PL/SQL LAB:

1. WAP to print any statement.

```
set serverput on
begin
dbms_output.put_line('======================================');
dbms_output.put_line('Welcome to PL/sql programming');
dbms_output.put_line('======================================');
```

2. WAP to enter any two numbers and find out their sum, difference, product, quotient and remainder.

```
Set serveroutput on
Declare
no1 number(5);
no2 number(5);
s number(5);
d number(5);
p number(5);
q number(5);
rem number(5);
begin
no1:=&number1;
no2:=&number2;
S:=no1+no2;
d:=no1-no2;
p:=no1*no2;
q:=no1/no2;
rem:=mod(no1,no2);
dbms_output.put_line('Sum='||s);
dbms_output.put_line('Difference='||d);
dbms_output.put_line('Product='||p);
dbms_output.put_line('Quotient='||q);
dbms_output.put_line('Remainder='||rem);
end;
```

3. WAP to find out the greatest of any two numbers

```
set serveroutput on
declare
no1 number(5);
no2 number(5);
begin
no1:=&number(5);
no2:=&number(5);
if no1>=no2 then
dbms_output.put_line(no1||' is greatest');
else
dbms_output.put_line(no2||' is greatest');
end if;
end;
```

4. WAP to enter any number and find out whether it's positive or negative or zero

```
set serveroutput on
declare
num number(5);
begin
num:=&number;
if num>0 then
dbms_output.put_line(num||' is positive');
else if num<0 then
dbms_output.put_line(num ||' is negative');
else
dbms_output.put_line(num ||' is equal to zero');
end if;
end;
```

5. WAP to accept the monthly salary of any employee and find the bonus of 12% on annual salary if experience is more than 3 years and otherwise the bonus is Rs. 1000. After all calculate the total salary received by the employ on that month along with the bonus amount.

```
Declare
Msal number(7,2):=&msal;
Annsal number(9,2);
Bonus number(7,2);
doj date:='&date_of_join';
Exp number(3);
Totsal number(9,2);
Begin
Exp:=months_between(sysdate,doj)/12;
Annsal:=msal*12;
If exp>3 then
Bonus:=annsal*12/100;
Else
Bonus:=1000;
End if;
Totsal:=msal+bonus;
Dbms_output.put_line('Annual salary=Rs. '||annsal);
Dbms_output.put_line('Experience='||exp||' years');
Dbms_output.put_line('Bonus amount=Rs. '||bonus);
Dbms_output.put_line('Total salary drawn=Rs. '||totsal);
End;
```

6. WAP to accept any number and check whether that is a multiple of only 3 or only 5 or both 3 and 5 or none of them.

```
Declare
No number(4):=&no;
Begin
If mod(no,3)=0 then
 If mod(no,5)=0 then
```

```
Dbms_output.put_line(no||' is multiple of both 3 and 5');
Else
Dbms_output.put_line(no||' is multiple of only 3');
End if;
Else
If mod(no,5)=0 then
Dbms_output.put_line(no||' is multiple of only 5');
Else
Dbms_output.put_line(no||' is multiple of none of 3 and 5);
End if;
End if;
End;
```

7.
```
Declare
r number(3);
s1 number(2);
s2 number(2);
s3 number(2);
t number(3);
a number(5,2);
re varchar2(6);
d varchar2(8);
begin
r:=&no;
select sub1,sub2,sub3 into s1,s2,s3 from stud where sno=r;
t:=s1+s2+s3;
a:=t/3;
if s1>=35 and s2>=35 and s3>=35 then
re:='pass';
else
re:='fail';
end if;
if re='pass' and a>=60 then
d:='first';
else if re='pass' and a>=50 and a<60 then
d:='second';
else if re:='pass'  and a>=35 and a<50 then
d:='third';
else
d:='nill';
end if;
update stud set tot=t,avr=a,res=re,div=d where sno=r;
end;
```

8. WAP to input any number and check whether it's even or odd.

```
set serveroutput on
declare
num number(4);
```

```
begin
num:=&number;
if mod(num,2)=0 then
dbms_output.put_line(num||' is even');
else
dbms_output.put_line(num||' is odd);
end if;
end;
```

9. WAP to find out the greatest of any three numbers.

```
set serveroutput on
declare
no1 number(4);
no2 number(4);
no3 number(4);
begin
no1:=&number1;
no2:=&number2;
no3:=&number3;
if no1>no2 and no1>no3 then
dbms_output.put_line(no1||' is the greatest');
elsif no>no3 and no2>no1 then
dbms_output.put_line(no2||' is the greatest');
else
dbms_output.put_line(no3||' is the greatest');
end if;
end;
```

