EXPERIMENT 12
Implement PL/SQL Programs on Case Study 3& 6
(WAREHOUSE SYSTEM) & (PAINITING HIRE BUSINESS)

PRE-LAB

1. Consider the following schedule for transactions T1, T2 and T3 .

T1	T2	T3
Read (X)		
	Read (Y)	
		Read (Y)
	Write (Y)	
Write (X)		
		Write (X)
	Read (X)	
	Write (X)	

State the appropriate schedules which matches the correct serialization of the above?

Ans)

Explanation: T1 can complete before T2 and T3 as there is no conflict between write(x) of T1 and the operations in T2 and T3 which occurs before write(x) of T1 in the above diagram

2. What is displayed on the screen after I execute the following statements?

```
CREATE TABLE plch_stuff
(
 id   INTEGER PRIMARY KEY,
 nm   VARCHAR (5) UNIQUE
)
/
DECLARE l_count PLS_INTEGER;
BEGIN
INSERT INTO plch_stuff
VALUES (1, 'Hat');
INSERT INTO plch_stuff
 VALUES (1, 'Jacket');
  DECLARE EXIT HANDLER FOR SQLEXCEPTION
  BEGIN
  SELECT COUNT (*) INTO l_count FROM plch_stuff;
  PUT_LINE (CONCAT('Rows = ' , ifnull(l_count,'')));
  END;
END;
/
```

Ans)

An error is displayed during execution of following statements because the id is declared as primary key (Unique) but insertion of id is done with same id as '1'

3. Consider the transactions T1, T2, and T3 and the schedules S1 and S2 given below.

T1:r1(X);r1(Z);w1(X);w1(Z)

T2:r2(Y);r2(Z);w2(Z)

T3:r3(Y);r3(X);w3(Y)

S1: r1(X); r3(Y); r3(X); r2(Y); r2(Z); w3(Y); w2(Z); r1(Z); w1(X); w1(Z)

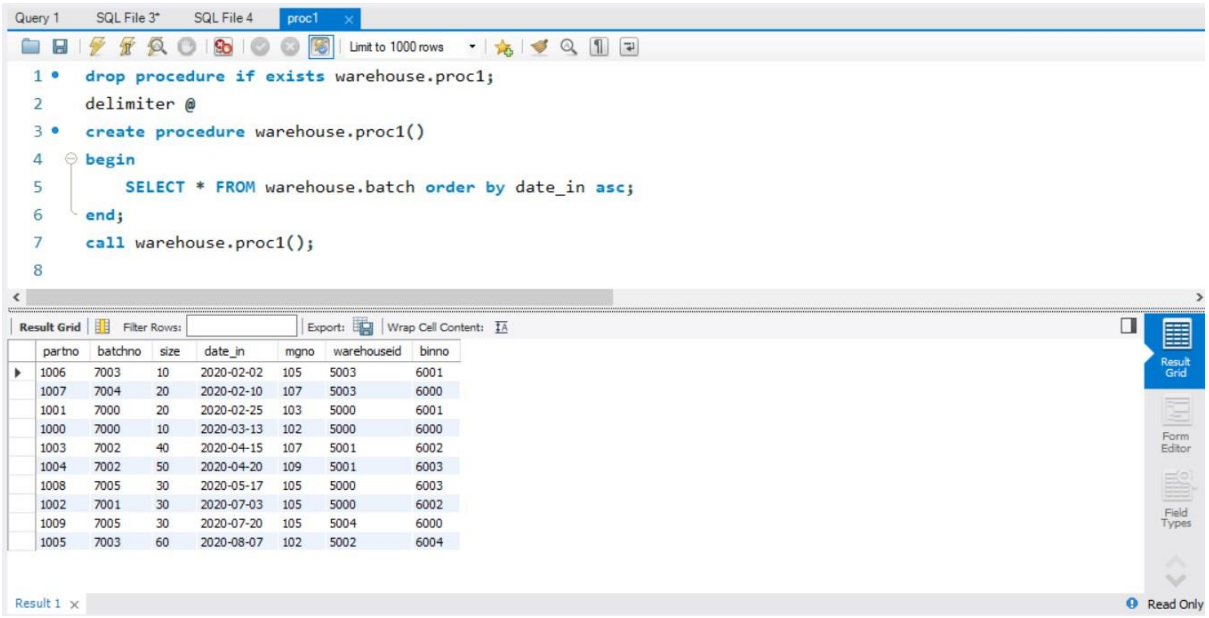
S2: r1(X); r3(Y); r2(Y); r3(X); r1(Z); r2(Z); w3(Y); w1(X); w2(Z); w1(Z)
Which one of the following statements about the schedules is TRUE?
(A) Only S1 is conflict-serializable (B) Only S2 is conflict-serializable
(C) Both S1 and S2 are conflict-serializable (D) Neither S1 nor S2 is conflict-serializable

