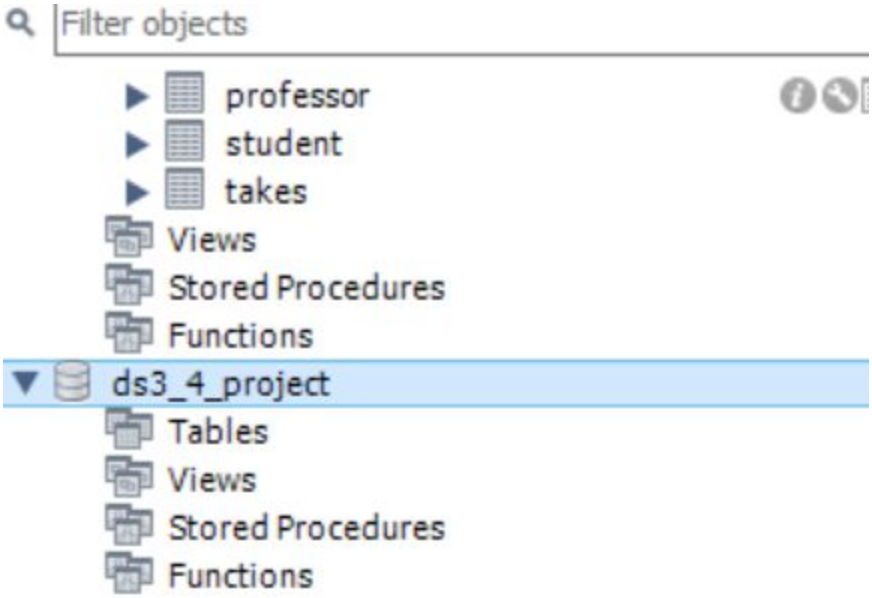
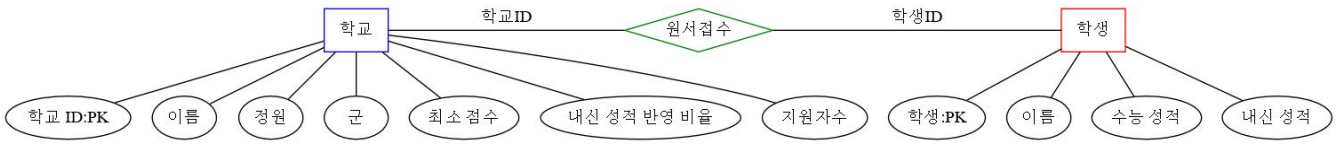
1. DB 접속정보
    a. host : ds1.snu.ac.kr
    b. 아이디 : ds3_4
    c. 비번 : 1q2w3e4r5t!
    d. Schema : ds3_4_project
    e. 

2. 테이블정의 : Key/Constraint/Null허용여부 등 체크
    a. 학교(Entity)
        i. - 학교 ID: 정수 (Primary key) ⇒ school_id
        ii. - 이름: 문자열 (최대 200 자) ⇒ school_name
        iii. - 정원: 정수 (1 이상의 값) ⇒ capacity
        iv. - 군: 문자열 (대문자 ‘A’, ‘B’, ‘C’ 중 1 개의 값을 가져야 함) ⇒ school_district
        v. - 최소 점수: 정수 (0 이상의 값) ==>min_scrore
        vi. - 내신 성적 반영 비율: 실수 (0 이상의 값) ==>adjut_ratio

    b. 학생(Entity)
        i. - 학생 ID: 정수 (Primary key) ⇒ student_id
        ii. - 이름: 문자열 (최대 200 자) ⇒ student_name
        iii. - 수능 성적: 정수 (0 이상 400 이하의 값) ⇒ test_score
        iv. - 내신 성적: 정수 (0 이상 100 이하의 값) ⇒ school_grades

    c. 원서접수(Relation)
        i. 학생 ID : 정수(Foreign Key) ⇒ student_id
        ii. 학교 ID : 정수(Foreign Key) ⇒ school_id



