ADBMS LAB

CO2- Programs

PL/SQL

Q1: Write a PL/SQL program to find the sum of 2 numbers.

```
declare a number;
 b number;
c number;
begin
 a:=&a;
 b:=&b;
 c:=a+b;
dbms_output.put_line('sum :'||c);
end;
/
```

OUTPUT

```
###################################
Enter value for a:  10
old   6:  a:=&a;
new       6:     a:=  10;
Enter  value  for  b:  5
old   7: b:=&b;
new7:  b:=5;
sum :15
PL/SQL procedure successfully completed.
```

Q2: Write a PL/SQL program to find the factorial of a given number

```
declare
n number;
 fact number:=1;
begin
 n:=&n;
 for i in 1..n
loop
fact:=fact*i;
end loop;
 dbms_output.put_line('Factorial :'||fact);
end;
/
```

OUTPUT

########################

Enter value for n: 5
old 5: n:=&n; new
5: n:=5;
Factorial :120
PL/SQL procedure successfully completed.

Q3: Write a PL/SQL program to check whether the given no is prime or not

```
declare
 n number;
 i number:=2;
flag number:=1;
begin
n:=&n;
for i in 2..n/2
loop
if mod(n,i)=0
then
flag:=0;
exit;
end if;
end loop;

if flag=1
 then
dbms_output.put_line('prime');
else
dbms_output.put_line('not prime');
end if;
 end;
/
```

OUTPUT

##

Enter value for n: 7
old 6: n:=&n;
new 6: n:=7;
prime

Enter value for n: 4
old 6: n:=&n;
new 6: n:=4; not
prime
PL/SQL procedure successfully completed.

Functions

1. Write a PL/SQL program to find the sum of 2 numbers using functions

```
2  c number;
3  begin
4  c:=a+b;
5  return c;
6  end;
7  /
```

Function created.

```
SQL> declare
2  result number;
3  begin
4  result:=add_fun(30,40);
5  dbms_output.put_line('Result is: '||result);
6  end;
7  /
```

Result is: 70

PL/SQL procedure successfully completed.

2. Write a PL/SQL program to Check whether a number is Armstrong or not using functions

```
2    r number(10);
3    a number(10);
4    b number(10);
5    c number(10);
6    begin
7    b:=0;
8    c:=n;
9    while(c>0)
```

```
10  loop
11  r:=c mod 10;
12  b:=b+(r*r*r);
13  c:=floor(c/10);
14  end loop;
15  return b;
16  end armstrong;
17  /
Function created.
SQL> declare
2    n number(10);
3    m number(10);
4    begin
5    n:=&n;
6    m:=armstrong(n);
7    if(m=n) then
8    dbms_output.put_line('Given no is armstrong number');
9    else
10  dbms_output.put_line('Given no is not an armstrong number');
11  end if;
12  end;
13  /
Enter value for n: 153
old     5:  n:=&n;  new
5: n:=153;
Given no is armstrong number
PL/SQL procedure successfully completed.
```

3. **Create table that contains itemid,item_name & price of several items sold in a grocery shop, Using functions retrieve the item name & price from table when itemid is given as input.**

SQL> create table item(item_id integer primary key,itemname varchar(20),price float);

Table created.

SQL> insert into item values(&item_id,'&itemname',&price);

Enter value for item_id: 2334

Enter value for itemname: Geera Enter value for price:

206.25 old 1: insert into item

values(&item_id,'&itemname',&price) new 1: insert into

item values(2334,'Geera',206.25)

1 row created.

SQL> insert into item values(&item_id,'&itemname',&price);

Enter value for item_id: 2532

Enter value for itemname: Corn soup Enter value for price:

34.65 old 1: insert into item

values(&item_id,'&itemname',&price) new 1: insert into

item values(2532,'Corn soup',34.65)

1 row created.

SQL> insert into item values(&item_id,'&itemname',&price);

Enter value for item_id: 2124

Enter value for itemname: Lays Enter value for price: 20 old

1: insert into item values(&item_id,'&itemname',&price) new

1: insert into item values(2124,'Lays',20)

1 row created.

