**SENTENCIA SELECT**

**\*\* La base de datos usada para los ejemplos esta al final de este documento**

Lenguaje de manipulación de datos (DML)

Las sentencias de lenguaje de manipulación de datos (DML) son utilizadas para gestionar datos dentro de los schemas. Algunos ejemplos:

* INSERT - para insertar datos a una tabla.
* SELECT - para obtener datos de una base de datos.
* UPDATE - para modificar datos existentes dentro de una tabla.
* DELETE - elimina todos los registros de la tabla; no borra los espacios asignados a los registros.

### Consultar contenido en Tablas: SELECT

Esta sentencia recupera filas de la base de datos y realiza la selección de una o varias filas o columnas de una o varias tablas.

SELECT expressions

FROM tables

[WHERE conditions];

|  |
| --- |
| SELECT \* FROM Productos; -- Selecciona todo  SELECT IdProducto, Descripcion FROM Productos; -- Columnas código y nombre |

El \* específica que se deben devolver todas las columnas de todas las tablas y vistas de la cláusula FROM.

**Sintaxis Select:**

La sintaxis completa de la instrucción SELECT es compleja, aunque las cláusulas principales se pueden resumir del modo siguiente:

SELECT [ALL | DISTINCT]

[TOP (expression) [PERCENT] [WITH TIES]]

<select\_list>

[ INTO new\_table ]

[ FROM { <table\_source> } [ ,...n ] ]

[ WHERE <search\_condition> ]

[ GROUP BY <Group> [ ,...n ]

[ HAVING < search\_condition > ]

[ ORDER BY {order\_by\_expression | column\_position [ASC | DESC]}

[ ,...n ]]

**WHERE**

La cláusula WHERE se usa para filtrar valores, permite especificar una condición para manipular los datos que cumplen un criterio específico.

SELECT nombreColumna, nombreColumna

FROM nombreTabla

WHERE (nombreColumna operador valor);

Sólo las filas que cumplen las condiciones contribuyen con datos al conjunto de resultados. Los datos de las filas que no cumplen las condiciones no se usan.

|  |
| --- |
| SELECT \* FROM Clientes  WHERE Localidad = 'Madrid';  --Si los valores son textuales, se necesitan comillas simples (aunque la --mayoría de sistemas de bases de datos aceptan también comillas dobles)  -- Si los valores son numéricos, no se ponen comillas  SELECT \* FROM Clientes  WHERE IDCliente >= 2; |

WHEREpermite hacer consultas en varias tablas:

|  |
| --- |
| SELECT D.IDdepartamento, D.nombre, E.IDEmpleado,E.Nombre  FROM Departamento D, Empleados E  WHERE D.IDdepartamento = E. IDdepartamento |

WHERE- Se usa con Select, Update y Delete.

**Operadores**

Existen distintos tipos de operadores según la expresión que se quiere realiza, los operadores más utilizados son:

|  |  |
| --- | --- |
| **Operadores** | **Descripcion** |
| = | Igual |
| <> | Distinto |
| != | Distinto |
| > | Mayor que |
| >= | Mayor o igual que |
| < | Menor que |
| <= | Menor o igual que |
| IN ( ) | Coincide con un valor en una lista dada |
| NOT | Niega la condición |
| BETWEEN | Dentro de un rango |
| IS NULL | es NULL |
| IS NOT NULL | No es NULL |
| LIKE | Coincide con un patrón ( Se pueden usar comodines % \_) |
| EXISTS | La condición se cumple si la subconsulta devuelve al menos una fila |
| AND | TRUE si ambas expresiones son TRUE |
| OR | TRUE si cualquiera de las dos expresiones booleanas es TRUE |
| SOME | TRUE si alguna de las comparaciones de un conjunto es TRUE |
| ALL | TRUE si el conjunto completo de de comparaciones es TRUE. |

Cuando las expresiones son complejas y tienen múltiples operadores hay que tener en cuenta la prioridad de dichos operadores para determinar la secuencia de ejecución de las operaciones. El orden de ejecución puede afectar de manera significativa al valor resultante.

