**1\_basic\_output:**

declare

emp\_name varchar2(20);

begin

select ENAME into emp\_name

from EMP

where EMPNO=7698;

dbms\_output.put\_line('Employee name : '||emp\_name);

end;

**2\_variable\_declare:**

*/\*PL-SQL are 4 types:*

*1. Scalar variable = is hold a single value. Ex: varchar2, varchar, date, number etc.*

*2. Composit variable =*

*3. Reference variable =*

*4. Large object variable(LOV) =*

*#############################*

*PL\_SQL er baire SQL er variable gula k Bind variable bola hoi\*/*

declare

myName varchar2(20);

emp\_hiredate date;

emp\_deptNo number(2) not null default 10; */\*default and := is same. both are used to assignment for value\*/*

emp\_commission constant number := 1500; */\*for declare constant variable\*/*

begin

myName := 'Shifullah Ahmed Khan';

dbms\_output.put\_line('My name is : '||myName);

end;

**3\_scaler\_data\_type.sql:**

set serveroutput on;

declare

myName varchar2(20);

emp\_job varchar2(20);

count\_loop binary\_integer := 0;

dept\_total\_sal number(9,2) default 0; */\*default and := is same. both are used to assignment for value\*/*

order\_date date := sysdate;

c\_tax\_rate constant number(3,2):=8.5;

emp\_name EMP.ENAME%type;*/\*this is type attribute\*/*

begin

myName := 'Shifullah Ahmed Khan';

dbms\_output.put\_line('My name is : '||myName);

dbms\_output.put\_line('My date is : '||TO\_CHAR(order\_date, 'YYYY-MM-DD HH:MI:SS'));

end;

**5\_global\_local\_scope\_variable**

set serveroutput on;

<<global\_lebel>>

declare

*--Global Variable*

num1 number default 90;

num2 number := 95; */\*default and := is same. both are used to assignment for value\*/*

*--emp\_name EMP.ENAME%type;/\*this is type attribute\*/*

begin

dbms\_output.put\_line('Global Variable num1 : '||num1);

dbms\_output.put\_line('Global Variable num2 : '||num2);

dbms\_output.put\_line('###################################');

<<local\_lebel\_1>>

declare

*--Local Variable 1*

num1 number default 190;

num2 number := 195;

begin

dbms\_output.put\_line('Local Variable num1 : '||num1);

dbms\_output.put\_line('Local Variable num2 : '||num2);

dbms\_output.put\_line('global\_lebel Variable num1 : '||global\_lebel.num1);

dbms\_output.put\_line('global\_lebel Variable num2 : '||global\_lebel.num2);

dbms\_output.put\_line('###################################');

declare

*--Local Variable 1*

num1 number default 290;

num2 number := 295;

begin

dbms\_output.put\_line('Local Variable num1 : '||num1);

dbms\_output.put\_line('Local Variable num2 : '||num2);

dbms\_output.put\_line('local\_lebel\_1 Variable num1 : '||local\_lebel\_1.num1);

dbms\_output.put\_line('local\_lebel\_1 Variable num2 : '||local\_lebel\_1.num2);

dbms\_output.put\_line('global\_lebel Variable num1 : '||global\_lebel.num1);

dbms\_output.put\_line('global\_lebel Variable num2 : '||global\_lebel.num2);

end;

end;

end;

**6\_add\_sub\_div\_mul\_etc\_operator:**

SET SERVEROUTPUT ON;

DECLARE

num1 NUMBER := 8;

num2 NUMBER := 2;

BEGIN

DBMS\_OUTPUT.put\_line ('num1+num2='||TO\_NUMBER(num1+num2)); *--addition*

dbms\_output.put\_line('num1-num2='||TO\_NUMBER(num1-num2)); *--Substraction*

dbms\_output.put\_line('num1/num2='||TO\_NUMBER(num1/num2)); *--Division*

dbms\_output.put\_line('num1\*num2='||TO\_NUMBER(num1\*num2)); *--Multiplication*

dbms\_output.put\_line('num1^num2='||TO\_NUMBER(num1\*\*num2)); *--Exponential / Power*

END;

**6\_condition:**

SET SERVEROUTPUT ON;

DECLARE

num1 NUMBER := 8;

num2 NUMBER := 2;

num3 NUMBER := 2;

letter varchar2(1) := 'm';

letter2 varchar2(1);

a boolean := true;

b boolean := false;