By `graphviz, engine='dot'`

[Table 생성 sql] ⇒ 검토 후 보완할 필요 있음 보완 진행

```
use ds3_4;
drop table if exists Schools cascade;
drop table if exists Students cascade;
drop table if exists Apply cascade;
create table Schools
(
school_id int unsigned primary key auto_increment,
school_name nvarchar(200) not null,
capacity int not null,
school_district char(1) not null,
min_score int unsigned not null,
adjust_ratio float unsigned not null,
constraint capacity_ck check (capacity > 1),
constraint school_district_ck check (school_district in ('A','B','C')),
constraint min_score_ck check (min_score >=0)
);

create table Students
(
        student_id int unsigned  primary key auto_increment,
   student_name nvarchar(200) not null,
   test_score int unsigned not null,
   school_grades int unsigned not null,
   constraint test_score_ck check (test_score between 0 and 400),
   constraint school_grades_ck check (school_grades_ck between 0 and 100)
);

create table Apply
(
        student_id int unsigned not null,
   school_id int unsigned not null,
   constraint pk_apply primary key(student_id,school_id),
        constraint fk_apply_student foreign key(student_id) references students(student_id),
   constraint fk_apply_school foreign key(shool_id) references students(shool_id)
);
```

3. 파이썬 내 프로그램 필요 기능
    a. 공통함수
        i. Main 함수(while문으로 구현)
        ii. 메뉴 입력 함수
        iii. DB connection 함수
    b. 개별함수

실행예시
가 - A, D, G
나 - B, E, H
다 - C, F, I, J

A - 1. Print all universities
A - 2. Print all students
B - 3. Insert a new university
B - 4. Remove a university
C - 5. Insert a new student
C - 6. Remove a student
D - 7. Make a application
E - 8. Print all students who applied for a university
F - 9. Print all universities a students applied for
G - 10. Print expected successful applicants of a university
H - 11. Print universities expected to accept a student
I - 12. Exit
J - 13. Reset database

- (1/30) Term Project
  - Due Date: 2019/02/15(Fri) 11:59PM
  - 제출: lecture@europa.snu.ac.kr
    - 파일명: PRJ_조이름.zip (예: PRJ_12조.zip)
    - 메일 제목: [DB Project] 조이름 – 조원 이름1, 조원 이름2, 조원 이름3 (예: [DB Project] 12조 – 홍길동, 김철수, 김영희)
  - 제출 내용 (압축하여 zip으로 제출)
    - 1. Runnable python script 파일 (파일명: main.py)
    - 2. DB 관련 정보
      - 2-1. DB 테이블 스키마 (CREATE 문)
      - 2-2. DB 접속 정보 (main.py에 잘 적어서 제출해 주세요.)
        - 반드시 truncate한 후 제출해주세요!
      - 3. 리포트 (pdf 포맷으로 제출해주세요!)
        - 파일명: PRJ_조이름.pdf (예: PRJ_12조.pdf)

-. **Project spec**
https://drive.google.com/open?id=1UUORK9xoLzJtFi2wx6F_GsrT8GNd6h9Q

결과 레포트(Implementing a Simple Database Application)

2조(김정훈/한정민/모유찬)

1. 핵심모듈

a. 외부 Module : pymysql

b. 내부 Module

i. querytodatabase : DB 접속 및 query 실행 요청(fetch/commit)

```
def querytodatabase(sql,querytype=0, *args):
    '''
    sql : sqlstatement,
    querytype : 0 => select clause, 1 => ddl clause,
    args : optional arguments for sql statement
    return (1)fetch 시 tuples / (2) insert,remove등 진행시 반영 행수(cursor.rowcount)
    '''
```

