Course Number and Name: CSE 4308 Database Management Systems Lab	
Student Name: Nafisa Maliyat	**Student ID:** 200042133
Report Submission Date: 12 November, 2022	**Name of Lab Instructor:** Md. Bakhtiar Hasan, Lecturer, CSE Zannatun Naim Sristy, Lecturer, CSE

Lab 8: PL/SQL

Overview:

This lab provided us a manual for the basics of PL/SQL triggers and cursors. Using this knowledge, different queries had to be performed on the tables of a university schema given with the lab task.

On the next pages, I have mentioned the following :

- the problem statement
- analysis of the problem,
- SQL written to solve the problem,
- problems faced (if any) during solution of the tasks.

Task 1:

Problem Statement:

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

Analysis of the problem:

This requires updating instructor and setting a condition using a where clause. Next, the cursor SQL%NOTFOUND is used to find if there were any rows changed. SQL%FOUND could also be used similarly. If there were rows changed, the number of rows changed was printed using SQL%ROWCOUNT. Otherwise, a statement stating no rows were changed was printed.

Any problems faced and how it was solved:

There were no problems faced since the query was straightforward.

Code:

```
CREATE OR REPLACE PROCEDURE UPDATE_INSTRUCTOR_SALARY
AS
TOTAL_ROWS NUMBER (2);
BEGIN
UPDATE INSTRUCTOR
SET SALARY = SALARY * 1.1
WHERE SALARY < 75000 ;
IF SQL% NOTFOUND THEN
DBMS_OUTPUT . PUT_LINE ( 'No instructor satisfied the condition ');
ELSE
DBMS_OUTPUT . PUT_LINE ( SQL%ROWCOUNT || ' instructors updated ');
END IF;
END ;
/

BEGIN
UPDATE_INSTRUCTOR_SALARY;
END;
/
```

Results:

5 instructors updated

Task 2:

Problem Statement:

Write a procedure for printing time_slot of every teacher.

Analysis of the problem:

The required information comes from instructor table and time_slot table. To get the required information, natural join was used on the tables connecting them - teaches and section. Natural join was possible due to the attribute naming being identical in different tables (ID attribute of instructor and ID attribute in teaches was named the same). Otherwise, multiple where statements would have been used to achieve same results.

Any problems faced and how it was solved:

There were no problems faced since the query was straightforward.

Code:

```
CREATE OR REPLACE PROCEDURE SHOW_TIME_SLOT
AS
BEGIN
FOR i IN (SELECT NAME, ID, DAY, START_HR, START_MIN, END_HR, END_MIN
        FROM INSTRUCTOR NATURAL JOIN TEACHES NATURAL JOIN SECTION NATURAL
JOIN TIME_SLOT
        ) LOOP
DBMS_OUTPUT.PUT_LINE('Name: ' || i.NAME || ', ID: ' || i.ID || ', Day: ' ||
i.DAY || ', Start time: ' || i.START_HR ||
                        ' : ' || i.START_MIN || ', End time: ' || i.END_HR
|| ' : '|| i.END_MIN);
END LOOP;
END;
/
BEGIN
SHOW_TIME_SLOT;
END;
/
```

