PRELAB

1) Differentiate between % ROWTYPE and TYPE RECORD.

DBMS Skill-8 190031249
P.Mohith

1. %TYPE is used to get type of a variable or a table column

Ex: v_name customers.name%TYPE;

%ROWTYPE is used to obtain the type of line of a cursor or table.

Ex:
```
DECLARE
CURSOR c is SELECT id, name, type, email from
    customers WHERE type = 'CORPORATE';
cust_rec c%ROWTYPE;
BEGIN
OPEN c;
LOOP
    FETCH c INTO cust_rec;
    EXIT WHEN c%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(cust_rec.name);
END LOOP;
CLOSE c;
END;
```

2) What are the two types of exceptions?

2. The two types of exceptions defined in PL/SQL

1. user defined exceptions
2. System defined exceptions.

Syntax to write an exception

WHEN exception THEN statement;

1. System defined exceptions

These exceptions are predefined in PL/SQL which get raised WHEN certain database rule is violated.

This is further divided into two categories:-

(i) Named system exceptions
(ii) UnNamed System exceptions

(i) Named system exceptions

They have a predefined name by the system like ACCESS_INTO_NULL, LOGIN_DENIED etc....

1. NO DATA FOUND
2. TOO_MANY_ROWS
3. VALUE_ERROR
4. ZERO_DIVIDE

(ii) System UnNamed exceptions :-

Oracle doesn't provide name for some system exceptions called unnamed system exceptions These exceptions dont occur frequently. These exceptions have two parts code and an associated message. The way to handle to these exceptions is to assign name to them using pragma EXCEPTION_INIT

2. user defined exceptions

This type of users can create their own exceptions according to the need and to raise these exceptions explicitly raise command is used.

Ex: Division by zero not allowed

3) What are the rules to be applied to NULLs whilst doing comparisons?

3. Four Rules for NULL's

Rule-1 :- Use NULL's to indicate unknown / missing information only. Do not use NULL's in place of zeros, zero-length strings or other "known" blank values. Update your NULL's with proper information as soon as possible

Rule-2 :- In ANSI, SQL NULL is not equal to anything, even other NULL's ! Comparison with NULL always result in UNKNOWN

Rule-3 :- use SET ANSI_NULL'S ON and always use ANSI standard SQL syntax for NULL's. straying from the standard can cause problems including portability issues, incompa -tibility with existing code and databases and returning incorrect results.

Rule-4 :- The ANSI standard COALESCE () and CASE syntaxes are prefferred over ISNULL() or other proprietary syntax.

4) How is a process of PL SQL compiled?

4. When PL/SQL is loaded into the server it is compiled to byte code before execution. The process of native compilation converts PL/SQL stored procedures to native code shared libraries which are linked into the kernel resulting in performance increases for the procedural code. The compilation process does not affect the speed of database calls, only the procedural logic around them such as loops and calculations.

5) Explain Commit, Rollback and Savepoint.

5. Transaction Control language commands are used to manage transactions in the database. These are used to manage the changes made to the data in a table by DML statements. It is also allows statements to be grouped together into logical transactions.

COMMIT command

It is used to permanently save any transaction into the database

When we use any DML command like INSERT, UPDATE or DELETE, the changes made by these commands are not permanent, until the current session is closed the changes made by these commands can be rolled back.

To avoid that, we use the COMMIT command to mark the changes as permanent.

Syntax: COMMIT;

ROLLBACK command

This command restores the database to last commited state. It is also used with SAVEPOINT command to jump to a savepoint in an ongoing transaction.

If we have the UPDATE command to make some changes into the database and realise that these changes were not required, then we can use the ROLLBACK command to rollback those changes, if they were not commited using the COMMIT command.

Syntax ROLLBACK TO savepoint_name;

SAVEPOINT command

It is used to temporarily save a transaction so that you can rollback to that point whenever required.

Syntax : SAVEPOINT savepoint_name;

6) When is a declare statement required?

6. DECLARE statement

It identifies prepared SQL statements for later use in application programs

It is nonexecutable statement used in application programs to identify statement-name variables that can later be substitu-ted for various kinds of SQL statements.

