



Internacionalízate

BASE DE DATOS- SQL SERVER

ACTIVIDAD PROCESUAL HITO 3

INTEGRANTES: ISRAEL ALEJANDRO ZAMBRANA

UNIVERSIDAD: FRANZ TAMAYO

DOCENTE: WILLIAM RODDY BARRA PAREDES

SEMESTRE: SEGUNDO SEMESTRE

CÓDIGO DE ESTUDIANTE: SIS11073818

PARALELO: 3

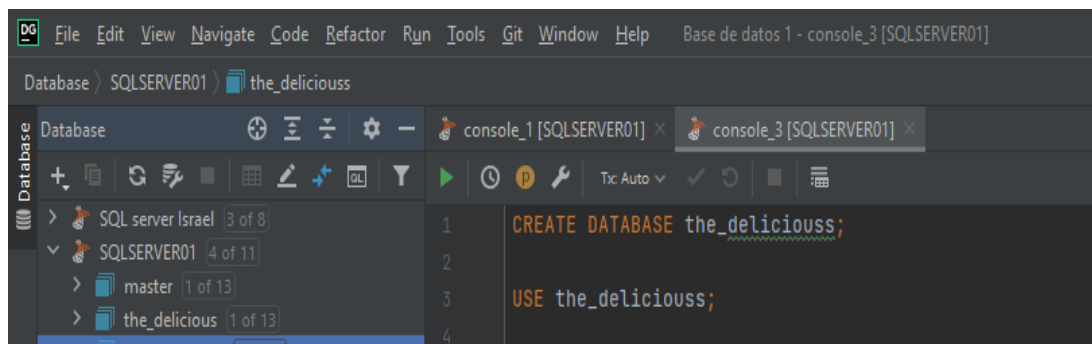
CONSIGNA

Diseñe un sistema de Base de Datos Relacional utilizando el gestor de Base de Datos SQL Server teniendo como premisa el uso de buenas prácticas en diseño de la base de datos aplicados al siguiente escenario.

Una pequeña empresa de comida rápida de nombre the delicious desea implementar un nuevo sistema para poder administrar los PEDIDOS de sus productos.

PASO A PASO (THE DELICIOUS)

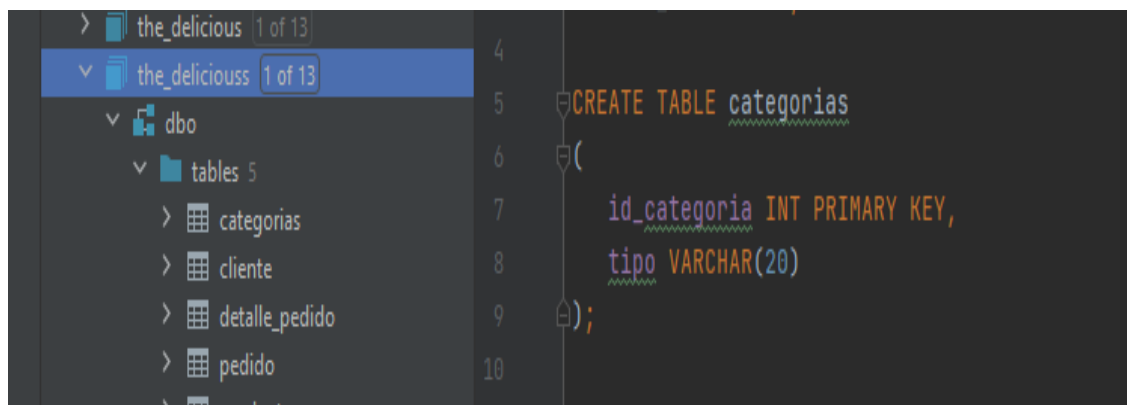
1ER PASO: crear un data base con el nombre de (the delicious) con un USE the delicious



The screenshot shows the SQL Server Enterprise Manager interface. In the left pane, the 'Database' folder is expanded, showing the 'the_delicious' database. The right pane shows the SQL console with the following code:

```
1 CREATE DATABASE the_delicious;  
2  
3 USE the_delicious;  
4
```

