EVACUACION PROCESUAL HITO 3

BASE DE DATOS I

PRESENTA:
RONALD URIEL CHOQUE PACO – SIS6972733

UNIFRANZ
Internacionalízate

INNOVACIÓN
EN EDUCACIÓN

EL ALTO, OCTUBRE DE 2021

SECCIONES

CREACION Y DISEÑO DE TABLAS

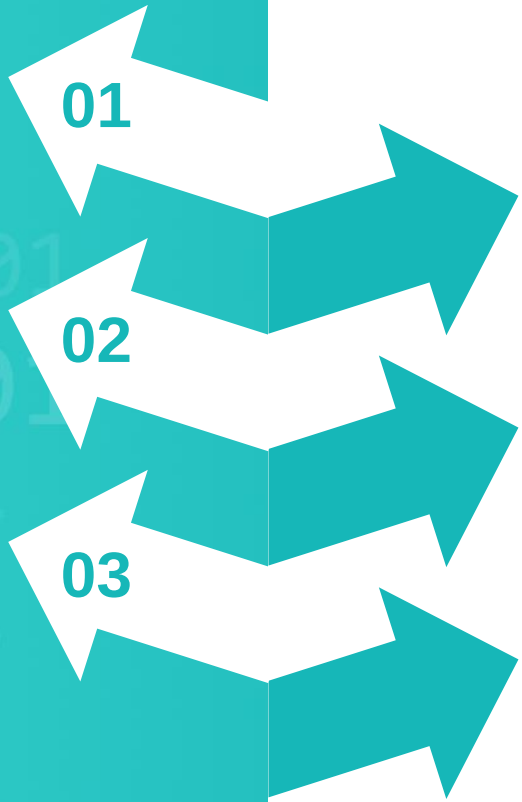
Se procederá a la explicación paso a paso en Datagrip

