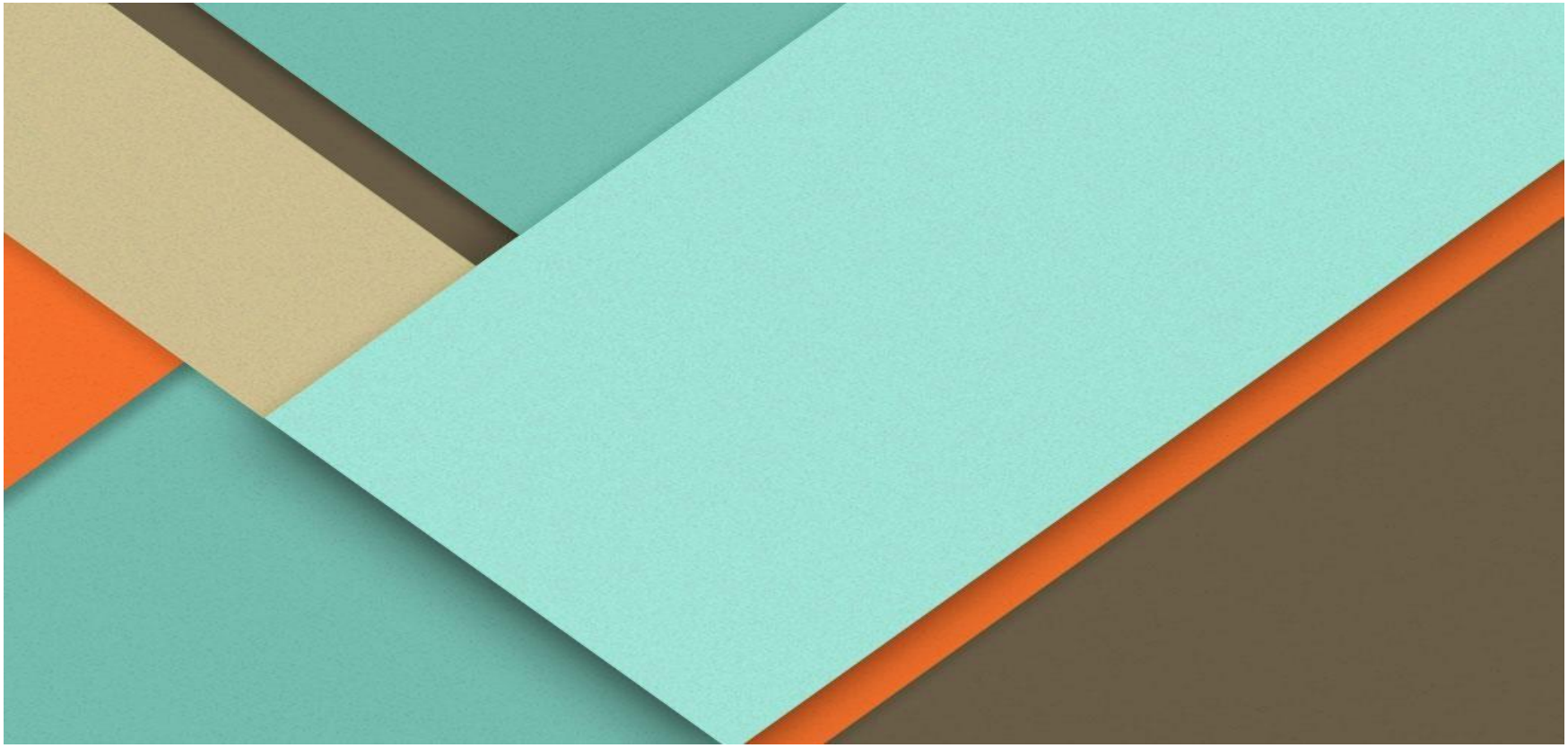


# Manejo de Vistas y Funciones - SQL Server

Procesual **Hito 4**

Base de Datos I - 2021

Nombre Christian  Miguel Delgado Mamani

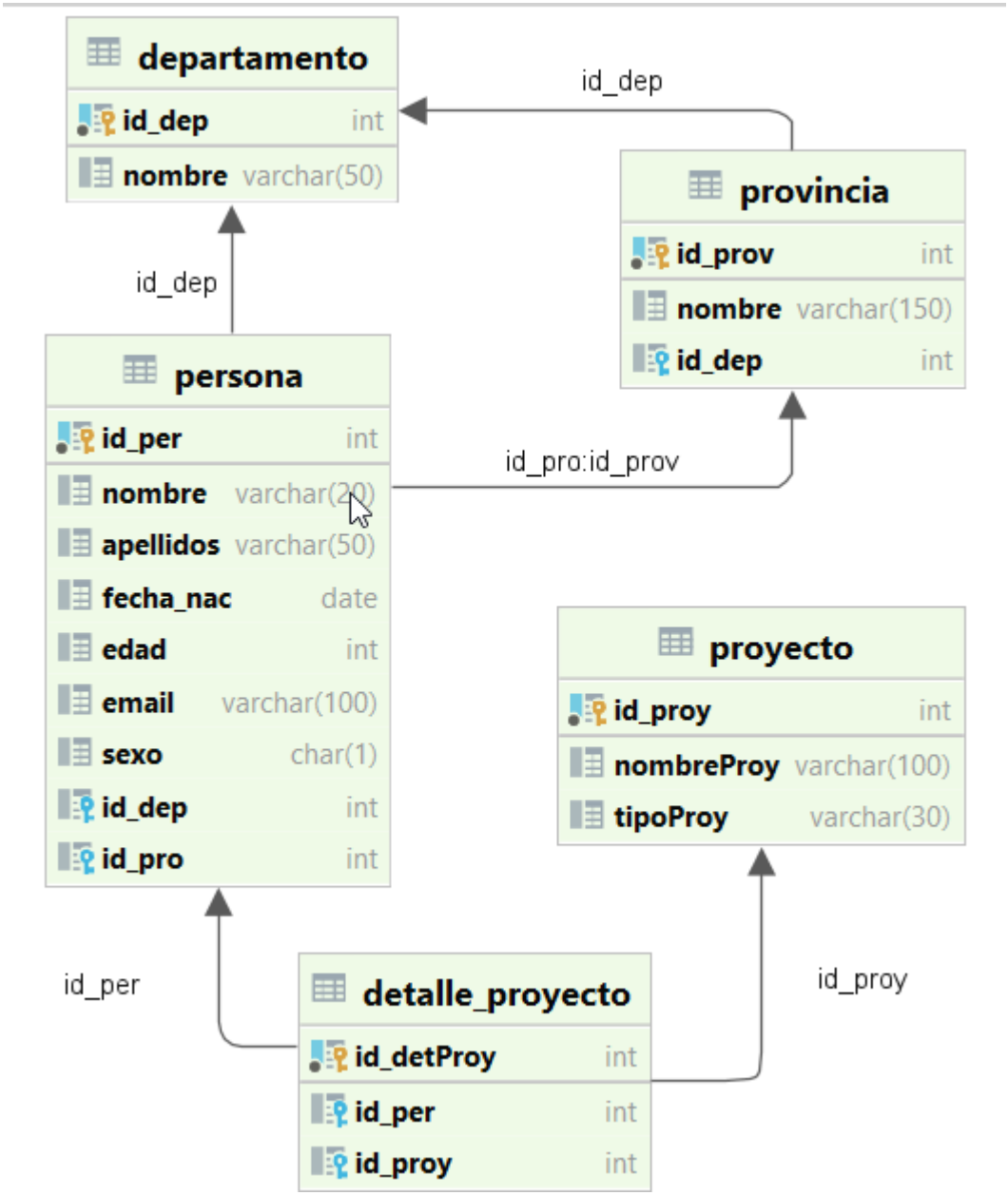
# Consigna

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 organización sin fines de lucro **ONG** requiere de un sistema de información para poder gestionar proyectos manejados por ellos.

