Islamic University of Technology
Lab Task 10
CSE 4308 - DBMS Lab

Submitted To :

MD Bakhtiar Hasan
Ast. Professor, CSE Department
Islamic University of Technology

Zannatun Naim Sristy
Lecturer, CSE Department
Islamic University of Technology

Submitted By :

Sian Ashsad
ID : 200042151
Prog. : SWE
Dept. : CSE

Task 1 :

Working code :

```
DECLARE
    TOTAL_ROWS NUMBER (2);
BEGIN
    UPDATE INSTRUCTOR
        SET SALARY = SALARY + (SALARY * 0.1)
        WHERE SALARY < 75000 ;
    IF SQL % NOTFOUND THEN
        DBMS_OUTPUT . PUT_LINE ( 'No instructor satisfied the condition ');
    ELSIF SQL % FOUND THEN
        TOTAL_ROWS := SQL % ROWCOUNT ;
    DBMS_OUTPUT . PUT_LINE ( TOTAL_ROWS || ' instructors incremented ');
    END IF;
END ;
/
```

Report :

Analysis : In this task we need to write PL/SQL statements to provide a 10% increment to the instructors that get a salary less than 75000 and then show the number of instructors that got increment.

Explanation of solution : A NUMBER variable TOTAL_ROWS is declared at the very beginning of the code. An SQL query is performed which updates the Instructor table and sets the salary by incrementing it by 10% if Salary is less than 75000.
Afterwards IF ELSIF conditionals are set to check if such instructors are found or not. This is done through the NOTFOUND and FOUND implicit cursors. If the SQL query affects at least one row, then the ROWCOUNT function is used to see how many were affected. The count is stored in the TOTAL_ROWS variable and output in the console. This is the number of instructors that got incremented.

Findings : The method of finding if any rows were affected or not was new to me. The usage of the ROWCOUNT function made counting affected rows simple.

Problems : At first, I implemented a query which counted the number of instructors whose salary was less than 75000 and showed that to the console. This required using two SQL queries instead of one. The order of these SQL queries could also affect the result. Therefore using cursors instead is the wiser implementation choice.

<u>Task 2 :</u>

<u>Working code :</u>

```
CREATE OR REPLACE
PROCEDURE PRINT_TIME_SLOT
AS
BEGIN

    FOR i IN (SELECT T.TIME_SLOT_ID, T.DAY, T.start_hr, T.start_min ,T.end_hr,
T.end_min FROM INSTRUCTOR I, TEACHES E, SECTION S,TIME_SLOT T WHERE I.ID = E.ID
AND
                    E.COURSE_ID = S.COURSE_ID AND E.SEC_ID = S.SEC_ID AND
E.SEMESTER = S.SEMESTER AND E.YEAR = S.YEAR AND
                    S.TIME_SLOT_ID = T.TIME_SLOT_ID) LOOP
        DBMS_OUTPUT . PUT_LINE (i.TIME_SLOT_ID || ' ' || i.DAY || ' ' ||
i.start_hr || ' ' || i.end_hr);
    END LOOP;

END;
/
```

<u>Report :</u>

Analysis :

Explanation of solution :

Findings :

Problems :

Task 3 :

Working code :

```
CREATE OR REPLACE
PROCEDURE FIND_ADVISORS(NUM IN NUMBER)
AS
ROW NUMBER(5);
BEGIN
    SELECT MAX(ROWNUM) INTO ROW
    FROM (SELECT I_ID, COUNT(S_ID) AS S_COUNT FROM ADVISOR GROUP BY I_ID ORDER BY
S_COUNT DESC);

    IF(NUM>ROW) THEN
        DBMS_OUTPUT . PUT_LINE ('Input exceeds number of entries');
        RETURN;
    END IF;

    FOR i IN (SELECT * FROM (SELECT I_ID, COUNT(S_ID) AS S_COUNT FROM ADVISOR
GROUP BY I_ID ORDER BY S_COUNT DESC) WHERE ROWNUM<=NUM) LOOP
        DBMS_OUTPUT . PUT_LINE (i.I_ID || ' ' || i.S_COUNT);
    END LOOP;

END;
/
```

Report :

Analysis :

Explanation of solution :

Findings :

Problems :

Task 4 :

Working code :

```
CREATE SEQUENCE STUDENT_SEQ
MINVALUE 1
MAXVALUE 9999
START WITH 1
INCREMENT BY 1
CACHE 20;

CREATE OR REPLACE
TRIGGER STUDENT_ID_GENERATOR
BEFORE INSERT ON STUDENT
FOR EACH ROW
BEGIN
    :NEW.ID := STUDENT_SEQ . NEXTVAL ;
END ;
/
```

Report :

Analysis : Using PL/SQL we need to create a trigger that automatically generates IDs for students when data is inserted into the STUDENT table.

Explanation of solution : A sequence is created (`STUDENT_SEQ`) which has the MINVALUE of 1 and MAXVALUE of 9999. The sequence starts with 1 and increments by 1. It has a cache value of 20.

A trigger `STUDENT_ID_GENERATOR` is created which activates before an insert operation on the student table. This trigger replaces the ID value of the insert values with the incremented value of the `STUDENT_SEQ` sequence.

Findings : `:NEW` is used to access the values of the input. In oracle 11g, it is possible to directly put the incremented value of the sequence into the input value.

Task 5 :

Working code :

```
CREATE OR REPLACE
    TRIGGER ADVISOR_ASSIGNER
    AFTER INSERT ON STUDENT
    FOR EACH ROW
DECLARE
    INS_ID INSTRUCTOR.ID% TYPE ;
BEGIN
    SELECT ID INTO INS_ID
    FROM(
        SELECT ID
        FROM INSTRUCTOR I
        WHERE I.DEPT_NAME = :NEW.DEPT_NAME
    )
    WHERE ROWNUM<=1;
    INSERT INTO ADVISOR VALUES(INS_ID, :NEW.ID);

END ;
/
```

Report :

Analysis :

Explanation of solution :

Findings :

Problems :