10. WAP to enter any number and check whether it's a multiple of 3 or 7 or both 3 and 7 or not of 3 or 7.

```
set serveroutput on
declare
num number(5);
begin
num:=&number;
if mod(num,3)=0 and mod(num,7)=0 then
dbms_output.put_line(num|| ' is multiple of both 3 and 7');
elsif mod(num,3)=0 then
dbms_output.put_line(num|| 'is multiple of only 3');
elsif mod(num,7)=0 then
dbms_output.put_line(num|| ' is multiple of only 7);
else
dbms_output.put_line(num|| ' is multiple of 3 nor of 7);
end if;
end;
```

11. WAP to enter any alphabet and check it whether it's a consonant or a vowel.

```
Declare
Ch  varchar2(1);
Begin
Ch:='&char';
If ch='a' or ch='e' or ch='I' or ch='o' or ch='u' or ch='A' or ch='E' or ch='I' or ch='O' or ch='U' then
dbms_output.put_line(num|| ' is a vowel);
else
dbms_output.put_line(num|| ' is a consonant);
end if;
end;
```

12. WAP to findout the factorial of any number.

```
declare
n number(3);
x number(2):=1;
f number(5):=1;
begin
n:=&number;
loop
f:=f*x;
x:=x+1;
exit when x>n;
end loop;
dbms_output.put_line('Factorial of' ||n||' ' is='||f);
end;
```

Cursors:-

13. WAP to delete the data for employees after entering job.

```
declare
no number(3);
begin
delete from employ where job='&job';
if sql%notfound then
dbms_output.put_line('There is no employee doing the job');
elsif sql%found then
dbms_output.put_line('Some data has been retrieved by the cursor');
enf if;
no:=sql%rowcount;
dbms_output.put_line('No of records deleted='||no);
end;
```

14. WAP to increase the sal of employees by 1000 rs. For deptno=10.

```
set serveroutput on
declare
x number(4);
```

```
begin
update emp set sal=sal+1000 where deptno=&deptno;
x:=sql%rowcount;
dbms_output.put_line(x||' no. of row updated');
end;

/
Set serveroutput on
declare
x number(4);
begin
delete from emp where job='&job';
x:=sql%rowcount;
dbms_output.put_line(x||' no. of row updated');
end;

/

Set serveroutput on
Declare
X number(4);
Begin
Update emp set sal=sal+1000 where deptno=&dno;
If sql%found then
dbms_output.put_line('some rows are retrieved by the query');
elsif sql%notfound then
dbms_output.put_line('query is not able to retrieve any row');
end if;
x:=sql%rowcount;
dbms_output.put_line(x||' no. of row updated);
end;

/
```

15. WAP to increase the salary for the employees of a particular department and enter the no of records updated,date,time,dept no and name of the person who increased the salary into another table called cursor_ret.

```
Create table cursor_ret(mess varchar2(30),update_date date, time varchar2(10),deptno
number(3),name varchar2(10));

Set serveroutput on
Declare
X number(4);
M varchar2(30);
T char(6);
D number(3);
Name varchar2(10) not null:=' ';
Begin
D:=&DNO;
```

```
Name:='&your_name';
Update employ set sal=sal+1000 where dno=D;
If sql%found then
dbms_output.put_line('some rows are retrieved by the query');
if sql%notfound then
dbms_output.put_line('query is  not able to retrieve any row');
end if;
x:=sql%rowcount;
dbms_output.put_line(x||' no. of rows updated');
m:=concat(to_char(x),' no of rows updated on');
t:=to_char(sysdate,'hh24:mi');
insert into cursor_ret values(m,to_date(sysdate),t,D,name);
end;
```

16. WAP to insert the datas of theclerks from emp table to another table.
Clerk table:-

```
Create table cl_table(empno number(4),ename varchar2(10),sal number(8,2),job
varchar2(10),deptno number(3));
Declare
eno number(4);
name varchar2(10);
salary number(8,2);
dno number(3);
j varchar2(10);
cursor c1 is select empno,ename,sal,job,deptno from emp where job='principal';
begin
open c1;
loop
fetch c1 into eno,name,salary,j,dno;
exit when c1%notfound;
insert into cl_table values(eno,name,salary,j,dno);
end loop;
close c1;
end;

/
```

NOTE: DECLARE AND BEGIN ARE TWO through which SQL knows it is PL/SQL statements to be executed.

