Assignment 1

Submission instructions

The assignment should be submitted in pairs. You should submit one zip file. The archived the zip file should contain 2 files:

- A txt file named sql.txt that contains the MSSQL part (use the attached file and write the commands under the appropriate comments).
- A Python file named hw1.py contains all the functions and comments.
- No need to submit the main function but using it for the internal tests is recommended.
- Do not leave unnecessary commands in the submission files – all files are automatically checked in full.
- The grade will be reduced for a submission that is not in the correct format.
- Before submitting, check the forum for updates and make sure your submission follows all clarifications.

The archive file name should be of form id1_id2.zip.
Deadline: 24.11.2025

1. MSSQL

a. Create a table **Films** with the following columns:

Column	Type	Description
FID	BIGINT	**Primary key** (film ID)
TITLE	VARCHAR(200)	Film title
PROD_YEAR	BIGINT	Production year
RATING	REAL	Film rating in range [0,10] (add as a constraint using CHECK)

b. Create a table **AnnualSummary** with the following columns:

Column	Type	Description
PROD_YEAR	BIGINT	**Primary key**
FID	BIGINT	Id of the top-rated film form the year. **Foreign key to Films.FID**
RATED_ABOVE_8	BIGINT	Number of films from the year which are rated above 8. (**>8**, not >=8)

c. Create a trigger **AutoAddFID** that runs on each insertion into the Films table.
- The trigger assigns the next FID value (starting from 0, increasing by 1 for each new row).
- The trigger keeps the rest of the features of the inserted row.
- Do not use a SQL sequence or identity column.

d. Create a stored function **GetBestFilm**:
   - The function gets 1 parameter: @P_YEAR (BIGINT).
   - The function returns: (BIGINT) – ID of the top-rated film produced in P_YEAR. (You can assume there's only one film with the top rating from each year.)

e. Create a stored function **GetNumAbove8**.
   - The function gets 1 parameter: @P_YEAR (BIGINT).
   - The function returns: (BIGINT) - **Number of films** produced in P_YEAR which are rated above 8. (**>8**, not >=8)

f. Create a stored procedure **AddSummaryItems**:
   - For each 'PROD_YEAR' in Films table:
     - Find the id of the top-rated film form PROD_YEAR.
     - Find the number of films from PROD_YEAR which are rated above 8.
     - Insert a new row into **AnnualSummary** table according to the description from section b.
   - Implement this procedure by calling the functions **GetBestFilm** & **GetNumAbove8**.

2. Pyodbc - use the attached template file hw1.py

a. Create a class **DatabaseManager** and write the class constructor:

```
class DatabaseManager
def __init__(self, driver: str, server: str, username: str, password: str, database: str
)
```

- The constructor receives five values:
  1. Local Driver
  2. Server IP
  3. DB username
  4. DB password
  5. Name of the DB

- The given parameters should be used for the creation of the connection(s) as seen in practice 2.

b. Implement the function **file_to_database**:

```
def file_to_database(self, path: str) -> None
```

- The function receives the path of the file (as str).
- The function reads the file content and inserts it into the Films table.
- The file is in CSV format, the first value is the title of the film, the second value is the production year and the third is the rating. [see attached file: films.csv]

c. Implement the function **add_summary_items**:

```
def insert_summary_items (self) -> None
```

- The function does not receive any parameter.
- The function Executes the SQL stored procedure '**AddSummaryItems**', which inserts the items into **AnnualSummary** table**.**
- The function does not return any values.

d. Implement the function **get_best_films**:

def get_best_films (self) -> dict <int: str>

- The function does not receive any parameter.
- The function returns a dictionary that maps each year to its top-rated film in the format: {year: film_title}

e. Implement the function **get_n_best_years**:

def get_n_best_years (self, n) -> list<int>

- The function receives an integer n.
- The function returns a list of the top n years, sorted by how many films produced in each year received a rating above 8.
  For example, if the output is [2001, 2007, 2003], it means:
  - 2001 had the most films rated above 8.
  - 2007 had the second most.
  - 2003 had the third most.
- If two years are equal, prioritize the earlier one among them.

Good Luck 😊

Dorri Caspi