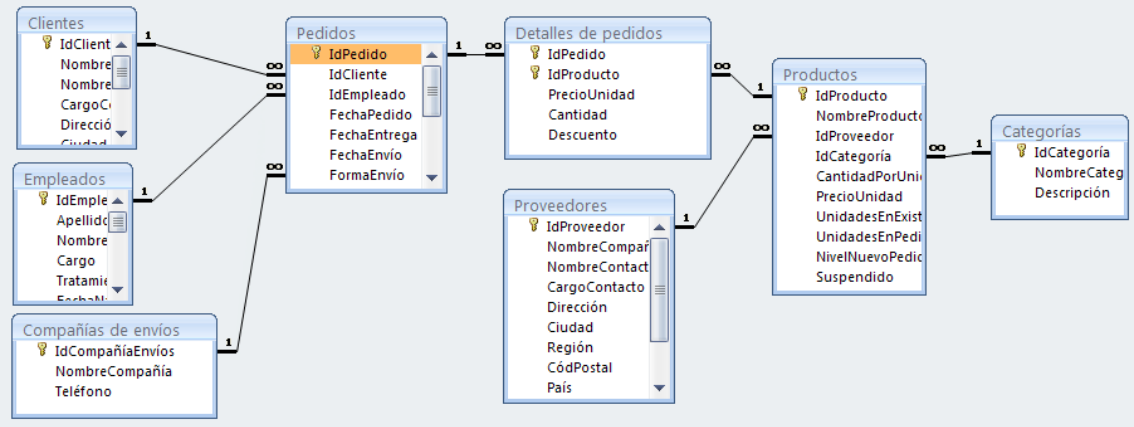
PRACTICAS PL-SQL

1) **SQL Developer**: En la carpeta material/empresa/carga disponemos de un fichero SQL, que crea las tablas de esta Base de datos y 8 ficheros CSV que representan los datos de cada tabla. Utilizando el **SQL-Developer** crear las tablas y luego importar los CSV. Teniendo en cuenta el esquema de la BD. OJO, algún campo no se llama igual pero las relaciones son correctas y se corresponden con el Script de SQL. Tener esto en cuenta en el orden de importación de los datos.



2) **Consultas**:

   a. Obtener el total del precio de los productos por categoría.

   b. Obtener las categorías que tienen una cantidad superior a 300 unidades. Dejar el resultado en otra tabla.

   c. Insertar una nueva empleada, es la hermana de King Robert Id = 7. Se llama Queen Robert, tratamiento = “Srta.”, FechaNacimiento = 5-abr-1965 y fecha de contratación: hoy. Resto de campos, copiarlos del hermano.

   d. Incrementar en un 15% el precio por unidad de los productos que tengan menos de 30 unidades en existencia.

   e. Obtener los pedidos (ids de pedidos) en los que está presente un producto en concreto. Buscarlo por texto, por ejemplo: ‘Sirope de regaliz’.

f. Indicar que clientes no ha realizado ningún pedido. Mostrar los datos de los clientes.
g. Obtener los nombres de los productos que se han pedido más de una vez.

3) SQL *Plus:

a. Definir variables de usuario y de bloque.
b. Leer variables de teclado y utilizar valores de sustitución.

4) **Consultas de varias tablas**: Disponemos de dos tablas: una tabla representa las personas que se apuntan a un equipo de fútbol y la otra tabla representa las personas que se apuntan a un equipo de Baloncesto.

a. Trabajar con el **SQL *Plus**.
b. Crear las dos tablas con una columna nombre. No crear la clave primaria porque se podrán repetir.
c. Dar de alta algunos registros teniendo en cuenta que habrá gente que se apunte sólo a baloncesto, sólo a futbol, a ambos o incluso al mismo deporte más de una vez.
    i. Futbol: Ana, Alberto, María, Andrés, Raúl.
    ii. Baloncesto: Alberto, Gema, Andrés, Tomás.
d. Mediante consultas resolver lo siguiente:
    i. Quien se ha apuntado sólo a Futbol.
    ii. Sólo a Baloncesto.
    iii. A ambos equipos.
    iv. Todas las personas que participan en el torneo con repetidos y sin repetidos.
    v. Duplicar algún registro en una de las tablas y hacer una consulta que indique cuales son los repetidos.

