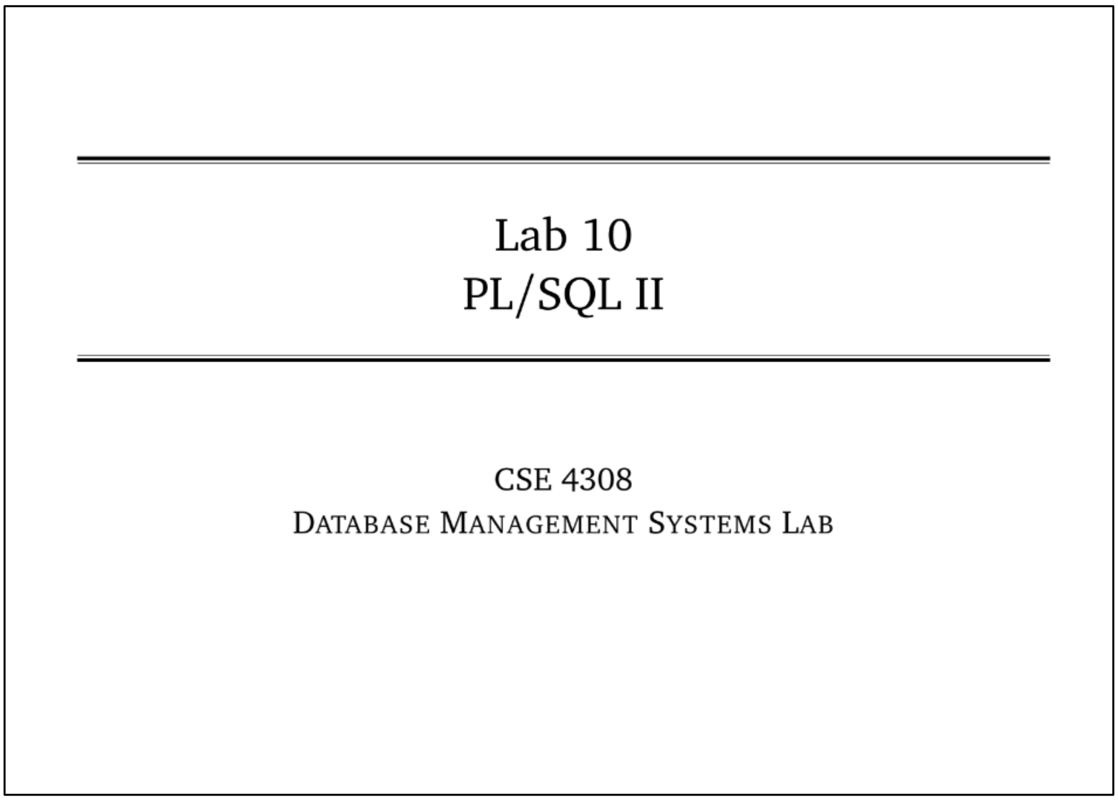
LAB REPORT



Lab 10

PL/SQL II

CSE 4308

DATABASE MANAGEMENT SYSTEMS LAB

NAME: CHOWDHURY ASHFAQ

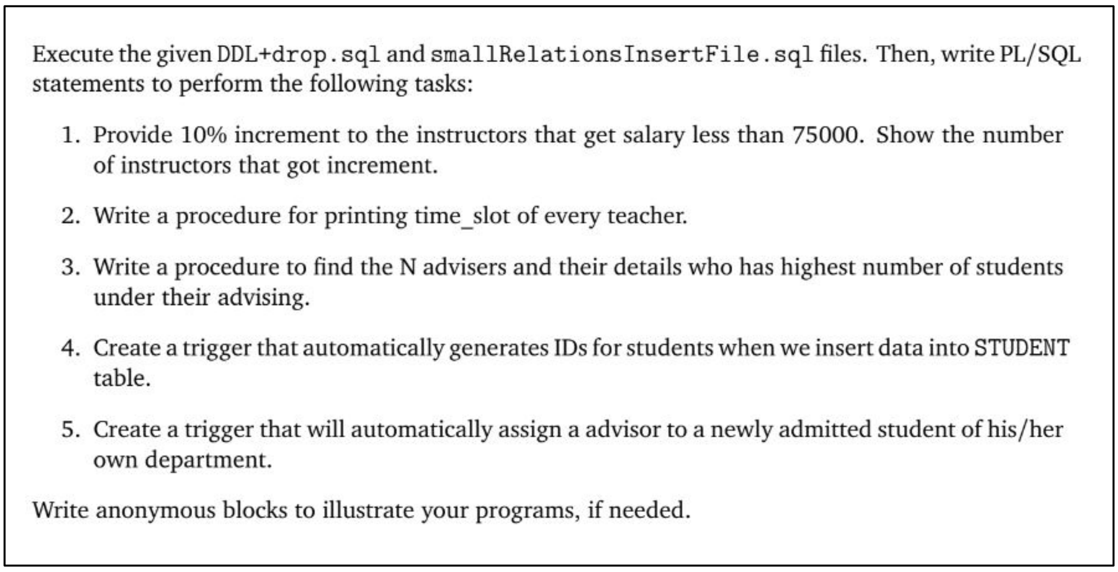
STUDENT ID: 200042123

PROGRAM: SWE

GROUP: 1A

DATE: 13/11/22

Tasks:



Execute the given DDL+drop.sql and smallRelationsInsertFile.sql files. Then, write PL/SQL statements to perform the following tasks:

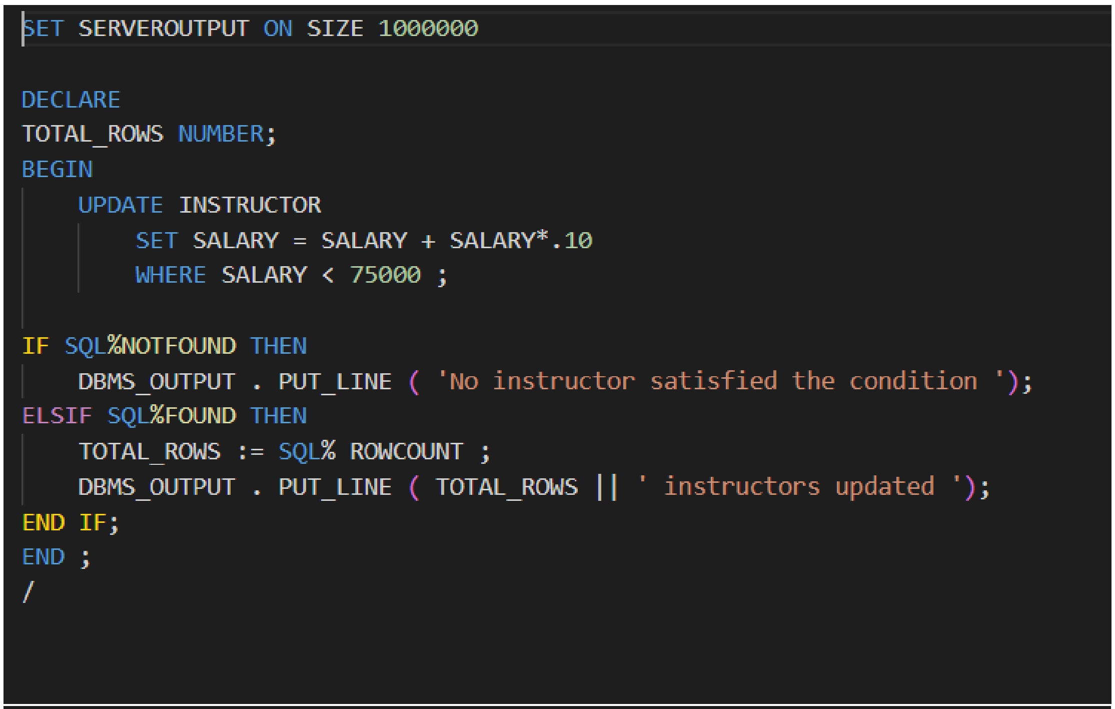
1. Provide 10% increment to the instructors that get salary less than 75000. Show the number of instructors that got increment.
2. Write a procedure for printing time_slot of every teacher.
3. Write a procedure to find the N advisers and their details who has highest number of students under their advising.
4. Create a trigger that automatically generates IDs for students when we insert data into STUDENT table.
5. Create a trigger that will automatically assign a advisor to a newly admitted student of his/her own department.

