NAME : RAHUL RAI

ROLL NO : 205119073

DBMS LAB MANUAL

DBMS LAB MANUAL

EXERCISE 1

Problem 1.1

```
CREATE TABLE EMP(
   EMPNO INT(6) NOT NULL PRIMARY KEY,
   ENAME VARCHAR(20) NOT NULL,
   JOB VARCHAR(10) NOT NULL,
   MGR INT(4),
   DEPTNO INT(4),
   SAL FLOAT(7, 2)
)
```

Problem 1.2

```
ALTER TABLE EMP ADD COMMISSION INT;
```

Problem 1.3

```
ALTER TABLE EMP MODIFY JOB VARCHAR(20);
```

Problem 1.4

```
CREATE TABLE DEPT(
DEPTNO INT(2) NOT NULL PRIMARY KEY,
DNAME VARCHAR(50),
LOC VARCHAR(50)
);
```

Problem 1.5

```
ALTER TABLE EMP ADD CONSTRAINT FK_DEPTNO FOREIGN KEY
(DEPTNO) REFERENCES DEPT(DEPTNO);
```

Problem 1.6

```
ALTER TABLE EMP ADD CHECK (EMPNO > 100);
```

Problem 1.7

```
ALTER TABLE EMP MODIFY SAL FLOAT(7,2) NOT NULL DEFAULT 5000;
```

Problem 1.8

```
ALTER TABLE EMP ADD DOB DATE;
```

EXERCISE 2

Problem 2.1

```
INSERT INTO DEPT VALUES (10,'MANAGEMENT','MAIN BLOCK'),
(20,'DEVELOPMENT','MAIN BLOCK'),
(30,'MAINTAINANCE','MAIN BLOCK'),
(40,'TRANSPORT','ADMIN BLOCK'),
(50,'SALES','HEAD OFFICE');
```

Problem 2.2

```
INSERT INTO EMP VALUES (7369,'SMITH','CLERK',7566,20,800,0,'1980-12-17'),
     (7399,'ASANT','SALESMAN',7566,20,1600,300,'1981-02-20'),
     (7499,'ALLEN','SALESMAN',7698,30,1600,300,'1981-02-20'),
     (7521,'WARD','SALESMAN',7698,30,1250,500,'1982-02-22'),
     (7566,'JONES','MANAGER',7839,20,5975,500,'1981-04-02'),
     (7698,'BLAKE','MANAGER',7839,30,9850,1400,'1979-05-01'),
     (7611,'SCOTT','HOD',7839,10,3000,NULL,'1976-07-12'),
     (7839,'CLARK','CEO',NULL,20,800,0,'1972-03-16'),
     (7368,'FORD','SUPERVIS',7366,20,800,0,'1980-12-12'),
     (7599,'ALLEY','SALESMAN',7698,30,1600,300,'1981-02-20'),
     (7421,'DRANK','CLERCK',7698,30,1250,500,'1982-01-22');
```