Utilizando los paréntesis se puede modificar este orden de prioridad. Ver más información sobre la prioridad de los operadores en:

<https://docs.microsoft.com/es-es/sql/t-sql/language-elements/operator-precedence-transact-sql?view=sql-server-ver15>

|  |
| --- |
| -- Listar clientes de Madrid o Barcelona con crédito mayor de 5000  SELECT \* FROM Clientes  WHERE Credito = 5000  AND (Localidad='Madrid' OR Localidad='Barcelona'); |

**IN**

Permite especificar múltiples valores en la cláusula WHERE:

SELECT nombreColumna

FROM nombreTabla

WHERE nombreColumna [NOT] IN (valor1, valor2...);

|  |
| --- |
| -- Seleccionar los clientes que tienen como ciudad Madrid o Barcelona  SELECT \* FROM Clientes  WHERE Localidad IN ('Madrid', 'Barcelona');  -- Seleccionar los clientes que no tienen como ciudad Valencia, Alicante o Sevilla  SELECT \* FROM Clientes  WHERE Localidad NOT IN ('Valencia', 'Alicante', 'Sevilla' ); |

**BETWEEN**

Se usa para seleccionar valores dentro de un rango. Los valores pueden ser números, texto o fechas.

SELECT nombreColumna

FROM nombreTabla

WHERE nombreColumna [NOT] BETWEEN valor1 AND valor2;

|  |
| --- |
| -- Productos cuyo precio está entre 400 y 600 euros:  SELECT \* FROM Productos WHERE PrecioActual BETWEEN 400 AND 600;  -- Productos cuyo precio no está entre 200 y 400 euros:  SELECT \* FROM Productos WHERE PrecioActual NOT BETWEEN 200 AND 400; |

**LIKE**

Determina si una cadena de caracteres específica coincide con un patrón especificado. Se pueden usar **Comodines**:

% número cualquiera de caracteres

\_ un único carácter

[] uno del intervalo o los valores

[^] no pertenece a ese conjunto.

Ejemplos:

|  |  |  |  |
| --- | --- | --- | --- |
| Tipo de coincidencia | Modelo | Coincide | No coincide |
| Varios caracteres | 'a%a' | 'aa', 'aBa', 'aBBBa' | 'aBC' |
| Carácter especial | 'a[%]a' | 'a%a' | 'aaa' |
| Varios caracteres | 'ab%' | 'abcdefg', 'abc' | 'cab', 'aab' |
| Un solo carácter | 'a\_a' | 'aaa', 'a3a', 'aBa' | 'aBBBa' |
| Un solo dígito | 'a\_a' | 'a0a', 'a1a', 'a2a' | 'a10a' |
| Rango de caracteres | '[a-z]' | 'f', 'p', 'j' | '2', '&' |
| Fuera de un rango | '[^a-z]' | '9', '&', '%' | 'b', 'a' |
| Distinto de un dígito | '[^0-9]' | 'A', 'a', '&', '~' | '0', '1', '9' |
| Combinada | 'a[^b-m]\_' | 'An9', 'az0', 'a99' | 'abc', 'aj0 |

**ORDER BY lista\_orden [ASC | DESC]**

La cláusula ORDER BY se usa para ordenar los resultados por una o más columnas. Por defecto, se ordenan en orden ascendente ASC. Para ordenar los resultados en orden descendente se usa la palabra DESC.

|  |
| --- |
| SELECT \* FROM Clientes  WHERE Localidad IN ('Madrid', 'Barcelona');  SELECT \* FROM Clientes  ORDER BY Localidad, Nombre; |

Los resultados ordenados por varias columnas se ordenan primero por la primera indicación de orden, y dentro de este mismo orden, se ordenan por la segunda indicación de orden, y así sucesivamente.

**SELECT DISTINCT**

La sentencia SELECT DISTINCT devuelve registros donde los valores de todas las columnas elegidas sean diferentes.

