**CC-215-L: Database Systems Lab**

*Faculty of Computing & Information Technology*



**BS(CS) Morning - Fall 2021, Semester Spring 2023**

**LAB – 11**

**Course & Lab Instructor:** Dr. Asif Sohail

**Objectives:**

1. PL/SQL Revision
2. Procedures & Functions Revision
3. Triggers

*Allowed time: 60 mins.*

***Instructions:***

1. Gossips are not allowed.
2. Teacher assistants are for your help, so be nice with them. Respect them as they are teaching you. Raise your hands if you have some problem and need help from TA. Avoid calling them by raising your voice and disturbing the environment of Lab.
3. TA may deduct your marks for any kind of ill-discipline or misconduct from your side.
4. Evaluation will be considered final and you cannot debate for the marks. So, focus on performing the tasks when the time is given to you.
5. Paste the query as well as result table screenshot as a result of each task

**Sample:**

**Display All the Employees from emp table**

**Solution:**

**Select \* from emp**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **EMPNO** | **ENAME** | **JOB** | **MGR** | **HIREDATE** | **SAL** | **COMM** | **DEPTNO** |
| 7369 | SMITH | CLERK | 7902 | 12/17/1980 | 800 | - | 20 |
| 7499 | ALLEN | SALESMAN | 7698 | 02/20/1981 | 1600 | 300 | 30 |
| 7521 | WARD | SALESMAN | 7698 | 02/22/1981 | 1250 | 500 | 30 |
| 7566 | JONES | MANAGER | 7839 | 04/02/1981 | 2975 | - | 20 |
| 7654 | MARTIN | SALESMAN | 7698 | 09/28/1981 | 1250 | 1400 | 30 |

**Scenario: Employee Attendance Management**

In this scenario, the corporate department needs to manage employee attendance records. Various triggers will be implemented to handle different aspects of attendance management, such as tracking late arrivals, calculating overtime, and managing leave requests.

**Task 01: [50 Marks]**

1. Create a database trigger **check\_late\_arrival** that fires after the insertion of each row of **attendance\_records** table. The trigger should check if the employee checked in late and upon late it should enter the information in late arrival table.

INSERT INTO attendance\_records VALUES (11, 7876, TO\_DATE('01-MAY-2023', 'DD-MON-YYYY'), TO\_TIMESTAMP('01-MAY-2023 09:00:00', 'DD-MON-YYYY HH24:MI:SS'), TO\_TIMESTAMP('01-MAY-2023 18:00:00', 'DD-MON-YYYY HH24:MI:SS'));

INSERT INTO attendance\_records VALUES (21, 7876, TO\_DATE('01-MAY-2023', 'DD-MON-YYYY'), TO\_TIMESTAMP('01-MAY-2023 09:00:00', 'DD-MON-YYYY HH24:MI:SS'), TO\_TIMESTAMP('01-MAY-2023 17:00:00', 'DD-MON-YYYY HH24:MI:SS'));

**CREATE OR REPLACE TRIGGER CHECK\_LATE\_ARAIVAL**

**AFTER INSERT ON attendance\_records**

**FOR EACH ROW**

**DECLARE**

**STD\_CHECK\_IN\_TIME TIMESTAMP := TO\_TIMESTAMP('09:00:00', 'HH24:MI:SS');**

**BEGIN**

**IF :NEW.CHECK\_IN\_TIME > STD\_CHECK\_IN\_TIME THEN**

**INSERT INTO attendance\_records VALUES (21, 7876, TO\_DATE('01-MAY-2023', 'DD-MON-YYYY'), TO\_TIMESTAMP('01-MAY-2023 09:00:00', 'DD-MON-YYYY HH24:MI:SS'), TO\_TIMESTAMP('01-MAY-2023 17:00:00', 'DD-MON-YYYY HH24:MI:SS'));**

**INSERT INTO attendance\_records VALUES (11, 7876, TO\_DATE('01-MAY-2023', 'DD-MON-YYYY'), TO\_TIMESTAMP('01-MAY-2023 09:00:00', 'DD-MON-YYYY HH24:MI:SS'), TO\_TIMESTAMP('01-MAY-2023 18:00:00', 'DD-MON-YYYY HH24:MI:SS'));**

**END IF;**

**END;**

****

1. Create a database trigger **check\_leave\_request** that fires before the insertion of each row of **leave\_requests table**. The trigger should check if the leave count of employee is or more than 3 or the number of late arrivals of employee is more than 4 then if it should prevent any insertion of that reocrd.

