1~40번 Idea외에 자기가 사용한 SQL문 있으면 모두 작성해 놓을것!

use registration;

create table student(

student\_id varchar(12) NOT NULL UNIQUE KEY,

password varchar(45) NOT NULL,

name varchar(45) NOT NULL,

department varchar(45) NOT NULL,

term varchar(45),

division varchar(45),

PRIMARY KEY(student\_id)

) ENGINE = InnoDB;

CREATE TABLE subject(

subject\_id varchar(100) not null,

term varchar(10) not null,

subject\_name varchar(100) not null,

division varchar(100) DEFAULT NULL,

is\_major tinyint(1) NOT NULL,

class\_hours INT NOT NULL,

professor\_name varchar(100) DEFAULT NULL,

day1 varchar(10) DEFAULT NULL,

time1 varchar(10) DEFAULT NULL,

day2 varchar(10) DEFAULT NULL,

time2 varchar(10) DEFAULT NULL,

people INT,

pre\_subject\_name varchar(100),

PRIMARY KEY(subject\_id, term)

) ENGINE = InnoDB;

CREATE TABLE subject\_eval(

subject\_id VARCHAR(100) NOT NULL,

term VARCHAR(10) NOT NULL,

subject\_eval varchar(2000) NOT NULL,

subject\_grade double,

CONSTRAINT FOREIGN KEY(subject\_id,term) REFERENCES subject(subject\_id,term) ON UPDATE CASCADE

) ENGINE = InnoDB;

show table status;

create table pre\_subject(

subject\_name varchar(100) not null primary key,

pre\_subject\_name varchar(100) not null

) engine=InnoDB;

create table score(

student\_id varchar(12) NOT NULL UNIQUE KEY,

term varchar(45),

credit\_hour int,

grade\_average double,

major\_grade\_average double,

electives\_grade\_average double,

PRIMARY KEY(student\_id),

CONSTRAINT FOREIGN KEY(student\_id) REFERENCES student(student\_id) ON UPDATE CASCADE

) engine=InnoDB;

create table attend(

student\_id varchar(12) NOT NULL UNIQUE KEY,

subject\_id varchar(100) not null,

term varchar(10) not null,

grade varchar(10),

CONSTRAINT FOREIGN KEY(student\_id) REFERENCES student(student\_id) ON UPDATE CASCADE,

CONSTRAINT FOREIGN KEY(subject\_id,term) REFERENCES subject(subject\_id,term) ON UPDATE CASCADE

) engine=InnoDB;

create table notice(

student\_id varchar(12) NOT NULL UNIQUE KEY,

notice\_id int auto\_increment primary key,

notice\_title varchar(100) not null,

notice\_content varchar(1000) not null,

notice\_writer varchar(20) not null,

notice\_write\_day date not null,

notice\_file varchar(300) not null, #어떻게 저장할지에 따라... 이건 로컬에 저장할때 이야기.

CONSTRAINT FOREIGN KEY(student\_id) REFERENCES student(student\_id) ON UPDATE CASCADE

) engine=InnoDB;

create table time\_log(

time\_id int auto\_increment primary key,

save\_time datetime

) engine=InnoDB;

#drop table subject\_eval

# alter table subject\_eval

#add CONSTRAINT FOREIGN KEY(subject\_id,term) REFERENCES subject(subject\_id,term) ON UPDATE CASCADE;

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

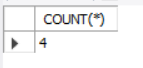
1.

* SQL  
  SELECT COUNT(\*) FROM student\_grade WHERE subject\_id IN(

SELECT subject\_id FROM subject WHERE subject\_name IN (

SELECT prerequisite FROM subject WHERE subject\_id = 'H000-1-4626-02'));

* 기능 : 특정 과목의 선수과목을 이수한 학생 수
* 결과(Image)

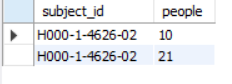


2.

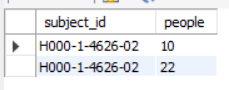
* SQL  
  UPDATE subject SET people = people + 1

WHERE term = '2020-2' AND subject\_id = 'H000-1-4626-02';

* 기능 : 개설된 교과목의 인원 수 +1
* 결과(Image)



(업데이트 전)



(업데이트 후)

3.

* SQL

select \* from student where department = '컴퓨터정보공학부';

* 기능 : 학과 소속 학생 나열
* 결과(Image)



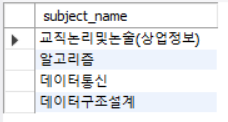
4.

* SQL

SELECT DISTINCT subject\_name FROM subject WHERE is\_major = 1 AND subject\_id IN(

SELECT subject\_id FROM student\_grade WHERE student\_id = '2015722083' AND grade LIKE 'A%');

* 기능 : 특정 학번의 수강한 전공과목 성적 중 A이상인 과목명을 출력
* 결과(Image)



5.

* SQL

DELETE FROM subject WHERE people <= 10 AND is\_major = 0 AND term =

'2020-2';

* 기능 : 10명 미만인 과목 삭제
* 결과(Image)



(삭제 전)

(삭제 후) : 검색안됨

6.

* SQL

* 결과(Image)

7.

* SQL

* 결과(Image)

8.

* SQL

* 결과(Image)

9.

* SQL

* 결과(Image)

10.

* SQL

* 결과(Image)

11.

* SQL

* 결과(Image)

12.

* SQL

* 결과(Image)

13.

