**Database Systems**

**Fall 2020**

**LAB – 13**

**The objective of this lab is to:**

* **Triggers and Recap**

**Course & Lab Instructor:** Sir Asif Sohail

Instructions:

* Work on this lab individually. Discussion is not allowed.
* Evaluation of tasks will be conducted in lab.
* Anyone caught being indulged in the act of plagiarism would be awarded an “F” grade in this lab.
* 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.

* **Allowed time: 1 hour and 30 minutes**
* Best of Luck!

**Note:** You will be using following tables in your lab tasks.

* EMP (EMPNO, ENAME, JOB, SAL, HIREDATE, COMM, MGR, DEPTNO)
* DEPT (DEPTNO, DNAME, LOC)
* PUCIT (STD\_NAME)
* **Perform the following tasks**

**Task 01: [30 Marks]**

1. Create a trigger that will show a line on insert operation.

“New Record has been added”. (2 Marks)

**Solution:**

*create or replace trigger ins\_in\_emp*

*after insert on emp*

*for each row*

*begin*

*dbms\_output.put\_line('New Record has been added');*

*end;*

1. Create a table for log that stores the Date, username doing the operation and operation name (insert, update and delete) on EMP table. (3 Marks)

**Solution: (Incomplete)**

*create table log*

*(*

*dates VARCHAR2(50),*

*username VARCHAR2(50),*

*operation\_name VARCHAR2(50)*

*)*

1. Before inserting new record add the commission of employee as 35% of his salary only if the employee is of department no. 10. (3 Marks)

**Solution:**

*CREATE OR REPLACE TRIGGER comm\_emp*

*before insert ON emp*

*FOR EACH ROW*

*BEGIN*

*if :new.deptno=10 then*

*:NEW.comm := :NEW.COMM+(35/100)\*:NEW.SAL;*

*END IF;*

*END;*

1. Create a trigger that will be fired after updating the salary of an employee. It should display the new sal, old sal and their difference (3 Marks)

**Solution:**

*CREATE OR REPLACE TRIGGER display\_sal\_change*

*after UPDATE ON emp*

*FOR EACH ROW*

*DECLARE*

*sal\_diff number;*

*BEGIN*

*sal\_diff := :NEW.sal - :OLD.sal;*

*dbms\_output.put\_line('Old salary: ' || :OLD.sal);*

*dbms\_output.put\_line('New salary: ' || :NEW.sal);*

*dbms\_output.put\_line('Salary difference: ' || sal\_diff);*

*END;*

|  |  |  |
| --- | --- | --- |
| Table Name: **mon\_salary** | | |
| **Col Name** | **Data Type** | **Constraints** |
| empno | NUMBER(4) | Primary Key,  Foreign Key references emp table |
| sal\_date | Date | Primary Key |
| sal | NUMBER(7,2) |  |
| bonus | NUMBER(4) |  |

|  |  |  |
| --- | --- | --- |
| Table Name: **funds** | | |
| **Col Name** | **Data Type** | **Constraints** |
| year | NUMBER(4) | Primary Key |
| amount | NUMBER(6) | Not Null |

1. Create the following **salaries** database and apply the described constraints:

(5 Marks)

**Solution:**

*CREATE TABLE mon\_salary*

*(empno NUMBER(4),*

*sal\_date Date,*

*sal NUMBER(7,2)* *,*

*bonus NUMBER(4)* *,*

*constraint mon\_sal\_pk PRIMARY KEY(empno,sal\_date),*

*constraint mon\_sal\_fk FOREIGN key (empno) references emp*

*)*

*CREATE TABLE funds*

*(*

*year NUMBER(4) Primary Key,*

*amount NUMBER(6) Not Null*

*)*

1. Create a database trigger that fires AFTER the insertion of each row in the mon\_salary table. The trigger should SUBRACT the sal+bonus of the inserted row from amount column of funds table. (4 Marks)

