1-.Crea un procedimiento que a través del parámetro DEPT_NO de un departamento muestre: el número de departamento, su nombre y su localidad...

```
CREATE OR REPLACE PROCEDURE EJER1 (NUMDEPT NUMBER)
AS
V_REG DEPART%ROWTYPE;
NUMEMPLE NUMBER(2);
BEGIN

SELECT * INTO V_REG FROM DEPART WHERE DEPT_NO = NUMDEPT;

SELECT COUNT(*) INTO NUMEMPLE FROM EMPLE WHERE DEPT_NO=NUMDEPT;

DBMS_OUTPUT.PUT_LINE('Datos del departamento: '|| V_REG.DEPT_NO ||'
'||V_REG.DNOMBRE||' '||V_REG.LOC);

DBMS_OUTPUT.PUT_LINE(''|| NUMEMPLE);
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Ninguna fila encontrada');
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Error');
END EJER1;
```

2-. Pasando el Dept_no como parámetro calcula para cada oficio el número de empleados que tiene, puede salir 0...

```
CREATE OR REPLACE PROCEDURE EJERCICIO_2 (NUMDEPT NUMBER)
AS
V_ANALISTA NUMBER(2);
V_EMPLEADO NUMBER(2);
BEGIN
```

```
SELECT COUNT(*) INTO V_ANALISTA FROM EMPLE WHERE OFICIO='ANALISTA' AND
DEPT_NO=NUMDEPT;
SELECT COUNT(*) INTO V_EMPLEADO FROM EMPLE WHERE OFICIO='EMPLEADO' AND
DEPT_NO=NUMDEPT;


           DBMS_OUTPUT.PUT_LINE('El numero de analistas es: '||V_ANALISTA);
DBMS_OUTPUT.PUT_LINE('El numero de empleados es: '||V_EMPLEADO);


                                                                    EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Error');


                                                              END EJERCICIO_2;
```

3-. Crea un procedimiento que al recibir el nombre de un fabricante nos muestra la facturación de ese fabricante, tanto en nº de pedidos como en facturación...

```
                    CREATE OR REPLACE PROCEDURE EJERCICIO_3 (FAB VARCHAR2)
AS
COD_FAB NUMBER(2);
NUM_PED NUMBER(3);
FACT NUMBER(5,1);
BEGIN


      SELECT COD_FABRICANTE INTO COD_FAB FROM FABRICANTES WHERE NOMBRE=FAB;


    SELECT COUNT(*) INTO NUM_PED FROM PEDIDOS WHERE COD_FABRICANTE=COD_FAB;


      SELECT SUM(UNIDADES_PEDIDAS*PRECIO_VENTA) INTO FACT FROM ARTICULOS A,
PEDIDOS P WHERE A.ARTICULO=P.ARTICULO AND A.COD_FABRICANTE=P.COD_FABRICANTE
 AND A.PESO=P.PESO AND A.CATEGORIA=P.CATEGORIA AND A.COD_FABRICANTE=COD_FAB
                                                        GROUP BY A.COD_FABRICANTE;
```

```
                        DBMS_OUTPUT.PUT_LINE('Numero de pedidos: '||NUM_PED);
DBMS_OUTPUT.PUT_LINE('Facturacion: '||FACT);

                                                                    EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE ('Dato no encontrado');
WHEN TOO_MANY_ROWS THEN
DBMS_OUTPUT.PUT_LINE('Demasiadas filas');
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Error general');
END EJERCICIO_3;
```

4-.Crea un procedimiento que al recibir el código de un colegio nos muestre la cantidad de profesores...

```
                    CREATE OR REPLACE PROCEDURE EJERCICIO4 (CODIGO NUMBER)
AS
PROFESOR NUMBER(2);
PERSONAL NUMBER(2);
BEGIN
SELECT COUNT(*) INTO PROFESOR FROM PROFESORES WHERE COD_CENTRO=CODIGO;
SELECT COUNT(*) INTO PERSONAL FROM PERSONAL WHERE COD_CENTRO=CODIGO;
DBMS_OUTPUT.PUT_LINE('PROFESORES:'||PROFESOR);
DBMS_OUTPUT.PUT_LINE('PERSONAL:'||PERSONAL);
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('NO EXISTE EL CENTRO');
END EJERCICIO4;
```