BEGIN

IF(num1=num2) THEN

DBMS\_OUTPUT.put\_line ('num1 and num2 is equal');

ELSIF(num1!=num2) THEN

DBMS\_OUTPUT.put\_line ('num1 and num2 are not equal');

ELSIF(num3<>num2) THEN

DBMS\_OUTPUT.put\_line ('num1 and num2 are not equal');

ELSIF(num3~=num2) THEN

DBMS\_OUTPUT.put\_line ('num1 and num2 are not equal');

END IF;

*--############################################################*

DBMS\_OUTPUT.put\_line ('##################################');

IF(num1 BETWEEN 2 and 10) THEN

DBMS\_OUTPUT.put\_line ('True');

END IF;

*--############################################################*

DBMS\_OUTPUT.put\_line ('##################################');

IF(letter in ('a','b','m')) THEN

DBMS\_OUTPUT.put\_line ('Letter is found');

ELSE

DBMS\_OUTPUT.put\_line ('Letter is not found ...');

END IF;

*--############################################################*

DBMS\_OUTPUT.put\_line ('##################################');

IF(letter2 is null) THEN

DBMS\_OUTPUT.put\_line ('Letter is Null');

ELSE

DBMS\_OUTPUT.put\_line ('Letter is not Null ...');

END IF;

*--############################################################*

DBMS\_OUTPUT.put\_line ('################ AND ##################');

IF(a and b) THEN

DBMS\_OUTPUT.put\_line ('True');

ELSE

DBMS\_OUTPUT.put\_line ('False');

END IF;

DBMS\_OUTPUT.put\_line ('################ OR ##################');

IF(a or b) THEN

DBMS\_OUTPUT.put\_line ('True');

ELSE

DBMS\_OUTPUT.put\_line ('False');

END IF;

DBMS\_OUTPUT.put\_line ('################ NOT ##################');

IF(not b) THEN

DBMS\_OUTPUT.put\_line ('True');

ELSE

DBMS\_OUTPUT.put\_line ('False');

END IF;

END;

**7\_case\_structure:**

DECLARE

a CHAR (1) := 'b';

BEGIN

CASE a

WHEN 'A'

THEN

DBMS\_OUTPUT.put\_line ('A found.');

WHEN 'B'

THEN

DBMS\_OUTPUT.put\_line ('B found.');

ELSE

DBMS\_OUTPUT.put\_line (a || ' is not found.');

END CASE;

END;

**8\_basic\_Loop\_structure:**

*/\*#### BASIC LOOP #####\*/*

DECLARE

x NUMBER := 10;

BEGIN

DBMS\_OUTPUT.put\_line ('before loop x is : ' || x);

LOOP

DBMS\_OUTPUT.put\_line ('In loop x is : ' || x);

x := x+10;

IF x > 50 THEN

EXIT;

END IF;

END LOOP;

DBMS\_OUTPUT.put\_line ('After loop x is :' || x);

END;

**8\_for\_Loop\_structure:**

DECLARE

x NUMBER;

BEGIN

DBMS\_OUTPUT.put\_line ('before loop x is : ' || x);

FOR x IN 10..15 LOOP

DBMS\_OUTPUT.put\_line ('In loop x is : ' || x);

*--x := x + 10;*

END LOOP;

*--######### Reverse For Loop ##########*

FOR x IN REVERSE 10..15 LOOP

DBMS\_OUTPUT.put\_line ('In loop x is : ' || x);

*--x := x + 10;*

END LOOP;

DBMS\_OUTPUT.put\_line ('After loop x is :' || x);

END;

**8\_Loop\_structure:**

*/\**

*###########PL-SQL loop########*

*3 types:*

*1. Basic loop*

*2. For loop*

*3. While loop*

*\*/*

**8\_nested\_Loop\_structure:**

DECLARE

x NUMBER;

y NUMBER;

BEGIN

DBMS\_OUTPUT.put\_line ('before loop x is : ' || x);

<<outer\_loop>>

FOR x IN 1..3 LOOP

<<inner\_loop>>

FOR y IN 1..3 LOOP

IF y>2 THEN

EXIT outer\_loop;

ELSE

DBMS\_OUTPUT.put\_line ('X is : '||x||' , '||'Y is : '||y);

END IF;

END LOOP;

END LOOP;

DBMS\_OUTPUT.put\_line ('After loop x is :' || x);

