**Assignment-8**

1. 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

);

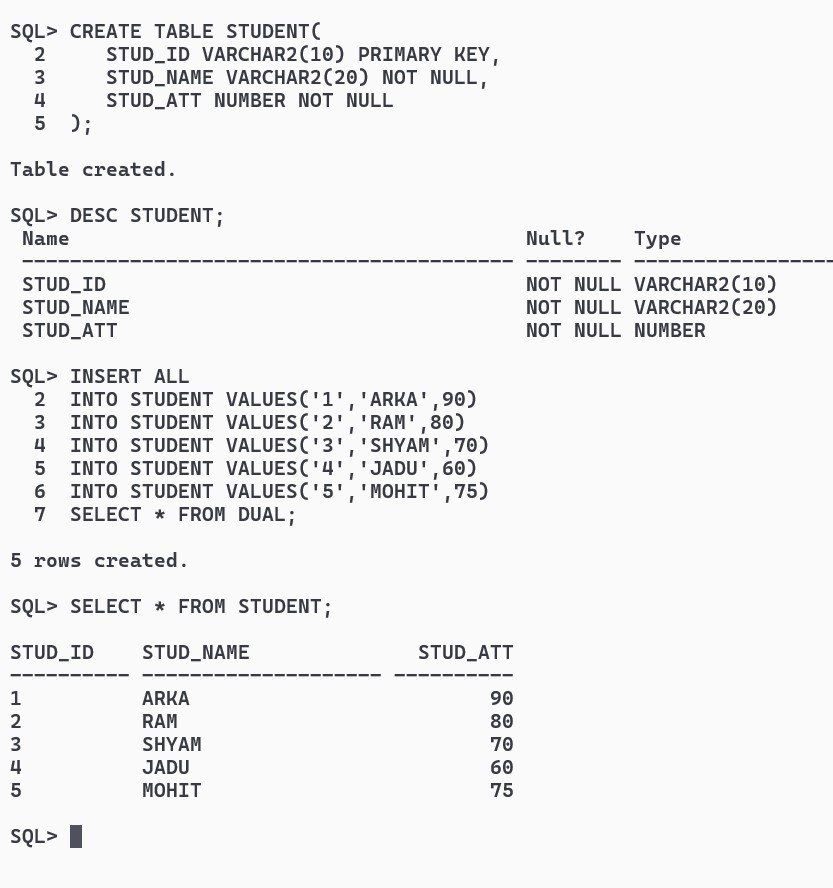
INSERT INTO STUDENT VALUES('1','ARKA',90);

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);



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;

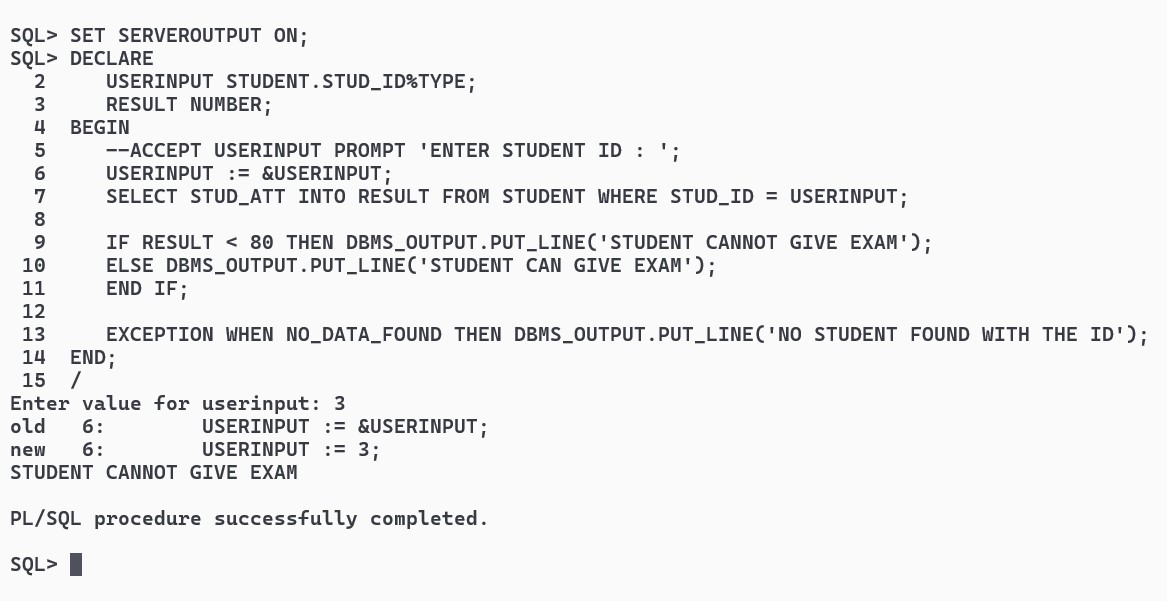
IF RESULT < 80 THEN DBMS\_OUTPUT.PUT\_LINE('STUDENT CANNOT GIVE EXAM');