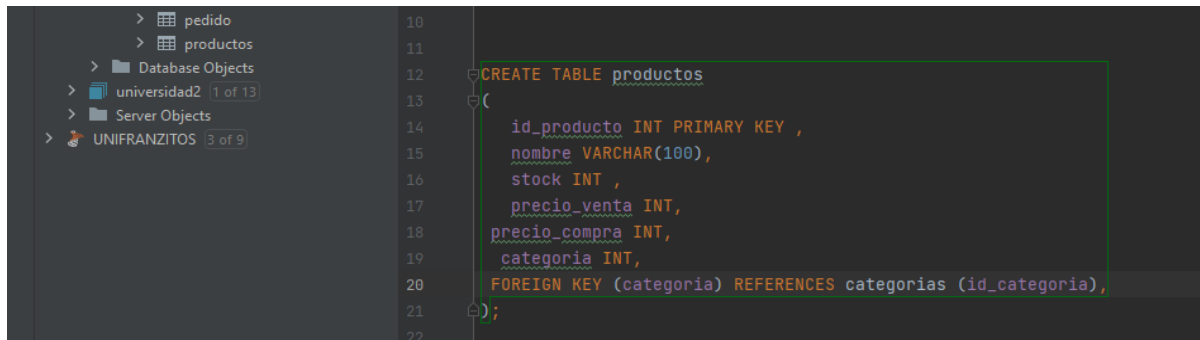
2DO PASO: crear una tabla con el nombre de categorías



The screenshot shows the SQL Server Enterprise Manager interface. In the left pane, the 'the_delicious' database is expanded, showing the 'dbo' schema and the 'categorias' table. The right pane shows the SQL console with the following code:

```
4  
5 CREATE TABLE categorias  
6 (  
7     id_categoria INT PRIMARY KEY,  
8     tipo VARCHAR(20)  
9 );  
10
```

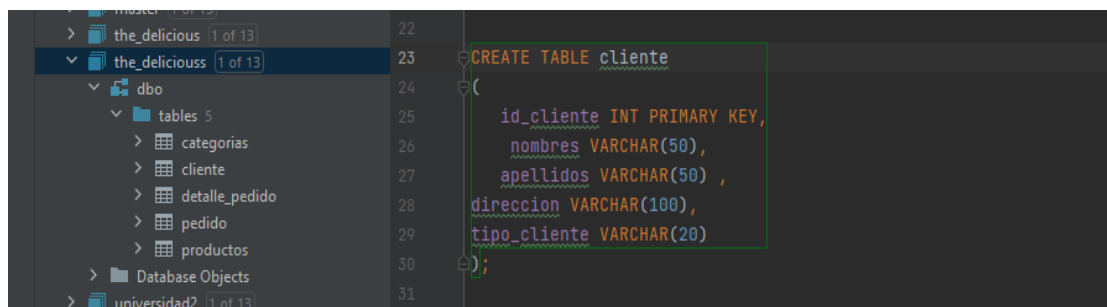
3ER PASO: crear una tabla con el nombre de productos



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Database Objects' tree is expanded, showing the 'productos' table. On the right, the SQL script for creating the 'productos' table is displayed. The script is as follows:

```
10  
11  
12 CREATE TABLE productos  
13 (  
14     id_producto INT PRIMARY KEY ,  
15     nombre VARCHAR(100),  
16     stock INT ,  
17     precio_venta INT,  
18     precio_compra INT,  
19     categoria INT,  
20     FOREIGN KEY (categoria) REFERENCES categorias (id_categoria),  
21 );  
22
```

