Data Science HW#1

산업공학과 201911532 송용재

(사용환경: Visual Studio Code, 사용언어:Python 3.11)

소스코드 및 설명

1.

db_conn.py (default)

```python
import pymysql

from pymysql.constants.CLIENT import MULTI_STATEMENTS

def open_db(dbname='DS2023'):

    #conn 에서 db_conn 으로 임의적 변경 (가시성 목적)
    db_conn =pymysql.connect(host='localhost',
                             user='root',
                             passwd='Steve72041!',
                             db=dbname,
                             client_flag=MULTI_STATEMENTS,
                             charset='utf8mb4')

    cursor = db_conn.cursor(pymysql.cursors.DictCursor)

    return db_conn, cursor

def close_db(db_conn, cur):
    cur.close()
    db_conn.close()

if __name__ == '__main__':
    db_conn, cur = open_db()
    close_db(db_conn, cur)
```

먼저, pymysql을 활용하기 위해, db의 open과 close를 간소화 하고자 지정한 python 모듈입니다.
def로 지정해놓고, **from db_conn import *** 를 통해 자유롭게 사용이 가능합니다.
배포한 파일에서, 가시성 목적으로 **conn을 모두 db_conn으로 임의변경** 하였습니다.

if __name__ == '__main__': 에 대해서 간략하게 언급하고 싶습니다.
모듈을 그 자체로 실행 가능한 스크립트로 사용할 때 추가되는 함수입니다.

이를 통해,

```python
db_conn, cur = open_db()
close_db(db_conn, cur)
```

위 코드를 파이썬 파일에서 sql언어 사용을 목적으로 계속 활용하겠습니다.

2.

ds-hw#1 (201911532 송용재).py

```
from db_conn import *
import pandas as pd
import pymysql
import re

#table 을 create 하기
db_conn, cur = open_db()

create_table_query = """
    CREATE TABLE hw0908 (
    id INT AUTO_INCREMENT PRIMARY KEY,
    Movie_Name VARCHAR(255),
    Year_of_Release VARCHAR(255),
    Watch_Time INT,
    Movie_Rating DECIMAL(10,1),
    Metascore_of_movie INT,
    Gross DECIMAL(10,2),
    Votes VARCHAR(255)
    )
    """
cur.execute(create_table_query)

#data 처리 작업, insert (description 은 용량 문제로 제외)
file_name='top_movies.csv'
df = pd.read_csv(file_name)

df.drop(columns=['Description'], inplace=True)

insert_query = """
    INSERT INTO hw0908 (Movie_Name, Year_of_Release, Watch_Time,
    Movie_Rating, Metascore_of_movie, Gross, Votes) VALUES (%s, %s, %s, %s, %s, %s,
%s)
    """

db_conn, cur = open_db('DS2023')

truncate_sql = """truncate table hw0908;"""
cur.execute(truncate_sql)
db_conn.commit()

for index, row in df.iterrows():
    movie_name = str(row['Movie Name'])
    year_of_release = str(row['Year of Release'])
    watch_time = int(row['Watch Time'])
    movie_rating = float(row['Movie Rating'])
    #NaN 은 None 으로 대치
    metascore_of_movie = int(row['Metascore of movie']) if not pd.isna(row['Metascore
of movie']) else None
```

```
    gross = float(row['Gross']) if not pd.isna(row['Gross']) else None
    votes = re.sub(r'[^0-9]', '', row['Votes'])

    cur.execute(insert_query, (movie_name, year_of_release, watch_time,
                               movie_rating, metascore_of_movie, gross, votes))

    #터미널에서 출력
    print((movie_name, year_of_release, watch_time,
           movie_rating, metascore_of_movie, gross, votes))

db_conn.commit()

#sql 에서 select 하기, workbench 에서 직접 확인
sql = "SELECT * FROM hw0908;"
rows = cur.fetchall()

for row in rows:
    print(row)

cur.execute(sql)

close_db(db_conn, cur)
```

정의된 과제를 수행하기 위해 작성한 소스코드 입니다.
사용된 csv파일은 배포하신 'top_movies.csv' 이며, Kaggle에서 download 하였습니다.
수행 목적은 mysql의 table로 입력하고, table을 검색하는 python 코드 작성입니다.

자세한 설명 이전에, 제 소스코드에 대한 유의사항으로는 csv내의 'Description'이라는 column은 sql에서 LONGTEXT로 지정하였음에도, 용량문제가 지속적으로 발생하여 임의로 제외하고 나머지 column들 만을 dataframe으로 읽어 들였습니다.
추가적으로, Gross column에서 '#222'라는 값이 유일하게 float나 int로 읽어들일 수 없는 문제가 발생하여, 편의상 csv파일 내에서 직접 '222'로 수정을 하고 진행하였습니다.

