

Defensa 'Hito 2' {

[Materia: Base de Datos II]

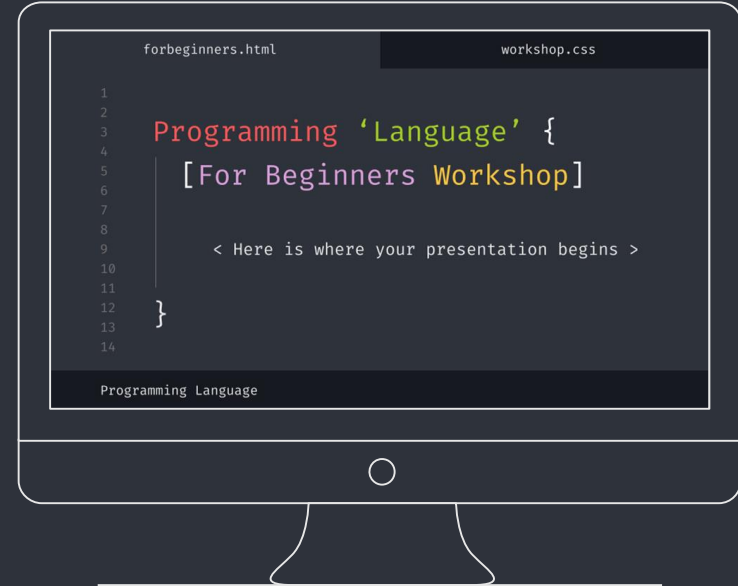
< Docente: William Roddy Barra Paredes >

}

# Primera Parte {

La primera parte  
corresponde a la parte  
TEÓRICA necesaria, en  
donde se encuentra un  
conglomerado de preguntas  
relacionadas a BASES DE  
DATOS RELACIONALES

}



# 01 {

[¿A que se refiere cuando se habla de una Base de Datos Relacionales?]

< Una base de datos relacional es una base de datos que almacena y proporciona acceso a puntos de datos que están relacionados entre sí. Las bases de datos relacionales se basan en el modelo relacional, que es una forma intuitiva y sencilla de representar datos en tablas.>

}

01



¿Qué son las  
bases de  
datos  
relacionales?



## 02 {

[¿A que se refiere cuando se habla de una Base de Datos No Relacionales?]

< Las bases de datos NoSQL están diseñadas para una variedad de patrones de acceso a datos, incluidas las aplicaciones de baja latencia. Las bases de datos de búsqueda NoSQL están diseñadas para el análisis de datos semiestructurados. El modelo relacional normaliza los datos en una tabla que consta de filas y columnas.

>

}

02

}



}



## 03 {



[¿Qué es MySQL y MariaDB? Explique si existen diferencias o son iguales, etc.]

< MySQL es un sistema de gestión de bases de datos relacionales (RDBMS) de código abierto con tecnología de Oracle y basado en el lenguaje de consulta estructurado (SQL). MySQL se ejecuta en casi todas las plataformas, incluidas Linux, UNIX y Windows.>

}

## 03 {



< MariaDB es un sistema de gestión de base de datos.

Además, permite a los desarrolladores y diseñadores realizar cambios en un sitio web cambiando solo un archivo (sin modificar todo el código web) para ejecutarlos en toda la estructura de datos compartida en la red.

}>



## 03 {

<Aunque MariaDB es una bifurcación de MySQL, los dos sistemas de administración de bases de datos siguen siendo muy diferentes: MariaDB tiene licencia GPL, mientras que MySQL tiene un enfoque de licencia dual. Cada mango se apila de forma diferente. MariaDB admite diferentes tipos de motores de almacenamiento.>

}

## MOTORES DE BASES DE DATOS RELACIONALES

## 1. ORACLE

ORACLE

El motor relacional comercial más antiguo.

Su creador, Larry Ellison, estuvo en el comité que definió SQL.



## 2. MICROSOFT SQL SERVER

Microsoft SQL Server

Multiplataforma desde 2017. Son líderes en Business Intelligence (integrando más apps en el mismo paquete).

## 3. MYSQL

MySQL

Es el motor más usado en la web (y preferido por los CMS clásicos que usan PHP como WordPress, Drupal, Magento, etc.).



