

**PL/SQL Programs on Case Study 2 & 5
(EMERGENCY ROOM INFORMATION SYSTEM) & (TOUR OPERATING SYSTEM)
PRE-LAB**

```
1) Declare
   fvar number := null; svar number := 5
Begin
   goto << fproc>>
   if fvar is null then
   << fproc>>
   svar := svar + 5
   end if;
End;
```

What will be the value of svar after the execution ?

Ans) Ouput : Syntax Error.

2) What is a stored procedure?

Ans) A stored procedure is a prepared sql code that you can save, so the code can be reused over and over again.

3) What are the different datatypes supported in PL/SQL

Ans) pl/sql provides many pre-defined data-types like integer, floating point , charcater , Boolean, date , collection,refrence and large object (lob) types.

4) What is the Result of the following 'VIK' || 'NULL' || 'RAM' ?

Ans) VIKRAM

5) A database is an extensive collection of records. In what form are they stored?

Ans) Database is a collection of data and records. They are stored in form of simple tables. Tables are related if they contain common fields.

6) In the index allocation scheme of blocks to a file, the maximum possible size of the life depends on _____

Ans) the size of the blocks, the number of blocks used for the index and size of address of blocks.

7) How many Clustered indexes can be created on table and why?

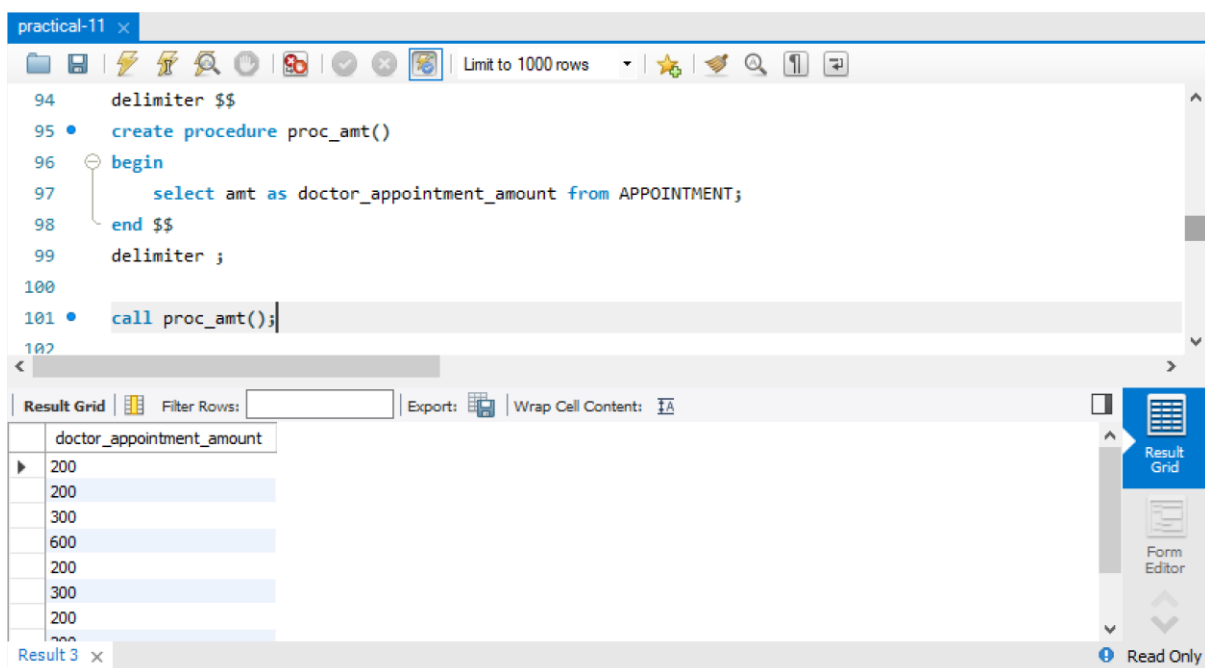
Ans) There can only be one clustered index per table , because the data row themselves can be stored in only one order. The only time the data rows in a table are stored in sorted order is when the table contains a clustered index.