END;

**8\_while\_Loop\_structure:**

DECLARE

x NUMBER := 10;

BEGIN

DBMS\_OUTPUT.put\_line ('before loop x is : ' || x);

WHILE x <= 50 LOOP

DBMS\_OUTPUT.put\_line ('In loop x is : ' || x);

x := x + 10;

END LOOP;

DBMS\_OUTPUT.put\_line ('After loop x is :' || x);

END;

**10\_array:**

DECLARE

TYPE name\_array IS VARRAY (5) OF VARCHAR2 (20); *-- type attribute declared*

TYPE grade\_array IS VARRAY (5) OF INTEGER; *-- type attribute declared*

names name\_array;

marks grade\_array;

total NUMBER;

BEGIN

names :=

name\_array ('Shown',

'Shifullah',

'Papon',

'Ratul',

'Seium');

marks :=

grade\_array (95,

90,

82,

54,

72);

total := names.COUNT;

DBMS\_OUTPUT.put\_line ('Total number of students : ' || total);

FOR i IN 1 .. total

LOOP

DBMS\_OUTPUT.put\_line (

'Student ' || names (i) || ' Get ' || marks (i) || ' marks');

END LOOP;

END;

**11\_Non\_stand\_alone\_procedure:**

declare

a number;

b number;

c number;

PROCEDURE find\_minimum(x in number, y in number, z out number) IS *-- AS = Non stand alone procedure*

BEGIN

if x<y then

z:=x;

else

z:=y;

end if;

END;

begin

a:=23;

b:=45;

find\_minimum(a,b,c);

dbms\_output.put\_line('minimum valoue of '||a||' and '||b||' is '||c);

end;

**11\_square\_return\_procedure:**

DECLARE

a NUMBER;

output\_text VARCHAR2 (100);

PROCEDURE find\_minimum (x IN OUT NUMBER)

IS *-- AS = Non stand alone procedure*

BEGIN

x := x \* x;

END;

BEGIN

a := 11;

output\_text := 'Square value of ' || TO\_CHAR (a) || ' is : ';

find\_minimum (x => a);

DBMS\_OUTPUT.put\_line (output\_text || a);

END;

**11\_stand\_alone\_procedure:**

CREATE OR REPLACE PROCEDURE greetings AS *-- AS = stand alone procedure*

BEGIN

dbms\_output.put\_line('Hello world');

END;

**11\_call\_stand\_alone\_procedure.sql:**

begin

greetings;

end;

**12\_call\_function:**

declare

a number;

b number;

call\_func number;

BEGIN

a := 50;

b := 100;

call\_func := find\_maximum(a,y=>b);

dbms\_output.put\_line('Max number is : '||call\_func);

END;

**12\_find\_max\_function\_with\_param:**

create or replace function find\_maximum(x in number, y in number)

return number

IS

max\_num number := 0; *-- AS = Non stand alone procedure*

BEGIN

if x>y then

max\_num := x;

else

max\_num := y;

end if;

return max\_num;

END;

**12\_function:**

create or replace function total\_employees

return number

IS

total\_number number := 0; *-- AS = Non stand alone procedure*

BEGIN

select count(EMPNO) into total\_number

from emp;

return total\_number;

END;

**13\_Explicit\_Courser\_rowcount:**

declare

emp\_id emp.empno%type;

emp\_name EMP.ENAME%type;

cursor my\_employees is

select empno, ename

from emp;

begin

open my\_employees;

loop

fetch my\_employees into emp\_id, emp\_name;

if my\_employees%found then

dbms\_output.put\_line('ID: '||emp\_id||' Name: '||emp\_name);

elsif my\_employees%notfound then

exit;

end if;

end loop;

close my\_employees;

end;

**13\_Explicit\_Courser\_singleRowFetch:**

declare

emp\_id emp.empno%type;

emp\_name EMP.ENAME%type;

emp\_position EMP.JOB%type;

cursor my\_employees is

select empno, ename, job

from emp

where empno=7839;

begin

open my\_employees;

*--loop*

fetch my\_employees into emp\_id, emp\_name, emp\_position;

if my\_employees%found then

dbms\_output.put\_line('ID: '||emp\_id||' Name: '||emp\_name||' Designation: '||emp\_position);

elsif my\_employees%notfound then

dbms\_output.put\_line('Data not found...');

end if;

*--end loop;*

close my\_employees;

end;