5) **DDL**: Mediante código DDL crear las tablas que consideres necesarias indicando los campos, claves primarias y secundarias para almacenar la red de metro de Madrid. Grabar el fichero SQL.

6) **Secuencias**: Definir las secuencias que consideres y probar a insertar algún registro en la tabla de estaciones y líneas de la red de Metro. Dar de alta alguna línea y estaciones. Probar la integridad referencial con la tabla relacionada. Establecer el borrado en cascada, restrict, set null y probar a borrar estaciones o líneas, tenido registros en la tabla de relación.

7) **Vistas**:

   a. Sobre las tablas de Academia, hacer las siguientes Vistas. Probar después hacer inserciones con las vistas. Probar with check option, read only.

      i. Obtener los alumnos que no tienen teléfono.
      ii. Obtener el nombre y el número de matrícula de los alumnos que hayan nacido en Madrid, Barcelona o Bilbao. Ordenar la consulta alfabéticamente por nombre.
      iii. Que profesores no tienen un puesto fijo y el sueldo es superior a 1000 €.
      iv. Alumnos que son de Madrid, que tienen una edad de 30 a 40 años y la academia no tiene el número de teléfono.
      v. Obtener todos los profesores que cuya fecha de nacimiento esté entre dos fechas que indicará el usuario.

   b. Empresa (***Se puede realizar en varias vistas***): Implementar una vista que muestre el id del cliente y % de ganancia que la empresa tiene por cada cliente. Teniendo en cuenta el importe total de todos los pedidos y de cada cliente. NO tener en cuenta el cargo del pedido, sólo operar con los detalles.

8) **Funciones predefinidas de Oracle:**
   a. Cadenas: Definir una variable, se le asigna un nombre (nombre apellido1 apellido2), escribir un bloque de código PL-SQL que reparta cada parte en una variable distinta. Después imprimirlas por pantalla. Siempre existen los dos espacios en blanco.

9) **Sentencias de control**:
   a. Empresa, **case**: Una sentencia SQL para obtener el número de pedidos que realiza cada compañía de envío y añadir una columna calculada que nos indique lo siguiente:
      i. 150 a 250 → Simple, 250 a 300 → Normal, 300 a 350 → Super
      ii. Se verá algo así:

```
ID                   NOMBRE             CUENTA               CLASIFICACION
-------------------- ------------------ -------------------- -------------
1                    Speedy Express     248                  Simple
2                    United Package     327                  Super
3                    Federal Shipping   255                  Normal

3 rows selected
```

   b. **Fechas**, **while**: A partir de dos variables: mes y año de tipo number generar las fechas correspondientes de ese mes. Imprimirlas, teniendo en cuenta el número de días del mes, si el año es bisiesto, etc.
      i. **Para imprimir dbms_output.putline('texto' || var);**

10) **Funciones definidas por el usuario**:

a. Escribir una función que reciba el idpedido y calcule el importe total. Tener en cuenta el cargo y el cálculo de los detalles.

b. Crear con SQL Developer una tabla Socios(login, pass): Escribir una función que compruebe si existe un socio en la base de datos. La función recibirá el login y las password por parámetro, devolverá un 0 cuando el usuario exista y la password sea correcta, 1 cuando solo exista el login y un 2 en caso contrario, no existe el socio.

c. Academia: Calcular el sueldo final de cada profesor sabiendo que tienen 12 pagas + 500 € por beneficios. El cálculo se realiza en una función. La función recibirá el código del profesor, si no existe devolverá -1. Después en un script escribir una consulta que utilice la función.

d. Academia: Calcular el sueldo de los profesores de la siguiente manera: si el sueldo es mayor o igual que 1000 se aumenta un 5% y si es menor 8%. El cálculo se realiza en una función, si no existe devolverá -1. Después en un script escribir una consulta que utilice la función.

11) **Procedimientos Almacenados**:

a. Empresa: Definir un procedimiento que recibe como parámetro el id de un pedido y lo elimina. Utilizar transacciones y control de excepciones.