ELSE DBMS\_OUTPUT.PUT\_LINE('STUDENT CAN GIVE EXAM');

END IF;

END;

/



1. 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

);

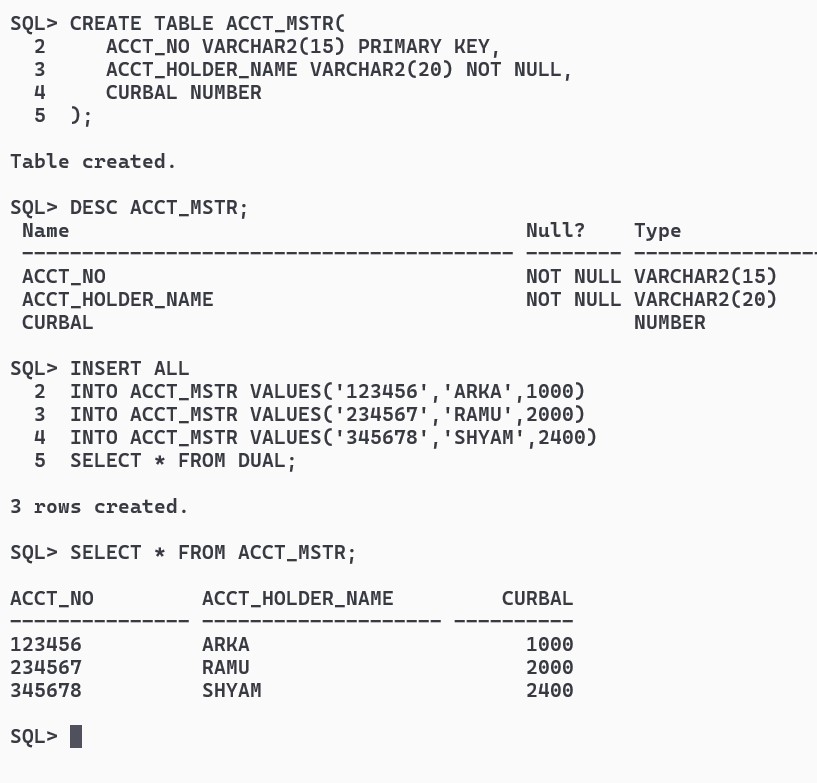
INSERT ALL

INTO ACCT\_MSTR VALUES('123456','ARKA',1000)

INTO ACCT\_MSTR VALUES('234567','RAMU',2000)

INTO ACCT\_MSTR VALUES('345678','SHYAM',2400)

SELECT \* FROM DUAL;



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;

