- Nos Conectamos como administradores con:
  conn sys as sysdba

- Creamos un usario llamado c##jardineria y le concedemos los permisos.

  create user c##jardineria identified by jardineria default tablespace users;
  grant connect, resource,DBA to c##jardineria;
  CONNECT c##jardineria/jardineria;

Para la base de datos jardinería (jardineria.sql).

DQL

1) Defina las siguientes consultas:

1. Sacar el código de oficina y la ciudad donde hay oficinas.

```
SELECT codigoOficina, ciudad
FROM oficinas;
```

2. Cuantos empleados hay en la compañía.

```
SELECT COUNT(codigoEmpleado)
FROM empleados;
```

3. Cuantos clientes tiene cada país.

```
SELECT COUNT(codigoEmpleado), pais
FROM empleados e, oficinas o
WHERE e.codigoOficina = o.codigoOficina
GROUP BY pais;
```

4. Pago medio en 2005.

```
SELECT AVG(cantidad)
FROM pagos
WHERE fechapago LIKE '%05';
```

5. Cuantos pedidos están en cada estado ordenado descendente por el número de pedidos.

```
SELECT COUNT(codigoPedido), estado
FROM pedidos
GROUP BY estado
ORDER BY COUNT(codigoPedido) DESC
```

6. Obtener el precio del producto más caro y del más barato.

```
SELECT nombre, precioVenta
FROM productos
WHERE precioVenta IN (SELECT MAX(precioVenta) FROM productos);
```

```
SELECT nombre, precioVenta
FROM productos
WHERE precioVenta IN (SELECT MIN(precioVenta) FROM productos);
```

7. Obtener el nombre del cliente con mayor límite de crédito.

```
SELECT nombreCliente, limiteCredito
FROM clientes
WHERE limiteCredito IN (SELECT MAX(limiteCredito) FROM clientes);
```

8. Obtener el nombre, apellido1 y cargo de los empleados que no representen a ningún cliente.

```
SELECT nombre, apellido1, puesto, codigoEmpleado
FROM empleados e
WHERE codigoEmpleado NOT IN (SELECT (codigoEmpleadoRepVentas) FROM clientes);
```

9. Obtener el nombre del producto más caro.

```
SELECT nombre
FROM productos
WHERE precioVenta IN (SELECT MAX(precioVenta) FROM productos);
```

10. Obtener el nombre del producto del que más unidades se han vendido en un mismo pedido.

```
SELECT nombre
FROM productos p, detallePedidos d
WHERE p.codigoProducto=d.codigoProducto AND d.cantidad IN (SELECT MAX(cantidad)
FROM detallePedidos);
```

11. Obtener los clientes cuya línea de crédito sea mayor que los pagos que haya realizado.

```
SELECT DISTINCT nombreCliente, cantidad, limitecredito
FROM clientes c
NATURAL JOIN pagos p
WHERE c.limitecredito > p.cantidad;
```

12. Sacar el producto que más unidades tiene en stock y el que menos unidades tiene en stock.

```
SELECT nombre, cantidadEnStock
FROM productos
WHERE cantidadEnStock IN (SELECT MAX(cantidadEnStock) FROM productos);

SELECT nombre, cantidadEnStock
FROM productos
WHERE cantidadEnStock IN (SELECT MIN(cantidadEnStock) FROM productos);
```

13. Numero de productos que se piden en cada pedido.

```
SELECT p.nombre, d.cantidad, d.codigoPedido
FROM productos p
JOIN detallePedidos d ON p.codigoproducto = d.codigoproducto;
```

14. Listado con el nombre de cada cliente y el nombre y apellido de su representante de ventas.

```
SELECT c.nombreCliente, e.nombre, e.apellido1
FROM clientes c
JOIN empleados e ON c.codigoEmpleadoRepVentas = e.codigoEmpleado;

SELECT c.nombreCliente, e.nombre, e.apellido1
FROM clientes c, empleados e
WHERE c.codigoEmpleadoRepVentas = e.codigoEmpleado;
```