Ans)
T1: r1(x) ; r1(z) : w1(x) ; w1(z)
T2: r2(y) ; r2(z) ; w2(z)
T3: r3(y) ; r3(x) ; w3(y)
S1: r1(x) : r3(y) : r3(x) ; r2(y) ; r2(z) ; W3(y) ; w2(z) ; r1(z) ; w1(x) ; w1(z)
S2: r1(x) ; r3(y) ; r2(y) ; r3(x) ; r1(z); R2(z); w3(y); w1(x); w2(z) ; w1(z) ;

Solution: Only S1 is conflict -serializable.

IN-LAB
PL/SQL PROGRAMS ON WAREHOUSE SYTEM

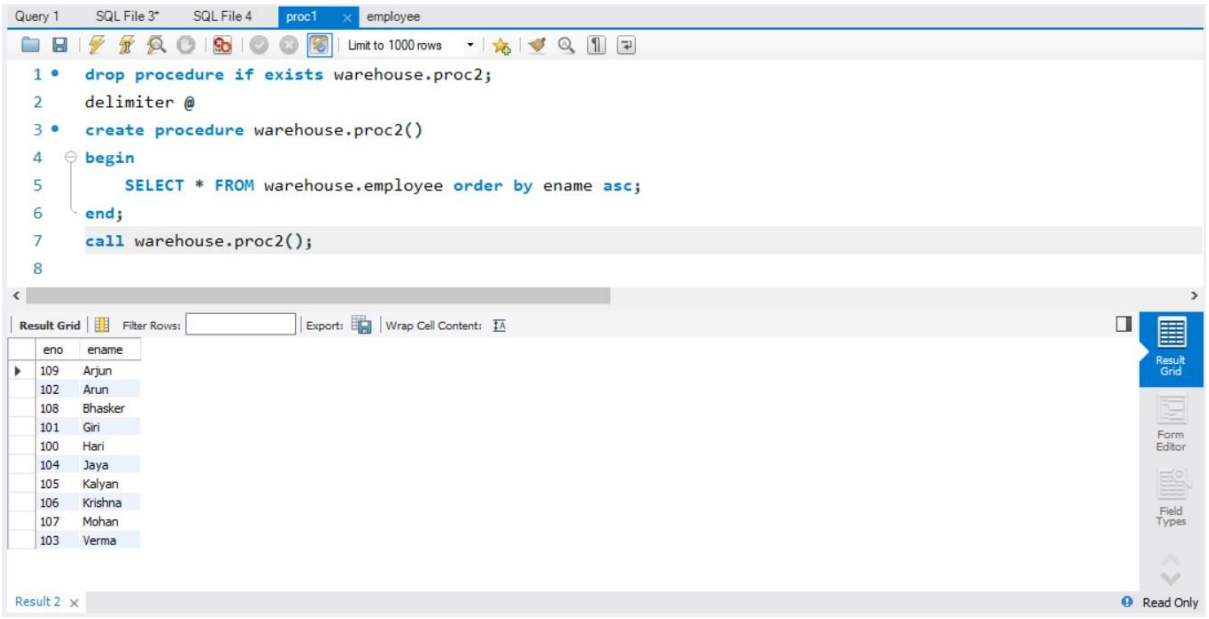
1) Write PL/SQL Program to display Batch details in ascending order of Date



```
drop procedure if exists warehouse.proc1;
delimiter @
create procedure warehouse.proc1()
begin
    SELECT * FROM warehouse.batch order by date_in asc;
end;
call warehouse.proc1();
```

partno	batchno	size	date_in	mgno	warehouseid	binno
1006	7003	10	2020-02-02	105	5003	6001
1007	7004	20	2020-02-10	107	5003	6000
1001	7000	20	2020-02-25	103	5000	6001
1000	7000	10	2020-03-13	102	5000	6000
1003	7002	40	2020-04-15	107	5001	6002
1004	7002	50	2020-04-20	109	5001	6003
1008	7005	30	2020-05-17	105	5000	6003
1002	7001	30	2020-07-03	105	5000	6002
1009	7005	30	2020-07-20	105	5004	6000
1005	7003	60	2020-08-07	102	5002	6004