**13\_Explicit\_Courser\_singleRowFetch\_IsOpen:**

declare

emp\_id emp.empno%type;

emp\_name EMP.ENAME%type;

emp\_position EMP.JOB%type;

cursor my\_employees is

select empno, ename, job

from emp

where empno=7839;

begin

if my\_employees%isopen = False then

open my\_employees;

end if;

fetch my\_employees into emp\_id, emp\_name, emp\_position;

if my\_employees%found then

dbms\_output.put\_line('ID: '||emp\_id||' Name: '||emp\_name||' Designation: '||emp\_position);

elsif my\_employees%notfound then

dbms\_output.put\_line('Data not found...');

end if;

close my\_employees;

end;

**13\_implicit\_Courser\_rowcount:**

declare

total\_rows number;

begin

update emp

set ename = 'Blake'

where empno =7698;

if sql%notfound then

dbms\_output.put\_line('No data found...');

elsif sql%found then

total\_rows := sql%rowcount;

dbms\_output.put\_line(total\_rows||' rows updated');

end if;

end;

**14\_custom\_exception\_hanelling:**

declare

emp\_id emp.empno%type :=7839;

emp\_name EMP.ENAME%type;

invalid\_id exception;

begin

if emp\_id<0 then

raise invalid\_id;

else

select empno, ename into emp\_id, emp\_name

from emp

where empno=emp\_id;

dbms\_output.put\_line('ID: '||emp\_id||' Name: '||emp\_name);

end if;

Exception

when invalid\_id then

dbms\_output.put\_line('Id shouldn''t be less than 0');

when no\_data\_found then

dbms\_output.put\_line('Such employee not found...');

end;

**14\_exception\_hanelling:**

declare

emp\_id emp.empno%type;

emp\_name EMP.ENAME%type;

begin

select empno, ename into emp\_id, emp\_name

from emp

where empno=7839;

dbms\_output.put\_line('ID: '||emp\_id||' Name: '||emp\_name);

Exception

when no\_data\_found then

dbms\_output.put\_line('Such employee not found...');

end;

**15\_update\_or\_delete\_trigger:**

create or replace trigger "FEES\_COLLECTION\_T1"

BEFORE

update or delete on "FEES\_COLLECTION"

REFERENCING NEW AS NEW OLD AS OLD

for each row

when (NEW.COLLECTION\_ID=OLD.COLLECTION\_ID)

begin

insert into ADMIN.FEES\_COLLECTION\_LOG

values(

:OLD.COLLECTION\_ID,

:OLD.STUDENT\_ID,

:OLD.FEES\_TYPE,

:OLD.MONTH\_CODE,

:OLD.FEE\_AMOUNT,

:OLD.REMARKS,

:NEW.STUDENT\_ID,

:NEW.FEES\_TYPE,

:NEW.MONTH\_CODE,

:NEW.FEE\_AMOUNT,

:NEW.REMARKS

);

end;

**create\_schema\_or\_user:**

*--CREATE USER smith IDENTIFIED BY password*

*--CREATE USER SHIFULLAH IDENTIFIED BY SHIFULLAH;*

**CUSTOMER\_LOG:**

create or replace trigger customer\_log

before delete or update on SHIFULLAH.MY\_CUSTOMER

REFERENCING NEW AS NEW OLD AS OLD

for each row

when(new.id=old.id)

begin

insert into SHIFULLAH.MY\_CUSTOMER\_LOG

values(

:old.id,

:old.name,

:old.salary,

:new.salary

);

end;

**delete\_duplicate\_data:**

*delete usermaster*

*where rowid not in(select max(rowid) from usermaster group by username)*

*select max(rowid) from SHIFULLAH.STUDENT\_INFO group by ID*

*delete SHIFULLAH.STUDENT\_INFO*

*where rowid not in(select max(rowid) from SHIFULLAH.STUDENT\_INFO group by ID)*

**update\_table:**

update shifullah.MY\_CUSTOMER

set salary = 25000

where id = 2;

commit WORK;

**dynamic\_menu\_create:**

select level,

PROGDESC label,

'f?p=&APP\_ID.:'||PAGEID||':'||:APP\_SESSION target,

'NO' is\_current,

'#APP\_IMAGES#del.gif' image

from MENU

start with PARENTCODE is null

connect by prior PROGCODE = PARENTCODE

order siblings by PROGDESC