15. Nombre de los clientes que no han realizado pagos junto con el nombre de representante.

```
SELECT c.nombreCliente, e.nombre, e.apellido1
FROM pagos p, clientes c, empleados e
WHERE p.codigoCliente = c.codigoCliente AND p.codigoCliente = c.codigoCliente;
```

16. Nombre, unidades vendidas, total facturado y total con impuestos de los productos que hayan facturado más de 3000 euros.

```
SELECT p.nombre, SUM(d.cantidad) AS unidadesVendidas,
SUM(d.cantidad*d.preciounidad) AS totalFacturado, SUM(d.cantidad*d.preciounidad*1.21)
AS totalConImpuestos
FROM productos p
JOIN detallePedidos d ON p.codigoProducto = d.codigoProducto
GROUP BY d.codigoProducto, p.nombre
HAVING SUM(d.cantidad*d.preciounidad) > 3000;
```

17. Dirección de las oficinas que tengan clientes de Fuenlabrada.

```
SELECT o.lineaDireccion1, o.lineaDireccion2, o.ciudad, o.region, o.pais, o.codigoPostal
FROM oficinas o
JOIN empleados e ON o.codigoOficina = e.codigoOficina
JOIN clientes c ON c.codigoEmpleadoRepVentas = e.codigoEmpleado
WHERE c.ciudad = 'Fuenlabrada';
```

18. Listado con el precio total de cada pedido.

```
SELECT SUM(cantidad * precioUnidad), codigoPedido
FROM detallePedidos
GROUP BY codigoPedido;
```

19. Cliente que hizo el pago con mayor cuantía y el que hizo el pago con menor cuantía.

```
SELECT c.nombreCliente, p.cantidad
FROM clientes c, pagos p
WHERE cantidad IN (SELECT MAX(cantidad) FROM PAGOS) AND c.codigoCliente =
p.codigoCliente;
```

```
SELECT c.nombreCliente, p.cantidad
FROM clientes c, pagos p
WHERE cantidad IN (SELECT MIN(cantidad) FROM PAGOS) AND c.codigoCliente =
p.codigoCliente;
```

20. Clientes que hayan hecho pedidos en el 2008 por una cuantía superior a 2000 euros.

```
SELECT  DISTINCT c.nombreCliente
FROM clientes c
JOIN pedidos p ON c.codigoCliente = p.codigoCliente
JOIN detallePedidos d ON p.codigoPedido = d.codigoPedido
WHERE p.fechaPedido LIKE '%/%/08'
GROUP BY d.codigoPedido, c.nombreCliente
HAVING SUM(d.Cantidad * d.PrecioUnidad) > 2000;
```

21. ¿Pedido más caro del empleado que más clientes tiene?.

22. Ciudad y teléfono de las oficinas de EEUU.

```
SELECT ciudad, telefono
FROM oficinas
WHERE pais LIKE '%EEUU%';
```

23. Nombre, apellidos y email de los empleados a cargo de Alberto Soria.

```
SELECT nombre, apellido1, apellido2
FROM empleados
WHERE codigoJefe IN (SELECT codigoEmpleado FROM empleados WHERE nombre
LIKE '%Alberto%' AND apellido1 LIKE '%Soria%');
```

24. Nombre, apellidos y cargo de aquellos que no sean representantes de ventas.

```
SELECT nombre, apellido1, apellido2, puesto
FROM empleados
WHERE puesto NOT LIKE '%Representante Ventas%';
```

25. Nombre de los clientes españoles

```
SELECT nombreCliente
FROM clientes
WHERE pais LIKE '%Espa%a%';
```

26. Número de clientes de las ciudades que empiezan por M.

```
SELECT COUNT(codigoCliente) AS numeroClientes, ciudad
FROM clientes
WHERE ciudad LIKE 'M%'
GROUP BY ciudad;
```

27. Código de empleado y el número de clientes al que atiende cada representante de ventas.

```
SELECT c.codigoEmpleadoRepVentas, COUNT(*) AS numClientes
FROM clientes c
GROUP BY c.codigoEmpleadoRepVentas;
```