5-.Crea 1 tabla con los mismos datos de fabricantes y con 2 columnas más:

a.Facturación.

b.Número_unidades.
Cada vez que se recibe un pedido (estos son los parámetros del procedure) el programa recalcula la facturación y el número de unidades de ese fabricante...

```
CREATE TABLE FACTURACION
AS
SELECT F.COD_FABRICANTE, NOMBRE, PAIS, SUM(UNIDADES_PEDIDAS*PRECIO_VENTA)
FACT, SUM(UNIDADES_PEDIDAS) UNIDADES
FROM FABRICANTES F, PEDIDOS P, ARTICULOS A
WHERE F.COD_FABRICANTE=A.COD_FABRICANTE AND A.ARTICULO=P.ARTICULO AND
A.COD_FABRICANTE=P.COD_FABRICANTE AND A.PESO=P.PESO AND
A.CATEGORIA=P.CATEGORIA
GROUP BY F.COD_FABRICANTE, NOMBRE, PAIS

SELECT * FROM FACTURACION;
DESCRIBE FABRICANTES;



      CREATE OR REPLACE PROCEDURE EJERCICIO5 (N VARCHAR2, ART VARCHAR2, COD
                              NUMBER, PES NUMBER, CAT VARCHAR2, UNI NUMBER)
AS
V_FACT NUMBER(6,2);
V_UNI NUMBER(4);
BEGIN

          INSERT INTO PEDIDOS VALUES (N, ART, COD, PES, CAT, SYSDATE, UNI);

                      SELECT SUM(UNIDADES_PEDIDAS*PRECIO_VENTA) INTO V_FACT
FROM ARTICULOS A, PEDIDOS P WHERE A.ARTICULO=P.ARTICULO AND
A.COD_FABRICANTE=P.COD_FABRICANTE AND A.PESO=P.PESO AND
A.CATEGORIA=P.CATEGORIA AND P.COD_FABRICANTE=COD GROUP BY P.COD_FABRICANTE;

                SELECT SUM(UNIDADES_PEDIDAS) INTO V_UNI FROM PEDIDOS WHERE
                              COD_FABRICANTE=COD GROUP BY COD_FABRICANTE;

UPDATE FACTURACION
SET FACT=V_FACT, UNIDADES=V_UNI
```

```
WHERE COD_FABRICANTE=COD;

                                                                    EXCEPTION
WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('No hay datos');

                                                                WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Error');

                                                                END EJERCICIO5;
```

6-. Crea un procedimiento que inserta un artículo nuevo y pasamos todos sus datos como parámetros. Después el procedimiento intenta insertar un pedido y una venta...

```
      CREATE OR REPLACE PROCEDURE EJERCICIO6 (ART VARCHAR2, COD NUMBER, PES
                  NUMBER, CAT VARCHAR2, PV NUMBER, PC NUMBER, EXIST NUMBER)
AS
V_UNI_PED NUMBER(3);
V_UNI_VEN NUMBER(3);
BEGIN
INSERT INTO ARTICULOS VALUES (ART, COD, PES, CAT, PV, PC, EXIST);
INSERT INTO PEDIDOS SELECT NIF, ART, COD, PES, CAT, SYSDATE, 5 FROM
TIENDAS;
INSERT INTO VENTAS SELECT NIF, ART, COD, PES, CAT, SYSDATE, 5 FROM TIENDAS;
SELECT SUM(UNIDADES_PEDIDAS) INTO V_UNI_PED FROM PEDIDOS WHERE ARTICULO=ART
AND COD_FABRICANTE= COD AND PESO=PES AND CATEGORIA=CAT;
SELECT SUM(UNIDADES_VENDIDAS) INTO V_UNI_VEN FROM VENTAS WHERE ARTICULO=ART
AND COD_FABRICANTE= COD AND PESO=PES AND CATEGORIA=CAT;
DBMS_OUTPUT.PUT_LINE('Las unidades pedidas son:'||V_UNI_PED);
DBMS_OUTPUT.PUT_LINE('Las unidades vendidas son:'||V_UNI_VEN);

                                                                    EXCEPTION
```

```
                                                    WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('NO HAY DATOS');

                                                        WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('ERROR');

                                                      END EJERCICIO6;

                                                      OTRA SOLUCION:

      CREATE OR REPLACE PROCEDURE EJERCICIO6 (ART VARCHAR2, COD NUMBER, PES
                  NUMBER, CAT VARCHAR2, PV NUMBER, PC NUMBER, EXIS NUMBER)
AS
SUM_UNI_VEN NUMBER(3);
SUM_UNI_PED NUMBER(3);
BEGIN
INSERT INTO ARTICULOS VALUES (ART, COD, PES, CAT, PV, PC, EXIS);
IF (SQL%ROWCOUNT = 0) THEN
DBMS_OUTPUT.PUT_LINE('ARTICULO NO INSERTADO');
ELSE
INSERT INTO PEDIDOS SELECT NIF, ART, COD, PES, CAT, SYSDATE, 5 FROM
TIENDAS;
INSERT INTO VENTAS SELECT NIF, ART, COD, PES, CAT, SYSDATE, 5 FROM TIENDAS;

           SELECT SUM(UNIDADES_PEDIDAS) INTO SUM_UNI_PED FROM PEDIDOS WHERE
        ARTICULO=ART AND COD_FABRICANTE=COD AND CATEGORIA=CAT AND PESO=PES;
SELECT SUM(UNIDADES_VENDIDAS) INTO SUM_UNI_VEN FROM VENTAS WHERE
ARTICULO=ART AND COD_FABRICANTE=COD AND CATEGORIA=CAT AND PESO=PES;

           DBMS_OUTPUT.PUT_LINE('Suma de unidades pedidas: '||SUM_UNI_PED);
DBMS_OUTPUT.PUT_LINE('Suma de unidades vendidas: '||SUM_UNI_VEN);
END IF;
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('ERROR GENERAL');
END EJERCICIO6;
```

