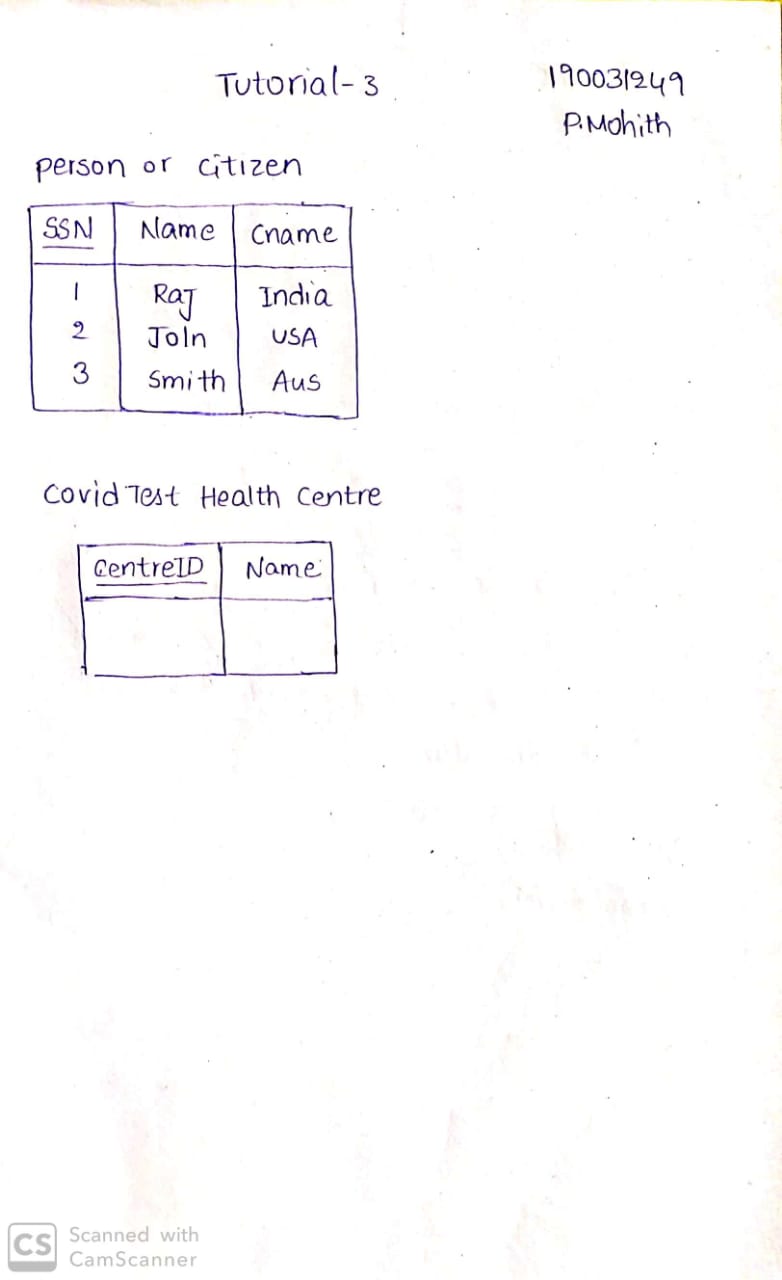
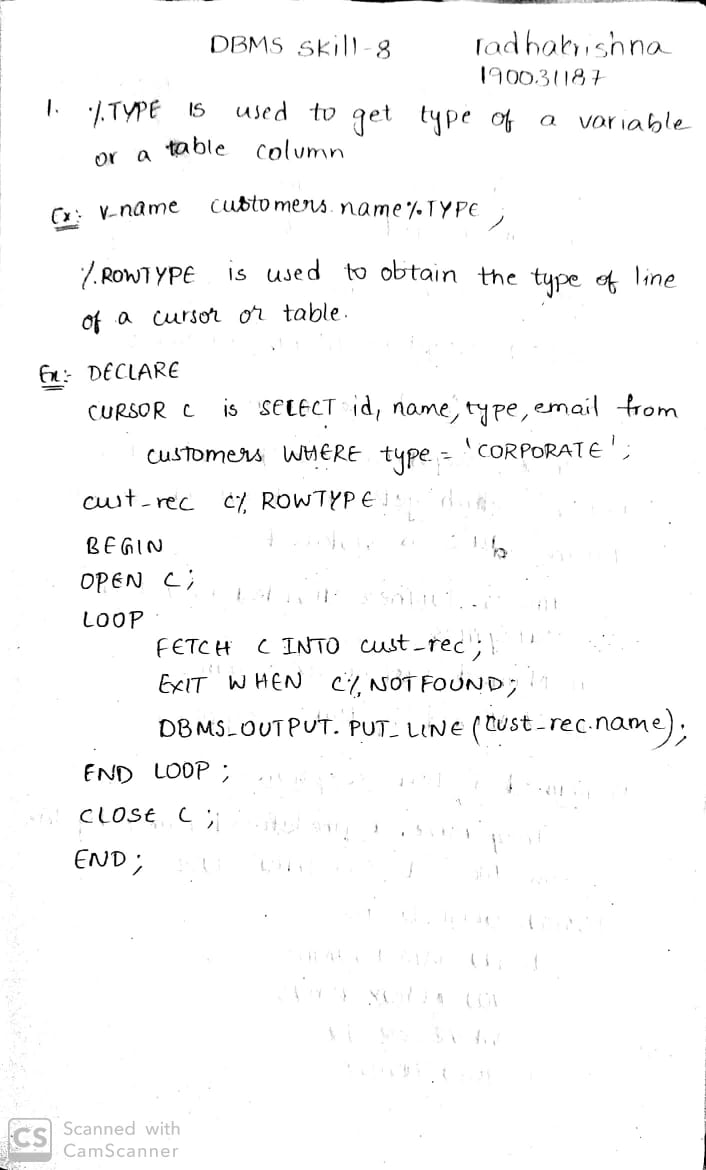
