DATABASE SYSTEMS ASSIGNMENT4

Application

과제 개요1

```
else:
    column_names = [desc[0] for desc in cur.description]
    #
    #print_rows_to_file(column_names, rows)
    #make_csv(column_names, rows)
    #
    print_rows(column_names, rows)
    return True
```

```
if __name__ == "__main__":
    #
    #print_command_to_file()
    #
    start = time.time()
```

- **요구사항 : "영화 ott" 애플리케이션의 핵심 기능 개발**
  - HW4_ApplicationDevelopment 내의 아래 5가지 파이썬 파일의 프로그램 내용을 완성
    - connection.py, customer.py, movie.py, role.py, participant.py 내의 [TODO] 는 필수로 작성
  - 작성한 5가지 파이썬 파일을 주어진 작동 요구에 따라 실행 및 정상 동작 확인 후 제출
    - 작성한 파이썬 파일에 대한 동작 요구 및 실행결과 값은 answer 디렉토리(디렉토리 구조는 추후 설명)에 "해당_파이썬_파일의_파일명.txt"에 기술되어 있음.
      - 주어진 동작요구 마다 실행 결과 값은 answer 디렉토리에 "주어진_각_동작_요구.csv"에 저장되어 있음
    - 파일명.txt 파일을 통해 각 커맨드에 대한 출력화면 확인 가능
      - 각 csv 파일(파일명: 실행파일명_커맨드_옵션_값)의 내용은 해당 파일의 이름에 따라 "python 실행파일명 커맨드 옵션 값"을 실행한 결과 값임.
    - 해답 csv 파일과 본인이 실행하여 얻은 csv 파일 결과(src 디렉토리에 위치, 디렉토리 구조는 추후 설명)가 일치하는지 확인한 후 제출
  - 제공되는 각 파일 작성시 추가 요구사항
    - customer.py의 경우 위 사진의 주석 코드를 활성화하면, answer 디렉토리 내의 csv 파일과 txt 파일과 같이 출력파일을 얻을 수 있음
    - 다른 파이썬 파일()의 경우 customer.py를 참고하여 추가할 것
    - 제출 시에는 4가지 프로그램 파일(customer.py, movie.py, role.py, participant.py)은 위 사진와 같이 주석 처리할 것
    - 해당 코드의 동작은 helpers/utils.py 에 구현되어 있으므로 확인해볼 수 있음

과제 개요2

- **결과물 – 작성된 SQL문들과 실행 결과물**
  - 제출파일
    - HW4_ApplicationDevelopment 폴더에 있는 모든 파일 및 하위 디렉토리들을 이름_학번_DB04.zip 파일로 압축하여 제출. (예: 홍길동_2024123456_DB04.zip)
    - 실행 후 생성된 본인의 txt 및 csv 파일은 제외하고 제출할 것
- **점수 반영**
  - **요구 사항 외에 추가 구현한 내용에 따라 가산점 부여**
  - 기본 제출기한 내 : 100%
  - 추가 제출기한 내: 50%
- **카피 적발시**
  - 점수반영 0%
  - 과제 점수 25%에서 2%씩 감점

제공되는 파일의 디렉토리 구조

```
HW4_ApplicationDevelopment
├── src
│   ├── answer
│   ├── helpers
│   │   ├── connection.py
│   │   └── utils.py
│   ├── customer.py
│   ├── movie.py
│   ├── role.py
└   └── participant.py
```

설치가 필요한 라이브러리

- Psycopg2
    - #pip install --upgrade pip
    - #pip install psycopg2

기술적 요구사항

- CLI
  - 에러가 발생했을 때, 데이터베이스에 변경사항이 없어야 함
  - 에러가 발생했을 때, 프로그램은 종료되어야 함

- 출력은 display_info 함수를 통해서만 할 것

- 기본적으로 데이터모델링 및 처리를 돕는 외부 라이브러리는 사용을 금함
  - sqlalchemy, pandas 등

시나리오 개괄

- 1.사용자 정보 생성/갱신/검색/삭제
  - 1-1. 새로운 사용자 생성 (선호 장르 입력)
  - 1-2. 사용자 정보 갱신 (이메일, 비밀번호, 전화번호)
  - 1-3. 사용자 검색
    - id , 이름, 선호장르
  - 1-4. 사용자 삭제
    - 시청기록, 코멘트 cascade