**Detalles puntuales y análisis del problema**

| **ONG la muralla verde** | |
|---|---|
| **Diseño** | En función al escenario se identificó que las posibles entidades son **departamento, provincia, persona, proyecto y detalle proyecto**, pues un cliente vive en cierta localidad y esta persona trabaja en un proyecto específico.<br><br>El objetivo es poder tener un control de todos los proyectos, por ejemplo quiero saber donde vive una persona y en qué proyecto participa.<br><br>En la primera fase solo debe resolver este requerimiento inicial manejando **Vistas** y **Funciones.**<br><br>• Determinar donde vive una persona y en qué proyecto participa. |


departamento
id_dep int
nombre varchar(50)
id_dep
provincia
id_prov int
nombre varchar(150)
id_dep int
id_dep
persona
id_per int
nombre varchar(20)
apellidos varchar(50)
fecha_nac date
edad int
email varchar(100)
sexo char(1)
id_dep int
id_pro int
id_pro:id_prov
proyecto
id_proy int
nombreProy varchar(100)
tipoProy varchar(30)
id_per
detalle_proyecto
id_detProy int
id_per int
id_proy int
id_proy

# 1. Diseño de base de datos.

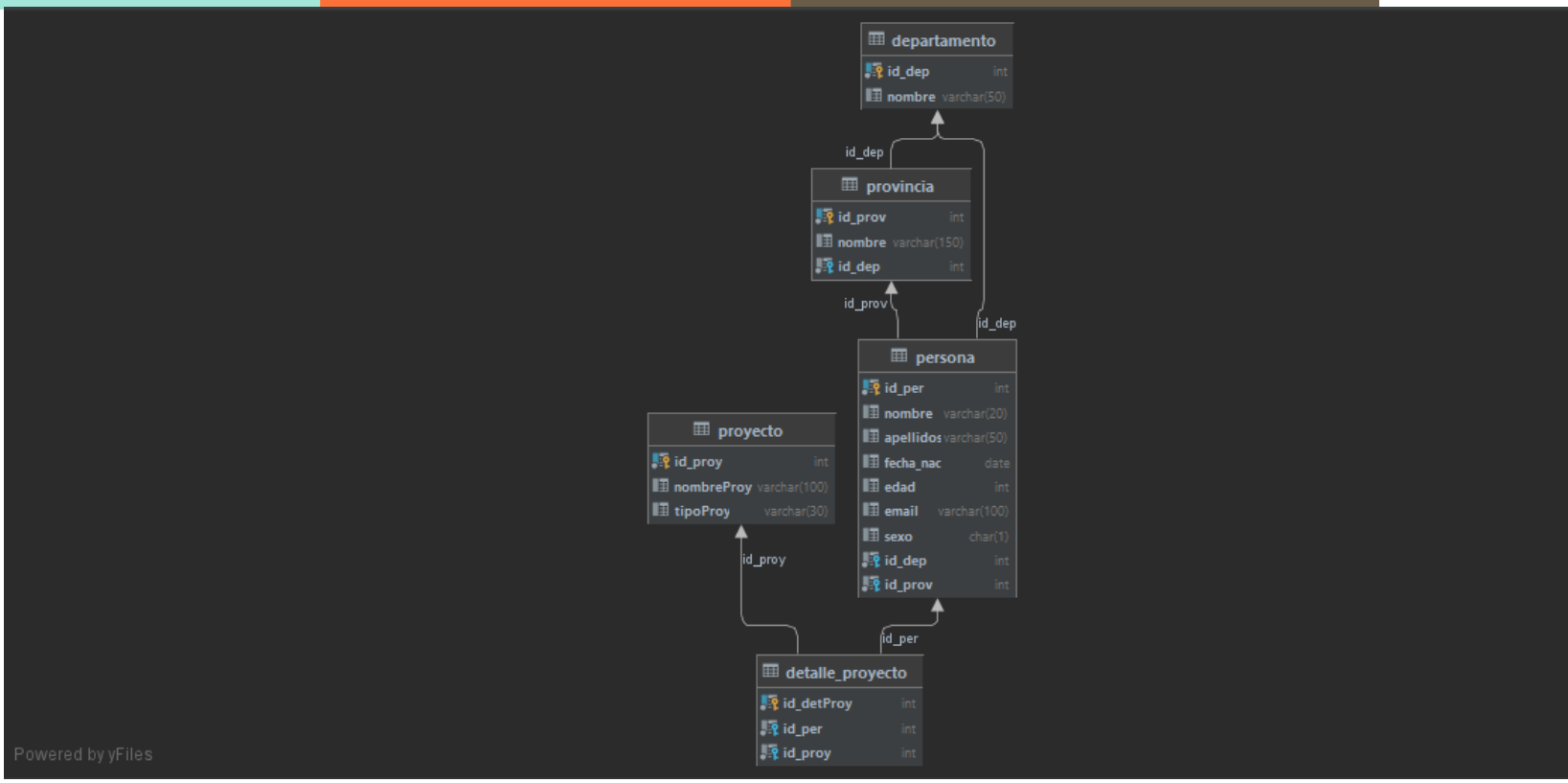
1.1. Adjuntar el código SQL que genera la **base de datos, tablas y los registros** correspondientes.