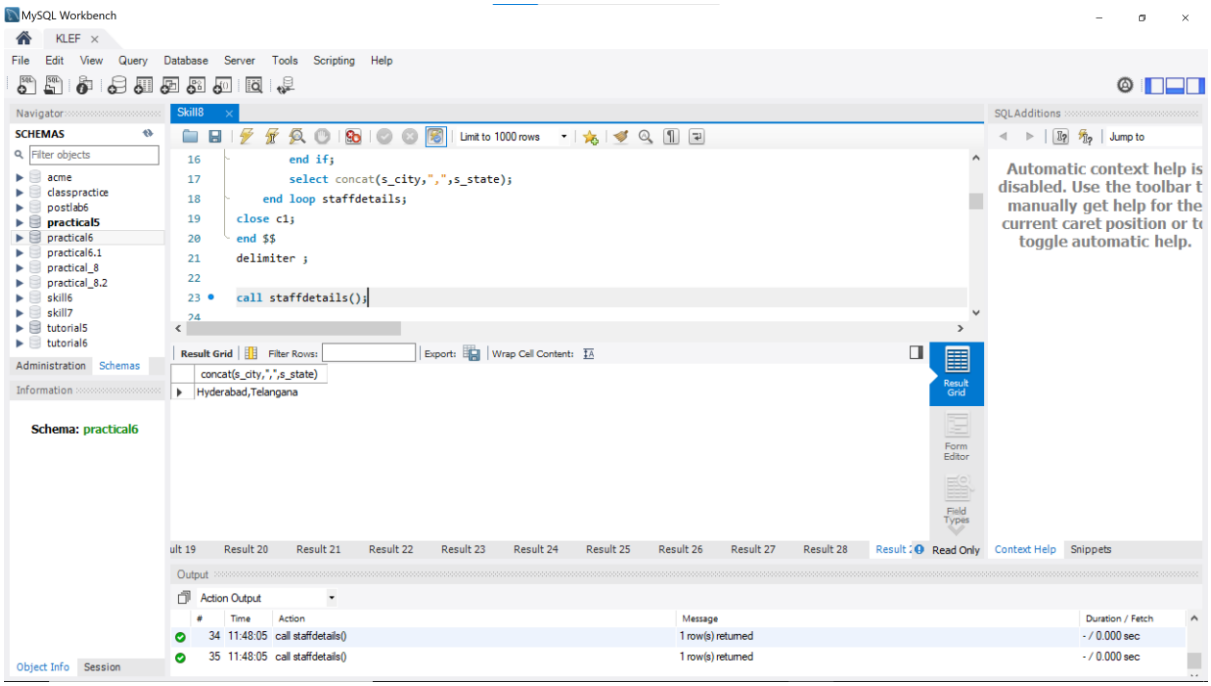
It is used in PL/SQL anonymous blocks such as with stand alone, non-stored PL/SQL procedures. It must come first in a PL/SQL standalone file if it is used

INLAB

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

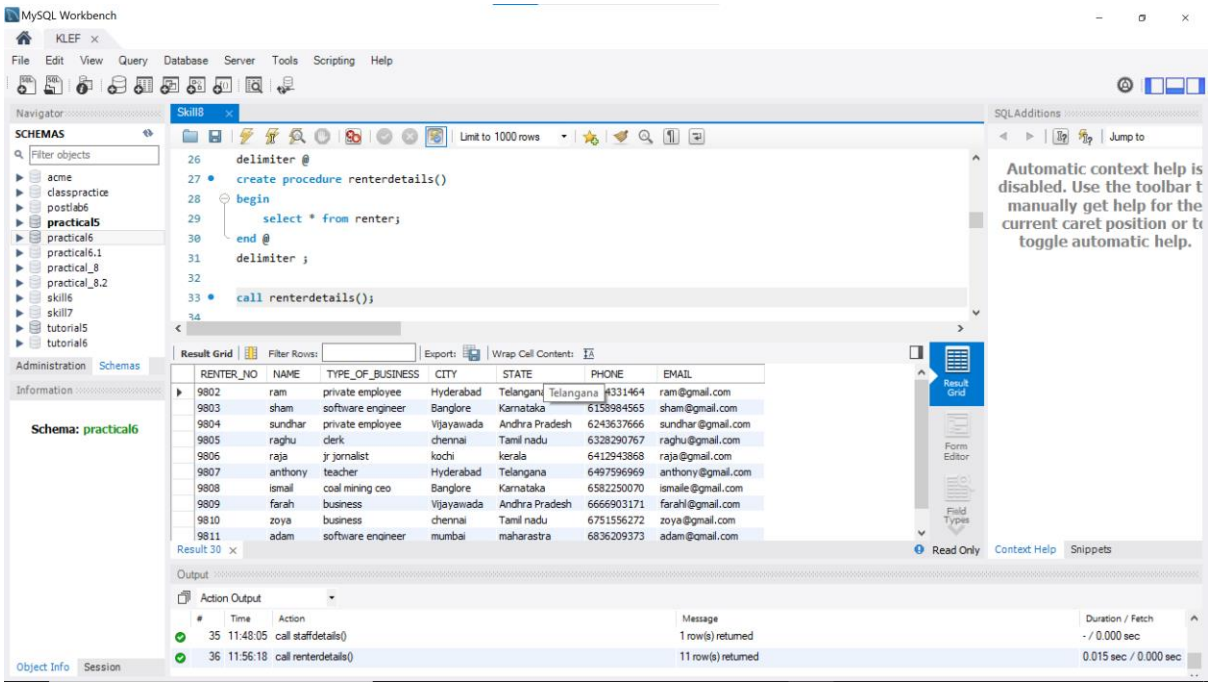
1) Create a cursor display the staff working in a particular branch

```
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();
```