1. Write a PL/SQL block. Take the salary of an employee into a variable and check if his or her salary is less than 3000. If ist is less than 3000 then update the EMP table with 3000.
   Accept the empno from the user. USE a substitution variable.(&)
2. USE a BIND variable. Write a PL/SQL block to write the deptno, into a table called RESULTS, which has been selected with a condition say of department name is 'ACCOUNTING' . if you don't have a table called RESULTS create one single column as number datatype.

NOTE: BIND variables are declared in SQL*Plus. and are referenced within a PL/SQL block as a HOST variable(with preceding COLON).(:).

3. Write a PL?SQL block. Declare a variable empno and assign it a value of 7788.(Ensure that empno = 7788 exists in your EMP table.). Update the EMP table with sal = 9000 where empno = empno.

Note : See the result the ROLLBACK else all the date will become irrelevant for next practical classes. Why is it happening so?

4. Write a PL/SQL block. Declare a variable v_empno and assign it a value of 7788.(Ensure that empno = 7788 exists in your EMP table.). Update the EMP table with sal = 9000 where empno = v_empno.

Warning : Be careful with the variable names.

IMPLICT CURSOR.

5. Write a PL/SQL block to delete all employees belonging to department 50. And check the number of rows that have been deleted. USE the SQL%ROWROWCOUNT attribute. Insert the value into a table del_history and see the result. Try the same block with department 20.

DO NOT FORGET TO ROLLBACK. if you don't have a table called del_history create one.

NOTE : SQL%ROWCOUNT returns the number of rows processed by the SQL statement.(an integer value).

6. Write a PL/SQL block to delete all employees belonging to department 50. And check if atleast one row has been processed. USE the SQL %found attribute. Insert the value into a table dels_history and see the result. Try the same block with department 20.

NOTE : SQL%FOUND returns TRUE if atleast one row was processed. Otherwise FALSE.(a BOOLEAN value).

7. Write a PL/SQL block to delete all employees belonging to department 50. And check if any rows have been processed. USE the SQL%NOTFOUND attribute. Insert the value into a table dels_history and see the result. Try the same block with department 20.

NOTE : SQL%FOUND returns TRUE if no rows were processed. Otherwise FALSE. (a BOOLEAN value).

If you have a time : For Q6 and Q7 try another logic of inserting the Boolean values.

8. Write a PL/SQL block to delete all employees whose job = 'CLERK' and update the EMP file with job = 'CAPTAIN' and sal = 8000. And check if any rows have been processed. USE the SQL%NOTFOUND attribute. If it is TRUE then ROLLBACK if it is FALSE then commit. Insert the value into table dels_history and see the result.

INTO CLAUSE

9. In a PL/SQL block, Declare two variables(specify the datatypes_ and select data from a table into the variables, change the values and through the PL/SQL block insert the values in a table.

%TYPE

10. In a PL/SQL block , Declare two variables , datatypes should be same as that in the table, and select data from a table into variables , change the values and through a PL/SQL block insert the values in a table. USE%TYPE.

SAVE POINT

11. Write a PL/SQL block to insert values into a file. When there is a duplication of values on index rolls back to a SAVE POINT that has been declared earlier in else it writes into a table error message.
NOTE : Insert values into a file that has various values .Make a unique Index file and then Insert a value that is existing in a file. An error araises.
NOTE : 1. INDEX files speed up the retrieval.
2. When there is a where clause then ORACLE uses index file
if existing otherwise it uses its own convenient and easy path analysing the EXPLAIN PLAN
Create the index file.
DROP the index file.

**********PROCEDURES**********

1. Write a Procedure debit_account that accepts a account number and amount to update the account.
2. Write a procedure to create a department. The procedure accepts two parameters dept no, deptname
3. Write a Procedure to raise salary of an employee. The procedure accepts two parameters employee code, amount increased.

**********FUNCTIONS**********

4. Write a function that checks if salary is in range or not. The procedure accepts two parameters employeeno , designation. Call it from a procedure that accepts only employee code. You have to write the procedure also.
NOTE: PROCEDURES AND FUNCTIONS are stored in the executable code and not source code.
5. Write a function that returns the balance of an account. The function accepts only one parameter account no. call it from procedure that accepts only one account no. you have to write the procedure also.

**********out parameter**********

6. Write a procedure that calculates the bonus of an employee. It accepts the employee number and bonus value as parameters. This procedure should return value i.e. the latest bonus i.e. the last bonus+ the bonus value sent. Call this procedure from another procedure where you should send only the employee code.
USE AN OUT PARAMETER.

**********In out parameter**********

7. Write a procedure that updates the account when an amount is withdrawn i.e. balance-the amount withdrawn. This procedure should return the balance amount. The procedure accepts two parameters account no, amount. Call this procedure from another where you should send the account code and the amount to debit.