2.

```
create database ONG
use  ONG
create table departamento(
    id_dep integer identity (1,1) primary key,
    nombre varchar(50)
);

create table provincia(
    id_prov integer identity (1,1) primary key,
    nombre varchar(150),
      id_dep integer not null,
      FOREIGN KEY (id_dep) REFERENCES departamento(id_dep)
);

create table proyecto(
    id_proy integer identity (1,1) primary key,
    nombreProy varchar(100),
    tipoProy varchar(30)
);

create table persona(
    id_per integer identity (1,1) primary key,
    nombre varchar(20),
      apellidos varchar(50),
      fecha_nac date,
      edad integer,
      email varchar(100),
      sexo char(1),
      id_dep integer not null,
      id_prov integer not null,
      FOREIGN KEY (id_dep) REFERENCES departamento(id_dep),
      FOREIGN KEY (id_prov) REFERENCES provincia(id_prov),
);

create table detalle_proyecto(
    id_detProy integer identity (1,1) primary key,
      id_per integer not null,
      id_proy integer not null,
```

```
        FOREIGN KEY (id_per) REFERENCES persona(id_per),
        FOREIGN KEY (id_proy) REFERENCES proyecto(id_proy),
);
INSERT INTO departamento (nombre) VALUES ('Cochabamba');
INSERT INTO departamento (nombre) VALUES ('La Paz');
INSERT INTO departamento (nombre) VALUES ('Santa Cruz');
INSERT INTO departamento (nombre) VALUES ('Beni');
INSERT INTO departamento (nombre) VALUES ('Pando');

INSERT INTO provincia (nombre, id_dep) VALUES ('Quillacollo', 1);
INSERT INTO provincia (nombre, id_dep) VALUES ('Sacaba', 1);
INSERT INTO provincia (nombre, id_dep) VALUES ('Mizque', 1);
INSERT INTO provincia (nombre, id_dep) VALUES ('Murillo', 2);
INSERT INTO provincia (nombre, id_dep) VALUES ('Robore', 3);

INSERT INTO proyecto (nombreProy, tipoProy) VALUES ('Sembrado de Arboles', 'FORESTACION');
INSERT INTO proyecto (nombreProy, tipoProy) VALUES ('Estudio de Semillas', 'FORESTACION');
INSERT INTO proyecto (nombreProy, tipoProy) VALUES ('Alfabetizacion', 'EDUCACION');
INSERT INTO proyecto (nombreProy, tipoProy) VALUES ('Creacion de Escuelas', 'EDUCACION');
INSERT INTO proyecto (nombreProy, tipoProy) VALUES ('Apoyo al dibujo', 'ARQUITECTURA');

INSERT INTO persona (nombre, apellidos, fecha_nac, edad, email, sexo, id_dep, id_prov) VALUES
                    ('nombre1', 'apellidos1', '1990-10-30', 28, 'nombre1@gmail.com', 'm', 1, 1),
                    ('nombre2', 'apellidos2', '1992-10-30', 28, 'nombre2@gmail.com', 'm', 1, 2),
                    ('nombre3', 'apellidos3', '1994-10-30', 26, 'nombre3@gmail.com', 'm', 1, 3),
                    ('nombre4', 'apellidos4', '1996-10-30', 24, 'nombre4@gmail.com', 'm', 2, 4),
                    ('nombre5', 'apellidos5', '1992-10-30', 28, 'nombre5@gmail.com', 'm', 3, 5),
                    ('nombre6', 'apellidos6', '1999-10-30', 19, 'nombre6@gmail.com', 'f', 3, 5);

INSERT INTO detalle_proyecto (id_per, id_proy) VALUES
                             (1, 1),
                             (2, 1),
                             (3, 1),
                             (3, 2),
                             (4, 3),
                             (4, 4),
                             (5, 5),
                             (6, 5);
```

2.1. Después de generar el **modelo lógico** de la base de datos.




## 2 Manejo de Vistas

2.1Mostrar a todas las **personas** que viven en el departamento de **Cochabamba**.

```
select *
from persona as p
LEFT JOIN departamento dep on dep.id_dep = p.id_dep
where p.id_dep = 1
```