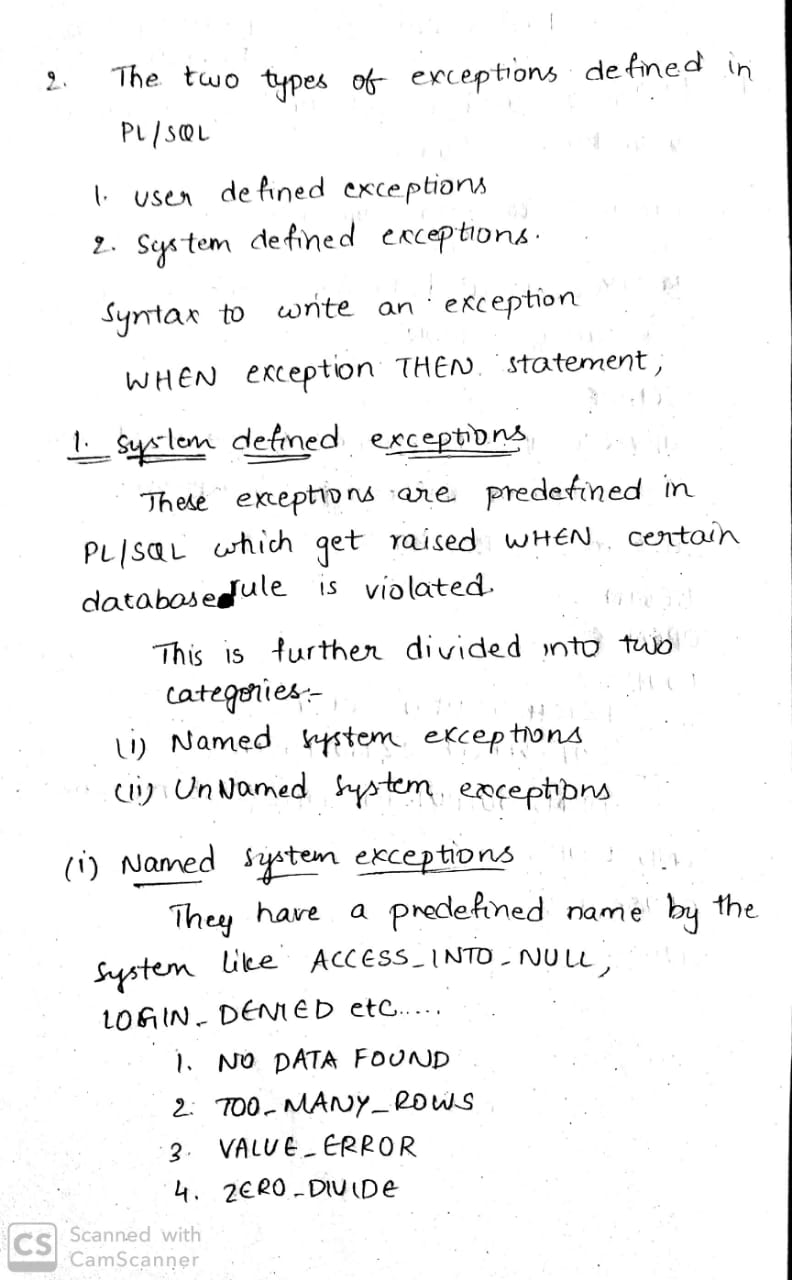
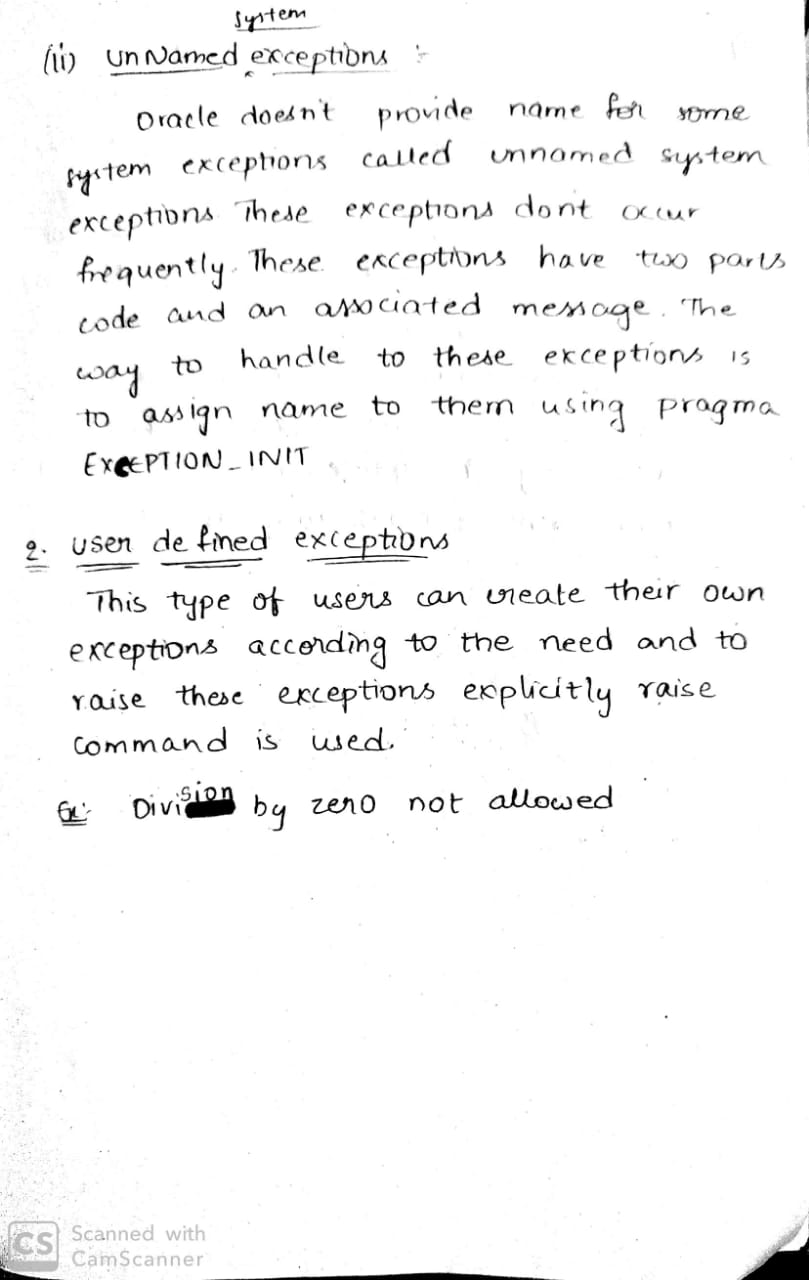
**PRELAB**

