PRELAB

1. How many triggers can be applied to a table?

A) There are 6 types of triggers in mysql

1. Before Insert
2. After Insert
3. Before Update
4. After Update
5. Before Delete
6. After Delete

2. Show the two PL/SQL cursor exceptions

A) There are 2 types of PL/SQL cursor exceptions

1. CURSOR_ALREADY_OPEN   - Reason for this exception is when you open a cursor that is already open
2. INVALID_CURSOR - Reason for this exception is When you perform an invalid operation on a cursor like closing a cursor, fetch data from a cursor that is not opened.

3.Explain 3 basic parts of trigger

A) A trigger has 3 basic parts

1. Triggering event or statement
2. Trigger Restriction
3. Trigger Action

4.What are character function?

A) A character function is a function that takes one or more character values as parameters and returns either a character value or a number value. The Oracle Server and PL/SQL provide a number of different character datatypes, including CHAR, VARCHAR, VARCHAR2, LONG, RAW, and LONG RAW.

5.Explain TTITLE and BTITLE
A) **SQL***Plus substitution variables (& variables) are expanded before **BTITLE** is executed. The resulting string is stored as the **BTITLE** text.

You can avoid this double substitution in a TTITLE command by not using the & prefix for variables that are to be substituted on each page of results. If you want to use a substitution variable to insert unchanging text in a TTITLE, enclose it in quotes so that it is only substituted once.

6. What are the uses of SYSDATE and USER keywords?

A) SYSDATE refers to the current server system date. It is a pseudo column. USER is also a pseudo column but refers to the current user logged onto the session. They are used to monitor changes happening in table.

7. How does ROWID help in running a query faster?

A) ROWID s are the fastest way to access a row of data, but if you can do an operation in a single DML statement, that is faster than selecting the data first, then supplying the ROWID to the DML statement. If rows are moved, the ROWID will change. Rows can move due to maintenance operations like shrinks and table moves.

INLAB

Implement PL/SQL Programs on Case Study 8 (SAINT GOBAIN)

1) Write a cursor to select the five Expected amount from Quotation tables.
```
delimiter @
create procedure exp_amount()
begin
            declare exp_amount int;
    declare count int default 0;
    declare q_finished int default 0;
    declare c1 cursor for select exp_amt from quotation;
    declare continue handler for not found set q_finished=1;
open c1;
            q_details:loop
                    fetch c1 into exp_amount;
        select exp_amount;
        set count=count+1;
        if count=5 then
                    leave q_details;
        elseif q_finished=1 then
        leave q_details;
        end if;
                end loop q_details;
close c1;
end @
delimiter ;

call exp_amount();
```

2) Write the Pl/SQL program provides information on the customers who paid the highest Advance amount from Quotation table
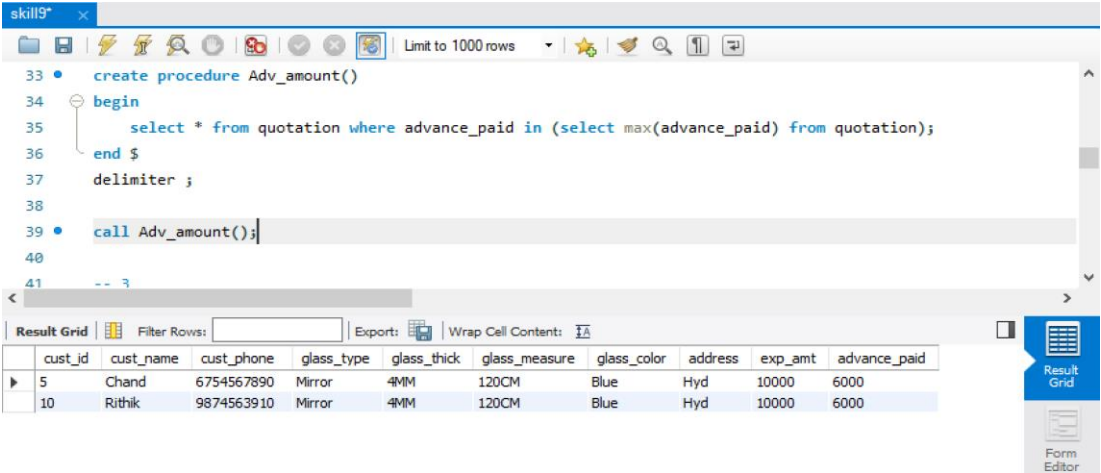
```
delimiter $
create procedure Adv_amount()
begin
          select * from quotation where advance_paid in (select max(advance_paid) from
quotation);
end $
delimiter ;

call Adv_amount();
```