28. Número de clientes que no tiene asignado representante de ventas.

```
SELECT COUNT(*) AS clientesSinRepVentas
FROM clientes
WHERE codigoEmpleadoRepVentas IS NULL;
```

29. Código de cliente de aquellos clientes que hicieron pagos en 2008.

```
SELECT codigoCliente
FROM pagos
WHERE fechaPago LIKE '%/%/08';
```

30. Distintos estados por los que puede pasar un pedido.

```
SELECT DISTINCT UPPER(estado)
FROM pedidos;
```

31. Numero de pedido, código de cliente, fecha requerida y fecha de entrega de los pedidos cuya fecha de entrega ha sido al menos dos días antes de la fecha requerida.

```
SELECT codigoPedido, codigoCliente, fechaEsperada, fechaEntrega
FROM pedidos
WHERE fechaEntrega <= fechaEsperada - 2;
```

32. Facturación que ha tenido la empresa en toda la historia, indicando la base imponible, el IVA y el total facturado. Nota: La base imponible se calcula sumando el coste del producto por el número de unidades vendidas. El IVA es el 21% de la base imponible.

```
SELECT SUM(d.cantidad*d.preciounidad) AS baseImponible,
    SUM(d.cantidad*d.preciounidad*0.21) AS IVA,
    SUM(d.cantidad*d.preciounidad*1.21) AS totalFacturado
FROM detallePedidos d;
```

33. Listado con el nombre de los empleados junto con el nombre de sus jefes.

```
SELECT e1.nombre AS Empleado, e1.apellido1 AS apellidoEmpleado,
        e2.nombre AS Jefe, e2.apellido1 AS apellidoJefe
FROM empleados e1, empleados e2
WHERE e1.codigoJefe = e2.codigoEmpleado;
```

34. Listado de clientes indicando el nombre del cliente y cuantos pedidos ha realizado.

```
SELECT c.nombreCliente, COUNT(p.codigoPedido) AS PEDIDOSREALIZADOS
FROM pedidos p
JOIN clientes c ON c.codigoCliente = p.codigoCliente
GROUP BY c.nombreCliente;
```

35. Listado con los nombres de los clientes y el total pagado por cada uno de ellos.

```
SELECT c.nombreCliente, SUM(p.cantidad) AS TOTALPAGADO
FROM pagos p
JOIN clientes c ON c.codigoCliente = p.codigoCliente
GROUP BY c.nombreCliente;
```

36. Cuantos empleados tiene cada oficina, mostrando el nombre de la ciudad donde está la oficina.

```
SELECT COUNT(e.codigoEmpleado) AS TOTALEMPLEADOS, o.ciudad
FROM empleados e
JOIN oficinas o ON e.codigoOficina = o.codigoOficina
GROUP BY o.ciudad;
```

37. Nombre, apellido, oficina (ciudad) y cargo del empleado que no represente a ningún cliente.

```
SELECT e.nombre, e.apellido1, o.ciudad, e.puesto
FROM empleados e
JOIN oficinas o ON e.codigoOficina = o.codigoOficina
WHERE e.codigoempleado NOT IN (SELECT codigoEmpleadoRepVentas FROM clientes);
```

38. Media de unidades de stock de los productos agrupados por gama.

```
SELECT AVG(cantidadEnStock), gama
FROM productos
GROUP BY gama;
```

39. Clientes que residan en la misma ciudad donde hay una oficina, indicando dónde está la oficina.

```
SELECT c.nombreCliente, c.ciudad, o.lineaDireccion1, o.lineaDireccion2
FROM clientes c, oficinas o
WHERE c.ciudad IN (
    SELECT o.ciudad
    FROM oficinas)
ORDER BY c.ciudad;
```

```
SELECT c.nombreCliente, c.ciudad, o.lineaDireccion1, o.lineaDireccion2
FROM clientes c
JOIN oficinas o ON c.ciudad = o.ciudad
ORDER BY c.ciudad;
```