* 1. Differentiate between % ROWTYPE and TYPE RECORD.

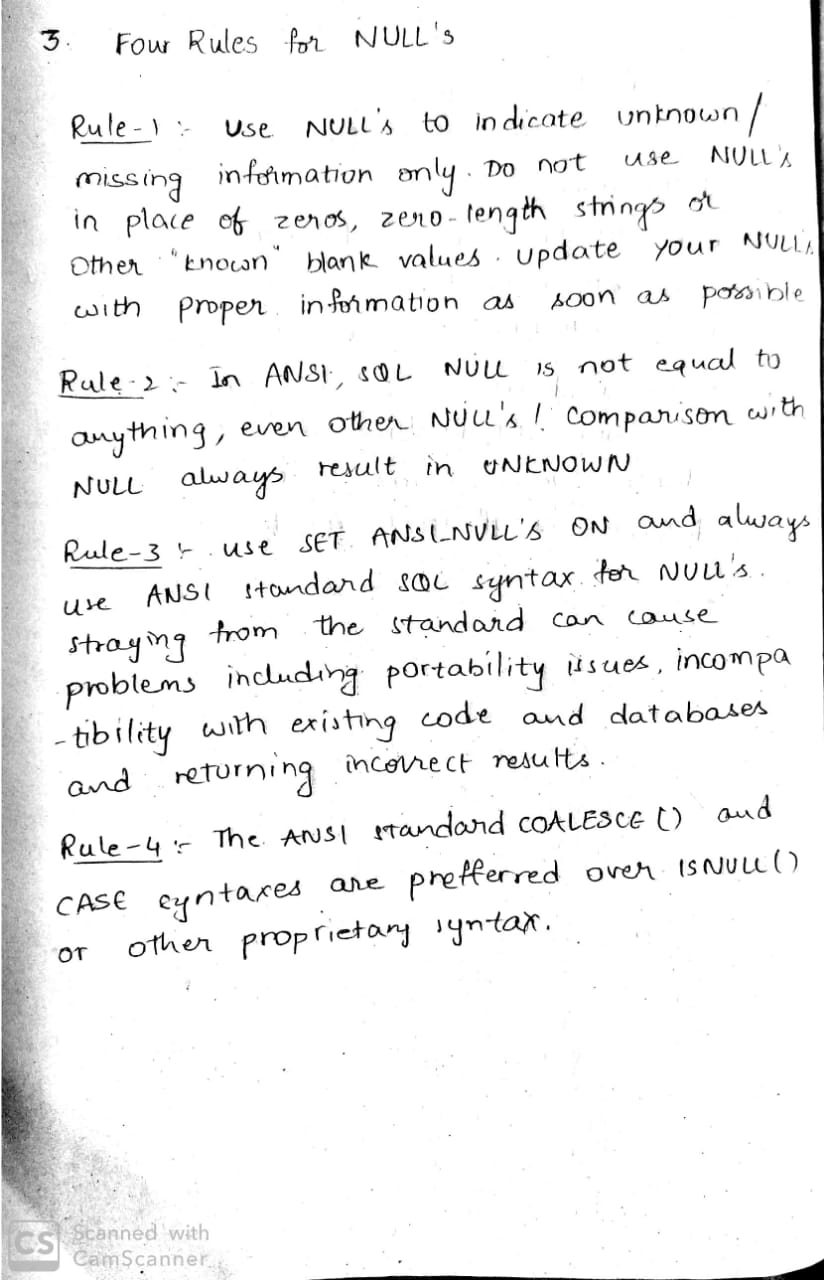
****` 

* 1. What are the two types of exceptions?