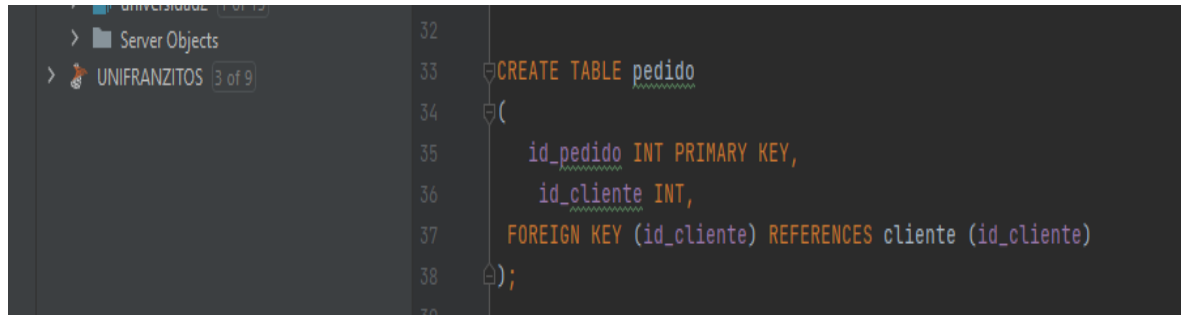
4TO PASO: crear una tabla con el nombre de clientes



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Database Objects' tree is expanded, showing the 'cliente' table. On the right, the SQL script for creating the 'cliente' table is displayed. The script is as follows:

```
22  
23 CREATE TABLE cliente  
24 (  
25     id_cliente INT PRIMARY KEY,  
26     nombres VARCHAR(50),  
27     apellidos VARCHAR(50) ,  
28     direccion VARCHAR(100),  
29     tipo_cliente VARCHAR(20)  
30 );  
31
```

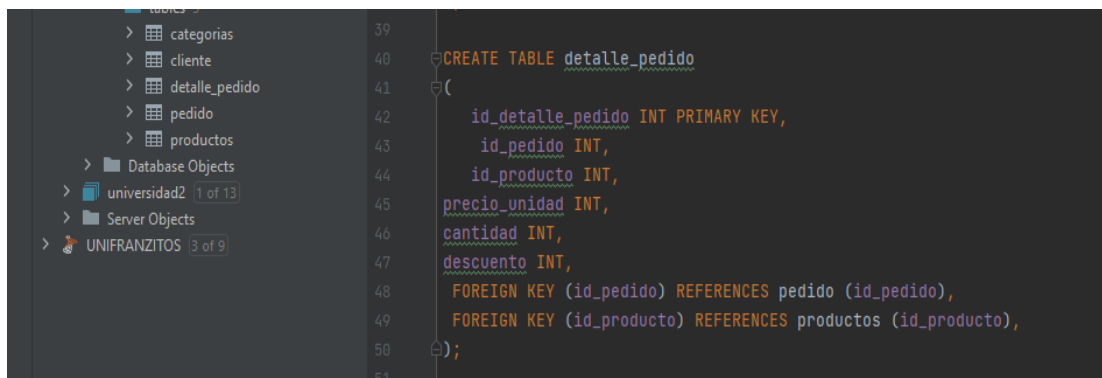
5TO PASO: crear una tabla de pedido



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Server Objects' tree is expanded to 'UNIFRANZITOS'. The main pane displays the SQL script for creating the 'pedido' table. The script is as follows:

```
32  
33 CREATE TABLE pedido  
34 (  
35     id_pedido INT PRIMARY KEY,  
36     id_cliente INT,  
37     FOREIGN KEY (id_cliente) REFERENCES cliente (id_cliente)  
38 );  
39
```

6TO PASO: crear una tabla con el detalle del pedido



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Server Objects' tree is expanded to 'UNIFRANZITOS'. The main pane displays the SQL script for creating the 'detalle_pedido' table. The script is as follows:

```
39  
40 CREATE TABLE detalle_pedido  
41 (  
42     id_detalle_pedido INT PRIMARY KEY,  
43     id_pedido INT,  
44     id_producto INT,  
45     precio_unidad INT,  
46     cantidad INT,  
47     descuento INT,  
48     FOREIGN KEY (id_pedido) REFERENCES pedido (id_pedido),  
49     FOREIGN KEY (id_producto) REFERENCES productos (id_producto),  
50 );  
51
```