Write anonymous blocks to illustrate your programs, if needed.

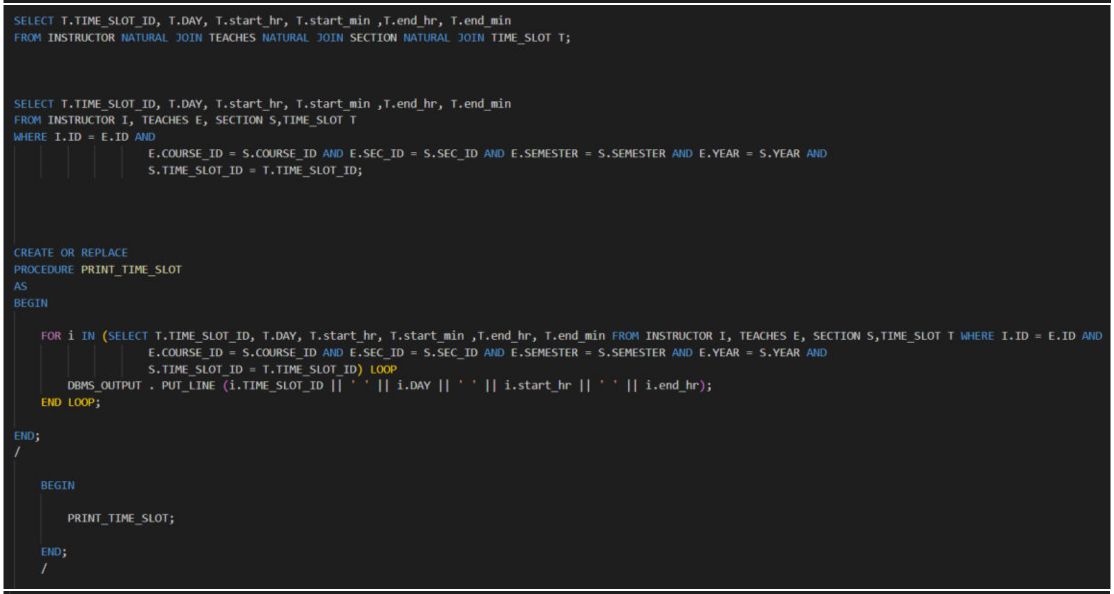
Analysis of the problem:

We had to write some queries in PL/SQL.

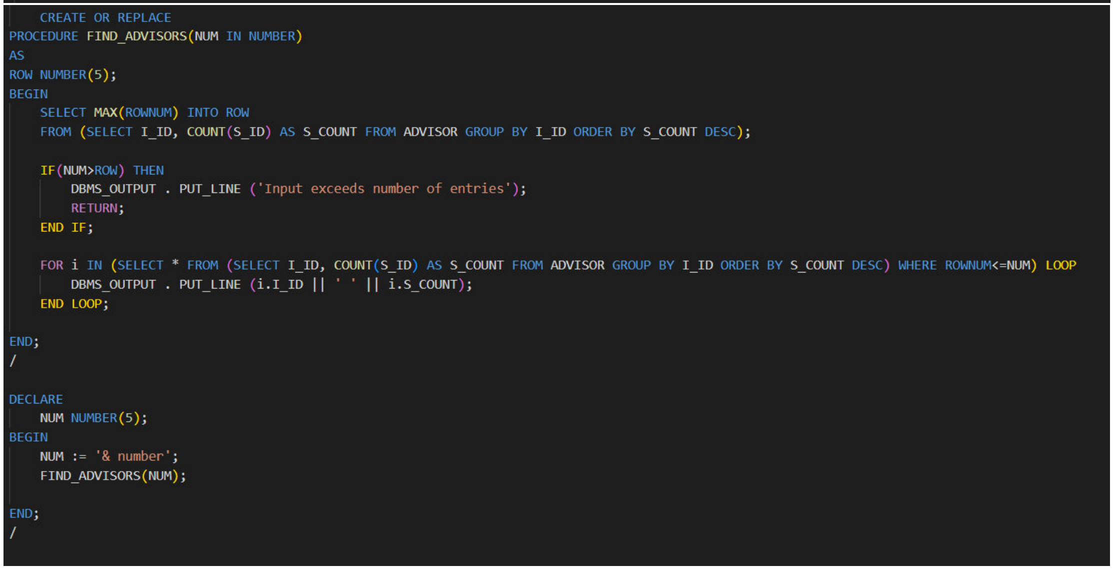
Solution:



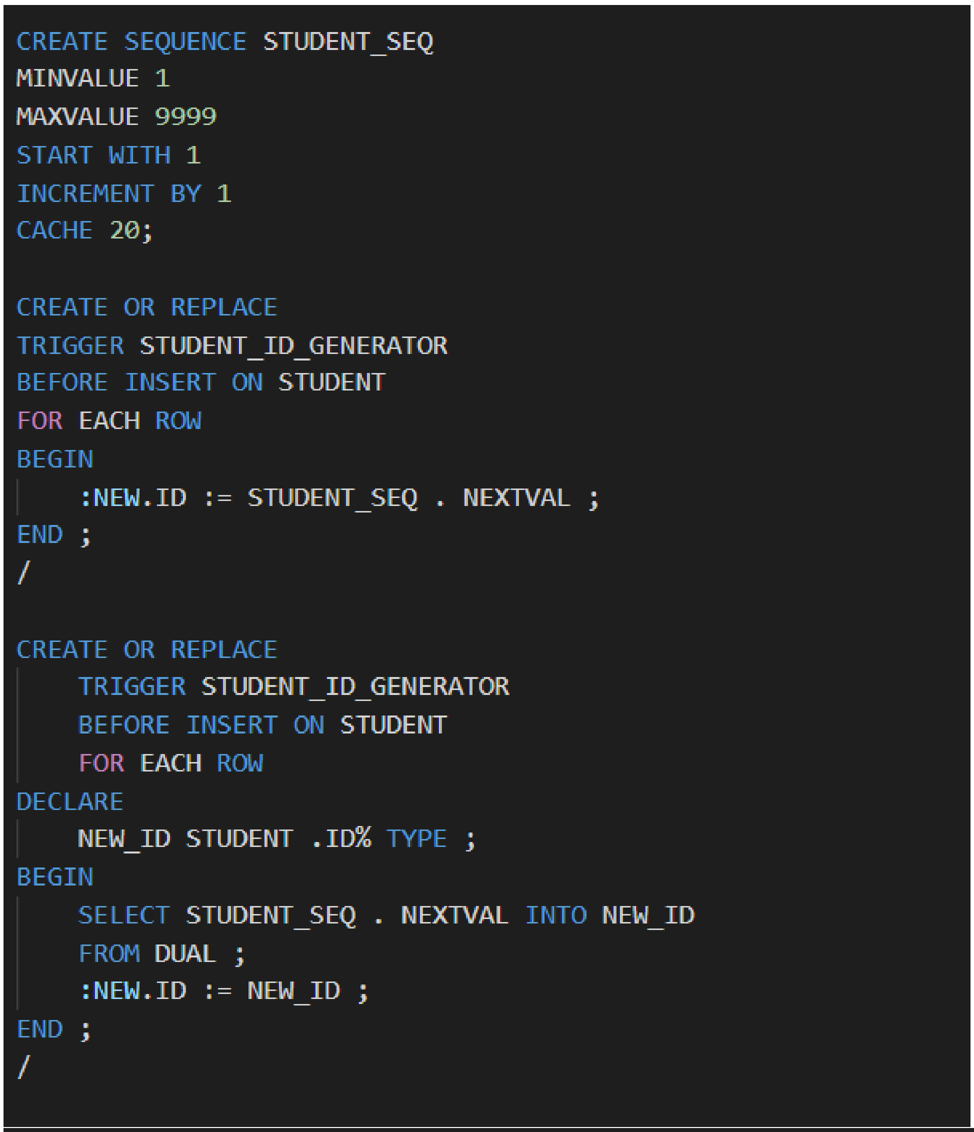
```
SET SERVEROUTPUT ON SIZE 1000000

DECLARE
TOTAL_ROWS NUMBER;
BEGIN
    UPDATE INSTRUCTOR
        SET SALARY = SALARY + SALARY*.10
        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 ;
/
```