* SQL

* 결과(Image)

14.

* SQL

* 결과(Image)

15.

* SQL

* 결과(Image)

16.

* SQL

* 결과(Image)

17.

* SQL

* 결과(Image)

18.

* SQL

* 결과(Image)

19.

* SQL

* 결과(Image)

20.

* SQL

* 결과(Image)

21.

* SQL

* 결과(Image)

22.

* SQL

* 결과(Image)

23.

* SQL

* 결과(Image)

24.

* SQL

* 결과(Image)

25.

* SQL

* 결과(Image)

26.

* SQL

* 결과(Image)

27.

* SQL

* 결과(Image)

28.

* SQL

* 결과(Image)

29.

* SQL

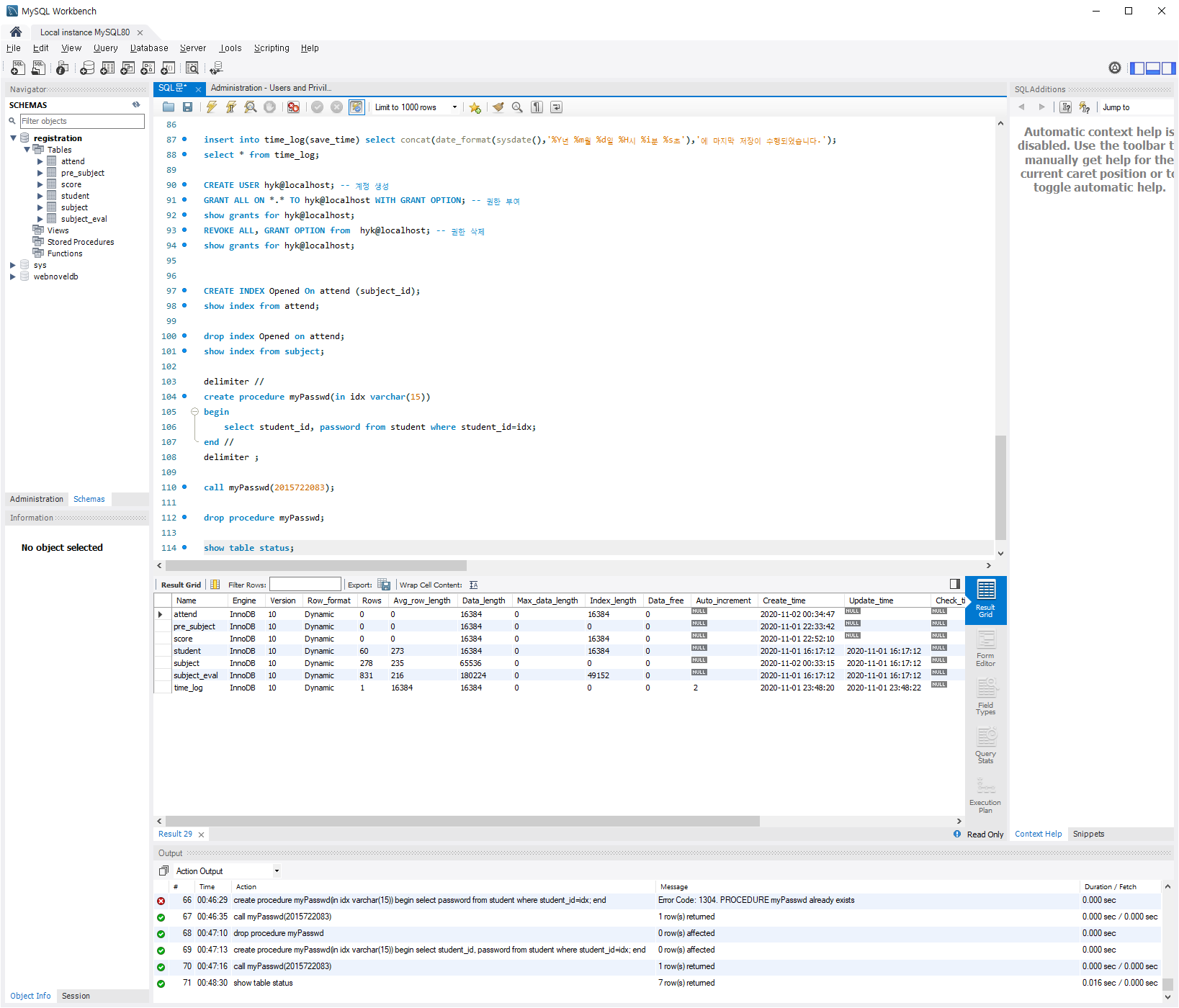
* 결과(Image)

30.

* SQL

show table status; **이건 여기저기서 사용되니 적절히 제거**

* 결과(Image)