LOS INSERTS DE CADA TABLA

```
INSERT INTO categorias (id_categoria, tipo)
VALUES ('1','electrodomestico');
INSERT INTO categorias (id_categoria, tipo)
VALUES ('2','juguetes');
INSERT INTO categorias (id_categoria, tipo)
VALUES ('3','verduras');

INSERT INTO productos (id_producto,
nombre,stock,precio_venta,precio_compra,categoria)
VALUES ('1','refrigerador','15','1500','1000','1');
INSERT INTO productos (id_producto,
nombre,stock,precio_venta,precio_compra,categoria)
VALUES ('2','microondas','4','800','500','1');
INSERT INTO productos (id_producto,
nombre,stock,precio_venta,precio_compra,categoria)
VALUES ('3','los_vengadores','2','2500','1700','2');

INSERT INTO cliente (id_cliente,
nombres,apellidos,direccion,tipo_cliente)
VALUES ('1','nombre_cliente1','apellidos_cliente1','6 de agosto
Avenida','GOLD');
INSERT INTO cliente (id_cliente,
nombres,apellidos,direccion,tipo_cliente)
VALUES ('2','nombre_cliente2','apellidos_cliente2','Plaza
Avaroa','VIP');
INSERT INTO cliente (id_cliente,
nombres,apellidos,direccion,tipo_cliente)
VALUES ('3','nombre_cliente3','apellidos_cliente3','Plaza del
estudiante','NORMAL');
INSERT INTO cliente (id_cliente,
nombres,apellidos,direccion,tipo_cliente)
VALUES ('4','nombre_cliente4','apellidos_cliente4','Teatro al
aire','NORMAL');

INSERT INTO pedido (id_pedido, id_cliente)
VALUES ('1','1');
INSERT INTO pedido (id_pedido, id_cliente)
VALUES ('2','2');

INSERT INTO detalle_pedido
(id_detalle_pedido,id_pedido,id_producto,precio_unidad,cantidad,descue
nto)
VALUES ('1','1','1','1000','2','0');
INSERT INTO detalle_pedido
(id_detalle_pedido,id_pedido,id_producto,precio_unidad,cantidad,descue
nto)
VALUES ('2','1','2','800','1','0');
INSERT INTO detalle_pedido
(id_detalle_pedido,id_pedido,id_producto,precio_unidad,cantidad,descue
nto)
VALUES ('3','2','2','800','1','0');
```

DISEÑO DE BASE DE DATOS.

