BAN401
"Applied Programming and Data Analysis for Business"

FINAL GROUP-BASED PROJECT

SUBMISSION DEADLINE:

DATE: **14 November 2018**
TIME: **14.00**

I. REPORT PREPARATION:

You should submit TWO separate files:

File 1: One Report file (only PDF format) that includes the following:

1.1 Problems 1-3 (Python - part). For each problem:

a. Python code
   - *Copy your code and past it into you report*
   - *Keep your code formatting when pasting it into your report (for the best readability)*

b. Explanation of your solution
   - *Describe the idea of your designed solution*
   - *How your code solves the problem*

c. Screenshot of the results from the PyCharm

d. Required:
   - *Comment in your code explaining what you are doing. For example:*

```
s = [5, 35, 26, 44, 4, 1] # list s is created
```

   - *Code should be implemented in Python 3 (i.e., not in Python 2)*

1.2 Problem 4-6 (R - part). For each problem:

a. R code
   - *Copy your code and past it into you report*
   - *Keep your code formatting when pasting it into your report (for the best readability)*

b. Explanation of your solution
   - *Describe the idea of your designed solution*
   - *How your code solves the problem*

c. Screenshots of the results from the RStudio

d. Required:
   - *Comment in your code explaining what you are doing. For example:*

```
y <- c(1:10) # create a data vector with elements 1-10
```

1.3. Problem 7 (SQL – part)

- *Follow Problem 7 to report your results.*
- *REMEMBER: you **must** report the overall ER-model (relational schema) of your database.*

GENERAL REQUIREMENTS:

- Report should be typed (not handwritten)
- Report should be written in English
- It should be possible to copy text from your report. This does not apply to figures (if any) and screenshots.

File 2: One compressed file (ZIP or RAR format) including the following:

2.1 Problems 1-3 (Python):

a. 3 code files (**.py** format) - for problems 1-3, respectively
   - *Names of the .r files should include problems' numbers: problem_1.py; problem_2.py; problem_3.py*

2.2 Problems 4-6 (R):

a. 3 code files (**.r** format) - for problems 4-6, respectively
   - *Names of the .r files should include problems' numbers: problem_4.r; problem_5.r; problem_6.r*

b. All CSV files generated by your R-script in Problem 6.

2.3 Problem 7 (SQL):

a. Database file (**.db** format)
b. The **.txt** file with the SQL queries' code
c. The overall ER-model (relational schema) of your database (in the PDF format)

II. SUBMISSION PROCEDURE:

You must submit your final group-based project via WISEFlow.
Please, remember, that you must submit two files prepared according to "I. REPORT PREPARATION" procedure specified above:

File 1: One Report file (only PDF format)
File 2: One compressed file (ZIP or RAR format)

ETHICS:

- You are forbidden to discuss solutions with anyone outside your group.
- You are NOT allowed to distribute/post any parts of the given group-based project (in any format) in the internet.

YOUR SUBMISSION WILL BE CHECKED FOR PLAGIARISM!

PROBLEM 1

Problem 1.1:

You learned how to create functions using **def** keyword in Python. Write a Python code to solve the following problem.

1. Create a Python function `list_product(my_list)` that takes an argument called `my_list`:
   - `my_list` is a list of integers
   - `my_list` can be of any length

2. Function `list_product(my_list)` should return a new list. Each element `i` of the given returned list should be equal to the product of all elements of the list `my_list` except the element `my_list[i]`.