7-. Crea un procedimiento que recibe los datos de un empleado (EMP_NO, APELLIDO, OFICIO, DIR, FECHA_ALT, COMISION, DEPT_NO) La comisión por defecto será 0...

```
CREATE OR REPLACE PROCEDURE EJERCICIO7(EMP NUMBER, APE VARCHAR2, OFI
  VARCHAR2, D NUMBER, FECHA DATE, DEPT NUMBER, COM NUMBER DEFAULT 0)
AS
V_SALARIO NUMBER(6,2);
V_NUM NUMBER(1);
BEGIN
SELECT COUNT(*) INTO V_NUM FROM DEPART WHERE DEPT_NO=DEPT;

                                        IF (V_NUM=1 AND COM>=0) THEN
SELECT TRUNC(AVG(SALARIO)*1.03) INTO V_SALARIO FROM EMPLE WHERE
DEPT_NO=DEPT;
INSERT INTO EMPLE VALUES (EMP, APE, OFI, D, FECHA, V_SALARIO, COM, DEPT);
ELSE
DBMS_OUTPUT.PUT_LINE('EL DEPARTAMENTO NO EXISTE');
END IF;

                                                        EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('EL DEPARTAMENTO NO EXISTE');
END EJERCICIO7;DBMS_OUTPUT.PUT_LINE('EL DEPARTAMENTO NO EXISTE');
END IF;

                                                        EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('EL DEPARTAMENTO NO EXISTE');
END EJERCICIO7;
```

8-. Escribe desde SQL*Plus el ejemplo número 1 del epígrafe «Uso de subprogramas almacenados» y prueba a ejecutarlo con distintos valores.

```
                                                    CREATE OR REPLACE
PROCEDURE ver_depart (numdepart NUMBER)
AS
v_dnombre VARCHAR2(14);
v_localidad VARCHAR2(14);
BEGIN
SELECT dnombre, loc INTO v_dnombre, v_localidad
FROM depart
WHERE dept_no = numdepart;
 DBMS_OUTPUT.PUT_LINE('Num depart:'||numdepart|| ' * Nombre dep:'||
v_dnombre ||
 ' * Localidad:'||v_localidad);
 EXCEPTION
 WHEN NO_DATA_FOUND THEN
 DBMS_OUTPUT.PUT_LINE('No encontrado departamento ');
 END ver_depart;
```