MANEJO DE CONSULTAS

Se procederá a la explicación paso a paso en Datagrip

MANEJO DE CONCEPTOS

Explicación sobre las preguntas planteadas sobre lo avanzado durante el Hito 3

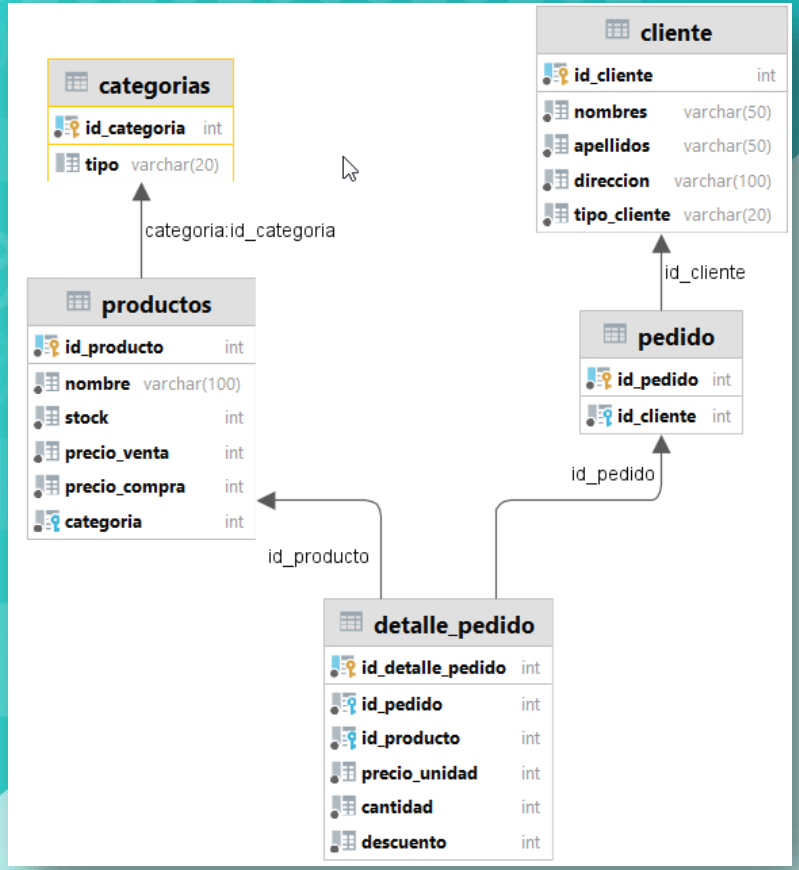


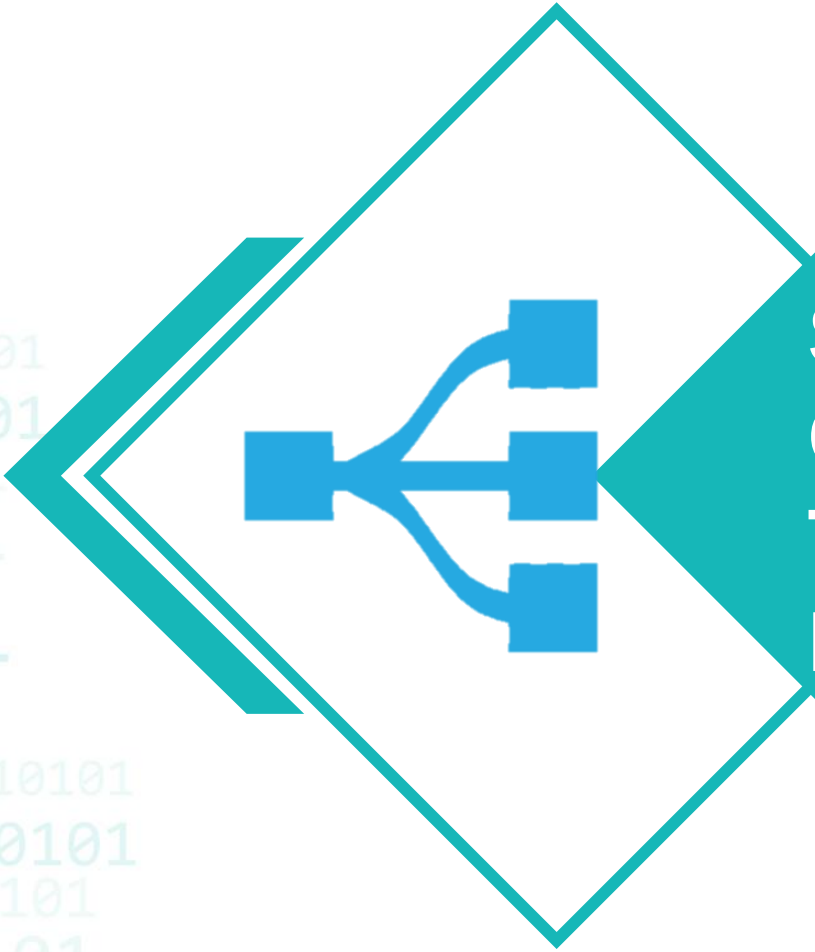
CONSIGAS A CUMPLIR:

The delicious FF	
Problema	En función al escenario se identificó que las posibles entidades son **categorías, productos, cliente, pedido y detalle pedido**, pues un cliente realiza un pedido de productos, en donde cada producto tiene una categoría a la cual pertenece y la descripción del pedido debería tener un detalle de pedido. En tal sentido se deberían crear las siguientes **entidades** y/o **tablas**. • **categorias** • **productos** • **cliente** • **pedido** • **detalle_pedido** Detalle de las entidades.

BREVE EXPLICACION.

Diseñe un sistema de Base de Datos Relacional utilizando e l gestor de Base de Datos **SQL Server** teniendo como premi sa el
uso de buenas prácticas en diseño de la base de datos aplic ados 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.





SECCION 1: CREACION DE TABLAS Y DIAGRAMAS

CONCEPTOS PARA LA CREACION TABLA

IMPORTANTE: se selecciona desde un comienzo es esquema DBO, especificando nuestro rol dentro del proyecto.

El comando CREATE TABLE permite crear la tabla, mientras que el comando USE da lugar al uso de la tabla creada.

El comando CREATE TABLE se usa para la creación de tablas, dichas tablas deben presentar campos o variables las cuales pueden ser VARCHAR (similar a los string en c#) , CHAR() o INTEGER (variable de tipo entero).

NOT NULL es usado para que no se acepte campos vacíos o sin llenar.

PRIMARY KEY o llave primaria es para declarar a un campo el cual no puede repetirse y no puede ser n ulo; ejemplo: para llenar el carnet en la tabla.

FOREIGN KEY o clave foránea, sirve para relacionar tablas y sus columnas, haciendo referencia mediante REFERENCES a la columna de otra tabla en la tabla que se presenta.