Results:

```
Name: Einstein, ID: 22222, Day: M, Start time: 8 : 0, End time: 8 : 50
Name: Srinivasan, ID: 10101, Day: M, Start time: 8 : 0, End time: 8 : 50
Name: Brandt, ID: 83821, Day: M, Start time: 8 : 0, End time: 8 : 50
Name: Crick, ID: 76766, Day: M, Start time: 8 : 0, End time: 8 : 50
Name: Einstein, ID: 22222, Day: W, Start time: 8 : 0, End time: 8 : 50
Name: Srinivasan, ID: 10101, Day: W, Start time: 8 : 0, End time: 8 : 50
```

```
Name: Brandt, ID: 83821, Day: W, Start time: 8 : 0, End time: 8 : 50
Name: Crick, ID: 76766, Day: W, Start time: 8 : 0, End time: 8 : 50
Name: Einstein, ID: 22222, Day: F, Start time: 8 : 0, End time: 8 : 50
Name: Srinivasan, ID: 10101, Day: F, Start time: 8 : 0, End time: 8 : 50
Name: Brandt, ID: 83821, Day: F, Start time: 8 : 0, End time: 8 : 50
Name: Crick, ID: 76766, Day: F, Start time: 8 : 0, End time: 8 : 50
Name: Wu, ID: 12121, Day: M, Start time: 9 : 0, End time: 9 : 50
Name: Katz, ID: 45565, Day: M, Start time: 9 : 0, End time: 9 : 50
Name: Crick, ID: 76766, Day: M, Start time: 9 : 0, End time: 9 : 50
Name: Wu, ID: 12121, Day: W, Start time: 9 : 0, End time: 9 : 50
Name: Katz, ID: 45565, Day: W, Start time: 9 : 0, End time: 9 : 50
Name: Crick, ID: 76766, Day: W, Start time: 9 : 0, End time: 9 : 50
Name: Wu, ID: 12121, Day: F, Start time: 9 : 0, End time: 9 : 50
Name: Katz, ID: 45565, Day: F, Start time: 9 : 0, End time: 9 : 50
Name: Crick, ID: 76766, Day: F, Start time: 9 : 0, End time: 9 : 50
Name: El Said, ID: 32343, Day: M, Start time: 11 : 0, End time: 11 : 50
Name: Kim, ID: 98345, Day: M, Start time: 11 : 0, End time: 11 : 50
Name: Brandt, ID: 83821, Day: M, Start time: 11 : 0, End time: 11 : 50
Name: El Said, ID: 32343, Day: W, Start time: 11 : 0, End time: 11 : 50
Name: Kim, ID: 98345, Day: W, Start time: 11 : 0, End time: 11 : 50
Name: Brandt, ID: 83821, Day: W, Start time: 11 : 0, End time: 11 : 50
Name: El Said, ID: 32343, Day: F, Start time: 11 : 0, End time: 11 : 50
Name: Kim, ID: 98345, Day: F, Start time: 11 : 0, End time: 11 : 50
Name: Brandt, ID: 83821, Day: F, Start time: 11 : 0, End time: 11 : 50
Name: Mozart, ID: 15151, Day: M, Start time: 13 : 0, End time: 13 : 50
Name: Srinivasan, ID: 10101, Day: M, Start time: 13 : 0, End time: 13 :
50
Name: Mozart, ID: 15151, Day: W, Start time: 13 : 0, End time: 13 : 50
Name: Srinivasan, ID: 10101, Day: W, Start time: 13 : 0, End time: 13 :
50
Name: Mozart, ID: 15151, Day: F, Start time: 13 : 0, End time: 13 : 50
Name: Srinivasan, ID: 10101, Day: F, Start time: 13 : 0, End time: 13 :
50
Name: Brandt, ID: 83821, Day: T, Start time: 10 : 30, End time: 11 : 45
Name: Brandt, ID: 83821, Day: R, Start time: 10 : 30, End time: 11 : 45
Name: Katz, ID: 45565, Day: T, Start time: 14 : 30, End time: 15 : 45
Name: Katz, ID: 45565, Day: R, Start time: 14 : 30, End time: 15 : 45
Name: Srinivasan, ID: 10101, Day: W, Start time: 10 : 0, End time: 12 :
30

PL/SQL procedure successfully completed.
```

Task 3:

Problem Statement:

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

Analysis of the problem:

This procedure takes a number N as an in parameter. First it is checked to see if this N is greater than the number of entries in advisor table. If so, a message is printed and a return statement is written so the rest of the procedure will not execute.
Next a query fetches N rows of the advisor grouped by instructor ID that is sorted according to the number of students and a loop is used to print it.

Any problems faced and how it was solved:

There were no problems faced since the query was straightforward.

Code:

```
CREATE OR REPLACE PROCEDURE SORTED_ADVISOR_LIST (N IN NUMBER)
AS
MAX_ROW NUMBER;
BEGIN
SELECT COUNT(I_ID) INTO MAX_ROW
FROM (SELECT I_ID
        FROM ADVISOR
        GROUP BY I_ID);

IF(N > MAX_ROW) THEN
    DBMS_OUTPUT . PUT_LINE ('N IS TOO LARGE!');
    RETURN;
END IF;

FOR i IN (SELECT * FROM
        (SELECT *
        FROM INSTRUCTOR, (SELECT I_ID
                        FROM (SELECT I_ID, COUNT(S_ID)
                                FROM ADVISOR
                                GROUP BY I_ID
                                ORDER BY COUNT(S_ID) DESC
                            ))
        WHERE I_ID = INSTRUCTOR.ID
        AND INSTRUCTOR.ID IN (SELECT I_ID FROM ADVISOR))
        WHERE ROWNUM <=N)LOOP
```

```
        DBMS_OUTPUT.PUT_LINE('ID: ' || i.ID || ', Name: ' || i.NAME || ',
DEPT_NAME: ' || i.DEPT_NAME || ', SALARY: ' || i.SALARY);
END LOOP;
END;
/
```

```
BEGIN
        SORTED_ADVISOR_LIST(4);
        SORTED_ADVISOR_LIST(7);
END;
/
```