SQL> insert into item values(&item_id,'&itemname',&price);

Enter value for item_id: 4531

Enter value for itemname: Set Enter value for price: 99.99

old 1: insert into item values(&item_id,'&itemname',&price)

new 1: insert into item values(4531,'Set',99.99)

1 row created.

SQL> insert into item values(&item_id,'&itemname',&price);

Enter value for item_id: 2319

Enter value for itemname: Duracell Enter value for price:

45.5 old 1: insert into item

values(&item_id,'&itemname',&price) new 1: insert into

item values(2319,'Duracell',45.5)

1 row created.

SQL> select * from item;

ITEM_ID	ITEMNAME	PRICE
2334	Geera	206.25
2532	Corn soup	34.65
2124	Lays	20
4531	Set	99.99
2319	Duracell	45.5

SQL> create or replace function itemprice(id number) return number is

2 p item.price % type;

3 begin

```
4  select price into p from item where item_id=id;
5  return(p);
6  end;
7  /
Function created.
SQL> declare
2  pr number;
3  id number;
4  begin
5  id:=&itemid;
6  pr:=itemprice(id);
7  dbms_output.put_line('item price is RS:'||pr);
8  end;
9  /
Enter value for itemid: 2124
old   5: id:=&itemid;
new   5: id:=2124;
item price is RS:20
PL/SQL procedure successfully completed.
```

4. **Write a PL/SQL function called POW that takes two numbers as argument and return the value of the first number raised to the power of the second .**

```
SQL> create or replace function pow (n1 number,n2 number) return number as
2  res number;
3  begin
4  select power (n1,n2) into res from dual;
5  return res;
6  end;
7  /
Function created.
```

```
SQL> select power (2,4) from dual;

POWER(2,4)
----------
        16
SQL> declare
  2  a number;
3  b number;
4  begin
5  a:=&a;
6  b:=&b;
7  dbms_output.put_line('power(n1,n2)='||pow(a,b));
8  end;
9  /

Enter value for a: 2
old   5: a:=&a; new
5:     a:=2;     Enter
value for b: 4
old   6: b:=&b; new
6:                 b:=4;
power(n1,n2)=16
PL/SQL procedure successfully completed.
```

PROCEDURE

Q4: PROCEDURE – Selected record's price is incremented by 100, executing the procedure created & displaying the updated table.

Price updated

```
SQL> create table product (product_id integer,product_name varchar(20),price number);

Table created.

SQL> insert into product values(&product_id,'&product_name',&price);

Enter value for product_id: 12

Enter value for product_name: Colgate


Enter value for price: 185


 old   1: insert into productvalues(&product_id,'&product_name',&price)
```

new 1: insert into product values(12,'Colgate',185)

1 row created.

SQL> insert into product values(&product_id,'&product_name',&price);

Enter value for product_id: 13

Enter value for product_name: lays

Enter value for price: 35

old 1: insert into product values(&product_id,'&product_name',&price)

new 1: insert into product values(13,'lays',35)

1 row created.

SQL> insert into product values(&product_id,'&product_name',&price);

Enter value for product_id: 14

Enter value for product_name: book

Enter value for price: 90

old 1: insert into productvalues(&product_id,'&product_name',&price)

new 1: insert into product values(14,'lays',90)

1 row created.

SQL> select * from product;

PRODUCT_ID	PRODUCT_NAME	PRICE
12	Colgate	135
13	lays	35
14	book	90

```
SQL> create or replace procedure product1(id number,total number) is
1   p number;
2   null_price exception;
3   begin
4   select price into p from product where product_id=id;
5   if p is null then
6   raise null_price;
7   else
8   update product set price=price+total where product_id=id;
9   end if;
10  exception
11  when null_price then
12  dbms_output.put_line('Price is null');
13  end;
14  /

Procedure created.

SQL> exec product1(13,100)
PL/SQL procedure successfully completed.
SQL> select * from product;

PRODUCT_ID PRODUCT_NAME            PRICE
---------- -------------------- ----------
12                 colgate                185
13                lays                    135
14                  book                  90
```

Q5: Write a PL/SQL program to Perform Banking Operations Using Procedures

create table acc(acno integer primary key,name varchar(20),balance float);

Table created.