2.1. 코드 상세 설명

2.1.1. (TABLE : hw0908)

```
create_table_query = """
    CREATE TABLE hw0908 (
    id INT AUTO_INCREMENT PRIMARY KEY,
    Movie_Name VARCHAR(255),
    Year_of_Release VARCHAR(255),
    Watch_Time INT,
    Movie_Rating DECIMAL(10,1),
    Metascore_of_movie INT,
    Gross DECIMAL(10,2),
    Votes VARCHAR(255)
    )
    """
cur.execute(create_table_query)
```

pymysql기능을 활용하여, mysql언어를 python환경에서 지정하는 코드입니다.
Id를 Primary Key로 추가로 7개의 항목을 넣었습니다. (Description 제외)
VARCHAR(255)는 제한된 frame에 문자열, INT는 정수를 지정하며,
DECIMAL(i, j)는 소수점 표기가 가능합니다. i는 가능한 자릿수를, j는 표현 소수점 범위를 지정합니다.

2.1.2.

```
file_name='top_movies.csv'
df = pd.read_csv(file_name)

df.drop(columns=['Description'], inplace=True)

insert_query = """
    INSERT INTO hw0908 (Movie_Name, Year_of_Release, Watch_Time,
    Movie_Rating, Metascore_of_movie, Gross, Votes) VALUES (%s, %s, %s, %s, %s, %s,
%s)
    """

db_conn, cur = open_db('DS2023')

truncate_sql = """truncate table hw0908;"""
cur.execute(truncate_sql)
db_conn.commit()

for index, row in df.iterrows():
    movie_name = str(row['Movie Name'])
    year_of_release = str(row['Year of Release'])
    watch_time = int(row['Watch Time'])
    movie_rating = float(row['Movie Rating'])
    #NaN 은 None 으로 대치
    metascore_of_movie = int(row['Metascore of movie']) if not pd.isna(row['Metascore
of movie']) else None
    gross = float(row['Gross']) if not pd.isna(row['Gross']) else None
    votes = re.sub(r'[^0-9]', '', row['Votes'])

    cur.execute(insert_query, (movie_name, year_of_release, watch_time,
                               movie_rating, metascore_of_movie, gross, votes))

    #터미널에서 출력
    print((movie_name, year_of_release, watch_time,
           movie_rating, metascore_of_movie, gross, votes))

db_conn.commit()
```

먼저, pandas를 통해 csv파일을 dataframe으로 읽어 들였습니다. Table의 이름은 'hw0908' 입니다.

이후, sql내에서 instert를 실행하는 코드를 작성하였으며, %s 개수로 column의 바인딩을 조정했습니다. 앞서 언급한 바와 같이, 'Description'은 drop을 먼저 하고 진행하였으며, sql에서 올바르게 읽기 위해 약간의 데이터 처리 과정을 거쳤습니다. 'Metascore of movie'에서는 결측값들을 모두 None으로 변경 하였고, 'Votes'에서는 올바르게 읽기 위해 ,를 모두 replace 했습니다.

3.
Output

먼저, 2.1.2.에서 작성되어 있는,

```
print((movie_name, year_of_release, watch_time,
          movie_rating, metascore_of_movie, gross, votes))
```

에 따라 터미널에서 출력된 결과값들은 아래 사진과 같습니다.