Results:

```
ID: 98345, Name: Kim, DEPT_NAME: Elec. Eng., SALARY: 80000
ID: 45565, Name: Katz, DEPT_NAME: Comp. Sci., SALARY: 75000
ID: 22222, Name: Einstein, DEPT_NAME: Physics, SALARY: 95000
ID: 76543, Name: Singh, DEPT_NAME: Finance, SALARY: 80000
N IS TOO LARGE!

PL/SQL procedure successfully completed.
```

Task 4:

Problem Statement:

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

Analysis of the problem:

First a sequence STUDENT_SEQ is created. Next, a trigger STUDENT_ID_GENERATOR is written that assigns a student ID equivalent to the next number on the sequence before insertion of a row in student table. For testing purposes, a value is inserted and the student table is checked to see if the change took place.

Any problems faced and how it was solved:

Since ID was not entered at time of inserting data into student table, there was an error stating 'not enough values'. This was solved by giving the column names explicitly in the insertion statement.

Code:

```
DROP SEQUENCE STUDENT_SEQ;
CREATE SEQUENCE STUDENT_SEQ
MINVALUE 1
MAXVALUE 10000
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 ;
/

BEGIN
INSERT INTO STUDENT (NAME, DEPT_NAME, TOT_CRED) VALUES ('Nafisa',
'Biology', 110);
END;
/
SELECT * FROM STUDENT;
```

Results:

```
ID    NAME                 DEPT_NAME            TOT_CRED
----- -------------------- -------------------- ----------
00128 Zhang                Comp. Sci.                  102
12345 Shankar              Comp. Sci.                   32
19991 Brandt               History                      80
23121 Chavez               Finance                     110
44553 Peltier              Physics                      56
45678 Levy                 Physics                      46
54321 Williams             Comp. Sci.                   54
55739 Sanchez              Music                        38
70557 Snow                 Physics                       0
76543 Brown                Comp. Sci.                   58
76653 Aoi                  Elec. Eng.                   60

ID    NAME                 DEPT_NAME            TOT_CRED
----- -------------------- -------------------- ----------
98765 Bourikas             Elec. Eng.                   98
98988 Tanaka               Biology                     120
1     Nafisa               Biology                     110 <-------

14 rows selected.
```

Task 5:

Problem Statement:

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

Analysis of the problem:

This required an after trigger that would be fired on event of an insertion to a student table. A variable is declared to store the result of the query selecting a random ID from instructor table. This was done by ordering by DBMS_RANDOM.RANDOM. Additionally, it was also checked that the instructor was from the same department using a where clause. This variable is then inserted into advisor table along the newly inserted student' s ID.

Any problems faced and how it was solved:

There were some confusion regarding the fact if the random ID generated would be from instructor table or from the advisor table. The problem was then solved assuming all instructors could be advisors.

Code:

```
DROP TRIGGER ASSIGN_ADVISOR;
CREATE OR REPLACE TRIGGER ASSIGN_ADVISOR
AFTER INSERT ON STUDENT
FOR EACH ROW
DECLARE
RANDOM_ID ADVISOR.I_ID%TYPE;
BEGIN
SELECT ID INTO RANDOM_ID
FROM (SELECT ID FROM INSTRUCTOR WHERE DEPT_NAME = :NEW.DEPT_NAME ORDER BY
DBMS_RANDOM.RANDOM)
WHERE ROWNUM<=1;
INSERT INTO ADVISOR VALUES(:NEW.ID, RANDOM_ID);
END;
/
BEGIN
INSERT INTO STUDENT (NAME, DEPT_NAME, TOT_CRED) VALUES ('SHANTA', 'Comp.
Sci.', 43);
END;
/
SELECT * FROM ADVISOR;
```

Results:

```
S_ID  I_ID
----- -----
00128 45565
12345 10101
23121 76543
44553 22222
45678 22222
76543 45565
76653 98345
98765 98345
98988 76766
2     45565 <-----

10 rows selected.
```