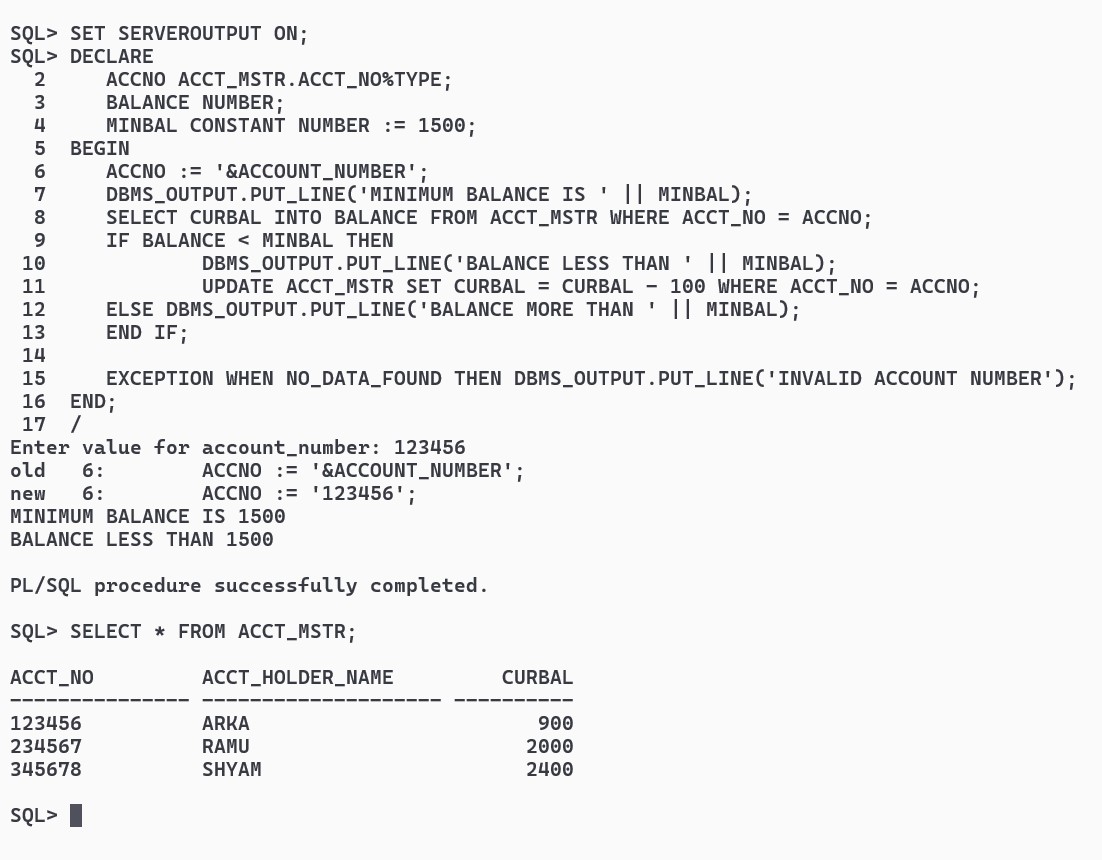
ELSE DBMS\_OUTPUT.PUT\_LINE('BALANCE MORE THAN ' || MINBAL);

END IF;

EXCEPTION WHEN NO\_DATA\_FOUND THEN DBMS\_OUTPUT.PUT\_LINE('INVALID ACCOUNT NUMBER');

END;

/



1. 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.