Problem 2.3

```
UPDATE EMP SET COMMISSION = 1000 WHERE JOB = 'MANAGER';
```

Problem 2.4

```
CREATE TABLE EMPPSEUDO AS SELECT * FROM EMP;
```

Problem 2.5

```
DELETE FROM EMP WHERE JOB = 'SUPERVISOR';
```

Problem 2.6

```
DELETE FROM EMP WHERE EMPNO = 7599;
```

Problem 2.7

```
SELECT * FROM EMP ORDERBY SAL;
```

Problem 2.8

```
SELECT * FROM EMP ORDER BY SAL;
```

Problem 2.9

```
SELECT * FROM EMP WHERE DEPTNO = 30;
```

Problem 2.10

```
SELECT DISTINCT DEPTNO FROM DEPT;
```

Problem 2.11

```
SELECT * FROM EMP ORDER BY ENAME;
```

Problem 2.12

```
CREATE TABLE MANAGER AS SELECT * FROM EMP WHERE JOB = 'MANAGER';
```

Problem 2.13

```
SELECT * FROM EMP WHERE COMMISSION = NULL
```

Problem 2.14

```
SELECT E.ENAME, D.DNAME FROM EMP E, DEPT D WHERE E.DEPTNO = D.DEPTNO;
```

EXERCISE 3

Problem 3.1

```
SELECT * FROM EMP WHERE DEPTNO IN (7369,7499);
```

Problem 3.2

```
SELECT * FROM EMPLOYEE WHERE EMPNAME LIKE ‘S%’;
```

Problem 3.3

```
SELECT * FROM EMPLOYEE WHERE EMPNAME NOT LIKE ‘S%’;
```

Problem 3.4

```
SELECT * FROM EMPLOYEE WHERE EMPNO BETWEEN 7500 AND 7600 ;
```

Problem 3.5

```
SELECT * FROM EMPLOYEE WHERE EMPNO NOT BETWEEN 7500 AND 7600 ;
```

Problem 3.6

```
SELECT SQRT(SAL) FROM EMP;
```

Problem 3.7

```
SELECT COUNT(*) FROM EMP;
```

Problem 3.8

```
SELECT SUM(SAL),AVG(SAL) FROM EMP;
```

Problem 3.9

```
SELECT MIN(SAL) “MIN_SAL”, MAX(SAL) “MAX_SAL” FROM EMP;
```

Problem 3.10

```
SELECT SUM (SAL) FROM EMP ;
```

Problem 3.11

```
SELECT JOB,SUM (SAL) FROM EMP GROUP BY JOB;
```

Problem 3.12

```
SELECT TO_CHAR(TO_DATE(’14-JUL-09’),’MONTH’) FROM DUAL;
```

Problem 3.13

```
SELECT TO_DATE(DOJ,’DD-MM-YY’) FROM EMP;
```

Problem 3.14

```
SELECT ADD_MONTHS(DOB,2) FROM EMP;
```

Problem 3.15

```
SELECT LAST_DAY(’05-OCT-09’) FROM DUAL;
```

Problem 3.16

```
SELECT ROUND(TO_DATE(DOB),’MONTH’) FROM EMP;
SELECT ROUND(TO_DATE(DOB),’YEAR’) FROM EMP;
SELECT ROUND(TO_DATE(DOB),’DAY’) FROM EMP;
```

Problem 3.17

```
SELECT(SYSDATE-60) FROM DUAL;
```

Problem 3.18

```
SELECT ENAME , SAL , SAL+0.15* SAL FROM EMP;
```

Problem 3.19

```
SELECT ENAME FROM EMP WHERE ENAME LIKE ‘B%’ OR ENAME
LIKE ‘C%’;
```

Problem 3.20

```
SELECT ENAME, SAL,MGR FROM EMP WHERE SAL IN (SELECT
MIN(SAL) FROM EMP GROUP BY MGR);
```

Problem 3.21

```
SELECT DNAME, COUNT (ENAME) FROM EMP, DEPT WHERE
EMP.DEPTNO=DEPT.DEPTNO GROUP BY DNAME;
```

Problem 3.22

```
SELECT EMPNAME FROM EMP WHERE LENGTH (EMPNAME) <=5;
```

Problem 3.23

```
SELECT ENAME FROM EMP WHERE MGR IN(7602,7566,7789);
```

Problem 3.24

```
SELECT COUNT (DISTINCT JOB) FROM EMP;
```

Problem 3.25

```
SELECT MAX(SAL)-MIN(SAL) FROM EMP;
```

Problem 3.26

```
SELECT COUNT(DISTINCT DEPTNO) FROM EMP;
```

Problem 3.27

```
SELECT EMPNAME , DOB FROM EMP WHERE TO_CHAR
(DOB,’MON’)=’FEB’;
```

Problem 3.28

```
SELECT PNAME FROM PROGRAMMER WHERE
TO_CHAR(DOB,’MON’) LIKE TO_CHAR (SYSDATE, ‘MON’);
```

Problem 3.29

```
SELECT ENAME FROM EMP WHERE ENAME REGEXP ‘^ [S]’  AND
ENAME REGEXP ‘[H]$’;
```

Problem 3.30

```
SELECT ENAME FROM EMP WHERE SALARY >5000 AND SALARY
>6000;
```

Exercise 4

Problem 4.1

```
SELECT ENAME,DNAME FROM EMP,DEPT WHERE
DNAME='MAINTAINANCE' OR DNAME='DEVELOPMENT' ;
```

Problem 4.2

```
SELECT  ENAME FROM EMP WHERE SAL >(SELECT MIN(SAL)FROM
EMP) AND JOB LIKE ('M%');
```

Problem 4.3

```
SELECT ENAME FROM EMP WHERE JOB =( SELECT JOB FROM EMP
WHERE ENAME='JONES');
```

Problem 4.4

```
 SELECT  *  FROM EMP WHERE SAL >ANY( SELECT SAL FROM EMP
WHERE DEPTNO=30 );
```

Problem 4.5

```
 SELECT * FROM EMP WHERE JOB =( SELECT JOB FROM EMP
  WHERE ENAME='JONES') AND SAL>=( SELECT SAL FROM EMP
  WHERE ENAME='FORD');
```