INLAB

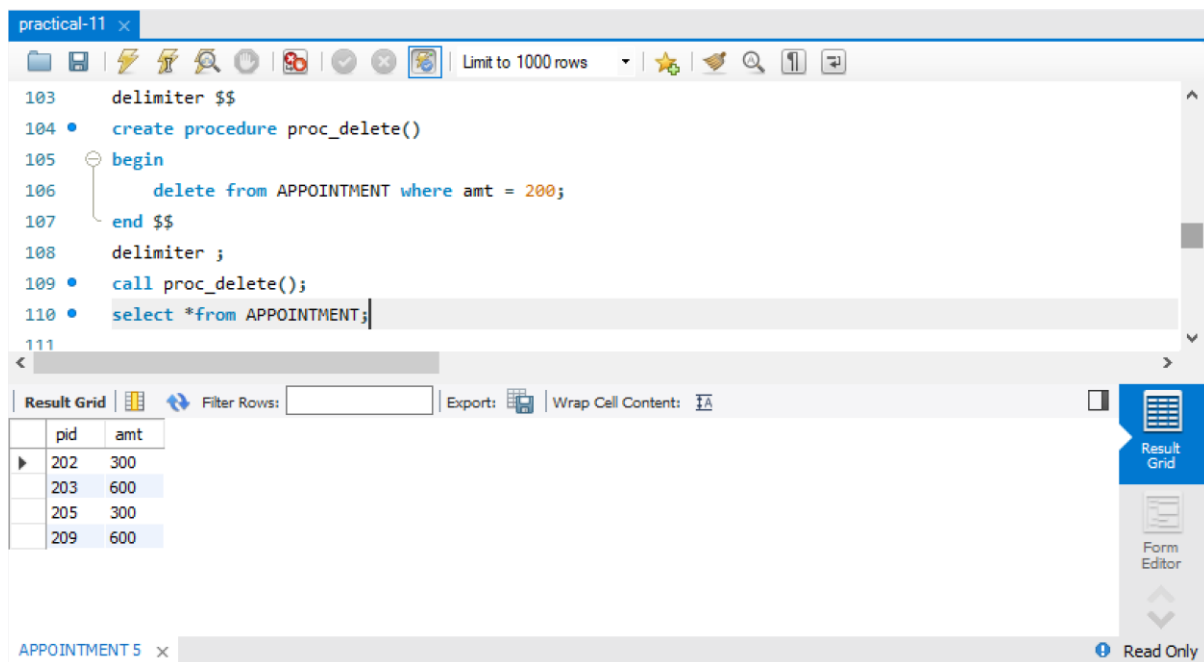
1) Write PL/SQL program to display doctor appointment fee amount value.

```
delimiter $$
create procedure proc_amt()
begin
    select amt as doctor_appointment_amount from APPOINTMENT;
end $$
delimiter ;

call proc_amt();
```

**2) Write PL/SQL program to delete appointment amount value 200**

```
delimiter $$
create procedure proc_delete()
begin
    delete from APPOINTMENT where amt = 200;
end $$
delimiter ;
call proc_delete();
select *from APPOINTMENT;
```



The screenshot shows an SQL IDE window titled 'practical-11'. The script editor contains the following PL/SQL code:

```
103 delimiter $$
104 • create procedure proc_delete()
105 • begin
106 •     delete from APPOINTMENT where amt = 200;
107 • end $$
108 delimiter ;
109 • call proc_delete();
110 • select *from APPOINTMENT;
111
```

Below the script editor, the 'Result Grid' is displayed, showing the output of the 'select *from APPOINTMENT;' statement. The grid has two columns: 'pid' and 'amt'. The data rows are:

pid	amt
202	300
203	600
205	300
209	600

The IDE interface includes a toolbar with various icons, a 'Limit to 1000 rows' dropdown, and a 'Read Only' status indicator at the bottom right.

3) Write a PL/SQL program using functions to display the address details from where the number of patients are more than 3.

```
delimiter $@
create function Q12() returns varchar(100)
deterministic
begin
    declare city varchar(45);
    select address into city from Patient group by address order by count(*) desc limit 1;
    return city;
end $@
delimiter ;

select Q12();
```

The screenshot shows a SQL Developer window titled 'practical-11'. The editor contains the following PL/SQL code:

```

128 declare city varchar(45);
129 select address into city from Patient group by address order by count(*) desc limit 1;
130 return city;
131 end $$
132 delimiter ;
133
134 • select Q12();

```

The 'Result Grid' at the bottom shows the output of the query:

Q12()
Hyderabad

The interface includes a toolbar with icons for file operations, a 'Limit to 1000 rows' dropdown, and a 'Read Only' status indicator at the bottom right.

