

Assignment-8

i. Implement a PL/SQL block that will accept student id number from the user, and check if the student attendance is less than 80% then display message that student cannot appear in exam. [Table: STUDENT (STUD_ID, primary key, STUD_NAME, STUD_ATT)].

```
CREATE TABLE STUDENT(  
  
    STUD_ID VARCHAR2(10) PRIMARY KEY,  
  
    STUD_NAME VARCHAR2(20) NOT NULL,  
  
    STUD_ATT NUMBER NOT NULL  
  
);
```

```
INSERT INTO STUDENT VALUES('1','ARKA',90);
```

```
INSERT INTO STUDENT VALUES('2','RAM',80);
```

```
INSERT INTO STUDENT VALUES('3','SHYAM',70);
```

```
INSERT INTO STUDENT VALUES('4','JADU',60);
```

```
INSERT INTO STUDENT VALUES('5','MOHIT',75);
```

```
SQL> CREATE TABLE STUDENT(  
2     STUD_ID VARCHAR2(10) PRIMARY KEY,  
3     STUD_NAME VARCHAR2(20) NOT NULL,  
4     STUD_ATT NUMBER NOT NULL  
5 );
```

Table created.

```
SQL> DESC STUDENT;
```

Name	Null?	Type
STUD_ID	NOT NULL	VARCHAR2(10)
STUD_NAME	NOT NULL	VARCHAR2(20)
STUD_ATT	NOT NULL	NUMBER

```
SQL> INSERT ALL
```

```
2 INTO STUDENT VALUES('1','ARKA',90)  
3 INTO STUDENT VALUES('2','RAM',80)  
4 INTO STUDENT VALUES('3','SHYAM',70)  
5 INTO STUDENT VALUES('4','JADU',60)  
6 INTO STUDENT VALUES('5','MOHIT',75)  
7 SELECT * FROM DUAL;
```

5 rows created.

```
SQL> SELECT * FROM STUDENT;
```

STUD_ID	STUD_NAME	STUD_ATT
1	ARKA	90
2	RAM	80
3	SHYAM	70
4	JADU	60
5	MOHIT	75

```
SQL> █
```

```

SET SERVEROUTPUT ON;
DECLARE
    USERINPUT STUDENT.STUD_ID%TYPE;
    RESULT NUMBER;
BEGIN
    USERINPUT := '&SID';
    SELECT STUD_ATT INTO RESULT FROM STUDENT WHERE STUD_ID =
USERINPUT;
    IF RESULT < 80 THEN DBMS_OUTPUT.PUT_LINE('STUDENT CANNOT GIVE
EXAM');
    ELSE DBMS_OUTPUT.PUT_LINE('STUDENT CAN GIVE EXAM');
    END IF;
END;
/

```

```

SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2     USERINPUT STUDENT.STUD_ID%TYPE;
  3     RESULT NUMBER;
  4 BEGIN
  5     --ACCEPT USERINPUT PROMPT 'ENTER STUDENT ID : ';
  6     USERINPUT := &USERINPUT;
  7     SELECT STUD_ATT INTO RESULT FROM STUDENT WHERE STUD_ID = USERINPUT;
  8
  9     IF RESULT < 80 THEN DBMS_OUTPUT.PUT_LINE('STUDENT CANNOT GIVE EXAM');
 10     ELSE DBMS_OUTPUT.PUT_LINE('STUDENT CAN GIVE EXAM');
 11     END IF;
 12
 13     EXCEPTION WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('NO STUDENT FOUND WITH THE ID');
 14 END;
 15 /
Enter value for userinput: 3
old 6:      USERINPUT := &USERINPUT;
new 6:      USERINPUT := 3;
STUDENT CANNOT GIVE EXAM

PL/SQL procedure successfully completed.

SQL> █

```

ii. Implement a PL/SQL code block that will accept an account number from the user. Check if the user's balance is less than the minimum balance, only then deduct Rs.100 from the balance. The process is fired on the ACCT_MSTR table. [Table: ACCT_MSTR (ACCT_NO, ACCT_HOLDR_NAME, CURBAL)].

```

CREATE TABLE ACCT_MSTR(
    ACCT_NO VARCHAR2(15) PRIMARY KEY,
    ACCT_HOLDER_NAME VARCHAR2(20) NOT NULL,
    CURBAL NUMBER
);