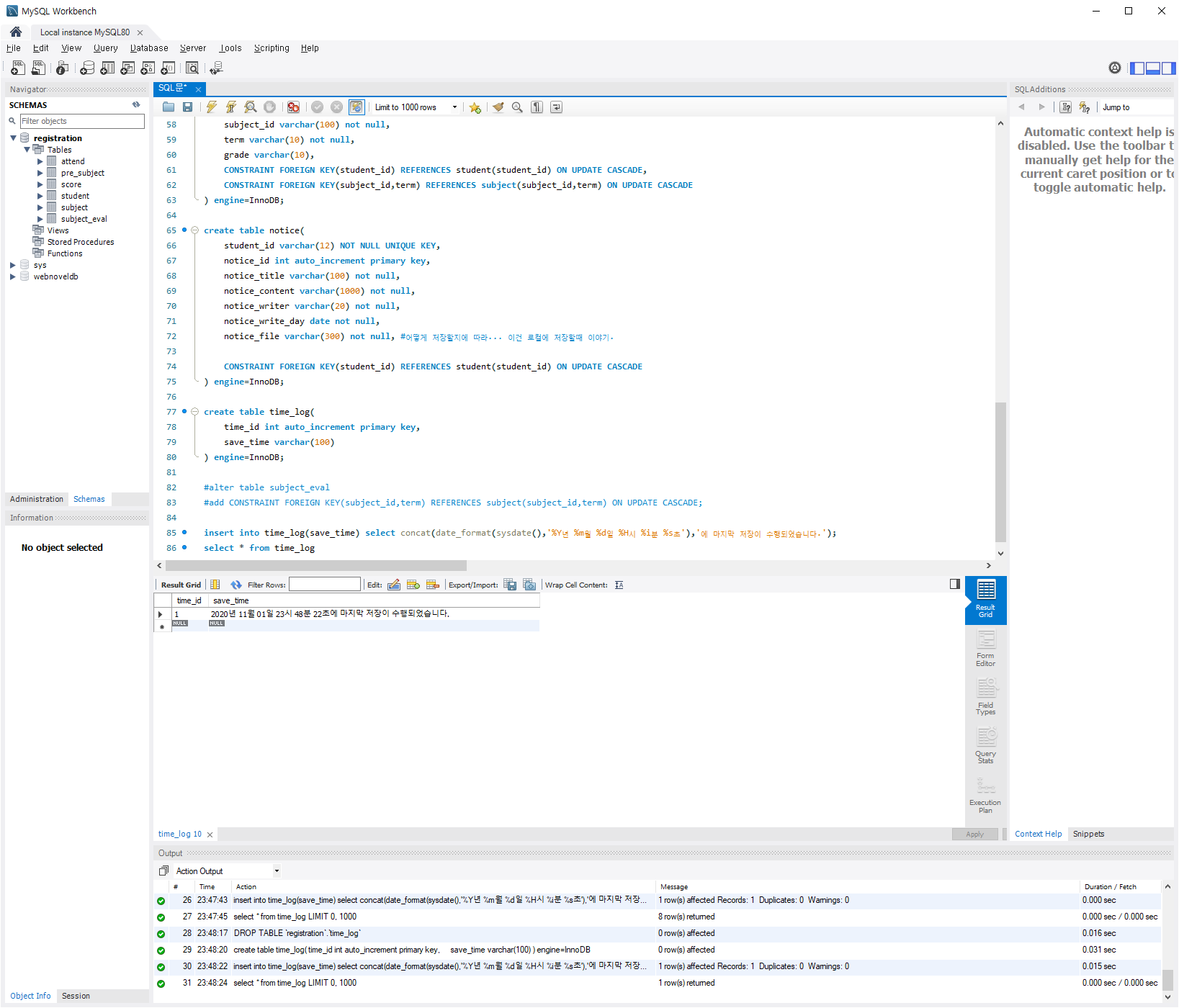
31.

* SQL

insert into time\_log(save\_time) select concat(date\_format(sysdate(),'%Y년 %m월 %d일 %H시 %i분 %s초'),'에 마지막 저장이 수행되었습니다.');

select \* from time\_log

* 결과(Image)



32.

* SQL

* 결과(Image)

33.

* SQL

* 결과(Image)

34.