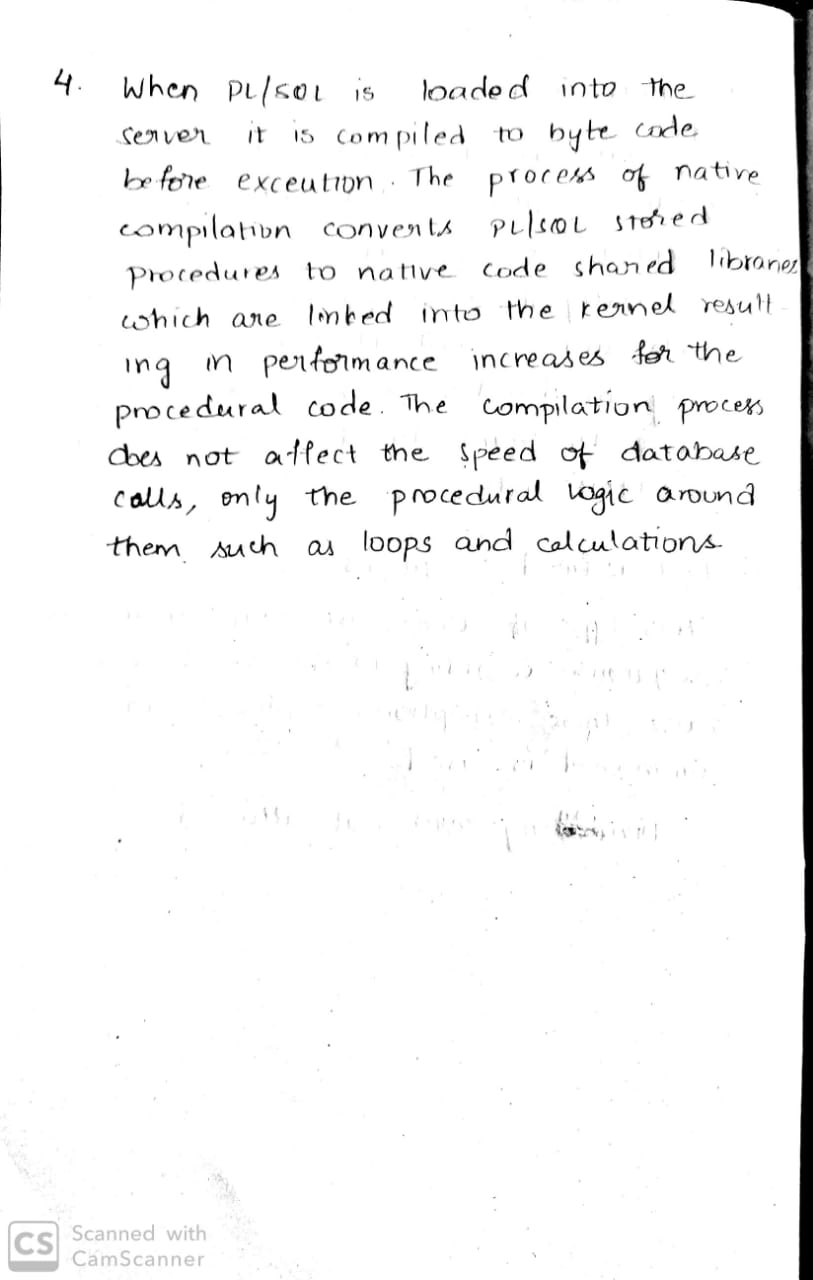




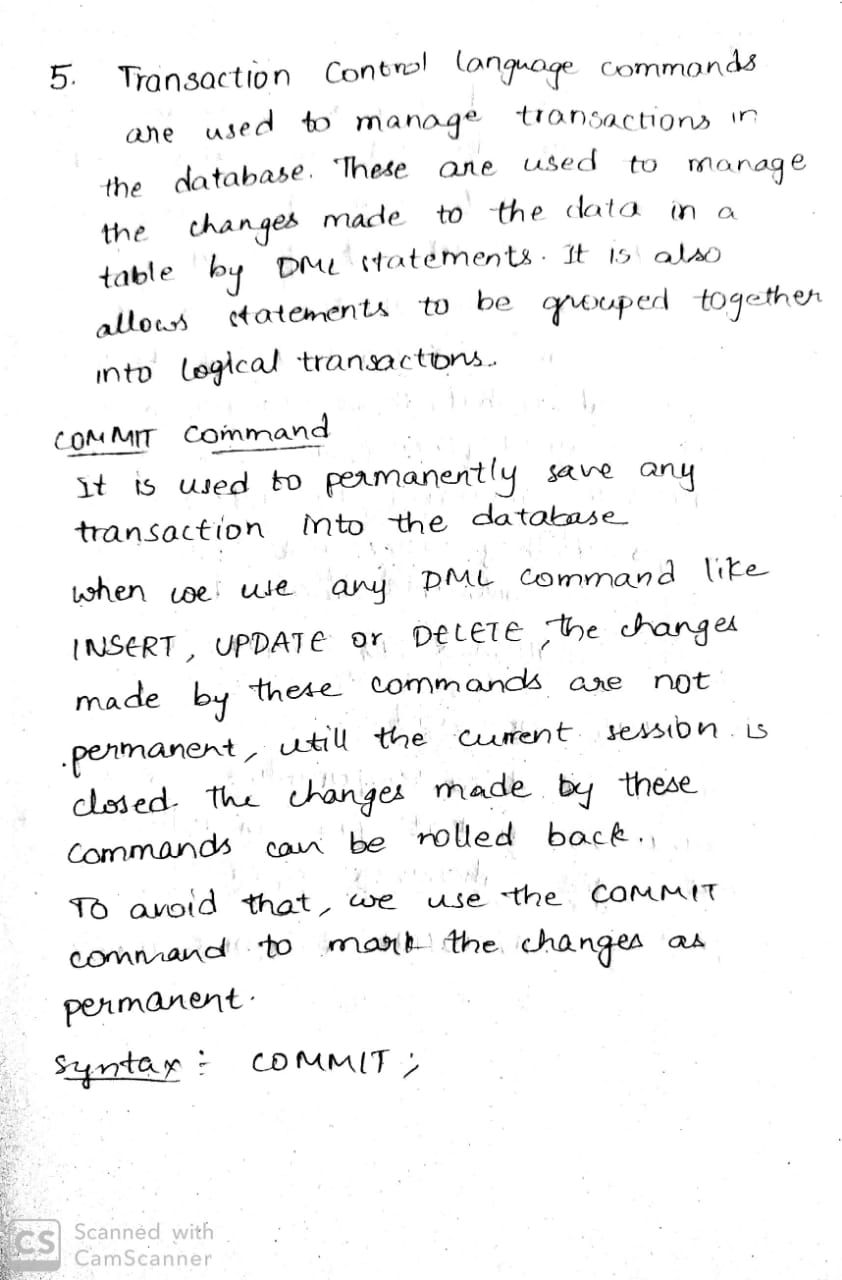
* 1. What are the rules to be applied to NULLs whilst doing comparisons?

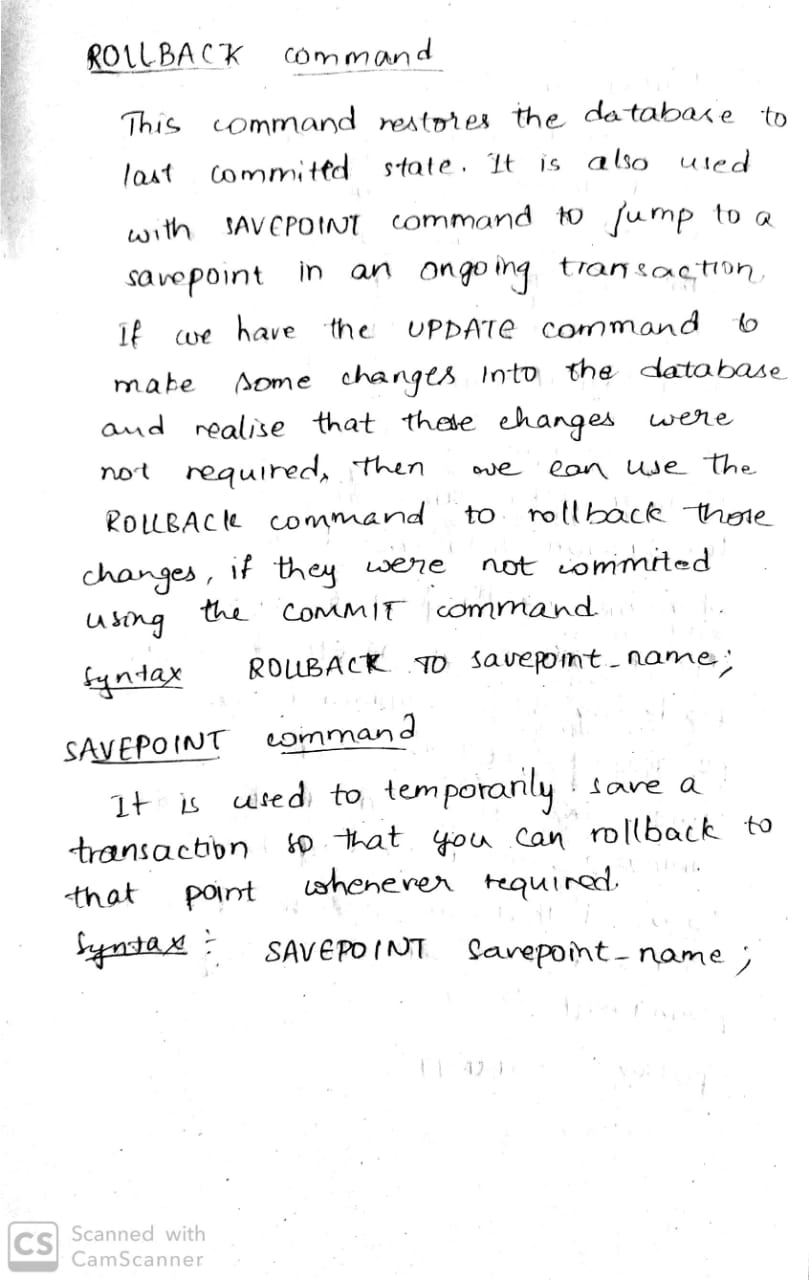


* 1. How is a process of PL SQL compiled?