Problem 4.6

```
SELECT ENAME, JOB FROM EMP WHERE DEPTNO=10 AND  JOB
IN(SELECT JOB FROM EMP,DEPT WHERE
EMP.DEPTNO=DEPT.DEPTNO AND DNAME='MANAGEMENT');
```

Problem 4.7

```
SELECT * FROM EMP WHERE SAL >(SELECT AVG(SAL)FROM EMP);
```

Problem 4.8

```
SELECT ENAME,JOB,DNAME FROM EMP,DEPT WHERE
EMP.DEPTNO=DEPT.DEPTNO;
```

Problem 4.9

```
SELECT * FROM EMP WHERE JOB IN (SELECT JOB FROM EMP,DEPT
WHERE EMP.DEPTNO=DEPT.DEPTNO AND LOC='MAIN BLOCK');
```

Problem 4.10

```
SELECT * FROM EMP WHERE DEPTNO=10 AND  JOB IN(SELECT JOB
FROM EMP,DEPT WHERE EMP.DEPTNO=DEPT.DEPTNO AND
DNAME='DEVELOPMENT');
```

Problem 4.11

```
SELECT * FROM EMP WHERE JOB =( SELECT JOB FROM EMP
WHERE ENAME='FORD') AND SAL=( SELECT SAL FROM EMP WHERE
ENAME='FORD');
```

Problem 4.13

```
SELECT  *  FROM EMP WHERE DEPTNO=20 AND JOB=ANY( SELECT
JOB FROM EMP WHERE DEPTNO=30 );
```

Problem 4.14

```
SELECT ENAME FROM EMP WHERE SAL >ANY( SELECT SAL FROM
EMP WHERE DEPTNO IN (20,30));
```

Problem 4.15

```
SELECT MAX(SAL) FROM EMP WHERE SAL>9000;
```

Problem 4.16

```
SELECT MIN(SAL) FROM EMP WHERE SAL BETWEEN 1000 AND 5000;
```

Problem 4.17

```
SELECT * FROM EMP,DEPT WHERE EMP.DEPTNO=DEPT.DEPTNO;
```

Problem 4.18

```
SELECT * FROM EMP,DEPT WHERE NOT
EMP.DEPTNO=DEPT.DEPTNO;
```

Problem 4.19

```
SELECT ENAME,DNAME FROM EMP LEFT JOIN DEPT ON
EMP.DEPTNO=DEPT.DEPTNO;
```

Problem 4.20

```
SELECT ENAME,DNAME FROM EMP RIGHT JOIN DEPT ON
EMP.DEPTNO=DEPT.DEPTNO;
```

Problem 4.21

```
SELECT ENAME,DNAME FROM EMP FULL OUTER JOIN DEPT ON
EMP.DEPTNO=DEPT.DEPTNO;
```

Problem 4.22

```
SELECT ENAME,JOB FROM EMP WHERE JOB='MANAGER';
```

Problem 4.23

```
SELECT ENAME,JOB,SAL FROM EMP WHERE JOB='MANAGER';
```

Problem 4.24

```
SELECT ENAME,JOB,DNAME,LOC FROM EMP NATURAL JOIN DEPT;
```

EXERCISE 5

Problem 5.1: Display all the dept numbers available with the dept and accdept.

Ans: SELECT D.DEPT_NO FROM DEPT D UNION SELECT A.DEPT_NO FROM ACCDEPT A;

Problem 5.2: Display all the dept numbers available with the dept and accdept.

Ans: SELECT D.DEPT_NO FROM DEPT D UNION ALL SELECT A.DEPT_NO FROM ACCDEPT A;

Problem 5.3: Display dept no available in both the dept and acc dept tables.

Ans: SELECT D.DEPT_NO FROM DEPT D INTERSECT SELECT A.DEPT_NO FROM ACCDEPT A;

Problem 5.4: Display all the dept numbers available in dept and not in accdept.

