EMPLOYEE TABLE AND VARIOUS QUERIES

AIM: Create a table for Employee details with Employee Number as primary key and following fields: Name, Designation, Gender, Age, Date of Joining and Salary. Insert at least ten rows and perform various queries using any one Comparison, Logical, Set, Sorting and Grouping operators.

Program

1.CREATE A TABLE FOR EMPLOYEE DETAILS:

create table employee(empno number primary key,empname varchar2(20),designation varchar2(30),gender varchar2(6),age number,dojdate,salary number);

Table created.

2. DESCRIBE A TABLE:

SQL>desc employee;

3.INSERT A VALUES FOR A EMPLOYEE TABLE:

SQL> insert into employee
values(&empno,'&empname','&designation','&gender',&age,'&doj',&salary);

4.SELECT ALL THE ROWS FROM THE EMPLOYEE TABLE:

SQL>select * from employee;

5.COMPARISON:

(i) SQL> select * from employee where salary>30000;

(ii) SQL> select * from employee where age between 25 and 30;

(iii) SQL> select * from employee where empname like '%a';

(iv) SQL> select * from employee where salary in(35000,30000**,28000);**

(v) SQL>select * from employee where empno=103;

6.LOGICAL:

(i) SQL> select * from employee where salary<50000 and salary>30000;

(ii) SQL> select * from employee where designation='manager' or designation='admin';

(iii) SQL> select * from employee where not salary<30000;

7.SORTING:

(i) SQL> select * from employee order by empno;

(ii) SQL> select * from employee order by empno desc;

8.SET OPERATION:

(i) SQL> select * from employee where salary>30000 union select * from employee where age between 25 and 30;

(ii) SQL> select * from employee where salary>30000 union all select * from employee where age between 25 and 30;

(iii) SQL> select * from employee where salary>30000 intersect select * from employee where age between 25 and 30;

(iv) SQL> select * from employee where salary>30000 minus select * from employee where age between 25 and 30;

9.GROUP FUNCTION;

(i) SQL> select count(*) from employee where gender='female';

(ii) SQL> select sum(salary) from employee;

(iii) SQL> select avg(salary) from employee;

(iv) SQL> select max(salary) from employee;

(v) SQL> select min(salary) from employee;

PL/SQL IN INVENTORY TABLE

AIM : Write a PL/SQL to update the rate field by 20% more than the current rate in inventory table which has the following fields: Prono, ProName and Rate. After updating the table a new field (Alter) called for Number of item and place for values for the new field without using PL/SQL block.

1.CREATE A TABLE AS INVENTORY:

```
SQL> create table inventory(prono number primary key, pronamevarchar(20), rate number);
```

2.INSERT VALUES INTO INVENTORY TABLE:

```
SQL> insert into inventory values(&prono,'&proname',&rate);
```

3.SELECT ALL THE ROWS IN INVENTORY TABLE:

```
SQL> select * from inventory;
```

4.UPDATE THE COLUMN AS RATE:

```
SQL>    begin
          update inventory set rate=rate+(rate*20/100);
         commit;
          end;
SQL> /
PL/SQL procedure successfully completed.
```

5. SELECT ALL THE ROWS IN INVENTORY TABLE:

```
SQL> select * from inventory;
```

6.ADDING ANEW COLUMN AS NO_OF_ITEMS IN INVENTORY TABLE:

```
SQL> alter table inventory add no_of_items number;
```

7.DESCRIBEA INVENTORY TABLE:

SQL>desc inventory;

8.UPDATE AN EXISTING COLUMNS IN INVENTORY TABLE:

```
SQL> update inventory set no_of_items=case prono
2  when 1001 then 10
3  when 1002 then 8
4  when 1003 then 6
5  when 1004 then 4
6  end;
```

9. SELECT ALL THE ROWS IN INVENTORY TABLE:

SQL> select * from inventory;

TRIGGERS IN INVENTORY MANAGEMENT SYSTEM

AIM: which are based on inventory management system for checking data validity. Assume the necessary fields for both tables.