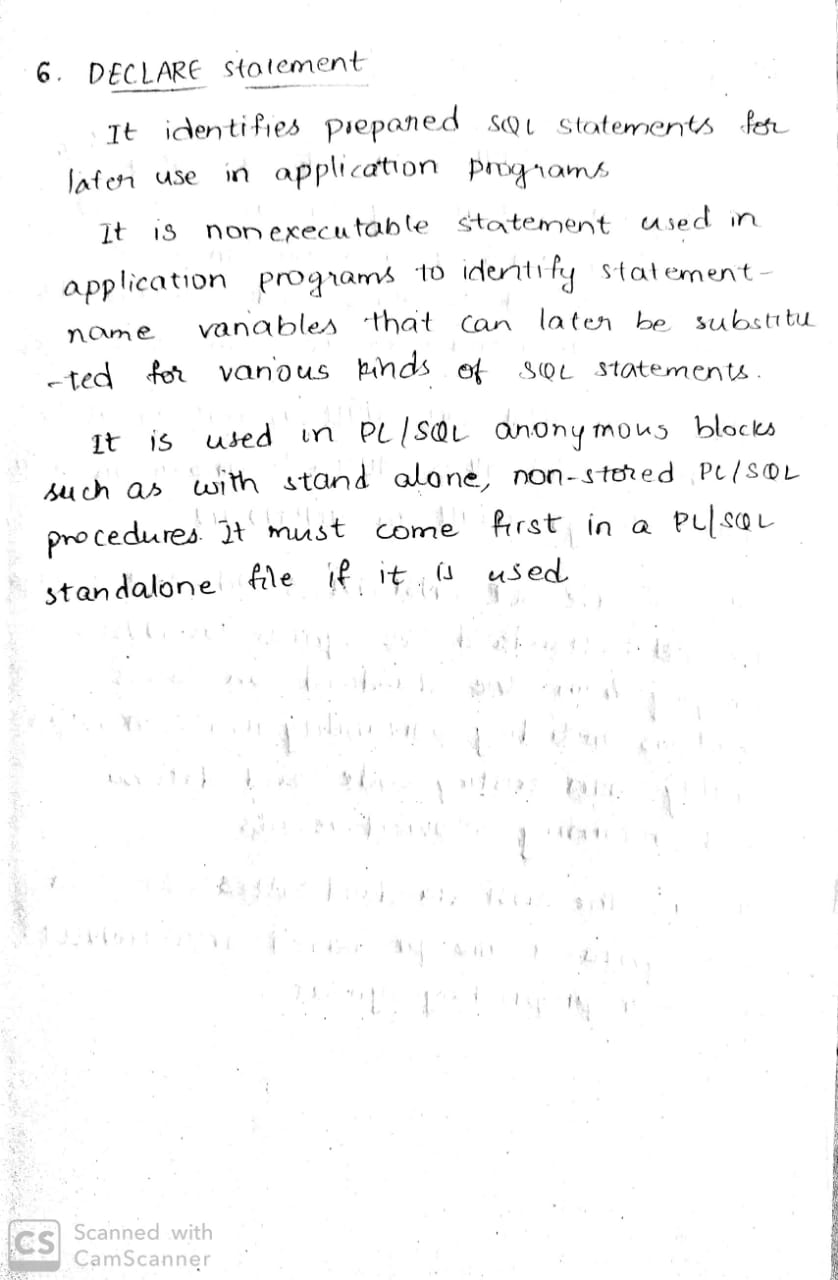


* 1. Explain Commit, Rollback and Savepoint.





* 1. When is a declare statement required?



**INLAB**

Implement PL/SQL Programs on Case Study 7 **(PROPERTY RENTAL INFORMATION SYSTEM)**

1. Create a cursor display the staff working in a particular branch

delimiter $$

create procedure staffdetails()

begin

declare s\_city varchar(50);

declare s\_state varchar(50);

declare s\_finished integer default 0;

declare c1 cursor for select city,state from staff;

declare continue handler for not found set s\_finished=1;

open c1;

staffdetails:loop

fetch c1 into s\_city,s\_state;

if s\_finished=1 then

leave staffdetails;

end if;

select concat(s\_city,",",s\_state);

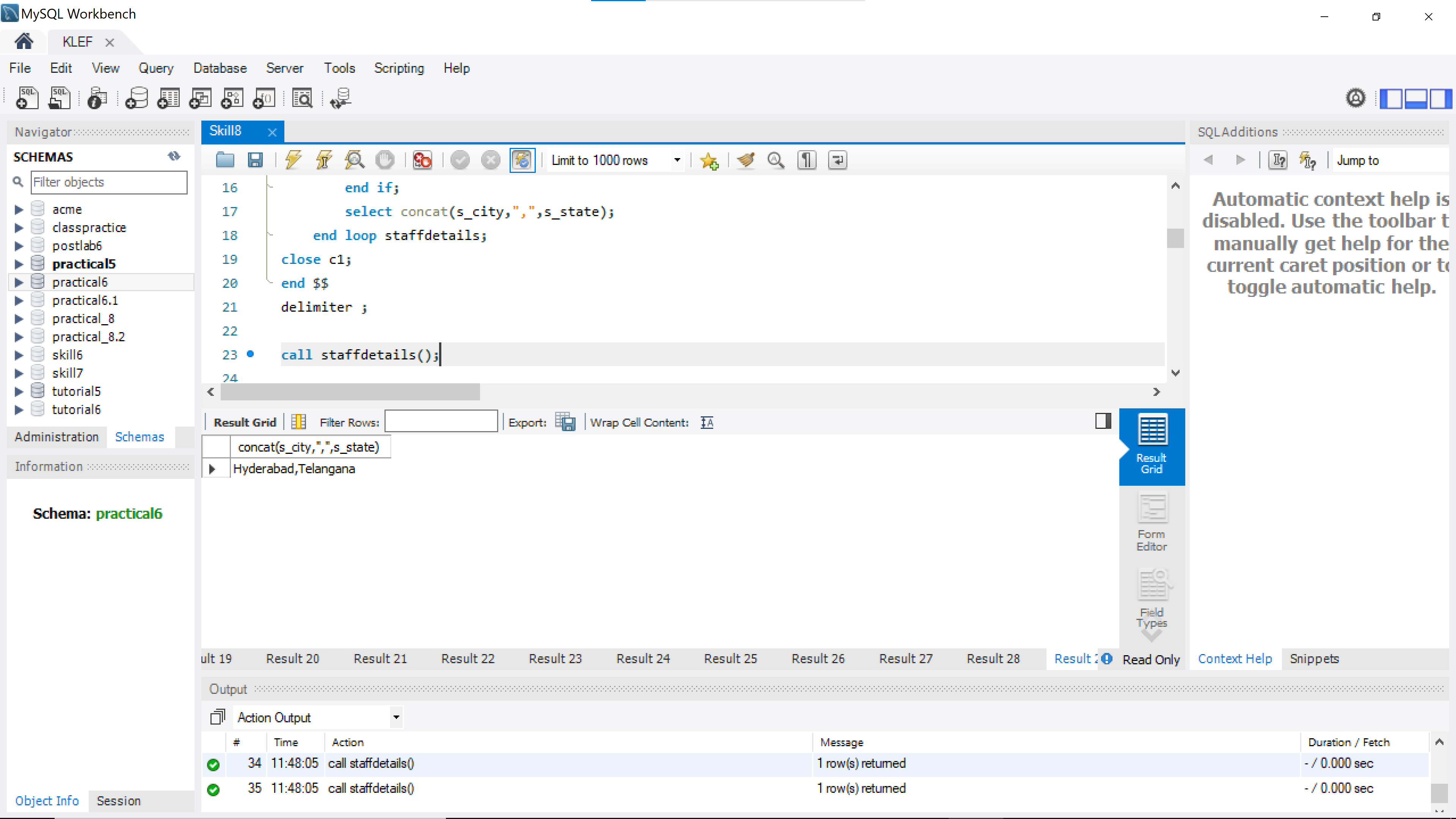
end loop staffdetails;

