <placeholder1>
having  <placeholder2> ;
```

2. Query **Q** can also be solved without grouping and aggregation. Consider the following incomplete SQL statement to answer query **Q**. Replace <placeholder> with a subquery such that the complete SQL query returns the correct result. No grouping or aggregation is allowed.

```
with symBorders(country1, country2) as (
  select country1, country2 from borders
  union
  select country2, country1 from borders
)
select distinct name
from country join symBorders on code = country1
where code not in ( <placeholder> );
```

Consider the following query **Q** expressed in natural language:

Q: The country codes of all countries which are directly or indirectly connected to Switzerland over any number of countries that share a border with each other. 'CH' should not be included in the output of the query. (e.g. Austria is directly sharing a border with Switzerland and Russia is indirectly connected to Switzerland over the countries Belarus, Poland and Germany.)

Consider the following incomplete SQL statement to answer query **Q**. Replace the placeholders such that the complete SQL query returns the correct result.

```
create view symBorders(country1, country2) as (
  select country1, country2 from borders
  union
  select country2, country1 from borders
);

with recursive countryCodes(country) as (
        select country2
        from symBorders
        where country1 = 'CH'
        union
        ( <placeholder1> )
)
select *
from countryCodes
where ( <placeholder2> )
```

What happens if **union** is replaced by **union all**?