10) Write PL-SQL program to find sum and average

=> declare

a number := &a;

b number := &b;

c number := &c;

Sm number;

av number;

begin

Sm := a+b+c;

av := Sm/3;

dbms_output.put_line ('Sum = '||Sm);

dbms_output.put_line ('Average = '||av);

end;

/

=> Value of a = 5

" " b = 6

" " c = 7

=> Sum = 18

Average = 6

VII Write a PL-SQL prog. to find Simple interest.

SI: PRT / 100

=> declare
p number (3,2);
r number (3,2);
t number (3,2);
Si number (3,2);
begin
p := &p;
r := &r;
t := &t;
Si = (p*r*t) / 100;
dbms_output.put_line ('Simple Interest = ' || Si);
End;
/

=> Enter value of p: 2
" " r: 3
" " t = 4

Simple Interest = 24

12/06/24

1. Write PL/SQL program to find factorial of a number.

=> Set serveroutput on;

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

Output

Enter value for n: 5

5! = 120

2. write PL/SQL code to find greatest of three numbers.

```
set serveroutput on;
```

=> declare

```
a number := &a;
b number := &b;
c number := &c;
begin
if a>b and a>c then
dbms_output.put_line(a||' is greatest');
else if b>a and b>c then
dbms_output.put_line(b||' is greatest');
else
dbms_output.put_line(c||' is greater');
end if;
end if;
end;
/
```

=> a: 3

b: 4

c: 5

∴ 5 is greater.

3. write PL/SQL code to find reverse of a number

```
Set serveroutput on;
declare
N number;
S number:=0;
R number;
k number;
begin
N:=&N;
k:=N;
loop
exit when N=0;
S:=S*10;
R:=MOD(N,10);
S:=S+R;
N:=TRUNC(N/10);
end loop;
dbms_output.put_line('The Reversed digit
                      of '||k||' is '||S);
end;
/
```

Output

n: 2024

The Reversed digit of 2024: 4202

2. write PL/SQL code to find reverse of a number.

=> Set serveroutput on;
declare
N number;
S Number:=0;
R Number;
k Number;
begin
N:=&N;
k:=N;
loop
exit when N=0;
S:=S*10;
R:=MOD(N,10);
S:=S+R;
N:=TRUNC(N/10);
end loop;
dbms_output.put_line('The Reversed digit of '||k||' is '||S);
end;
/

Output

n: 2024

The Reversed digit of 2024: 4202

6. write a plsql program using case statement

=>
```
Set serveroutput on;
declare
grade char(1) := 'A';
begin
case grade
when 'A' then dbms_output.put_line('Excellent');
when 'B' then dbms_output.put_line('very good');
when 'C' then dbms_output.put_line('good');
when 'D' then dbms_output.put_line('Average');
when 'E' then dbms_output.put_line('passed with
                                    grace');
else dbms_output.put_line('Failed');
end case;
end;
/
```

Output

Excellent

1. write pl-sql program used nested loop to find the prime numbers.

=) Set serveroutput on;
declare
i number(3);
j number(3);
begin
i := 2
loop
j := 2
loop
exit when ((mod(i,j) = 0) or (j = i));
j := j+1;
end loop;
If (j = i) then
dbms_output.put_line(i || 'is prime');
end If;
i := i+1;
exit when i = 50;
end loop;
end;
/

Output: 2 is prime
3 " "
5 " "
7 " "
11 " "
13 " "
17 " "
:

Query.

Before insert triggers

```
set serveroutput on;
Creat or replace
trigger person_insert_before
before
insert
on person
for each row
begin
dbms_output.put_line('before insert of '||:
                                new.name);
End;
/
```

Trigger Created

After insert trigger

```
Set serveroutput on;
Creat or replace
trigger person_insert_after
after
insert
on person
for each row
begin
dbms_output.put_line('after insert of '||:
                      new.name);
End;
/
```

See: insert into person values
(2, 'Smith', '03-mar-34');

Output:

before insert of Smith
after insert of Smith

update trigger statement

```
set serveroutput on;
create or replace
trigger person_update_before
before update
on person
for each row
begin
dbms_output.put_line('before updating
from person');
end;
/
```

Output:

```
Trigger created
update person set dob=sysdate;
before updating from person
'2' rows updated
```

After update triggers Statement

```
Set Serveroutput On;
Creat or replace
trigger person_update_after
after update
 on person
for each row
 begin
dbms_output.put_line ('after updating
 from person');
 end;
/
```

Output

Trigger Created.

Before delete trigger

```
set serveroutput on;
create or replace
trigger person_delete_before
before delete
on person
for each row
begin
dbms_output.put_line('before delete
from person');
end;
/
```

Output:

Triggers created

SQL: delete person
before delete
1 row deleted.

After delete triggers

```
Set serveroutput on;
Creat or replace
trigger person_delete_after
after delete
on person
for each row
begin
dbms_output.put_line('after delete
from person');
end;
/
```

Output.

Triggers Created

SQL> delete person where name = 'Smith' after delete of Smith-