Experiment -13

13. Write PL/SQL program to implement Stored Procedure on table.

```
CREATE TABLE SAILOR(ID NUMBER(10) PRIMARY KEY,NAME VARCHAR2(100))
```

Table created.

```
CREATE OR REPLACE PROCEDURE INSERTUSER
(ID IN NUMBER,
NAME IN VARCHAR2)
IS
BEGIN
INSERT INTO SAILOR VALUES(ID,NAME);
DBMS_OUTPUT.PUT_LINE('RECORD INSERTED SUCCESSFULLY');
END;
```

Procedure created.

```
DECLARE
CNT NUMBER;
BEGIN
INSERTUSER(101,'NARASIMHA');
SELECT COUNT(*) INTO CNT FROM SAILOR;
DBMS_OUTPUT.PUT_LINE(CNT||' RECORD IS INSERTED SUCCESSFULLY');
END;
```

Statement processed.
RECORD INSERTED SUCCESSFULLY
1 RECORD IS INSERTED SUCCESSFULLY

Experiment – 14

14. Write PL/SQL program to implement Stored Function on table.

```
CREATE OR REPLACE FUNCTION ADDER(N1 IN NUMBER, N2 IN NUMBER)
RETURN NUMBER
IS
N3 NUMBER(8);
BEGIN
N3 :=N1+N2;
RETURN N3;
END;
```

Function created.

```
DECLARE
N3 NUMBER(2);
BEGIN
N3 := ADDER(11,22);
DBMS_OUTPUT.PUT_LINE('ADDITION IS: ' || N3);
END;
```

Statement processed.
ADDITION IS: 33

```
CREATE FUNCTION fact(x number)
RETURN number
IS
f number;
BEGIN
IF x=0 THEN
f := 1;
ELSE
f := x * fact(x-1);
END IF;
RETURN f;
END;
```

Function created.

```
DECLARE
num number;
factorial number;
BEGIN
num:= 6;
factorial := fact(num);
dbms_output.put_line(' Factorial '|| num || ' is ' || factorial);
END;
```

```
Statement processed.
Factorial 6 is 720
```

DROP FUNCTION fact;

Experiment – 15

Write PL/SQL program to implement Trigger on table.

```
CREATE TABLE INSTRUCTOR
(ID VARCHAR2(5),
NAME VARCHAR2(20) NOT NULL,
DEPT_NAME VARCHAR2(20),
SALARY NUMERIC(8,2) CHECK (SALARY > 29000),
    PRIMARY KEY (ID),
    FOREIGN KEY (DEPT_NAME) REFERENCES DEPARTMENT(DEPT_NAME)
ON DELETE SET NULL
)
```

Table created.

```
CREATE TABLE DEPARTMENT
(DEPT_NAME VARCHAR2(20),
BUILDING VARCHAR2(15),
BUDGET NUMERIC(12,2) CHECK (BUDGET > 0),
PRIMARY KEY (DEPT_NAME)
)
```

Table created.

```
insert into department values ('Biology', 'Watson', '90000')
```

1 row(s) inserted.

```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE UPDATE ON instructor
FOR EACH ROW
WHEN (NEW.ID = OLD.ID)
DECLARE
sal_diff number;
BEGIN
sal_diff := :NEW.salary - :OLD.salary;
dbms_output.put_line('Old salary: ' || :OLD.salary);
dbms_output.put_line('New salary: ' || :NEW.salary);
dbms_output.put_line('Salary difference: ' || sal_diff);
END;
```

Trigger created.

```
DECLARE
total_rows number(2);
BEGIN
UPDATE instructor
SET salary = salary + 5000;
IF sql%notfound THEN
dbms_output.put_line('no instructors updated');
ELSIF sql%found THEN
total_rows := sql%rowcount;
dbms_output.put_line( total_rows || ' instructors updated ');
END IF;
END;
```

Statement processed.
no instructors updated

Experiment – 16

Write PL/SQL program to implement Cursor on table.

```
CREATE TABLE customers(
ID NUMBER PRIMARY KEY,
NAME VARCHAR2(20) NOT NULL,
AGE NUMBER,
ADDRESS VARCHAR2(20),
SALARY NUMERIC(20,2))
```

Table created.

```
INSERT INTO customers VALUES(1,'Ramesh',23,'Allabad',25000)
```

1 row(s) inserted.

```
INSERT INTO customers VALUES(2, 'Suresh',22,'Kanpur',27000)
```

1 row(s) inserted.

```
INSERT INTO customers VALUES(3, 'Mahesh',24,'Ghaziabad',29000)
```

1 row(s) inserted.

```
DECLARE
total_rows number(2);
BEGIN
UPDATE customers
SET salary = salary + 5000;
IF sql%notfound THEN
dbms_output.put_line('no customers updated');
ELSIF sql%found THEN
total_rows := sql%rowcount;
dbms_output.put_line( total_rows || ' customers updated ');
END IF;
END;
```

Statement processed.
3 customers updated

```
DECLARE
c_id customers.id%type;
c_name customers.name%type;
c_addr customers.address%type;
CURSOR c_customers is
SELECT id, name, address FROM customers;
BEGIN
OPEN c_customers;
LOOP
FETCH c_customers into c_id, c_name, c_addr;
EXIT WHEN c_customers%notfound;
dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);
END LOOP;
CLOSE c_customers;
END;
```

Statement processed.
2 Suresh Kanpur
1 Ramesh Allabad
3 Mahesh Ghaziabad