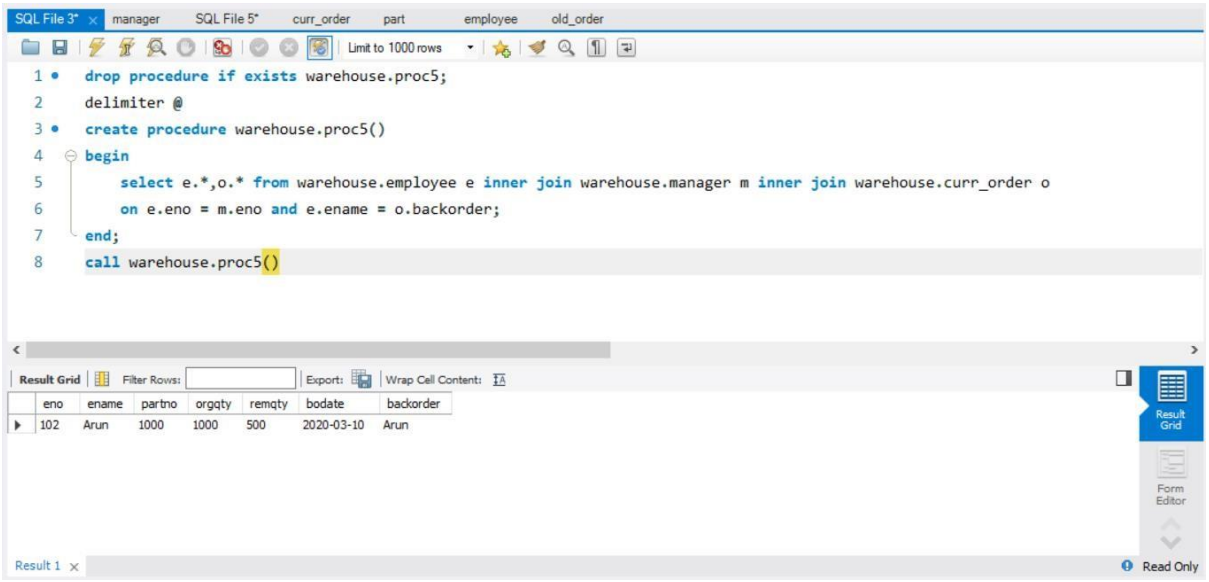
2) Write PL/SQL Program to display the names and employee_id for all the managers. Names should be listed in alphabetic order.



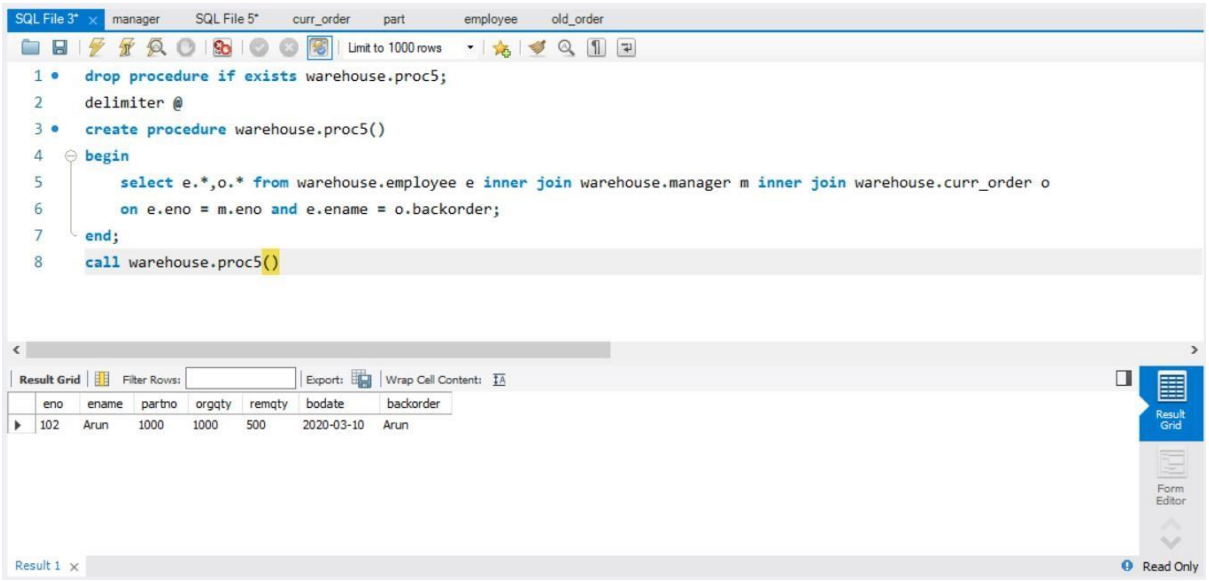
```
drop procedure if exists warehouse.proc2;
delimiter @
create procedure warehouse.proc2()
begin
    SELECT * FROM warehouse.employee order by ename asc;
end;
call warehouse.proc2();
```