SELECT DISTINCT nombreColumna, nombreColumna

FROM nombreTabla;

En una tabla, una columna puede contener valores duplicados, y en ocasiones sólo deseas mostrar valores únicos.

|  |
| --- |
| SELECT DISTINCT Localidad FROM Clientes  WHERE Credito =5000  ORDER BY Localidad DESC; |

**SELECT TOP** (expresión) [PERCENT] [WITH TIES]

La sentencia SELECT TOP devolverá un primer conjunto o porcentaje de filas del resultado de la consulta. *Expresión* puede ser un número o un porcentaje de filas. Si la consulta contiene una cláusula ORDER BY, se devolverán las primeras filas de expresión o el porcentaje de filas de expresión ordenadas por la cláusula ORDER BY.

Por ejemplo, TOP (10) devolvería las 10 primeras filas. TOP (8) PERCENT, devolvería el 8% superior del conjunto de resultados completo.

WITH TIES es Opcional. Se devuelven las filas empatadas en último lugar dentro del conjunto de resultados limitado. Debe usar este argumento con la cláusula ORDER BY. WITH TIES puede hacer que se devuelvan más filas que las del valor especificado en *expression*.

|  |
| --- |
| -- Devuelve el 50% de los clientes que coincida con el mismo comercial ordenados por localidad.  SELECT TOP(50) PERCENT  Idcliente, Nombre  FROM Clientes  WHERE IDComercial = '7654'  ORDER BY Localidad;  -- Devuelve los datos de los que tengan el mismo comercial.  -- comprobar que pasa con 3 con TIES (saca 4)y con 3 sin TIES saca 3.  SELECT TOP(3) WITH TIES  idcliente, Nombre, IDComercial, Localidad  FROM Clientes  WHERE IDComercial = '7654'  ORDER BY Localidad;  -- Por ejemplo, si dos filas adicionales coinciden con los valores de las -- columnas ORDER BY en la fila 5, es decir, coincide con el apellido del -- 5º, el conjunto de resultados contendrá 7 filas. |

**Aliases**

Los alias SQL se usan para renombrar temporalmente una tabla o columna, de esta forma son más legibles, especialmente para consultas algo más complejas.

Para columnas:

SELECT nombreColumna **AS** alias

FROM nombreTabla

Para tablas:

SELECT nombreColumna

FROM nombreTabla **AS** alias

Los alias son útiles cuando hay más de una tabla involucrada, se usan funciones SQL, se usan Joins, los nombres de las columnas son largos o poco legibles o se combinan dos o más columnas.

|  |
| --- |
| -- Alias Columnas  -- Sin Alias  SELECT nombre  FROM empleados  WHERE nombre = 'Lopez';  -- Con Alias  SELECT nombre AS Empleado  FROM empleados  WHERE nombre = 'Lopez';  -- Alias Tablas  SELECT Nombre, 'Ciudad - ' + Localidad AS Ciudad  FROM Clientes; |

**Funciones de Agregado**

Las funciones de agregación en SQL permiten efectuar operaciones sobre un conjunto de resultados, pero devolviendo un único valor agregado para todos ellos.

Si exceptuamos la función COUNT, todas las funciones de agregado ignoran los valores NULL. Si la colección de valores está vacía, Count devuelve cero y las demás NULL.

Las funciones de agregado solo se pueden usar como expresiones en las situaciones siguientes:

* La lista de selección de una instrucción SELECT (una subconsulta o una consulta externa).
* Cláusula HAVING.

<https://docs.microsoft.com/es-es/sql/t-sql/functions/aggregate-functions-transact-sql?view=sql-server-ver15>

Las funciones de agregación básicas que soportan todos los gestores de datos son las siguientes:

**COUNT -** Nos devuelve el número de registros de la selección. El tipo de datos que devuelve es int.

SELECT COUNT ({[[ALL | DISTINCT] expression] | \*})

FROM nombreTabla;

ALL - Aplica la función de agregado a todos los valores. ALL es el valor predeterminado.

DISTINCT - Especifica que COUNT devuelva el número de valores únicos no NULL.

*expression* Es una expresión de cualquier tipo excepto text, image o ntext. No se permite utilizar funciones de agregado ni subconsultas.

\* - Especifica que se deben contar todas las filas para devolver el número total de filas de una tabla. COUNT(\*) no acepta parámetros y no se puede utilizar con DISTINCT. COUNT (\*) devuelve el número de filas de una tabla especificada sin deshacerse de las duplicadas. Cuenta cada fila por separado. Se incluyen las filas que contienen valores NULL.