**CREATE OR REPLACE TRIGGER check\_leave\_request**

**BEFORE INSERT ON leave\_requests**

**FOR EACH ROW**

**DECLARE**

**v\_leave\_count employees.leave\_count%TYPE;**

**v\_late\_arrivals employees.late\_arrivals%TYPE;**

**BEGIN**

**-- Fetch the leave count and late arrivals for the employee**

**SELECT leave\_count, late\_arrivals INTO v\_leave\_count, v\_late\_arrivals**

**FROM employees**

**WHERE empno = :NEW.empno;**

**-- Check the conditions**

**IF v\_leave\_count >= 3 OR v\_late\_arrivals > 4 THEN**

**RAISE\_APPLICATION\_ERROR(-20001, 'Insertion not allowed: Employee has 3 or more leaves or more than 4 late arrivals.');**

**END IF;**

**END;**

/

1. The company has decided to award the employees who worked overtime. You are required to help them in this. Write a procedure **calculate\_overtime** which should display the total overtime employee worked with his name in this whole duration. According to standards 8 hours is company time anything more than it is overtime.

**You can use**

EXTRACT(HOUR FROM ())

**CREATE OR REPLACE PROCEDURE CALCULATE\_OVERTIME AS**

**CURSOR OVERTIME\_CURSOR IS**

**SELECT**

**EMPNO , ENAME , SUM(EXTRACT(HOUR FROM (A.check\_out\_time - A.check\_in\_time)) - 8) AS TOTAL\_OVERTIME\_HOURS**

**FROM**

**EMP E JOIN attendance\_records A ON E.EMPNO = A.EMPNO**

**GROUP BY**

**E.EMPNO , E.ENAME**

**HAVING**

**SUM(EXTRACT(HOUR FROM (A.check\_out\_time - A.check\_in\_time)) - 8) > 0;**

**EMPNOO EMP.EMPNO%TYPE;**

**EMPNAME EMP.ENAME%TYPE;**

**TOTAL\_OVERTIME\_HOURS NUMBER;**

**BEGIN**

**FOR OVERTIME\_RECORD IN OVERTIME\_CURSOR LOOP**

**EMPNOO := OVERTIME\_RECORD.EMPNO;**

**EMPNAME := OVERTIME\_RECORD.ENAME;**

**TOTAL\_OVERTIME\_HOURS := OVERTIME\_RECORD.TOTAL\_OVERTIME\_HOURS;**

**DBMS\_OUTPUT.PUT\_LINE('NAME: ' || EMPNAME || ', ID: ' || EMPNOO ||**

**', TOTAL OVER TIME HOURS: ' || TOTAL\_OVERTIME\_HOURS);**

**END LOOP;**

**END;**

1. Create a function **is\_eligible\_for\_bonus** that returns 'YES' if an employee is eligible for a monthly bonus based on their overtime hours and late days, otherwise returns 'NO'. The criteria for eligibility are having at least 10 overtime hours and no more than 2 late arrivals in the current month.

**CREATE OR REPLACE FUNCTION IS\_ELIGIBLE\_FOR\_BONUS(EMPNO EMP.EMPNO%TYPE)**

**RETURN VARCHAR2 IS**

**OVERTIME\_HOURS NUMBER;**

**LATE\_DAYS NUMBER;**

**BEGIN**

**SELECT SUM(check\_out\_time),COUNT(check\_in\_time)**

**INTO OVERTIME\_HOURS , LATE\_DAYS**

**FROM attendance\_records**

**WHERE EMPNO = IS\_ELIGIBLE\_FOR\_BONUS.EMPNO**

**AND TO\_CHAR(attendance\_date , 'DD/MM/YYYY') = TO\_CHAR(SYSDATE , 'DD/MM/YYYY');**

**IF OVERTIME\_HOURS >= 10 AND LATE\_DAYS <=2 THEN**

**RETURN 'YES';**

**ELSE**

**RETURN 'NO';**

**END IF;**

**END;**

1. Create a trigger to ensure that the sal can't be updated more than 50% of the current sal. The trigger should display the raise amount. Furthermore, sal of any employee can't be raised more than once in a month. For this purpose, add a new col to emp first that maintains the date at which the sal was raised last time.

**CREATE OR REPLACE TRIGGER check\_salary\_update**

**BEFORE UPDATE OF sal ON emp**

**FOR EACH ROW**

**DECLARE**

**v\_raise\_amount NUMBER;**

**BEGIN**

**-- Calculate the raise amount**

**v\_raise\_amount := :NEW.sal - :OLD.sal;**

**-- Check if the raise is more than 50% of the current salary**

**IF v\_raise\_amount > 0 AND v\_raise\_amount > (:OLD.sal \* 0.5) THEN**

**RAISE\_APPLICATION\_ERROR(-20002, 'Salary raise amount exceeds 50% of the current salary.');**

**END IF;**

**-- Check if the salary is being raised more than once in a month**

**IF :OLD.last\_raise\_date IS NOT NULL AND**

**MONTHS\_BETWEEN(SYSDATE, :OLD.last\_raise\_date) < 1 THEN**

**RAISE\_APPLICATION\_ERROR(-20003, 'Salary cannot be raised more than once in a month.');**

**END IF;**

**-- Display the raise amount**

**DBMS\_OUTPUT.PUT\_LINE('Raise Amount: ' || v\_raise\_amount);**

**-- Update the last\_raise\_date**

**IF v\_raise\_amount > 0 THEN**

**:NEW.last\_raise\_date := SYSDATE;**

**END IF;**

**END;**

/