Ans: SELECT D.DEPT_NO FROM DEPT D MINUS SELECT A.DEPT_NO FROM ACCDEPT A;

Problem 5.5: The organization wants to display only the details of the employees those who are managers.(horizontal portioning)

Ans: CREATE OR REPLACE VIEW MANAGERS AS SELECT * FROM EMP WHERE POST='MANAGER';

SELECT * FROM MANAGERS;

Problem 5.6: The organization wants to display only the details like empno, empname, deptno, deptname of the employees. (vertical portioning)

Ans: CREATE OR REPLACE VIEW GENERAL AS SELECT EMPNO,EMPNAME,DEPTNO,DEPTNAME FROM EMP;

SELECT * FROM GENERAL;

Problem 5.7: The organization wants to display only the details like empno, empname, deptno, deptname of the all the employees except the hod and ceo . (full portioning)

Ans: CREATE OR REPLACE VIEW ALL AS SELECT EMPNO, EMPNAME, DEPTNO, DEPTNAME FROM EMP WHERE EMPPOST NOT IN (‘HOD’, ’CEO’);

SELECT * FROM ALL;

Problem 5.10: Drop a view.

Ans: DROP VIEW ALL;

EXERCISE 6

Problem 6.1
Write a pl/sql program to swap two numbers with out taking third variable.

Ans :

```
DECLARE

A NUMBER(10);

B NUMBER(10);

BEGIN

A:=&A;
```

```
B:=&B;
DBMS_OUTPUT.PUT_LINE('THE PREV VALUES OF A AND B WERE');
DBMS_OUTPUT.PUT_LINE(A);
DBMS_OUTPUT.PUT_LINE(B);
A:=A+B;
B:=A-B;
A:=A-B;
DBMS_OUTPUT.PUT_LINE('THE VALUES OF A AND B ARE');
DBMS_OUTPUT.PUT_LINE(A);
DBMS_OUTPUT.PUT_LINE(B);
END;
/
```

Problem 6.2

Write a pl/sql program to swap two numbers by taking third variable.

```
DECLARE
A NUMBER(10);
B NUMBER(10);
C NUMBER(10);
BEGIN
A:=&A;
B:=&B;
DBMS_OUTPUT.PUT_LINE(‘THE PREV VALUES OF A AND B WERE');
DBMS_OUTPUT.PUT_LINE(A);
DBMS_OUTPUT.PUT_LINE(B);
C:=A;
```

```
A:=B;
B:=C;
DBMS_OUTPUT.PUT_LINE('THE VALUES OF A AND B ARE');
DBMS_OUTPUT.PUT_LINE(A);
DBMS_OUTPUT.PUT_LINE(B);
END;
/
```

Problem 6.3 Write a pl/sql program to find the largest of two numbers.

```
DECLARE
A NUMBER(10);
B NUMBER(10);
BEGIN
A:=&A;
B:=&B;
IF A=B THEN
DBMS_OUTPUT.PUT_LINE('BOTH ARE EQUAL');
ELSIF A>B THEN
DBMS_OUTPUT.PUT_LINE('A IS GREATER');
ELSE
DBMS_OUTPUT.PUT_LINE('B IS GREATER');
END IF;
END;
/
```

Problem 6.4

Write a pl/sql program to find the total and average of 6 subjects and display the grade.

```
DECLARE
A NUMBER;
B NUMBER;
C NUMBER;
D NUMBER;
E NUMBER;
F NUMBER;
TOTAL NUMBER;
PER NUMBER;
BEGIN
DBMS_OUTPUT.PUT_LINE('ENTER MARKS OF SUBJECT');
A:=&A;
B:=&B;
C:=&C;
D:=&D;
E:=&E;
F:=&F;
TOTAL:=(A+B+C+D+E+F);
PER:=(TOTAL/600)*100;
IF A<40 OR B<40 OR C<40 OR D<40 OR E<40 OR F<40 THEN
DBMS_OUTPUT.PUT_LINE('FAIL');
ELSIF PER>75 THEN
DBMS_OUTPUT.PUT_LINE('GRADE A');
```

```
ELSIF PER>65 AND PER<=75 THEN
DBMS_OUTPUT.PUT_LINE('GRADE B');
ELSIF PER>50 AND PER<=65 THEN
DBMS_OUTPUT.PUT_LINE('GRADE C');
ELSE
DBMS_OUTPUT.PUT_LINE('GRADE D');
END IF;
DBMS_OUTPUT.PUT_LINE('TOTAL='||TOTAL);
DBMS_OUTPUT.PUT_LINE('PERCENTAGE='||PER);
END;
/
```