close c1;

end $$

delimiter ;

call staffdetails();



1. Create a procedure to display all the renter details

delimiter @

create procedure renterdetails()

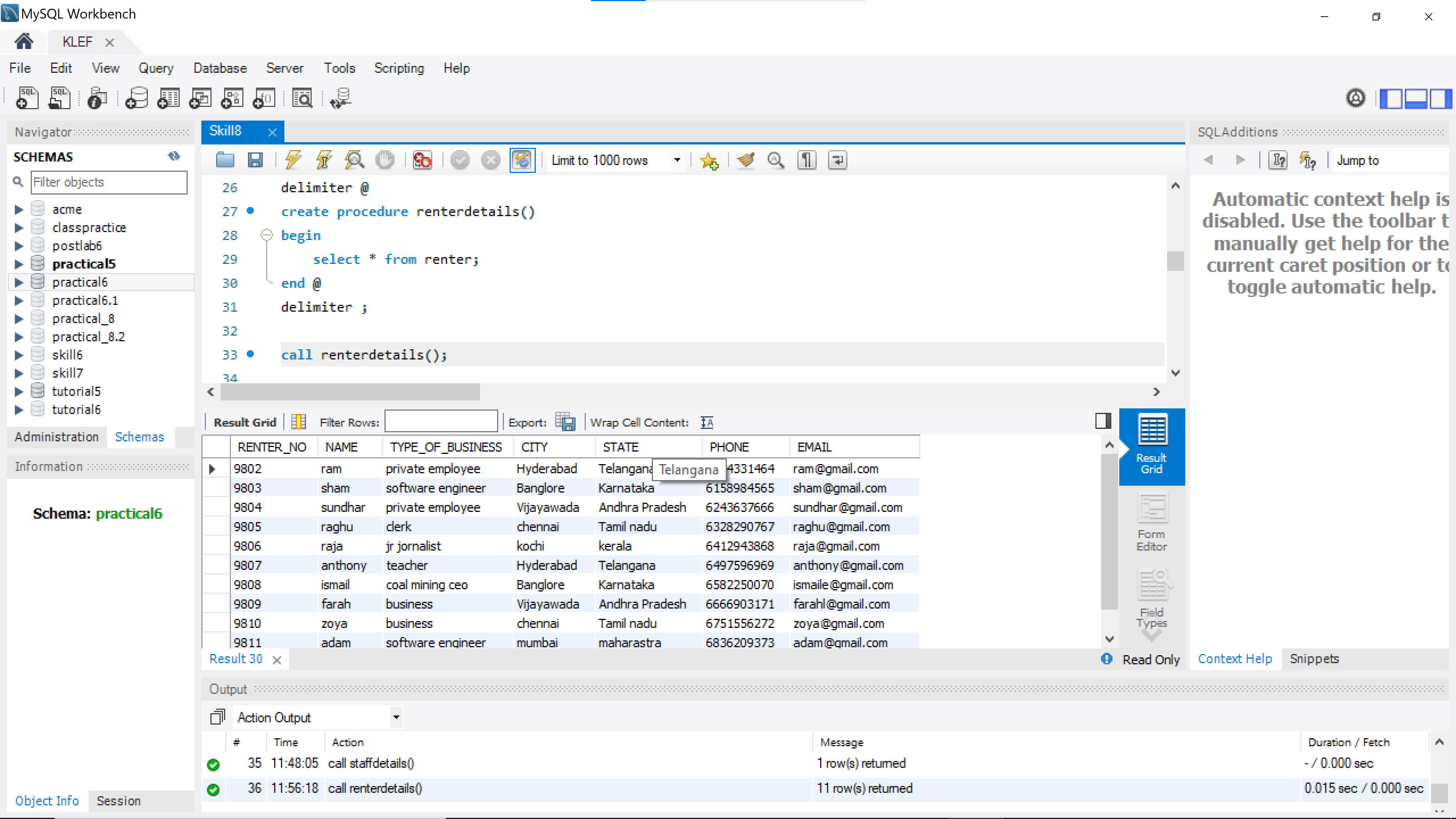
begin

select \* from renter;

end @

delimiter ;

call renterdetails();



1. Create a cursor to update the salary of an employee by 200 rupees

delimiter @@

create procedure salary\_update(a varchar(50))

begin

declare new\_salary int;

declare s\_finished integer default 0;

declare c cursor for select salary from staff where name=a for update;

declare continue handler for not found set s\_finished=1;

open c;

salary:loop

fetch c into new\_salary;

if s\_finished=1 then

leave salary;

end if;

update staff set salary=new\_salary+200 where name=a;

end loop salary;