CREACION TABLA CATEGORIAS

```sql
CREATE TABLE dbo.categorias
(
    id_categorias INTEGER IDENTITY PRIMARY KEY NOT NULL,
    tipo VARCHAR(50) NOT NULL,
);
```

LLENADO TABLA CATEGORIAS

```sql
INSERT INTO categorias(tipo) VALUES
('electrodometicos'),
('juguetes'),
('verduras');
```

El llenado de la tabla puede acortarse mediante el uso de “,” al final de una cadena de valores, cortando el proceso con “;”

CREACION Y LLENADO TABLA CLIENTE

```sql
CREATE TABLE dbo.cliente
(
    id_cliente INTEGER IDENTITY PRIMARY KEY NOT NULL,
    nombres VARCHAR(50) NOT NULL,
    apellidos VARCHAR(50) NOT NULL,
    direccion VARCHAR(100) NOT NULL,
    tipo_cliente VARCHAR(20) NOT NULL,
);
```

```sql
INSERT INTO cliente(nombres, apellidos, direccion, tipo_cliente) VALUES
('nombre_cliente1', 'apellidos_cliente1', '6 de agosto edificio fernandez', 'GOLD'),
('nombre_cliente2', 'apellidos_cliente2', 'plaza abaroa', 'VIP'),
('nombre_cliente3', 'apellidos_cliente3', 'plaza del estudiante', 'NORMAL'),
('nombre_cliente4', 'apellidos_cliente4', 'teatro al aire libre', 'NORMAL');
```

Se da uso al comando IDENTITY el cual ejecuta un AUTOINCREMENTO de un campo.

El llenado de la tabla puede acortarse mediante el uso de “,” al final de una cadena de valores, cortando el proceso con “;”

CREACION Y LLENADO TABLA PRODUCTOS

```sql
CREATE TABLE dbo.productos
(
    id_productos INTEGER PRIMARY KEY NOT NULL,
    nombre VARCHAR(100) NOT NULL,
    stock INTEGER NOT NULL,
    precio_venta INTEGER NOT NULL,
    precio_compra INTEGER NOT NULL,
    categoria INTEGER NOT NULL,
    FOREIGN KEY (categoria) REFERENCES categorias(id_categorias),
);
```

```sql
INSERT INTO productos(id_productos, nombre, stock, precio_venta, precio_compra, categoria) VALUES
(1, 'refrigerador', 15, 1500, 1000, 1),
(2, 'microonda', 4, 800, 500, 1),
(3, 'los vengadores', 2, 2500, 1700, 2);
```

El llenado de la tabla puede acortarse mediante el uso de “,” al final de una cadena de valores, cortando el proceso con “;”

CREACION Y LLENADO TABLA PEDIDO

```
55  CREATE TABLE dbo.pedido
56  (
57      id_pedido INTEGER PRIMARY KEY NOT NULL,
58      id_cliente INTEGER NOT NULL,
59      FOREIGN KEY (id_cliente) REFERENCES cliente(id_cliente),
60  );
```

```
62  INSERT INTO pedido(id_pedido, id_cliente) VALUES
63  (1, 1),
64  (2, 2);
```

El llenado de la tabla puede acortarse mediante el uso de “,” al final de una cadena de valores, cortando el proceso con “;”