*****TEMA 9******

1-. Crea un procedimiento que recibe los datos de un profesor:
a.Comprueba que el centro existe.
b.Comprueba que su clave no se repite.
c.Si los datos son correctos ◊ el registro se inserta.
d.Gestiona posibles excepciones.

```
                                CREATE OR REPLACE PROCEDURE INSERTA_PROFE
(CODCENTRO NUMBER, D NUMBER, APELL VARCHAR2, ESPCLIDAD VARCHAR2)
AS
V_CENTROS NUMBER(2);
V_PROFESOR (2);

                                                                BEGIN
SELECT COUNT(*) INTO V_CENTROS FROM CENTROS WHERE COD_CENTRO = CODCENTRO;
SELECT COUNT(*) INTO V_PROFESOR FROM PROFESORES WHERE DNI = D;
IF(V_CENTROS=1 AND V_PROFESOR=0) THEN
 INSERT INTO PROFESORES VALUES(CODCENTRO, D , APELL , ESPCLIDAD );
 DBMS_OUTPUT.PUT_LINE('PROFESOR INSERTADO');
 ELSE
 DBMS_OUTPUT.PUT_LINE('EL CENTRO NO EXISTE O EL DNI SE REPITE');
 END IF;
EXCEPTION
```

```
WHEN OTHERS THEN
 DBMS_OUTPUT.PUT_LINE('ERROR GENERAL');
END INSERTA_PROFE;
```

```
BEGIN INSERTA_PROFE(10, 1548, 'PEREZ', 'MATES'); END;
```

2-. Crea un procedimiento que recibe los datos de un empleado y realiza las siguientes acciones:

a.Comprueba que existe el dir y el departamento del empleado.
b.Comprueba que el salario es positivo.
c.Si todos los datos son correctos ◊ inserta el empleado.
d.Muestra por pantalla el gasto total en salarios del departamento del nuevo empleado.
e.Gestiona posibles excepciones.

```
CREATE OR REPLACE PROCEDURE INSERTA_EMPLE
(NRO NUMBER, APE VARCHAR2, OFIC VARCHAR2, DR NUMBER, FECHAALT DATE, SALARIO
NUMBER, COM NUMBER, DEPTO NUMBER)
AS
V_EXIS_DIR NUMBER(2);
V_EXIS_DEPTO NUMBER(2);
V_TOTAL_DEPTO NUMBER(8);
BEGIN
SELECT COUNT(*) INTO V_EXIS_DIR FROM EMPLE WHERE EMP_NO = DR; --TIENE QUE
DAR UNO PARA VALIDO
SELECT COUNT(*) INTO V_EXIS_DEPTO FROM EMPLE WHERE DEPT_NO = DEPTO;--UNO
PARA VALIDO
  IF(V_EXIS_DIR=1 AND V_EXIS_DEPTO>0) THEN
    IF(SALARIO>0) THEN
      INSERT INTO EMPLE
      VALUES(NRO, APE, OFIC, DR, FECHAALT, SALARIO, COM, DEPTO);
      SELECT SUM(SALARIO) INTO V_TOTAL_DEPTO FROM EMPLE WHERE DEPT_NO =
DEPTO;
      DBMS_OUTPUT.PUT_LINE('EL GASTO EN SALARIOS
        DEL DEPARTAMENTO DEL EMPLEADO INSERTADO ES: '||V_TOTAL_DEPTO);
    ELSE
      DBMS_OUTPUT.PUT_LINE('EL SALARIO NO ES POSITIVO');
    END IF;
  ELSE
```

```
      DBMS_OUTPUT.PUT_LINE('EL DIRECTOR O EL DEPARTAMENTO NO EXISTE');
  END IF;
  EXCEPTION
WHEN OTHERS THEN
 DBMS_OUTPUT.PUT_LINE('ERROR GENERAL');
END INSERTA_EMPLE;


BEGIN INSERTA_EMPLE(1458, 'LOPEZ', 'EMPLEADO', 7900, SYSDATE, 2049, 0, 99);
                                                                    END;
```