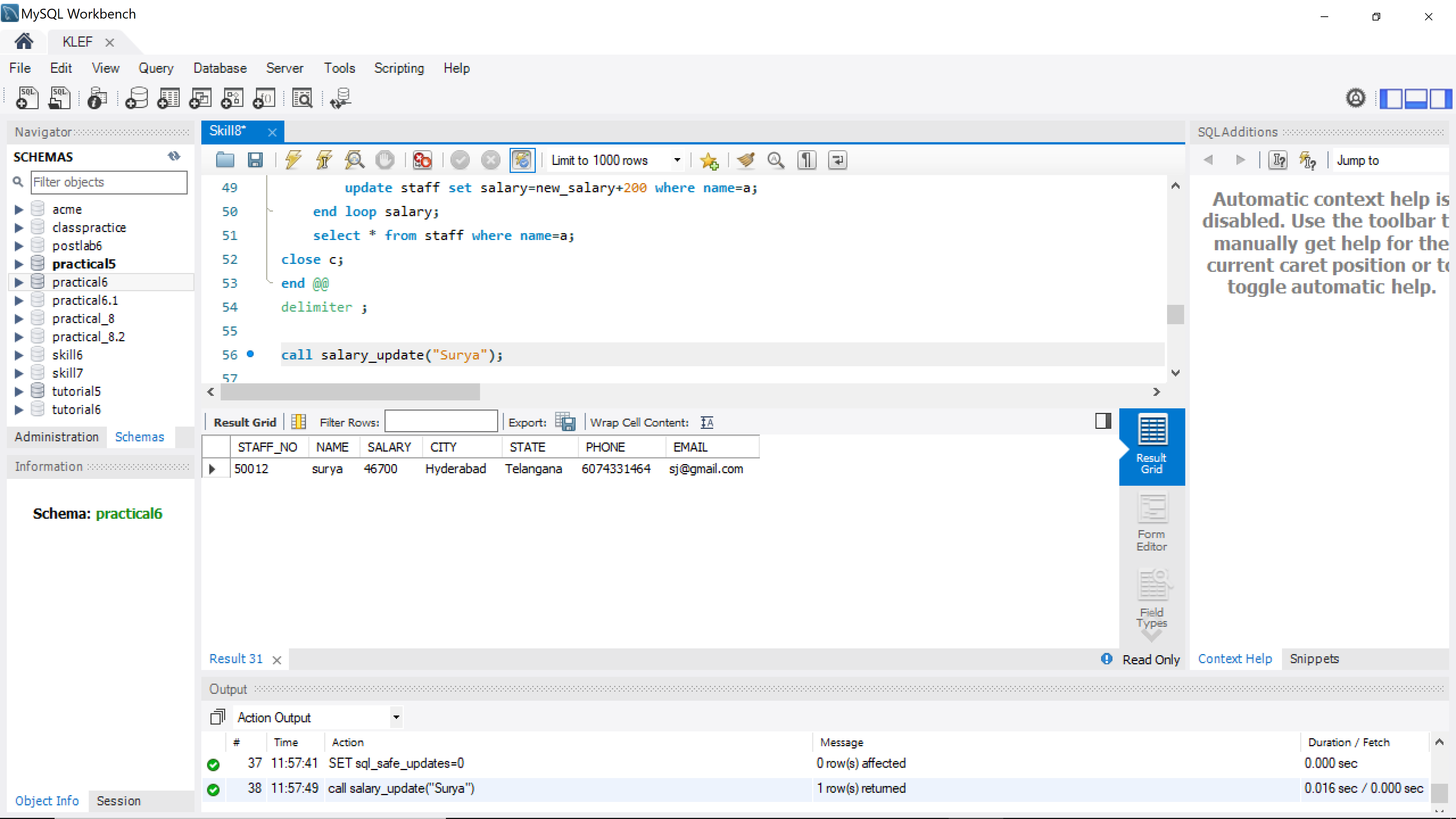
select \* from staff where name=a;

close c;

end @@

delimiter ;

call salary\_update("Surya");



1. Create a trigger to create a log file and inset data into that file when you update the staff details

create table Staff\_Log(before\_staff\_no int,updated\_staff\_no int,before\_name varchar(25), updated\_name varchar(25), before\_sal int, updated\_sal int, before\_city varchar(25), updated\_city varchar(25), before\_state varchar(25), updated\_state varchar(25), before\_phno bigint, updated\_phno bigint, before\_email varchar(25), updated\_email varchar(25), changed\_At TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP);

delimiter $$

create trigger staff\_update after update on STAFF

for each row

begin

insert into Staff\_Log values (OLD.STAFF\_NO, NEW.STAFF\_NO, OLD.NAME, NEW.NAME, OLD.SALARY, NEW.SALARY, OLD.CITY, NEW.CITY, OLD.STATE, NEW.STATE, OLD.PHONE, NEW.PHONE, OLD.EMAIL, NEW.EMAIL, current\_timestamp);

end $$

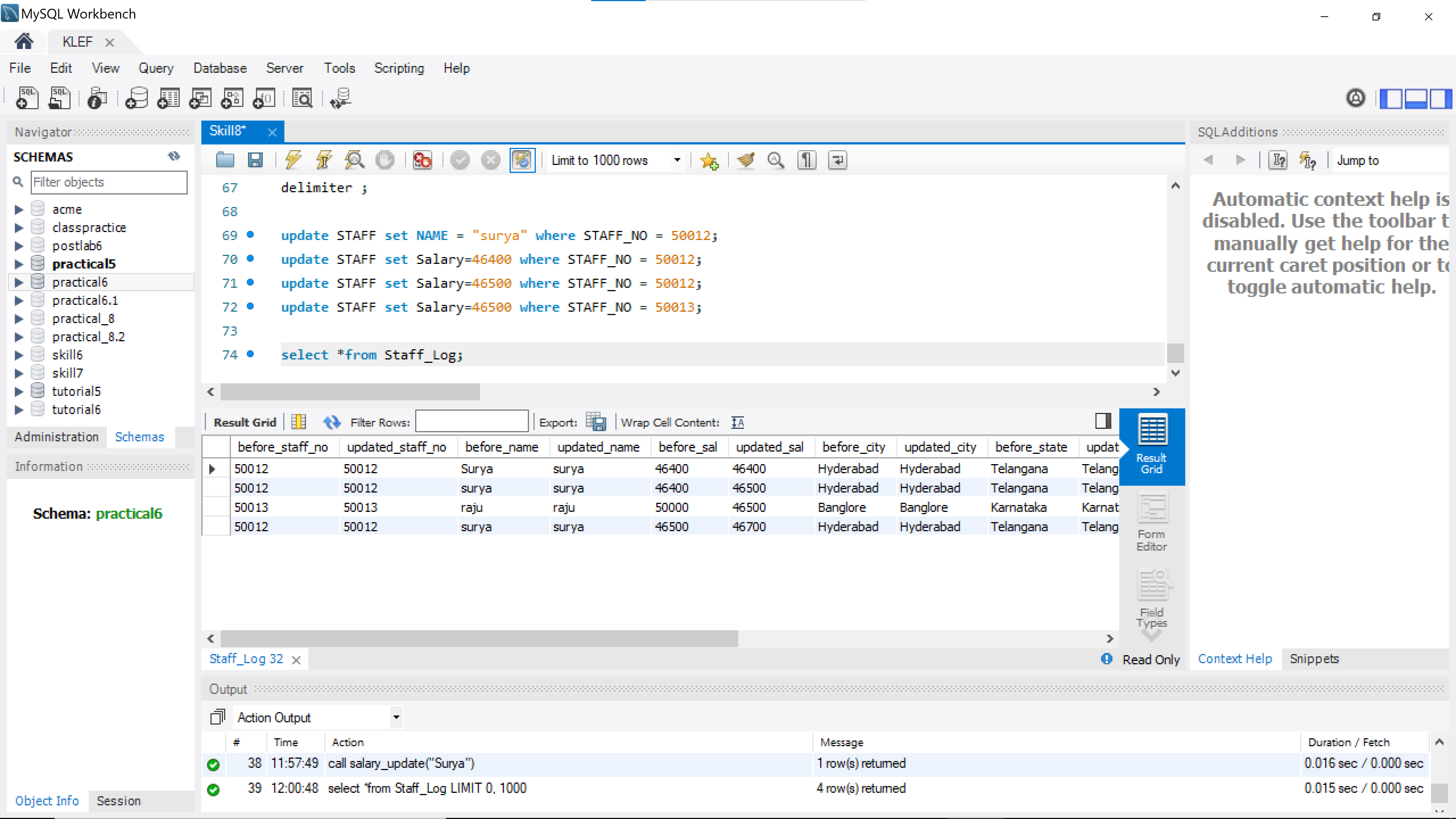
delimiter ;