3) Write the procedure to Display the Maximum to Minimum Exp_amt in the Quotation table.

```
Delimiter $$
create procedure sorted_quotation()
begin
          select cust_id,cust_name,exp_amt from quotation order by exp_amt desc;
end $$
delimiter ;
call sorted_quotation();
```
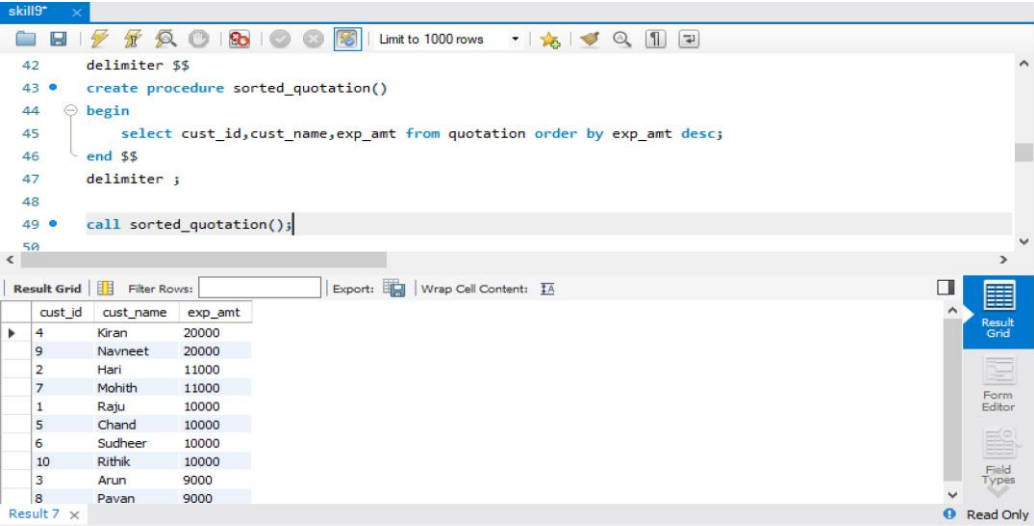
4) Create a function that takes Cust_Id and returns the name of the customer

```
delimiter @@
create function Customer_name(c_id int) returns varchar(20)
begin
        declare c_name varchar(20);
   select cust_name into c_name from quotation where cust_id=c_id;
   return c_name;
end @@
delimiter ;

select Customer_name(1);
```
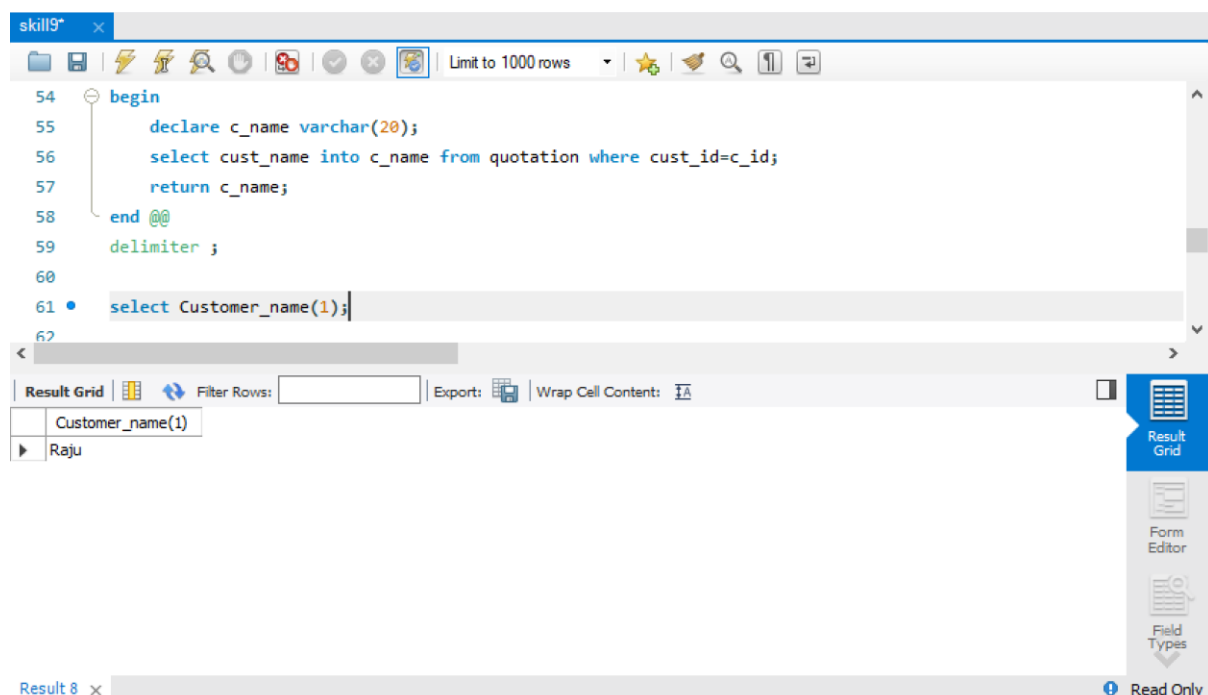