* **AVG -** Utilizada para calcular el promedio de los valores de un campo determinado. calcula la media de un conjunto de valores dividiendo la suma de estos valores por el recuento de valores no NULL. El tipo de valor devuelto viene determinado por el tipo del resultado evaluado de expression.

SELECT AVG([ALL | DISTINCT] expression)

FROM nombreTabla;

ALL - Aplica la función de agregado a todos los valores. ALL es el valor predeterminado.

DISTINCT - Especifica que AVG tenga en cuenta sólo una instancia única de un valor, sin importar el número de veces que aparezca el valor.

*expression* - Es una expresión de la categoría de tipo de datos numérico exacto o numérico aproximado, excepto para el tipo de datos bit. No se permite utilizar funciones de agregado ni subconsultas.

* **SUM -** Nos devuelve la suma de todos los valores de un campo determinado

SELECT SUM([ALL | DISTINCT] expression)

FROM nombreTabla;

ALL - Aplica la función SUM a todos los valores. ALL es el valor predeterminado.

DISTINCT - Especifica que SUM devuelve la suma de los valores únicos.

*expression* - Se trata de una constante, columna o función y cualquier combinación de operadores aritméticos, bit a bit y de cadena. expression es una expresión de la categoría de tipo de datos numérico exacto o numérico aproximado, excepto para el tipo de datos bit. No se permiten funciones de agregado ni subconsultas.

* **MAX -** Nos devuelve el valor más alto de un campo especificado.

SELECT MAX(expression)

FROM nombreTabla;

*expression* - Se trata de una constante, nombre de columna o función y cualquier combinación de operadores aritméticos, bit a bit y de cadena. MAX se puede usar con columnas de tipo numeric, carácter, uniqueidentifier y datetime, pero no con columnas de tipo bit. No se permiten funciones de agregado ni subconsultas.

Devuelve un valor de tipo igual a expression. MAX pasa por alto los valores NULL.

* **MIN -** Nos devuelve el valor más bajo de un campo especificado.

SELECT MIN([ALL | DISTINCT] expression)

FROM nombreTabla;

Se comporta igual que MAX.

Ejemplos:

|  |
| --- |
| SELECT COUNT(\*) AS TotalFilas, COUNT(IDPedido) AS FilasNoNulas,-- Da igual, pues no hay nulos.  MIN(FECHA\_PEDIDO) AS FechaMin, MAX(FECHA\_PEDIDO) AS FechaMax  FROM Pedidos;  --Hallar el salario medio, el mínimo, el máximo y la media  --aritmética de todos los empleados.  SELECT AVG(salario) as SalarioMedio, min(salario) as Min, max(salario) as  MAx, (min(salario)+max(salario))/2 as Media  FROM Empleados; |

Habitualmente, antes de aplicar estas funciones se constituyen uno o más grupos con las filas de la tabla, estos grupos se hacen con la cláusula GROUP BY. El resultado dará tantas filas como grupos se hayan hecho.

Si no está la cláusula Group By, se sobreentiende que hay un solo grupo formado por todas las filas de la tabla que satisfagan el predicado de la cláusula WHERE. En este caso el resultado es una sola fila con tantas columnas como columnas hayamos incluido en la cláusula SELECT.

**Cláusula GROUP BY**

GRUPO BY se usa en una instrucción SELECT para recopilar datos en varios registros y agrupar los resultados por una o más columnas.

SELECT expression1, expression2, ... expression\_n,

aggregate\_function (expression)

FROM tables

[WHERE conditions]

GROUP BY expression1, expression2, ... expression\_n;

Combina los registros con valores idénticos, en la lista de campos especificados en By, en un único registro. Para cada registro se crea un valor sumario si se incluye una función SQL agregada, como por ejemplo Sum o Count, en la instrucción SELECT.

