Pl/Sql Assignment 9-10

All Pl/Sql statements will be run keeping **emp** table and records same initially which is given below.

create table emp

(

empno number(5),

ename varchar(25),

job varchar(15),

salary number(6),

deptno number(2)

);

insert into emp values (100,'Fiora','Marketing',48000,10);

insert into emp values (200,'Garen','Sales',45000,20);

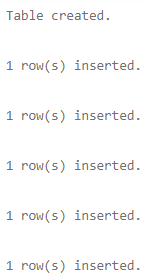
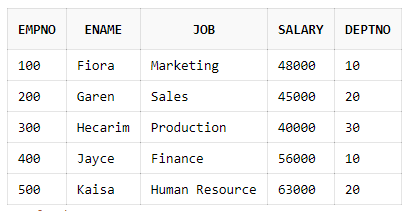
insert into emp values (300,'Hecarim','Production',40000,30);

insert into emp values (400,'Jayce','Finance',56000,10);

insert into emp values (500,'Kaisa','Human Resource',63000,20);

select \* from emp;

Output:



1. Code:

create or replace trigger t1

before insert or update of ename on emp

for each row

begin

:new.ename:=upper(:new.ename);

end;

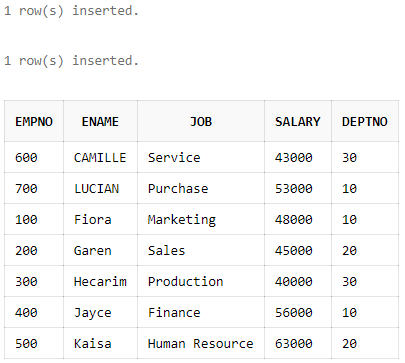
insert into emp values (600,'Camille','Service',43000,30);

insert into emp values (700,'Lucian','Purchase',53000,10);

select \* from emp;

Output: New records have employee names in uppercase.





2. Code

create or replace trigger t2

after insert or update of ename on emp

for each row

begin

dbms\_output.put\_line('Old name:'||:old.ename||' New name:'||:new.ename);

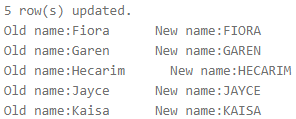
end;

update emp set ename=upper(ename);

select \* from emp;

Output: The employee names are changed to uppercase.





3. Code:

create table emp\_audit

(

empno number(5),

name\_of\_operation varchar(25),

date\_of\_operation date

);

create or replace trigger t3

after insert or update or delete on emp

for each row

begin

if inserting then

insert into emp\_audit values(:new.empno,'Record Insertion',sysdate);

elsif updating then

insert into emp\_audit values(:new.empno,'Record Updation',sysdate);

elsif deleting then

insert into emp\_audit values(:old.empno,'Record Deletion',sysdate);

end if;

end;

insert into emp values (600,'Camille','Service',43000,30);

update emp set salary=50000 where empno=100;

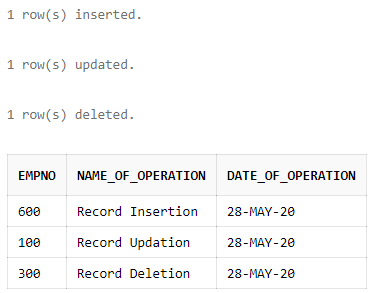
delete from emp where empno=300;

select \* from emp\_audit;

Output:







4. Code:

create or replace trigger t4

after insert or update or delete on emp

for each row

begin

if trim(to\_char(sysdate,'Day'))='Sunday' then

raise\_application\_error(-20100,'No operation can be performed on EMP table on Sunday.');

else

dbms\_output.put\_line('You are allowed to make changes in EMP table today.');

end if;

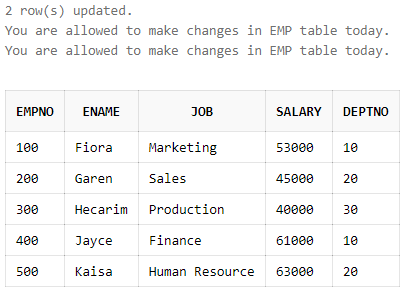
end;

update emp set salary=salary+5000 where deptno=10;

select \* from emp;

Output: Since day was Friday, the changes were allowed on employee table and thus salary was increased by 5000.