```

```

INSERT ALL
INTO ACCT_MSTR VALUES('123456','ARKA',1000)
INTO ACCT_MSTR VALUES('234567','RAMU',2000)
INTO ACCT_MSTR VALUES('345678','SHYAM',2400)
SELECT * FROM DUAL;

```

```
SQL> CREATE TABLE ACCT_MSTR(
  2     ACCT_NO VARCHAR2(15) PRIMARY KEY,
  3     ACCT_HOLDER_NAME VARCHAR2(20) NOT NULL,
  4     CURBAL NUMBER
  5 );
```

Table created.

```
SQL> DESC ACCT_MSTR;
```

Name	Null?	Type
ACCT_NO	NOT NULL	VARCHAR2(15)
ACCT_HOLDER_NAME	NOT NULL	VARCHAR2(20)
CURBAL		NUMBER

```
SQL> INSERT ALL
```

```
  2 INTO ACCT_MSTR VALUES('123456','ARKA',1000)
  3 INTO ACCT_MSTR VALUES('234567','RAMU',2000)
  4 INTO ACCT_MSTR VALUES('345678','SHYAM',2400)
  5 SELECT * FROM DUAL;
```

3 rows created.

```
SQL> SELECT * FROM ACCT_MSTR;
```

ACCT_NO	ACCT_HOLDER_NAME	CURBAL
123456	ARKA	1000
234567	RAMU	2000
345678	SHYAM	2400

```
SQL> █
```

```
SET SERVEROUTPUT ON;
```

```
DECLARE
```

```
    ACCNO ACCT_MSTR.ACCT_NO%TYPE;
```

```
    BALANCE NUMBER;
```

```
    MINBAL CONSTANT NUMBER := 1500;
```

```
BEGIN
```

```
    ACCNO := '&ACCOUNT_NUMBER';
```

```
    DBMS_OUTPUT.PUT_LINE('MINIMUM BALANCE IS ' || MINBAL);
```

```
    SELECT CURBAL INTO BALANCE FROM ACCT_MSTR WHERE ACCT_NO =
```

```
ACCNO;
```

```
    IF BALANCE < MINBAL THEN
```

```
        DBMS_OUTPUT.PUT_LINE('BALANCE LESS THAN ' || MINBAL);
```

```
        UPDATE ACCT_MSTR SET CURBAL = CURBAL - 100 WHERE
```

```
ACCT_NO = ACCNO;
```

```
    ELSE DBMS_OUTPUT.PUT_LINE('BALANCE MORE THAN ' || MINBAL);
```

```
    END IF;
```

```
    EXCEPTION WHEN NO_DATA_FOUND THEN
```

```
DBMS_OUTPUT.PUT_LINE('INVALID ACCOUNT NUMBER');
```

```
END;
```

```
/
```

```

SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2     ACCNO ACCT_MSTR.ACCT_NO%TYPE;
  3     BALANCE NUMBER;
  4     MINBAL CONSTANT NUMBER := 1500;
  5 BEGIN
  6     ACCNO := '&ACCOUNT_NUMBER';
  7     DBMS_OUTPUT.PUT_LINE('MINIMUM BALANCE IS ' || MINBAL);
  8     SELECT CURBAL INTO BALANCE FROM ACCT_MSTR WHERE ACCT_NO = ACCNO;
  9     IF BALANCE < MINBAL THEN
10         DBMS_OUTPUT.PUT_LINE('BALANCE LESS THAN ' || MINBAL);
11         UPDATE ACCT_MSTR SET CURBAL = CURBAL - 100 WHERE ACCT_NO = ACCNO;
12     ELSE DBMS_OUTPUT.PUT_LINE('BALANCE MORE THAN ' || MINBAL);
13     END IF;
14
15     EXCEPTION WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('INVALID ACCOUNT NUMBER');
16 END;
17 /
Enter value for account_number: 123456
old 6:      ACCNO := '&ACCOUNT_NUMBER';
new 6:      ACCNO := '123456';
MINIMUM BALANCE IS 1500
BALANCE LESS THAN 1500

PL/SQL procedure successfully completed.

SQL> SELECT * FROM ACCT_MSTR;

ACCT_NO          ACCT_HOLDER_NAME          CURBAL
-----
123456           ARKA                      900
234567           RAMU                     2000
345678           SHYAM                    2400

SQL> █