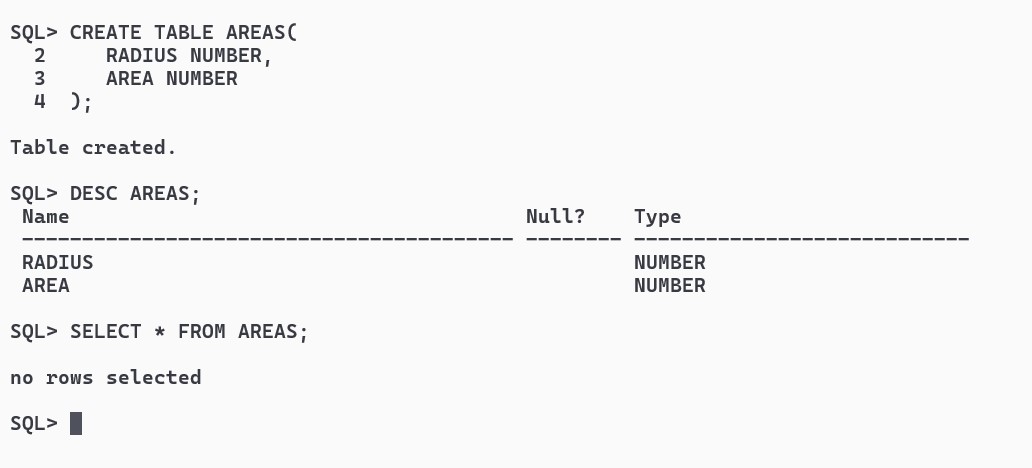
[Table: AREAS (RADIUS, AREA)].

CREATE TABLE AREAS(

RADIUS NUMBER,

AREA NUMBER

);



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);

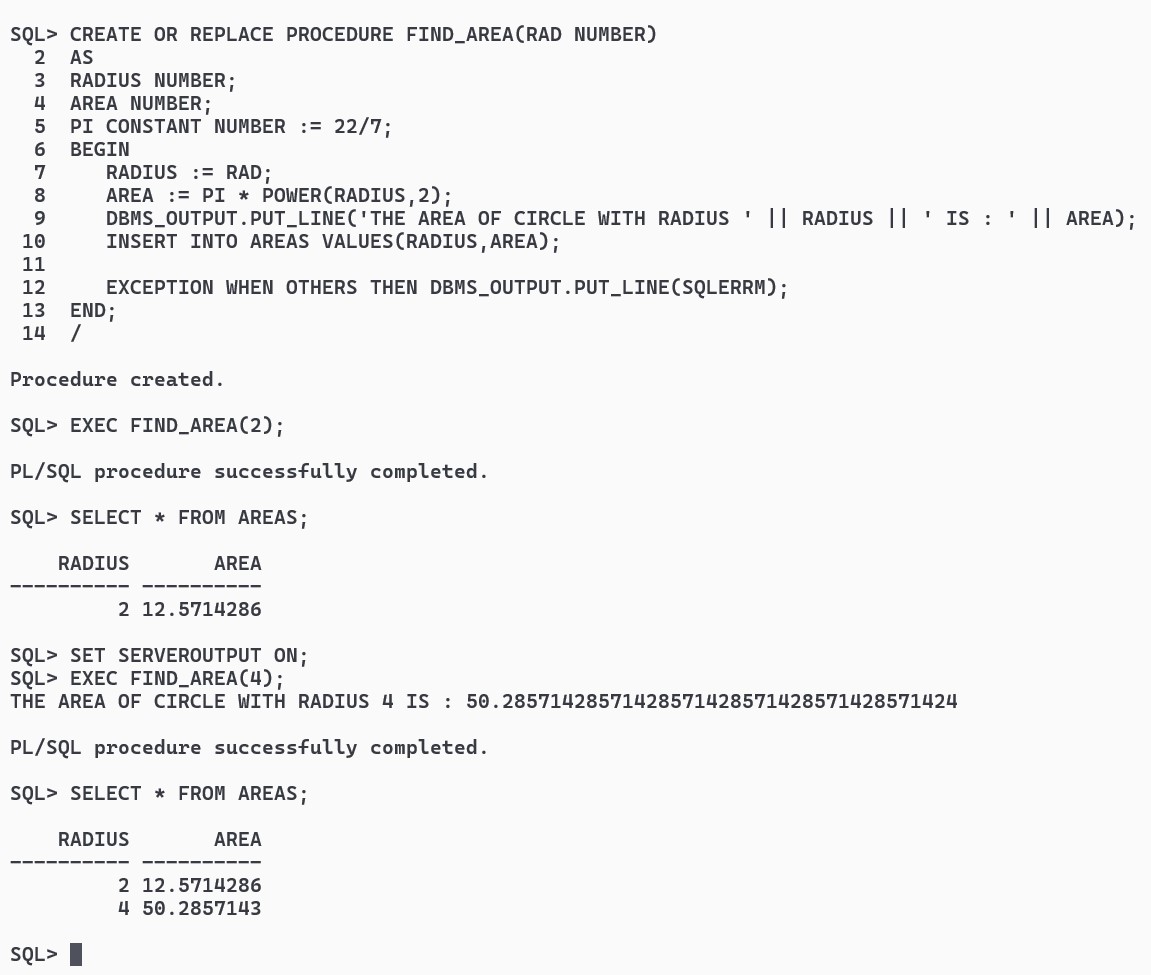
DBMS\_OUTPUT.PUT\_LINE('THE AREA OF CIRCLE WITH RADIUS ' || RADIUS || ' IS : ' || AREA);

