# PROGRAM 6:

**JOIN OPERATION**

**AIM:** To display the employee details with department name using join.

# ALGORITHM:

STEP 1: Start the process

STEP 2: Create a department table with the following fields

Department ID number (3) as a primary key, department name varchar2 (10)

STEP 3: Create An employee table with the following fields

Employee number number (6) as a Primary Key, employee name varchar2 (10),

department ID number (3) as a Foreign key STEP 4: Describe the table structure

STEP 5: Insert rows into the employee and department table.

STEP 6: Display the employee details with department name using join operation. STEP 7: Stop the process.

CODING:

**Department Table Creation:**

SQL>create table department(deptid number(3),constraint deptid\_pk primary key(deptid),deptname varchar2(10));

Table Created

**Employee Table Creation:**

**SQL>**create table employee(empno number(6),constraint empno\_pk primary key(empno),empname varchar2(10),deptid number(3),constraint deptid\_fk foreign key(deptid)references department);

Table Created

Inserting values into department table:

SQL>insert into department values(100,’sales’);

SQL>insert into department values(101,’IT’);

SQL>insert into department values(102,’marketing’);

SQL>insert into department values(103,’admin’);

SQL>insert into department values(104,’purchase’);

SQL>insert into department values(105,’HR’);

**SQL>select \*from department;**

|  |  |
| --- | --- |
| **DEPTID** | **DEPTNAME** |
| 100 | sales |
| 101 | IT |
| 102 | marketing |
| 103 | admin |
| 104 | purchase |
| 105 | HR |
|  |  |

Inserting values into employee table:

SQL>insert into employee values(1000,’raj’,100);

SQL>insert into employee values(1001,’ram’,100);

SQL>insert into employee values(1002,’kavya’,101);

SQL>insert into employee values(1003,’jon’,102);

SQL>insert into employee values(1004,’kelvin’,103);

SQL>insert into employee values(1005,’surya’,104);

SQL>select \*from employee;

|  |  |  |
| --- | --- | --- |
| **EMPNO** | **EMPNAME** | **DEPTID** |
| 1000 | raj | 100 |
| 1001 | ram | 100 |
| 1002 | kavya | 101 |
| 1003 | jon | 102 |
| 1004 | kelvin | 103 |
| 1005 | surya | 104 |

**Equijoin**

SQL>select empno,empname,deptname from employee,department where department.deptid=employee.deptid;

|  |  |  |
| --- | --- | --- |
| **EMPNO** | **EMPNAME** | **DEPTNAME** |
| 1000 | raj | sales |
| 1001 | ram | sales |
| 1002 | kavya | IT |
| 1003 | jon | marketing |
| 1004 | kelvin | admin |
| 1005 | surya | purchase |

# PROGRAM 7:

**PL/SQL BLOCK USING SELECTION STATEMENT**

**AIM:** To write a PL/SQL block using selection statement.

# ALGORITHM:

STEP 1: Start the process

STEP 2: Write a PL/SQL block using any one of the following selection statement

IF statement

* IF\_ELSE statement
* ELSE\_IF statement
* CASE structure

STEP 3: Execute the PL/SQL block and display the result. STEP 4: Stop the process.

SELECTION STATEMENT

**IF:**

declare

a number(2);

begin

a:=10;

if(a=10) then

dbms\_output.put\_line('value of a='||a);

end if;

end;

O/P:

**PL/SQL procedure successfully completed**  
value of a=10

**IF-ELSE:**

declare

age number(2);

begin

age:=17;

if(age>=18) then

dbms\_output.put\_line('Eligible for vote');

else

dbms\_output.put\_line('Not Eligible for vote');

end if;

end;

O/P:

**PL/SQL procedure successfully completed**  
Not Eligible for vote

**ELSEIF:**

declare

position number(2);

begin

position:=6;

if(position=1)then

dbms\_output.put\_line('bonus is rs.20000');

elsif(position=2)then

dbms\_output.put\_line('bonus is rs.15000');

elsif(position=3)then

dbms\_output.put\_line('bonus is rs.10000');

elsif(position=4)then

dbms\_output.put\_line('bonus is rs.5000');

else

dbms\_output.put\_line('bonus nil');

end if;

end;

O/P:

**PL/SQL procedure successfully completed**  
 bonus nil

**CASE STRUCTURE:**

declare

num number(2);

a number(2);

begin

a:=10;

num:=mod(a,2);

case

when num=0 then

dbms\_output.put\_line('number is even');

when num=1 then

dbms\_output.put\_line('number is odd');

end case;

end;

O/P:

**PL/SQL procedure successfully completed**  
number is even

# PROGRAM 8:

**PL/SQL BLOCK USING LOOPING STATEMENT**

**AIM:** To write a PL/SQL block using looping statement.

# ALGORITHM:

STEP 1: Start the process

STEP 2: Write a PL/SQL block using any one of the following looping statement

* Basic loop
* While loop
* For loop

STEP 3: Execute the PL/SQL block and display the result. STEP 4: Stop the process.