Problem 6.5
Write a pl/sql program to find the sum of digits in a given number.

```
DECLARE
A NUMBER;
D NUMBER:=0;
SUM1 NUMBER:=0;
BEGIN
A:=&A;
WHILE A>0
LOOP
D:=MOD(A,10);
SUM1:=SUM1+D;
A:=TRUNC(A/10);
END LOOP;
```

```
DBMS_OUTPUT.PUT_LINE('SUM IS'|| SUM1);
END;
/
```

Problem 6.6 Write a pl/sql program to display the number in reverse order.

```
DECLARE
A NUMBER;
REV NUMBER;
D NUMBER;
BEGIN
A:=&A;
REV:=0;
WHILE A>0
LOOP
D:=MOD(A,10);
REV:=(REV*10)+D;
A:=TRUNC(A/10);
END LOOP;
DBMS_OUTPUT.PUT_LINE('REVERSE NO IS'|| REV);
END;
/
```

Problem 6.7
Write a pl/sql program to check whether the given number is prime or not.

```
DECLARE
A NUMBER;
```

```
C NUMBER:=0;
I NUMBER;
BEGIN
A:=&A;
FOR I IN 2..A-1
LOOP
IF MOD(A,I)=0 THEN
C:=C+1;
END IF;
END LOOP;
IF C=0 THEN
DBMS_OUTPUT.PUT_LINE(A ||'IS A PRIME NUMBER');
ELSE
DBMS_OUTPUT.PUT_LINE(A ||'IS NOT A PRIME NUMBER');
END IF;
END;
/
```

Problem 6.8 Write a pl/sql program to find the factorial of a given number.

```
DECLARE
N NUMBER;
F NUMBER:=1;
BEGIN
N:=&N;
FOR I IN 1..N
```

```
LOOP
F:=F*I;
END LOOP;
DBMS_OUTPUT.PUT_LINE('THE FACTORIAL IS'|| F);
END;
/
```

Problem 6.9

Write a pl/sql code block to calculate the area of a circle for a value of radius varying from 3 to 7.

```
DECLARE
PI CONSTANT NUMBER(4,2):=3.14;
RADIUS NUMBER(5):=3;
AREA NUMBER(6,2);
BEGIN
WHILE RADIUS<7 LOOP
AREA:=PI*POWER(RADIUS,2);
INSERT INTO AREAS VALUES(RADIUS,AREA);
RADIUS:=RADIUS+1;
END LOOP;
END;
/
```

Problem 6.10

Write a pl/sql code block that will accept an account number from the user, check if the users balance is less than minimum balance, only then deduct rs.100/- from the balance.this process is fired on the acct table.

```
DECLARE
XACC_NO NUMBER(5);
XMINBAL NUMBER(5):=1000;
XBALANCE NUMBER(5);
BEGIN
XACC_NO:=&XACC_NO;
SELECT BALANCE INTO XBALANCE FROM ACCT_MASTER WHERE
ACCT_NO=XACC_NO;

IF(XBALANCE < XMINBAL) THEN
UPDATE ACCT_MASTER SET BALANCE=BALANCE-100 WHERE
ACC_NO=XACC_NO;
XBALANCE:=XBALANCE-100;
DBMS_OUTPUT.PUT_LINE('RS 100 IS DEDUCTED AND CURRENT
BALANCE IS'||XBALANCE);

ELSE
DBMS_OUTPUT.PUT_LINE('CURRENT BALANCE IS'||XBALANCE);
END IF;

END;
```

EXERCISE 7

7.1: Addition of two numbers.

```
CREATE OR REPLACE PROCEDURE ADDITION(A NUMBER,B
NUMBER) IS

C NUMBER;

BEGIN

C:=A+B;

DBMS_OUTPUT.PUT_LINE('THE ADDITION IS'||C);
```

```
6* END;
SQL> /
PROCEDURE CREATED.
SQL> EXEC ADDITION(50,51);
THE ADDITION IS 101
```