3. You code should display the initial list `my_list` and the resulting list returned by the function `list_product(my_list)`

Apply your coding solution to the following list: `[ 25 , 5 , 4 , 15 , 3 ]`

Requirements:
- You are not allowed to use a division operator (i.e. "/") in your code
- You are not allowed to use/import any modules
- Code must be implemented in Python 3 (i.e., not in Python 2)
- Provide the screenshot of your results from the PyCharm

Problem 1.2:

Consider the following Python code:

```
def main():
    r=len(A)-1
    p=0
    i=2
    Result=SELECT(A, p, r, i)
    print(Result, A)
```

1) Complete the given Python code based on the following pseudo-codes for the functions `SELECT, RANDOMIZED` and `PARTITION`:

```
SELECT(A, p, r, i)
1  if p == r
2      return A[p]
3  q = RANDOMIZED(A, p, r)
4  k = q − p + 1
5  if i == k
6      return A[q]
7  elseif i < k
8      return SELECT(A, p, q − 1, i)
9  else return SELECT(A, q + 1, r, i − k)
```

```
RANDOMIZED(A, p, r)
1  i = RANDOM(p, r)
2  exchange A[r] with A[i]
3  return PARTITION(A, p, r)
```

```
PARTITION(A, p, r)
1  x = A[r]
2  i = p − 1
3  for j = p to r − 1
4      if A[j] ≤ x
5          i = i + 1
6          exchange A[i] with A[j]
7  exchange A[i + 1] with A[r]
8  return i + 1
```

Notations:

- *A* is a list of numbers;

- Function RANDOM (in the line 1 of the function RANDOMIZED):
  - it is a function `random.randrange` (from the Python module called `random`) with arguments `start` equal to *p*, `stop` equal to *r*, and `step` equal to 1.
  - Specifically, see `random.randrange(start, stop[, step])` with an example of basic usage at the following link: https://docs.python.org/3.1/library/random.html

- Pseudo-codes presented above are adapted for Python 3 and use 0-based indexing. Indexes in your Python code should not be modified, i.e. they stay the same as indexes in the pseudo-codes. For example, *A*[*p*] (see line 2 in `SELECT`) stays to be *A*[*p*] and does not require *p* to be modified/shifted in your Python code.

- Regarding exchange-procedures (see line 2 in `RANDOMIZED` and lines 6-7 in `PARTITION`):
  Consider we have two objects X and Y. *Exchange X with Y* means that X should become equal to Y, and Y should become equal to X: X=Y and Y=X.

- In the function PARTITION:
  - *step* in the `for`-cycle is equal to 1 (i.e., incrementing by 1)
  - In line 3: since the pseudo-code is adapted for Python 3, (*p*, *r*-1) is the range include the first element *p* and exclude the last element *r*-1.

Requirements:
- `math` and `random` are the only modules that can be imported in your Python code.
- Code must be implemented in Python 3 (i.e., not in Python 2)

2) Apply your code to A = `[34.5, 5, -33, 13, 77, 5, 44 ,8, 0]` and provide the screenshot of your results from the PyCharm.

PROBLEM 2

Consider the social network G that consists of seven nodes connected to each other in the following way (Figure 1):

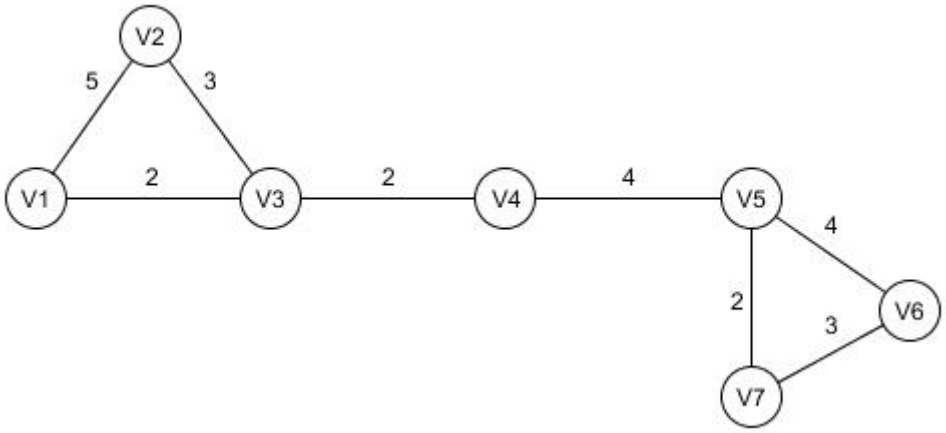

Figure 1. Social network G

Notations:

- V(G) is a set of all nodes in G: V(G) = $\{v_1, v_2, \dots, v_7\}$
- W(G) is a set of weights assigned to the direct links in G:
  W(G) = $\{w_{(v_1,v_2)} = 5, w_{(v_1,v_3)} = 2, \dots, w_{(v_6,v_7)} = 3\}$
- D(v) is a set of the shortest distances from node v to all other nodes in G:
  $D(v) = \{u_1 : d_1, \dots, u_{|V|-1} : d_{|V|-1}\} = \{(u : d)_1, \dots, (u : d)_{|V|-1}\} = \{(u : d)_{i=1,\dots,|V|-1}\}$,
  where:
  - $|V|$ is a size of G (i.e., the number of nodes in the network G);
  - $(u : d)_i$ is a pair of values that have the following interpretation:
    The shortest distance from node v to some other node u_i is equal to d_i
- Shortest distance:
  - The shortest distance from the node A to the node B is the smallest sum of links' weights on the path from A to B.
  - For example (See Figure 1): the shortest distance from node v_2 to v_7 is equal to 11 if we follow the path $v_2 \to v_3 \to v_4 \to v_5 \to v_7$. Note that no other path from v_2 to v_7 can give us the distance that is less than or equal to 11. For example, $v_2 \to v_1 \to v_3 \to v_4 \to v_5 \to v_7$ is a path with distance equal to 15.
- Direct link is the link that connects two nodes directly. For example: v_4 and v_5 are connected via direct link, v_4 and v_3 are connected via direct link, and so on.

For each node, we can measure its importance value $\mathbf{I}_v$ based on the following algorithm (pseudo-code):

Algorithm "Importance":

1 **FOR** each node from G do:
2 | set its I-value to be equal to zero;
3 **END**
4 **FOR** each node v from G do:
5 | DS = D(v) sorted in descending order by d-values;
6 | sum = 0; index = |V|-1; prevDistance = -1; prev_I = -1;
7 | **WHILE** index > 0 do:
8 | | **IF** $d_{DS[index-1]}$ == prevDistance **THEN**:
9 | | | curr_I = prev_I
10 | | **ELSE:**
11 | | | curr_I = $\frac{f(d_{DS[index-1]})}{1+index}$ - sum
12 | | **END**
13 | | $I_{u_{DS[index-1]}} = I_{u_{DS[index-1]}}$ + curr_I
14 | | sum += $\frac{f(d_{DS[index-1]})}{index*(1+index)}$
15 | | prevDistance = $d_{DS[index-1]}$
16 | | prev_I = curr_I
17 | | index = index - 1
18 | **END**
19 | $I_v = I_v + (f(0) - sum)$
20 **END**
21 print: I_v of each node
22 print: $\sum I_v$, which is a sum of all I_v in G

NOTATIONS:

- In line 5:
  - D(v) should be considered as unsorted.
  - all pairs of values (u : d) in DS should be sorted in descending order by d-values;

- In lines 8, 11, 13-15:
  - DS[index -1] is a pair (u : d) available in the data object DS (see line 5) at the index [index − 1];

- In lines 8, 11, 14, and 15:
  - $d_{DS[index-1]}$ is a value of d from a pair (u : d), where (u : d) is the pair from DS (see line 5) at the index [index − 1];

- In line 13:
  - $I_{u_{DS[index-1]}}$ is a value of I for the node u from a pair (u : d), where (u : d) is the pair from DS (see line 5) at the index [index − 1];

- function f: $f(value)=e^{-value}$, where e is an Euler's number

- Pseudo-code *Algorithm "Importance"* is adapted for Python 3 and uses 0-based indexing.
  Indexes in your Python code should not be modified, i.e. they stay the same as indexes in the pseudo-code.
  For example, DS[index − 1] (see line 8) stays to be DS[index − 1] and does not require "index − 1" to be modified/shifted in your Python code.