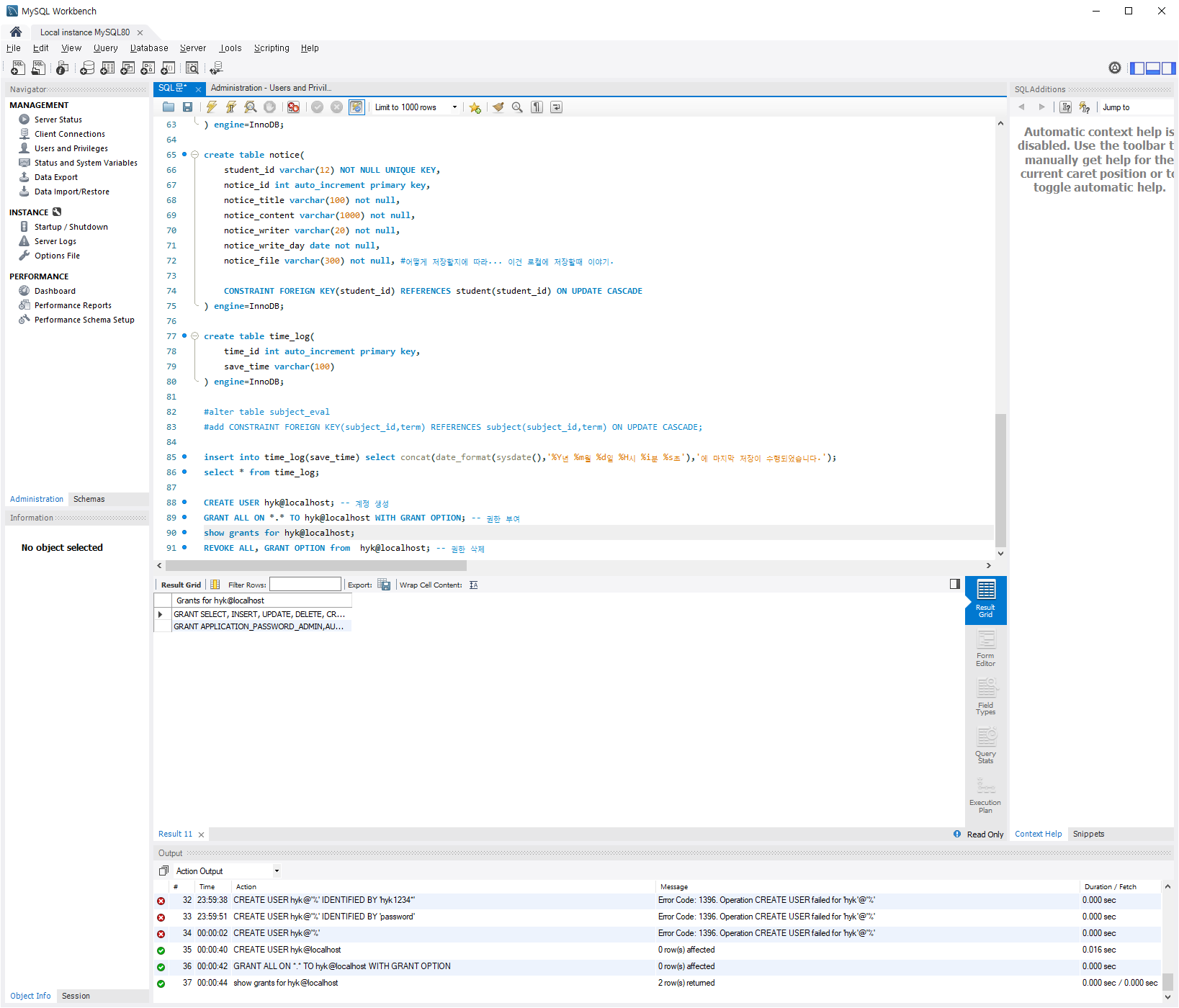
* SQL
* Navigator의 Administration - Users and privilige -> Add hyk이 선행되어야함

CREATE USER hyk@localhost; -- 계정 생성

GRANT ALL ON \*.\* TO hyk@localhost WITH GRANT OPTION; -- 권한 부여

show grants for hyk@localhost;

* 결과(Image) **\* 수정필요**



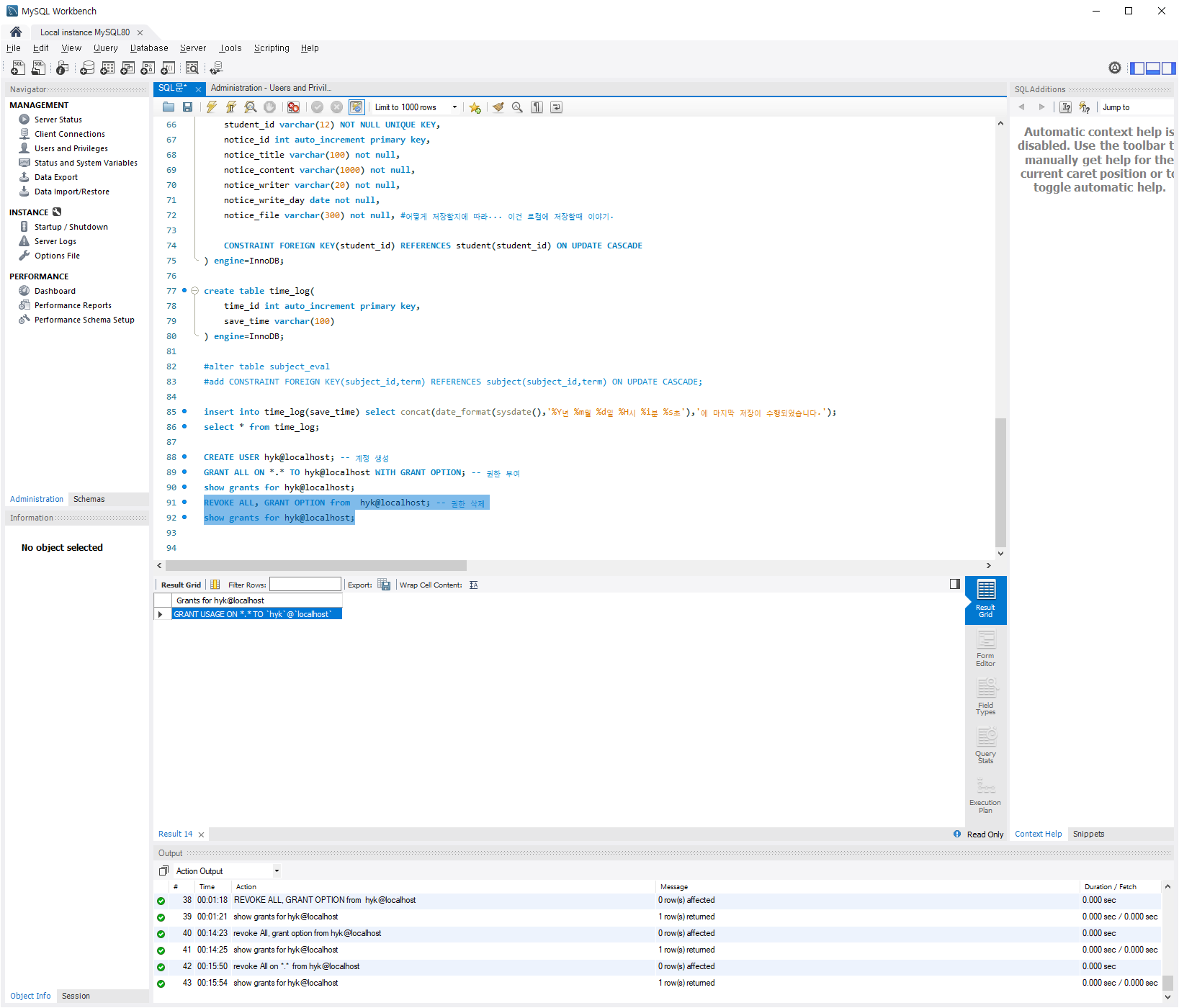
35.