INSERT INTO AREAS VALUES(RADIUS,AREA);

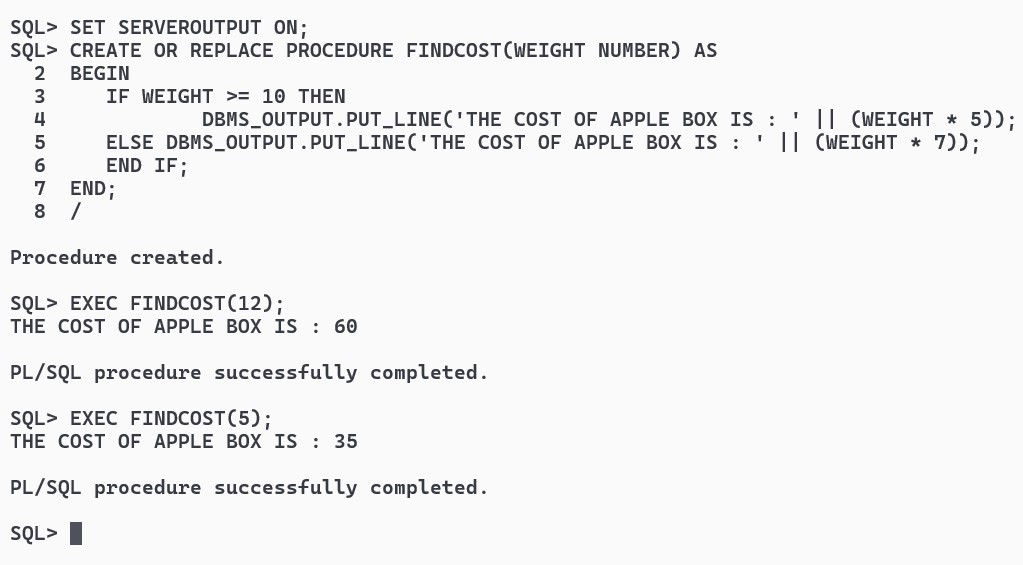
EXCEPTION WHEN OTHERS THEN DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

/



1. 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.



1. 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);