update STAFF set NAME = "surya" where STAFF\_NO = 50012;

update STAFF set Salary=46500 where STAFF\_NO = 50012;

update STAFF set Salary=46500 where STAFF\_NO = 50013;

select \*from Staff\_Log;



**POSTLAB**

1. You are the head of a new division in company ABC. For the division, you have to make a new team. Following information is known to you about the company:
2. Employees are given ids starting from number 1. Every employee has a unique id.
3. Every team has some employees working in it. All employees working in a team have consecutive ids, i.e., if first employee in a team has id 1 and last employee had id 5, it means the team has all employees with ids 1,2,3,4,5.
4. Employees working in a team are comfortable among each other.
5. An employee may be working in multiple teams.
6. Any team will have atleast one employee.
7. Any employee will be working in atleast one team.

You have to make a team with employees such that all of them are comfortable with each other. You have to print the number of different teams you can make. The team must contain atleast one employee excluding you. Two teams are considered different if one team has atleast one employee which is not there in another team. You are given a table which consists of three fields are describe below:

1. TeamId: Id of the team.
2. StartEmpId: Id of the first employee of the team.
3. EndEmpId: Id of the last employee of the team. The table is ordered as below:

If team i comes before team j in table, then StartEmpIdi<StartEmpIdj and EndEmpIdi<EndEmpIdj

|  |  |
| --- | --- |
| Field | Type |
| TeamId | integer |
| StartEmpId | integer |
| EndEmpId | integer |

Input Format:

Table : Teams Output Format:

|  |  |
| --- | --- |
| Field | Type |
| Answer | int |

Sample Input:

Sample Teams Table: Sample Output:

|  |  |  |
| --- | --- | --- |
| TeamId | StartEmpId | EndEmpId |
| 1 | 1 | 3 |
| 3 | 2 | 4 |

|  |
| --- |
| Answer |
| 9 |

Explanation:

Employee 1 is comfortable with employees 2,3; 2 is comfortable with 1,3,4; 3 is comfortable with 1,2,4; 4 is comfortable with 2,3. The possible teams are: (1,1), (1,2), (1,3), (2,2), (2,3), (2,4), (3,3), (3,4), (4,4)

**Ans)**

**-- (SELECT @prev\_end:=0,@n:=0)**

**set @prev\_end:=0;**

**set @n:=0;**

**SELECT Cast(SUM(ttt.team\_combi) - SUM(ttt.remove\_cnt) as UNSIGNED INT) as Answer**

**FROM (**

**SELECT @n:=(t.EndEmpId - t.StartEmpId + 1),(@n \* (@n+1))/2 AS team\_combi,**

**CASE**

**WHEN StartEmpId <= @prev\_end AND @prev\_end <= EndEmpId**

**THEN ((@prev\_end - StartEmpId + 1)\*(@prev\_end - StartEmpId + 2))/2**

**WHEN @prev\_end <= StartEmpId THEN 0**

**ELSE 0**

**-- (@n \* (@n-1))/2 + @n**

**END AS remove\_cnt,**

**@prev\_end:=EndEmpId**

**FROM (SELECT \* FROM Teams) t) ttt;**

1. Recently, a Data Analyst Hiring contest was conducted at HackerEarth and our guy Fredo is assigned the task to provide a list of all shortlisted students. He is given three tables:
   1. Candidates : It consists of Name of candidate, his UserId and his Skill (in particular Skill rating). All candidates have different UserId and Skills.
   2. Problems: It consists of ProblemName and it's Score.
   3. Submissions: It consists of UserId and ProblemName. Any row of this table indicates

that the particular candidate has solved that problem and will be awarded the score attached to that problem. The candidates will get shortlisted on the basis of their total scores,i.e., candidate having more total score will be preferred over one having less total score. If two candidates have same total scores, the one having more skill rating will be preferred.

The cutoff score is 50, meaning that only those students having total score atleast 50 can be shortlisted. If there are more than 8 candidates clearing the cut off score , only top 8 would be selected. The output table should be ordered by total score. In case total score of two candidates is the same, it should be ordered by their skill rating. Total score of a candidate means the sum of the scores of all the problems solved by him. It is guaranteed that atleast one candidate will be shortlisted.

|  |  |
| --- | --- |
| Field | Type |
| UserId | text |
| ProblemName | text |

Input Format:

|  |  |
| --- | --- |
| Field | Type |
| ProblemName | text |
| Score | int |

Table : Candidates Table: Problems Table: Submissions

|  |  |
| --- | --- |
| Field | Type |
| Name | text |
| UserId | Text |
| Skills | Int |

**Ans)**

**select name ,s.userid ,sum(score) as ss from Candidates c join Submissions s on c.userid=s.userid join Problems p on p.problemname=s.problemname group by s.userid having sum(score)>=50 order by ss desc,skills desc limit 8**

1. A database, normalized as per 2NF rules, has been split into 10 tables. Each of the tables has exactly two columns: one key attribute and one non-key attribute. What is the minimum number of tables required to express this database in 3NF form? Enter the integer in the text box below. Do not leave any leading or trailing spaces.

**Ans)**

**10**