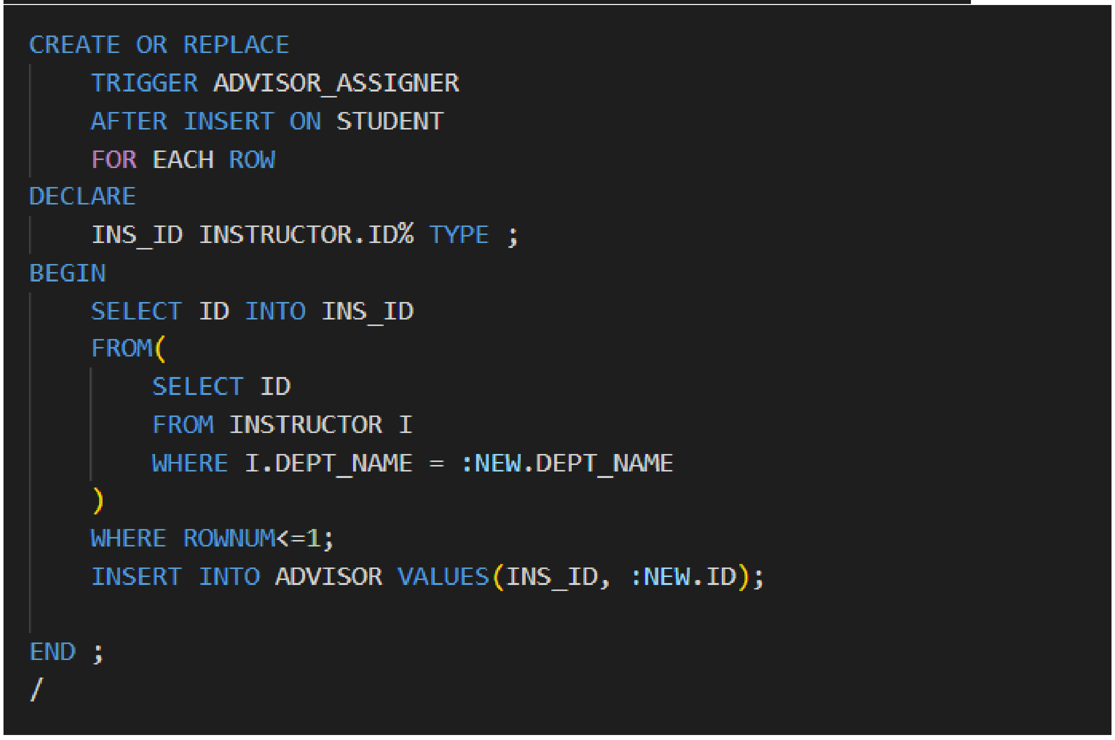
```
SELECT T.TIME_SLOT_ID, T.DAY, T.start_hr, T.start_min ,T.end_hr, T.end_min
FROM INSTRUCTOR NATURAL JOIN TEACHES NATURAL JOIN SECTION NATURAL JOIN TIME_SLOT T;


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;



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;
/

    BEGIN

        PRINT_TIME_SLOT;

    END;
    /
```



```
    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;
/

DECLARE
    NUM NUMBER(5);
BEGIN
    NUM := '& number';
    FIND_ADVISORS(NUM);

END;
/
```



```
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 ;
/

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 ;
/
```



```
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 ;
/
```