

DEFENSA HITO 2

ESTUDIANTE:

Karla Belen Diaz Flores

ASIGNATURA:

Base De Datos II

CARRERA:

Ingeniería De Sistemas

DOCENTE:

Ing. William Roddy Barra

GESTIÓN:

2022



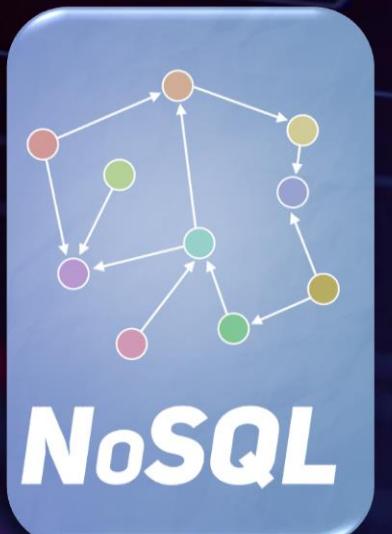
1. ¿A que se refiere cuando se habla de bases de datos relacionales?

Son una colección de elementos de datos organizados en un conjunto de tablas formalmente descritas, desde donde se puede acceder a los datos sin tener que reorganizar las tablas de la base. La interfaz estándar de programa de usuario y aplicación a una base de datos relacional, es el Lenguaje de Consultas Estructuradas (SQL).



2. ¿A que se refiere cuando se habla de bases de datos no relacionales?

Una base de datos no relacional es aquella que no usa el esquema tabular de filas y columnas que se encuentra en la mayoría de los sistemas de base de datos más tradicionales.



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

MySQL:

Es un sistema de gestión de bases de datos relacional (Relational Database Management System, RDBMS) de código abierto, multihilo y multiusuario.

MariaDB:

Es un potente sistema de base de datos objeto-relacional de código abierto.

Diferencias:

MariaDB es un sustituto de MySQL, con licencia GPL, en donde se incorporan todas las mejoras con más funcionalidades y un máximo rendimiento que permite modificar, almacenar y extraer información para servicios SQL sólidos y escalables. Fue desarrollado por Michael Widenius, fundador de MySQL y la comunidad de desarrolladores de software libre.



4. ¿Qué son las funciones de agregación?

Las funciones de agregación se usan dentro de la cláusula SELECT en grupos de registros de devolver un único valor que se aplica a un grupo de registros.

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.



5. ¿Qué llegaría a ser XAMPP?

XAMPP es una distribución de Apache completamente gratuita y fácil de instalar que contiene MariaDB, PHP y Perl. El paquete de instalación de XAMPP ha sido diseñado para ser increíblemente fácil de instalar y usar.



XAMPP

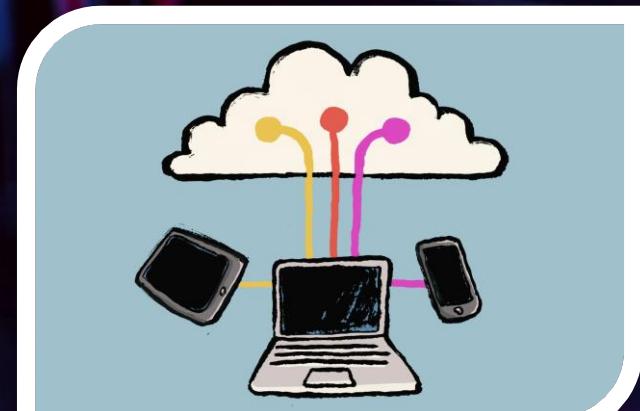
6. ¿Cuál es la diferencia entre las funciones de agregación y funciones creados por el DBA? Es decir funciones creadas por el usuario.

La diferencia está en que, las funciones definidas por el usuario de SQL Server son rutinas que aceptan parámetros, realizan una acción, como un cálculo complejo, y devuelven el resultado de esa acción como un valor.



7. ¿Para qué sirve el comando USE?

El comando USE le indica al MySQL que use la base de datos creada con el nombre que le ponga el usuario por defecto para los comandos que se utilicen a continuación se puedan ejecutar.



8. Qué es DML y DDL?

DML:

Utilizando instrucciones de SQL, permite a los usuarios introducir datos para posteriormente realizar tareas de consultas o modificación de los datos que contienen las Bases de Datos.