**Solution: (this one has error)**

*create or replace trigger ins\_mon\_sal*

*after insert on emp*

*for each row*

*begin*

*:old.amount:=:old.amount-(:new.sal+:new.bonus);*

*end;*

1. Create a PL/SQL procedure **generate\_salary\_slip(empno,sal\_date,bonus)**. The procedure should fetch **sal** of the given employee from emp table and then insert a row in **mon\_salary** table. The procedure should also handle the exceptions. (4 Marks)

*create or replace procedure generate\_salary\_slip(empno emp.empno%type,sal\_date date,bonus number) is*

*salary emp.sal%type;*

*begin*

*select sal into salary from emp where empno=empno;*

*insert into mon\_salary values (empno, sal\_date, salary, bonus);*

*exception*

*when no\_data\_found then*

*dbms\_output.put\_line('No data!!!');*

*end;*

1. Create a database trigger that fires BEFORE the insertion of each row in the **mon\_salary** table. The trigger should ensure that an employee is not issued more than one salary in a month. (4 Marks)

***(Incomplete)***

1. i) Disable the any of the above inserted triggers

ii)Drop the trigger (2 Marks)

**Solution:**

**(i)** *alter trigger ins\_in\_emp disable;*

**(ii)** *drop trigger ins\_in\_emp;*

**Task 02: [10 Marks]**

Create the following **sales/purchase** database and apply the described constraints:

|  |  |  |
| --- | --- | --- |
| Table Name: **sales\_invoice\_master** | | |
| **Col Name** | **Data Type** | **Constraints** |
| sales\_invoice# | NUMBER(5) | Primary Key |
| sales\_inv\_date | DATE | DEFAULT sysdate |
| cust\_name | VARCHAR2(25) | Not Null |
| sales\_inv\_total | NUMBER(6) |  |

|  |  |  |
| --- | --- | --- |
| Table Name: **sales\_invoice\_details** | | |
| **Col Name** | **Data Type** | **Constraints** |
| sales\_invoice# | NUMBER(5) | Primary Key,  Foreign Key references sales\_Invoice\_master |
| prod# | NUMBER(5) | Primary Key |
| qty\_sold | NUMBER(4) | >0 |

|  |  |  |
| --- | --- | --- |
| Table Name: **purchase\_invoice\_master** | | |
| **Col Name** | **Data Type** | **Constraints** |
| purchase\_invoice# | NUMBER(5) | Primary Key |
| purchase\_inv\_date | DATE | DEFAULT sysdate |
| supp\_name | VARCHAR2(25) | Not Null |
| purchase\_inv\_total | NUMBER(6) |  |

|  |  |  |
| --- | --- | --- |
| Table Name: **purchase\_invoice\_details** | | |
| **Col Name** | **Data Type** | **Constraints** |
| sales\_invoice# | NUMBER(5) | Primary Key,  Foreign Key references Invoice |
| prod# | NUMBER(5) | Primary Key |
| qty\_purchased | NUMBER(4) | >0 |

|  |  |  |
| --- | --- | --- |
| Table Name: **product** | | |
| **Col Name** | **Data Type** | **Constraints** |
| prod# | NUMBER(5) | Primary Key |
| p\_name | VARCHAR2(50) | Not Null |
| price | NUMBER(5) | Above 5 |
| stock | NUMBER(5) |  |

*create table sales\_invoice\_master*

*(*

*sales\_invoice#* *NUMBER(5)* *Primary Key,*

*sales\_inv\_date* *DATE* *DEFAULT sysdate,*

*cust\_name* *VARCHAR2(25)* *Not Null,*

*sales\_inv\_total* *NUMBER(6)*

*)*

*create table sales\_invoice\_details*

*(*

*sales\_invoice#* *NUMBER(5),*

*prod#* *NUMBER(5),*

*qty\_sold NUMBER(4) check (qty\_sold>0),*

*constraint sales\_pk PRIMARY KEY(sales\_invoice#,prod#),*

*constraint sales\_fk FOREIGN key (sales\_invoice#) references sales\_Invoice\_master*