Write a Python code of the *Algorithm "Importance"* and apply it to the social network presented in Figure 1.

REQUIREMENTS:
- You must use `for` and `while` control flow statements in your code.
- You are not allowed to use/import any modules except modules `math` and `operator` (if needed)

NOTES:
- You do not have to write a code to calculate the shortest distances between nodes. You can provide the values of the shortest distances directly in you code (i.e., pre-calculated manually);
- According to the lines 21-22 of the *Algorithm "Importance"* your code should print the following results:
  - importance value I_v of each node
  - sum of the importance values of all nodes in the network: $\sum I_v$

- Your results should be printed in a similar format:

```
I(node 1) = . . .
I(node 2) = . . .
      . . .
I(node 7) = . . .
SUM:  . . .
```

  (Your computational results should be displayed in the fields `. . .`)

- Code must be implemented in Python 3 (i.e., not in Python 2)
- Provide the screenshot of your results from the PyCharm applied to the network presented in Figure 1.

PROBLEM 3

You learned control flow statements FOR and WHILE.

3.1 Develop your own programming problem that shows the advantage of using WHILE over FOR.

(a) The problem should be well described and structured

- All notations should be explained (if any)
- The problem should be original (it will be checked for plagiarism!)

(b) Write a Python code that solves your problem

- The code must be implemented in Python 3 (i.e., not in Python 2)
- Provide the screenshot of your results from the PyCharm.

3.2 Choose one industry: manufacturing, public health, retail, or finance. Discuss a real-life situation/case (for the chosen industry) where the programming problem that you developed for task 3.1 can be applied. Perform analysis:

- Describe how the programming problem can be applied in the discussed real-life situation/case.
- Explain why WHILE should be employed to solve the programming problem in the discussed real-life situation/case.
- Argue why FOR cannot be employed to solve the programming problem in the discussed real-life situation/case.

Maximum (for the sub-problem 3.2): 1000 words excluding code (if any) and references (if any). Text above this limit will be ignored.

PROBLEM 4

Three natural numbers, $p < j < y$, that satisfy $y^2 = p^2 + j^2$, are forming "SuperBox". "SuperBox" can be formed by different numbers $p, j,$ and y. However, there exist only one "SuperBox" that satisfies the following condition (1):

$$p + j + y = 1000 \quad (1)$$

Write an R code:

(a) to detect $p, j,$ and y, that satisfy condition (1);
(b) to calculate $p * j * y$ based on the results from (a);

Your code should display the results from (a) and (b) in RStudio console.

Notes:
- Natural numbers are positive integers (whole numbers) 1, 2, 3, etc.
- Solving this problem, you are not allowed to have any packages loaded into your R script (no library()-functions)
- Provide the screenshot of your results (from the RStudio console)

PROBLEM 5

Consider the following pseudo-code "*Order*". For any vector of numbers A, the given pseudo-code gives an updated vector with all numbers (from the original A) in non-decreasing order.

Algorithm "Order":

```
1  for t ← 2 to length[A]
2       do key ← A[t]
3          n ← t - 1
4          while n > 0 and A[n] > key
5              do A[n + 1] ← A[n]
6                 n ← n - 1
7          A[n + 1] ← key
```

- Modify algorithm *"Order"* to write an R function that will return an updated vector with all numbers (from the original A) in the non-increasing order.
- Apply your code to the following vector of numbers: 5.4, -1, 4, 6.3, 1, 2.1, 1
- Your code should display results in the RStudio console.

Notes:

- Pseudo-code presented above is adapted for R and uses 1-based indexing.
  Indexes in your R code should not be modified, i.e. they stay the same as indexes in the pseudo-code.
  For example, $A[n+1]$ (see line 5) stays to be $A[n+1]$ and does not require "$n+1$" to be modified/shifted in your R code.

