

NAME: NAZIA KARIM KHAN OISHEE

STUDENT ID: 200042137

COURSE CODE: CSE 4308

SUBJECT: DATABASE MANAGEMENT SYSTEMS LAB

Introduction

Our task in this lab was to use PL to conduct sql queries. PL stands for Procedural Language.

Procedural Language for Structured Query Language (PL/SQL) lets us write code in such a way that it can be deployed in the database nearest to data. PL/SQL simplifies application development, optimizes execution, and improves resource utilization in the database. The language is a case-insensitive programming language.

Our task was to mainly implement triggers and procedures. A procedure is a module that performs one or more actions. A trigger is a statement that is executed automatically by the system as a side effect of a modification to the database.

Task

We were given a diagram of the university database. Based on that we were to perform sql queries.

1. Provide a 10% increment to the instructors that get a salary less than 75000. Show the number of instructors that got incremented.

Solution:

```
SET SERVEROUTPUT ON SIZE 1000000
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 updated ');
END IF;
END ;
/

```

Explanation:

To complete this task ,I first conducted a query to update the salary of instructors under the given condition. Then with SQL% NOTFOUND I checked if any rows were affected. SQL% NOTFOUND would return true if no rows were affected.Else SQL% FOUND would be true. Then I counted the number of affected rows which means the number of instructors that got incremented using SQL% ROWCOUNT .

2. Write a procedure for printing the time_slot of every teacher.

Solution:

```

SET SERVEROUTPUT ON SIZE 1000000
CREATE OR REPLACE
PROCEDURE TIME_SLOT
AS
BEGIN

    FOR ROW IN (SELECT INSTRUCTOR.ID AS ID,INSTRUCTOR.NAME AS
NAME,TIME_SLOT.TIME_SLOT_ID AS TIME_SLOT_ID, TIME_SLOT.DAY AS DAY,
TIME_SLOT.start_time AS start, TIME_SLOT.end_time AS end, T
    FROM INSTRUCTOR, TEACHES, SECTION ,TIME_SLOT WHERE INSTRUCTOR.ID =
TEACHES.ID AND TEACHES.COURSE_ID = SECTION.COURSE_ID AND TEACHES.SEC_ID =
SECTION.SEC_ID AND TEACHES.SEMESTER = SECTION.SEMESTER AND TEACHES.YEAR =
SECTION.YEAR AND SECTION.TIME_SLOT_ID = TIME_SLOT.TIME_SLOT_ID) LOOP
        dbms_output.put_line(ROW.ID || ' ' ||ROW.NAME || ' ' ||ROW.TIME_SLOT_ID
|| ' ' || ROW.DAY || ' ' || ROW.start||' ' ||ROW.end);
    END LOOP;

```

```

END;
/

BEGIN
    TIME_SLOT;
END;
/

```

Explanation:

In the given scenario the Instructor entity was connected to Teaches. Teaches was connected to Section. Section was connected to Time_Slot. For printing the time_slot of every teacher/instructor I wrote a query to select each instructor's time slot regarding information combining these four entities. Then I passed the query in a for loop . The loop would repeat as many times as the number of rows in the resulting query and print the time slots.

3. Write a procedure to find the N advisers and their details who has the highest number of students under their advising.

Solution:

```

SET SERVEROUTPUT ON SIZE 1000000
CREATE OR REPLACE
PROCEDURE AdvisorWithHighestStudent(N IN NUMBER)
AS
    MAX_ROW NUMBER;
BEGIN
    SELECT count(*) INTO MAX_ROW FROM (SELECT * FROM (SELECT
i_id,count(*)studentCount FROM advisor group by i_id order by studentCount
desc)StudentTable JOIN instructor on countTable.i_id=instructor.ID);
    IF N>=MAX_ROW THEN
        DBMS_OUTPUT.put_line('N exceeds number of instructors');
        RETURN;
    END IF;

```

```

        FOR ROW IN (SELECT * FROM (SELECT * FROM (SELECT i_id,count(*)studentCount
FROM advisor group by i_id order by studentCount desc)StudentTable JOIN
instructor on countTable.i_id=instructor.ID) WHERE ROWNUM<=N) LOOP
            DBMS_OUTPUT.PUT_LINE(ROW.ID || ' ' || ROW.name || ' ' || ROW.dept_name
|| ' ' || ROW.salary || ' ' || ROW.studentCount);
        END LOOP;
END;
/

DECLARE
    N NUMBER;
BEGIN
    N:='&number';
    AdvisorWithHighestStudent(N);
END;
/

```

Explanation:

In this task I took user input for N number. Then I performed a query using Instructor and Advisor entity to count the number of students under each instructor and to collect information about the instructors. If the input is bigger than the number of instructors then an error message will be displayed. Otherwise the information will be displayed.

4. Create a trigger that automatically generates IDs for students when we insert data into the STUDENT table.

Solution:

```

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
DECLARE
    NEW_ID STUDENT.ID% TYPE ;
BEGIN
SELECT STUDENT_SEQ . NEXTVAL INTO NEW_ID
FROM DUAL ;
:NEW.ID := NEW_ID ;
END ;
/
insert into student (name,dept_name,tot_cred)values ('nazia', 'Comp. Sci.',
100);
insert into student (name,dept_name,tot_cred)values ('oishee', 'Comp. Sci.',
100);
select * from student where name ='nazia';
select * from student where name ='oishee';

```

Explanation:

To complete the task I used a sequence which will generate ID for students sequentially. Sequences are counting structures that maintain a persistent awareness of their current value. A sequence caches values by groups of 20 by default.

For an INSERT trigger, :old. contains no value, and :NEW. contains the new values. The STUDENT_ID_GENERATOR trigger will be executed before inserting any data into the student table. It will assign the next value of sequence to the newly generated student's ID.

5. Create a trigger that will automatically assign an advisor to a newly admitted student of his/her own department.

Solution:

```
CREATE OR REPLACE
TRIGGER AssignAdvisor
AFTER INSERT ON student
FOR EACH ROW
DECLARE
Instructor_ID instructor.ID% TYPE ;
BEGIN
SELECT MAX(ID) INTO Instructor_ID FROM (SELECT ID FROM instructor WHERE
:NEW.dept_name=instructor.dept_name) WHERE ROWNUM=1;
INSERT INTO advisor VALUES (:NEW.ID,Instructor_ID);
END;
/
```


Explanation:

For this task I created a trigger named AssignAdvisor. This trigger will be fired after inserting values into the student. I performed a query to select an instructor from the same department as the newly inserted student, selected the top instructor from the query and inserted this instructor along with the student into the advisor entity. Thus every student will be assigned an instructor after admission automatically.

Problems I faced during the task:

Trigger was a new concept to implement for me. I was familiar with the procedure but not so skilled. I struggled a bit to write the syntaxes properly especially while doing the fourth task where I used sequence.

Other than that, time management was an issue for me. It took me a bit of time to come up with the idea of how to do the tasks. So I couldn't finish the whole task



in lab hours. I completed the first, fourth and fifth task during the lab and the rest afterwards.