ii. Inputpredicate : 사용자로부터 추가/삭제/조회등 할때 잘못된 값을 입력 받았을 시 재 입력을 받도록 처리해주는 모듈

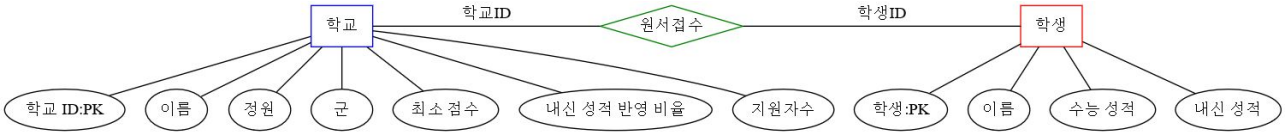
```
def inputwithpredicate(comment,dtype,predfunc=None,custom_errmsg=''):
    '''
        comment : 사용자 입력받을 때 입력안내 메시지,
        dtype : 입력받으려는 data type,
        --optional
        predfunc : 입력받는 데이터에 대해서 data type 외 제약조건을 설정할때(lambda이용 or 함수)
        errmsg : predfunc를 이용해서 검증할때 입력 에러 발생시 에러 메시지
    '''
```

iii. 어플리케이션기능 별 구현 : 기능별 함수로 분리

```
1:printalluniversities, #1.모든학교 정보 출력
2:printallstudents, #2.모든 학생 정보 출력
3:insertanewuniversity, #3.학교 추가
4:removeauniversity, #4.학교 삭제
5:insertanewstudent, #5. 학생 추가
6:removeastudent, #6. 학생 삭제
7:makeaapplication, #7. 원서 접수
8:printallstudentsappliedforauniversity, #8. 학교에 지원한 학생 목록 출력
9:printalluniversitiesastudentsappliedfor, #9. 학생의 원서 접수 목록 출력
10:printexpectedsuccessfulapplicantsofauniversity, #10. 학교의 예상 합격자 목록 출력
11:printuniversitiesexpectedtoacceptastudent, #11. 학생의 예상 합격 대상 목록 출력
13:resetdatabase, #데이터베이스 리셋 및 생성
14:dumptestdateset #test목적으로 만든 dump function
```

2. 구현내용

a. DB 구성



i. Entity : 학교(Schools) / 학생(Students)

ii. Relation : 원서접수 (Apply)

iii. Constaint 구현 :Trigger & stored procedure / Create 시 Constraint / Python 입력 제한

- MySql 의 경우 check constraint가 적용이 안되어, Trigger 로 입력 제한 필요

1. 대학교(Schools) : [학군 : Trigger 및 python처리 | 학교 ID : Auto_increment, PK | 그외 Data Type(Unsigend int) 으로 처리 가능]
2. 학생(Students) : [수능성적 및 내신성적 : Trigger 및 python처리 | 학생 ID : Auto_increment, PK | 그외 Data Type(Unsigned int) 으로 처리 가능]
3. 원서접수(Apply) : [학생 ID + 지원 학군 : Uniqueness | 학생ID : FK | 학교 ID : FK]

** Trigger 문

```
DELIMITER $$
create procedure check_school
(in pname nvarchar(200),
 in pcapacity int,
 in pdistrict char(2),
 in pminscore int,
 in padjustratio float
)
begin
    if pcapacity < 1 then
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Please enter value over 0 for Capacity';
    end if;

    if pdistrict not in ('A','B','C') then
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Please enter only A or B or C for school district';
    end if;
end$$
```

```
DELIMITER $$
create procedure check_student
(
    in pname nvarchar(200),
    in ptestscore int,
    in pschoolgrade int
)
begin
    if ptestscore < 0 or ptestscore > 400 then
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Please enter value between 0 and 400 for TestScore';
    end if;

    if pschoolgrade < 0 or pschoolgrade > 100 then
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Please enter value between 0 and 100 for SchoolGrade';
    end if;
end$$
DELIMITER ;
```