Los valores de resumen se omiten si no existe una función de agregado en la instrucción SELECT.   
Los valores Null en los campos GROUP BY se agrupan y no se omiten. No obstante, los valores Null no se evalúan en ninguna de las funciones SQL agregadas.

|  |
| --- |
| -- Saber la camtidad de productos  -- Saber la cantidad de productos  SELECT SUM(Stock) AS "Total"  FROM productos;  --Hallar el salario medio, el mínimo, el máximo y la media  --aritmética por departamento.  SELECT AVG(salario) as SalarioMedio, min(salario) as Min, max(salario) as  MAx, (min(salario)+max(salario))/2 as Media  FROM Empleados  GROUP BY IDDepartamento; |

Un campo de la lista de campos GROUP BY puede referirse a cualquier campo de las tablas que aparecen en la cláusula FROM, incluso si el campo no está incluido en la instrucción SELECT, siempre y cuando la instrucción SELECT incluya al menos una función SQL agregada.

Todos los campos de la lista de campos de SELECT deben o bien incluirse en la cláusula GROUP BY o como argumentos de una función de agregación.

|  |
| --- |
| -- Número de empleados por departamento  SELECT IDdepartamento, Count(IDdepartamento) AS "TotalDpto"  FROM Empleados  GROUP BY IDdepartamento;  select \* from empleados; |

Se utiliza la cláusula **WHERE** para excluir aquellas filas que no desea agrupar, y la cláusula **HAVING** para filtrar los registros una vez agrupados.

<https://docs.microsoft.com/es-es/sql/t-sql/queries/select-group-by-transact-sql?view=sql-server-ver15>

**Cláusula HAVING**

La cláusula HAVING se usa en combinación con la cláusula GROUP BY para restringir los grupos de filas devueltas. Una vez formados los grupos para cada uno de ellos se evalúan las condiciones de la cláusula HAVING, y los grupos que la cumplan generan una fila en la tabla final.

HAVING Especifica una condición de búsqueda para un grupo o agregado. HAVING solo se puede utilizar con la instrucción SELECT, es similar a WHERE, determina qué registros se seleccionan. Una vez que los registros se han agrupado utilizando GROUP BY, HAVING determina cuales de ellos se van a mostrar.

SELECT expression1, expression2, ... expression\_n,

aggregate\_function (expression)

FROM tables

[WHERE conditions]

GROUP BY expression1, expression2, ... expression\_n

HAVING search\_condition;

*expresión1, expresión2, ... expresión\_n* Las expresiones que no están encapsuladas dentro de una función agregada y deben incluirse en la cláusula GROUP BY.

*función agregada* Puede ser una función como las funciones SUM , COUNT , MIN , MAX o AVG .

*WHERE condiciones* Opcional. Las condiciones que deben cumplirse para que se seleccionen los registros.

Normalmente, HAVING se usa con una cláusula GROUP BY. Cuando no se usa GROUP BY, hay un solo grupo implícito agregado. Solo puede aplicar una cláusula HAVING a las columnas que también aparecen en la cláusula GROUP BY o en una función de agregado. Todos los campos de la lista de campos de SELECT deben o bien incluirse en la cláusula GROUP BY o como argumentos de una función de agregación.

|  |
| --- |
| -- Número de empleados por departamento  SELECT IDdepartamento, Count(IDdepartamento) AS "TotalDpto"  -- Count de una cadena vacia da cero.  FROM Empleados  GROUP BY IDdepartamento  HAVING count(IDEmpleado) > 1;  -- La cláusula HAVING filtrará los resultados para que solo se devuelvan  -- los departamentos con más de 1 empleado.  --select \* from empleados;  SELECT IDDepartamento, Count(IDEmpleado) AS "TotalDpto"  FROM Empleados  GROUP BY IDDepartamento  HAVING count(IDEmpleado) > 1;  -- La cláusula HAVING filtrará los resultados  -- para que solo se devuelvan los departamentos con más de 1 empleados.  -- En las dos selects anteriores fijaros en la diferencia que hay entre  -- que el Count lo haga por el el campo Idempleado y el IdDepartamento.  SELECT Oficio, COUNT(\*) AS "Number of employees"  FROM Empleados  GROUP BY Oficio  HAVING COUNT(\*) > 1;  -- Este ejemplo de cláusula HAVING utiliza la función COUNT para devolver el oficio y  -- el número de empleados que hay con ese oficio.  -- La cláusula HAVING filtrará los resultados para que solo se devuelvan los oficios  -- que tengan mas de un empleado. |

<https://docs.microsoft.com/es-es/sql/t-sql/queries/select-having-transact-sql?view=sql-server-ver15>

**Subselect o Subquerys**

En SQL Server, una subconsulta es una consulta dentro de una consulta. Puede crear subconsultas dentro de sus declaraciones SQL. Estas subconsultas pueden residir en la cláusula WHERE, la cláusula FROM o la cláusula SELECT.

