**강의명: 데이터 베이스**

**실습: 12주차**

**학생 이름: 황귀훈**

**학번: 201710885**

**1. 다음 SQL문을 수행하는 Python 프로그램을 작성 하시오.**

텍스트, 명판, 점수판이(가) 표시된 사진

자동 생성된 설명이번 12주차 실습을 하기 위해 먼저 12주차 강의 Python/MySQL 환경 설정을 보고 “script.txt”을 다운로드 받은 후, MySQL에서 실행하여 DB를 생성한다

텍스트, 명판, 점수판이(가) 표시된 사진

자동 생성된 설명.

그 후 소스코드 MySQLPython.py를 추가한 뒤 pymysql패키지를 파이참에 설치 후 실습을 진행하였다. 실습 1의 select문과 update문을 수행하기 위해 MySQLPython클래스를 생성하여 함수를 작성하고 사용하였다.

클래스의 \_\_init\_\_에서 database와 python을 연동한다. 이 때 접속이 정상적으로 된다면 con이라는 object를 만들고 그 con의 cursor()를 사용하여 cursorObject를 하나 만들어 준다. 이 cursorObject는 DB와 상호작용을 할 때 쓰인다. \_\_del\_\_은 프로그램이 종료될 때 수행하는 함수로 close()함수를 사용하여 프로그램이 종료될 때 DBMS와의 연동을 끊게 했다.

실습1에서 사용될 함수로 select(sql), update(sql)함수를 작성했는데 각 함수는 인수로 주어진 sql문을 cursorObject의 execute()를 사용하여 DBMS로 보낼 수 있다.

sql문(selet)의 수행 결과를 보여주기 위해 cursorObject의 fetchall()을 사용하여 rows에 tuple형태로 data를 받아온 뒤 for문을 사용하여 출력한다. DB의 상태를 변경시키는 sql문의 경우(updqte, delete등) 데이터 베이스의 변경된 상태가 영구히 반영되도록 cursorObject의 commit()을 사용해야 한다.

**1.1**

테이블이(가) 표시된 사진

자동 생성된 설명

**1.2**

텍스트이(가) 표시된 사진

자동 생성된 설명

실습 1.1과 1.2의 수행 결과는 아래와 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1.1의 두 sql문을 각각 select()함수에 넣고 수행한 결과 instructor의 dept\_name이 ‘Comp. Sci’ 이면서 salary가 70000이상인 이름이 Kats와 Brandt로 출력되어 올바르다. 또한 instructor.id와 teaches.id가 같고 instructor.dept\_name = ‘Biology’인 instructor의 이름과 course\_id는 “Crick, BIO-101”과 “Crick, BIO-301”로 출력되어 올바르게 sql문을 수행함을 알 수 있다.

1.2의 update sql문을 수행한 결과 salary값이 1.05배 곱해져서 출력된 것을 볼 수 있다. 텍스트, 칠판, 점수판, 명판이(가) 표시된 사진

자동 생성된 설명

실제 데이터 베이스에서 확인 한 결과 올바르게 수행됨을 알 수 있다.

**2. 영화 검색 프로그램**

프로그램 목표는 Python을 이용하여 MySQL에 데이터를 저장하고 가져오는 작업을 구현하는 것이다. 실습 #2를 위해 작성한 함수는 다음과 같다.

**printCli():** 사용자가 명령어 인터페이스를 통해 데이터베이스를 사용할 수 있도록 인터페이스를 출력하는 함수

**createRelation():** 릴레이션을 생성하는 함수

**insertDatas():** 릴레이션에 데이터를 삽입하는 함수

**serchTitle():** 사용자가 입력한 키워드를 이용하여 영화를 검색하는 함수, 영화 제목에 사용자가 입력한 키워드를 포함하는 모든 영화들을 출력하는 함수

**searchViewers():** 사용자가 입력한 관객 수보다 더 많은 관객 수를 기록한 영화들 만을 출력하는 함수

**searchReleaseDate():** 사용자가 입력한 두 날짜 사이에 개봉된 영화들 만을 출력하는 함수

각 serch함수들은 sql문을 수행하는데 사용자가 입력한 keyword를 사용하여 조건에 맞는 영화들을 select한 뒤 출력하여 보여준다.

**2.1 명령어 인터페이스**

**printCli():** 사용자가 명령어 인터페이스를 통하여 데이터 베이스를 사용할 수 있도록 다음과 같은 명령어 기반 인터페이스를 출력하도록 작성하였다.

텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명

0을 입력한 경우 프로그램이 종료됨을 확인할 수 있다.