CREACION Y LLENADO TABLA PEDIDO

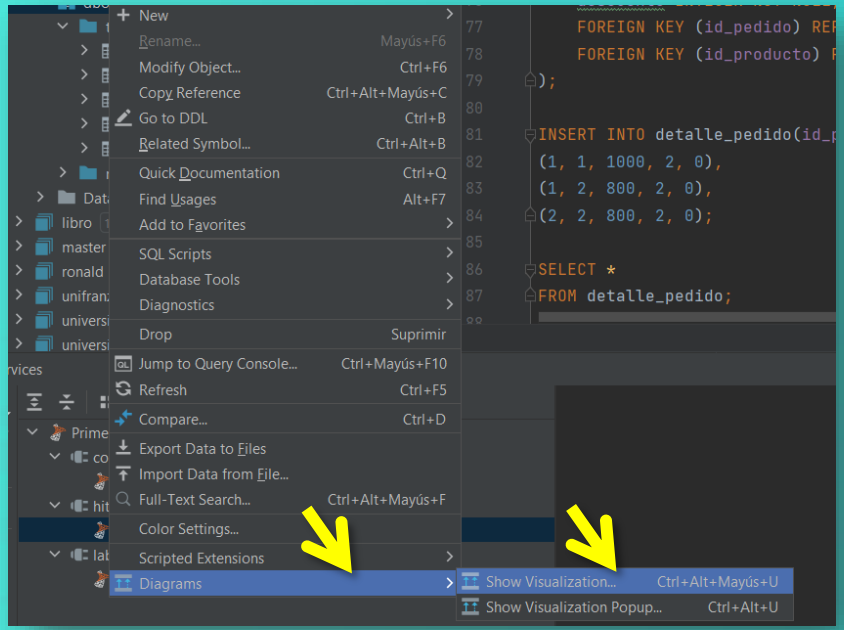
```sql
CREATE TABLE dbo.detalle_pedido
(
    id_detalle_pedido INTEGER IDENTITY PRIMARY KEY NOT NULL,
    id_pedido INTEGER NOT NULL,
    id_producto INTEGER NOT NULL,
    precio_unidad INTEGER NOT NULL,
    cantidad INTEGER NOT NULL,
    descuento INTEGER NOT NULL,
    FOREIGN KEY (id_pedido) REFERENCES pedido(id_pedido),
    FOREIGN KEY (id_producto) REFERENCES productos(id_productos)
);
```

```sql
INSERT INTO detalle_pedido(id_pedido, id_producto, precio_unidad, cantidad, descuento) VALUES
(1, 1, 1000, 2, 0),
(1, 2, 800, 2, 0),
(2, 2, 800, 2, 0);
```

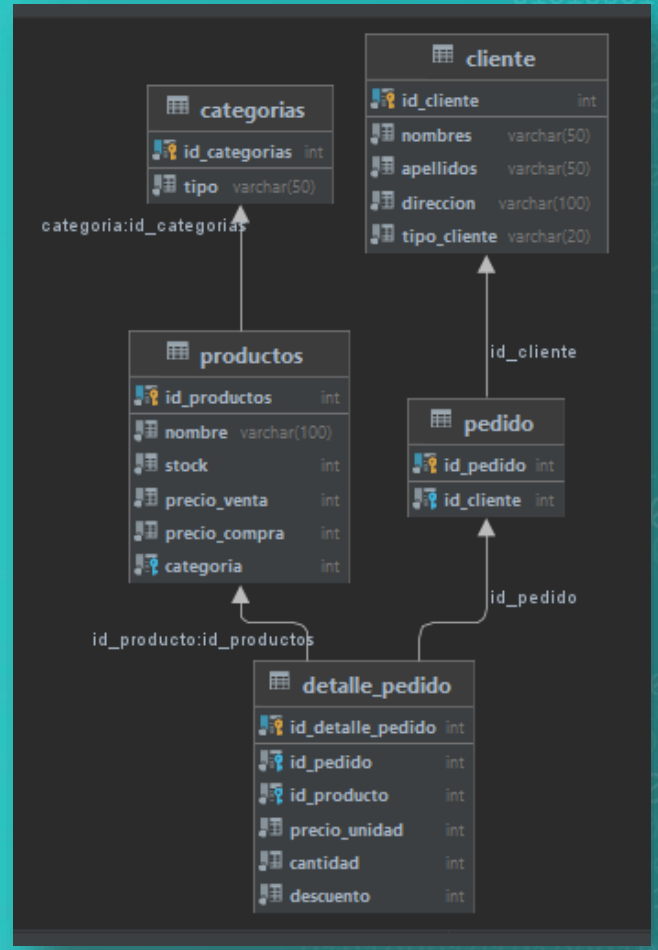
Se da uso al comando IDENTITY el cual ejecuta un AUTOINCREMENTO de un campo.

El llenado de la tabla puede acortarse mediante el uso de “,” al final de una cadena de valores, cortando el proceso con “;”