5) Write a function to list the Glass_type and Glass_feature in Quotation and Bill Table.

```
delimiter $@$
create function glass(cus_id int) returns varchar(50)
begin
        declare g_type varchar(20);
   declare g_feature varchar(20);
   select glass_type into g_type from quotation where cust_id=cus_id;
   select glass_feature into g_feature from bill where cust_name in (select cust_name from
quotation where cust_id=cus_id);
   return concat(g_type,',',g_feature);
end $@$
delimiter ;

select glass(1);
```

```
68        declare g_feature varchar(20);
69        select glass_type into g_type from quotation where cust_id=cus_id;
70        select glass_feature into g_feature from bill where cust_name in (select cust_name from quotation where
71        return concat(g_type,',',g_feature);
72    end $@$
73    delimiter ;
74
75 ●  select glass(1);
76
```

glass(1)
▶ Clear glass,Good

6) Write a procedure to delete, customers not paid any advance

```
delimiter @$
create procedure delete_noadvance()
begin
            delete from quotation where advance_paid is NULL or advance_paid = 0;
end @$
delimiter ;
insert into quotation
values(11,'RK',7286009239,'Mirror','4MM','120CM','Blue','Hyd',10000,0);
-- before delete
select * from quotation;
call delete_noadvance();
-- after delete
select * from quotation;
```

Before Delete


```
79 ●  create procedure delete_noadvance()
80    begin
81        delete from quotation where advance_paid is NULL or advance_paid = 0;
82    end @$
83    delimiter ;
84 ●  insert into quotation values(11,'RK',7286009239,'Mirror','4MM','120CM','Blue','Hyd',10000,0);
85    -- before delete
86 ●  select * from quotation;
87 ●  call delete noadvance();
```

cust_id	cust_name	cust_phone	glass_type	glass_thick	glass_measure	glass_color	address	exp_amt	advance_paid
4	Kiran	6754567890	Mirror	3MM	200CM	Blue	Hyd	20000	5000
5	Chand	6754567890	Mirror	4MM	120CM	Blue	Hyd	10000	6000
6	Sudheer	1122334455	Clear glass	4MM	140CM	Black	Hyd	10000	2000
7	Mohith	9988556644	Mirror	5MM	150CM	Blue	Delhi	11000	2500
8	Pavan	9856423183	Clear glass	6MM	120CM	Black	Mumbai	9000	1000
9	Navneet	7894561233	Mirror	3MM	200CM	Blue	Hyd	20000	5000
10	Rithik	9874563910	Mirror	4MM	120CM	Blue	Hyd	10000	6000
11	RK	7286009239	Mirror	4MM	120CM	Blue	Hyd	10000	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