12) **Paquetes**:

a. Escribir un paquete: Gestion_pedidos, tendrá los siguientes procedimientos almacenados:
    i. Añadir un pedido, con los siguientes parámetros:
        1. Idcliente, nombre del empleado y el nombre de la compañía y el país.
            a. el cargo será un valor fijo de 30 €.
            b. La fecha pedido, la de hoy.
            c. La fecha envío pedido + 7 días.
            d. La fecha de entrega pedido + 15 días.
    ii. Añadir un detalle a un pedido: Devolverá un mensaje si lo ha conseguido agregar o no.
    iii. Eliminar un detalle de un pedido.
        1. Devolverá un mensaje si lo ha conseguido o no.
    iv. Añadir también a este paquete la función que calcula el importe de un pedido.

13) **Triggers**:

a. Empresa: Definir un Trigger que se dispare cuando se inserta un nuevo detalle será el encargado de actualizar las unidades en existencia de la tabla de productos.

14) Cursores estáticos:

a. Implementar el resultado de la Vista b mediante un bloque de PL/SQL. Utilizar un cursor parametrizado, capturar excepciones e imprimir por pantalla una pantalla similar a esta (utilizar los métodos open, close, fetch):

anonymous block completed
ALFKI:,7%
ANATR:,41%
ANTON:,48%
AROUT:,62%
BERGS:3,57%
BLAUS:,64%
BLONP:2,2%
….
….
WARTH:1,61%
WELLI:,86%
WHITC:2,35%
WILMK:,69%
WOLZA:,43%
Ganancia Total: 1304759
Número total de clientes: 89
Media por Cliente: 14660,21

15) **EXECUTE IMMEDIATE**: A partir de la tabla pedidos crear una tabla para cada país, repartir los registros a cada tabla según corresponda.

Utilizar un cursor estático para obtener los distintos países. Luego mediante SQL Dinámico, crear un tabla para cada país seleccionando los registros: **create tabla TbpedidosAlemania as select * from Tbpedidos where paisdestinatario = 'Alemania';**

Para las comillas simples utilizar el Chr(39).

Tener en cuenta los espacios en blanco de los países.

El cursor estático utilizarlo con la sentencia **FOR IN.**

El script debe emitir el siguiente informe:

anonymous block completed
Se ha creado la tabla TbPedidosAlemania con 122 registros
Se ha creado la tabla TbPedidosArgentina con 16 registros
Se ha creado la tabla TbPedidosAustria con 40 registros
Se ha creado la tabla TbPedidosBelgica con 19 registros
Se ha creado la tabla TbPedidosBrasil con 83 registros
Se ha creado la tabla TbPedidosCanada con 30 registros
Se ha creado la tabla TbPedidosDinamarca con 18 registros
Se ha creado la tabla TbPedidosEspanya con 23 registros
Se ha creado la tabla TbPedidosEstadosUnidos con 122 registros
Se ha creado la tabla TbPedidosFinlandia con 23 registros
Se ha creado la tabla TbPedidosFrancia con 76 registros
Se ha creado la tabla TbPedidosIrlanda con 19 registros
Se ha creado la tabla TbPedidosItalia con 28 registros
Se ha creado la tabla TbPedidosMexico con 28 registros
Se ha creado la tabla TbPedidosNoruega con 6 registros
Se ha creado la tabla TbPedidosPolonia con 6 registros
Se ha creado la tabla TbPedidosPortugal con 13 registros
Se ha creado la tabla TbPedidosReinoUnido con 55 registros
Se ha creado la tabla TbPedidosSuecia con 38 registros
Se ha creado la tabla TbPedidosSuiza con 18 registros
Se ha creado la tabla TbPedidosVenezuela con 46 registros

Se puede implementar otro parecido para borrar.

16) **UTL_FILE**: Utilizando un cursor exportar a un fichero de texto el contenido de la tabla de clientes.

17) **Dbms_sql**: Utilizando este paquete exportar a un fichero de texto (CSV) como en la práctica anterior, pero en este caso la consulta será cualquiera y se escribirán los nombres de las columnas de los campos en la primera línea del fichero. Separar las columnas con un **;**