## 4. SQLITE

Es una base de datos embebida en el programa. Al estar integrado en todos los teléfonos se usa para almacenamiento interno de apps

## 5. MARIADB

MariaDB

Fork derivado de MySQL a partir de su compra por Oracle. Compatible con MySQL para poder cambiar un motor por otro.



## 6. POSTGRESQL

Inició como un proyecto universitario llamado INGRES, inspirado en Oracle. Usan funciones y triggers que MySQL no tuvo por años.

## 04 {

[¿Qué son las funciones de agregación?]

<Son aquellos que utilizan una cláusula SELECT, se aplican a un conjunto de registros y devuelven un solo valor.>

}

04

{

}

Funciones de Agregado	
Función	Descripción
<b>AVG</b>	Utilizada para calcular el promedio de los valores de un campo determinado
<b>COUNT</b>	Utilizada para devolver el número de registros de la selección
<b>SUM</b>	Utilizada para devolver la suma de todos los valores de un campo determinado
<b>MAX</b>	Utilizada para devolver el valor más alto de un campo especificado
<b>MIN</b>	Utilizada para devolver el valor más bajo de un campo especificado

## 05 {

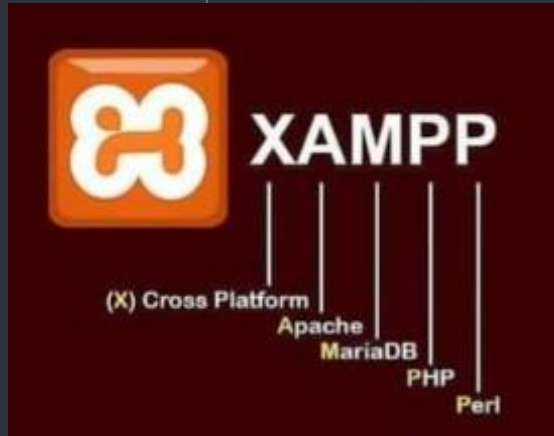
[¿Qué llegaría a ser XAMPP?]

< XAMPP es una distribución de Apache con HTTPS web gratuitos. El nombre está compuesto por las siglas de los programas que lo componen: el servidor web Apache, los sistemas de gestión de bases de datos relacionales MySQL y MariaDB, y los lenguajes de programación Perl y PHP.>

}

05

{



}



## 06 {

[¿Cuál es la diferencia entre las funciones de agregación y funciones creados por el DBA? Es decir funciones creadas por el usuario.]

<Las funciones de agregación ya se encuentran predeterminada y se la ejecuta con la cláusula SELECT.

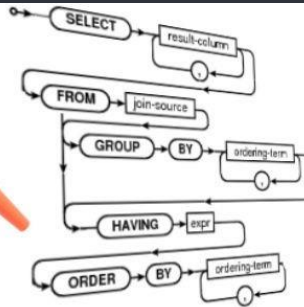
Las funciones creadas por el DBA se utilizan para realizar tareas o ejecutas datos específicos; y se ejecuta con la cláusula SELECT y WHERE.>

}

06

{

**SQL**  
Funciones  
de agregación



}

## Ejemplo de una función escalar definida por el usuario

### ■ Creación de la función

```

USE Neptuneo2013
GO
CREATE FUNCTION fn_num_pedidos_cliente
    (@idcliente char(5))
RETURNS int
AS
BEGIN
    RETURN
        (SELECT Count(idpedido) FROM Pedidos
         WHERE idcliente = @idcliente )
END
GO
    
```

### ■ Llamada a la función

```

SELECT dbo. fn_num_pedidos_cliente ('ALFKI')
    
```



07 {

[¿Para qué sirve el comando USE? ]

<El comando USE se utiliza para  
posicionarse en la base de datos a  
utilizar.>

}





# 07 {

[¿Para qué sirve el comando USE? ]

<El comando USE se utiliza para  
posicionarse en la base de datos a  
utilizar.>

}



07 {

[¿Para qué sirve el comando USE? ]

<El comando USE se utiliza para  
posicionarse en la base de datos a  
utilizar.>