## Cláusula WHERE

La mayoría de las veces, la subconsulta se encontrará en la cláusula WHERE. Estas subconsultas también se denominan subconsultas anidadas:

|  |
| --- |
| -- Nos da el Id de producto y la descripción de aquellos productos de los que  -- nos hayan hecho pedidos de más de tres unidades.  SELECT p.IDproducto, p.Descripcion  FROM productos p  WHERE p.IDproducto IN  (SELECT p.IDproducto  FROM PEDIDOS p  WHERE p.UNIDADES > 3); |

NOTA: Esta subconsulta podría haberse escrito alternativamente como una combinación INNER JOIN.

## Cláusula FROM

También se puede encontrar una subconsulta en la cláusula FROM. Se denominan **vistas en línea**:

|  |
| --- |
| --Vistas en línea: la subconsulta va en la clausula FROM  --Buscar el nombre de cliente y la suma de unidades pedidas por cliente  SELECT c.nombre, subS1.totalP  FROM clientes c,  (SELECT IDCliente, SUM(pedidos.UNIDADES) AS totalP  FROM pedidos  GROUP BY IDCliente) subS1  WHERE subS1.IDCliente = c.IDCliente; |

## 

## Cláusula SELECT

También se puede encontrar una subconsulta en la cláusula SELECT. Estos se utilizan generalmente cuando desea recuperar un cálculo utilizando una función agregada como la función SUM , COUNT , MIN o MAX , pero no desea que la función agregada se aplique a la consulta principal.

|  |
| --- |
| --Busca el nombre del empleado y el salario maximo y la diferencia de  --cada salario con el maximo salario.  SELECT nombre, salario,  (SELECT MAX(salario)  FROM empleados )-salario subquery2  FROM empleados ; |

El truco para colocar una subconsulta en la cláusula select es que la subconsulta debe devolver un solo valor. Esta es la razón por la que una función agregada como la función SUM , COUNT , MIN o MAX se usa comúnmente en la subconsulta.

**SELECT - Orden de procesamiento**

El orden de las cláusulas en la instrucción SELECT es importante. Se puede omitir cualquiera de las cláusulas opcionales pero, cuando se utilizan, deben aparecer en el orden que marca la sintaxis.

SELECT [ALL | DISTINCT]

[TOP (expression) [PERCENT] [WITH TIES]]

<select\_list>

[ INTO new\_table ]

[ FROM { <table\_source> } [ ,...n ] ]

[ WHERE <search\_condition> ]

[ GROUP BY <Group> [ ,...n ]

[ HAVING < search\_condition > ]

[ ORDER BY {order\_by\_expression | column\_position [ASC | DESC]}

[ ,...n ]]

Los pasos siguientes muestran el orden de procesamiento de una instrucción SELECT:

1. FROM : selecciona las tablas donde se va a buscar
2. ON : selecciona las tablas donde se va a buscar
3. JOIN : selecciona las tablas donde se va a buscar
4. WHERE: excluye las filas que no cumplen su condición de búsqueda.
5. GROUP BY: recopila las filas seleccionadas en un grupo para cada valor único de la cláusula GROUP BY.
6. HAVING excluye las filas que no cumplen su condición de búsqueda una vez aplicado el Group by.
7. SELECT selecciona los datos que queremos de esas filas.
8. DISTINCT selecciona los distintos
9. ORDER BY los ordena
10. TOP devuelve el número de filas que le indique

Más información:

[https://docs.microsoft.com/es-es/sql/t-sql/queries/select-transact-sql?view=sql-server-ver15#logical-processing-order-of-the-select-statement](https://docs.microsoft.com/es-es/sql/t-sql/queries/select-transact-sql?view=sql-server-ver15" \l "logical-processing-order-of-the-select-statement)

**Base de datos usada para los ejemplos**

DROP DATABASE IF EXISTS Empresa\_2024;

GO

CREATE DATABASE Empresa\_2024;

GO

USE Empresa\_2024;

CREATE TABLE DEPARTAMENTOS

( IDDepartamento TINYINT,

Nombre VARCHAR(14),

Localidad VARCHAR(10),

PRIMARY KEY (IDDepartamento));

CREATE TABLE EMPLEADOS

( IDEmpleado INT,

Nombre VARCHAR(8),

Oficio VARCHAR(10),

FechaAlta DATE,

Salario SmallMoney,

Comision DECIMAL(4,2),

IDDepartamento TINYINT, -- Id departamento

PRIMARY KEY (IDEmpleado),

FOREIGN KEY (IDDepartamento) REFERENCES DEPARTAMENTOS(IDDepartamento) ON DELETE SET NULL

);

CREATE TABLE CLIENTES

(IDCliente INT,

Nombre VARCHAR(25),

Localidad VARCHAR(14),

IDComercial INT,

Debe Money,

Haber Money,

Credito SmallMoney,

PRIMARY KEY (IDCliente),

FOREIGN KEY (IDComercial) REFERENCES EMPLEADOS(IDEmpleado) ON DELETE SET NULL

);

CREATE TABLE PRODUCTOS

(IDProducto INT,

Descripcion VARCHAR(30),

PrecioActual MONEY,

Stock INT,

PRIMARY KEY (IDProducto)

);

CREATE TABLE PEDIDOS

(IDPedido INT,

IDProducto INT,

IDCliente INT,

UNIDADES INT,

FECHA\_PEDIDO DATE,

PRIMARY KEY (IDPedido),

FOREIGN KEY (IDProducto) REFERENCES PRODUCTOS(IDProducto) ON DELETE CASCADE,

FOREIGN KEY (IDCliente) REFERENCES CLIENTES(IDCliente) ON DELETE CASCADE

);

-- DEPARTAMENTOS

INSERT INTO DEPARTAMENTOS VALUES(10, 'CONTABILIDAD', 'BARCELONA');

INSERT INTO DEPARTAMENTOS VALUES(20, 'INVESTIGACION', 'VALENCIA');

INSERT INTO DEPARTAMENTOS VALUES(30, 'VENTAS', 'MADRID');

INSERT INTO DEPARTAMENTOS VALUES(40, 'PRODUCCION', 'SEVILLA');

-- EMPLEADOS

INSERT INTO EMPLEADOS VALUES (7839,'REY', 'PRESIDENTE','1981-11-17',6000, NULL, NULL);

INSERT INTO EMPLEADOS VALUES (7698,'GARRIDO', 'DIRECTOR', '1981-05-01',3850.12,NULL, 30);

INSERT INTO EMPLEADOS VALUES (7782,'MARTINEZ','DIRECTOR', '1981-06-09',2450, NULL, 10);

INSERT INTO EMPLEADOS VALUES (7499,'ALONSO', 'VENDEDOR', '1981-02-23',1400, 0.04,30);

INSERT INTO EMPLEADOS VALUES (7521,'LOPEZ', 'EMPLEADO', '1981-05-08',1350.50,NULL,10);

INSERT INTO EMPLEADOS VALUES (7654,'MARTIN', 'VENDEDOR', '1981-09-28',1500, 0.16, 30);

INSERT INTO EMPLEADOS VALUES (7844,'CALVO', 'VENDEDOR', '1981-09-08',1800, 0, 30);

INSERT INTO EMPLEADOS VALUES (7876,'GIL', 'ANALISTA', '1982-05-06',3350, NULL, 20);

INSERT INTO EMPLEADOS VALUES (7900,'JIMENEZ', 'EMPLEADO', '1983-03-24',1400, NULL, 20);

-- CLIENTES

INSERT INTO CLIENTES VALUES (101, 'DISTRIBUCIONES GOMEZ', 'MADRID', 7499, 0,0,5000);