Se utilizan los siguientes elementos:

SELECT, esta sentencia se utiliza para realizar consultas sobre los datos.

INSERT, con esta instrucción podemos insertar los valores en una base de datos.

UPDATE, sirve para modificar los valores de uno o varios registros.

DELETE, se utiliza para eliminar las filas de una tabla.

DDL:

Este lenguaje permite a los programadores de un sistema gestor de base de datos definir las estructuras que almacenarán los datos así como los procedimientos o funciones que permitan consultarlos.

Para definir las estructura disponemos de tres sentencias:

CREATE, se usa para crear una base de datos, tabla, vistas, etc.

ALTER, se utiliza para modificar la estructura, por ejemplo añadir o borrar columnas de una tabla.

DROP, con esta sentencia, podemos eliminar los objetos de la estructura, por ejemplo un índice o una secuencia.



9. ¿Qué cosas características debe de tener una función? Explique sobre el nombre, el return, parametros, etc.

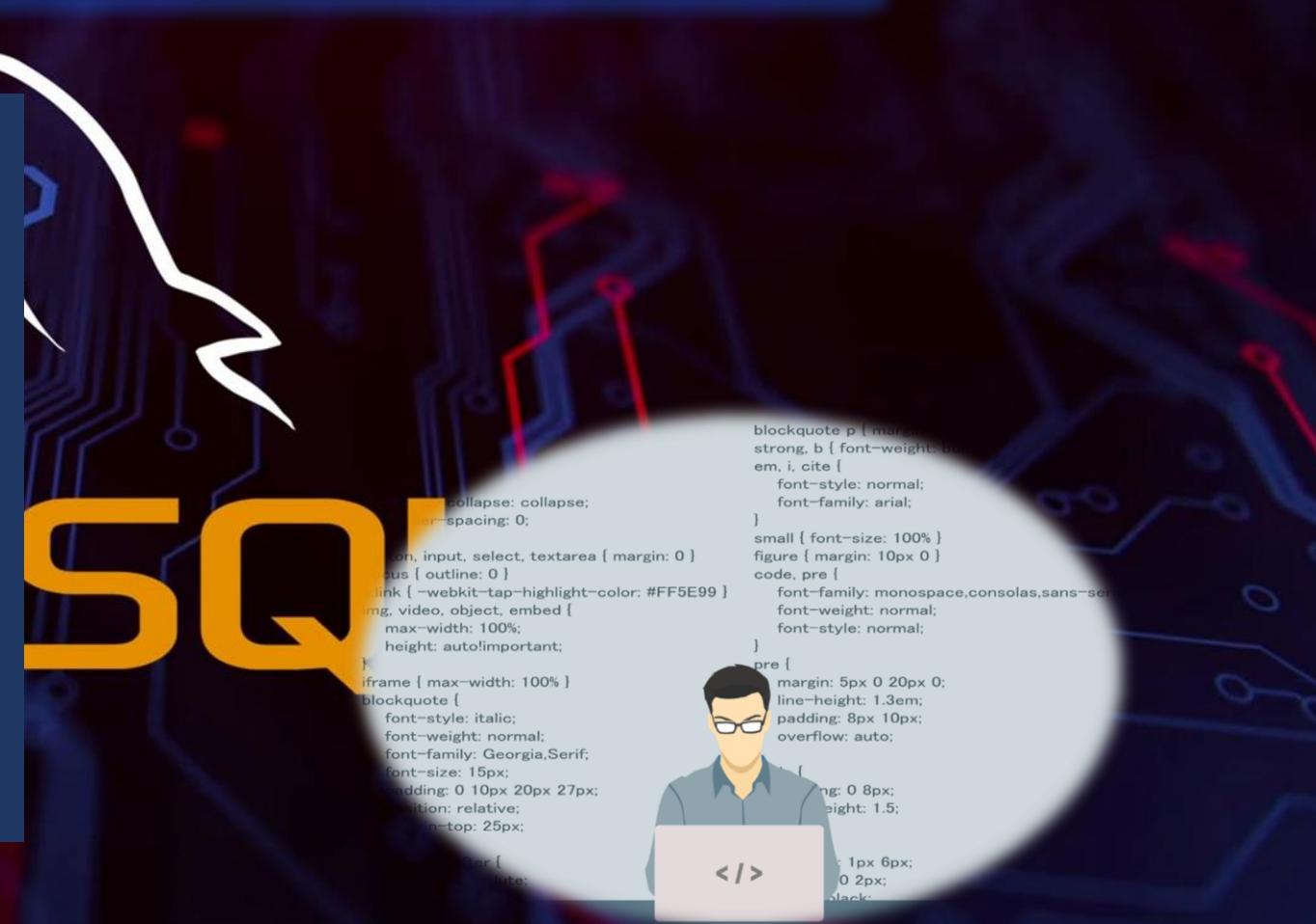
Las funciones deben tener las siguientes características:

NOMBRE: El usuario debe determinar el nombre de la función después de la palabra clave function

RETURN: El return en nuestra función, nos devolverá un resultado en integer para los números y varchar para las palabras.

PARAMETROS: Los parámetros son variables cuyo valor se utiliza dentro del procedimiento almacenado, como una función.

DECLARACIONES: Las declaraciones nos sirven para realizar condiciones.



10. ¿Cómo crear, modificar y cómo eliminar una función?

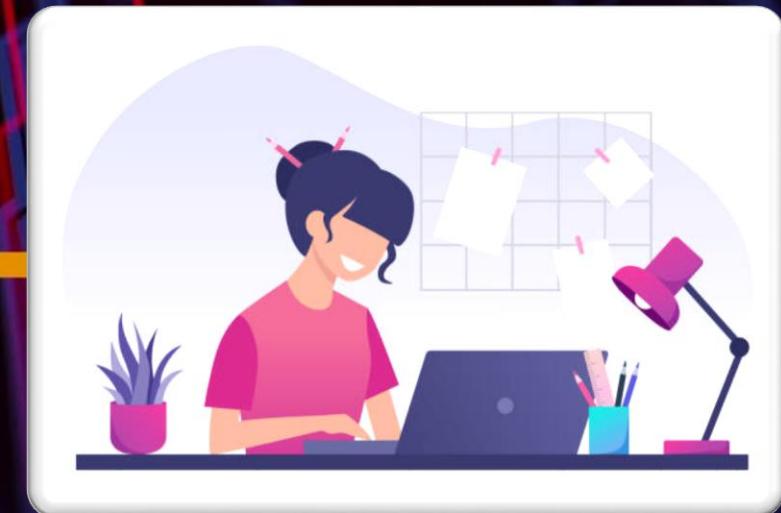
Primeramente para crear una función usamos CREATE FUNCTION luego designamos un nombre a nuestra función y los parámetros si se necesitan, luego viene el returns para devolver valores y declaramos nuestra función de esta forma ya estaría creada nuestra función.

Para modificar nuestra función usamos OR REPLACE el cual agregamos a nuestra función de la siguiente forma:

CREATE OR REPLACE FUNCTION

Para eliminar una función debemos ejecutar la siguiente sentencia:

DROP FUNCTION y el nombre de nuestra función.



PARTE PRACTICA

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

```
CREATE DATABASE POLLOS_COPA;
USE POLLOS_COPA;

CREATE TABLE cliente
(
    id_cliente INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
    fullname VARCHAR(50),
    lastname VARCHAR(50),
    edad INTEGER,
    domicilio VARCHAR(50)
);

CREATE TABLE Pedido
(
    id_pedido INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
    articulo VARCHAR(80),
    costo VARCHAR(50),
    fecha VARCHAR(50)
);

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

INSERT INTO cliente(fullname, lastname, edad, domicilio)
VALUES ('Juan', 'Velazco Gonzales', 25, 'Av. Mejillones-100A');
INSERT INTO cliente(fullname, lastname, edad, domicilio)
VALUES ('Lucia', 'Fernandez Torrez', 25, 'Av. Rosa-255R');

INSERT INTO Pedido(articulo, costo, fecha)
VALUE ('LA SUPER BURGUER', '20Bs', '12/05/2021');
INSERT INTO Pedido(articulo, costo, fecha)
VALUE ('POLLO-SPIEDO', '15Bs', '22/06/2021');

INSERT INTO detalle_pedido (id_cliente, id_pedido)
VALUES (1,1);
INSERT INTO detalle_pedido (id_cliente, id_pedido)
VALUES (2,2);
```