4) Write an PL/SQL program using triggers to raise an exception on invalid patient ID

delimiter \$\$

create trigger trig_patient before insert on Patient

for each row

begin

if new.pid not in (select pid from Patient) then

SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid Patient ID';

end if;

end \$\$

delimiter ;

insert into Patient values(210,'Naveen','Kumar','jghfr@gmail.com',2099135327,'Hyderabad',STR_TO_DATE('22-04-2020','%d-%m-%Y'));

The screenshot shows a SQL Developer window titled 'practical-11'. The editor contains the following PL/SQL code:

```

113 • create trigger trig_patient before insert on Patient
114 for each row
115 begin
116 if new.pid not in (select pid from Patient) then
117 SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid Patient ID';
118 end if;
119 end $$
120 delimiter ;
121
122 • insert into Patient values(210,'Naveen','Kumar','jghfr@gmail.com',2099135327,'Hyderabad',STR_TO_DATE('22-04-2020','%d-%m-%Y'));
123

```

The 'Output' window at the bottom shows the execution results:

#	Time	Action	Message	Duration / Fetch
79	23:30:41	insert into Patient values(210,'Naveen','Kumar','jghfr@gmail.com',2099135327,'Hy...	Error Code: 1644. Invalid Patient ID	0.016 sec

The interface includes a toolbar with icons for file operations, a 'Limit to 1000 rows' dropdown, and a 'Context Help' button on the right.

PL/SQL programs on TOUR OPERATING SYSTEM

1) Create a procedure to display the tourist details who are visiting the same place

```

1  delimiter @@
2  • create procedure Q1 ()
3  • begin
4  •   select customer.c_name,tour.tr_dest from customer inner join tour on customer.c_addr = tour.tr_start;
5  • end @@
6  • delimiter ;
7
8  • call Q1();
9

```

Result Grid

c_name	tr_dest
Raju	vij
Hari	hyd
Kiran	tnl
Giri	gnt
jaya	Mumbai

Result 4 x Read Only

2) Create a cursor to display the details of the customers/tourists

```

15 • create procedure Q2()
16 • begin
17 •   declare s_id int;
18 •   declare s_name varchar(45);
19 •   declare s_addr varchar(45);
20 •   declare s_mobile mediumtext;
21 •   declare s_finished integer default 0;
22 •   declare c1 cursor for select * from customer;
23 •   declare continue handler for not found set s_finished=1;
24 •   open c1;
25 •   customerdetails:loop
26 •     fetch c1 into s_id,s_name,s_addr,s_mobile;
27 •     if s_finished=1 then
28 •       leave customerdetails;
29 •     end if;
30 •     select concat(s_id,"",s_name,"",s_addr,"",s_mobile);
31 •   end loop customerdetails;
32 •   close c1;
33 • end @@
34 • delimiter ;
35
36 • call Q2();

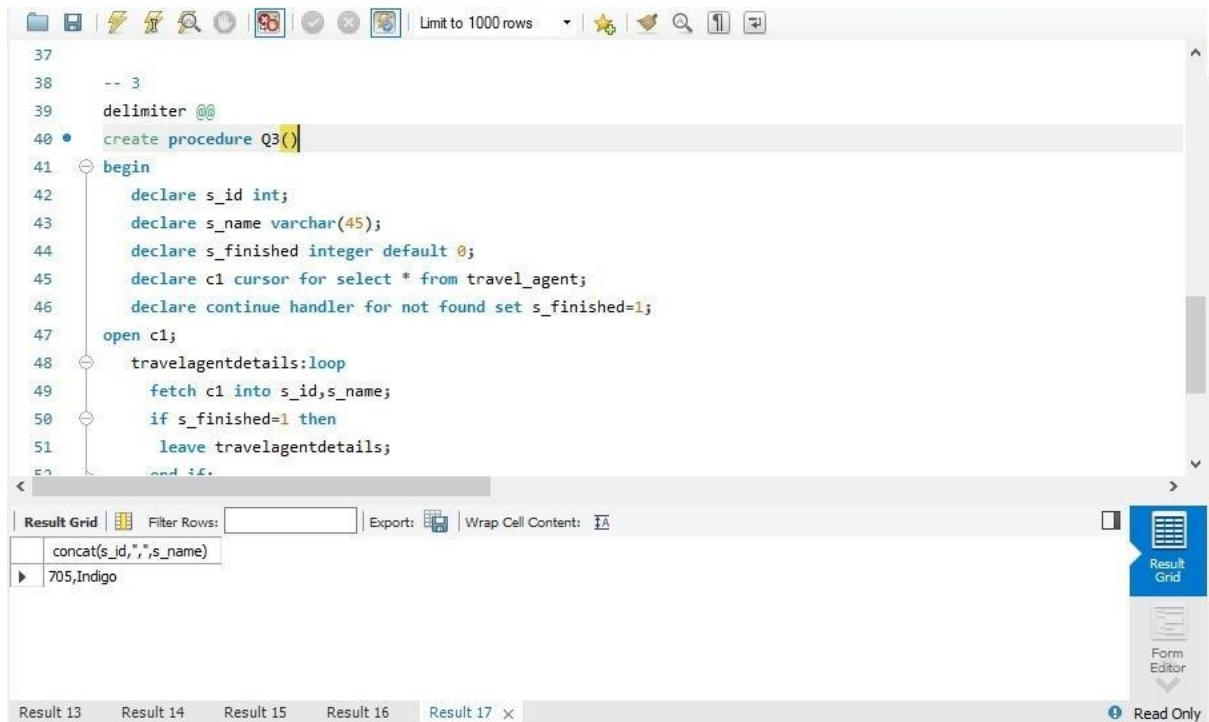
```