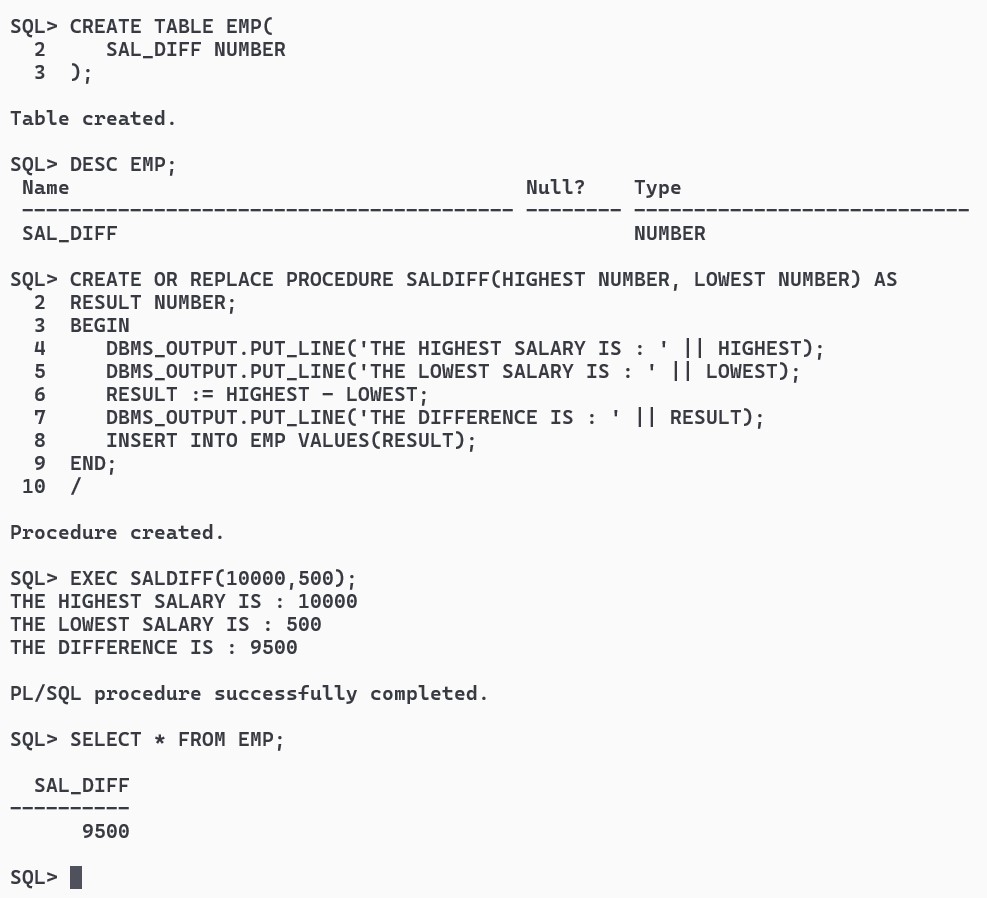
RESULT := HIGHEST - LOWEST;

DBMS\_OUTPUT.PUT\_LINE('THE DIFFERENCE IS : ' || RESULT);

INSERT INTO EMP VALUES(RESULT);

END;

/



1. 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

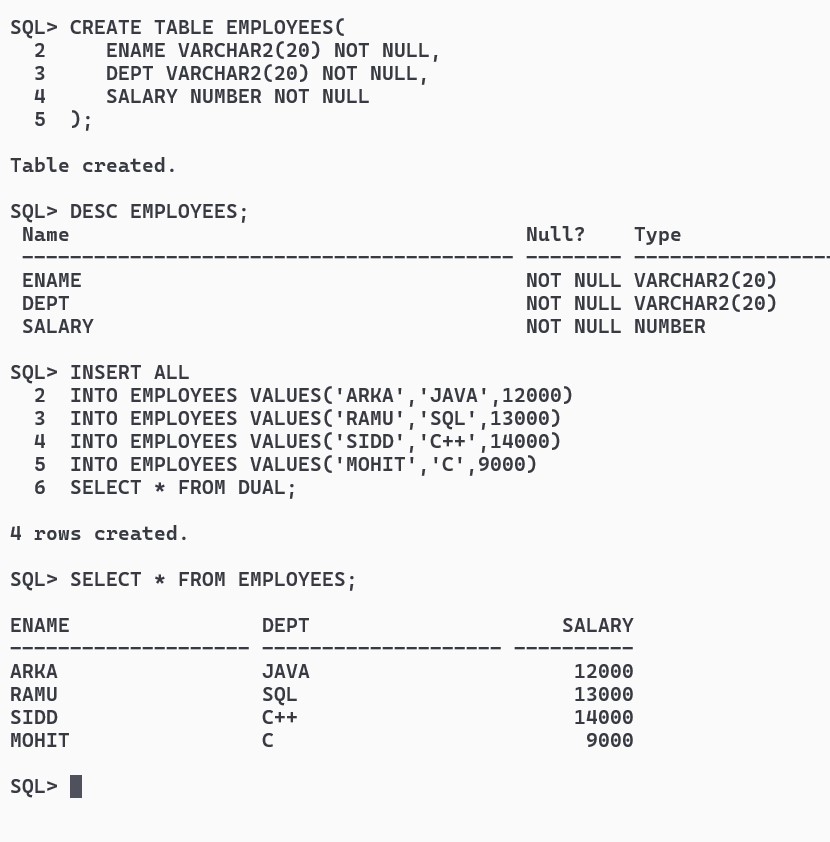
INTO EMPLOYEES VALUES('ARKA','JAVA',12000)