- Solving this problem, you are not allowed to have any packages loaded into your R script (no library()-functions)
- Provide the screenshot of your results (from the RStudio console)

PROBLEM 6

I. NETWORK

Consider the network G that consists of seven companies connected to each other in the following way (Figure 2):

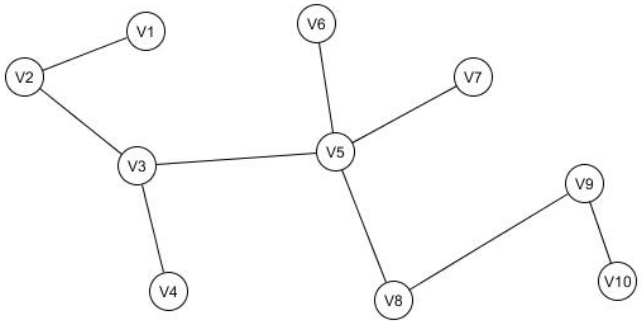

Figure 2. Network G

Notations:

1. Based on Figure 2, each company **v** has a number of its direct links (in the network G) denoted by $\deg_G(v)$.
   For example: $\deg_G(v_3) = 3$, because company v_3 has three direct links (specifically, to companies v_2, v_4 and v_5)

2. List of companies (i.e., neighbors) that are reachable from the company **v** in at most one hop (i.e., via direct link) within network G is denoted by $N_G(\mathbf{v})$. Each company **u** from $N_G(\mathbf{v})$ has a number of its direct links (in the network G) denoted by $\deg_G(u)$.

   Note: direct link is the link that directly connects two companies. For example: v_5 and v_6 are connected via direct link; v_4 and v_3 are connected via direct link, and so on.

3. Network G (from Figure 2) can be presented as an edge list:

   1 – 2
   2 – 3
   3 – 4
   3 – 5
   5 – 6
   5 – 7
   5 – 8
   8 – 9
   9 – 10

   The given edge list is presented in the .csv format. Please, find the attached "Connections.csv"-file in the WISEFlow.

II. ALGORITHM

For each company, we can measure its value of power **Power(v)** based on the following algorithm (pseudo-code):

Algorithm "Power":

FOR each company v from G do:
    Power(v) = $\frac{1}{1+\deg_G(v)}$
    FOR each company u from $N_G(v)$ do:
        Power(v) += $\frac{1}{1+\deg_G(u)}$
    END
END
print: for each company v its Power(v)
print: $\sum$ Power(v), which is a sum of all Power(v)-s in G

III. TASKS

Write an R code (one .r script) to perform the following tasks:

1. You code should read the edge list from the CSV file "Connections.csv" into R, and create a graph (i.e., network) based on the "Connections.csv". In your code, the created graph should be stored in the variable called `g`. To create a graph, you should use R package called `igraph`: http://igraph.org/r/

2. Your code should find communities of companies in the created graph using a function called `cluster_louvain()`:
   http://igraph.org/r/doc/cluster_louvain.html

   Note: Function `cluster_louvain()` is integrated into the `igraph`-package. This means that you should not create a mechanism to find a community structure. To find a community structure you just need to employ the given developed `cluster_louvain()` function.