}

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14

Parte {  
Practica;  
}  
}

## Ejercicio < /1 > {



<Crear las tablas y 2 registros para cada tabla para el siguiente modelo ER.>

}

## Ejercicio < /2 > {



<Crear una consulta SQL en base al ejercicio anterior.>

}

## Ejercicio < /3 > {



<Crear un función que compare dos códigos de materia.>

}

## Ejercicio < /4 > {



<Crear una función que permita obtener el promedio de las edades del género masculino o femenino de los estudiantes inscritos en la asignatura ARQ-104. >

}

## Ejercicio < /5 > {



<Crear una función que permita concatenar 3 cadenas.>

}

## Ejercicio < /6 > {



<Crear una función de acuerdo a lo siguiente:  
Mostrar el nombre, apellidos y el semestre de todos los estudiantes que estén inscritos. Siempre y cuando la suma de las edades del sexo femenino o masculino sea par y mayores a cierta edad. >

}

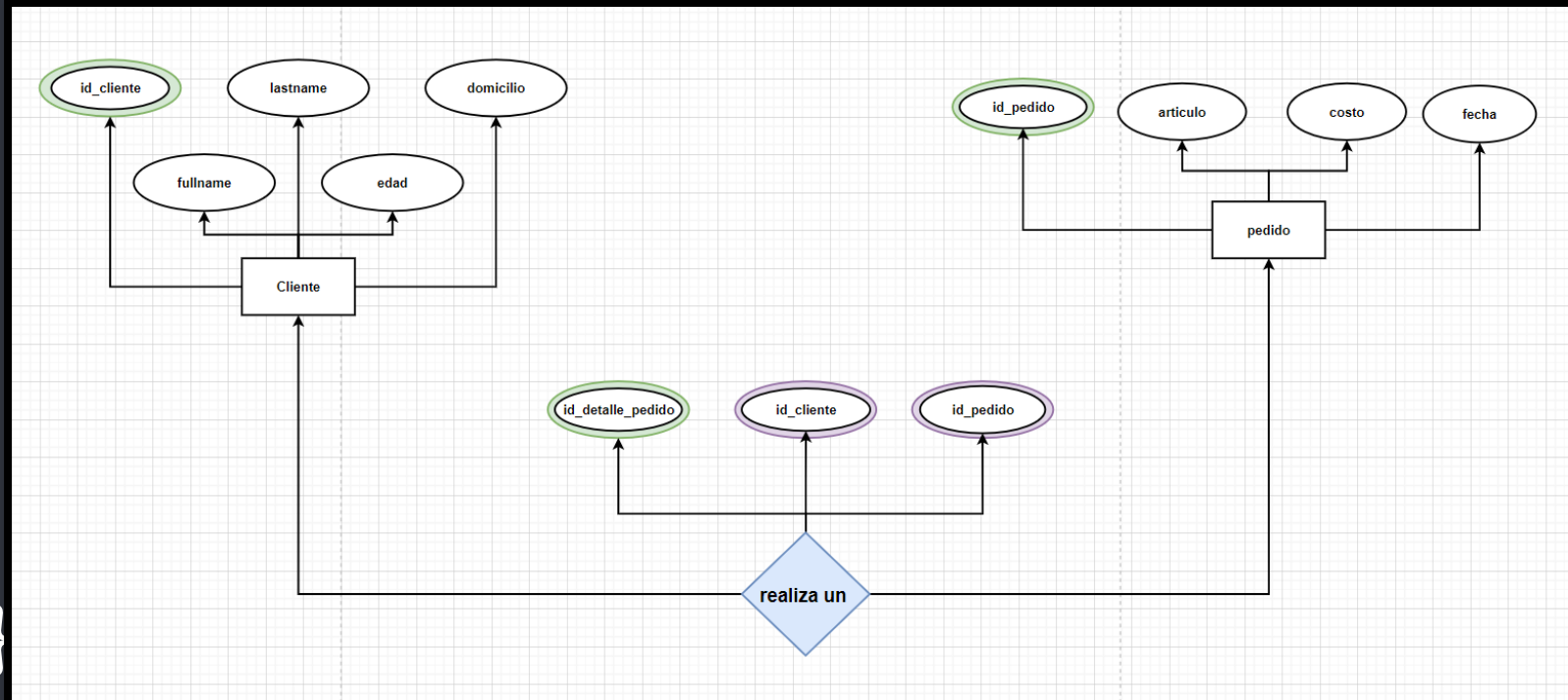
## Ejercicio < /7 > {



<Crear una función de acuerdo a lo siguiente: o  
Crear una función sobre la tabla estudiantes que  
compara un nombre y apellidos. (si existe este  
nombre y apellido mostrar todos los datos del  
estudiante).>