3-. Crea un procedimiento que recibe los datos de un pedido y realiza las siguientes acciones:

a. Comprueba que el artículo existe en la tabla padre.
b. Comprueba que la tienda existe.
c. Si las unidades del pedido es superior a las de la venta, sólo asigna al pedido el 20% de
las existencias.
d. La fecha de pedido es la actual.
e. Una vez insertado el pedido resta las unidades pedidas a las existencias del artículo.

```
                                CREATE OR REPLACE PROCEDURE INSERTA_PEDIDO
(NF VARCHAR2, ARTIC VARCHAR2, CODFABRI NUMBER, PES NUMBER, CATEG VARCHAR2,
FECHA DATE, PEDID NUMBER)
AS
V_ARTI_EXIST NUMBER(1);
V_TIEN_EXIST NUMBER(1);
V_VENTAS NUMBER(6);
V_TOTALAPEDIR NUMBER(6);
BEGIN
IF(FECHA = SYSDATE)THEN
  SELECT COUNT(*) INTO V_ARTI_EXIST FROM ARTICULOS WHERE
   ARTICULO = ARTIC AND COD_FABRICANTE = CODFABRI AND PESO = PES AND
CATEGORIA = CATEG;


            SELECT COUNT(*) INTO V_TIEN_EXIST FROM TIENDAS WHERE NIF = NF;
```

```
  IF(V_TIEN_EXIST=1 AND V_ARTI_EXIST=1) THEN
    SELECT SUM(UNIDADES_VENDIDAS) INTO V_VENTAS FROM VENTAS WHERE
      ARTICULO = ARTIC AND COD_FABRICANTE = CODFABRI AND PESO = PES AND
CATEGORIA = CATEG;


                                                    IF (V_VENTAS > PEDID) THEN
     V_TOTALAPEDIR := PEDID;
     ELSE
      SELECT TRUNC(EXISTENCIAS*0.2) INTO V_TOTALAPEDIR FROM ARTICULOS WHERE
       ARTICULO = ARTIC AND COD_FABRICANTE = CODFABRI AND PESO = PES AND
CATEGORIA = CATEG;
    END IF;--DE V VENTAS
      --HACE EL INSERT
      INSERT INTO PEDIDOS VALUES (NF, ARTIC, CODFABRI, PES, CATEG, FECHA,
V_TOTALAPEDIR);
      --CAMBIA LOS DATOS DE EXISTENCIA
      UPDATE ARTICULOS
      SET EXISTENCIAS = EXISTENCIAS - V_TOTALAPEDIR
      WHERE ARTICULO = ARTIC AND COD_FABRICANTE = CODFABRI AND PESO = PES
AND CATEGORIA = CATEG;
  ELSE
    DBMS_OUTPUT.PUT_LINE('EL ARTICULO O LA TIENDA NO EXISTEN');
  END IF;-- DEL TIEN EXIST Y ARTI EXIST
ELSE
  DBMS_OUTPUT.PUT_LINE('TIENE QUE SER DE LA FECHA ACTUAL');
END IF;--DE LA FECHA ACTUAL

 EXCEPTION
WHEN OTHERS THEN
 DBMS_OUTPUT.PUT_LINE('ERROR GENERAL');
END INSERTA_PEDIDO;


   BEGIN INSERTA_PEDIDO('5555-B','Atún', 10, 3,'Segunda', SYSDATE, 4); END;
SELECT * FROM PEDIDOS WHERE ARTICULO = 'Atún';
SELECT EXISTENCIAS FROM ARTICULOS WHERE ARTICULO = 'Atún';
```

4. Crea un procedimiento que al recibir los datos de una venta realiza las siguientes acciones:

a. Muestra la facturación de la tienda (en ventas) antes de insertarse la venta.
b. Se comprueba que todos los datos de la venta son correctos, como en el ejercicio 3.
c. Se inserta la venta.
d. Vuelve a mostrar la facturación de la tienda una vez insertada la venta.
e. Gestiona las posibles excepciones.

```
                                    CREATE OR REPLACE PROCEDURE INSERTA_VENTA
(NF VARCHAR2, ARTIC VARCHAR2, CODFABRI NUMBER, PES NUMBER, CATEG VARCHAR2,
FECHA DATE, VENDIDOS NUMBER)
AS
FACT NUMBER(6);
V_TIENDA_EXIST NUMBER(1);
V_ARTI_EXIST NUMBER(1);
--TOMA DATOS
SELECT COUNT(*) INTO V_ARTI_EXIST FROM ARTICULOS WHERE
  ARTICULO = ARTIC AND COD_FABRICANTE = CODFABRI AND PESO = PES AND
CATEGORIA = CATEG;
SELECT COUNT(*) INTO V_TIENDA_EXIST FROM TIENDAS WHERE
  NIF = NF;


                                                                    --VERIFICA
IF(V_TIENDA_EXIST != 1) THEN
  DBMS_OUTPUT.PUT_LINE('LA TIENDA NO EXISTE');
  GOTO LOULTIMO;
END IF;


                                                                --FACTURACION
SELECT SUM(PRECIO_VENTA * UNIDADES_VENDIDAS) INTO FACT FROM ARTICULOS A,
VENTAS V
WHERE A.ARTICULO = V.ARTICULO
AND A.PESO = V.PESO
AND A.CATEGORIA = V.CATEGORIA
AND A.COD_FABRICANTE = V.COD_FABRICANTE
AND V.NIF = NF
GROUP BY V.NIF;
DBMS_OUTPUT.PUT_LINE('LA FACTURACION ACTUAL ES:'||FACT);


                                                IF(V_ARTI_EXIST != 1) THEN
  DBMS_OUTPUT.PUT_LINE('EL ARTICULO NO EXISTE');
```

```
  GOTO LOULTIMO;
END IF;

                                                        <<LOULITMO>>
END INSERTA_VENTA;
--MUESTRA FACTURACION PREVIA
```

