EXPRIMENT-8

A-CREATING AND WORKING WITH INSERT TRIGGER USING BEFORE CLAUSE.

```
SQL> CREATE OR REPLACE TRIGGER emp_41_INSERT_TRIGGER
  2  BEFORE INSERT ON emp_41
  3  FOR EACH ROW
  4  BEGIN
  5      DBMS_OUTPUT.PUT_LINE('INSERTING NEW EMPLOYEE RECORD: ' || :NEW.ID);
  6  END;
  7  /

Trigger created.

SQL> insert into emp_41 values(106,'Shukla',1600,'admin',106);
INSERTING NEW EMPLOYEE RECORD: 106

1 row created.
```

B-CREATING AND WORKING WITH UPDATE TRIGGER USING BEFORE CLAUSE.

```
SQL> CREATE OR REPLACE TRIGGER emp_41_UPDATE_TRIGGER
  2  BEFORE UPDATE ON emp_41
  3  FOR EACH ROW
  4  BEGIN
  5     DBMS_OUTPUT.PUT_LINE('UPDATING EMPLOYEE RECORD: ' || :OLD.ID);
  6  END;
  7  /

Trigger created.

SQL> UPDATE  emp_41 SET SALARY=1200 WHERE ID=101;
UPDATING EMPLOYEE RECORD: 101

1 row updated.

SQL> SELECT * FROM emp_41;

        ID NAME                    SALARY
---------- -------------------- ----------
JOB                                                DEPT_ID
-------------------------------------------------- ----------
       101 Kundan                       1200
clerk                                                 105

       102 Gaurav                    2500000
hr                                                    104

       103 Nilesh                      25000
finance                                               103


        ID NAME                    SALARY
---------- -------------------- ----------
JOB                                                DEPT_ID
-------------------------------------------------- ----------
       104 komal                       20000
clerk                                                 101

       105 shubham                     23000
manager                                               102

       106 Shukla                       1600
admin                                                 106
```

```
6 rows selected.
```

C-CREATING AND WORKING WITH DELETE TRIGGER USING AFTER CLAUSE.

```
SQL> CREATE OR REPLACE TRIGGER emp_41_DELETE_TRIGGER
  2  AFTER DELETE ON emp_41
  3  FOR EACH ROW
  4  BEGIN
  5      DBMS_OUTPUT.PUT_LINE('EMPLOYEE RECORD DELETED: ' || :OLD.ID); --
Replace ID with the correct column name if necessary
  6  END;
  7  /

Trigger created.

SQL> DELETE FROM emp_41 WHERE ID=103;
EMPLOYEE RECORD DELETED: 103

1 row deleted.
```

EXPRIMENT-09

WRITE AN IMPLICIT AND EXPLICIT CURSOR TO COMPLETE THE TASK.

EXPLICIT CURSOR

```
SQL> DECLARE
  2      employee_name VARCHAR2(20);
  3      CURSOR CUR_EMP IS
  4          SELECT NAME FROM emp_41 WHERE ID = 105;
  5  BEGIN
  6      OPEN CUR_EMP;
  7
  8      LOOP
  9          FETCH CUR_EMP INTO employee_name;
 10                  EXIT WHEN CUR_EMP%NOTFOUND;
 11
 12          DBMS_OUTPUT.PUT_LINE(employee_name);
 13      END LOOP;
 14      IF CUR_EMP%NOTFOUND THEN
 15          DBMS_OUTPUT.PUT_LINE('RECORD NOT FOUND');
 16      END IF;
 17
 18      CLOSE CUR_EMP;
 19  END;
 20  /
shubham
RECORD NOT FOUND

PL/SQL procedure successfully completed.
```