* SQL

REVOKE ALL, GRANT OPTION from hyk@localhost; -- 권한 삭제

show grants for hyk@localhost;

* 결과(Image) **\* 수정필요**



36.

* SQL

create table time\_log(

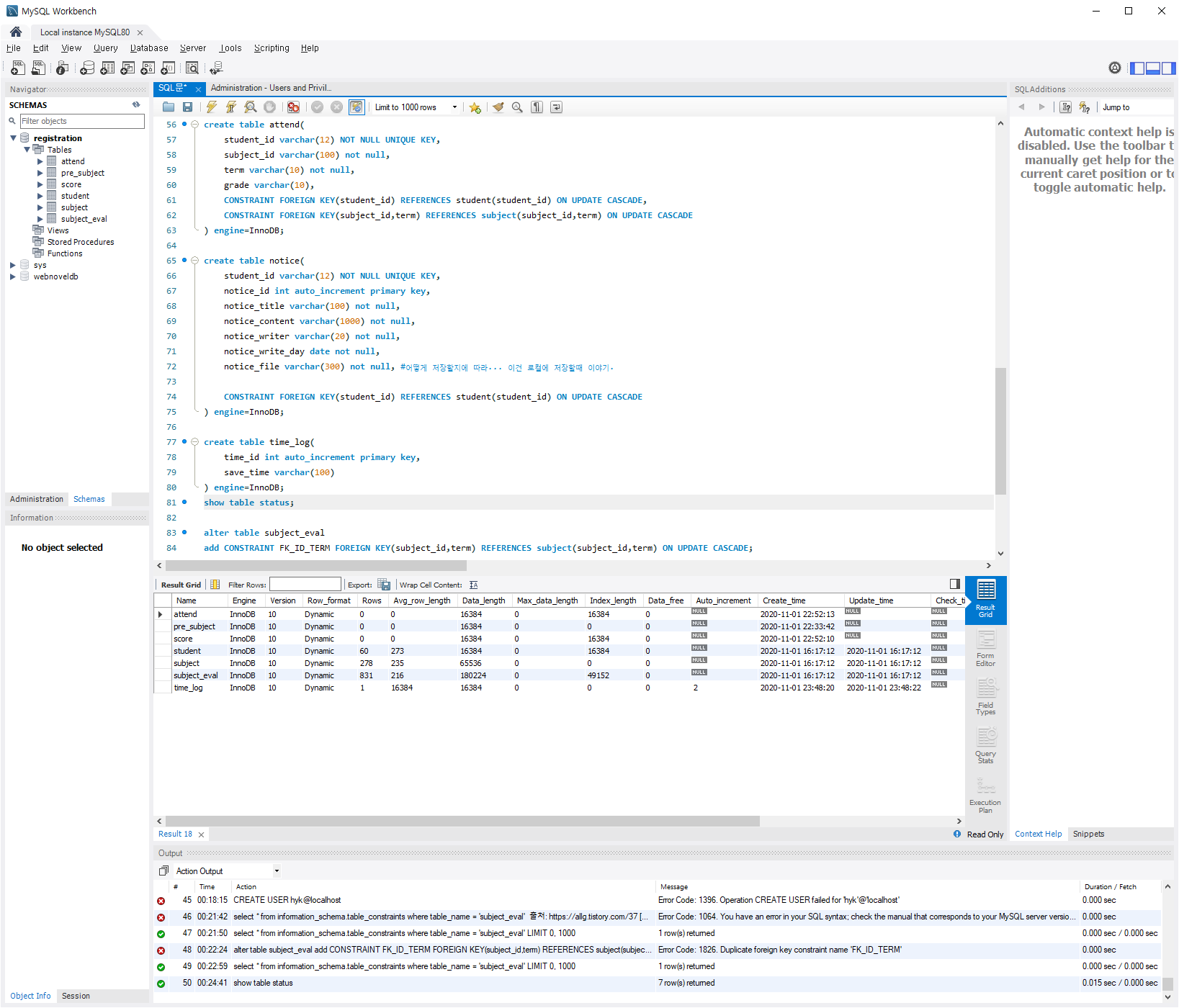
time\_id int auto\_increment primary key,

save\_time varchar(100)

) engine=InnoDB;

show table status;

* 결과(Image)



37.