- 2.영화 검색
  - id , 이름, 장르, 상영시작년도, 상영종료년도, 성인등급 여부, 평점

- 3.참여자 검색
  - id, 이름, 직업

- 4. 참여 역할(role) 기반 참여자 검색
  - 4-1.전체 영화에 대해서 참여 역할 기반 참여자 검색
  - 4-2.한 영화에 대해서 참여 역할 기반 참여자 검색

customer.py

```
python customer.py info [-i/-n/-g/-a] [field value]
python customer.py insert –g genre1 genre2 genre3 ([customer value])+
python customer.py update -i C_ID [-m/-p/-ph] [new field value]
python customer.py delete –i C_ID
```

- 기능
  - customer 검색
  - customer 정보 삽입
  - Customer 정보 변경
  - Customer 정보 삭제
- 커맨드
  - Info : 특정 옵션을 기준으로 customer 정보를 출력
    - -a : 전체 검색
    - -i : c_id 값으로 검색
    - -n : c_name 값으로 검색
    - -g : genre 값으로 검색
  - Insert : 새로운 customer 정보를 3개의 선호 장르와 함께 삽입
  - Update : 특정 옵션을 기준으로 입력한 c_id에 해당하는 customer의 정보를 변경
    - -m : email 정보를 변경
    - -p : 비밀번호 정보를 변경
    - -ph : 전화번호 정보를 변경
  - Delete : 입력한 c_id에 해당하는 customer의 정보를 삭제

customer.py (INFO)

```
c_id,c_name,email,gender,phone,preferred_genres
176,Kelsey Becker,q7Tf479@yahoo.com,F,+1 679-685-7236,"Animation, Mystery, Romance"
291,Stephen Tucker,Stephen8565@yahoo.com,M,+1 531-069-6582,"Adventure, Comedy, Drama"
742,Chelsea Copeland,s9VdH493@hotmail.com,F,+1 963-781-4447,"Action, Adventure, Romance"
776,Paul Dunn,s7Goq29@outlook.com,M,+1 298-579-0603,"Drama, News, Sport"
855,Sarah Chapman,Sarah92@google.com,F,+1 144-555-8515,"Animation, Crime, Drama"
869,Michelle Roberts,Michelle883@yahoo.com,F,+1 439-414-4036,"Animation, Drama, Romance"
1013,Stephen Greene,0MBhb4223@hotmail.com,M,+1 347-330-3861,"Adventure, Documentary, War"
1087,Jessica Clarke,Jessica924@yahoo.com,F,+1 335-309-1103,"Comedy, Game-Show, Horror"
1106,David Payne,David2506@google.com,M,+1 897-319-0338,"Animation, Sci-Fi, Thriller"
1319,Oscar Howard,8NNZf396@hotmail.com,M,+1 165-158-2795,"Adventure, Biography, Comedy"
```

- python customer.py info [-i/-n/-g/-a] [field value]
  - 공통
    - c_id, c_name, email, gender, phone, gr_name(preferred_genres) 모두 출력
    - 단, gr_name(preferred_genres) 는 3개를 모두 “,” 로 연결해서 출력할 것
      - 예시 참조
    - c_id 오름차순으로 정렬

  - -i : c_id 값으로 검색
  - -n : c_name 값으로 검색
  - -a 전체 검색
    - -a의 경우 customer 테이블의 데이터 중 c_id를 기준으로 오름차순 정렬, 입력한 숫자의 개수만큼 row가 출력
  - -g : genre 값으로 검색
    - -g의 경우 한가지 장르를 검색하는 것을 기준

customer.py (INSERT)

- python customer.py insert [customer_values] -g genre1 genre2 genre3
  - 해당 c_id 값이 없을 경우, error 메시지 출력
  - -g genre1 genre2 genre3 의 경우 순서 상관 없이 가능
  - Insert 완료 후, 새롭게 삽입된 데이터에 대해 display_info 함수로 출력

customer.py (UPDATE)