CODING:

**BASIC LOOP OR DO\_WHILE LOOP**

declare

v\_sum number(3):=0;

v\_count number(2);

begin

v\_count:=1;

loop

v\_sum:=v\_sum+v\_count;

v\_count:=v\_count + 1;

exit when v\_count=11;

end loop;

dbms\_output.put\_line('sum of the digits 1 to 10 is:'||to\_char(v\_sum));

end;

O/P:

**PL/SQL procedure successfully completed**  
**sum of the digits 1 to 10 is: 55**

**WHILE LOOP**

declare

v\_sum number(3):=0;

v\_count number(2);

begin

v\_count:=1;

while v\_count<=10

loop

v\_sum:=v\_sum+v\_count;

v\_count:=v\_count + 1;

end loop;

dbms\_output.put\_line('sum of the digits 1 to 10 is:'||to\_char(v\_sum));

end;

O/P:

**PL/SQL procedure successfully completed**  
**sum of the digits 1 to 10 is: 55**

**FOR LOOP**

declare

v\_sum number(3):=0;

v\_count number(2);

begin

for v\_count in 1..10 loop

v\_sum:=v\_sum+v\_count;

end loop;

dbms\_output.put\_line('sum of the digits 1 to 10 is:'||to\_char(v\_sum));

end;

O/P:

**PL/SQL procedure successfully completed**  
**sum of the digits 1 to 10 is: 55**

# PROGRAM 9:

**EXCEPTION HANDLING**

**AIM:** To write a PL/SQL block to implement the concept of exception handling.

# ALGORITHM:

STEP 1: Start the process

STEP 2: Write a PL/SQL block to implement the concept of exception handling.

STEP 3: Execute the PL/SQL block and display the result. STEP 4: Stop the process.

CODING:

SQL>create table empp(empid number(3),depid number(3),deptname varchar2(20));

Table Created

SQL>insert into empp values(100,10,'sales');

SQL> insert into empp values(101,10,'sales');

SQL> insert into empp values(102,11,'IT');

SQL> insert into empp values(103,12,'purchase');

SQL>insert into empp values(104,13,'marketing');

SQL>select \*from empp;

|  |  |  |
| --- | --- | --- |
| **EMPID** | **DEPID** | **DEPTNAME** |
| 100 | 10 | sales |
| 101 | 10 | sales |
| 102 | 11 | IT |
| 103 | 12 | purchase |
| 104 | 13 | marketing |

**PL/SQL Procedure:**

DECLARE

V\_EMPID NUMBER(3);

V\_DEPID NUMBER(3);

V\_DEPTNAME VARCHAR2(20);

BEGIN

SELECT EMPID, DEPID,DEPTNAME

INTO V\_EMPID, V\_DEPID,V\_DEPTNAME

FROM EMPP

WHERE DEPID =10;

DBMS\_OUTPUT.PUT\_LINE(V\_EMPID || ', ' || V\_DEPID ||','||V\_DEPTNAME);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('NO SUCH DEPARTMENT WITH EMPLOYEES');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('MORE THAN ONE EMPLOYEE IN THAT DEPARTMENT');

END;

**OUTPUT:**

PL/SQL Procedure successfully completed.  
MORE THAN ONE EMPLOYEE IN THAT DEPARTMENT

# PROGRAM 10:

**TRIGGER CREATION**

**AIM:** To write a before trigger for insert query of an employee table.

# ALGORITHM:

STEP 1: Start the process

STEP 2: Create an employee table with the following fields

Employee number number (6) employee name varchar2 (10), salary number(10)

STEP 3: Insert rows into the employee table.

STEP 4: Create a before trigger for insert query of an employee table. STEP 5: Stop the process.

CODING:

SQL>create table employee(empid number(3),name varchar2(20),salary number(10));

Table Created

SQL>insert into employee values(100,'ram',10000);

SQL>insert into employee values(101,'sachin',20000);

SQL>insert into employee values(102,'ravi',30000);

SQL>insert into employee values(103,'raj',40000);

SQL>select \*from employee;

**TRIGGER CREATION:**

CREATE OR REPLACE TRIGGER EMP\_TRIG

BEFORE INSERT ON EMPLOYEE

FOR EACH ROW

DECLARE

V\_SAL NUMBER(10);

BEGIN

V\_SAL := :NEW.SALARY;

IF V\_SAL <=0 THEN

RAISE\_APPLICATION\_ERROR(-20011,'PLEASE CHECK THE SALARY');

END IF;

DBMS\_OUTPUT.PUT\_LINE(V\_SAL);

END;

**O/P:**

TRIGGER CREATED

SQL>insert into employee values (103,'raj',0);

ORA-20011: PLEASE CHECK THE SALARY