** Create 문

```
create table Schools
(
school_id int unsigned primary key auto_increment,
school_name nvarchar(200) not null,
capacity int unsigned not null,
school_district char(2) not null,
min_score int unsigned not null,
adjust_ratio float unsigned not null,
constraint capacity_ck check (capacity > 1),
constraint school_district_ck check (school_district in ('A','B','C')),
constraint min_score_ck check (min_score >=0)
);

create table Students
(
    student_id int unsigned  primary key auto_increment,
    student_name nvarchar(200) not null,
    test_score int unsigned not null,
    school_grades int unsigned not null,
    constraint test_score_ck check (test_score between 0 and 400),
    constraint school_grades_ck check (school_grades_ck between 0 and 100)
);

create table Apply
(
    student_id int unsigned not null,
    school_id int unsigned not null,
    constraint pk_apply primary key(student_id,school_id),
    constraint fk_apply_student foreign key(student_id) references Students(student_id),
    constraint fk_apply_school foreign key(school_id) references Schools(school_id)
);
```

b. 메뉴구성 : Dictionary (key : menu 번호 / value : menu구현 function)

```
menu_selection={
    1:printalluniversities,
    2:printallstudents,
    3:insertanewuniversity,
    4:removeauniversity,
    5:insertanewstudent,
    6:removeastudent,
    7:makeaapplication,
    8:printallstudentsappliedforauniversity,
    9:printalluniversitiesastudentsappliedfor,
    10:printexpectedsuccessfulapplicantsofauniversity,
    11:printuniversitiesexpectedtoacceptastudent,
    13:resetdatabase,
    14:dumptestdateset
}
```

```
def main():
    while True:
        print(menu_list)
        menu_num = eval(input('Select your action : '))
        if menu_num in menu_selection.keys():
            menu_selection[menu_num]()
        elif menu_num == 12:
            print('Bye!')
            break
        else:
            print('잘못된 번호 입력하였습니다')
```

c. Table 출력 : Header / Tail String 선언 , string.format 이용하여 구현

```
scholls_head ='''
-------------------------------------------------------------------------
id\tname\t\tcapacity\tgroup\tcutline\tweight\tappled
-------------------------------------------------------------------------'''

students_head ='''
-------------------------------------------------------------------------
id\tname\t\tcsat_score\tschool_score
-------------------------------------------------------------------------'''

tail='''
-------------------------------------------------------------------------'''
```

d. 핵심 Query Logic

i. 10번 : 학교 ID를 입력받아서, getpassstudentlist(temp)함수에 대입하여 지원학생 중 total_score(수능성적 + 내신성적*학교당 반영률) >= Min_score(최소점수) 인 학생의 list를 total_score / 내신성적으로 내림차순 정렬한 table t1을 만들고 capacity(정원) > 해당 table의 학생 수 이면 t1을 그대로 출력. 그렇지 않을 경우에는 t1에서 동점자의 갯수를 세아린 list temp(ex. 1등부터 동점자가 1명, 1명, 2명, 3명일 경우 list의 원소는 [1,1,2,3]) 에서 for문으로 원소를 하나씩 가져와 가져온 원소들의 합계 만큼씩 t1의 학생들을 가져와서 temp list에 저장하다가 합계가 capacity*1.1 이상이되면 break, temp에서 중복을 제거(set으로 변경)해서 출력

ii. 11번 : 학생 ID를 입력받아서 Apply table에서 지원 학교 list를 받아와 getpassstudentlist(temp)함수를 사용해 합격자 명단을 가져오고, 입력받은 학생 ID가 명단에 있으면 해당 학교를 출력

e. 각메뉴 구현 : Query 로 Select / Insert / Delete 문 구현 ⇒ querytodatabase 함수 호출

i. `select school_id, school_name, capacity,school_district,min_score,adjust_ratio,count(Apply.student_id) as appled from Schools Natural left outer join Apply group by school_id`

ii. `select * from Students`

iii. `insert into Schools(school_name,capacity,school_district,min_score,adjust_ratio) values(%s,%s,%s,%s,%s)`