SQL> insert into acc values(1,'Renju',10000);

1 row created.

SQL> insert into acc values(2,'Tony',25000);

1 row created.

SQL> insert into acc values(3,'Rohini',7000);

1 row created.

SQL> insert into acc values(4,'Deena',15000);

1 row created.

SQL> insert into acc values(5,'krishna',35000);

1 row created.

SQL> select *from acc;

ACNO	NAME	BALANCE
1	Renju	10000
2	Tony	25000
3	Rohini	7000
4	Deena	15000
5	krishna	35000

```
create or replace procedure withdraw(ac_no1 in number,amount1 in number) is
2  begin
3  update acc set balance=balance-amount1 where acno=ac_no1;
4  end;
5  /
```

Procedure created.

```
SQL> create or replace procedure deposit(ac_no1 in number,amount1 in number) is
2  begin
3  update acc set balance=balance+amount1 where acno=ac_no1;
4  end;
5  /

Procedure created.

declare
choice       number;
ac_no1  number(5);
amount number(5);
begin

ac_no1:=&ac_no1;
choice:=&choice;
amount:=&amount;
dbms_output.put_line('1.withdraw');
dbms_output.put_line('2.deposit');
 if choice=1 then
withdraw(ac_no1,amount);
else
deposit(ac_no1,amount);
end if;
 end;
/
Enter value for accno1: 1

old   6: accno1:=&accno1;

new   6: accno1:=1;

Enter value for choice: 1
old   7: choice:=&choice;

new   7: choice:=1:

Enter value for amount: 1000

old     8:  amount:=&amount;

new   8: amount:=1000;

PL/SQL procedure successfully completed.

SQL> select * from acc;

     ACNO NAME          BALANCE
---------- --------------------         ----------
1          Renju                9000
```

```
2      Tony              24000
3      Rohini                 6000
4      Deena                  14000

SQL> declare

2    choice number;

3    accno1 number(5);

4    amount number(5);

5    begin

6    accno1:=&accno1;

7    choice:=&choice;

8    amount:=&amount;

9    if choice=1 then

10   deposit(accno1,amount);

11   else

12   withdraw(accno1,amount);

13   end if;

14   end;

15   / Enter value for accno1: 1 old   6: accno1:=&accno1; new   6:

     accno1:=1; Enter value for choice: 1

old    7:  choice:=&choice;

new    7: choice:=1; Enter

value for amount: 500 old

8:       amount:=&amount;

new   8: amount:=500;

PL/SQL procedure successfully completed.

select * from acc;

     ACNO NAME          BALANCE
---------- --------------------        ----------
1              Renju               8500
           2       Tony             23500
           3       Rohini                  5500
```

4 Deena 13500

Trigger

1. Create a Simple Trigger that does not allow Insert Update and Delete Operations on the Table

SQL> connect system/12345678

Connected.

SQL> select * from item;

ITEM_ID	ITEMNAME	PRICE
2334	Geera	206.25
2532	Corn soup	34.65
2124	Lays	20
4531	Set	99.99
2319	Duracell	45.5

SQL> create trigger tr1

2 BEFORE INSERT OR UPDATE OR DELETE ON item FOR EACH ROW

3 begin

4 raise_application_error(-20010,'you are not permitted to do this operation');

5 end;

6 /

Trigger created.

SQL> insert into item values(5555,'Sweets',100); insert

into item values(5555,'Sweets',100)

ERROR at line 1:

ORA-20010: you are not permitted to do this operation

ORA-06512: at "SYSTEM.TR1", line 2

ORA-04088: error during execution of trigger 'SYSTEM.TR1'

2. Create a trigger that displays a message after update, Delete, Insert operations on a table.

SQL> connect system/12345678

Connected.