}

# Ejercicio 1; { 'Diseño Entidad Relacion'

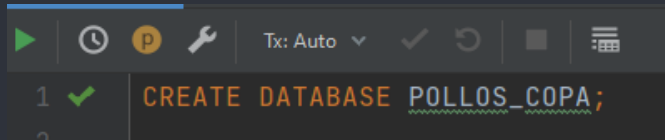




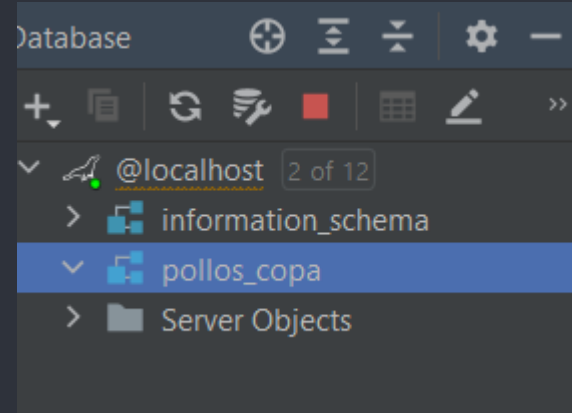
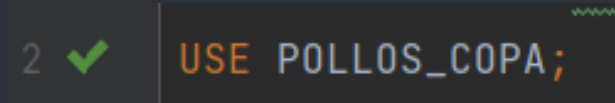
# Ejercicio 1 {

Creando la Base de Datos

MySQL  60%



Posicionandose en la Base de Datos



# Ejercicio 1 {

Creando la Tabla Cliente

```
CREATE TABLE cliente
(
    id_cliente INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
    fullname VARCHAR(100) NOT NULL,
    lastname VARCHAR(100) NOT NULL,
    edad INT NOT NULL,
    domicilio VARCHAR (200) NOT NULL
);
```

}

# Ejercicio 1 {

Creando la Tabla pedido

```
CREATE TABLE pedido
(
    id_pedido INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
    articulo VARCHAR (90) NOT NULL,
    costo INT NOT NULL,
    fecha DATE NOT NULL
);
```

}

# Ejercicio 1 {

Creando la Tabla detalle\_pedido

```
CREATE TABLE detalle_pedido
(
    id_detalle_pedido INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
    id_cliente INT NOT NULL,
    FOREIGN KEY (id_cliente) REFERENCES cliente (id_cliente),
    id_pedido INT NOT NULL,
    FOREIGN KEY (id_pedido) REFERENCES pedido (id_pedido)
);
```

## Ejercicio 1 {

Ingresando registros la Tabla  
Cliente

```
INSERT INTO cliente (id_cliente, fullname, lastname, edad, domicilio)
VALUES (9238756, 'Sofia', 'Gonzales', 18, 'Av. Chacaltaya #489'),
      (7028456, 'David', 'Rojas', 21, 'Z. 16 de Julio, C. Nelly #12'),
      (1234556, 'Mickey', 'Torrez Garza', 19, 'Z. Villa Adela'),
      (457878, 'Tomas', 'Vasquez', 25, 'Rio Seco'),
      (14579312, 'Ana Luisa', 'Mendoza', 17, 'Domicilio123' );
```

}

# Ejercicio 1 {

Ingresando registros la Tabla  
pedido

```
INSERT INTO pedido (articulo, costo, fecha)
VALUES ('Balde de 8 presas', 120, '2022/02/07'),
       ('1 pollo individual', 25, '2022/05/17'),
       ('2 pollos pequeños , mas dos refrescos', 36, '2022/04/23'),
       ('Balde de 16 presas, 2 arroz, 1 refresco 2 lts.', 180, '2022/01/12'),
       ('3 Combo Fiesta ', 90, '2022/03/15');
```