1.1. Dado el detalle explicado en la parte inicial de este documento debería generar el modelo entidad relación.

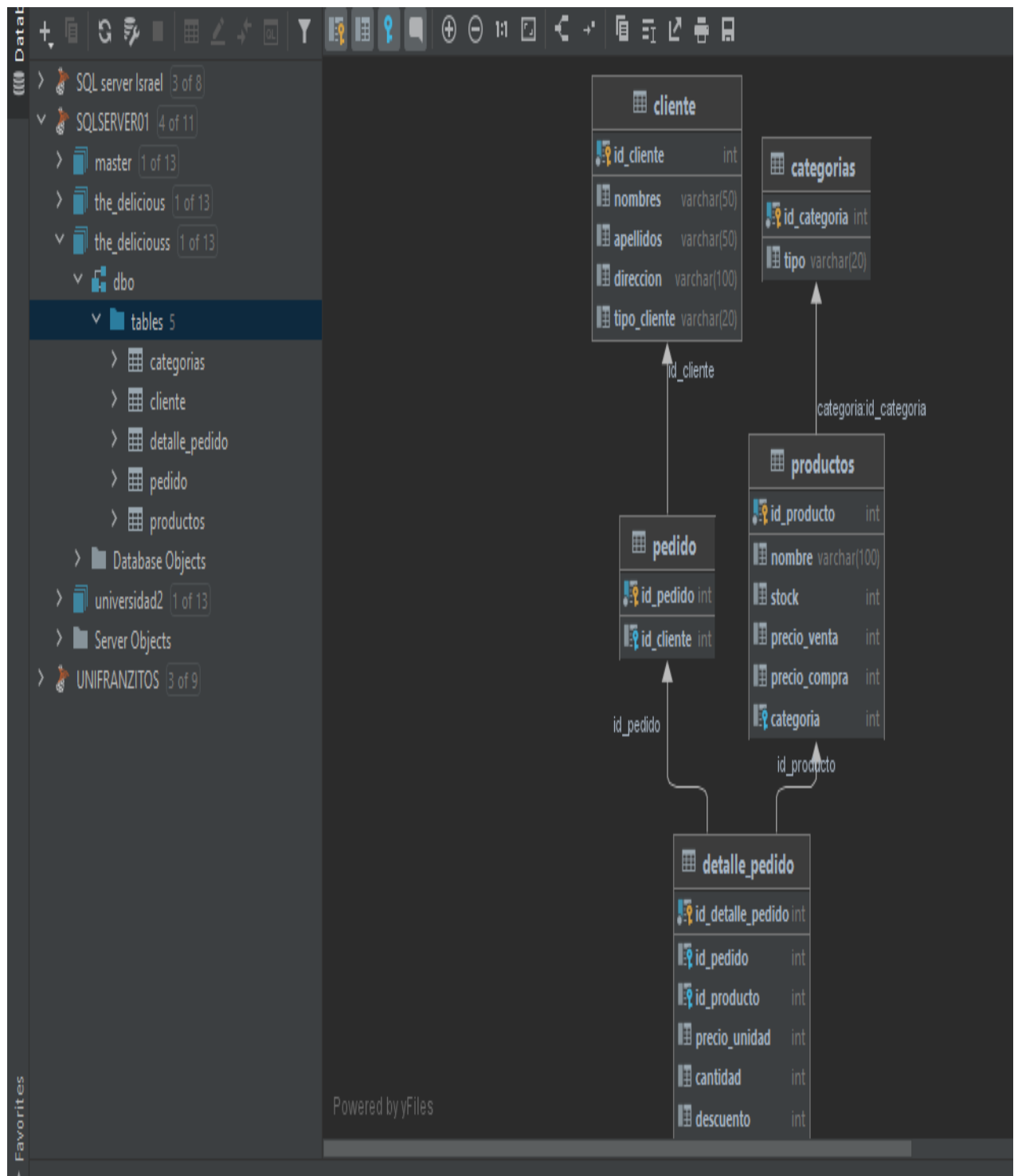
- Para poder generar el diagrama entidad relación, podría utilizar la plataforma Diagrams

Link del diagrama:

https://drive.google.com/file/d/1u_OwspPguh9J-qyAeziNPVDZ1mQleX5I/view?usp=sharing

1.2. Después de generar el modelo lógico de la base de datos. El mismo debería quedar similar a lo siguiente:

- Utilizar Datagrip para poder generar el diagrama



2. MANEJO DE CONCEPTOS

2.1. Que es el modelo entidad relación.

R. Un modelo entidad-relación es una herramienta para el modelo de datos, la cual facilita la representación de entidades de una base de datos.

2.2. Que es el modelo lógico en bases de datos relacionales.

R. Se trata de una estructura formada por filas y columnas que almacena los **datos** referentes a una determinada entidad o relación del mundo real. Representa una propiedad que posee esa tabla. ... Se corresponde con la idea de campo o columna.

2.3. Describe y menciona que formas(shapes) se utiliza para graficar un modelo entidad relación.

R. Dentro de los diagramas de entidad-relación, las relaciones se usan para documentar la interacción entre dos entidades. Las relaciones generalmente son verbos como asignar, asociar o rastrear y proporcionan información útil que no podría discernirse solo con los tipos de entidades.



2.4. Qué es una función de agregación.

R. Una función de agregación es una función invocada por SQL que toma valores que dependen de todas las filas que selecciona la consulta y devuelve información sobre estas filas. El servidor de bases de datos da soporte a las funciones de agregación que escribe el usuario, denominadas agregados definidos por el usuario .

