ASSIGNMENT 4:

Q1).

declare roll\_number number(5);

att number(4);

begin roll\_number := &roll\_number;

select attendance into att from student where roll\_no=roll\_number;

if att>=75 then

dbms\_output.put\_line('Term granted');

update student set status='ND' where roll\_no=roll\_number;

else

dbms\_output.put\_line('Term not granted');

update student set status='D' where roll\_no=roll\_number;

end if;

exception when NO\_DATA\_FOUND then

dbms\_output.put\_line('Exception:data not found');

end;

Output:

Enter value for roll\_number: 147 old 5: roll\_number := &roll\_number;

new 5: roll\_number := 147;

Term granted PL/SQL

procedure successfully completed.

SQL> /

Enter value for roll\_number: 150 old 5: roll\_number := &roll\_number;

new 5: roll\_number := 150;

Term not granted PL/SQL

procedure successfully completed.

SQL> select \*from student;

ROLL\_NO ATTENDANCE STATUS

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

147 76 ND

150 60 D

159 72

140 65

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

Q2).

declare transaction\_amount number;

account\_no int;

curr\_balance number;

choice varchar(2);

deposit number;

TRANS\_EXP Exception;

begin choice := '&choice';

account\_no := &account\_no;

transaction\_amount :=&transaction\_amount;

select acc\_balance into curr\_balance from acc where acc\_no=account\_no;

if choice='W' then

if transaction\_amount > curr\_balance then

raise TRANS\_EXP;

else

curr\_balance := curr\_balance -transaction\_amount;

update acc set acc\_balance=curr\_balance where acc\_no=account\_no;

end if;

elsif choice='D' then

curr\_balance := curr\_balance + transaction\_amount;

update acc set acc\_balance=curr\_balance where acc\_no=account\_no;

end if;

exception when TRANS\_EXP then

dbms\_output.put\_line('Insufficient balance');

end;

Output:

Enter value for choice: W old 9: choice := '&choice';

new 9:choice := 'W';

Enter value for account\_no: 111 old 10: account\_no :=&account\_no;

new 10: account\_no := 111;

Enter value for transaction\_amount: 10000 old 11: transaction\_amount := &transaction\_amount;

new 11: transaction\_amount := 10000;

Insufficient balance PL/SQL procedure successfully completed.

SQL> /

Enter value for choice: W old 9: choice := '&choice';

new 9: choice := 'W';

Enter value for account\_no: 222 old 10: account\_no := &account\_no;

new 10: account\_no := 222;

Enter value for transaction\_amount: 500 old 11: transaction\_amount := &transaction\_amount;

new 11: transaction\_amount := 500;

PL/SQL procedure successfully completed.

SQL> /

Enter value for choice: D old 9: choice := '&choice';

new 9: choice := 'D';

Enter value for account\_no: 777 old 10: account\_no := &account\_no;

new 10: account\_no := 777;

Enter value for transaction\_amount: 10000 old 11: transaction\_amount := &transaction\_amount;

new 11: transaction\_amount := 10000;

PL/SQL procedure successfully completed.

SQL> select \*from acc;

ACC\_NO ACC\_BALANCE

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

111 5500

222 1500

777 12500

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

Q3).

declare account\_no

number;

bal number;

LESS\_THAN\_ZERO Exception;

begin account\_no := &account\_no;

select balance into bal from client where acc\_no=account\_no;

if bal<0 then

raise LESS\_THAN\_ZERO;

else

dbms\_output.put\_line('everything ok!');

end if;

exception when LESS\_THAN\_ZERO then

dbms\_output.put\_line('balance is negative');

end;

Output:

Enter value for account\_no: 3 old 6: account\_no := &account\_no;

new 6: account\_no := 3;

balance is negative PL/SQL procedure

successfully completed.

SQL> /

Enter value for account\_no: 1 old 6: account\_no := &account\_no;

new 6: account\_no := 1;

everything ok!

PL/SQL procedure successfully completed.

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

Q4).

declare no\_of\_days number;

roll\_number number;

issue\_date DATE;

fine\_amount number;

no\_fine Exception;

begin fine\_amount :=0;

roll\_number := &roll\_number;

select date\_of\_issue into issue\_date from borrower where roll\_no = roll\_number;

no\_of\_days := to\_date(sysdate) - to\_date(issue\_date);

if no\_of\_days < 15 then

raise no\_fine;

elsif no\_of\_days>=15 and no\_of\_days<30 then

fine\_amount := (no\_of\_days-15)\*5;

else

fine\_amount := 75 + (no\_of\_days-30)\*50;

end if;

update borrower set status='R' where roll\_no = roll\_number;

insert into fine values(roll\_number,sysdate,fine\_amount);

exception when NO\_DATA\_FOUND then

dbms\_output.put\_line('Roll number not present!');

when no\_fine then

dbms\_output.put\_line('No fine applicable!');

end;