7.2 :To display a string using procedure.

```
SQL> CREATE OR REPLACE PROCEDURE HELLO(NAME VARCHAR)
2 IS
3 BEGIN
4 DBMS_OUTPUT.PUT_LINE('HI HELLO'||NAME);
5 END;
6 /
SQL> EXEC HELLO('NIVETHA');
HI HELLONIVETHA
PL/SQL PROCEDURE SUCCESSFULLY COMPLETED.
SQL> EXEC HELLO(' NIVETHA');
HI HELLO NIVETHA
```

```
SQL> CREATE OR REPLACE PROCEDURE HELLO
2 IS
3 BEGIN
4 DBMS_OUTPUT.PUT_LINE('HI HELLO');
5* END;
SQL> /
```

```
PROCEDURE CREATED.

SQL> EXEC HELLO

HI HELLO
```

7.3 Write a procedure to add an amount of rs.1000 for the employees whose salaries is greater than 5000 and who belongs to the deptno passed as an argument.

Answer :

```
CREATE TABLE PRODUCT_MASTER
(PRODUCT_ID VARCHAR2(3) PRIMARY KEY,
PRODUCT_NAME VARCHAR2(15),
PRICE NUMBER(10,3));

CREATE TABLE OLD_PRICE
(PRODUCT_ID VARCHAR2(3),
PRODUCT_NAME VARCHAR2(15),
PRICE NUMBER(10,3));

INSERT INTO PRODUCT_MASTER VALUES('P01','HARD DISK',5000);

DECLARE
	CURSOR PRODUCT_CUR IS SELECT * FROM
PRODUCT_MASTER;
BEGIN
	FOR I IN PRODUCT_CUR
	LOOP
		IF
PRO_PRICE(I.PRODUCT_ID,I.PRODUCT_NAME,I.PRICE)=1 THEN
			UPDATE PRODUCT_MASTER SET
PRICE=PRICE+1000 WHERE PRICE>5000;
			DBMS_OUTPUT.PUT_LINE('0');
		ELSE
			INSERT INTO OLD_PRICE
VALUES(I.PRODUCT_ID,I.PRODUCT_NAME,I.PRICE);
			DBMS_OUTPUT.PUT_LINE('1');
		END IF;
	END LOOP;
```

```
END;
/
CREATE OR REPLACE FUNCTION PRO_PRICE(P_ID IN
VARCHAR2,P_NM IN VARCHAR2,RS IN NUMBER) RETURN NUMBER
AS
BEGIN
        IF RS >= 5000 THEN
                RETURN 0;
        ELSE
                RETURN 1;
        END IF;
END;
/
```

7.4 Create or replace procedure salary(deptid number) as

Ans:

```
 BEGIN
         UPDATE EMP SET SAL=SAL+1000 WHERE SAL>5000 AND
        DEPTNO=DEPTID;
 END;
```

7.5 create or replace procedure salary1(empid number) as

```
BEGIN
  UPDATE EMP SET SAL=SAL+SAL*(0.1) WHERE
  EMPNO=EMPID;
END;
```

7.6 create or replace procedure get_sal(dept number) as

```
BEGIN
     FOR S IN (SELECT * FROM EMP WHERE DEPTNO = DEPT)
     LOOP
        DBMS_OUTPUT.PUT_LINE(S.SAL);
     END LOOP;
  END;
```

7.7 create or replace procedure get_nature(dept number) as

```
BEGIN
  FOR S IN (SELECT * FROM EMP WHERE DEPTNO = DEPT)
  LOOP
     DBMS_OUTPUT.PUT_LINE(S.JOB);
   END LOOP;
```

```
         END;
7.8   create or replace procedure dep_name(deptid number)  as
      BEGIN
          SELECT DEPT.DNAME FROM DEPT,EMP WHERE
         EMP.DEPTNO=DEPT.DEPTNO;
      END;
```

EXERCISE 8

8.1: Create a trigger to convert lowercase to uppercase.

```
SQL> CREATE OR REPLACE TRIGGER UPPERCASE
2 BEFORE INSERT OR UPDATE ON EMPLOYEE
3 REFERENCING NEW AS N FOR EACH ROW
4 BEGIN
5 :N.ENAME:=UPPER(:N.ENAME);
6 END;
7 /
TRIGGER CREATED.
SQL> INSERT INTO EMPLOYEE VALUES(11,'RAVI',20000);
1 ROW CREATED.
SQL> SELECT * FROM EMPLOYEE;
ENO ENAME SALARY
11 RAVI 20000
```