quotation 10 ×

After Delete

```
82     end @$
83     delimiter ;
84  •  insert into quotation values(11,'RK',7286009239,'Mirror','4MM','120CM','Blue','Hyd',10000,0);
85     -- before delete
86  •  select * from quotation;
87  •  call delete_noadvance();
88     -- after delete
89  •  select * from quotation;
```
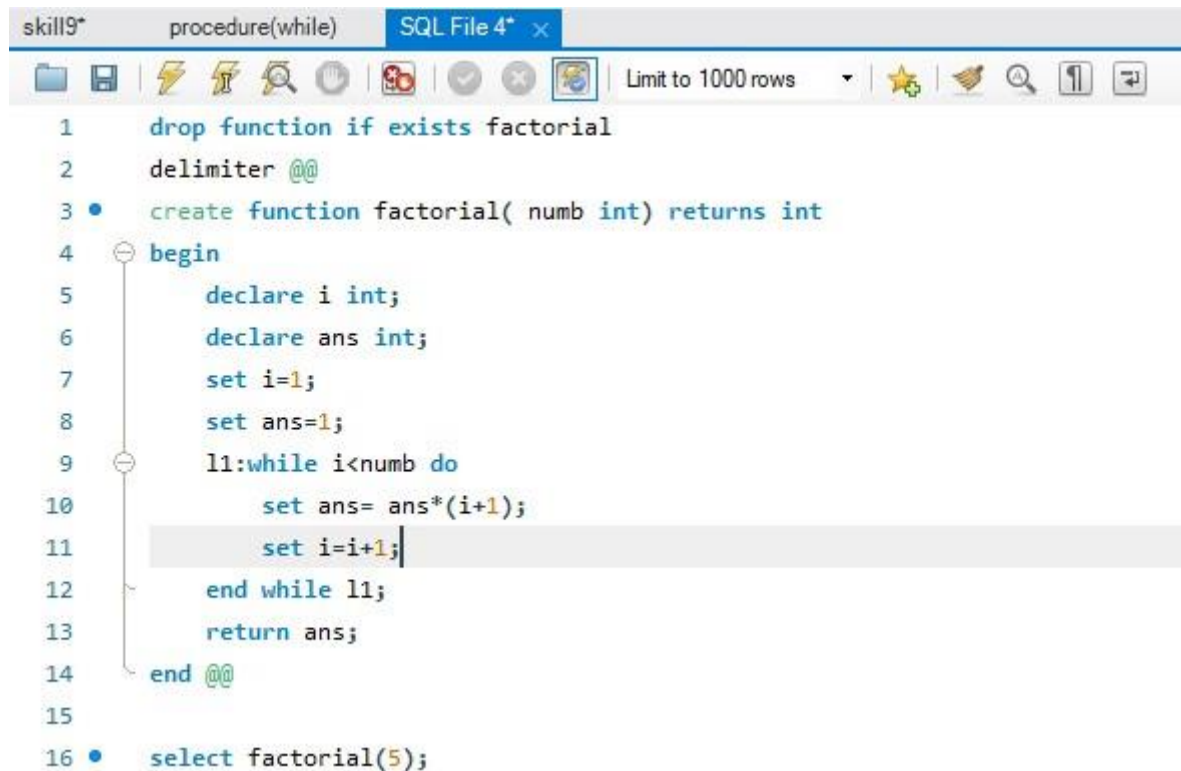
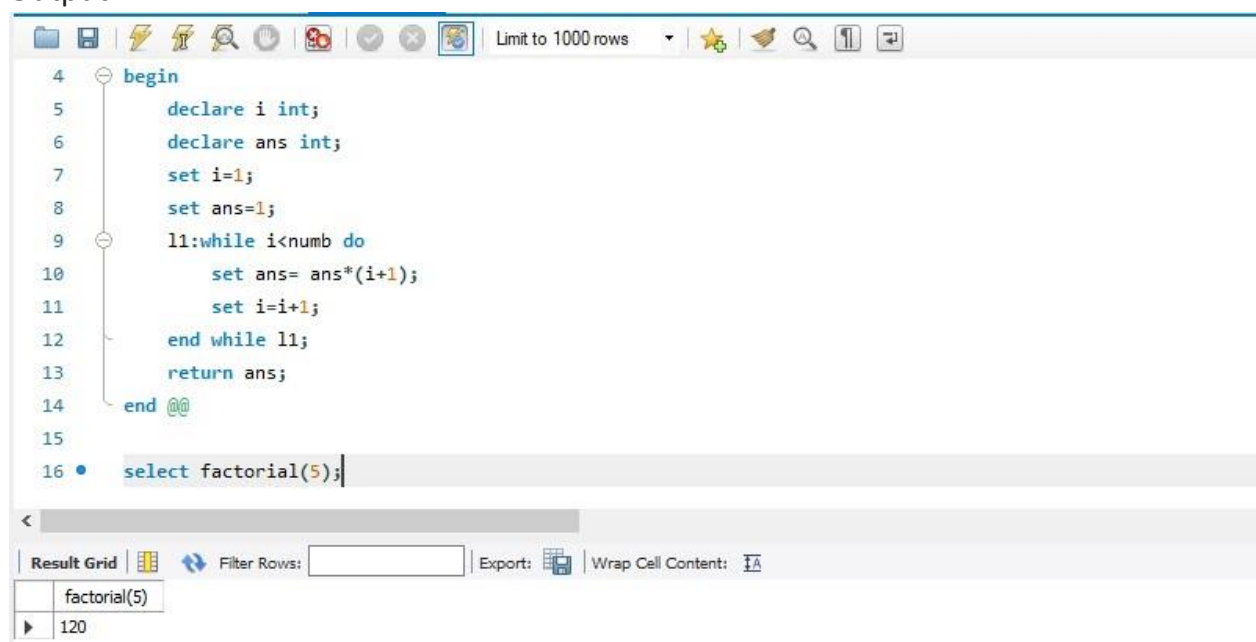| cust_id	cust_name	cust_phone	glass_type	glass_thick	glass_measure	glass_color	address	exp_amt	advance_paid
3	Arun	7567896546	Clear glass	6MM	120CM	Black	Mumbai	9000	1000
4	Kiran	6754567890	Mirror	3MM	200CM	Blue	Hyd	20000	5000
5	Chand	6754567890	Mirror	4MM	120CM	Blue	Hyd	10000	6000
6	Sudheer	1122334455	Clear glass	4MM	140CM	Black	Hyd	10000	2000
7	Mohith	9988556644	Mirror	5MM	150CM	Blue	Delhi	11000	2500
8	Pavan	9856423183	Clear glass	6MM	120CM	Black	Mumbai	9000	1000
9	Navneet	7894561233	Mirror	3MM	200CM	Blue	Hyd	20000	5000
10	Rithik	9874563910	Mirror	4MM	120CM	Blue	Hyd	10000	6000
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