eno	ename
109	Arjun
102	Arun
108	Bhasker
101	Giri
100	Hari
104	Jaya
105	Kalyan
106	Krishna
107	Mohan
103	Verma

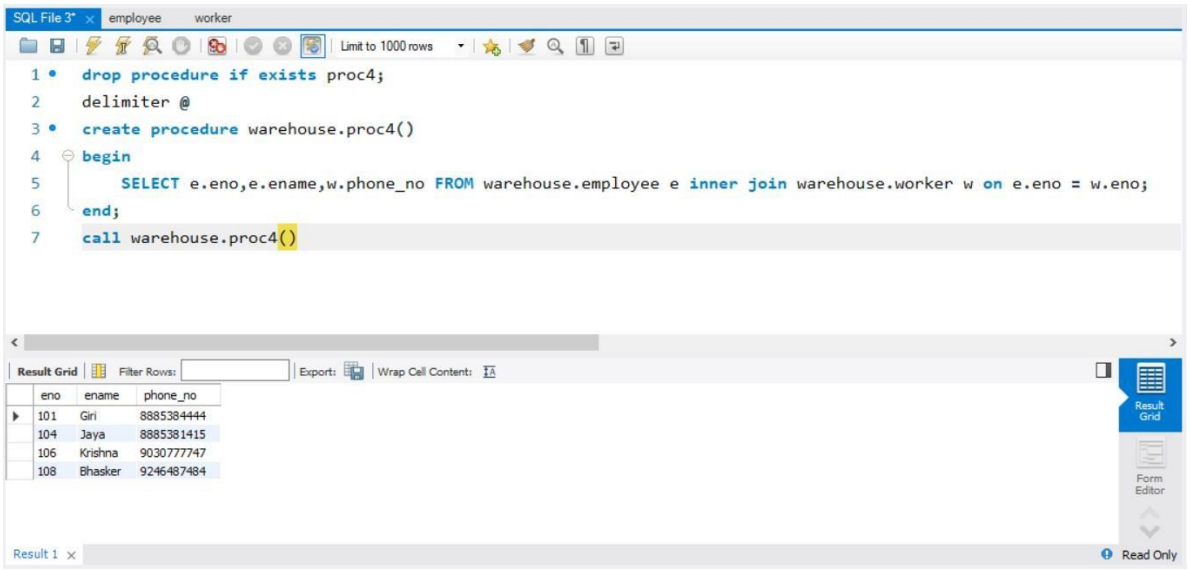
3) Write PL/SQL Program to list all current and old backorders done by the each manager. For each backorder you have to list the part no, backorder date, and fulfilled date. For current backorders, list a fulfilled date '2020-01-01'.



4) Write PL/SQL Program to list out the remaining capacity of the bin for each warehouse.



5) Write a PL/SQL Program to display all the phones and employee no for all the workers.