8.2: Create a trigger to check the salary is above 1000.

```
SQL> CREATE OR REPLACE TRIGGER SALCONDITION
2 BEFORE INSERT OR UPDATE ON EMPLOYEE
```

```
3 REFERENCING NEW AS N FOR EACH ROW
4 BEGIN
5 IF(:N.SALARY<1000)THEN
6 RAISE_APPLICATION_ERROR(-20001,'SALARY MUST BE GREATER THAN 1000');
7 END IF;
8 END;
9 /
TRIGGER CREATED.
SQL> INSERT INTO EMPLOYEE VALUES(12,'RAM',999);
INSERT INTO EMPLOYEE VALUES(12,'RAM',999)
*

ERROR AT LINE 1:
ORA-20001: SALARY MUST BE GREATER THAN 1000
ORA-06512: AT "SHAR.SALCONDITION", LINE 3
ORA-04088: ERROR DURING EXECUTION OF TRIGGER 'SHAR.SALCONDITION'
SQL> INSERT INTO EMPLOYEE VALUES(12,'RAM',1001);
1 ROW CREATED.
SQL> SELECT * FROM EMPLOYEE;
ENO ENAME SALARY
--------- -------------------- ---------
11 RAVI 20000
12 RAM 1001
```

8.3: Create a trigger to avoid deletion on Wednesday.

```
SQL> CREATE OR REPLACE TRIGGER DATEDELETE
2 BEFORE DELETE ON EMPLOYEE
3 DECLARE
4 DATE1 CHAR(5);
5 BEGIN
6 DATE1:=TO_CHAR(SYSDATE,'DY');
7 if date1 in ('wed','WED')then
8 raise_application_error(-20002,'records cannot be deleted');
9 end if;
10 end;
11 /
Trigger created.
SQL> delete from employee where eno=11;
delete from employee where eno=11
ERROR at line 1:
ORA-20002: records cannot be deleted
ORA-06512: at "SHAR.DATEDELETE", line 6
ORA-04088: error during execution of trigger 'SHAR.DATEDELETE'
```

8.4: Create a trigger to avoid deletion of particular date

```
SQL> create or replace trigger namedelete
2 before delete on employee
3 referencing new as n for each row
4 begin
```

```
5 if rtrim(:n.ename)in('ravi','RAVI')then
6 raise_application_error(-20003,'record cannot be deleted for ravi');
7 end if;
8 end;
9 /
Trigger created.
SQL> delete from employee where ename='RAVI';
delete from employee where ename='RAVI'
*
ERROR at line 1:
ORA-20002: records cannot be deleted for ravi
ORA-06512: at "SHAR.DATEDELETE", line 6
ORA-04088: error during execution of trigger 'SHAR.DATEDELETE'
```

8.5: CREATE A TRIGGER TO AVOID INSERTION OF PARTICULAR DATA

```
SQL> create or replace trigger inserteno
2 before insert on employee
3 referencing new as n for each row
4 begin
5 if(:n.eno=14)then
6 raise_application_error(-20003,'cannot insert this eno');
7 end if;
8* end;
SQL> /
Trigger created.
```

```
SQL> insert into employee values(14,'seetha',20000);

insert into employee values(14,'seetha',20000)

*

ERROR at line 1:

ORA-20003: cannot insert this eno

ORA-06512: at "SHAR.INSERTENO", line 3

ORA-04088: error during execution of trigger 'SHAR.INSERTENO'
```

```
8.6:CREATE OR RELPLACE TRIGGER trig1 before insert on DEPT for each row DECLARE a number;

    BEGIN

        if(:new.DEPTNO is Null) then

            raise_application_error(-20001,'error:: DEPTNO cannot be null');

        else

            select count(*) into a from DEPT where DEPTNO =:new.DEPTNO;

            if(a=1) then

                raise_application_error(-20002,'error:: cannot have duplicate DEPTNo ');

            end if;

        end if;

    END;
```

```
8.7:CREATE [OR REPLACE] TRIGGER trig2 After delete on DEPT FOR EACH ROW

    BEGIN
```

```
        DELETE FROM emp WHERE emp.deptno=:new.deptno;
    END;
```

```
8.8: CREATE TRIGGER trig3 AFTER DELETE ON emp FOR EACH ROW
    BEGIN
        INSERT INTO log(val1, val2, ...) VALUES (old.val1, old.val2,
    ...);
     END;
```