12. Crear una consulta SQL en base al ejercicio anterior.

- ❖ Debe de utilizar las 3 tablas creadas anteriormente.
- ❖ Para relacionar las tablas utilizar JOINS.
- ❖ Adjuntar el código SQL generado

```
SELECT cli.id_cliente, cli.fullname, cli.lastname, cli.edad, cli.domicilio,  
ped.articulo, ped.costo, ped.fecha  
FROM cliente AS cli
```

```
    INNER JOIN detalle_pedido AS dp ON cli.id_cliente =  
dp.id_detalle_pedido
```

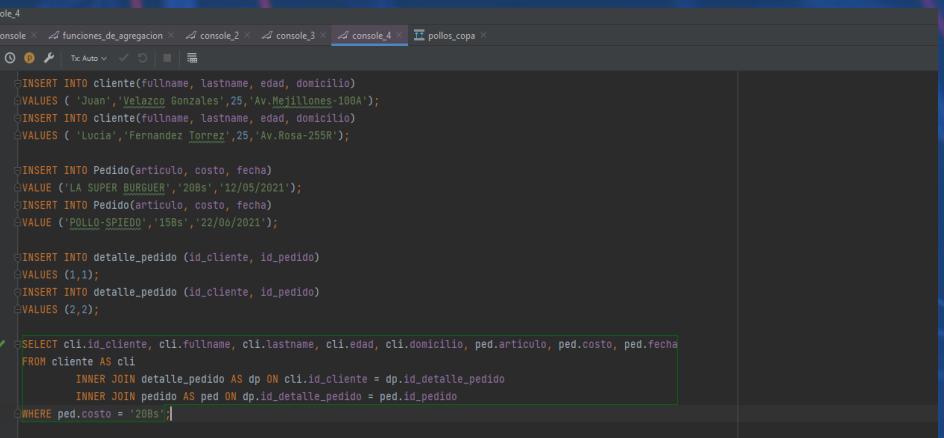
```
    INNER JOIN pedido AS ped ON dp.id_detalle_pedido = ped.id_pedido  
WHERE ped.costo = '15Bs';
```



```
29  
30 INSERT INTO cliente(fullname, lastname, edad, domicilio)  
31 VALUES ('Juan', 'Velazco Gonzales', 25, 'Av.Mejillones-100A');  
32 INSERT INTO cliente(fullname, lastname, edad, domicilio)  
33 VALUES ('Lucia', 'Fernandez Torrez', 25, 'Av.Rosa-255R');  
34  
35 INSERT INTO Pedido(articulo, costo, fecha)  
36 VALUE ('LA SUPER BURGUER', '20Bs', '12/05/2021');  
37 INSERT INTO Pedido(articulo, costo, fecha)  
38 VALUE ('POLLO-SPIEDO', '15Bs', '22/06/2021');  
39  
40 INSERT INTO detalle_pedido (id_cliente, id_pedido)  
41 VALUES (1,1);  
42 INSERT INTO detalle_pedido (id_cliente, id_pedido)  
43 VALUES (2,2);  
44  
45 ✓ SELECT cli.id_cliente, cli.fullname, cli.lastname, cli.edad, cli.domicilio, ped.articulo, ped.costo, ped.fecha  
FROM cliente AS cli  
    INNER JOIN detalle_pedido AS dp ON cli.id_cliente = dp.id_detalle_pedido  
    INNER JOIN pedido AS ped ON dp.id_detalle_pedido = ped.id_pedido  
WHERE ped.costo = '15Bs';
```

Output x Result 11 x

id_cliente	fullname	lastname	edad	domicilio	articulo	costo	fecha
1	2 Lucia	Fernandez Torrez	25	Av.Rosa-255R	POLLO-SPIEDO	15Bs	22/06/2021



```
30  
31 INSERT INTO cliente(fullname, lastname, edad, domicilio)  
32 VALUES ('Juan', 'Velazco Gonzales', 25, 'Av.Mejillones-100A');  
33 INSERT INTO cliente(fullname, lastname, edad, domicilio)  
34 VALUES ('Lucia', 'Fernandez Torrez', 25, 'Av.Rosa-255R');  
35  
36 INSERT INTO Pedido(articulo, costo, fecha)  
37 VALUE ('LA SUPER BURGUER', '20Bs', '12/05/2021');  
38 INSERT INTO Pedido(articulo, costo, fecha)  
39 VALUE ('POLLO-SPIEDO', '15Bs', '22/06/2021');  
40  
41 INSERT INTO detalle_pedido (id_cliente, id_pedido)  
42 VALUES (1,1);  
43 INSERT INTO detalle_pedido (id_cliente, id_pedido)  
44 VALUES (2,2);  
45 ✓ SELECT cli.id_cliente, cli.fullname, cli.lastname, cli.edad, cli.domicilio, ped.articulo, ped.costo, ped.fecha  
FROM cliente AS cli  
    INNER JOIN detalle_pedido AS dp ON cli.id_cliente = dp.id_detalle_pedido  
    INNER JOIN pedido AS ped ON dp.id_detalle_pedido = ped.id_pedido  
WHERE ped.costo = '20Bs';
```