Result Grid

concat(s_id,"",s_name,"",s_addr,"",s_mobile)
307,Kiran,Guntur,7322938936

Result 5 Result 6 Result 7 Result 8 Result 9 Result 10 Result 11 Result 12 x Read Only

3) Create a cursor to display the details of the travel agents where the tourists booked their tours



The screenshot displays a database IDE interface. The main editor window contains a PL/SQL procedure named Q3, which is designed to iterate through travel agents and update a 's_finished' flag. The procedure includes variable declarations for agent ID and name, a cursor for the 'travel_agent' table, and a loop structure with a 'fetch' statement and conditional logic. The 'Result Grid' at the bottom shows the output of the procedure, displaying a single row with the concatenated string '705,Indigo'.

```
37
38 -- 3
39 delimiter @@
40 • create procedure Q3()
41 begin
42     declare s_id int;
43     declare s_name varchar(45);
44     declare s_finished integer default 0;
45     declare c1 cursor for select * from travel_agent;
46     declare continue handler for not found set s_finished=1;
47     open c1;
48     travelagentdetails:loop
49         fetch c1 into s_id,s_name;
50         if s_finished=1 then
51             leave travelagentdetails;
52         end if;
```

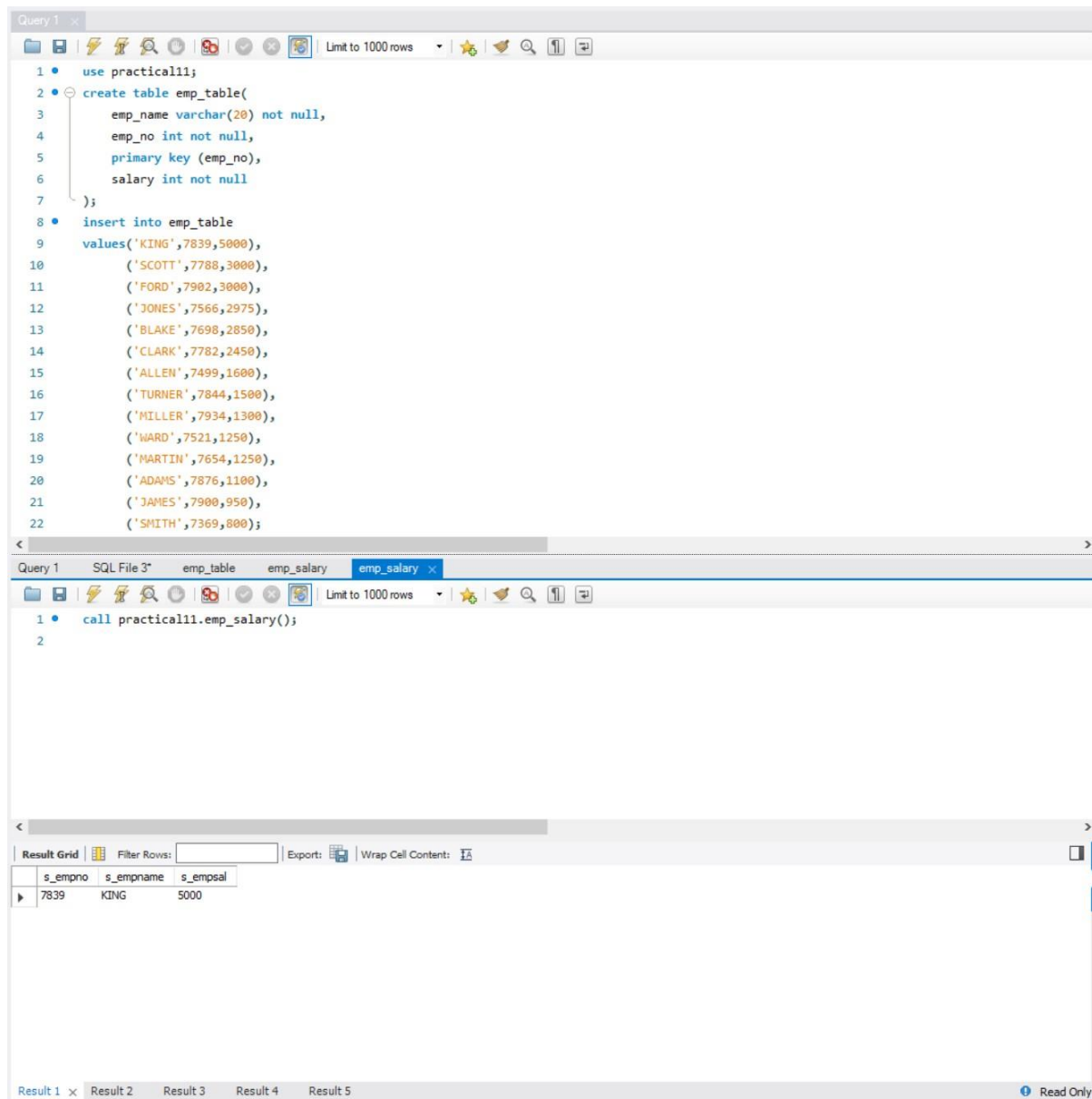
Result Grid

concat(s_id,',',s_name)
705,Indigo

Result 13 Result 14 Result 15 Result 16 Result 17 x Read Only

POSTLAB

- 1) Write a PL/SQL PROGRAM to select the five highest paid employees from the emp table using CURSORS.



The screenshot displays the SQL Developer interface. The top pane shows the execution of SQL statements to create a table and insert data. The bottom pane shows the execution of a call to a procedure, and the results are displayed in a grid below.

```
1 • use practical11;