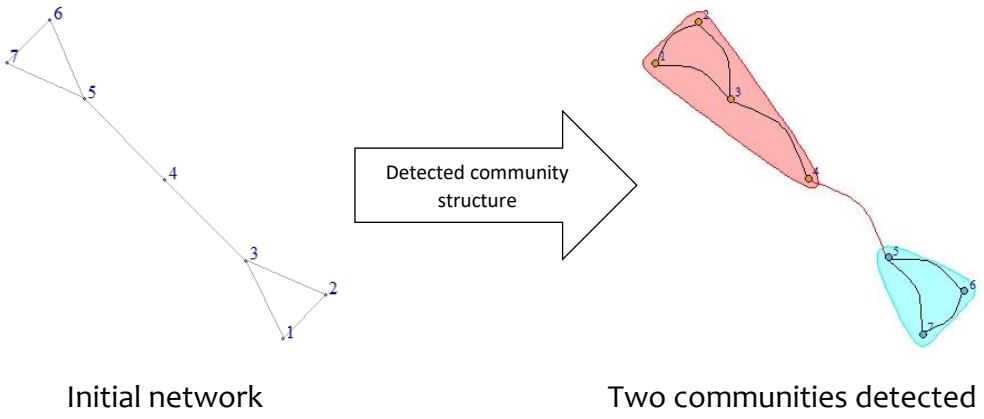

Figure 3. Illustrative example of a community structure

Note: it is **not required** to visualize the community structure. The given example (Figure 3) is for the illustrative purposes only. The illustrative example presented in Figure 3 is not related to the network presented in Figure 2.

3. Based on the results from the task 2 (in "III. TASKS"), your code should calculate the power of all companies (i.e., **Power(v)**) in each community based on the *Algorithm "Power"* introduced in "II. ALGORITHM"

- When your code calculates the power of companies, each community should be considered separately (as an independent network). It means that for each community, only internal companies and links (within a community) should be taken into account (i.e., internal community structure should be analyzed).

For example, in Figure 3 we have two detected communities:

- Community_1 consists of companies 1, 2, 3, and 4 and links 1-2, 1-3, 2-3, and 3-4. Thus, the power of each company (i.e., **Power(v)**) should be calculated based only on the given structure of the Community_1.
- Community_2 consists of companies 5, 6, and 7 and links 5-6, 5-7, and 6-7. Thus, the power of each company (i.e., **Power(v)**) should be calculated based only on the given structure of the Community_2.

- Based on the lines 7-8 of the *Algorithm "Power"*, your code should display the following results (in the RStudio console):
  - **Power(v)** of each company in the community
  - Sum of the Power-values of all companies in the community: $\sum \mathbf{Power}(\mathbf{v})$
  - Results should be displayed in a similar format (in the RStudio-console):

```
Power values in community 1 :

                Company_ID                          Power
[1,]            . . .                               . . .
[2,]            . . .                               . . .
. . .
[…,]            . . .                               . . .

SUM(Powers) =  . . .

                        . . .

Power values in community . . . :

                Company_ID                          Power
[1,]            . . .                               . . .
[2,]            . . .                               . . .
. . .
[…,]            . . .                               . . .

SUM(Powers) =  . . .
```

Note: Your computational results should be displayed in the fields: `. . .`

- Provide the screenshot of your results (from the RStudio console) applied to the network presented in Figure 2.

4. You code should export results for each detected community to a separate CSV file:

- Each CSV file should contain the following information:
  i. `Company_ID:` company that is a member of the current community
  ii. `Power:` corresponding **Power(v)**-value

- Each CSV file should include the column headers (i.e., `Company_ID` and `Power`) generated by your R code.

- Your R code should export all CSV files to the same folder where the R script is located.

Illustrative Example (for the Task 4):