40. Clientes que residan en ciudades donde no hay oficinas ordenado por la ciudad donde residen.

```
SELECT nombreCliente, ciudad
FROM clientes
WHERE ciudad NOT IN (SELECT ciudad FROM oficinas)
ORDER BY ciudad;
```

DML

2) Crea y ejecuta un script ‘actualiza.sql’ que realice las siguientes acciones:

1. Inserta una oficina con sede en Fuenlabrada

INSERT INTO oficinas(codigoOficina, ciudad, pais, region, codigoPostal, telefono, lineaDireccion1)

VALUES ('FUE-ESP', 'Madrid', 'Espania', 'Fuenlabrada', '28555', '+34 91 555577', 'desconocida');

2. Inserta un empleado para la oficina de Fuenlabrada que sea representante de ventas

INSERT INTO empleados(codigoEmpleado, nombre, apellido1, extension, email, codigoOficina, puesto)

VALUES (32, 'Sofia', 'Arcoiris', '1234', 'sArco@jardineria.com', 'FUE-ESP', 'Representante ventas');

3. Inserta un cliente del representante de ventas en el punto 2.

INSERT INTO clientes(codigoCliente, nombreCliente, telefono, fax, lineadireccion1, ciudad, codigoEmpleadoRepVentas)

VALUES (39, 'Amapolas S.A.', '111222333', '444555666', 'c/pradera 5', 'Toledo', 32);

4. Inserta un pedido del cliente anterior (con su detalle) de al menos 2 productos.

INSERT INTO pedidos VALUES(150, '14-04-2023', '17-04-2023', '17-04-2023', 'Pendiente', NULL, 39);

INSERT INTO detallePedidos VALUES(150, '21636', 1, 14, 1);

INSERT INTO detallePedidos VALUES(150, '22225', 1, 12, 2);

5. Actualiza el código del cliente insertado y averigua si hubo cambios en las tablas relacionadas.

UPDATE clientes SET codigoCliente = 50 WHERE codigoCliente = 39;;

No me deja cambiar el código del cliente insertado, ya que es clave ajena de la tabla Pedidos.

6. Borra el cliente y verifica si hubo cambios

Antes de eliminar el nuevo cliente, tenemos que borrar el pedido que tiene asociado ese cliente, si no, no nos deja eliminarlo.

DELETE FROM detallePedidos WHERE codigoPedido = 150;

DELETE FROM pedidos WHERE codigoPedido = 150;

DELETE FROM clientes WHERE codigoCliente = 39;

3) Usando subconsultas en los filtros, realiza las siguientes actualizaciones:

1.Borra los clientes que no tengan pedidos

```
DELETE FROM clientes
WHERE codigoCliente NOT IN (SELECT codigoCliente FROM pedidos);
```

2.Incrementa en un 20% el precio de los productos

```
UPDATE productos SET precioVenta = precioVenta*1.20;
```

3.Borra los pagos del cliente con menor límite de credito.

```
DELETE FROM pagos WHERE codigoCliente IN
    (SELECT codigoCliente FROM clientes WHERE limiteCredito IN
        (SELECT MIN(limiteCredito) FROM clientes))
```

4.Establece a 0 el límite de crédito del cliente que menos unidades pedidas tenga del producto ‘OR-179’

```
UPDATE clientes SET limiteCredito = 0 WHERE codigoCliente IN(
    SELECT codigoCliente FROM pedidos WHERE codigoPedido IN(
        SELECT codigoPedido FROM detallepedidos
                WHERE codigoProducto = 'OR-179'
                GROUP BY codigoPedido
                ORDER BY SUM(cantidad)
                FETCH FIRST 1 ROW ONLY));
```