올바르게 작동되는지 확인 목적의 코드입니다.

```
...   ('The Shawshank Redemption', '1994', 142, 9.3, 82, 28.34, '2777378')
      ('The Godfather', '1972', 175, 9.2, 100, 134.97, '1933588')
      ('The Dark Knight', '2008', 152, 9.0, 84, 534.86, '2754087')
      ("Schindler's List", '1993', 195, 9.0, 95, 96.9, '1397886')
      ('12 Angry Men', '1957', 96, 9.0, 97, 4.36, '824211')
      ('The Lord of the Rings: The Return of the King', '2003', 201, 9.0, 94, 377.85, '1904166')
      ('The Godfather Part II', '1974', 202, 9.0, 90, 57.3, '1314609')
      ('Spider-Man: Across the Spider-Verse', '2023', 140, 8.9, 86, 15.0, '198031')
      ('Pulp Fiction', '1994', 154, 8.9, 95, 107.93, '2131189')
      ('Inception', '2010', 148, 8.8, 74, 292.58, '2444816')
      ('Fight Club', '1999', 139, 8.8, 67, 37.03, '2212960')
      ('The Lord of the Rings: The Fellowship of the Ring', '2001', 178, 8.8, 92, 315.54, '1932439')
      ('Forrest Gump', '1994', 142, 8.8, 82, 330.25, '2160038')
      ('Il buono, il brutto, il cattivo', '1966', 161, 8.8, 90, 6.1, '784276')
      ('The Lord of the Rings: The Two Towers', '2002', 179, 8.8, 87, 342.55, '1718332')
      ('Jai Bhim', '2021', 164, 8.8, 0, 219.0, '208742')
      ('777 Charlie', '2022', 136, 8.8, 0, 0.0, '35870')
      ('Oppenheimer', '2023', 180, 8.7, 88, 29.0, '266774')
      ('Interstellar', '2014', 169, 8.7, 74, 188.02, '1956197')
      ('GoodFellas', '1990', 145, 8.7, 92, 46.84, '1205052')
      ("One Flew Over the Cuckoo's Nest", '1975', 133, 8.7, 84, 112.0, '1037205')
      ('The Matrix', '1999', 136, 8.7, 73, 171.48, '1976616')
      ('Star Wars: Episode V - The Empire Strikes Back', '1980', 124, 8.7, 82, 290.48, '1333333')
      ('Rocketry: The Nambi Effect', '2022', 157, 8.7, 0, 0.0, '54505')
      ('Soorarai Pottru', '2020', 153, 8.7, 0, 0.0, '120200')
      ...
      ('Un long dimanche de fiançailles', '2004', 133, 7.6, 76, 6.17, '75004')
      ('Shine', '1996', 105, 7.6, 87, 35.81, '55589')
      ('The Invisible Man', '1933', 71, 7.6, 87, 0.0, '37822')
      ('Celda 211', '2009', 113, 7.6, 0, 0.0, '69464')
      Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

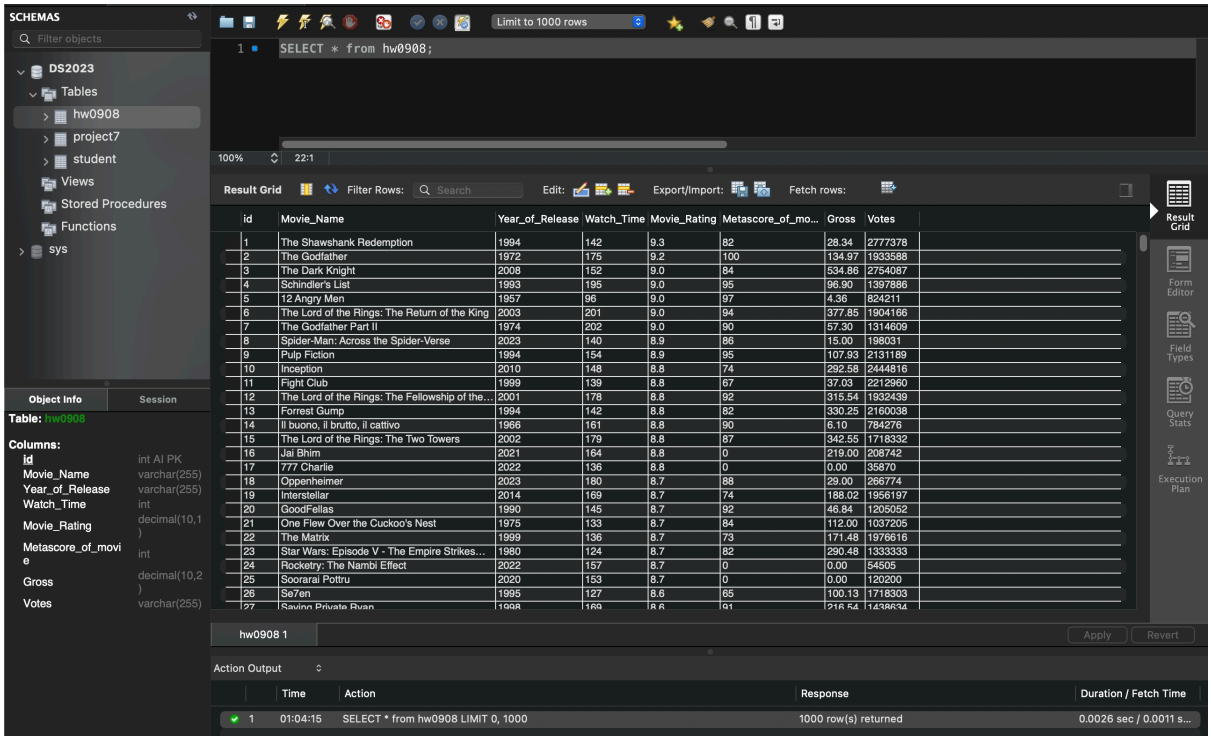
다음은 mysql에서 SELECT문을 작동시키기 위한 코드입니다.

```
sql = "SELECT * FROM hw0908;"
rows = cur.fetchall()

for row in rows:
    print(row)

cur.execute(sql)

close_db(db_conn, cur)
```

테이블의 모든 row를 읽어들이게 됩니다.

다음은 MySQL Workbench에서 직접 확인한 테이블의 모습입니다.

이제 지정된 테이블에서 다양한 작업 및 활용이 가능합니다.



4. Reference

ChatGPT