Output | Result 6

3 rows | CSV

| | id_per | p.nombre | apellidos | fecha_nac | edad | email | sexo | p.id_dep | id_prov | dep.id_dep | dep.nombr |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | nombre1 | apellidos1 | 1990-10-30 | 28 | nombre1@gmail.com | m | 1 | 1 | 1 | Cochabamba |
| 2 | 2 | nombre2 | apellidos2 | 1992-10-30 | 28 | nombre2@gmail.com | m | 1 | 2 | 1 | Cochabamba |
| 3 | 3 | nombre3 | apellidos3 | 1994-10-30 | 26 | nombre3@gmail.com | m | 1 | 3 | 1 | Cochabamba |

2.2. Mostrar la persona (**nombres y apellidos**) y el **nombre del proyecto** en donde trabajan.

```
select CONCAT(p.nombre,' ',p.apellidos) as nombres, pro.nombreProy as proyecto
from detalle_proyecto as dp
INNER JOIN persona p on p.id_per = dp.id_per
INNER JOIN proyecto pro on pro.id_proy = dp.id_proy
```

Output | Result 7

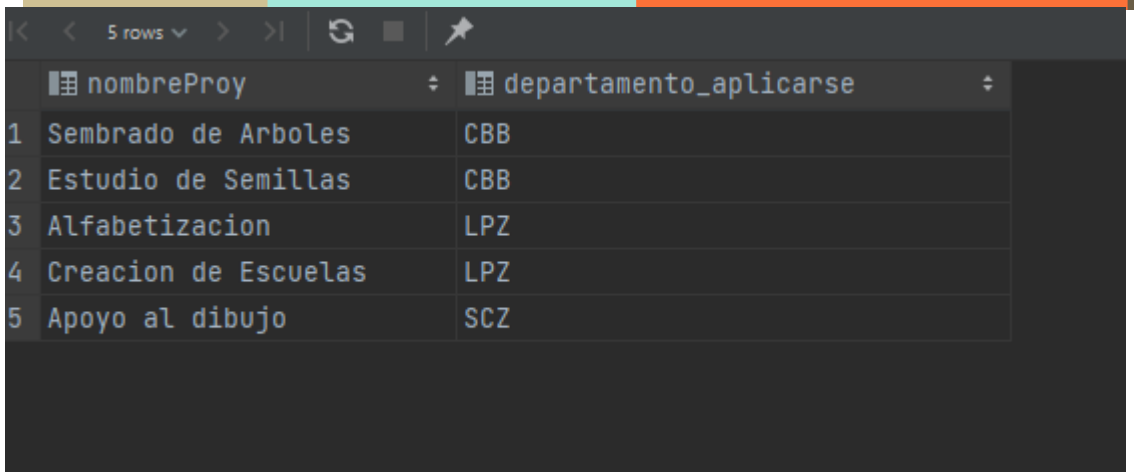
8 rows

| | nombres | proyecto |
|---|---|---|
| 1 | nombre1 apellidos1 | Sembrado de Arboles |
| 2 | nombre2 apellidos2 | Sembrado de Arboles |
| 3 | nombre3 apellidos3 | Sembrado de Arboles |
| 4 | nombre3 apellidos3 | Estudio de Semillas |
| 5 | nombre4 apellidos4 | Alfabetizacion |
| 6 | nombre4 apellidos4 | Creacion de Escuelas |
| 7 | nombre5 apellidos5 | Apoyo al dibujo |
| 8 | nombre6 apellidos6 | Apoyo al dibujo |

2.3. Asumir que tiene 3 tipos de proyectos (TIPO_A, TIPO_B, TIPO_C)

- El objetivo es crear una Vista con los mismos datos de la tabla proyectos
- Sin embargo generar una nueva columna en la vista de nombre **departamento_aplicarse**
- Si el tipo de proyecto es de **TIPO_A** asignar **CBB**
- Si el tipo de proyecto es de **TIPO_B** asignar **LPZ**
- Si el tipo de proyecto es de **TIPO_C** asignar **SCZ**
- Si el tipo de proyecto es **otro tipo** asignar '**En proceso de análisis**'

```
select nombreProy,
CASE
   WHEN tipoProy = 'FORESTACION' THEN 'CBB'
   WHEN tipoProy = 'EDUCACION' THEN 'LPZ'
   WHEN tipoProy = 'ARQUITECTURA' THEN 'SCZ'
   ELSE 'EN PROCESO DE ANÁLISIS'
END as departamento_aplicarse
from proyecto
```