2) Create a procedure to display all the renter details

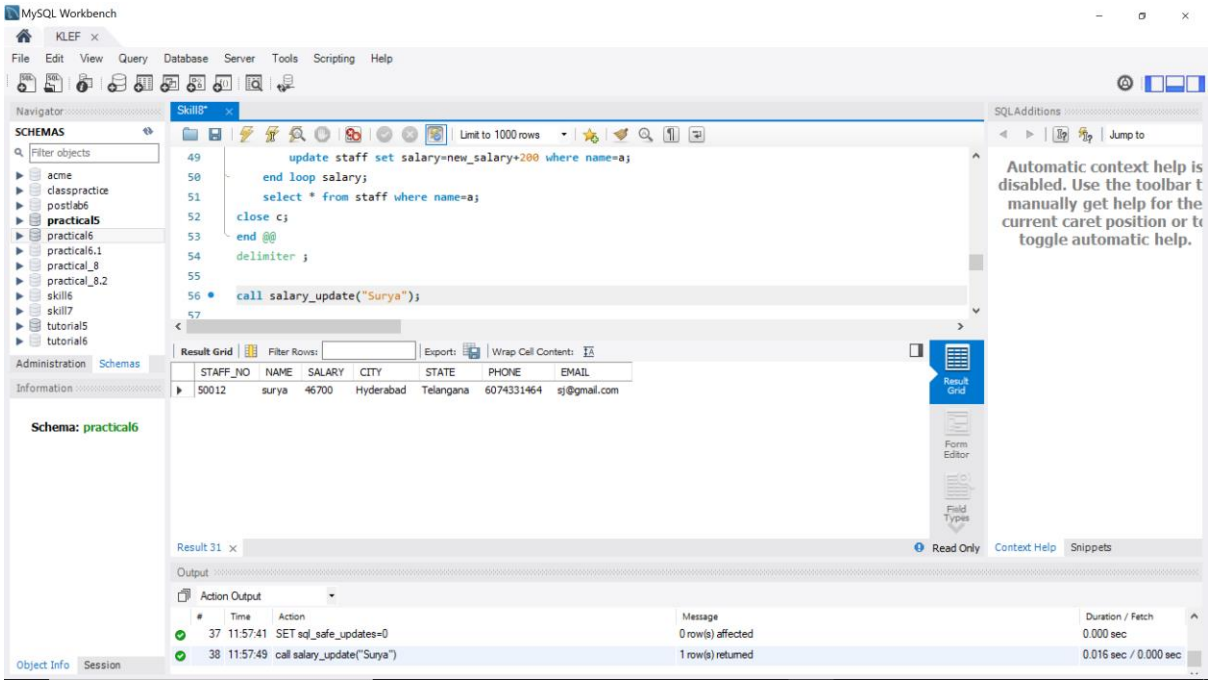
```
delimiter @
create procedure renterdetails()
begin
		select * from renter;
end @
delimiter ;

call renterdetails();
```



