Accessing an SQL database from R

Accessing an SQL database from R

In this laboratory you will learn how to connect to a MariaDB / MySQL database from R, send `SELECT` queries and work with the results as regular R data frames. We will use the public `world` database on the teaching server and repeat most of the `SELECT` exercises from the SQL lab, but in the context of R.

The database server is:

- host: `info3.meil.pw.edu.pl`
- database: `world`
- user: `statistics`
- password: `statistics`

1. Connecting to MariaDB from R

1.1 Required packages

To connect from R we will use the `DBI` interface and the `RMariaDB` (or `RMySQL`) driver.

```
install.packages("RMariaDB")    # or RMySQL if preferred
```

Then load the packages:

```
library(RMariaDB)
```

1.2 Opening a connection

Create a connection object to the `world` database:

```
con <- dbConnect(
  RMariaDB::MariaDB(),
  host     = "info3.meil.pw.edu.pl",
  dbname   = "world",
  user     = "statistics",
  password = "statistics"
)
```

If the connection is successful, `con` will be a DBI connection object. You can list tables in the database:

```
dbListTables(con)
```

and list columns of a given table, for example `Country`:

```
dbListFields(con, "Country")
```

At the end of the session, always close the connection:

```
dbDisconnect(con)
```

In the rest of this instruction we assume that `con` is an open connection to the `world` database.

2. Basic `SELECT` queries from R

To run a `SELECT` query and get the results as a data frame, use `dbGetQuery(con, "SQL HERE")`.

2.1 Selecting all columns

SQL task: display all information contained in the `Country` table.

```
qry <- "SELECT * FROM Country;"
country <- dbGetQuery(con, qry)
head(country)
str(country)
```

Task 2.1

1. Run the code above and examine the structure of the `country` data frame.
2. Check how many rows and columns it has using `dim(country)`.

2.2 Selecting specific columns

SQL task: display all values for two columns, for example `Name` and `Region`.

```
qry <- "SELECT Name, Region FROM Country;"
country_nr <- dbGetQuery(con, qry)
head(country_nr)
```

Task 2.2
Modify the query so that it returns only `Name` and `Continent`.

2.3 Filtering rows with `WHERE`

SQL task: display names of all countries in Europe together with life expectancy.

```
qry <- "
SELECT Name, LifeExpectancy
FROM Country
WHERE Continent = 'Europe';
"
eu_life <- dbGetQuery(con, qry)
head(eu_life)
```

Task 2.3

1. Count how many European countries are in the result using `nrow(eu_life)`.
2. Compute the average life expectancy in Europe using `mean(eu_life$LifeExpecta`
   `na.rm = TRUE)`.

2.4 Filtering with `IN`

SQL task: display names of all countries in Europe and Asia with life expectancy.

```
qry <- "
SELECT Name, LifeExpectancy
FROM Country
WHERE Continent IN ('Europe', 'Asia');
"
ea_life <- dbGetQuery(con, qry)
head(ea_life)
```

Task 2.4
Using R, compute separate averages of `LifeExpectancy` for Europe and for Asia from `ea_life`.
Hint: you can use `tapply(ea_life$LifeExpectancy, ea_life$Continent, mean, na.rm = TRUE)`.

2.5 Ordering results

SQL task: sort the information from the previous point by life expectancy.

```
qry <- "
SELECT Name, Continent, LifeExpectancy
FROM Country
WHERE Continent IN ('Europe', 'Asia')
ORDER BY LifeExpectancy DESC;
"
ea_sorted <- dbGetQuery(con, qry)
head(ea_sorted)
tail(ea_sorted)
```

Task 2.5
Identify: - the country with the highest life expectancy in Europe or Asia, - the country with the lowest life expectancy in these continents.

2.6 Aggregate functions: `SUM`, `AVG`

SQL task: display the total population and the average population of countries in Europe.

```
qry_sum <- "
SELECT SUM(Population) AS total_pop
FROM Country
WHERE Continent = 'Europe';
"
dbGetQuery(con, qry_sum)

qry_avg <- "
SELECT AVG(Population) AS avg_pop
FROM Country
WHERE Continent = 'Europe';
"
dbGetQuery(con, qry_avg)
```

Task 2.6

1. Store the result of `qry_sum` in an object and extract the numeric value (e.g. `res$total_pop`).
2. Do the same for `qry_avg`.

2.7 Pattern matching with `LIKE`

SQL task: display names and codes of all countries whose names start with `"Ch"`.

```
qry <- "
SELECT Name, Code
FROM Country
WHERE Name LIKE 'Ch%';
"
ch <- dbGetQuery(con, qry)
ch
```

Task 2.7
Modify the query to select all countries whose names contain the substring `"land"` (use `'%land%'`).

2.8 Working with the `City` table

SQL task: display all cities in Finland (country code `"FIN"`).

```
qry <- "
SELECT *
FROM City
WHERE CountryCode = 'FIN';
"
fin_cities <- dbGetQuery(con, qry)
head(fin_cities)
```

Task 2.8

1. Count how many Finnish cities are in the table.
2. Find the city with the largest population in Finland (you can either use SQL `ORDER BY` or sort in R using `fin_cities[order(-fin_cities$Population), ]`).

SQL task: display all cities in Poland, sorted by district.

```
qry <- "
SELECT *
FROM City
WHERE CountryCode = 'POL'
ORDER BY District;
"
pol_cities <- dbGetQuery(con, qry)
head(pol_cities)
```

Task 2.9
Using R, create a table showing, for each district in Poland (`District`), how many cities are listed in the table.
Hint: `table(pol_cities$District)`.