```

- iii. Implement a PL/SQL code block to calculate the area of a circle for a value of radius varying from 3 to 7. Store the radius and the corresponding values of calculated area in an empty table named AREAS, consisting of two columns Radius and Area.
[Table: AREAS (RADIUS, AREA)].

```

CREATE TABLE AREAS(
    RADIUS NUMBER,
    AREA NUMBER
);

```

```

SQL> CREATE TABLE AREAS(
  2     RADIUS NUMBER,
  3     AREA NUMBER
  4 );

Table created.

SQL> DESC AREAS;

Name          Null?    Type
-----
RADIUS        NUMBER
AREA          NUMBER

SQL> SELECT * FROM AREAS;

no rows selected

SQL> █

```

```

CREATE OR REPLACE PROCEDURE FIND_AREA(RAD NUMBER)
AS
RADIUS NUMBER;
AREA NUMBER;
PI CONSTANT NUMBER := 22/7;
BEGIN
    RADIUS := RAD;
    AREA := PI * POWER(RADIUS,2);
    DBMS_OUTPUT.PUT_LINE('THE AREA OF CIRCLE WITH RADIUS ' || RADIUS
|| ' IS : ' || AREA);
    INSERT INTO AREAS VALUES(RADIUS,AREA);

    EXCEPTION WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/

```

```

SQL> CREATE OR REPLACE PROCEDURE FIND_AREA(RAD NUMBER)
2 AS
3 RADIUS NUMBER;
4 AREA NUMBER;
5 PI CONSTANT NUMBER := 22/7;
6 BEGIN
7     RADIUS := RAD;
8     AREA := PI * POWER(RADIUS,2);
9     DBMS_OUTPUT.PUT_LINE('THE AREA OF CIRCLE WITH RADIUS ' || RADIUS || ' IS : ' || AREA);
10    INSERT INTO AREAS VALUES(RADIUS,AREA);
11
12    EXCEPTION WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE(SQLERRM);
13 END;
14 /

```

Procedure created.

```
SQL> EXEC FIND_AREA(2);
```

PL/SQL procedure successfully completed.

```
SQL> SELECT * FROM AREAS;
```

RADIUS	AREA
2	12.5714286

```
SQL> SET SERVEROUTPUT ON;
```

```
SQL> EXEC FIND_AREA(4);
```

THE AREA OF CIRCLE WITH RADIUS 4 IS : 50.28571428571428571428571428571424

PL/SQL procedure successfully completed.

```
SQL> SELECT * FROM AREAS;
```

RADIUS	AREA
2	12.5714286
4	50.2857143

```
SQL> █
```

- iv. Implement a PL/SQL procedure that takes weight of an apple box as input from the user. If the weight is ≥ 10 kg, rate = Rs. 5/kg. If weight is < 10 kg, rate = Rs. 7/kg. Calculate the cost of the apple box. Display the output on the screen.

```

SQL> SET SERVEROUTPUT ON;
SQL> CREATE OR REPLACE PROCEDURE FINDCOST(WEIGHT NUMBER) AS
2 BEGIN
3     IF WEIGHT >= 10 THEN
4         DBMS_OUTPUT.PUT_LINE('THE COST OF APPLE BOX IS : ' || (WEIGHT * 5));
5     ELSE DBMS_OUTPUT.PUT_LINE('THE COST OF APPLE BOX IS : ' || (WEIGHT * 7));
6     END IF;
7 END;
8 /

Procedure created.

SQL> EXEC FINDCOST(12);
THE COST OF APPLE BOX IS : 60

PL/SQL procedure successfully completed.

SQL> EXEC FINDCOST(5);
THE COST OF APPLE BOX IS : 35

PL/SQL procedure successfully completed.

SQL> █