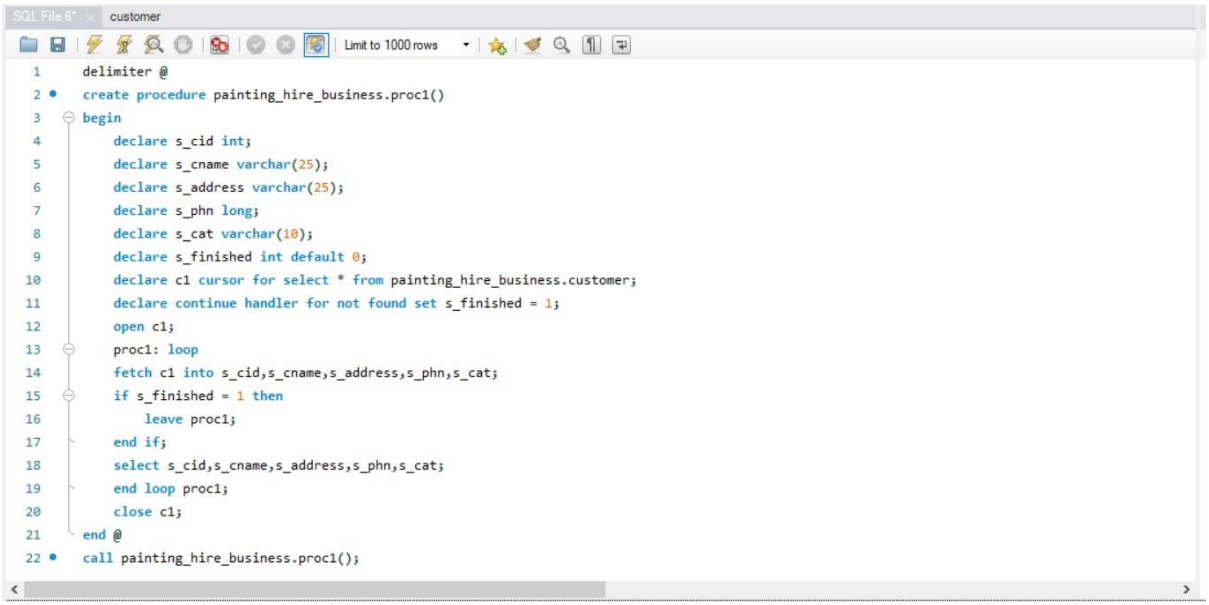


```
drop procedure if exists proc4;
delimiter @
create procedure warehouse.proc4()
begin
    SELECT e.eno,e.ename,w.phone_no FROM warehouse.employee e inner join warehouse.worker w on e.eno = w.eno;
end;
call warehouse.proc4()
```

eno	ename	phone_no
101	Giri	8885384444
104	Jaya	8885381415
106	Krishna	9030777747
108	Bhasker	9246487484

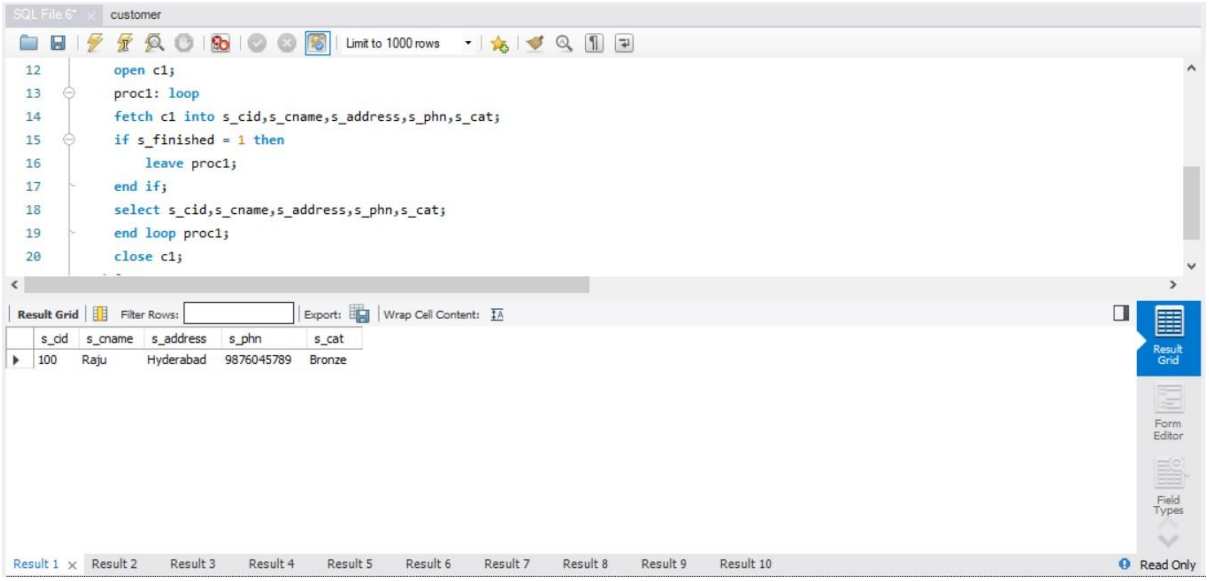
PL/SQL PROGRAMS ON PAINTING HIRE BUSINESS