SQL> desc emp1;

Name	Null?	Type
ID		NUMBER(38)
NAME		VARCHAR2(20)
SALARY		NUMBER(38)

SQL> select * from emp1;

ID	NAME	SALARY
1	maneesha	25000

2 anu 30000

3 aravind 27000

4 manu 20000

```
SQL> create or replace trigger trg
2    after update or insert or delete on emp1
3    for each row
4    begin
5    if updating then
6    dbms_output.put_line('updated');
7    elsif inserting then
8    dbms_output.put_line('insertion done');
9    elsif deleting then
10   dbms_output.put_line('deleted');
11   end if;
12   end;
13   /
```

Trigger created.

SQL> insert into emp1 values(5,'Babu',29000);

insertion done

1 row created.

SQL> select * from emp1;

ID NAME SALARY

---------- -------------------- ----------

1 maneesha 25000

2 anu 30000

3 aravind 27000

4 manu 20000

5 Babu 29000

SQL> delete from emp1 where id=4; deleted

1 row deleted.

SQL> select * from emp1;

ID NAME SALARY

---------- -------------------- ----------

1 maneesha 25000

2 anu 30000

3 aravind 27000

5 Babu 29000

SQL> update emp1 set salary=32000 where id=2; updated

1 row updated.

SQL> select * from emp1;

ID NAME SALARY

---------- -------------------- ----------

1 maneesha 25000

2 anu 32000

3 aravind 27000

5 Babu 29000

3. Create a trigger that gets invoked before insert operation on a table. The trigger should convert employee name to uppercase before its stored in the table.

SQL> connect system/12345678

Connected.

SQL> set serveroutput on;

SQL> create or replace trigger tr2

2 before insert on emp1

3 for each row

4 begin

5 :new.name:=upper(:new.name);

6 end;

7 /

Trigger created.

SQL> select * from emp1;

ID NAME SALARY

---------- -------------------- ----------

1 Raju 25200

2 Babu 15200

3 Drshya 32200

4 John 32878

5 Kalam 48200

SQL> desc emp1;

Name	Null?	Type
ID		NUMBER(38)
NAME		VARCHAR2(20)
SALARY		NUMBER(38)

4. Create a row-level trigger for the customers table that would fire for UPDATE operations performed on the CUSTOMERS table. This trigger should display the salary difference between the old values and new values

```
SQL> create or replace trigger trg1
2  before update on Customers for each row
3  when(new.id > 0)
4  declare
5  sal_difference number;
6  begin
7  sal_difference:=:new.salary-:old.salary;
8  dbms_output.put_line('old salary:'||:old.salary);
9  dbms_output.put_line('new salary:'||:new.salary);
10 dbms_output.put_line('salary difference:'||sal_difference);
```

11 end;

12 /

Trigger created.

SQL> select * from customers;

ID	NAME	AGE	ADDRESS	SALARY
1	Krishna	32	AHMEDABAD	2000
2	Indu	25	DELHI	1500
3	Anandhu	32	MAYSOOR	2000
4	Unni	23	KOTA	2000
5	Amal	25	MUMBAI	6500
6	Manju	27	BHOPAL	8500
7	Raju	22	MP	4500
8	Anju	24	INDORE	1000

8 rows selected**.**

SQL> update Customers set salary=6000 where id=7;

1 row updated.

SQL> select * from Customers;

ID	NAME	AGE	ADDRESS	SALARY
1	Krishna	32	AHMEDABAD	2000
2	Indu	25	DELHI	1500

3	Anandhu	32	MAYSOOR	2000
4	Unni	23	KOTA	2000
5	Amal	25	MUMBAI	6500
6	Manju	27	BHOPAL	8500
7	Raju	22	MP	6000
8	Anju	24	INDORE	1000

Cursors

1. Calculate Interest for Fixed Deposit Amount Using Cursors

sql> create table amount(accno int,years int,amount int,interest int); table created.

sql> insert into amount values(101,2,1000,100); 1 row created.