INTO EMPLOYEES VALUES('RAMU','SQL',13000)

INTO EMPLOYEES VALUES('SIDD','C++',14000)

INTO EMPLOYEES VALUES('MOHIT','C',9000)

SELECT \* FROM DUAL;



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);

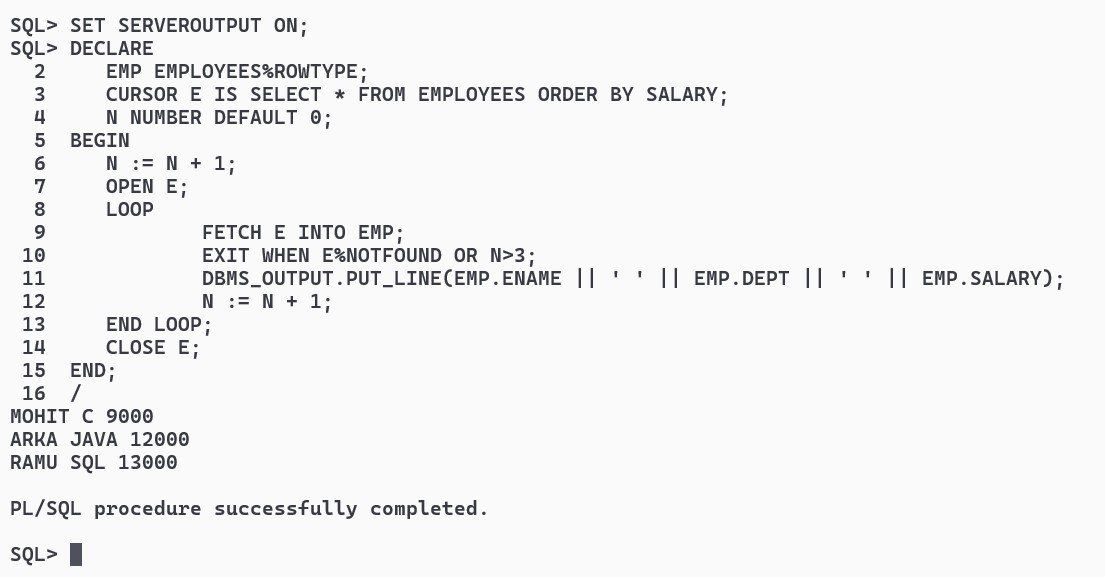
N := N + 1;

END LOOP;

CLOSE E;

END;

/



1. 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 IF;

UPDATE EMPLOYEES

SET SALARY = NEWSAL

WHERE ENAME = NAME;

DBMS\_OUTPUT.PUT\_LINE('Employee: ' || NAME || ', Old Salary: ' || OLDSAL || ', New Salary: ' || NEWSAL);

END LOOP;

END;

/