MySQLPython클래스의 \_\_del\_\_함수에 close()후에 ‘Exit Program’을 출력하게 하여 프로그램 종료 시 DBMS와의 연동이 끊어지는 것을 확인했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**2.2 릴레이션 생성 및 데이터 추가**

eCampus에 제공된 create\_table.txt를 이용하여 movie릴레이션을 생성하고, movie\_data.txt에 저장된 데이터를 movie릴레이션에 삽입하는 기능을 파이썬 프로그램 안에서 구현했다.

createRelation()함수를 사용하여 create\_table.txt을 open하여 read한 뒤 cursorObject의 execute()를 사용하여 movie릴레이션을 생성하고, insertDatas()함수를 사용하여 movie\_data.txt에 저장된 데이터를 read한 뒤 cursorObject의 execute()를 사용하여 movie\_data.txt의 데이터들을 movie릴레이션에 삽입했다. 데이터 베이스의 변경을 저장하기 위해 commit()을 호출했다.

1을 입력하여 릴레이션 생성 및 데이터를 추가한 결과 다음과 같다. (중간 생략)

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

정상적으로 데이터 베이스에 movie 릴레이션과 그 data들이 생성된 것을 확인했다.

**2.3**

2를 입력한 경우 영화 제목을 이용하여 릴레이션에서 영화를 검색한다.

‘7번방’을 입력하여

**Select \***

**From movie**

**Where title like ‘%7번방%’** (입력한 영화 제목을 포함한 모든 영화)

위와 같은 sql문이 수행된 결과는 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

‘7번방’을 포함하는 영화 ‘7번방의 선물’이 출력되어 프로그램이 정상 수행함을 알 수 있다.

**2.4**

3을 입력한 경우 입력한 관객 수보다 더 많은 관객 수를 기록한 영화들을 출력한다.

9000000을 입력하여

**Select \***

**From movie**

**Where totalnum > 9000000**

위와 같은 sql문이 수행된 결과는 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

영화 관객 수가 **9000000**을 초과한 영화들만 출력되어 프로그램이 정상 수행됨을 알 수 있다.

**2.5**

4를 입력한 경우 입력한 두 날짜 사이에 개봉된 영화들을 출력한다.

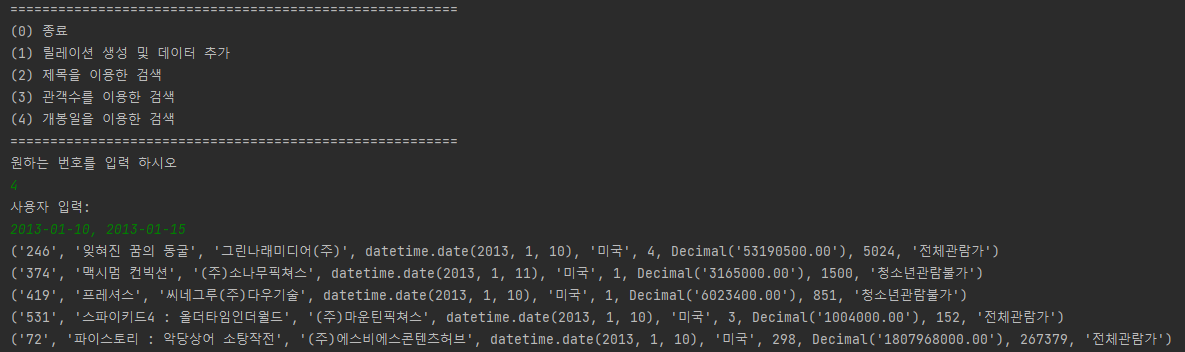
2013-01-10, 2013-01-15를 입력하여

**Select \***

**From movie**

**Where DATE(releasedate) between ‘2013-01-10’ and ‘2013-01-15’**

위와 같은 sql문이 수행된 결과는 다음과 같다.



입력한 두 날짜 사이에 개봉된 영화들만 출력되어 프로그램이 정상 수행함을 알 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명프로그램을 시작하면 while 문을 사용하여 0번을 입력해 프로그램을 종료할 때까지 계속 실습 2가 수행되도록 하였고, CLI의 0~4가 아닌 다른 입력을 하면 에러 메시지를 출력한 뒤 다시 입력 받도록 프로그램을 작성하였다.