sql> insert into amount values(102,4,2000,200);

1 row created.

sql> insert into amount values(103,3,3000,300); 1 row created.

sql> insert into amount values(104,4,4000,400); 1 row created.

sql> insert into amount values(105,5,5000,500);

1 row created.

sql> select * from amount;

ACCNO	YEARS	AMOUNT	INTEREST
101	2	1000	100
102	4	2000	200
103	3	3000	300
104	4	4000	400
105	5	5000	500

sql> update amount set interest=0 where years<=4;

4 rows updated.

sql> select * from amount;

ACCNO	YEARS	AMOUNT	INTEREST
101	2	1000	0
102	4	2000	0
103	3	3000	0
104	4	4000	0
105	5	5000	500

```
sql> declare
2    cursor amount is select * from amount;
3    begin
4    for i in amount
5    loop
6    if i.amount<=1000 then
7    update amount set interest=i.amount*1 where accno=i.accno;
```

```
8   elsif i.amount>1000 and i.amount<=5000 then
9   update amount set interest=i.amount*2 where accno=i.accno;
10  else
11  update amount set interest=i.amount*3 where accno=i.accno;
12  end if;
13  end loop;
14  end;
15  /
```

PL/SQL procedure successfully completed.

sql> select * from amount;

ACCNO	YEARS	AMOUNT	INTEREST
101	2	1000	1000
102	4	2000	4000
103	3	3000	6000
104	4	4000	8000
105	5	5000	10000

2. Calculate Electricity Bill Using Cursors

sql> create table ebill(ebno int primary key,name varchar(20),units int,charges float);

table created.

sql> insert into ebill values(1,'Krishna',100,99.9);

1 row created.

sql> insert into ebill values(2,'Indu',200,88.8); 1

row created.

sql> insert into ebillvalues(3,'Ammu',300,77.7);

1 row created.

sql> insert into ebill values(4,'Unni',400,66.6);

1 row created.

sql> select * from ebill;

ebno	name	units	charges
1	Krishna	100	99.9
2	Indu	200	88.8
3	Ammu	300	77.7
4	Unni	400	66.6

```
sql> declare
2    cursor bill is select * from ebill;
3    begin
4    for i in bill
5    loop
6    if i.units<=100 then
7    update ebill set charges=i.units*1 where ebno=i.ebno;
8    elsif i.units>100 and i.units<=400 then
9    update ebill set charges=i.units*2 where ebno=i.ebno;
10   else
11   update ebill set charges=i.units*3 where ebno=i.ebno;
12   end if;
13   end loop;
14   end;
15   /
```

pl/sql procedure successfully completed.

SQL> SELECT * FROM EBILL;

EBNO	NAME	UNITS	CHARGES
1	Krishna	100	100
2	Indu	200	400
3	Ammu	300	600
4	Unni	400	800

cursor-3

write pl/sql code to update values in create tables by using implicit cursors.

sql> set serveroutput on;

sql> desc emp1;

name	null?	type
id		number(38)
name		varchar2(20)
salary		number(38)

```
SQL> select * from emp1;

        ID NAME                 SALARY
---------- -------------------- ----------
1 Unni                  25000
2 Ammu                  15000
3 Krishna               32000
4 Indu                  32678
5 Thumbi                48000

SQL> declare
2    num_rows number(5);
3    begin
4    update emp1 set salary=salary+200;
5    if sql%notfound then
6    dbms_output.put_line('None of the salaries where updated');
7    else if sql%found then
8    num_rows:=sql%rowcount;
9    dbms_output.put_line('Salaries for '||num_rows||''||'
     '||'employees are updated');
10   end if;
11   end if;
12   end;
13   /
Salaries for 5 employees are updated

PL/SQL procedure successfully completed.
```

SQL> select * from emp1;

ID	NAME	SALARY
1	Unni	25200
2	Ammu	15200
3	Krishna	32200
4	Indu	32878
5	Thumbi	48200