PL/SQL

4) Triggers

Cree un disparador para que cuando se borre un cliente de la base de datos, se almacene en una tabla que ya existirá previamente, que se llame “clientes_borrados” con la siguiente información: Nombre_cliente,Fecha_borrado,Importe_facturación.

```
CREATE OR REPLACE TRIGGER clientesBorrados
AFTER DELETE ON clientes
FOR EACH ROW
DECLARE
   v_importe_facturacion clientes_borrados.importe_facturacion%type;
BEGIN
   SELECT SUM(cantidad) INTO v_importe_facturacion
   FROM pagos
   WHERE codigoCliente = :OLD.codigoCliente;

   INSERT INTO clientes_borrados (nombre_cliente, fecha_borrado,
importe_facturacion)
   VALUES(:OLD.nombreCliente, SYSDATE, v_importe_facturacion);
END;
```

5) Procedimientos

Deseamos tener un procedimiento que pasemos **el código de un cliente** y nos liste los datos de ese cliente: Código, Nombre, Ciudad y País, así como los pagos que ha realizado, ordenados cronológicamente. Para finalizar que muestre la cantidad total pagada. Fíjate bien en la imagen capturada para ver todos los detalles que debe mostrar el procedimiento. Implementa también el tratamiento de excepciones.

```
SQL> EXEC PAGOS_CLIENTE(1);
CODIGO CLIENTE: 1
NOMBRE CLIENTE: DGPRODUCTIONS GARDEN
CIUDAD CLIENTE: San Francisco
PAIS   CLIENTE: USA
================================================
ID-TRANSACCION  FECHA      FORMA  CANTIDAD
================================================
ak-std-000001   10/11/08   PayPal 2000
ak-std-000002   10/12/08   PayPal 2000
================================================
TOTAL PAGOS EFECTUADOS: 4000

Procedimiento PL/SQL terminado correctamente.

SQL> EXEC PAGOS_CLIENTE(3);
CODIGO CLIENTE: 3
NOMBRE CLIENTE: Gardening Associates
CIUDAD CLIENTE: Miami
PAIS   CLIENTE: USA
================================================
ID-TRANSACCION  FECHA      FORMA  CANTIDAD
================================================
ak-std-000003   16/01/09   PayPal 5000
ak-std-000004   16/02/09   PayPal 5000
ak-std-000005   19/02/09   PayPal 926
================================================
TOTAL PAGOS EFECTUADOS: 10926

Procedimiento PL/SQL terminado correctamente.
```

```
CREATE OR REPLACE PROCEDURE listar_cliente(p_codigoCliente
clientes.codigoCliente%type)
IS
   v_codigoCliente clientes.codigoCliente%type;
   v_nombreCliente clientes.nombreCliente%type;
   v_ciudad clientes.ciudad%type;
   v_pais clientes.pais%type;
   v_cantidadTotal pagos.cantidad%type := 0;

BEGIN
   SELECT codigoCliente, nombreCliente, ciudad, pais
   INTO v_codigoCliente, v_nombreCliente, v_ciudad, v_pais
   FROM clientes
   WHERE codigoCliente = p_codigoCliente;

   DBMS_OUTPUT.PUT_LINE('CODIGO CLIENTE: '|| v_codigoCliente);
   DBMS_OUTPUT.PUT_LINE('NOMBRE CLIENTE: '|| v_nombreCliente);
   DBMS_OUTPUT.PUT_LINE('CIUDAD CLIENTE: '|| v_ciudad);
   DBMS_OUTPUT.PUT_LINE('PAIS CLIENTE: '|| v_pais);

   FOR pago IN (
      SELECT idTransaccion, fechaPago, formaPago, cantidad
      FROM pagos
      WHERE codigoCliente = p_codigoCliente
      ORDER BY fechaPago)
   LOOP
      DBMS_OUTPUT.PUT_LINE('ID-TRANSACCION: '|| pago.idTransaccion || ' FECHA: '||
pago.fechaPago || ' FORMA: '|| pago.formaPago || ' CANTIDAD: '|| pago.cantidad);

      v_cantidadTotal := v_cantidadTotal + pago.cantidad;
   END LOOP;

   DBMS_OUTPUT.PUT_LINE('TOTAL PAGOS EFECTUADOS: '|| v_cantidadTotal);
END;
```