INSERT INTO CLIENTES VALUES (102, 'LOGITRONICA S.L', 'BARCELONA', 7654,0,0,5000);

INSERT INTO CLIENTES VALUES (103, 'INDUSTRIAS LACTEAS S.A.', 'LAS ROZAS', 7844,0,0, 10000);

INSERT INTO CLIENTES VALUES (104, 'TALLERES ESTESO S.A.', 'SEVILLA', 7654, 0, 0, 5000);

INSERT INTO CLIENTES VALUES (105, 'EDICIONES SANZ', 'BARCELONA', 7499, 0,0,5000);

INSERT INTO CLIENTES VALUES (106, 'SIGNOLOGIC S.A.', 'MADRID', 7654,0,0,5000);

INSERT INTO CLIENTES VALUES (107, 'MARTIN Y ASOCIADOS S.L.', 'ARAVACA' , 7844,0,0, 10000);

INSERT INTO CLIENTES VALUES (108, 'MANUFACTURAS ALI S.A.', 'SEVILLA', 7654, 0, 0, 5000);

-- PRODUCTOS

INSERT INTO PRODUCTOS VALUES(10,'MESA DESPACHO MOD. GAVIOTA', 550, 50);

INSERT INTO PRODUCTOS VALUES (20, 'SILLA DIRECTOR MOD. BUFALO', 670, 25);

INSERT INTO PRODUCTOS VALUES (30, 'ARMARIO NOGAL DOS PUERTAS', 460, 20);

INSERT INTO PRODUCTOS VALUES (40, 'MESA MODELO UNI�N',340, 15);

INSERT INTO PRODUCTOS VALUES (50, 'ARCHIVADOR CEREZO',1050, 20);

INSERT INTO PRODUCTOS VALUES (60, 'CAJA SEGURIDAD MOD B222', 280, 15);

INSERT INTO PRODUCTOS VALUES (70, 'DESTRUCTORA DE PAPEL A3',450, 25);

INSERT INTO PRODUCTOS VALUES (80, 'MODULO ORDENADOR MOD. ERGOS', 550, 25);

--PEDIDOS

INSERT INTO PEDIDOS VALUES(1000, 20, 103, 3, '1999-10-06');

INSERT INTO PEDIDOS VALUES(1001, 50, 106, 2, '1999-10-06');

INSERT INTO PEDIDOS VALUES(1002, 10, 101, 4, '1999-10-07');

INSERT INTO PEDIDOS VALUES(1003, 20, 105, 4, '1999-10-16');

INSERT INTO PEDIDOS VALUES(1004, 40, 106, 8, '1999-10-20');

INSERT INTO PEDIDOS VALUES(1005, 30, 105, 2, '1999-10-20');

INSERT INTO PEDIDOS VALUES(1006, 70, 103, 3, '1999-11-03');

INSERT INTO PEDIDOS VALUES(1007, 50, 101, 2, '1999-11-06');

INSERT INTO PEDIDOS VALUES(1008, 10, 106, 6, '1999-11-16');

INSERT INTO PEDIDOS VALUES(1009, 20, 105, 2, '1999-11-26');

INSERT INTO PEDIDOS VALUES(1010, 40, 102, 3, '1999-12-08');

INSERT INTO PEDIDOS VALUES(1011, 30, 106, 2, '1999-12-15');

INSERT INTO PEDIDOS VALUES(1012, 10, 105, 3, '1999-12-06');

INSERT INTO PEDIDOS VALUES(1013, 30, 106, 2, '1999-12-06');

INSERT INTO PEDIDOS VALUES(1014, 20, 101, 4, '2000-01-07');

INSERT INTO PEDIDOS VALUES(1015, 70, 105, 4, '2000-01-16');

INSERT INTO PEDIDOS VALUES(1017, 20, 105, 6, '2000-01-20');

;

SELECT \* FROM DEPARTAMENTOS;

SELECT \* FROM EMPLEADOS;

SELECT \* FROM CLIENTES

SELECT \* FROM PRODUCTOS;

SELECT \* FROM PEDIDOS;