In the second project assignment, you will use the database design that you created in the first project assignment. You will design a queries that will meet the following requirements:

- Create a query that will retrieve only selected columns from the selected table
- Create a query that will select user/person or similar table based on the email.
- Create at least one UPDATE, INSERT, DELETE, and ALTER TABLE query
- Create a series of queries that will separately use the following:
  - { WHERE
  - { LIKE; NOT LIKE
  - { SUBSTRING; TRIM
  - { COUNT; SUM; MIN; MAX; AVG;
  - { GROUP BY; GROUP BY and HAVING; GROUP BY, HAVING, and WHERE;
  - { UNION ALL / UNION
  - { DISTINCT
  - { LEFT JOIN; RIGHT JOIN; FULL OUTER JOIN
- Use in one query: LEFT JOIN, GROUP BY, HAVING, ORDER BY, AVG and DISTINCT
- Create a query that will return the data from an arbitrary table for the last one and half days (1day + 12 hours, i.e., 36 hours). Do not hard code the query (e.g., created_at > 7-11-2021)!
  - { Do it programmatically with DATE functions.
- Create a query that will return data from the last month (starting from the first day of the month)
- Write a select that will remove accents on a selected string (e.g., á will be converted to a)
  - { Beforehand, you will need to save data that contain accents in the database (e.g., save some Czech surname in the database that has accents)
- Create a query for pagination in an application (use LIMIT and OFFSET)
- Create a query that will use subquery in FROM
- Create a query that will use subquery in WHERE condition
- Create a query that will use any aggregate function and GROUP BY with HAVING
- Create a query that will join at least five tables
- Create a query that will join at least three tables and will use GROUP BY, HAVING, and WHERE
- Modify the database from the first project assignment to improve integrity constraints (e.g., reduce the size for `varchar` columns)
  - { Set cascading, explain places where you used cascading and why?
- Create database indexes (create it only on columns where it can make a sense – explain in the provided document why it make sense on a certain column(s))
  - { Before you create a database index perform a query that will use WHERE condition on a column without index and then perform the same query on the column with index (note: use EXPLAIN keyword to note the differences – provide a comparison of the execution plans)

- Create arbitrary database procedure (consider some complex case)
- Create arbitrary database trigger
- Create arbitrary database view (consider some complex case)
- Create database materialized view (consider some complicated SQL query with several joins, aggregate function, GROUP BY with HAVING and complex WHERE condition). Explain why this materialized view is beneficial/needed.
- Create two roles `teacher` and `student` in your database. Assign for `teacher` privileges to SELECT, INSERT, UPDATE, and DELETE everything in arbitrary table. Furthermore, set for `teacher` the possibility to view only certain fields (e.g., without salary from „person" or your „user" object). For `student` assign a possibility to select only certain tables.
- **Optional task**: encrypt your database.

Create a document that will note all these requirements together with the queries (or commands) performing them (please add also the attachments including the screen snippets of the query results).

This project assignment is performed by each student solely.

Please upload the document with attachments in a single **.zip** file named `UCO_YOUR-NAME_project-assignment-2.zip`. The **.zip** will be uploaded to the predefined folder on Google Drive.

This project assignment is for 6 points.

- **Target date: 16th of November (inclusive)**
- 3 for satisfying all the requirements (if any is not met, the number of points is set to 0)
- 0-3 based on the quality of a solution