PROGRAM:

i) CREATE INVENTORY MASTER TABLE:-

SQL>create table inventory_master1(orderid number primary key,custid number,orderdate date,amount number);

Table created

ii) CREATE INVENTORY TRANSACTION TABLE:-

SQL>create table inventory_trans1(orderid number references inventory_master1,productid number,productname varchar2(30),quantity number,unitprice number);

Table created

iii) CREATE TRIGGER ON INVENTORY MASTER:-

SQL>set serveroutput on

```
create or replace trigger
check1 before insert on inventory_master1 for each row
declare
begin
if(:new.amount<0)then
raise_application_error(-20040,'invalid amount');
end if;
end;
```

Trigger created.

iv) INSERT INTO INVENTORY TABLES:-

SQL>insert into inventory_master1 values(&orderid,&custid,'&orderdate',&amount);

SQL>insert into inventory_master1 values(&orderid,&custid,'&orderdate',&amount);

Enter value for orderid: 5002
Enter value for custid: 102
Enter value for orderdate: 13-JUL-20
Enter value for amount:0
old 1: insert into inventory values(&orderid,&custid,'&orderdate',&amount)
new 1: insert into inventory values(5002,102,13-JUL-20,0)
ORA-20202: INVALID AMOUNT
ORA-06512: AT "SURESH.CHECK1", LINE 16
ORA-04088: ERROR DURING EXECUTION OF TRIGGER 'SURESH.CHECK1'

v) CREATE TRIGGER ON INVENTORY TRANSACTION:-

SQL>create or replace trigger
check2 before insert on inventory_trans1 for each row
declare
begin
if(:new.quantity<0)then
raise_application_error(-20040,'invalid quantity');
end if;
end;

Trigger created.

vi) INSERT INTO INVENTORY TRANSACTION:-

SQL>insert into inventory_trans1 values
(&orderid,&productid,'&productname',&quantity,&unitprice);

Enter value for orderid: 5002
Enter value for productid: 89
Enter value for productname: Airconditioner
Enter value for quantity:-10
Ener value for unitprice: 40000
old 1: insert into inventory values(&orderid,&productid,'&productname',&quantity,&unitprice)
new 1: insert into inventory values(5002,89,'Airconditioner',-10,40000)
ORA-20040: INVALID QUANTITY
ORA-06512: AT "SURESH.CHECK2", LINE 16
ORA-04088: ERROR DURING EXECUTION OF TRIGGER 'SURESH.CHECK2'

PROCEDURES USING STUDENT EXAM TABLE

Aim : Create a Stored Procedure for Inserting, Updating & Deleting records in TBMARKS table.

PRPGRAM:

Step 1 : Creating table

```
> create table tbmarks(rno number(3),name varchar(15),mark1 number(3),mark2 number(3),
 total number(3),AVERAGE number(5,2),result varchar(10));
```

Step 2 : Creating procedure for Insertinn records in tbmarks table

```
>create or replace procedure prmarksins(prno in number,pname in varchar,pmark1 in number,pmark2
in number)
is
vtotal number;
vaverage number;
vresult varchar(10);
begin
vtotal:=pmark1+pmark2;
vaverage:=vtotal/2;
if(pmark1>=40 and pmark2>=40) then
vresult:='Pass';
else
vresult:='Fail';
end if;
insert into tbmarks values(prno,pname,pmark1,pmark2,vtotal,vaverage,vresult);
commit;
end;
```

Step 3: To Run the Procedure

```
SQL> exec prmarksins(1,'KING',89,90);
PL/SQL procedure successfully completed.
SQL> exec prmarksins(2,'SCOTT',39,56);
PL/SQL procedure successfully completed.

SQL> select * from tbmarks;
```

Step 4 : Creating Procedure for Modifying records

```
>create or replace procedure prmarksupd(prno in number,
pmark1 in number,pmark2 in number)
is
vtotal number;
vaverage number;
vresult varchar(10);
begin
vtotal:=pmark1+pmark2;
vaverage:=vtotal/2;
if(pmark1>=40 and pmark2>=40) then
vresult:='Pass';
```

```
else
vresult:='Fail';
end if;
update tbmarks set mark1=pmark1,mark2=pmark2,total=vtotal,average=vaverage,
result=vresult where rno=prno;
commit;
end;
```

Step 5 :

```
SQL> select * from tbmarks;


SQL> exec prmarksupd(2,55,56);

PL/SQL procedure successfully completed.

SQL> select * from tbmarks;
```

Step 6: Creating Procedure for deleting the record in TBMARKS table

```
>create or replace procedure prmarksdel(prno in number)
is
begin
delete from tbmarks where rno=prno;
commit;
end;

SQL>  select * from tbmarks;


SQL> exec prmarksdel(2);

PL/SQL procedure successfully completed.

SQL>  select * from tbmarks;
```