5. Code:

alter table emp

add commission number(6);

create or replace trigger t5

before insert or update or delete on emp

for each row

declare

more\_commission exception;

begin

if :new.commission>:new.salary then

raise more\_commission;

else

dbms\_output.put\_line('Your changes were performed successfully.');

end if;

exception

when more\_commission then

:new.commission:=:old.commission;

end;

update emp set commission=50000;

select \* from emp;

Output: The employees with salary greater than 50000 has their commission updated. For the rest, it remained the same.







6. Code:

create or replace trigger t6

before insert or update on emp

for each row

declare

entry emp%rowtype;

begin

if :new.empno is NULL then

raise\_application\_error(-20100,'Employee no. cannot be empty.');

end if;

select \* into entry from emp where empno=:new.empno;

raise\_application\_error(-20200,'Employee no. must be unique.');

exception

when no\_data\_found then

NULL;

end;

insert into emp(ename,job,salary,deptno) values ('Camille','Service',43000,30);

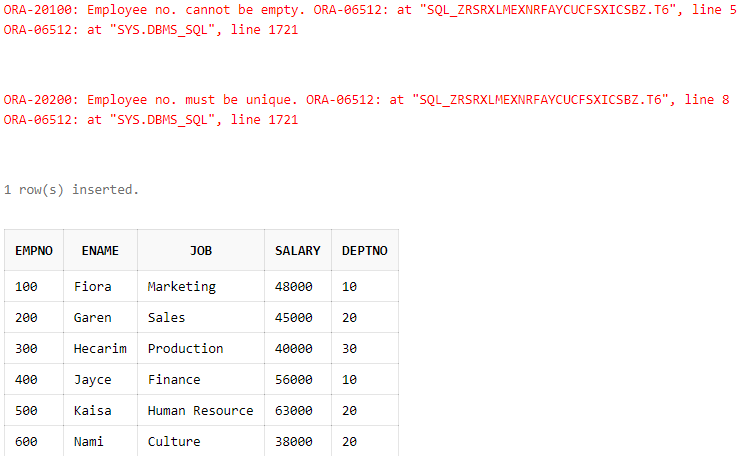
insert into emp values (300,'Lucian','Purchase',53000,10);

insert into emp values (600,'Nami','Culture',38000,20);

select \* from emp;

Output:





7. Code:

create table cust

(

cno number(2) primary key,

meterno number(3) unique,

prev\_reading number(4),

current\_reading number(4),

units number(4),

bill\_amount number(6,2)

);

create or replace trigger t7

before insert or update of prev\_reading,current\_reading on cust

for each row

begin

:new.units:=:new.current\_reading-:new.prev\_reading;

if :new.units<=100 then

:new.bill\_amount:=0.5\*:new.units;

else

:new.bill\_amount:=0.75\*(:new.units-100)+0.5\*100;

end if;

end;

insert into cust(cno,meterno,prev\_reading,current\_reading) values (89,100,300,800);

insert into cust(cno,meterno,prev\_reading,current\_reading) values (67,101,790,1000);

insert into cust(cno,meterno,prev\_reading,current\_reading) values (90,200,800,1200);

insert into cust(cno,meterno,prev\_reading,current\_reading) values (62,789,200,800);

insert into cust(cno,meterno,prev\_reading,current\_reading) values (70,889,3200,8700);

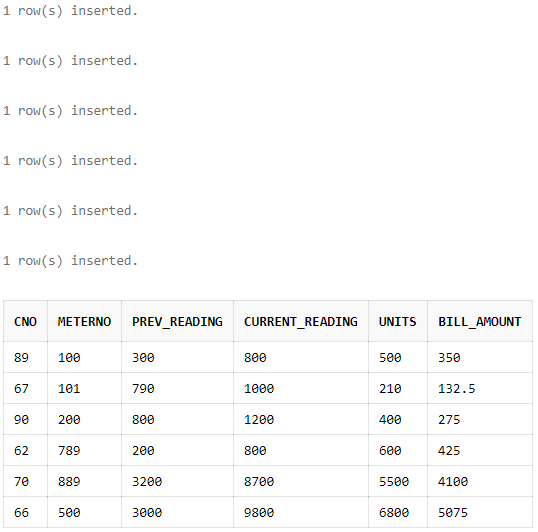
insert into cust(cno,meterno,prev\_reading,current\_reading) values (66,500,3000,9800);

select \* from cust;