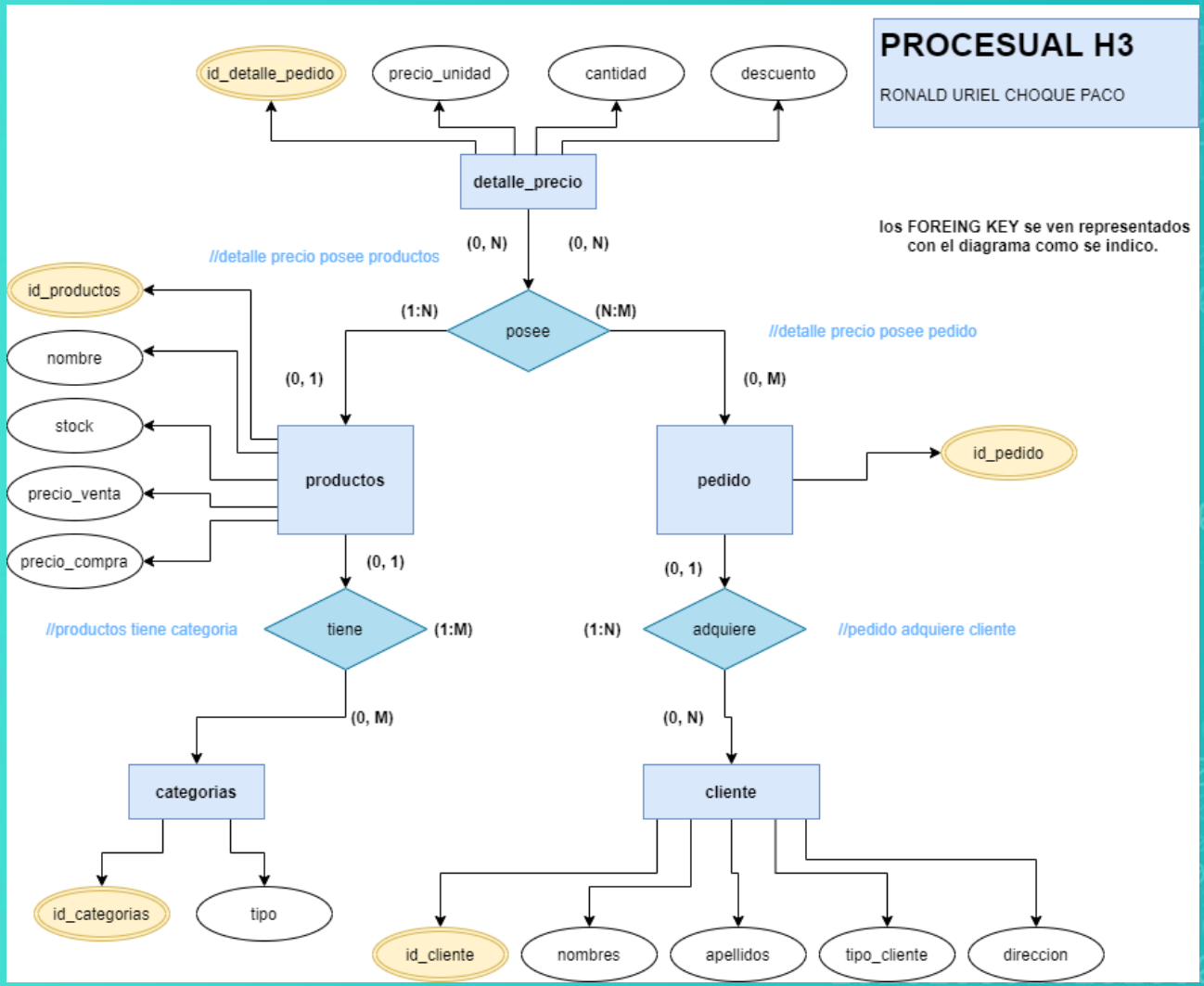
DIAGRAMA DEL PROYECTO



Haciendo un clic derecho sobre la carpeta de tablas, vaya y seleccione DIAGRAMS y la opción SHOW VISUALIZATION para poder ver el diagrama.



MODELO ENTIDAD RELACION



SECCION 2: MANEJO DE CONSULTAS

3.1

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

```sql
SELECT pro.nombre, pro.stock, cat.id_categorias, cat.tipo
FROM productos AS pro
INNER JOIN categorias AS cat ON cat.id_categorias = pro.categoria
WHERE pro.stock >= 10;
```

Ejecucion:

1 row

	nombre	stock	id_categorias	tipo
1	refrigerador	15	1	electrodometicos

3.2

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

```sql
SELECT pro.nombre, pro.categoria, cat.tipo
FROM categorias AS cat
INNER JOIN productos AS pro ON pro.categoria = cat.id_categorias
WHERE cat.tipo = 'electrodometicos';
```

Ejecucion:

2 rows

	nombre	categoria	tipo
1	refrigerador	1	electrodometicos
2	microonda	1	electrodometicos

