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.

The archive file name should be of form id1_id2.zip.
Deadline: 28.11.2024

Assignment Instructions:

a. Create a table MediaItems with the following columns:
   - MID (BIGINT) –primary key
   - TITLE (VARCHAR (200))
   - PROD_YEAR (BIGINT)
   - TITLE_LENGTH (BIGINT)

b. Create a table Similarity with the following columns:
   - MID1 (BIGINT) – primary key, foreign key to MediaItems.MID
   - MID2 (BIGINT) – primary key, foreign key to MediaItems.MID
   - SIMILARITY (REAL)

c. Create a trigger AutoIncrement:
   - On each insertion of the row into the table MediaItems, the trigger generates and updates 2 values in the row: MID index and TITLE_LENGTH – length of the title.
   - The first MID index should be 0.
   - Do not use any sequence to generate MID

d. Create the MSSQL function MaximalDistance:
   - The function does not receive any values.
   - The function returns the maximal distance between all the items – a number
   - Two items distance is the absolute difference between the production years of the given items. E.g: $d(a, b) = |PROD\_YEAR_a - PROD\_YEAR_b|$

   Maximal distance is the maximal result of the distance calculation between all the item pairs.

   e.g. $F = max(\ d\ (a, b) \mid \forall\ a, b \in MediaItems)$

e. Create an MSSQL function SimCalculation which calculates the similarity between 2 media items:
   - The function receives 2 MIDs and the number maximal_distance

- The function returns the similarity - a number in the range [0,1] (real)
- The similarity is defined as: $1 - \frac{two\_items\_distance}{maximal\_distance}$

f. Create a stored procedure AddSummaryItems:

- For each unique `PROD_YEAR` in the `MediaItems` table, check how many media items have the same production year.
- If more than one item exists for a particular year, insert a new media item into the `MediaItems` table with the following properties:
  The `TITLE` should be in the format: `<amount> items in <year>`, where `<amount>` is the number of items with that `PROD_YEAR` and `<year>` is the PROD_YEAR.
  The `PROD_YEAR` of the new media item should be the same as the original year.

1. Python – 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)
```

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

- The given parameters should be used for the creation of the connection(s) as seen in practice 3.

b. Write a Python function file_to_data_base:

```
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 to the MediaItems table.
- The file is in CSV format, the first value is the title of the item, and the second value is the production year [see attached file: films.csv]

c. Write a Python function calculate_similarity:

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

- The function does not receive any values.
- The function calculates the similarity between every pair of items in the MediaItems table using the SimCalculation and MaximalDistance MSSQL functions and inserts or updates the row in the Similarity table. The inserted row should contain MID1, MID2, SIMILARITY. Each pair of MID's should appear once.

d. Write a Python function print_similar_items:

def print_similar_items(self, mid: int) -> None

- The function receives a number mid.
- The function retrieves in ascending order from the database all the items that the similarity between them and the given item is at least 0.25.
- The function prints all the titles and the similarity value of the similar items (using print).

e. Write a Python function add_summary_items:

def add_summary_items (self) -> None

- The function does not receive any values.
- The function Executes the SQL stored procedure `AddSummaryItems`, which adds summary media items based on the number of items with the same production year.
- The function does not return any values.

Good Luck,
Yarin Benyamin.