}

# Ejercicio 1 {

Ingresando registros la Tabla  
detalle\_pedido

```
INSERT INTO detalle_pedido (id_cliente, id_pedido)
VALUES (9238756, 1),
       (7028456, 2),
       (1234556, 3),
       (457878, 4),
       (14579312, 5);
```

}

## Ejercicio 2 {

- o Debe de utilizar las 3 tablas creadas anteriormente.
- o Para relacionar las tablas utilizar JOINS.

```
✓ SELECT depd.id_detalle_pedido, cli.fullname, cli.lastname, cli.domicilio, ped.costo
FROM cliente AS cli
    INNER JOIN detalle_pedido AS depd ON cli.id_cliente = depd.id_cliente
    INNER JOIN pedido AS ped ON depd.id_pedido = ped.id_pedido
WHERE ped.costo >=90;
```

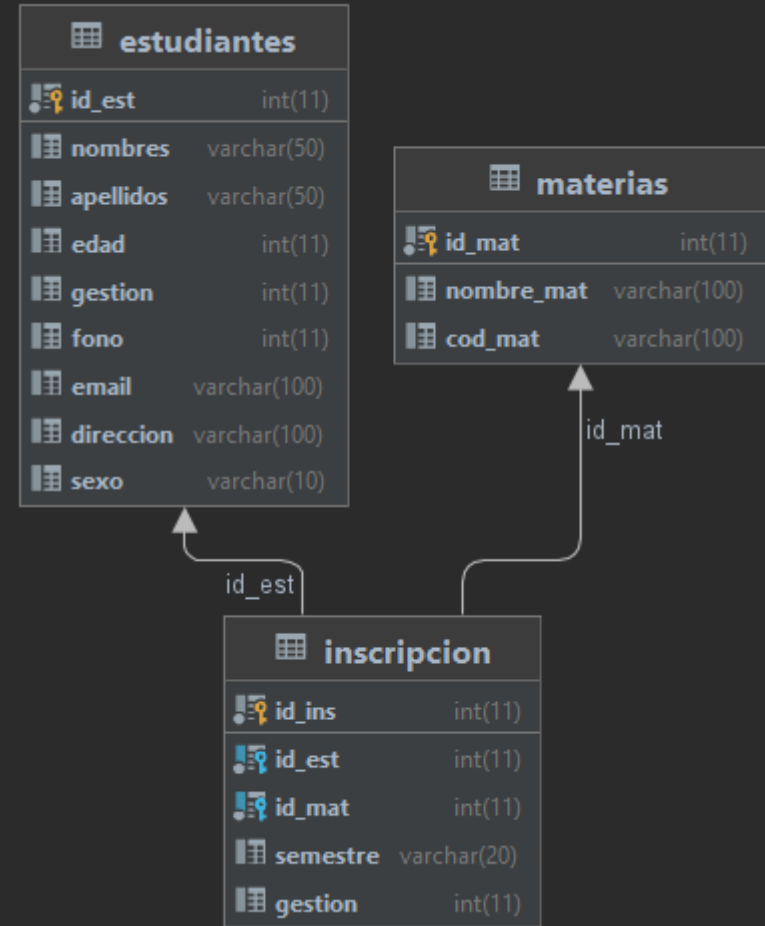
	id_detalle_pedido	fullname	lastname	domicilio	costo
1	1	Sofia	Gonzales	Av. Chacaltaya #489	120
2	4	Tomas	Vasquez	Rio Seco	180
3	5	Ana Luisa	Mendoza	Domicilio123	90



# Ejercicio 3 {

Recrear la  
siguiente base de  
datos:

}



# Ejercicio 3 {

Creando la Base de Datos

MySQL  60%

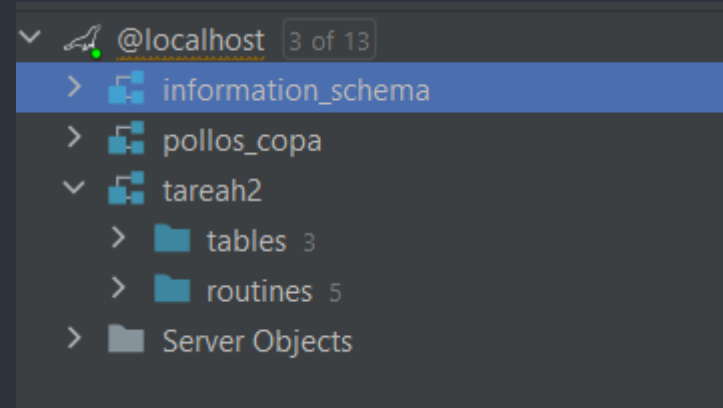
```
CREATE DATABASE TareaH2;
```

Posicionandose en la Base de Datos



```
USE TareaH2;
```

}



# Ejercicio 3 {

Creando la Tabla estudiantes

```
CREATE TABLE estudiantes
(
    id_est INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
    nombres VARCHAR(50),
    apellidos VARCHAR(50),
    edad INTEGER,
    gestion INTEGER,
    fono INTEGER,
    email VARCHAR(100),
    direccion VARCHAR(100),
    sexo VARCHAR(10)
);
```

# Ejercicio 3 {

Creando la Tabla materias

```
CREATE TABLE materias
(
    id_mat INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
    nombre_mat VARCHAR(100),
    cod_mat VARCHAR(100)
);
```

}

## Ejercicio 3 {

Creando la Tabla inscripcion

```
CREATE TABLE inscripcion
(
    id_ins INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
    id_est INT NOT NULL,
    FOREIGN KEY (id_est) REFERENCES estudiantes (id_est),
    id_mat INT NOT NULL,
    FOREIGN KEY (id_mat) REFERENCES materias (id_mat),
    semestre VARCHAR(20),
    gestion INTEGER
);
```

# Ejercicio 3 {

Registros en la Tabla estudiantes

```
INSERT INTO estudiantes (nombres, apellidos, edad, fono, email, direccion, sexo)
VALUES ('Miguel' , 'Gonzales Veliz', 20, 2832115, 'miguel@gmail.com', 'Av. 6 de Agosto', 'masculino'),
       ('Sandra' , 'Mavir Uria', 25, 2832116, 'sandra@gmail.com', 'Av. 6 de Agosto', 'femenino'),
       ('Joel' , 'Adubiri Mondar', 30, 2832117, 'joel@gmail.com', 'Av. 6 de Agosto', 'masculino'),
       ('Andrea' , 'Arias Ballesteros', 21, 2832118, 'andrea@gmail.com', 'Av. 6 de Agosto', 'femenino'),
       ('Santos' , 'Montes Valenzuela', 24, 2832119, 'santos@gmail.com', 'Av. 6 de Agosto', 'masculino');
```

}

## Ejercicio 3 {

Registro en la Tabla materias

```
INSERT INTO materias (nombre_mat, cod_mat)
VALUES ('Introduccion a la Arquitectura', 'ARQ-101'),
       ('Urbanismo y Diseno', 'ARQ-102'),
       ('Dibujo y Pintura Arquitectonico', 'ARQ-103'),
       ('Matematica discreta', 'ARQ-104'),
       ('Fisica Basica', 'ARQ-105');
```

}

## Ejercicio 3 {

Registrs en la Tabla inscripcion

```
INSERT INTO inscripcion (id_est, id_mat, semestre, gestion)
VALUES (1, 1, '1er Semestre', 2015),
       (1, 2, '2do Semestre', 2015),
       (2, 4, '1er Semestre', 2016),
       (2, 3, '2do Semestre', 2016),
       (3, 3, '2do Semestre', 2017),
       (3, 1, '3er Semestre', 2017),
       (4, 4, '4to Semestre', 2017),
       (5, 5, '5to Semestre', 2017);
```