iv. `delete from Schools where school_id = %s`

v. `insert into Students(student_name,test_score,school_grades) values (%s,%s,%s);`

vi. `delete from Students where student_id = %s`

vii. `insert into Apply select %s,school_id,school_district from Schools where school_id=%s`

viii. `select student_id, student_name, test_score,school_grades from Schools natural join Apply natural join Students where school_id =%s`

ix. `select school_id, school_name, capacity,  school_district, min_score, adjust_ratio,count(Apply.student_id) as appled from Students  natural join Apply natural join Schools where student_id = %s group by school_id`

x. `x = '(select school_id, school_name, student_id, student_name, test_score, capacity,  min_score,school_grades,`

```
(test_score+school_grades*adjust_ratio) as total_score from Students
natural join Apply natural join Schools where school_id = %s) as x'
y = 'select school_id, school_name, student_id, student_name,
capacity,test_score, min_score, total_score,school_grades  from '+x+'
where total_score>=min_score order by total_score desc,school_grades
desc'
t = 'select count(student_id) as t from (' + y + ') as y group by
total_score,school_grades order by total_score desc,school_grades desc
'
z = 'select count(student_id) as count_s from (' + y + ') as y group
by school_id'
```

xi.
```
x = 'select school_id from Students  natural join Apply natural join
Schools where student_id = %s'
y = 'select student_id,student_name,school_grades,test_score from
Students  where student_id = %s'
z = 'select school_id, school_name, capacity,  school_district,
min_score, adjust_ratio,count(Apply.student_id) as appled from Schools
natural join Apply where school_id = %s group by school_id '
```

xii. NA

xiii. DDL 문 string 으로 묶어 한줄씩 순차 진행(Drop table/trigger => Create table/trigger)

```
import pymysql
import pymysql.cursors

ddlscripts = '''
use project;|
drop table if exists Apply cascade;|
drop table if exists Schools cascade;|
drop table if exists Students cascade;|
DROP PROCEDURE IF EXISTS check_school;|
DROP PROCEDURE IF EXISTS check_student;|

CREATE TABLE `Schools` (
  `school_id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `school_name` varchar(200) NOT NULL,
  `capacity` int(11) NOT NULL,
  `school_district` char(2) NOT NULL,
  `min_score` int(10) unsigned NOT NULL,
  `adjust_ratio` float unsigned NOT NULL,
```

```
def resetdatabase():
    try:
        print('Start to reset database.')
        querylist = ddlscripts.replace('\n','').split('|')
        for q in querylist:
            print(q)
            querytodatabase(q,1)
        print('''
Done to reset database. Executed commands below.
1. Drop tables : Apply -> Schools & Students
2. Drop Procedure of Students & Schools for input constraint
3. Create tables : Students & Schools -> Apply
4. Reset Autoincrement columns : Students & Schools
3. Create Procedure and Trigger of Students & Schools for input constraint
Please enter new data.''')
    except Exception:
        print('Error occured while trying to reset database.')
```

f. 추가 처리 모듈 : 10번 / 11번 메뉴 (TBD …)

3. 구현하지 못한 내용 : 없음

4. 가정한 것들

a. 출력 string format 에 대해서 \t 만큼 만 공간 확보(그이상 데이터 저장될 시 정렬이 틀어지나, 이부분은 trivial 부분이라 생각하여 추가 구현 x)

b.
- 입력 시 각 입력의 데이터 타입은 지켜진다고 가정 (예: 정원의 입력으로 String 이 들어오지 않음)

입력이 다른 데이터타입이 들어오지않는다고 하였으나,혹시 잘못된 정보가 들어올 수 있다는 판단하에 입력 구문에 대한 예외처리 및 재입력 받을수 있도록(program crash 방지)

c.
- 여러 개의 입력을 받는 경우, 입력을 차례로 받다가 특정 입력에 대해 에러가 있을 시 그 즉시 에러 메시지를 출력하고 해당 명령을 종료함.

