**SELECT COUNT(DISTINCT localidad) as localidades**

**FROM empleado;**

**-- 8- El nombre y direcci�n de empleado que menor salario cobra.**

**SELECT nombre, direccion**

**FROM empleado**

**WHERE salario = (SELECT MIN(salario) from empleado)**

**-- 12-Obtener un listado con los nombres y apellidos de los empleados, as� como los a�os de que llevan en la empresa ordenados por ordenado por a�os en la empresa de forma descendente.**

**SELECT nombre, ape1 || ' ' || ape2 as apellidos,**

**floor(months\_between(SYSDATE, fechaingreso) /12) as antiguedad**

**FROM empleado**

**ORDER BY antiguedad DESC;**

**SELECT \***

**FROM empleado**

**WHERE coddpto = ANY(SELECT coddpto FROM DPTO WHERE presupuesto > 90000);**

**SELECT \***

**FROM empleado**

**WHERE upper(nombre) LIKE '%AN%';**

**-- 17-Obtener un listado ordenado alfab�ticamente donde aparezcan los nombres de los empleados y a continuaci�n el literal "CON comision" si la tiene, y "no SIN comision" si no la tiene.**

**SELECT CASE WHEN**

**comision is null**

**then nombre || ' CON comision'**

**else nombre || ' no SIN comision' end as info\_emp**

**FROM empleado**

**order by nombre**

**-- 20-Obtener a los nombres, apellidos de los empleados que no son jefes de departamento.**

**SELECT nombre, ape1 || ' ' || ape2 as apellidos**

**FROM empleado**

**WHERE codemple NOT IN (SELECT codemplejefe from dpto);**

**INSERT INTO** tabla **VALUES** ( V1, V2, V3)

**INSERT INTO** tabla **(COSA1, COSA2) VALUES** ( V1, V2)

**UPDATE** tabla **SET** cosa1= ’cosa’ **WHERE** condicion;

**UPDATE tabla SET cosa =**

**CASE**

**WHEN condicion THEN x**

**ELSE y**

**END**

**DELETE FROM tabla WHERE cosa= cosa**

**—------------------------------------------------------------------------------------------------------------**

**CREATE OR REPLACE FUNCTION nombreF(param: tipo)**

**RETURN tipo**

**AS**

**var1 tipo**

**var2 tipo**

**BEGIN**

**logica y return**

**END;**

**—----------------------------------------------------**

**CREATE OR REPLACE PROCEDURE nombrep(param: tipo)**

**AS**

**var1 tipo**

**var2 tipo**

**BEGIN**

**logica y return**

**END;**

**—------------------------------------------------------------------------------------------------------------**

**CREATE OR REPLACE TRIGGER nombretrigger**

**BEFORE / AFTER insert or update on agentes for each row**

**begin**

**logica**

**end;**

**raise\_application\_error(-20001, 'La clave no puede ser menor de 5');**

**—-----------------------------------------------------------------------------------------------------------**

**CREATE OR REPLACE TYPE usuario AS OBJECT (**

**login varchar2 (10),**

**nombre varchar2 (10),**

**credito number,**

**member procedure incremento( inc number)**

**)**

**CREATE OR REPLACE type BODY Usuario AS**

**member procedure incremento (inc number) IS**

**begin**

**credito = credito + 1**

**end incremento;**

**end;**

**/**

**—------------------------------------------------------------------------------------------------------------**

**ALTER TYPE usuario DROP ATRIBUTTE nombre;**

**ALTER TYPE usuario**

**ADD ATRIBUTE (nomrbe varchar2)**

**MODIFY ATRIBUTE nombre varchar2**

**—------------------------------------------------------------------------------------------------------------**

**CREATE OR REPLACE TYPE BODY usuario AS**

**CONSTRUCTOR FUNCTION usuario ( login varchar2, credito number )**

**RETURN SELF AS RESULT**

**IS**

**BEGIN**

**logica…..**

**END;**

**—------------------------------------------------------------------------------------------------------------**

**u1 = NEW USUARIO (DATA, DATO, DATO);**

**credito = u1.getCredito( );**

**—------------------------------------------------------------------------------------------------------------**

**HERENCIA:**

**CREATE TYPE persona AS object (**

**nombre varchar**

**apellido varchar**

**) NOT FINAL;**

**/**

**CREATE TYPE usuario UNDER persona (**

**f\_anc varchar**

**credito varchar**

**);**

**/**

**DROP TYPE USUARIO;**

**DROP TYPE PERSONA;**

**EXAMEN —-------------------------------------------------------------------------------------------------------------------**