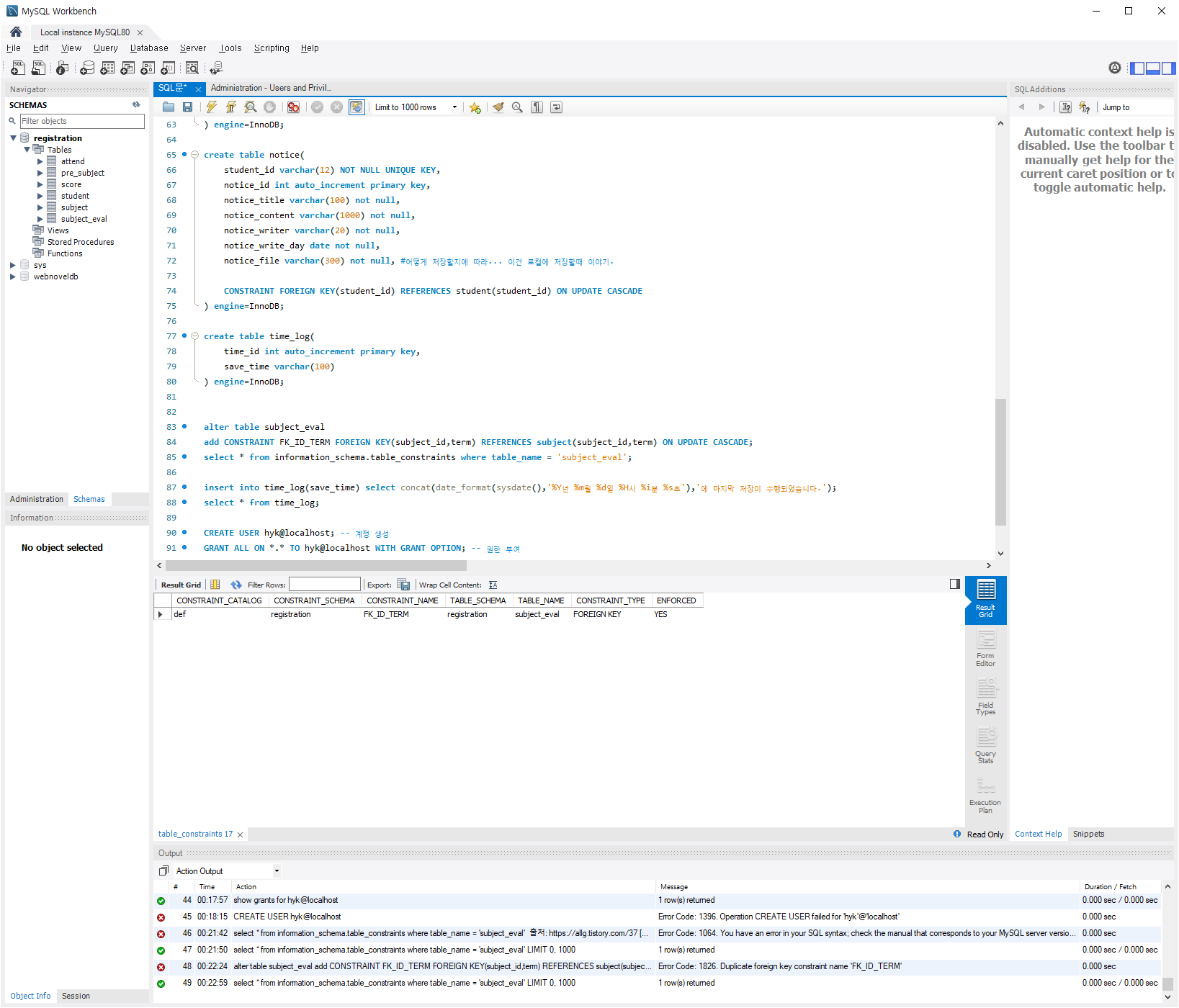
* SQL

alter table subject\_eval

add CONSTRAINT FK\_ID\_TERM FOREIGN KEY(subject\_id,term) REFERENCES subject(subject\_id,term) ON UPDATE CASCADE;

select \* from information\_schema.table\_constraints where table\_name = 'subject\_eval';

* 결과(Image)