IMPLICIT CURSOR

```
SQL> DECLARE
  2      V_NO NUMBER(8);
  3  BEGIN
  4      UPDATE emp_41 SET SALARY = SALARY + 1000  -- Adjust column name to
SALARY
```

```
  5      WHERE ID = 104;
  6
  7      V_NO := SQL%ROWCOUNT;
  8
  9      IF V_NO > 0 THEN
 10          DBMS_OUTPUT.PUT_LINE('SALARY OF ' || V_NO || ' EMPLOYEE UPDATED.');
 11      ELSE
 12          DBMS_OUTPUT.PUT_LINE('EMPLOYEE NOT FOUND');
 13      END IF;
 14  END;
 15  /
UPDATING EMPLOYEE RECORD: 104
SALARY OF 1 EMPLOYEE UPDATED.

PL/SQL procedure successfully completed.

SQL> SELECT * FROM emp_41;

        ID NAME                      SALARY
---------- -------------------- ----------
JOB                                                DEPT_ID
-------------------------------------------------- ----------
       101 Kundan                     1200
clerk                                                 105

       102 Gaurav                  2500000
hr                                                    104

       104 komal                     21000
clerk                                                 101


        ID NAME                      SALARY
---------- -------------------- ----------
JOB                                                DEPT_ID
-------------------------------------------------- ----------
       105 shubham                    23000
manager                                               102

       106 Shukla                      1600
admin                                                 106
```

EXPRIMENT-10

CREATE PACKAGES AND USE IT IN SQL BLACK TO COMPLETE THE TASK.

```
SQL> CREATE OR REPLACE PACKAGE EMPLOYEE_MANAGEMENT AS
  2      PROCEDURE UPDATE_SALARIES(PERCENTAGE IN NUMBER);
  3  END EMPLOYEE_MANAGEMENT;
  4  /

Package created.

SQL> CREATE OR REPLACE PACKAGE BODY EMPLOYEE_MANAGEMENT AS
  2      PROCEDURE UPDATE_SALARIES(PERCENTAGE IN NUMBER) AS
  3      BEGIN
  4          UPDATE emp_41
  5          SET SALARY = SALARY * (1 + PERCENTAGE / 100);
  6      END UPDATE_SALARIES;
  7  END EMPLOYEE_MANAGEMENT;
  8  /

Package body created.
```

```
SQL> DECLARE
  2      SALARY_INCREASE_PERCENTAGE NUMBER := 5;
  3  BEGIN
  4      EMPLOYEE_MANAGEMENT.UPDATE_SALARIES(SALARY_INCREASE_PERCENTAGE);
  5  END;
  6  /
UPDATING EMPLOYEE RECORD: 101
UPDATING EMPLOYEE RECORD: 102
UPDATING EMPLOYEE RECORD: 104
UPDATING EMPLOYEE RECORD: 105
UPDATING EMPLOYEE RECORD: 106

PL/SQL procedure successfully completed.

SQL> SELECT * FROM emp_41;

        ID NAME                      SALARY
---------- -------------------- ----------
JOB                                                DEPT_ID
-------------------------------------------------- ----------
       101 Kundan                     1260
clerk                                                 105

       102 Gaurav                  2625000
hr                                                    104

       104 komal                     22050
clerk                                                 101


        ID NAME                      SALARY
---------- -------------------- ----------
JOB                                                DEPT_ID
-------------------------------------------------- ----------
       105 shubham                    24150
manager                                               102

       106 Shukla                      1680
admin                                                 106
```

EXPRIMENT-11

WRITE A SQL BLOCK TO HANDLE EXCEPTION BY WRITING:
A. PREDEFINED EXCEPTIONS,
B. USER-DEFINED EXCEPTIONS,
C. REDECLARED PREDEFINED EXCEPTIONS