quotation 11 ×

POSTLAB
1) Write a PL/SQL block to show single and multiline comments.
A) Single-line comments begin with a double hyphen (- -) anywhere on a line and extend to the end of the line. Multi-line comments begin with a slash-asterisk (/*), end with an asterisk-slash (*/), and can span multiple lines.

2) Write a PL/SQl program to display the factorial of a number

```
skill9*        procedure(while)      SQL File 4*  ×

1       drop function if exists factorial
2       delimiter @@
3 •     create function factorial( numb int) returns int
4       begin
5           declare i int;
6           declare ans int;
7           set i=1;
8           set ans=1;
9           l1:while i<numb do
10              set ans= ans*(i+1);
11              set i=i+1;
12          end while l1;
13          return ans;
14      end @@
15
16 •    select factorial(5);
```

Output:

```
4       begin
5           declare i int;
6           declare ans int;
7           set i=1;
8           set ans=1;
9           l1:while i<numb do
10              set ans= ans*(i+1);
11              set i=i+1;
12          end while l1;
13          return ans;
14      end @@
15
16 •    select factorial(5);
```

| Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

factorial(5)
120

3) **Given a year, report if it is a leap year. The tricky thing here is that a leap**
   year in the Gregorian calendar occurs: on every year that is evenly
   divisible by 4   except every year that is evenly divisible by 100   unless
   the year is also evenly divisible by 400

For example, 1997 is not a leap year, but 1996 is. 1900 is not a leap year, but 2000
is. If your language provides a method in the standard library that does this look-
up, pretend it doesn't exist and implement it yourself. Find the solution using pl/sql

```
1     delimiter @@
2 •   create function leapyear_check ( y int) returns varchar(45)
3     begin
4         declare i varchar(45);
5         declare j varchar(45);
6         set i= "This is leap year";
7         set j= "This is not a leap year";
8         if y % 4 =0 then
9           if y % 100 =0 then
10            if y % 400 =0 then
11                return i;
12            else
13                return j;
14            end if;
15          else
16              return i;
17          end if;
18        else
19            return j;
20        end if;
21    end @@;
22    delimiter
23
24 •  select leapyear_check(2012);
```

Output:

```
21      end @@;
22      delimiter
23
24 •   select leapyear_check(1999);
```

leapyear_check(1999)
▶ This is not a leap year

Result 3 ✕   ❶ Read Only

4) Find the difference between the square of the sum and the sum of
the squares of the first N natural numbers. The square of the sum of the
first ten natural numbers is $(1 + 2 + ... + 10)^2 = 55^2 = 3025$. The sum of
the squares of the first ten natural numbers is $1^2 + 2^2 + ... + 10^2 = 385$.

Hence the difference between the square of the sum of the first ten natural numbers and the sum of the squares of the first ten natural numbers is 3025 - 385 = 2640. Find the solution using pl/sql.

```
delimiter @@
create function fourthsum (numb int) returns int
begin
    declare i int;
    declare ans1 int;
    declare sumofsquares int;
    declare squaresofsum int;
    declare final_answer int;
    set i=1;
    set ans1=0;
    loop1: while i<=numb do
        set ans1 = ans1 + (i);
        set i=i+1;
    end while loop1;
    set sumofsquares= ans1*ans1;
    set i=1;
    set ans1=0;
    loop2: while i<=numb do
        set ans1=ans1 + (i*i);
        set i=i+1;
    end while loop2;
    set squaresofsum = ans1;
    set final_answer=sumofsquares-squaresofsum;
    return final_answer;
end @@
```

Output:

```
select fourthsum(5);
```

fourthsum(5)
170