2.5. Muestre ejemplo del uso de 2 funciones de agregación.

- **COUNT:** devuelve el número total de filas seleccionadas por la consulta.
- **MIN:** devuelve el valor mínimo del campo que especifiquemos.
- **MAX:** devuelve el valor máximo del campo que especifiquemos.
- **SUM:** suma los valores del campo que especifiquemos. Sólo se puede utilizar en columnas numéricas.
- **AVG:** devuelve el valor promedio del campo que especifiquemos. Sólo se puede utilizar en columnas numéricas.

2.6. Muestre un ejemplo del uso de JOINS.

R. Unir tablas para obtener los datos necesarios para una consulta, script o procedimiento almacenado es un concepto clave a medida que de aprende sobre el desarrollo en SQL Server. En pocas palabras, las uniones se realizan normalmente en la cláusula FROM de una tabla o vista para las sentencias SELECT , INSERT ... SELECT , SELECT ... INTO , UPDATE y DELETE . En versiones anteriores de SQL Server, la lógica de unión también podría haberse incluido en la cláusula WHERE con la sintaxis = (INNER JOIN), * = (LEFT OUTER JOIN), = * (RIGHT OUTER JOIN), EJEMPLO CON INNER JOIN.

```
SELECT TOP 100 P.ProductID,  
  
P.Name,  
  
P.ListPrice,  
  
P.Size,  
  
P.ModifiedDate,  
  
SOD.UnitPrice,  
  
SOD.UnitPriceDiscount,  
  
SOD.OrderQty,  
  
SOD.LineTotal  
  
FROM Sales.SalesOrderDetail SOD  
  
INNER JOIN Production.Product P  
  
ON SOD.ProductID = P.ProductID  
  
WHERE SOD.UnitPrice > 1000  
  
ORDER BY SOD.UnitPrice DESC
```

2.7. Qué es SQL y No SQL.

R. SQL permite combinar de forma eficiente diferentes tablas para extraer información relacionada, mientras que No SQL no lo permite o muy limitadamente. No SQL permite distribuir grandes cantidades de información; mientras que SQL facilita distribuir bases de datos relacionales.

2.8. A que se refiere cuando se habla de ISO, que es una ISO.

R. ISO calidad de software de datos.

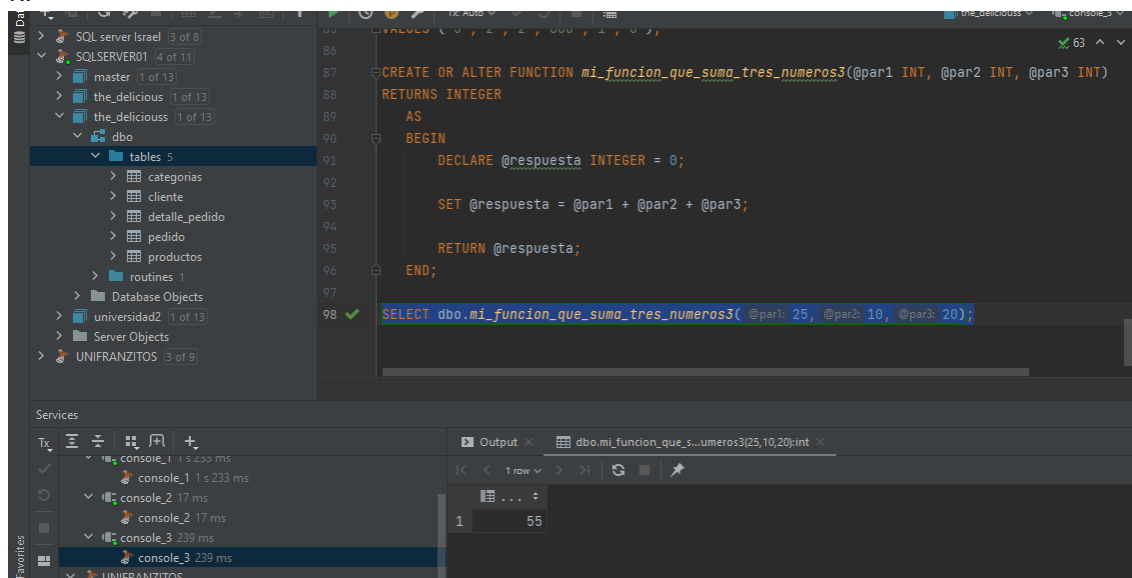
El modelo de Calidad de Datos representa los cimientos sobre los cuales se construye un sistema para la evaluación de un producto de datos. En un modelo de Calidad de Datos se establecen las características de Calidad de Datos que se deben tener en cuenta a la hora de evaluar las propiedades de un producto de datos determinado.