3. **Given the table works(emp_id,company_name,salary).write a cursor to select the three highest paid employees from the table.**

SQL> desc works;

Name	Null?	Type
EMP_ID	NOT NULL	CHAR(8)
COMPANY_NAME	NOT NULL	VARCHAR2(18)
SALARY		FLOAT(126)

SQL> select * from works;

EMP_ID COMPANY_NAME SALARY

```
-------- ------------------ ----------
E-101    SBI                71000
E-102    SBI                108900
E-103    SBT                40000
E-104    Federal            37000
SQL> declare
  2  i number:=0;
  3  cursor cur is select emp_id,company_name,salary from works order by
     salary desc;
  4  r cur%rowtype;
  5  begin
  6  open cur;
  7  loop
  8  exit when i=3;
  9  fetch cur into r;
 10  dbms_output.put_line(r.emp_id||' '||r.company_name||' '||r.salary);
 11  i:=i+1;
 12  end loop;
 13  close cur;
 14  end;
 15  /
E-102    SBI 108900
E-101    SBI 71000
E-103    SBT 40000
PL/SQL procedure successfully completed.
```

SQL> select emp_id,company_name,salary from works order by salary desc;

EMP_ID COMPANY_NAME SALARY

-------- ---------- ----------

E-102 SBI 108900

E-101 SBI 71000

E-103 SBT 40000

E-104 Federal 37000

5. Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.

SQL> DESC IAMARKS;

Name	Null?	Type
REGNO	NOT NULL	NUMBER
SCODE		VARCHAR2(20)
TEST1		NUMBER(5)
TEST2		NUMBER(5)
TEST3		NUMBER(5)
FINAL_IAMARKS		NUMBER(5)

SQL> set serveroutput on;

SQL> create table IAmarks(reg_no int primary key,scode varchar(10),Test1 number(10),Test2 number(10),Test3 number(10),Final_IAmarks number(10));

Table created.

SQL> insert into IAmarks values(101,'CLAB112',45,34,12,null); 1

row created.

SQL> insert into IAmarks values(104,'DBMS123',33,22,35,null); 1

row created.

SQL> insert into IAmarks values(345,'Pythn236',44,43,42,null);

1 row created.

SQL> desc IAmarks;

Name	Null?	Type
REG_NO	NOT NULL	NUMBER(38)
SCODE		VARCHAR2(10)
TEST1		NUMBER(10)
TEST2		NUMBER(10)
TEST3		NUMBER(10)
FINAL_IAMARKS		NUMBER(10)

SQL> select * from IAmarks;

REG_NO	SCODE	TEST1	TEST2	TEST3	FINAL_IAMARKS
101	CLAB112	45	34	12	
104	DBMS123	33	22	35	
345	Pythn236	44	43	42	

SQL> create or replace procedure avgmarks is

2 cursor curs is

3 select greatest(Test1,Test2) as A,greatest(Test1,Test3) as B,greatest(Test3,Test2) as C

4 from IAmarks where Final_IAmarks is null for update;

5 C_A number;

```
6       C_B number;
7       C_C number;
8       C_SM number;
9       C_AV number;
10      begin
11      open curs;
12      loop
13      fetch curs into C_A, C_B, C_C;
14      exit when curs%notfound;
15      dbms_output.put_line(C_A || ' ' || C_B || ' ' || C_C);
16      if (C_A != C_B) then
17      C_SM:=C_A+C_B;
18      else
19      C_SM:=C_A+C_C;
20      end if;
21      C_AV:=C_SM/2;
22      update IAmarks set Final_IAmarks=C_AV where current of curs;
23      end loop;
24      close curs;
25      end;
26      /
Procedure created.
SQL> exec avgmarks;
45 45 34
```

33 35 35

44 44 43

PL/SQL procedure successfully completed.

SQL> select * from IAmarks;

REG_NO	SCODE	TEST1	TEST2	TEST3	FINAL_IAMARKS
101	CLAB112	45	34	12	40
104	DBMS123	33	22	35	34
345	Pythn236	44	43	42	44