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


```sql
--Que productos(nombre) tiene el pedido con id igual a = 1.
SELECT pro.nombre, pro.categoria
FROM productos AS pro
INNER JOIN cliente AS cli ON pro.id_productos = cli.id_cliente
WHERE pro.id_productos = 1;
```

Ejecucion:

1 row

	nombre	categoria
1	refrigerador	1

3.4

```sql
SELECT COUNT(*) AS cantidad_de_prod_con_ese_id
FROM productos AS pro
WHERE pro.id_productos = 2;
```

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

Verificacion:

```sql
SELECT pro.id_productos, pro.nombre, pro.categoria
FROM productos AS pro
INNER JOIN cliente AS cli ON pro.id_productos = cli.id_cliente
WHERE pro.id_productos = 2;
```

Ejecucion:

1 row

	cantidad_de_prod_con_ese_id
1	1

1 row | Tx: Auto | DDL

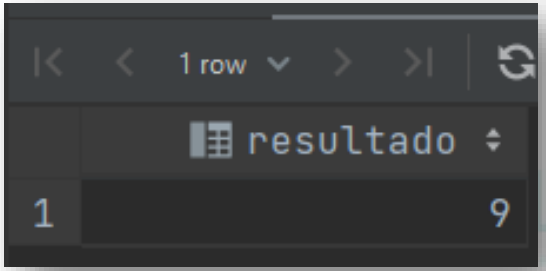
	id_productos	nombre	categoria
1	2	microonda	1

Crear una función que permita sumar 3 números.

3.5

```
119  CREATE OR ALTER FUNCTION funcion_suma_tres_numeros(@par1 INT, @par2 INT, @par3 INT)
120  RETURNS INTEGER
121      AS
122      BEGIN
123          --declara la variable
124          DECLARE @respuesta INTEGER;
125          --operaciones
126          SET @respuesta = @par1 + @par2 + @par3;
127          RETURN @respuesta;
128      END;
129
130  SELECT dbo.funcion_suma_tres_numeros( @num1: 7,  @num2: 14,  @num3: 5) AS resultado;
```

Ejecucion:

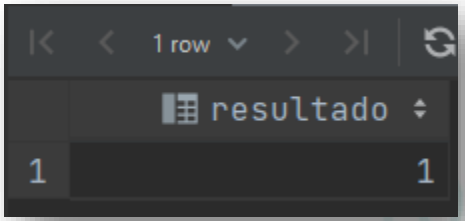


Crear una función que permita restar 3 números.

3.6

```
132  CREATE OR ALTER FUNCTION funcion_resta_tres_numeros(@par1 INT, @par2 INT, @par3 INT)
133  RETURNS INTEGER
134      AS
135      BEGIN
136          --declara la variable
137          DECLARE @respuesta INTEGER;
138          --operaciones
139          SET @respuesta = @par1 + @par2 + @par3;
140          RETURN @respuesta;
141      END;
142
```

Ejecucion:



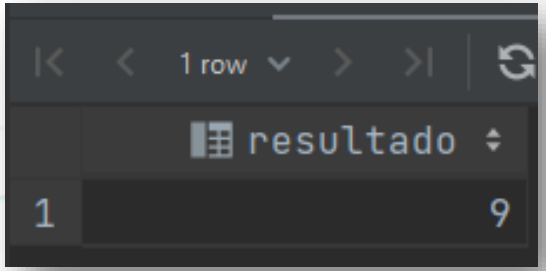
3.7

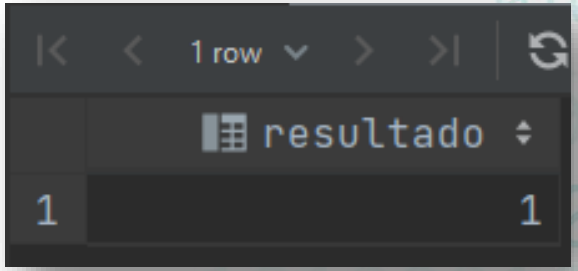
```sql
CREATE OR ALTER FUNCTION funcion_tres_numeros(@num1 INT, @num2 INT, @num3 INT, @opcion INT)
RETURNS INTEGER
    AS
    BEGIN
        --declara la variable
        DECLARE @respuesta INTEGER;
        --operaciones
        IF @opcion = 1
            BEGIN
                SET @respuesta = @num1 + @num2 + @num3;
            end
        ELSE IF @opcion = 2
            BEGIN
                SET @respuesta = @num1 - @num2 - @num3;
            end
        RETURN @respuesta;
    END;

SELECT dbo.funcion_tres_numeros( @num1: 5, @num2: 8, @num3: -4, @opcion: 1) AS resultado;
```