}



## Ejercicio 3 {

Mostrar los nombres y apellidos de los estudiantes inscritos en la materia ARQ-105, adicionalmente mostrar el nombre de la materia. Deberá de crear una función que reciba dos parámetros y esta función deberá ser utilizada en la cláusula WHERE.

```
CREATE FUNCTION compara (cod VARCHAR(60), mat_com VARCHAR(50))  
  RETURNS INT  
  BEGIN  
    RETURN cod = mat_com;  
  END;  
  
SELECT est.id_est, est.nombres, est.apellidos, mat.nombre_mat  
FROM inscripcion AS ins  
  INNER JOIN estudiantes AS est on ins.id_est = est.id_est  
  INNER JOIN materias AS mat on ins.id_mat = mat.id_mat  
WHERE compara ( cod: mat.cod_mat, mat_com: 'ARQ-105');
```

1 row				
	id_est	nombres	apellidos	nombre_mat
1	5	Santos	Montes Valenzuela	Fisica Basica

## Ejercicio 5 { Primera forma

```
1  
2  
3 CREATE FUNCTION concatenar (num1 VARCHAR(50), num2 VARCHAR(50), num3 VARCHAR(50))  
4 RETURNS TEXT  
5 BEGIN  
6     DECLARE n TEXT;  
7     SELECT CONCAT(num1, ' - ', num2, ' - ', num3) INTO n;  
8     RETURN (n);  
9 end;
```

```
10 SELECT concatenar( num1: 'Pepito', num2: 'Perez', num3: 50) AS tabla1;
```

```
11  
12  
13  
14 }
```

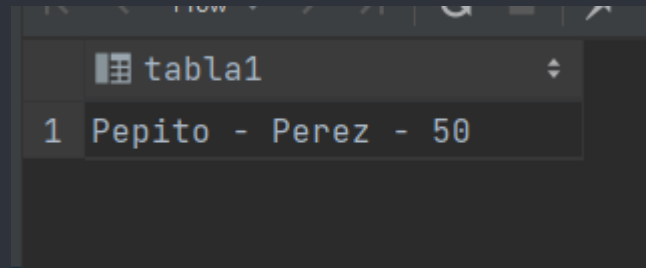


	tabla1
1	Pepito - Perez - 50

# Ejercicio 5 { Segunda forma

```
1 CREATE FUNCTION concatenar (num1 VARCHAR(50), num2 VARCHAR(50), num3 INT)
2 RETURNS TEXT
3 BEGIN
4     DECLARE n TEXT;
5     SELECT CONCAT(num1, ' - ', num2, ' - ', num3) INTO n;
6     RETURN (n);
7 end;
8 SELECT concatenar( num1: est.nombres, num2: est.apellidos, num3: est.edad) AS tabla1
9 FROM estudiantes AS est;
```

tabla1	
1	Miguel - Gonzales Veliz - 20
2	Sandra - Mavir Uria - 25
3	Joel - Adubiri Mondar - 30
4	Andrea - Arias Ballesteros - 21
5	Santos - Montes Valenzuela - 24

## Ejercicio 6 {

Debe de crear una función que sume las edades (recibir como parámetro el sexo, y la edad).

```
CREATE OR REPLACE FUNCTION suma_edades (genero VARCHAR(15), edad INT)
RETURNS INT
BEGIN
    RETURN (
        SELECT SUM(est.edad)
        FROM estudiantes AS est
        WHERE est.sexo = genero AND est.edad >= edad
    );
END;

SELECT est.nombres, est.apellidos, ins.semestre
FROM estudiantes AS est
INNER JOIN inscripcion AS ins on est.id_est = ins.id_est
INNER JOIN materias AS mat on ins.id_mat = mat.id_mat
WHERE suma_edades( genero: 'masculino', edad: 21) % 2 = 0 AND est.sexo = 'masculino' AND est.edad >= 21;
```

}

	nombres	apellidos	semestre
1	Joel	Adubiri Mondar	2do Semestre
2	Joel	Adubiri Mondar	3er Semestre
3	Santos	Montes Valenzuela	5to Semestre