- python customer.py update -i C_ID [-m/-p/-ph/-gs] [new field value]
  - -i는 변경하고자 하는 customer data의 c_id의 값을 받기 위한 필수적인 입력
  - -m : e-mail 정보를 변경
  - -p : password 정보를 변경
  - -ph : 전화번호 정보를 변경
    - 전화번호 예시와 같이 공백을 포함한 문자열을 사용하기 위해선 "~ ~"를 사용
  - -gs : 선호 장르 정보를 변경
    - 선호 장르 정보 변경 시, 항상 장르 3개 (genre1 genre2 genre3) 입력

  - CLI로의 출력은 업데이트 전 정보 1번, 업데이트 후 정보 1번 -> 총 2번의 출력

customer.py (DELETE)

- python customer.py delete -i C_ID
    - -i는 삭제하고자 하는 customer data의 c_id의 값을 받기 위한 필수적인 입력

    - CLI로의 출력은 삭제될 c_id에 해당하는 정보를 출력

movie.py

```
python movie.py info [-a/-i/-n/-g/-sy/-ey/-ad/-r] [value]
```

- 기능
  - Movie 검색
- 커맨드
  - Info : 특정 옵션을 기준으로 customer 정보를 출력
    - -a : m_id 기준 오름차순
    - -i : m_id 값으로 검색
    - -n : m_name 값으로 검색
    - -g : genre 값으로 검색
    - -t : type 값으로 검색
    - -sy : start_year 값으로 검색
    - -ey : end_year 값으로 검색
    - -ad : is_adult 값으로 검색
    - -r : rating 값으로 검색

movie.py (INFO)

```
m_id,m_name,m_type,start_year,end_year,is_adult,runtimes,imdb_rating,final_rating,genres
54544,The Hathaways,tvSeries,1961-01-01,1962-01-01,False,30,7.1,7.100000000000000,Comedy
63866,The Andy Williams Show,tvSeries,1969-01-01,1971-01-01,False,60,7.9,7.9018867924528302,"Comedy, Music"
63916,Jimmy Durante Presents the Lennon Sisters,tvSeries,1969-01-01,1970-01-01,False,60,7.5,7.500000000000000,"Comedy, Music"
63919,The Johnny Cash Show,tvSeries,1969-01-01,1971-01-01,False,60,8.6,8.600000000000000,Music
71002,Kodiak,tvSeries,1974-01-01,1974-01-01,False,30,7.2,7.240000000000000,"Action, Adventure, Crime"
```

- python movie.py info [-a/-i/-n/-g/-sy/-ey/-ad/-r] [value]
  - 공통
    - m_id, m_name, m_type, start_year, end_year, is_adult, runtimes, m_rating(imdb_rating), 최종 평점(final_rating), gr_name(genres) 모두 출력
    - 단, gr_name(genres) 는 모두 “,” 로 연결해서 출력할 것
      - 예시 참조
    - 최종 평점은 다음과 같이 계산
      - (m_rating * votes + ct.rating의 합) / (votes + ct.rating의 수)
    - 출력은 m_id를 기준으로 오름차순 정렬하여 출력

  - -a : 모든 movie 검색
    - m_id 기준 오름차순
  - -i : m_id 값으로 검색
  - -n : m_name 값으로 검색
  - -g : genre 값으로 검색
    - -g의 경우 한가지 장르를 검색하는 것을 기준
  - -sy : start_year 값으로 검색
    - -sy,ey의 경우 입력한 값 이후의 연도를 검색하는 것을 기준
  - -ey : end_year 값으로 검색
    - -sy,ey의 경우 입력한 값 이후의 연도를 검색하는 것을 기준
  - -ad : is_adult 값으로 검색
  - -r : rating 값으로 검색
    - -r의 경우 입력한 값 이상을 검색하는 것을 기준