5 rows

| | nombreProy | departamento_aplicarse |
|---|---|---|
| 1 | Sembrado de Arboles | CBB |
| 2 | Estudio de Semillas | CBB |
| 3 | Alfabetizacion | LPZ |
| 4 | Creacion de Escuelas | LPZ |
| 5 | Apoyo al dibujo | SCZ |

2.4. Crear una vista cualquiera que muestre 5 columnas.

```
CREATE VIEW vista_persona
 AS SELECT nombre, apellidos, fecha_nac, edad, email
 FROM persona
```

WHERE ORDER BY

| | nombre | apellidos | fecha_nac | edad | email |
|---|---|---|---|---|---|
| 1 | nombre1 | apellidos1 | 1990-10-30 | 28 | nombre1@gmail.com |
| 2 | nombre2 | apellidos2 | 1992-10-30 | 28 | nombre2@gmail.com |
| 3 | nombre3 | apellidos3 | 1994-10-30 | 26 | nombre3@gmail.com |
| 4 | nombre4 | apellidos4 | 1996-10-30 | 24 | nombre4@gmail.com |
| 5 | nombre5 | apellidos5 | 1992-10-30 | 28 | nombre5@gmail.com |
| 6 | nombre6 | apellidos6 | 1999-10-30 | 19 | nombre6@gmail.com |

# 3. Manejo de Funciones

3.1. Crear una **función** que permita saber cuántos proyectos distintos del **TIPO_A**, **TIPO_B** y **TIPO_C** existen

- La función no recibe ningún parámetro

```
CREATE function contarProyectos()
RETURNS VARCHAR(100)
AS
    BEGIN
         DECLARE @res VARCHAR(100)='';

         SELECT @res = COUNT(1)
         FROM proyecto
         WHERE tipoProy != 'FORESTACION' AND
         tipoProy != 'EDUCACION' AND
         tipoProy != 'ARQUITECTURA'
         GROUP BY tipoProy

         RETURN @res;
    end;
```

```sql
CREATE function contarProyectos()
RETURNS VARCHAR(100)
AS
    BEGIN
            DECLARE @res VARCHAR(100)='';

            SELECT @res = COUNT(1)
            FROM proyecto
            WHERE tipoProy != 'FORESTACION' AND
            tipoProy != 'EDUCACION' AND
            tipoProy != 'ARQUITECTURA'
            GROUP BY tipoProy

            RETURN @res;
    end;

SELECT dbo.contarProyectos() as cantidad_dif;
```

contarProyectos()

Output | cantidad_dif:varchar(100)