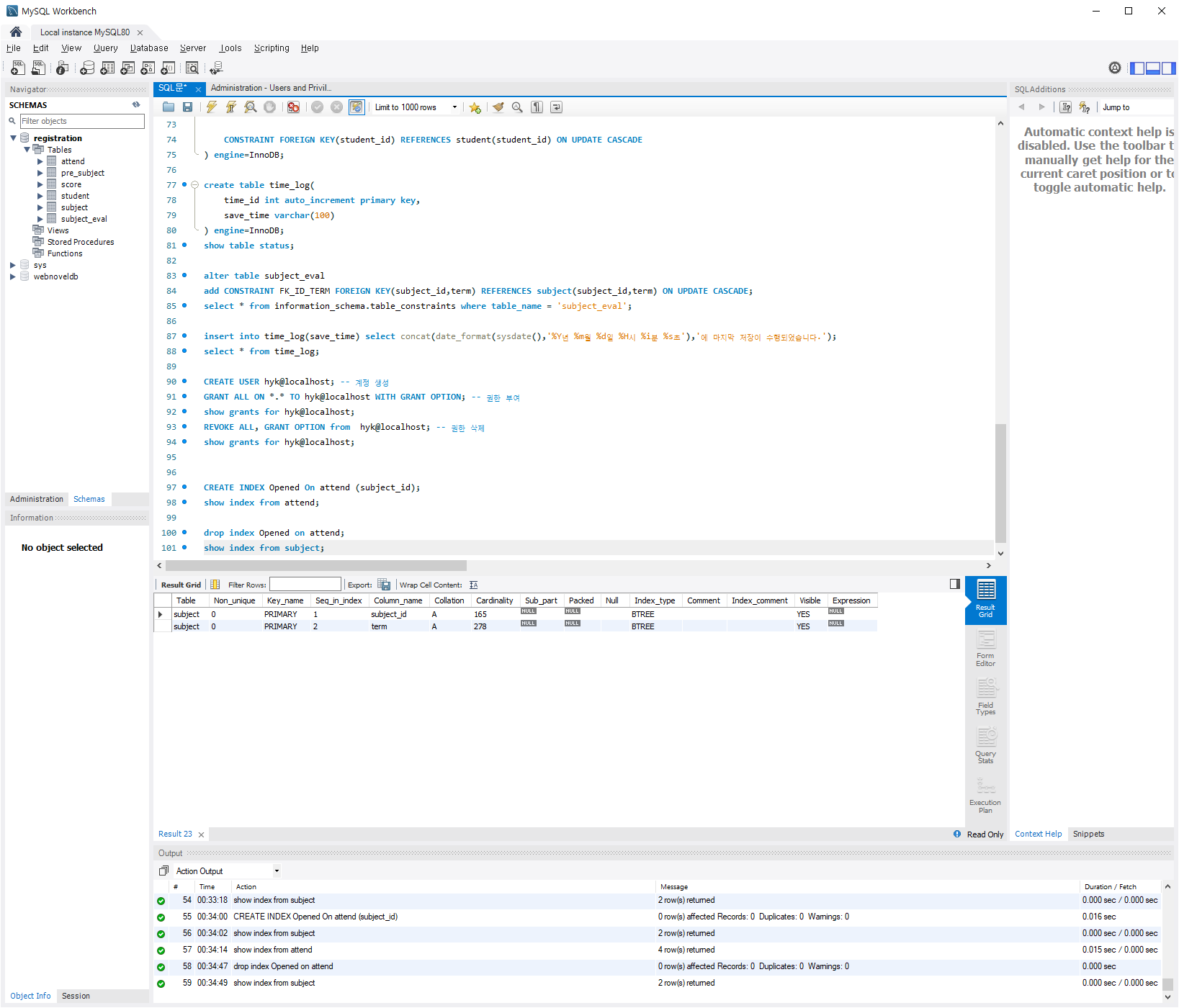
38.

* SQL

drop index Opened on attend;

show index from subject;

* 결과(Image)



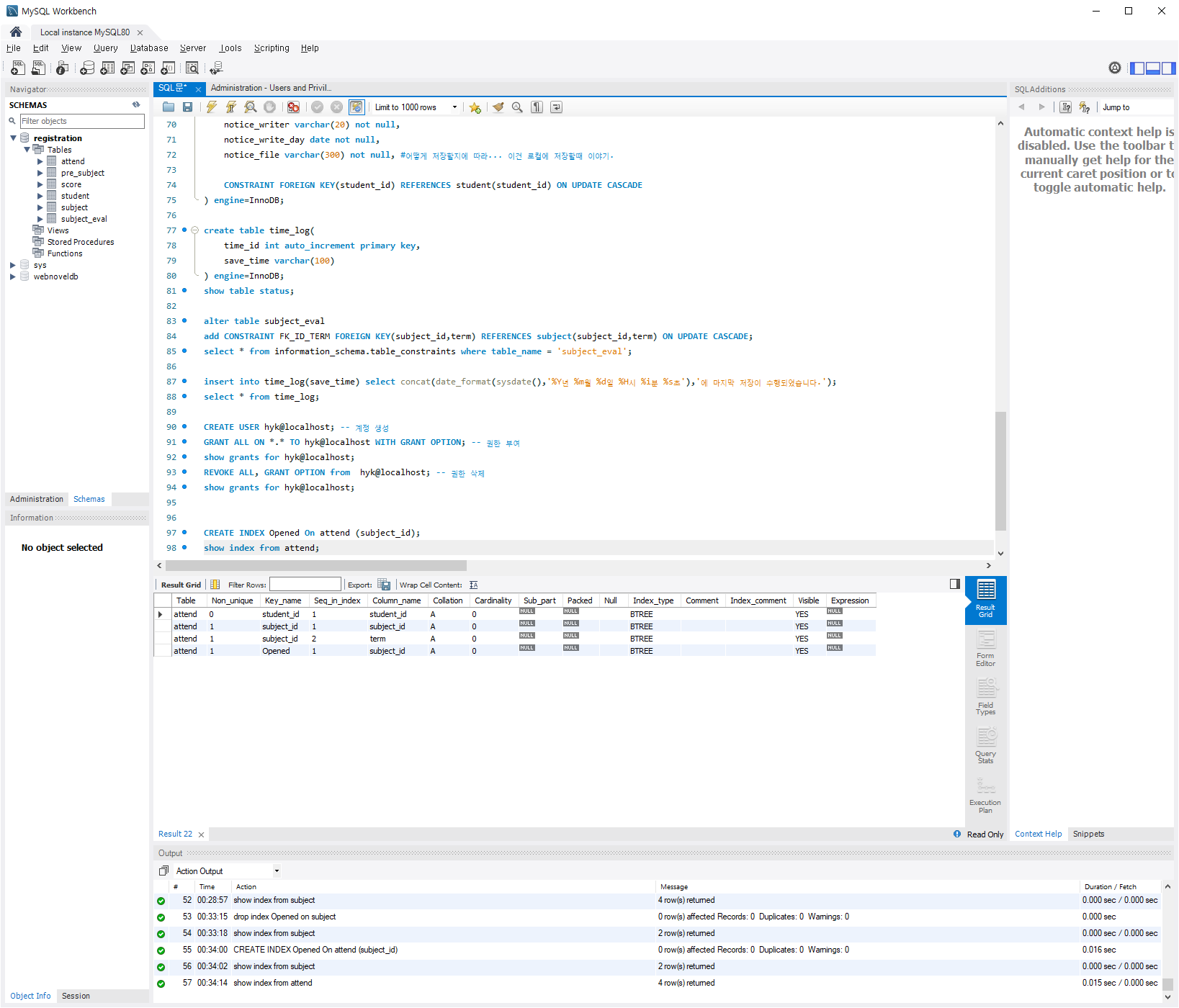
39.