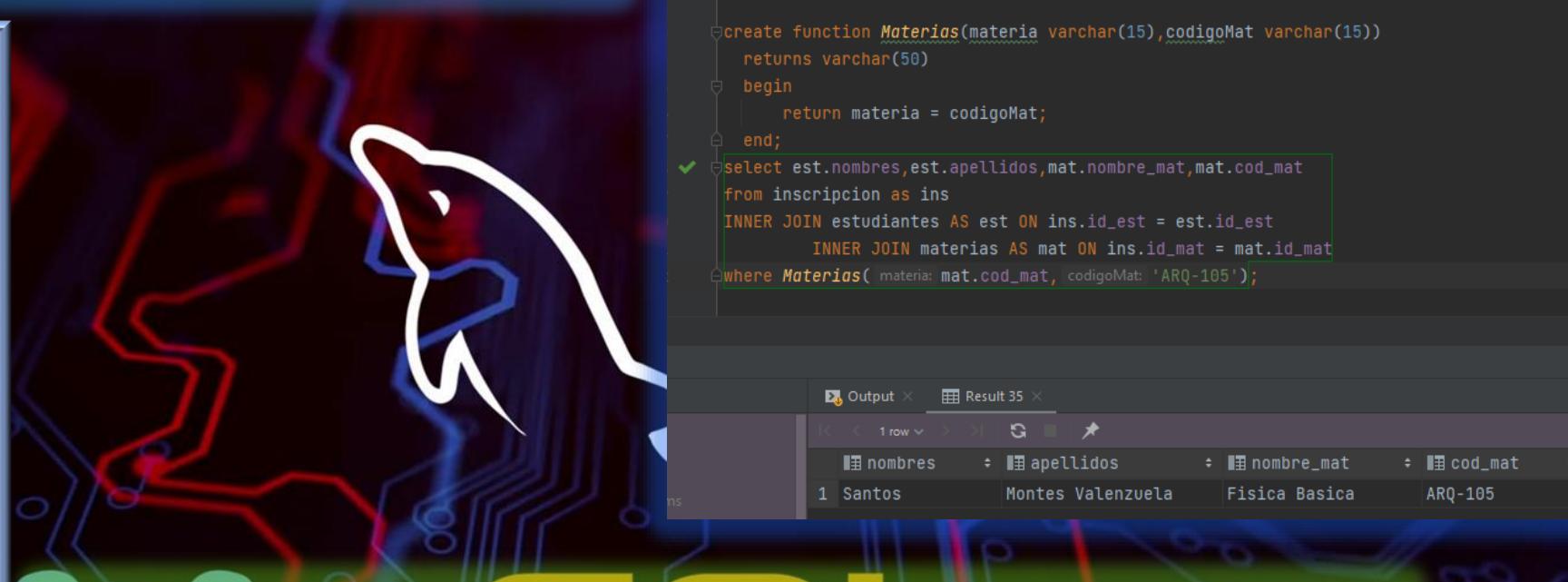
Output x Result 12 x

id_cliente	fullname	lastname	edad	domicilio	articulo	costo	fecha
1	1 Juan	Velazco Gonzales	25	Av.Mejillones-100A	LA SUPER BURGUER	20Bs	12/05/2021

13.Crear un función que compare dos códigos de materia.