*)*

*create table purchase\_invoice\_master*

*(*

*purchase\_invoice#* *NUMBER(5)* *Primary Key,*

*purchase\_inv\_date* *DATE* *DEFAULT sysdate,*

*supp\_name* *VARCHAR2(25)* *Not Null,*

*purchase\_inv\_total* *NUMBER(6)*

*)*

*create table purchase\_invoice\_details*

*(*

*purchase\_invoice#* *NUMBER(5),*

*prod#* *NUMBER(5),*

*qty\_purchased* *NUMBER(4) check(qty\_purchased >0),*

*constraint purchase\_pk PRIMARY KEY(purchase\_invoice#,prod#),*

*constraint purchase\_fk FOREIGN key (purchase\_invoice#) references purchase\_invoice\_master*

*)*

*create table product1*

*(*

*prod# NUMBER(5) Primary Key,*

*p\_name VARCHAR2(50) Not Null,*

*price NUMBER(5) check (price>5),*

*stock NUMBER(5)*

*)*

**Task 03: [5 Marks]**

1. Create a sequence sales\_invoice\_sequence.

*create sequence sales\_invoice\_sequence*

*increment by 1*

*start with 1*

*maxvalue 200*

*nominvalue*

*nocache*

*nocycle;*

1. Create a sequence purchases\_invoice\_sequence.

*create sequence purchases\_invoice\_sequence*

*increment by 1*

*start with 1*

*maxvalue 200*

*nominvalue*

*nocache*

*nocycle;*

1. Insert FIVE records in product table. Keep stock as NULL.

*insert into product1*

*values(12345,'shampoo',500,null)*

*insert into product1*

*values(12346,'soap',200,null)*

*insert into product1*

*values(12347,'Brush',150,null)*

*insert into product1*

*values(12348,'toothpaste',250,null)*

*insert into product1*

*values(12349,'Detergent',1000,null)*

Insert a record each in sales\_invoice\_master and purchase\_invoice\_master tables. Both sales\_inv\_total and purchase\_inv\_total are kept NULL.

*insert into sales\_invoice\_master*

*values(sales\_invoice\_sequence.nextval,default,'Laiba',null)*

*insert into purchase\_invoice\_master*

*values(purchases\_invoice\_sequence.nextval,default,'Shifa',null)*

**Task 04: (PL/SQL) [10 Marks]**

1. Create a **Procedure** **getTotalJobDays** that takes the employee's name as a parameter and returns the **number of days** since he was hired. Execute that procedure as well

*create or replace procedure getTotalJobDays (name emp.ename%type) is*

*d date;*

*n1 number(10);*

*begin*

*select hiredate into d from emp where ename = n1;*

*n1:= trunc(sysdate)-trunc(d);*

*dbms\_output.put\_line(' No. of days' || n1);*

*end;*

1. Write a function that takes the **month's name** and displays its **number**.

**Input:** January

**Output:** 1

*create or replace function mon\_num(mon varchar2)*

*return number is*

*begin*

*if mon = 'January' then*

*return 1;*

*elsif mon = 'February' then*

*return 2;*

*elsif mon = 'March' then*

*return 3;*

*elsif mon = 'April' then*

*return 4;*

*elsif mon = 'May' then*

*return 5;*

*elsif mon = 'June' then*

*return 6;*

*elsif mon = 'July' then*

*return 7;*

*elsif mon = 'August' then*

*return 8;*

*elsif mon = 'September' then*

*return 9;*

*elsif mon = 'October' then*

*return 10;*

*elsif mon = 'Noevember' then*

*return 11;*

*elsif mon = 'December' then*

*return 12;*

*end if;*

*end;*

1. Write a **procedure** that receives a number as a parameter and displays its **binary**.

***(Incomplete)***

1. Write a function named ‘isHiredLeapYear’ that recieve the empno and return True or False on the basis of their hiring year

*create or replace function leapYear(no emp.empno%type)*

*return char is*

*year number(5);*

*begin*

*select to\_number(hiredate,YYYY) into year from emp where empno=no;*

*IF MOD(year, 4)=0*

*AND*

*MOD(year, 100)!=0*

*OR*

*MOD(year, 400)=0 THEN*

*return 'true'*

*ELSE*

*return 'false'*

*END IF;*

**Task 05: (Viva) [5 Marks]**