* SQL

CREATE INDEX Opened On subject (subject\_id,term) order by subject\_id;

show index from subject;

* 결과(Image)



40.

* SQL

delimiter //

create procedure myPasswd(in idx varchar(15))

begin

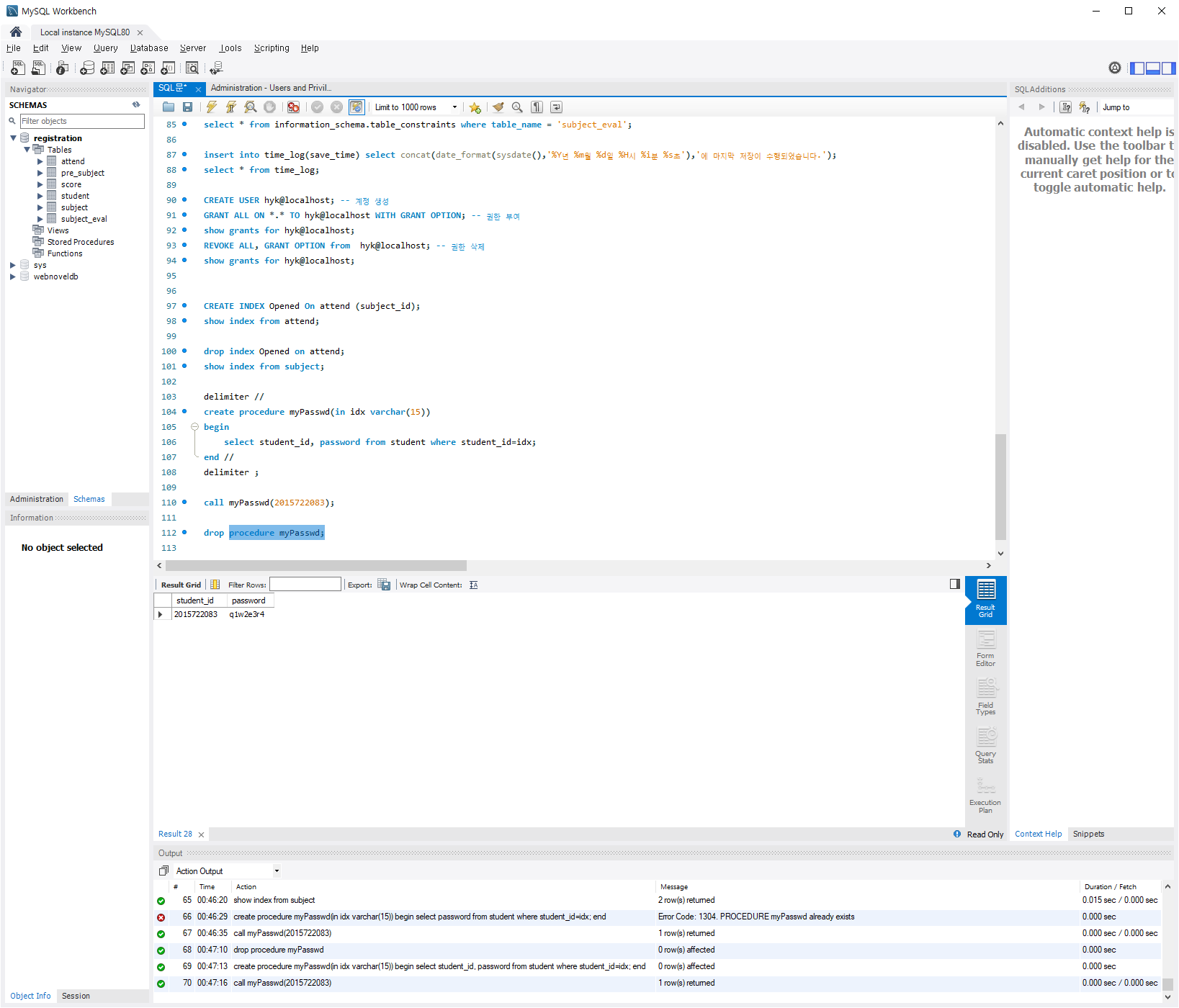
select student\_id, password from student where student\_id=idx;

end //

delimiter ;

call myPasswd(2015722083);

* 결과(Image)



41.

* SQL

insert into attend values('2015722083','0000-2-0254-01','2020-1','A+'),('2015722083','0000-2-0258-01','2020-1','A');

select avg(replace(replace(grade,'A+','4.5'),'A','4.0')) as avg\_grade from attend;

* 결과(Image)