role.py

```
python role.py info [-a/-o] [value]
```

- 기능
  - 영화에서의 역할(role) 기반 참여자(participant) 검색
- 커맨드
  - Info : 특정 옵션을 기준으로 customer 정보를 출력
    - -a : m_id 기준 오름차순
    - -o : 특정 영화의 m_id 값으로, 해당 영화에서의 role만 검색

role.py (INFO)

```
p_id,p_name,role,m_name,casting
53,Robert Mitchum,actor,Brotherhood of the Rose,John Eliot
100,Rowan Atkinson,actor,Not the Nine O'Clock News,"Various, Various Roles"
228,Kevin Spacey,actor,House of Cards,Francis Underwood
282,Scott Bairstow,actor,Breaking News,Ethan Barnes
317,Clancy Brown,actor,Breaking News,Peter Kozyck
```

```
p_id,p_name,role,casting
638,William Shatner,actor,Buzz Howard
2079,Steve Forrest,actor,Samson Toey
80867,Abner Biberman,actor,"Abraham Lincoln Imhook, Mr. Imhook"
103071,Richard Bradford,actor,Brock
553436,A Martinez,actor,
907636,Clint Walker,actor,"Cal 'Kodiak' McCay, Cal 'Kodiak' McKay, Kodiak"
```

- python role.py info [-a/-o] [value]
  - 공통
    - p_id 기준 오름차순 정렬

  - -a : role 값으로 검색
    - p_id, p_name, role, m_name, casting(casting) 모두 출력
    - 단, casting(casting) 은 모두 “,” 로 연결해서 출력할 것
      - 예시 참조
  - -o : m_id 값과 role 값으로 검색
    - p_id, p_name, role, casting(casting) 모두 출력
    - 단, casting(casting) 은 모두 “,” 로 연결해서 출력할 것
      - 예시 참조

participant.py

python participant.py info [-a/-i/-n/-pr] [value]

- 기능
  - participant 검색
- 커맨드
  - Info : 특정 옵션을 기준으로 participant 정보를 출력
    - -a : p_id 기준 오름차순
    - -i : p_id 값으로 검색
    - -n : p_name 값으로 검색
    - -pr : profession name 값으로 검색

participant.py (INFO)

- python participant.py info [-a/-i/-n/-pr] [value]
  - 공통
    - p_id, p_name, major_work, ocu_name(profession) 모두 출력
    - 단, ocu_name(profession) 은 모두 "," 로 연결해서 출력할 것
      - 예시 참조
    - p_id 기준 오름차순 정렬

  - -a : p_id 기준 오름차순
  - -i : p_id 값으로 검색
  - -n : p_name 값으로 검색
  - -pr : profession name 값으로 검색
    - -pr의 경우, 한가지 직업을 검색하는 것을 기준

```
p_id,p_name,major_work,profession
53,Robert Mitchum,0039689,"actor, producer, writer"
92,John Cleese,0095159,"actor, producer, writer"
100,Rowan Atkinson,0274166,"actor, producer, writer"
228,Kevin Spacey,0119488,"actor, producer, writer"
231,Oliver Stone,0091763,"director, producer, writer"
```

참고 사항

- 필요 시 utils.py에 함수 추가 허용 (과제 점수에 영향 x)
- 최우선적으로 helpers/connection.py를 완성해야 postgreSQL에 연결 가능
- Customer.py의 display_info()는 편의를 위해 제공, 편집 허용
- 출력을 display_info로만 하라는 것은 아래의 txt 파일 캡쳐를 보면 2개의 출력이 있는데 이를 각각 display_info를 2번 호출하여 출력하고 update_customer에서 직접 print 하지 말라는 의미

```
Command : [ .₩customer.py update -i 9999 -m dblab@gmail.com ]
Total rows: 1
c_id | c_name | email               | gender | phone           | preferred_genres
-------------------------------------------------------------------------------
9999 | dblab  | dblab@hanyang.ac.kr | M      | +82 123-456-7890 | Action, Drama, Romance

Total rows: 1
c_id | c_name | email           | gender | phone           | preferred_genres
-------------------------------------------------------------------------------
9999 | dblab  | dblab@gmail.com | M      | +82 123-456-7890 | Action, Drama, Romance
```