1 row

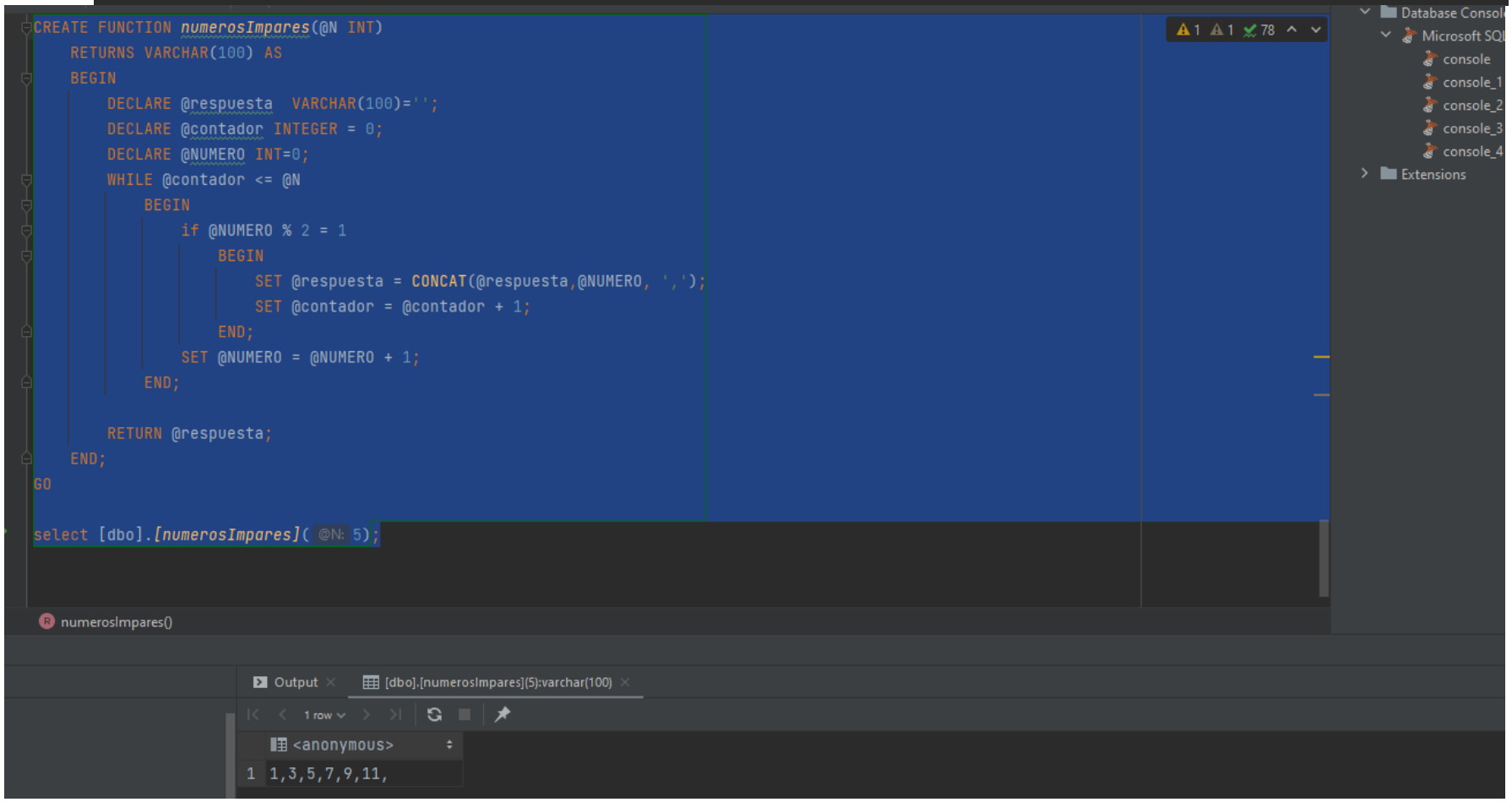
| | cantidad_dif |
|---|---|
| 1 | 0 |

3.2. Crear una **función** que genere los primeros N números impares.

- La función recibe solo un parámetro (el valor N)
- Si n es 5 la salida debe ser: 1, 3, 5, 7, 9,
- Si n es 2 la salida debe ser: 1, 3,
- Si n es 4 la salida debe ser: 1, 3, 5, 7

```sql
CREATE FUNCTION numerosImpares(@N INT)
    RETURNS VARCHAR(100) AS
    BEGIN
        DECLARE @respuesta  VARCHAR(100)='';
        DECLARE @contador INTEGER = 0;
        DECLARE @NUMERO INT=0;
        WHILE @contador <= @N
            BEGIN
                if @NUMERO % 2 = 1
```

```
                    BEGIN
                        SET @respuesta = CONCAT(@respuesta,@NUMERO, ',');
                        SET @contador = @contador + 1;
                    END;
                SET @NUMERO = @NUMERO + 1;
            END;

        RETURN @respuesta;
    END;
GO

select [dbo].[numerosImpares](5);
```