2.9 Conditions on numeric columns

SQL task: display names of countries that became independent after 1980.

```
qry <- "
SELECT Name, IndepYear
FROM Country
```

```
WHERE IndepYear > 1980;
"
indep_1980 <- dbGetQuery(con, qry)
head(indep_1980)
```

Task 2.10

1. How many countries in the result have `IndepYear` not `NA`?
2. Compute the earliest and latest independence year in this group.

SQL task: display names of North American countries that became independent between 1800 and 1900, sorted by `IndepYear`.

```
qry <- "
SELECT Name, IndepYear
FROM Country
WHERE Continent = 'North America'
  AND IndepYear > 1800
  AND IndepYear < 1900
ORDER BY IndepYear;
"
na_indep <- dbGetQuery(con, qry)
na_indep
```

Task 2.11
Using R, compute the time span (difference between maximum and minimum `IndepYear`) for these countries.

3. More advanced `SELECT` queries

3.1 Large cities and sorting

SQL task: display names of cities with population greater than 3,000,000, together with country codes and population, sorted by country code (descending) and population (descending).

```
qry <- "
SELECT Name, CountryCode, Population
FROM City
WHERE Population > 3000000
ORDER BY CountryCode DESC, Population DESC;
"
big_cities <- dbGetQuery(con, qry)
head(big_cities)
```

Task 3.1

1. How many such cities are there?
2. Using R, compute the average population of these cities.

3.2 Nested queries (subqueries)

SQL task: display all cities in Norway, without knowing the `CountryCode` in advance.

```
qry <- "
SELECT *
FROM City
WHERE CountryCode = (
  SELECT Code
  FROM Country
  WHERE Name = 'Norway'
);
"
norway_cities <- dbGetQuery(con, qry)
head(norway_cities)
```

Task 3.2
Repeat the query for **Spain** and **Poland**, changing only the country name in the nested `SELECT`.

Task 3.3

Display the name and population of the most populous country in South America. Modify the query to find the least populous country in South America (use `MIN(Population)` instead of `MAX`).

3.4 Grouping and aggregation with `GROUP BY`

SQL task: display the number of countries on each continent.

```
qry <- "
SELECT Continent, COUNT(*) AS NumberOfCountries
FROM Country
GROUP BY Continent;
"
countries_by_continent <- dbGetQuery(con, qry)
countries_by_continent
```

Task 3.4

1. In R, compute the total number of countries by summing `NumberOfCountries`.
2. Identify the continent with the largest number of countries.

4. Joins between tables

Now we repeat some join-based queries in R.

4.1 European capitals

SQL task: display names of capitals of European countries using a join between `City` and `Country`.

```
qry <- "
SELECT City.Name AS CityName, Country.Name AS CountryName
FROM City
INNER JOIN Country
  ON City.ID = Country.Capital
WHERE Country.Continent = 'Europe';
"
eu_capitals <- dbGetQuery(con, qry)
head(eu_capitals)
```

Task 4.1

1. How many European capitals are listed?
2. Sort the result in R by `CountryName` and inspect the first and last few rows.

4.2 Languages used in European countries

SQL task: display information about languages used in European countries.

```
qry <- "
SELECT Country.Name AS CountryName,
       CountryLanguage.Language,
       CountryLanguage.IsOfficial,
       CountryLanguage.Percentage
FROM Country
INNER JOIN CountryLanguage
  ON Country.Code = CountryLanguage.CountryCode
WHERE Country.Continent = 'Europe';
"
eu_lang <- dbGetQuery(con, qry)
head(eu_lang)
```

Task 4.2

1. Using R, create a table that shows, for each language, in how many European countries it appears.
   Hint: `table(eu_lang$Language)`.
2. Create a similar table only for languages where `IsOfficial = 'T'`.

4.3 Smallest country by surface area

SQL task: display the name and surface area of the smallest country in the world.

```
qry <- "
SELECT Name, SurfaceArea
FROM Country
WHERE SurfaceArea = (
  SELECT MIN(SurfaceArea)
  FROM Country
);
"
min_country <- dbGetQuery(con, qry)
min_country
```

Task 4.3

Modify the query so that it returns the smallest country **in Africa**.

4.4 Countries and their capitals

SQL tasks: - display names of countries and names of their capitals, - display only Asian countries and their capitals, - display African countries and their capitals, sorted by country name, using table aliases.

Example with aliases:

```
qry <- "
SELECT c.Name  AS CountryName,
       ci.Name AS CapitalName
FROM Country AS c
INNER JOIN City AS ci
  ON c.Capital = ci.ID;
"
country_capital <- dbGetQuery(con, qry)
head(country_capital)
```

Task 4.4

1. Modify the query above to return only **Asian** countries (`c.Continent = 'Asia'`).
2. Modify it again to return only **African** countries, sorted by `CountryName`.

4.5 Countries where people speak a given language

SQL tasks (adapted):

- display all countries where people speak **Polish**,
- display all languages spoken in **Spain**,
- display countries that became independent after 1900 and have **Spanish** as an official language,
  then repeat for **French** and **English**.

Example for countries where people speak Polish:

```
qry <- "
SELECT c.Name AS CountryName
FROM Country AS c
INNER JOIN CountryLanguage AS cl
  ON c.Code = cl.CountryCode
WHERE cl.Language = 'Polish';
"
polish_countries <- dbGetQuery(con, qry)
polish_countries
```

Task 4.5

1. Write and run a query that lists all languages spoken in Spain.
2. Write and run a query that lists countries which became independent after 1900 (`IndepYear > 1900`) and for which `Language = 'Spanish'` and `IsOfficial = 'T'`.
3. Repeat step 2 for French and English.

5. Disconnecting

When you finish working with the database, always close the connection:

```
dbDisconnect(con)
```

If you are using RStudio, it is also a good idea to clear the environment or restart R before the next lab.