# Ejercicio 7 { Primera Forma

```
CREATE FUNCTION comparanombres (nombre VARCHAR(60), nombrecom VARCHAR(50), apellido VARCHAR(50), apellidocomp VARCHAR(50))
RETURNS BOOLEAN
BEGIN
    DECLARE comp BOOLEAN DEFAULT FALSE;
    SET
        comp = (nombre = nombrecom AND apellido = apellidocomp);
    RETURN comp;
END;

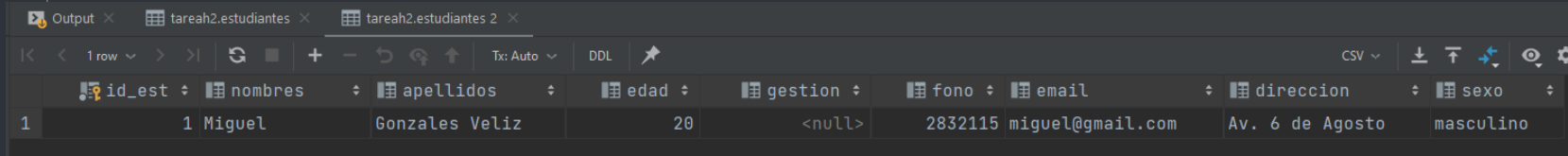
SELECT est.*
FROM estudiantes AS est
WHERE comparanombres ( nombre: est.nombres, nombrecom: est.apellidos, apellido: 'Miguel', apellidocomp: 'Gonzales Veliz');
```

id_est	nombres	apellidos	edad	gestion	fono	email	direccion	sexo
1	Miguel	Gonzales Veliz	20	<null>	2832115	miguel@gmail.com	Av. 6 de Agosto	masculino

}

# Ejercicio 7 { Primera Forma

```
SELECT est.*  
FROM estudiantes AS est  
WHERE comparanombres ( nombre: 'Miguel', apellido: 'Gonzales Veliz') AND est.nombres='Miguel' AND est.apellidos='Gonzales Veliz';
```



id_est	nombres	apellidos	edad	gestion	fono	email	direccion	sexo
1	1 Miguel	Gonzales Veliz	20	<null>	2832115	miguel@gmail.com	Av. 6 de Agosto	masculino

# Ejercicio 7 { Primera Forma

```
CREATE OR REPLACE FUNCTION comparanombres (nombre VARCHAR(60), apellido VARCHAR(50))
RETURNS BOOLEAN
BEGIN
    DECLARE cont BOOLEAN DEFAULT FALSE;
    DECLARE num VARCHAR (100);

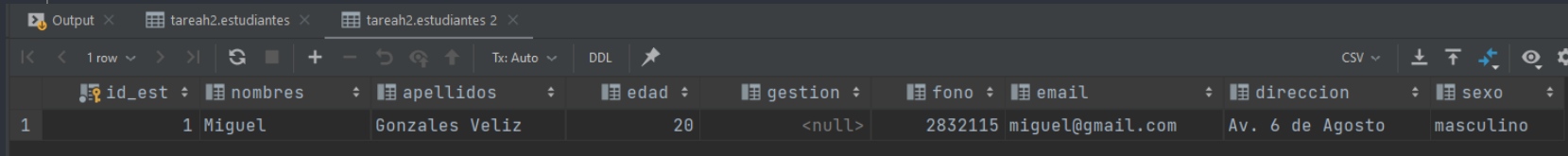
    SET num =
    (
        SELECT est.nombres
        FROM estudiantes AS est
        WHERE est.nombres = nombre AND apellido = est.apellidos
    );

    IF (num IS NOT NULL)
    THEN
        SET cont = 1;
    ELSE
        SET cont = 0;
    END IF;

    RETURN cont;
END;
```

# Ejercicio 7 { Primera Forma

```
SELECT est.*  
FROM estudiantes AS est  
WHERE comparanombres ( nombre: 'Miguel', apellido: 'Gonzales Veliz') AND est.nombres='Miguel' AND est.apellidos='Gonzales Veliz';
```



id_est	nombres	apellidos	edad	gestion	fono	email	direccion	sexo
1	1 Miguel	Gonzales Veliz	20	<null>	2832115	miguel@gmail.com	Av. 6 de Agosto	masculino



GRACIAS 'por Ver' { "Si  
puedes imaginarlo puedes  
programarlo" (Alejandro  
Taboada)

}