3) Create a cursor to update the salary of an employee by 200 rupees

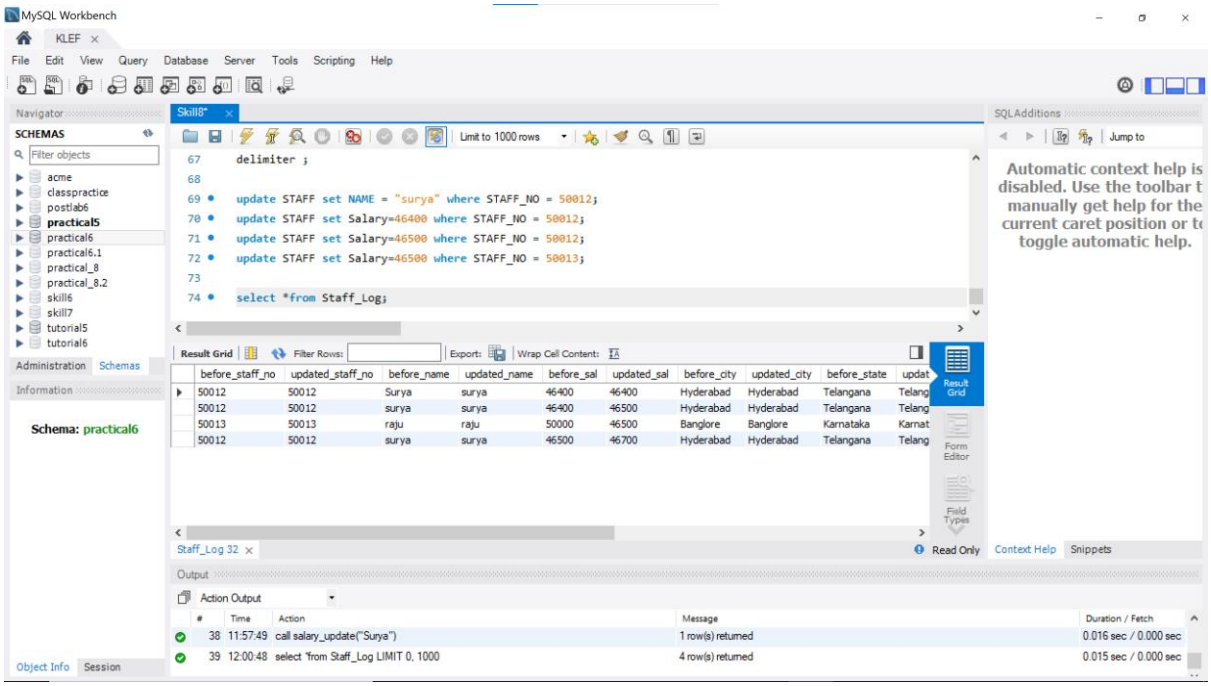
```
delimiter @@
create procedure salary_update(a varchar(50))
begin
		declare new_salary int;
  declare s_finished integer default 0;
		declare c cursor for select salary from staff where name=a for update;
		declare continue handler for not found set s_finished=1;
open c;
		salary:loop
		  fetch c 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");
```



4) 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;
```


MySQL Workbench
KLEF
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
acme
classpractice
postlab6
practical5
practical6
practical6.1
practical_8
practical_8.2
skill6
skill7
tutorial5
tutorial6
Administration Schemas
Information
Schema: practical6
Object Info Session
Skill8*
Limit to 1000 rows
67 delimiter ;
68
69 update STAFF set NAME = "surya" where STAFF_NO = 50012;
70 update STAFF set Salary=46400 where STAFF_NO = 50012;
71 update STAFF set Salary=46500 where STAFF_NO = 50012;
72 update STAFF set Salary=46500 where STAFF_NO = 50013;
73
74 select *from Staff_Log;
Result Grid Filter Rows: Export: Wrap Cell Content:
before_staff_no updated_staff_no before_name updated_name before_sal updated_sal before_city updated_city before_state updat
50012 50012 Surya surya 46400 46400 Hyderabad Hyderabad Telangana Telang
50012 50012 surya surya 46400 46500 Hyderabad Hyderabad Telangana Telang
50013 50013 raju raju 50000 46500 Banglore Banglore Karnataka Karnat
50012 50012 surya surya 46500 46700 Hyderabad Hyderabad Telangana Telang
Result Grid
Form Editor
Field Types
Staff_Log 32
Read Only
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
38 11:57:49 call salary_update("Surya") 1 row(s) returned 0.016 sec / 0.000 sec
39 12:00:48 select *from Staff_Log LIMIT 0, 1000 4 row(s) returned 0.015 sec / 0.000 sec

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:
   i. Employees are given ids starting from number 1. Every employee has a unique id.
   ii. 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.
   iii. Employees working in a team are comfortable among each other.
   iv. An employee may be working in multiple teams.
   v. Any team will have atleast one employee.
   vi. 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

Input Format:

Table : Teams

Field	Type
TeamId	integer
StartEmpId	integer
EndEmpId	integer

Output Format:

Field	Type
Answer	int

Sample Input:

Sample Teams Table:

TeamId	StartEmpId	EndEmpId
1	1	3
3	2	4

Sample Output:

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;
```

2. 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:

i) Candidates : It consists of Name of candidate, his UserId and his Skill (in particular Skill rating). All candidates have different UserId and Skills.

ii) Problems: It consists of ProblemName and it's Score.

iii) 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.

Input Format:

Table : Candidates

Field	Type
Name	text
UserId	Text
Skills	Int

Table: Problems

Field	Type
ProblemName	text
Score	int

Table: Submissions

Field	Type
UserId	text
ProblemName	text

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
```

3. 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