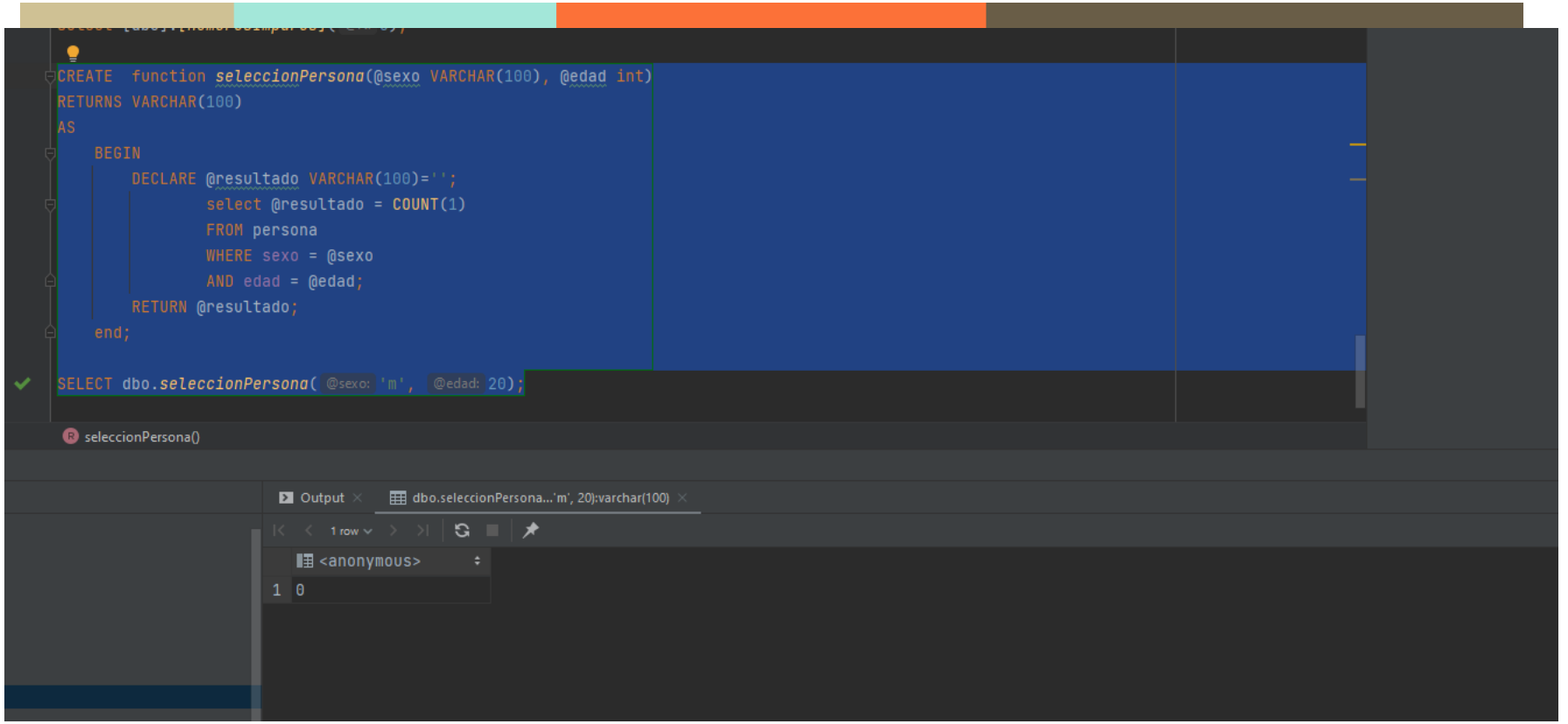
3.3. Crear una **función** que **permita insertar** un registro a la tabla **persona**.

- La función recibe los datos a insertarse a la tabla persona
- La función retorna un mensaje indicando que **se insertó satisfactoriamente el registro**.

**3.4.** Crear una **función** cualquiera**.**

- La función debe de recibir **2 parámetros.**
- Usar los parámetros en la lógica de la función.

```
CREATE  function seleccionPersona(@sexo VARCHAR(100), @edad int)
RETURNS VARCHAR(100)
AS
    BEGIN
        DECLARE @resultado VARCHAR(100)='';
            select @resultado = COUNT(1)
            FROM persona
            WHERE sexo = @sexo
            AND edad = @edad;
        RETURN @resultado;
    end;

SELECT dbo.seleccionPersona('m', 20);
```




## Detalles sobre la entrega de la tarea

1. Crear un documento(archivo PPT) por cada pregunta adjuntar una respuesta(texto explicativo) y una captura de pantalla(Screenshot). Es decir debe crear una presentación power point.
2. El documento generado en el paso 1 convertir en un archivo PDF.
3. Crear un único vídeo explicando todos los pasos realizados para resolver esta actividad.
4. El documento PDF y el video tienen que estar en una carpeta de nombre **procesual** en github (**hito4/procesua**l)
5. En la plataforma Moodle solo subir la carátula.
    a. En la parte inferior agregar **URL de github:** {{mi_url_de_github}}