2.9. Quien creo el modelo entidad relación o mas conocido como E-R

R. Peter Chen (también conocido como Peter Pin-Shan Chen) actualmente se desempeña como miembro de la facultad de la Universidad Carnegie Mellon ubicada en Pittsburgh y se le atribuye el desarrollo del modelo ER para el diseño de bases de datos en el año 70.

2.10. Crear una función que permita sumar 3 números .

R.



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Server Objects' tree is expanded to show the 'dbo' schema and its 'tables' folder. The main pane displays the SQL code for creating a function named 'mi_funcion_que_suma_tres_numeros3'. The code is as follows:

```
CREATE OR ALTER FUNCTION mi_funcion_que_suma_tres_numeros3(@par1 INT, @par2 INT, @par3 INT)
RETURNS INTEGER
AS
BEGIN
    DECLARE @respuesta INTEGER = 0;

    SET @respuesta = @par1 + @par2 + @par3;

    RETURN @respuesta;
END;
```

Below the code, the execution command is shown: `SELECT dbo.mi_funcion_que_suma_tres_numeros3(@par1: 25, @par2: 10, @par3: 20);`. The 'Output' pane at the bottom shows the result of the function call, which is 55.

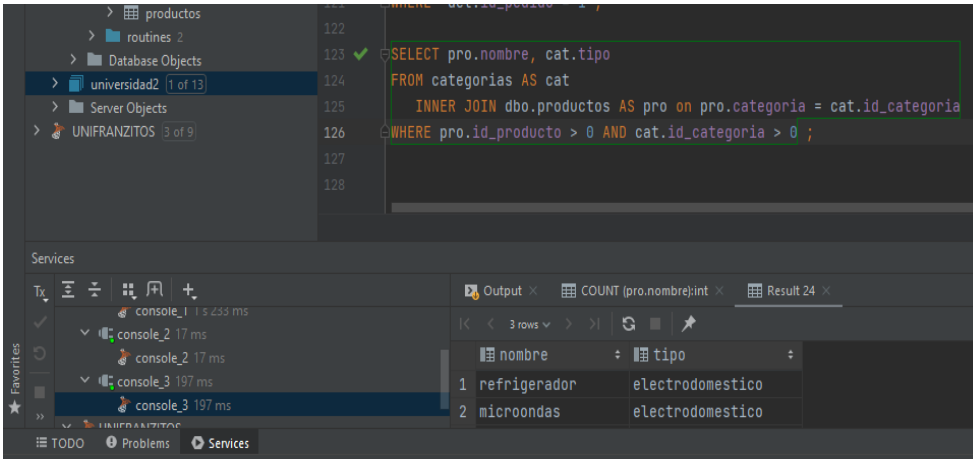
3. MANEJO DE CONSULTAS

3.1. Mostrar los productos (Nombre y stock) con stock mayor igual a 10.

R.

3.2. Mostrar el nombre del producto y la categoría de los productos pertenecen a la categoría de "electrodomésticos".

R.



The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the database structure, including 'universidad2' and 'UNIFRANZITOS'. The right pane shows a SQL query:

```
SELECT pro.nombre, cat.tipo
FROM categorias AS cat
INNER JOIN dbo.productos AS pro on pro.categoria = cat.id_categoria
WHERE pro.id_producto > 0 AND cat.id_categoria > 0 ;
```

The bottom pane shows the results of the query in a table with two columns: 'nombre' and 'tipo'.