여러개의 입력 받다가 잘못된 정보 입력되면 다시 입력하는 번거로움이 있기 때문에 해당 명령 강제종료 대신 재입력 받아 정상 진행할 수 있도록 처리함(이게 맞는 방향으로 생각)

d. School name : 8~15자, student name : 1~7자 기준으로 결과 화면 정렬함.

5. 실행예시

```
================================================================================
1. print all universities
2. print all students
3. insert a new university
4. remove a university
5. insert a new student
6. remove a student
7. make an application
8. print all students who applied for a university
9. print all universities a student applied for
10. print expected successful applicants of a university
11. print universities expected to accept a student
12. exit
13. reset database
================================================================================
Select your action :
```

```
================================================================================
Select your action : 1


--------------------------------------------------------------------------------
id      name            capacity        group   cutline weight  appled
--------------------------------------------------------------------------------
1       Alabama A & M   12              A       82      62.0    3
2       University of   87              A       66      95.0    2
3       Amridge Unive   19              A       22      8.0     10
4       University of   28              A       86      77.0    2
5       Alabama State   97              A       24      88.0    4
6       The Universit   99              A       39      39.0    12
```

```
Select your action : 2

id      name            csat_score      school_score
1       AARON           341             61
2       ABAD JR         314             41
3       ABBATACOLA              55              32
4       ABBATE          111             98
5       ABBATEMARCO             283             24
6       ABBOTT          358             43
7       ABDELHADI               379             81
```

```
Select your action : 3
University name: adsasdasd
University capacity: as
Please enter integer/float value
University capacity: 23
University group: g
Please enter a value between A and C
University group: a
Cutline score: s
Please enter integer/float value
Cutline score: 23
Weight of high school records: 2
A university is successfully inserted.
```

```
Select your action : 7
Student ID : 300
School ID : 300
No data inserted because of either of no student_id or no school_id
```

```
Select your action : 10
school_id: 3

id      name            csat_score      school_score
28      ABUZANAT                51              271
49      ADAMIK          89              13
72      ADELMAN         4               205
83      ADREANI         14              328
101     AGSALUD         98              153
104     AGUILA          67              77
112     AHEARN          86              234
150     ALBERTS         4               158
173     ALEBICH         52              176
193     ALFINI          26              368
```

```
==========
Select your action : 11
student_id: 1

------------------------------------------------------------------------
id      name            capacity        group   cutline weight  appled
------------------------------------------------------------------------
25      Heritage Chri   76              A       54      82.0    6

------------------------------------------------------------------------


------------------------------------------------------------------------
id      name            capacity        group   cutline weight  appled
------------------------------------------------------------------------
115     Hair Academy    74              B       57      61.0    9

------------------------------------------------------------------------


------------------------------------------------------------------------
id      name            capacity        group   cutline weight  appled