2 • create table emp_table(
3     emp_name varchar(20) not null,
4     emp_no int not null,
5     primary key (emp_no),
6     salary int not null
7 );
8 • insert into emp_table
9     values('KING',7839,5000),
10         ('SCOTT',7788,3000),
11         ('FORD',7902,3000),
12         ('JONES',7566,2975),
13         ('BLAKE',7698,2850),
14         ('CLARK',7782,2450),
15         ('ALLEN',7499,1600),
16         ('TURNER',7844,1500),
17         ('MILLER',7934,1300),
18         ('WARD',7521,1250),
19         ('MARTIN',7654,1250),
20         ('ADAMS',7876,1100),
21         ('JAMES',7900,950),
22         ('SMITH',7369,800);
```

```
1 • call practical11.emp_salary();
2
```

s_empno	s_empname	s_empsal
7839	KING	5000

- 2) Write a PL/SQL Program to check for an Armstrong Number

declare

 n number:=1634;

 s number:=0;

 r number;

 len number;

 m number;

begin

```
m := n;

len := length(to_char(n));

while n>0
loop
    r := mod(n , 10);
    s := s + power(r , len);
    n := trunc(n / 10);
end loop;

if m = s
then
    dbms_output.put_line('yes');
else
    dbms_output.put_line('no');
end if;

end;
```

3) Write a PL/SQL program to check whether a given character is letter or digit.

```
DECLARE
    get_ctr CHAR(1) := '&input_a_character';
BEGIN
    IF ( get_ctr >= 'A'
        AND get_ctr <= 'Z' )
        OR ( get_ctr >= 'a'
            AND get_ctr <= 'z' ) THEN
        dbms_output.Put_line ('The given character is a letter'); ELSE
        dbms_output.Put_line ('The given character is not a letter');

        IF get_ctr BETWEEN '0' AND '9' THEN
            dbms_output.Put_line ('The given character is a number'); ELSE
            dbms_output.Put_line ('The given character is not a number');
        END IF;
    END IF;
END;
```

Output: A

The given Character is a letter.