A-PREDEFINED EXCEPTIONS

```
SQL> DECLARE
  2      E_ID emp_41.ID%TYPE := 104;
  3      E_NAME emp_41.NAME%TYPE;
  4      E_JOB emp_41.JOB%TYPE;
  5  BEGIN
  6      SELECT NAME, JOB INTO E_NAME, E_JOB
  7      FROM emp_41
  8      WHERE ID = E_ID;
  9
 10      DBMS_OUTPUT.PUT_LINE('NAME: ' || E_NAME);
 11      DBMS_OUTPUT.PUT_LINE('JOB: ' || E_JOB);
 12
```

```
 13  EXCEPTION
 14      WHEN NO_DATA_FOUND THEN
 15          DBMS_OUTPUT.PUT_LINE('NO SUCH EMPLOYEE!');
 16      WHEN OTHERS THEN
 17          DBMS_OUTPUT.PUT_LINE('ERROR!');
 18  END;
 19  /
NAME: komal
JOB: clerk

PL/SQL procedure successfully completed.
```

```
B- USER-DEFINED EXCEPTIONS
--------------------------
SQL> DECLARE
  2      E_ID emp_41.ID%TYPE := 0;
  3      E_NAME emp_41.NAME%TYPE;
  4      E_JOB emp_41.JOB%TYPE;
  5
  6      EX_INVALID_ID EXCEPTION;
  7  BEGIN
  8      IF E_ID <= 0 THEN
  9          RAISE EX_INVALID_ID;
 10      ELSE
 11          SELECT NAME, JOB INTO E_NAME, E_JOB
 12          FROM emp_41
 13          WHERE ID = E_ID;
 14
 15          DBMS_OUTPUT.PUT_LINE('NAME: ' || E_NAME);
 16          DBMS_OUTPUT.PUT_LINE('JOB: ' || E_JOB);
 17      END IF;
 18
 19  EXCEPTION
 20      WHEN EX_INVALID_ID THEN
 21          DBMS_OUTPUT.PUT_LINE('ID MUST BE GREATER THAN ZERO!');
 22      WHEN NO_DATA_FOUND THEN
 23          DBMS_OUTPUT.PUT_LINE('NO SUCH EMPLOYEE!');
 24      WHEN OTHERS THEN
 25          DBMS_OUTPUT.PUT_LINE('ERROR!');
 26  END;
 27  /
ID MUST BE GREATER THAN ZERO!

PL/SQL procedure successfully completed.
```

```
C-REDECLARED PREDEFINED EXCEPTIONS
----------------------------------

SQL> DECLARE
  2      E_ID emp_41.ID%TYPE := 102;
  3      E_NAME emp_41.NAME%TYPE;
  4      E_JOB emp_41.JOB%TYPE;
  5      -- Redeclared predefined exceptions
  6      NO_DATA_FOUND EXCEPTION;
  7  BEGIN
  8      SELECT NAME, JOB INTO E_NAME, E_JOB
  9      FROM emp_41
 10      WHERE ID = E_ID;
 11
 12      DBMS_OUTPUT.PUT_LINE('NAME: ' || E_NAME);
 13      DBMS_OUTPUT.PUT_LINE('JOB: ' || E_JOB);
```

```
 14
 15  EXCEPTION
 16      WHEN NO_DATA_FOUND THEN  -- Handle the custom NO_DATA_FOUND exception
 17          DBMS_OUTPUT.PUT_LINE('NO SUCH EMPLOYEE!');
 18      WHEN OTHERS THEN
 19          DBMS_OUTPUT.PUT_LINE('ERROR!');
 20  END;
 21  /
NAME: Gaurav
JOB: hr

PL/SQL procedure successfully completed.
```

EXPRIMENT-12

CREATE NESTED TABLES AND WORK WITH NESTED TABLES.

```
SQL> CREATE OR REPLACE TYPE ADD_TYPE AS TABLE OF VARCHAR2(50);
  2  /

Type created.

SQL> CREATE TABLE EDB (
  2      EMP_NO NUMBER(4) PRIMARY KEY,
  3      E_NAME VARCHAR2(8),
  4      DEPT_NO NUMBER(2) DEFAULT 10,
  5      ADDRESSES ADD_TYPE
  6  ) NESTED TABLE ADDRESSES STORE AS NESTED_TAB_ADD;

Table created.

SQL> INSERT INTO EDB (EMP_NO, E_NAME, DEPT_NO, ADDRESSES)
  2  VALUES (1, 'RAM', 10, ADD_TYPE('103, NAVGHAR GAON', 'BHAYANDER'));

1 row created.

SQL> INSERT INTO EDB (EMP_NO, E_NAME, DEPT_NO, ADDRESSES)
  2  VALUES (4, 'JATIN', 20, ADD_TYPE('123 MAIN ST.'));

1 row created.

SQL> SELECT * FROM EDB;

 EMP_NO E_NAME      DEPT_NO       ADDRESSES
---------- -------- ----------    ------------------------------
         1 RAM              10    103, NAVGHAR GAON, BHAYANDER
         4 JATIN            20    123 MAIN ST.
```