```

- v. Implement a PL/SQL procedure to calculate the difference between highest salaried and lowest salaried employee. Store the information in a table.

```

CREATE TABLE EMP(
    SAL_DIFF NUMBER
);
CREATE OR REPLACE PROCEDURE SALDIFF(HIGHEST NUMBER, LOWEST
NUMBER) AS
RESULT NUMBER;
BEGIN
    DBMS_OUTPUT.PUT_LINE('THE HIGHEST SALARY IS : ' || HIGHEST);
    DBMS_OUTPUT.PUT_LINE('THE LOWEST SALARY IS : ' || LOWEST);
    RESULT := HIGHEST - LOWEST;
    DBMS_OUTPUT.PUT_LINE('THE DIFFERENCE IS : ' || RESULT);
    INSERT INTO EMP VALUES(RESULT);
END;
/

```

```

SQL> CREATE TABLE EMP(
2     SAL_DIFF NUMBER
3 );

Table created.

SQL> DESC EMP;
Name                               Null?    Type
-----
SAL_DIFF                           NUMBER

SQL> CREATE OR REPLACE PROCEDURE SALDIFF(HIGHEST NUMBER, LOWEST NUMBER) AS
2 RESULT NUMBER;
3 BEGIN
4     DBMS_OUTPUT.PUT_LINE('THE HIGHEST SALARY IS : ' || HIGHEST);
5     DBMS_OUTPUT.PUT_LINE('THE LOWEST SALARY IS : ' || LOWEST);
6     RESULT := HIGHEST - LOWEST;
7     DBMS_OUTPUT.PUT_LINE('THE DIFFERENCE IS : ' || RESULT);
8     INSERT INTO EMP VALUES(RESULT);
9 END;
10 /

Procedure created.

SQL> EXEC SALDIFF(10000,500);
THE HIGHEST SALARY IS : 10000
THE LOWEST SALARY IS : 500
THE DIFFERENCE IS : 9500

PL/SQL procedure successfully completed.

SQL> SELECT * FROM EMP;

SAL_DIFF
-----
9500

SQL> █

```


- vi. Implement a PL/SQL block using cursor that will display the name, department and the salary of the first 3 employees getting lowest salary.
[Table: Employee (ename, dept, salary)]

```
CREATE TABLE EMPLOYEES(  
    ENAME VARCHAR2(20) NOT NULL,  
    DEPT VARCHAR2(20) NOT NULL,  
    SALARY NUMBER NOT NULL  
);
```

```
INSERT ALL  
INTO EMPLOYEES VALUES('ARKA','JAVA',12000)  
INTO EMPLOYEES VALUES('RAMU','SQL',13000)  
INTO EMPLOYEES VALUES('SIDD','C++',14000)  
INTO EMPLOYEES VALUES('MOHIT','C',9000)  
SELECT * FROM DUAL;
```

```
SQL> CREATE TABLE EMPLOYEES(  
 2     ENAME VARCHAR2(20) NOT NULL,  
 3     DEPT VARCHAR2(20) NOT NULL,  
 4     SALARY NUMBER NOT NULL  
 5 );
```

Table created.

```
SQL> DESC EMPLOYEES;
```

Name	Null?	Type
ENAME	NOT NULL	VARCHAR2(20)
DEPT	NOT NULL	VARCHAR2(20)
SALARY	NOT NULL	NUMBER

```
SQL> INSERT ALL  
 2 INTO EMPLOYEES VALUES('ARKA','JAVA',12000)  
 3 INTO EMPLOYEES VALUES('RAMU','SQL',13000)  
 4 INTO EMPLOYEES VALUES('SIDD','C++',14000)  
 5 INTO EMPLOYEES VALUES('MOHIT','C',9000)  
 6 SELECT * FROM DUAL;
```