**CREATE OR REPLACE FUNCTION ejercicio1 (codproducto CHAR)**

**RETURN NUMBER IS**

**existe INTEGER;**

**sumaUnidades INTEGER;**

**BEGIN**

**SELECT COUNT(\*) INTO existe FROM producto**

**WHERE codigo\_producto = codproducto;**

**SELECT SUM(cantidad) INTO sumaUnidades**

**FROM detalle\_pedido**

**WHERE codigo\_producto = codproducto;**

**RETURN sumaUnidades;**

**END;**

**/**

**—------------------------------------------------------------------------------------------------------------------------**

**CREATE OR REPLACE PROCEDURE ejercicio2 (codPedido CHAR)**

**AS**

**esta INTEGER;**

**importe NUMBER;**

**codCliente NUMBER;**

**BEGIN**

**SELECT COUNT(\*) INTO esta FROM pedido**

**WHERE codigo\_pedido = codPedido;**

**IF esta = 0 THEN**

**RAISE\_APPLICATION\_ERROR (-20011, 'El pedido noexiste');**

**END IF;**

**SELECT SUM(cantidad\*precio\_unidad) INTO importe**

**FROM detalle\_pedido**

**WHERE codigo\_pedido = codPedido;**

**SELECT codigo\_cliente INTO codCliente**

**FROM pedido**

**WHERE codigo\_pedido = codPedido;**

**INSERT INTO pago (codigo\_cliente, total) VALUES**

**(codCliente, importe);**

**END;**

**—------------------------------------------------------------------------------------------------------------------------**

**CREATE OR REPLACE TRIGGER ejercicio03**

**BEFORE INSERT ON DETALLE\_PEDIDO**

**FOR EACH ROW**

**DECLARE**

**v\_pedido INTEGER;**

**v\_producto INTEGER;**

**BEGIN**

**SELECT COUNT(\*) INTO v\_pedido**

**FROM pedido**

**WHERE codigo\_pedido = :new.codigo\_pedido;**

**IF v\_pedido = 0 THEN**

**RAISE\_APPLICATION\_ERROR(-20011, 'El pedido no existe');**

**END IF;**

**SELECT COUNT(\*) INTO v\_producto FROM producto**

**WHERE codigo\_producto = :new.codigo\_producto;**

**IF v\_producto = 0 THEN**

**RAISE\_APPLICATION\_ERROR(-20012, 'El producto noexiste');**

**END IF;**

**END;**

**/**

**—------------------------------------------------------------------------------------------------------------------------**

**DELETE FROM oficina WHERE codigo\_oficina='BCN-ES';**

**UPDATE producto**

**SET precio\_venta = 1.1\*precio\_venta**

**WHERE gama='Herramientas';**

**INSERT INTO gama (gama, descripcion\_text)**

**VALUES**

**('Ferretería', 'Herramientas para el campo de mano');**

**CREATE OR REPLACE TRIGGER COMPRUEBA\_AGENTES**

**BEFORE INSERT OR UPDATE ON AGENTES FOR EACH ROW**

**BEGIN**

**IF LENGTH(:NEW.CLAVE) < 5 THEN**

**RAISE\_APPLICATION\_ERROR(-20001, 'La clave de un agente no puede ser inferior a 5');**

**END IF;**

**END;**

DECLARE

i number(1);

j number(1);

BEGIN

<< outer\_loop >>

FOR i IN 1..3 LOOP

<< inner\_loop >>

FOR j IN 1..3 LOOP

dbms\_output.put\_line('i is: '|| i || ' and j is: ' || j);

END loop inner\_loop;

END loop outer\_loop;

END;

/