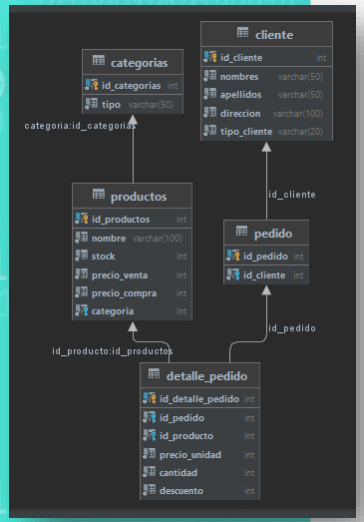
Cómo unificaria en una sola función el ejercicio 3.5 y 3.7(los dos anteriores).

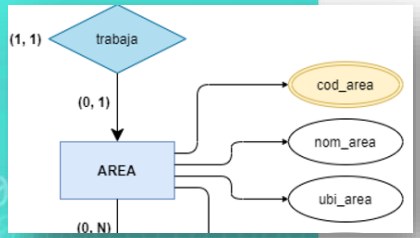
Ejecucion:





SECCION 3: MANEJO DE CONCEPTOS





MANEJO DE CONCEPTOS

Que es el modelo entidad relación.

El modelo **ENTIDAD RELACION** se refiere al conjunto de objetos o **ENTIDADES** y sus respectivas relaciones entre si, ayudando con la creación de esquemas o **DIAGRAMAS** a implementar en un base de datos.

Que es el modelo lógico en bases de datos relacionales.

El **MODELO LÓGICO** es el diagrama que genera automáticamente el gestor de la base de datos, representándolos con las mismas tablas.

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

Los componentes elementales o básicos del modelo E-R son: **LA ENTIDAD** (objeto ya existente), **LA RELACION** (define la dependencia y une a las entidades), **LOS ATRIBUTOS** (características o propiedades de la entidad, son los campos de las tablas en donde también ira incluido el **PRIMARY KEY**).

MANEJO DE CONCEPTOS

Qué es una función de agregación.

Son las funciones que vienen por defecto dentro de SQL (propias del grestor).

Muestre ejemplo del uso de 2 funciones de agregación.

```
146    SELECT COUNT(*) AS total_de_estudiantes
147    FROM estudiantes AS est;
148
149     --Crear una función que obtenga la menor edad de los estudiantes
150    SELECT MIN(est.edad) AS edad_menor
151    FROM estudiantes AS est;
152     --Crear una función que obtenga el promedio de las edades.
153    SELECT AVG(est.edad) AS edad_promedio
154    FROM estudiantes AS est;
155     --Crear una función que obtenga la mayor edad de los estudiantes
156    SELECT MAX(est.edad) AS edad_mayor
157     FROM estudiantes AS est
158    WHERE est.sexo = 'masculino';
```

MANEJO DE CONCEPTOS

Muestre un ejemplo del uso de JOINS.

```
98    SELECT est.nombres, est.apellidos, kar.nota_num
99    FROM kardex AS kar
100   INNER JOIN estudiantes AS est ON kar.id_est_cedula = est.id_est_cedula
101   WHERE kar.estado = 'APROBADO';
```

Qué es SQL y NoSQL.

SQL Es un lenguajeque facilitala comunicacioncon bases de datosrelacionalesdandoaccesoa la manipulacionde los mismos.
NO SQL es un lnguje que deja de lado al lenguaje SQL centrándose como sistema masivo de i nformación.

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

La ISO es la Orgsnizacion Mundial de Normalizacion, la cual elabora normas internaci onales.
NO ISO se refiere al desacato de dichas normas (no siempre con objetivos malos).

MANEJO DE CONCEPTOS

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

Fue Peter Chen en el año 1976, gracias a la necesidad de modelar adecuadame nte los sistemas de información desde ese entonces.

Crear una función que permita sumar 3 números.

Respuesta en manejo de consultas, punto 3.5.

¡GRACIAS POR SU ATENCION!

ronald.choque2111@gmail.com

Cel. 65648933

EL ALTO, OCTUBRE DE 2021

UNIFRANZ
Internacionalízate

INNOVACIÓN
EN EDUCACIÓN