5-. Crea un procedimiento que recibe 2 fechas, y muestra la facturación de la venta y los pedidos de todas las tiendas entre esas dos fechas.

```
   CREATE OR REPLACE PROCEDURE FACT_TOTAL_ENTRE(PRIMERO DATE, SEGUNDO DATE)
AS
V_TOTAL_FACTURACION NUMBER(6,2);
BEGIN
SELECT SUM(UNIDADES_VENDIDAS * PRECIO_VENTA) INTO V_TOTAL_FACTURACION FROM
ARTICULOS A,VENTAS V
WHERE A.ARTICULO = V.ARTICULO
AND A.COD_FABRICANTE = V.COD_FABRICANTE
AND A.PESO = V.PESO
AND A.CATEGORIA = V.CATEGORIA
AND FECHA_VENTA BETWEEN PRIMERO AND SEGUNDO;
 DBMS_OUTPUT.PUT_LINE('FACTURACION TOTAL:'|| V_TOTAL_FACTURACION);
 EXCEPTION
WHEN OTHERS THEN
 DBMS_OUTPUT.PUT_LINE('ERROR GENERAL');
END FACT_TOTAL_ENTRE;

                        BEGIN FACT_TOTAL_ENTRE('11/03/06','11/03/10'); END;
```

6-. Crea una función que recibe como parámetro el número de departamento y nos devuelve el número de empleados que existen en ese departamento. Después crea un procedimiento que realiza
una llamada a esa función.

```
CREATE OR REPLACE FUNCTION EMPLE_DEPART(DEPT NUMBER)
RETURN NUMBER
AS
V_EMPLE NUMBER(10);
BEGIN
SELECT COUNT(*) INTO V_EMPLE FROM EMPLE WHERE DEPT_NO = DEPT;
RETURN V_EMPLE;
 EXCEPTION
WHEN OTHERS THEN
 DBMS_OUTPUT.PUT_LINE('ERROR GENERAL');
END EMPLE_DEPART;
```

```
SELECT  DEPT_NO,EMPLE_DEPART(DEPT_NO) FROM EMPLE;
```

7-. Escribe una función que toma la PK de un artículo y nos devuelve la suma de unidades pedidas de ese artículo.

```
CREATE OR REPLACE FUNCTION UNI_PEDID(ART VARCHAR2, CODFABRI NUMBER, PES
NUMBER, CAT VARCHAR2)
RETURN NUMBER
AS
V_TOTAL_PEDIDOS NUMBER(10);
BEGIN
SELECT SUM(UNIDADES_PEDIDAS) INTO V_TOTAL_PEDIDOS FROM PEDIDOS
WHERE ARTICULO = ART
AND COD_FABRICANTE = CODFABRI
```

```
AND PESO = PES
AND CATEGORIA = CAT;
RETURN V_TOTAL_PEDIDOS;
 EXCEPTION
WHEN OTHERS THEN
 DBMS_OUTPUT.PUT_LINE('ERROR GENERAL');
END UNI_PEDID;
```

```
SELECT UNI_PEDID('Macarrones', 20, 1, 'Primera') FROM DUAL;
```