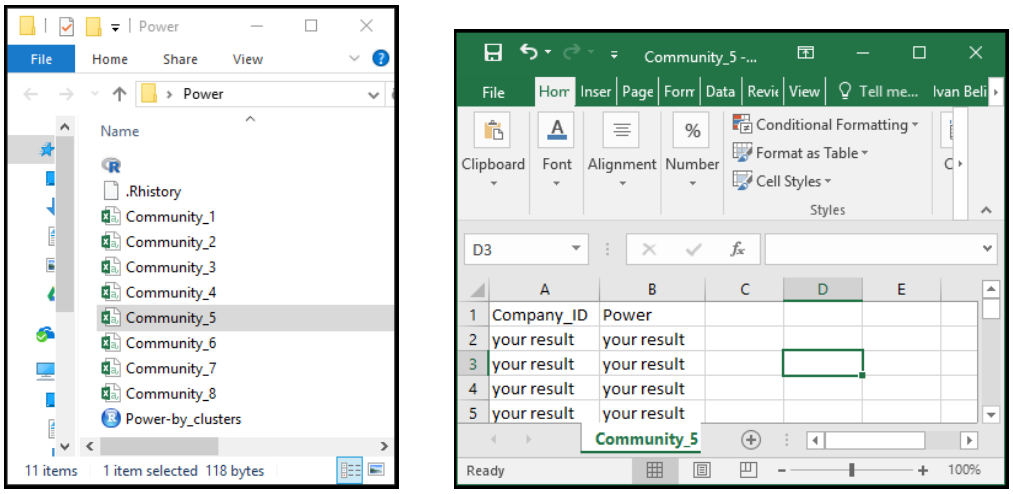

Figure 4. Illustrative example: export to the CSV files

Note: The given example (Figure 4) is for the illustrative purposes only. Results presented in Figure 4 are not related to Figures 2-3

REQUIREMENTS:

1. `igraph` is the only package that should be called (loaded) in your R script: `library(igraph).`
   No other packages are allowed to be used.
2. You must use functions `induced_subgraph()` and `degree()` from the `igraph`-package in your R code.

PROBLEM 7

Design and create your own database (based on the SQLite3).

Database should be logically consistent and should reflect the realistic entities, attributes and relations. It can reflect any industry such as telecom, retail, health care or any other industry (based on your preferences and interests).

PLAGIARISM CONTROL:

1. *It is prohibited to use any RDBMS database templates from the Internet.*
2. *Internet sources as well as CANVAS and ITLEARNING submissions (from the current and previous years) will be controlled!!!*
3. *Make sure you create your own original database!*

1. DATABASE IDEA:

Describe the idea: what does your database reflect in terms of the selected industry.

2. ER-MODEL:

Create the overall ER-model (relational schema) of your database.

Entities requirements:
Your database should contain **11-15 entities** (i.e., tables) with corresponding attributes.

Relationships requirements:
Your database should have:

- at least two one-to-one (1:1) relationships
- at least three one-to-many (1:N) / many-to-one (N:1) relationships that are not a part of N:N relationships
- at least two many-to-many (N:N) relationships

Please, see slide 31 (SQL - Lecture 1 "Databases" in CANVAS) to recall what is the overall database ER-model (relational schema). Your ER-model should contain all details regarding the entities, attributes, datatypes and relationships.

NOTE:
You can submit the scanned hand-drawn version of the overall database ER-model (relational schema). In this case, make sure that your hand-drawn ER-model is readable (i.e., in high-resolution).

3. DATABASE CREATION:

- The database schema and data records should be created using **only SQL queries**.

- Each table of the resulted database should be populated by at least five records using INSERT INTO statement.

- You must use **only SQLite DB Browser** (employed in this course) to create your database.

  You have to use "Execute SQL"-tab *from the SQLite DB Browser for all manipulations to create the database and to populate it with data records. No other tools are allowed.*

3.1 SQL code:

- Provide all SQL queries in the PDF-report.
  All SQL queries should be provided in the right order.

- Attach the **.txt** file with the overall SQL code to the submission.
  If you have difficulties to create and save .txt file you can use online tools such as:
  https://www.editpad.org/

- Attach your **.db** database file to the submission.

3.2. For each relationship (between tables)

a. Briefly describe the purpose of the relationship (its role in the database):

- Why the given relationship is created; which entities does it connect

- Specify the type of the relationship: 1:1, 1:N (or N:1), or N:N

b. For each table participating in the given relationship:

- Briefly describe the purpose of the table; what does it reflect

- For column(s), that is/are assigned to be primary and/or foreign key(s), explain briefly why the given specific column(s) is/are assigned to be primary key(s) and/or foreign key(s).