1) Create a cursor to display all the customer details

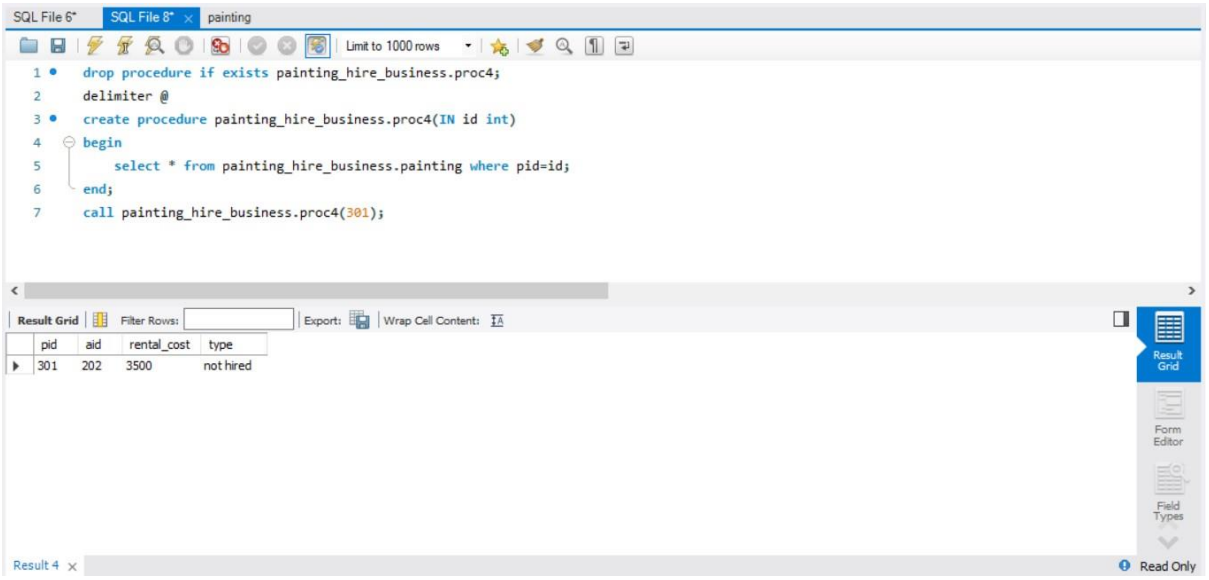


```
delimiter @
create procedure painting_hire_business.proc1()
begin
    declare s_cid int;
    declare s_cname varchar(25);
    declare s_address varchar(25);
    declare s_phn long;
    declare s_cat varchar(10);
    declare s_finished int default 0;
    declare c1 cursor for select * from painting_hire_business.customer;
    declare continue handler for not found set s_finished = 1;
    open c1;
    proc1: loop
    fetch c1 into s_cid,s_cname,s_address,s_phn,s_cat;
    if s_finished = 1 then
        leave proc1;
    end if;
    select s_cid,s_cname,s_address,s_phn,s_cat;
    end loop proc1;
    close c1;
end @
call painting_hire_business.proc1();
```

Output:



2) Create a procedure to display the required painting details



3) Create a cursor to display the painting details which are not hired

```
delimiter @
create procedure painting_hire_business.proc2()
begin
    declare s_pid int;
    declare s_aid int;
    declare s_rentcost int;
    declare s_type varchar(25);
    declare s_finished int default 0;
    declare c1 cursor for select * from painting_hire_business.painting where type = 'not hired';
    declare continue handler for not found set s_finished = 1;
    open c1;
    proc2: loop
    fetch c1 into s_pid,s_aid,s_rentcost,s_type;
    if s_finished = 1 then
        leave proc2;
    end if;
    select s_pid,s_aid,s_rentcost,s_type;
    end loop proc2;
    close c1;
end @
call painting_hire_business.proc2();
```