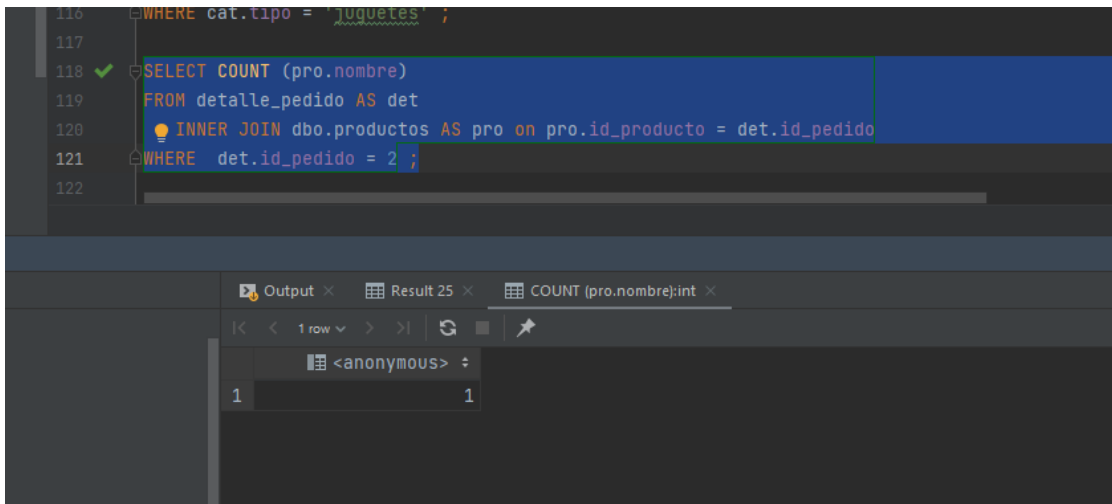
nombre	tipo
1 refrigerador	electrodomestico
2 microondas	electrodomestico

3.3. Que productos(nombre) tiene el pedido con id igual a = 1.

R.

3.4. Cuantos(count) productos tiene el pedido con id igual a = 2.

R.



The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the database structure, including 'universidad2' and 'UNIFRANZITOS'. The right pane shows a SQL query:

```
SELECT COUNT (pro.nombre)
FROM detalle_pedido AS det
INNER JOIN dbo.productos AS pro on pro.id_producto = det.id_pedido
WHERE det.id_pedido = 2 ;
```

The bottom pane shows the results of the query in a table with two columns: 'id' and 'count'.

id	count
1	1

3.5. Crear una función que permita sumar 3 números.

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Database Objects' tree is expanded to 'dbo' > 'tables 5'. The main pane displays the following T-SQL code:

```

87 CREATE OR ALTER FUNCTION mi_funcion_que_suma_tres_numeros3(@par1 INT, @par2 INT, @par3
88 RETURNS INTEGER
89 AS
90 BEGIN
91     DECLARE @respuesta INTEGER = 0;
92
93     SET @respuesta = @par1 + @par2 + @par3;
94
95     RETURN @respuesta;
96 END;
97
98 SELECT dbo.mi_funcion_que_suma_tres_numeros3( @par1: 25, @par2: 10, @par3: 20);

```

Below the code editor, the 'Output' window shows the result of the function call:

1	55

R.

3.6. Crear una función que permita restar 3 números.

R.

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Database Objects' tree is expanded to 'dbo' > 'productos'. The main pane displays the following T-SQL code:

```

99 CREATE OR ALTER FUNCTION mi_funcion_que Resta_tres_numeros3(@par1 INT, @par2 INT, @par3 INT)
100 RETURNS INTEGER
101 AS
102 BEGIN
103     DECLARE @respuesta INTEGER = 0;
104
105     SET @respuesta = @par1 - @par2 - @par3;
106
107     RETURN @respuesta;
108 END;
109
110
111 SELECT dbo.mi_funcion_que Resta_tres_numeros3( @par1: 25, @par2: 10, @par3: 20);

```

Below the code editor, the 'Output' window shows the result of the function call:

1	-5

3.7. Cómo unificaría en una sola función el ejercicio 3.5 y 3.7(los dos anteriores).

R.