8-. Crea un procedimiento que actualiza las existencias de un artículo que recibe como parámetro, restando a las existencias la suma de unidades pedidas de ese artículo. Utiliza
la función del ejercicio anterior.

```
 CREATE OR REPLACE PROCEDURE ACT_EXISTENCIAS(ART VARCHAR2, CODFABRI NUMBER,
                                                PES NUMBER, CAT VARCHAR2)
AS
V_TOTAL_PEDIDAS NUMBER(4);
BEGIN
SELECT UNI_PEDID(ART, CODFABRI, PES, CAT)INTO V_TOTAL_PEDIDAS FROM DUAL;
UPDATE ARTICULOS
SET EXISTENCIAS = EXISTENCIAS - V_TOTAL_PEDIDAS
WHERE ARTICULO = ART
AND COD_FABRICANTE = CODFABRI
AND PESO = PES
AND CATEGORIA = CAT;
 EXCEPTION
WHEN NO_DATA_FOUND
 DBMS_OUTPUT.PUT_LINE('NO HAY INFO');
WHEN OTHERS THEN
 DBMS_OUTPUT.PUT_LINE('ERROR GENERAL');
END ACT_EXISTENCIAS;
```

9.-Escribe un procedimiento que reciba dos números y visualice su suma.

```
DECLARE
importe NUMBER(8,2);
contador NUMBER(2,0) := 0;
nombre CHAR(20) NOT NULL := "MIGUEL";
BEGIN
end;
```

```
DECLARE
 Num1 NUMBER(8,2) := 0 --falta el punto y coma
 Num2 NUMBER(8,2) NOT NULL := 0; --nunca puede ser null
 Num3 NUMBER(8,2) NOT NULL; --Falta un valor
 Cantidad INTEGER(3); --PL/SQL dispone de subtipos de NUMBER que se
utilizan por compatibilidad y/o
 --para establecer restricciones. Son DECIMAL, NUMERIC, INTEGER, REAL,
SMALLINT, etcétera.
 Precio, Descuento NUMBER(6); --no se puede declarar en grupo
 Num4 Num1%ROWTYPE; --siempre y cuando num1 fuera una tabla
 Dto CONSTANT INTEGER;
BEGIN
...
END;
```

10.- Codifica un procedimiento que reciba una cadena y la visualice al revés.

```
CREATE OR REPLACE PROCEDURE REVES(PALABRA VARCHAR2)
AS
V_INVERSO VARCHAR(30);
v_posicion INTEGER :=1;
v_CARACTER CHAR;
BEGIN
v_CARACTER := SUBSTR(PALABRA ,v_posicion,1);
WHILE v_caracter BETWEEN 'A' AND 'z' LOOP
 v_INVERSO := v_CARACTER ||v_INVERSO;
 v_posicion := v_posicion + 1;
 v_CARACTER := SUBSTR(PALABRA ,v_posicion,1);
END LOOP;
 DBMS_OUTPUT.PUT_LINE('LA CADENA INVERSA ES:'||V_INVERSO);
END;
```

11.- Reescribe el código de los dos ejercicios anteriores para convertirlos en funciones que retornen los valores que mostraban los procedimientos.

```
CREATE OR REPLACE FUNCTION SUMADOR_FUNCTION (PRIMERO NUMBER, SEGUNDO NUMBER)
RETURN INTEGER
AS
 V_SUMA NUMBER(10);
BEGIN
 V_SUMA := PRIMERO + SEGUNDO;
RETURN V_SUMA;
END SUMADOR_FUNCTION;
```

```
CREATE OR REPLACE FUNCTION REVES_FUNCTION (PALABRA VARCHAR2)
AS
V_INVERSO VARCHAR(30);
v_posicion INTEGER :=1;
v_CARACTER CHAR;
BEGIN
v_CARACTER := SUBSTR(PALABRA ,v_posicion,1);
WHILE v_caracter BETWEEN 'A' AND 'z' LOOP
 v_INVERSO := v_CARACTER ||v_INVERSO;
 v_posicion := v_posicion + 1;
 v_CARACTER := SUBSTR(PALABRA ,v_posicion,1);
END LOOP;
RETURN V_INVERSO;
END REVES_FUNCTION;
```

12.- Escribe una función que reciba una fecha y devuelva el año, en número, correspondiente a esa fecha.

```
CREATE OR REPLACE FUNCTION ANIO(FECHA DATE)
RETURN NUMBER
```

```
AS
V_ANIO NUMBER(4);
BEGIN
V_ANIO:=TO_NUMBER(TO_CHAR(FECHA, 'YYYY'));
RETURN V_ANIO;
END ANIO;
```