Output:







8. Code:

create table dept

(

deptno number(2) primary key,

floorno number(1),

hod varchar(25)

);

insert into dept values (10,1,'Quinn');

insert into dept values (20,2,'Warwick');

insert into dept values (30,3,'Evelynn');

create or replace trigger t8

before insert or update of deptno on emp

for each row

declare

entry dept%rowtype;

begin

select \* into entry from dept where deptno=:new.deptno;

dbms\_output.put\_line('Your changes were performed successfully.');

exception

when no\_data\_found then

if inserting then

raise\_application\_error(-20300,'Department number does not exist.');

elsif updating then

:new.deptno:=:old.deptno;

dbms\_output.put\_line('Department number value is set to old value.');

end if;

end;

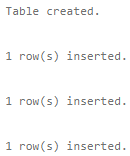
insert into emp values (600,'Camille','Service',43000,30);

insert into emp values (700,'Lucian','Purchase',53000,40);

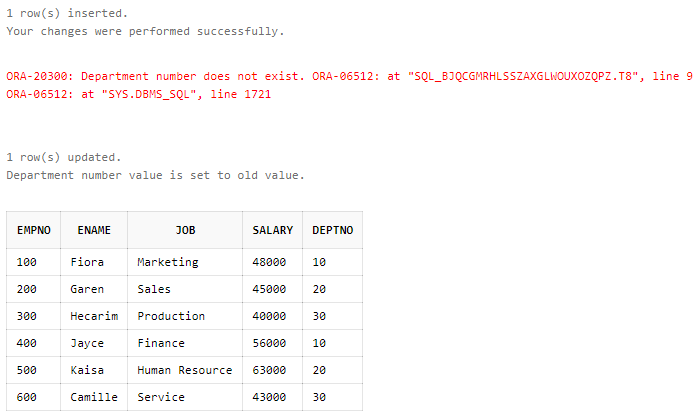
update emp set deptno=40 where empno=300;

select \* from emp;

Output: If value of deptno of record being inserted doesn’t match with the value of deptno in dept table then application error is raised to abort the insertion of record. While in case of updation, the value of deptno is set to old value.







9. Code:

create table emp\_bak

(

empno number(5),

ename varchar(25),

job varchar(15),

salary number(6),

deptno number(2)

);

insert into emp\_bak values (100,'Fiora','Marketing',48000,10);

insert into emp\_bak values (200,'Garen','Sales',45000,20);

insert into emp\_bak values (300,'Hecarim','Production',40000,30);

insert into emp\_bak values (400,'Jayce','Finance',56000,10);

insert into emp\_bak values (500,'Kaisa','Human Resource',63000,20);

create or replace trigger t9

after insert or update or delete on emp

for each row

begin

if inserting then

insert into emp\_bak values (:new.empno,:new.ename,:new.job,:new.salary,:new.deptno);

elsif updating then

update emp\_bak set empno=:new.empno,ename=:new.ename,job=:new.job,salary=:new.salary,deptno=:new.deptno where empno=:old.empno;

elsif deleting then

delete from emp\_bak where empno=:old.empno;

end if;

end;

insert into emp values (600,'Camille','Service',43000,30);

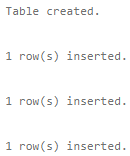
update emp set deptno=20 where empno=400;

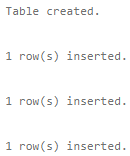
delete from emp where empno=100;

select \* from emp;

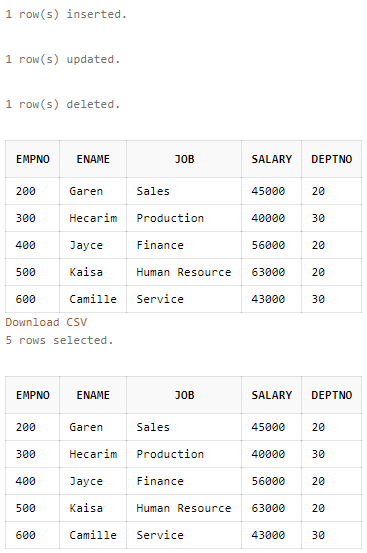
select \* from emp;

Output: After creating emp\_bak table, old enteries of emp table were manually put into the emp\_bak table for maintaining integrity of the data.









Created by:

Name: Priyanshu

Roll No.: 101803114

Group: COE-06