Output:

```
    open c1;
    proc2: loop
    fetch c1 into s_pid,s_aid,s_rentcost,s_type;
    if s_finished = 1 then
        leave proc2;
    end if;
    select s_pid,s_aid,s_rentcost,s_type;
    end loop proc2;
    close c1;
end @
call painting_hire_business.proc2();
```

s_pid	s_aid	s_rentcost	s_type
301	202	3500	not hired

4) Create a cursor to display customer details which have been hired

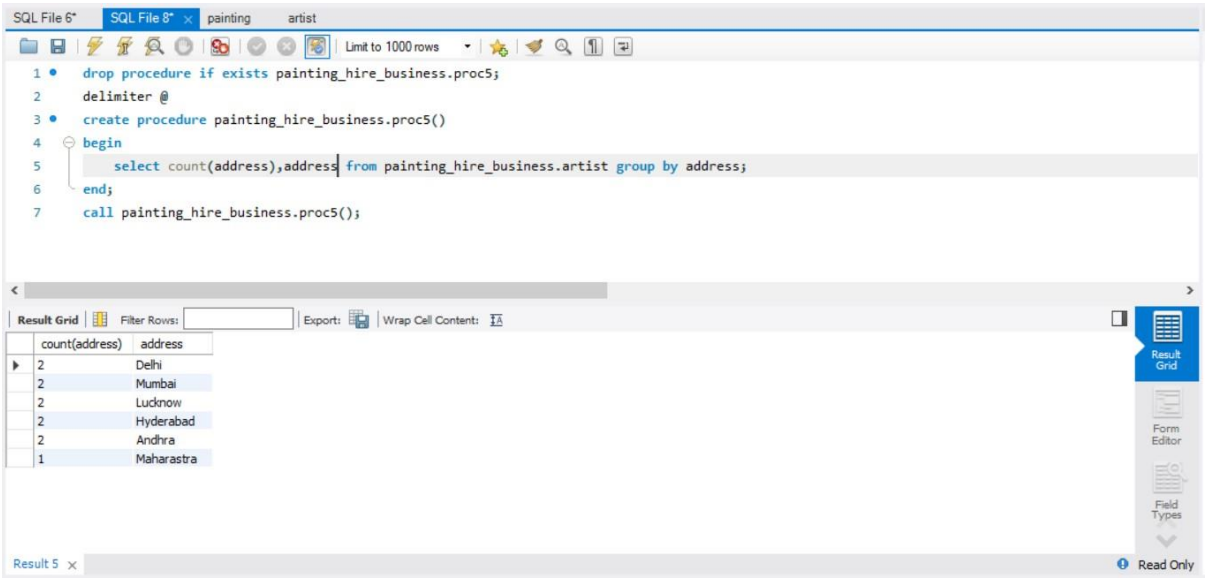
```
drop procedure if exists painting_hire_business.proc3;
delimiter @
create procedure painting_hire_business.proc3()
begin
    declare s_cid int;
    declare s_cname varchar(25);
    declare s_pid int;
    declare s_phn long;
    declare s_cat varchar(25);
    declare s_type varchar(25);
    declare s_finished int default 0;
    declare c1 cursor for select c.cid,cname,p.pid,phone,category,type from painting_hire_business.customer c inner join
    painting_hire_business.rent r inner join painting_hire_business.painting p on c.cid=r.cid and r.pid=p.pid
    where p.type='hired';
    declare continue handler for not found set s_finished = 1;
    open c1;
    proc3: loop
    fetch c1 into s_cid,s_cname,s_pid,s_phn,s_cat,s_type;
    if s_finished = 1 then
        leave proc3;
    end if;
    select s_cid,s_cname,s_pid,s_phn,s_cat,s_type;
    end loop proc3;
    close c1;
end @
call painting_hire_business.proc3();
```

Output:

```
delimiter @
create procedure painting_hire_business.proc3()
begin
    declare s_cid int;
    declare s_cname varchar(25);
    declare s_pid int;
    declare s_phn long;
    declare s_cat varchar(25);
    declare s_type varchar(25);
    declare s_finished int default 0;
    declare c1 cursor for select c.cid,cname,p.pid,phone,category,type from painting_hire_business.customer c inner join
    painting_hire_business.rent r inner join painting_hire_business.painting p on c.cid=r.cid and r.pid=p.pid
    where p.type='hired';
```

s_cid	s_cname	s_pid	s_phn	s_cat	s_type
105	Haritha	302	9611665513	Silver	hired

5) Create a procedure to display the count of artists in each city



POST-LAB

1) Write a PL/SQL block to describe the usage of LIKE operator including wildcard characters and escape character.