------------------------------------------------------------------------
134     Arkansas Stat   39              C       65      48.0    8
```

```
==========
Select your action : 13
Start to reset database.
use project;
drop table if exists Apply cascade;
drop table if exists Schools cascade;
drop table if exists Students cascade;
DROP PROCEDURE IF EXISTS check_school;
DROP PROCEDURE IF EXISTS check_student;
CREATE TABLE `Schools` (  `school_id` int(10) unsigned NOT NULL AUTO_INCREMENT,  `school_name` varchar(200) NOT NULL,  `
int(11) NOT NULL,  `school_district` char(2) NOT NULL,  `min_score` int(10) unsigned NOT NULL,  `adjust_ratio` float uns
NULL,  PRIMARY KEY (`school_id`)) ENGINE=InnoDB AUTO_INCREMENT=166 DEFAULT CHARSET=utf8;
CREATE TABLE `Students` (  `student_id` int(10) unsigned NOT NULL AUTO_INCREMENT,  `student_name` varchar(200) NOT NULL,
ore` int(10) unsigned NOT NULL,  `school_grades` int(10) unsigned NOT NULL,  PRIMARY KEY (`student_id`)) ENGINE=InnoDB A
ENT=6095 DEFAULT CHARSET=utf8;
CREATE TABLE `Apply` (  `student_id` int(10) unsigned NOT NULL,  `school_id` int(10) unsigned NOT NULL,  `school_distric
 DEFAULT NULL,  PRIMARY KEY (`student_id`,`school_id`),  UNIQUE KEY `uk_student_id_school_district` (`student_id`,`schoo
`),    CONSTRAINT `fk_apply_school` FOREIGN KEY (`school_id`) REFERENCES `Schools` (`school_id`),  CONSTRAINT `fk_apply_
OREIGN KEY (`student_id`) REFERENCES `Students` (`student_id`)) ENGINE=InnoDB DEFAULT CHARSET=utf8;
ALTER TABLE Schools AUTO_INCREMENT=1;
ALTER TABLE Students AUTO_INCREMENT=1;
create procedure check_school(in pname nvarchar(200), in pcapacity int, in pdistrict char(2), in pminscore int, in padju
oat)begin    if pcapacity < 1 then        SIGNAL SQLSTATE '45000'            SET MESSAGE_TEXT = 'Please enter value over
acity';    end if;    if pdistrict not in ('A','B','C') then        SIGNAL SQLSTATE '45000'            SET MESSAGE_TEXT
enter only A or B or C for school district';    end if;end
CREATE TRIGGER `check_school_before_insert` BEFORE INSERT ON `Schools`FOR EACH ROW    BEGIN        CALL check_school(new
me,new.capacity,new.school_district,new.min_score,new.adjust_ratio);END
CREATE TRIGGER `check_school_before_update` BEFORE UPDATE ON `Schools`FOR EACH ROW    BEGIN        CALL check_school(new
me,new.capacity,new.school_district,new.min_score,new.adjust_ratio);END
create procedure check_student(    in pname nvarchar(200),    in ptestscore int,    in pschoolgrade int )begin    if pte
0 or ptestscore > 400 then        SIGNAL SQLSTATE '45000'            SET MESSAGE_TEXT = 'Please enter value between 0 an
TestScore';    end if;    if pschoolgrade < 0 or pschoolgrade > 100 then        SIGNAL SQLSTATE '45000'            SET M
T = 'Please enter value between 0 and 100 for SchoolGrade';    end if;end
CREATE TRIGGER `check_student_before_insert` BEFORE INSERT ON `Students`FOR EACH ROW    BEGIN        CALL check_student(
t_name,new.test_score,new.school_grades);END
CREATE TRIGGER `check_student_before_update` BEFORE UPDATE ON `Students`FOR EACH ROW    BEGIN        CALL check_student(
t_name,new.test_score,new.school_grades);END

Done to reset database. Executed commands below.
1. Drop tables : Apply -> Schools & Students
2. Drop Procedure of Students & Schools for input constraint
3. Create tables : Students & Schools -> Apply
4. Reset Autoincrement columns : Students & Schools
3. Create Procedure and Trigger of Students & Schools for input constraint
```

6. 컴파일과 실행 방법 : main.py 파일 실행(pymysql 필요)

a. 만약 dump data 사용할 것이면 첨부 student_list.csv/university_list.csv 를 main.py와 동일 폴더에 저장하고 메뉴14 실행. 물론 12번으로 db초기화를 먼저 하는것이 좋음

7. 프로젝트하면서 느낀점

: 조원과 파이썬으로 첫 팀 프로젝트를 진행하였는데, Git을 이용하여 어느정도 성공적으로 공동 코드 작성을 한점, sql query 작성시 어려운 점이 많았는데, 서로 논의를 해서 풀어나간 점 등에서 전체적으로 성공적으로 프로젝트를 진행하였다고 생각합니다. 실제 점수 채점과 상관없이 처음 공동 프로젝트를 한 것 치고는 상당히 만족스러운 결과물을 만들어내서 뿌듯합니다.