13.- Escribe un bloque PL/SQL que haga uso de la función anterior.

```
BEGIN DBMS_OUTPUT.PUT_LINE(ANIO(SYSDATE)); END;
```

14.- Desarrolla una función que devuelva el número de años completos que hay entre dos fechas que se pasan como parámetros.

```
CREATE OR REPLACE FUNCTION ANIOS(PRIMERO DATE, SEGUNDO DATE)
RETURN NUMBER
AS
V_ANIOS NUMBER(10);
BEGIN
V_ANIOS := TRUNC((MONTHS_BETWEEN(PRIMERO, SEGUNDO))/12);
-- V_ANIOS := TRUNC((SEGUNDO - PRIMERO)/365.25);
RETURN V_ANIOS;
END ANIOS;
/
BEGIN DBMS_OUTPUT.PUT_LINE(ANIOS(SYSDATE, '15/05/2010')); END;
```

15.- Escribe una función que, haciendo uso de la función anterior, devuelva los trienios que hay entre dos fechas (un trienio son tres años).

```
CREATE OR REPLACE FUNCTION TRIENIOS(PRIMERO DATE, SEGUNDO DATE)
```

```
RETURN NUMBER
AS
V_TRIENIOS NUMBER(10);
BEGIN
V_TRIENIOS := TRUNC(ANIOS(PRIMERO, SEGUNDO)/3);
RETURN V_TRIENIOS;
END TRIENIOS;
/
BEGIN DBMS_OUTPUT.PUT_LINE(TRIENIOS(SYSDATE, '15/05/2000')); END;
```

16.-Codifica un procedimiento que reciba una lista de hasta cinco números y visualice su suma.

```
 CREATE OR REPLACE PROCEDURE SUMANDO(N1 NUMBER DEFAULT 0, N2 NUMBER DEFAULT
          0, N3 NUMBER DEFAULT 0, N4 NUMBER DEFAULT 0, N5 NUMBER DEFAULT 0)
AS
V_SUMA NUMBER(10);
BEGIN
 V_SUMA := N1 + N2 + N3 + N4 + N5;
 DBMS_OUTPUT.PUT_LINE(V_SUMA);
END;
```

17.- Codifica un procedimiento que permita borrar un empleado cuyo número se pasará en la llamada.

```
                CREATE OR REPLACE PROCEDURE BORRANDO_EMPLE(NROEMPLE NUMBER)
AS
BEGIN
 DELETE FROM EMPLE
WHERE EMP_NO = NROEMPLE;
EXCEPTION
WHEN OTHERS THEN
 DBMS_OTUPUT.PUT_LINE('ERROR');
END BORRANDO_EMPLE;
```

18.- Escribe un procedimiento que modifique la localidad de un departamento. El procedimiento recibirá como parámetros el número del departamento y la nueva localidad

```
  CREATE OR REPLACE PROCEDURE CAMBIAR_LOCALIDAD(PDEPT NUMER, PLOC VARCHAR2)
AS
BEGIN
UPDATE DEPART SET LOC=PLOC WHERE DEPT_NO=PDEPT;
IF(SQL%ROWCOUNT=1) THEN
  DBMS_OTUPUT.PUT_LINE('DEPARTAMENTO MODIFICADO');
ELSE
  DBMS_OTUPUT.PUT_LINE('EL DEPARTAMENTO NO EXISTE');
END IF;
EXCEPTION
 WHEN OTHERS THEN
   DBMS_OTUPUT.PUT_LINE('ERROR');
END CAMBIAR_LOCALIDAD;
```

19.- Visualiza todos los procedimientos y funciones del usuario almacenados en la base de datos y su situacion(valid o invalid)

Utiliza un bucle for para encontrar la suma de los n numero naturales, ingresado por parametro.

```
                          CREATE OR REPLACE PROCEDURE SUMATORIAS(N NUMBER)
AS
 i INTEGER;
 V_TOTAL NUMBER(10) DEFAULT 0;
BEGIN
FOR i IN 1..N LOOP
V_TOTAL := V_TOTAL + i;
END LOOP;
 DBMS_OUTPUT.PUT_LINE(V_TOTAL);
END SUMATORIAS;
```

```
BEGIN SUMATORIAS(10);
END;
```