Ans)

```
Declare
        Procedure pat_match( test_string varchar(10), pattern varchar(10)) is
Begin
          If test_string like pattern then.
           Dbms_output.put_line('true');
         Else
           Dbms_output.put_line('False');
End If;
End;
Begin
Path_match('blweate' , 'B/.a_e');
Path_match('Blweate' , 'B/.A_E');
End;
```

2) Write a PL/SQL block to show the operator precedence and parentheses in several more complex expressions.

Ans)

```
Declare
          Salary number :=40000;
          Commision number :=0.15;
   Begin
            Dbms_output.put_line('8+20/4=' || (8+20/4));
            Dbms_output.put_line('20/4+8=' || (20/4+8));
            Dbms_output.put_line('7+9/3=' || (7+9/3));
            Dbms_output.put_line('(7+9)/3=' || ((7+9)/3));
        Dbms_output.put_line('(salary *0.08) + (commission *0.12)=' || ((salary
*0.08)+((commission * 0.12)));
              Dbms_output.put_line('30+(30/6+(15-8))=' ||(30+(30/6+(15-8))));
 Dbms_output.put_line('salary *0.08 + commission * 0.12 =' || (salary *0.08 + commission
*0.12));
End;
/
```

3) Write a PL/SQL block to create a procedure using the "IS [NOT] NULL Operator" and show AND operator returns TRUE if and only if both operands are TRUE.

Ans)

```
Begin
      If bool_val is null then
            Dbms_output.put_line(boo_name || '=Null');
        Else If bool_val =True then
            Dbms_output.put_line(boo_name || '=True');
       Else
            Dbms_output.put_line(boo_name || '=False');
       End if;
End;
/
```

4) Write a PL/SQL block to show an invalid case-insensitive reference to a quoted and without quoted user-defined identifier.

Ans)

```
Declare
      "Welcome varchar(10):= 'welcome';
Begin
      Dbms_output.put_line("welcome");
End;
/
```

5) Write a PL/SQL block to explain single and multiline comments

Ans)

```
Declare
       Condition Boolean;
          Pi number :=3.145;
       Radius number :=10;
       Area   number ;
Begin
      If 2+2 =4 then
            Condition = true;
       End if;
       Area = pi *radius*2;
      Dbms_output.put_line('The area of circle is :' || area);
End;
/
```

6) Consider the following transaction involving two bank accounts x and y. read(x); x : = x– 50; write(x); read(y); y:=y+50; write(y) The constraint that the sum of the accounts x and y should remain constant is that of
   (A) Atomicity
   (B) Consistency
   (C) Isolation
   (D) Durability

Ans)

Option (B) Consistency
Explanation : Consistency in database system refer to the requirement that any given database transaction must only change affected data in allowed ways , that is sum of x and y must not change

7) Consider a simple checkpointing protocol and the following set of operations in the log. (start, T4); (write, T4, y, 2, 3); (start, T1); (commit, T4); (write, T1, z, 5, 7);(checkpoint); (start, T2); (write, T2, x, 1, 9); (commit, T2); (start, T3), (write, T3, z, 7, 2);If a crash happens now the system tries to recover using both undo and redo operations, what are the contents of the undo list and the redo list.
   (A) Undo: T3, T1; Redo: T2
   (B) Undo: T3, T1; Redo: T2, T4
   (C) Undo: none; Redo; T2, T4, T3, T1
   (D) Undo: T3, T1, T4; Redo: T2

Ans)
Option (A) Undo :T3, T1 ; Redo:T2.

8) Amongst the ACID properties of a transaction, the 'Durability' property requires that the changes made to the database by a successful transaction persist
   A. Except in case of an Operating system crash
   B. Expect in case of Disk Crash
   C. Expect in case of a power failure
   D. Always, even if there is a failure of any kind

Ans)
Option (D) Always ,even if there is a failure of any kind