4 rows created.

```
SQL> SELECT * FROM EMPLOYEES;
```

ENAME	DEPT	SALARY
ARKA	JAVA	12000
RAMU	SQL	13000
SIDD	C++	14000
MOHIT	C	9000

```
SQL> █
```

```

SET SERVEROUTPUT ON;
DECLARE
    EMP EMPLOYEES%ROWTYPE;
    CURSOR E IS SELECT * FROM EMPLOYEES ORDER BY SALARY;
    N NUMBER DEFAULT 0;
BEGIN
    N := N + 1;
    OPEN E;
    LOOP
        FETCH E INTO EMP;
        EXIT WHEN E%NOTFOUND OR N>3;
        DBMS_OUTPUT.PUT_LINE(EMP.ENAME || ' ' || EMP.DEPT || ' ' ||
EMP.SALARY);
        N := N + 1;
    END LOOP;
    CLOSE E;
END;
/

```

```

SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
2     EMP EMPLOYEES%ROWTYPE;
3     CURSOR E IS SELECT * FROM EMPLOYEES ORDER BY SALARY;
4     N NUMBER DEFAULT 0;
5 BEGIN
6     N := N + 1;
7     OPEN E;
8     LOOP
9         FETCH E INTO EMP;
10        EXIT WHEN E%NOTFOUND OR N>3;
11        DBMS_OUTPUT.PUT_LINE(EMP.ENAME || ' ' || EMP.DEPT || ' ' || EMP.SALARY);
12        N := N + 1;
13    END LOOP;
14    CLOSE E;
15 END;
16 /
MOHIT C 9000
ARKA JAVA 12000
RAMU SQL 13000

PL/SQL procedure successfully completed.

SQL> █

```

- vii. Implement a PL/SQL cursor that will update salary of all employees, such that, it allows an increment of 20% if the salary is less than 2000 otherwise increment of Rs.1000. It should print old and new salary for all employees.
[Table: Employee (ename, dept, salary)]

```

SET SERVEROUTPUT ON;
DECLARE
    CURSOR E IS SELECT ENAME, SALARY FROM EMPLOYEES;
    NAME EMPLOYEES.ENAME%TYPE;
    OLDSAL EMPLOYEES.SALARY%TYPE;
    NEWSAL EMPLOYEES.SALARY%TYPE;
BEGIN
    OPEN E;
    LOOP
        FETCH E INTO NAME,OLDSAL;
        EXIT WHEN E%NOTFOUND;

        IF OLDSAL < 2000 THEN NEWSAL := OLDSAL * 1.2;
        ELSE NEWSAL := OLDSAL + 1000;
    END LOOP;
END;

```


END IF;

UPDATE EMPLOYEES
SET SALARY = NEWSAL
WHERE ENAME = NAME;

DBMS_OUTPUT.PUT_LINE('Employee: ' || NAME || ', Old Salary: ' || OLDSAL || ', New
Salary: ' || NEWSAL);

END LOOP;

END;

/

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2   CURSOR E IS SELECT ENAME, SALARY FROM EMPLOYEES;
  3   NAME EMPLOYEES.ENAME%TYPE;
  4   OLDSAL EMPLOYEES.SALARY%TYPE;
  5   NEWSAL EMPLOYEES.SALARY%TYPE;
  6 BEGIN
  7   OPEN E;
  8   LOOP
  9     FETCH E INTO NAME,OLDSAL;
 10     EXIT WHEN E%NOTFOUND;
 11
 12     IF OLDSAL < 2000 THEN NEWSAL := OLDSAL * 1.2;
 13     ELSE NEWSAL := OLDSAL + 1000;
 14     END IF;
 15
 16     UPDATE EMPLOYEES
 17     SET SALARY = NEWSAL
 18     WHERE ENAME = NAME;
 19
 20     DBMS_OUTPUT.PUT_LINE('Employee: ' || NAME || ', Old Salary: ' || OLDSAL || ', New Salary: ' || NEWSAL);
 21   END LOOP;
 22 END;
 23 /
Employee: ARKA, Old Salary: 13000, New Salary: 14000
Employee: RAMU, Old Salary: 14000, New Salary: 15000
Employee: SIDD, Old Salary: 15000, New Salary: 16000
Employee: MOHIT, Old Salary: 10000, New Salary: 11000

PL/SQL procedure successfully completed.

SQL> █
```