- ❖ Resolver lo siguiente:
 - 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 Materias(materia varchar(15),codigoMat varchar(15))
returns varchar(50)
begin
    return materia = codigoMat;
end;
select est.nombres,est.apellidos,mat.nombre_mat,mat.cod_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 Materias( mat.cod_mat, codigoMat: 'ARQ-105');
```



```
create function Materias(materia varchar(15),codigoMat varchar(15))
returns varchar(50)
begin
    return materia = codigoMat;
end;
select est.nombres,est.apellidos,mat.nombre_mat,mat.cod_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 Materias( mat.cod_mat, codigoMat: 'ARQ-105');
```

The screenshot shows a database query editor window. The code in the editor is identical to the one in the text box above. Below the code, there are two tabs: 'Output' and 'Result 35'. The 'Result' tab displays a single row of data:

nombres	apellidos	nombre_mat	cod_mat
1 Santos	Montes Valenzuela	Fisica Basica	ARQ-105

14.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.

- ❖ La función recibe como parámetro solo el género.
- ❖ La función retorna un valor numérico.

```
CREATE FUNCTION Promedio_edades(genero varchar(20)) returns
int
begin
return (
    SELECT AVG(est.edad)
    FROM estudiantes AS est
    INNER JOIN inscripcion AS i ON est.id_est = i.id_est
    INNER JOIN materias AS m ON i.id_mat = m.id_mat
    WHERE m.cod_mat = 'ARQ-104'
);
end;

select Promedio_edades('masculino');
```

The screenshot shows a MySQL Workbench interface with a code editor and a results table.

Code Editor Content:

```
# La función retorna un valor numérico.
CREATE FUNCTION Promedio_edades(genero varchar(20)) returns
int
begin
return (
    SELECT AVG(est.edad)
    FROM estudiantes AS est
    INNER JOIN inscripcion AS i ON est.id_est = i.id_est
    INNER JOIN materias AS m ON i.id_mat = m.id_mat
    WHERE m.cod_mat = 'ARQ-104'
);

```

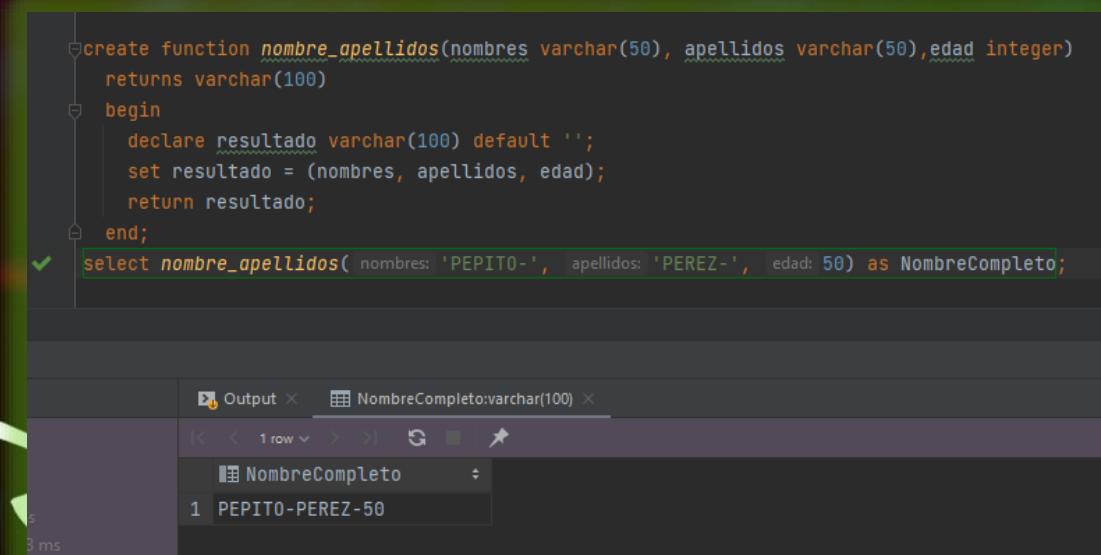
Execution Result:

Output	Promedio_edades('masculino') : int(11)
3 ms	1 23

15.Crear una función que permita concatenar 3 cadenas.

- ❖ La función recibe 3 parámetros.
- ❖ Si la cadenas fuesen:
 - ❖ ■ Pepito
 - ❖ ■ Perez
 - ❖ ■ 50
- ❖ o La salida debería ser:
Pepito - Perez - 50

```
create function nombre_apellidos(nombres varchar(50), apellidos  
varchar(50),edad integer)  
returns varchar(100)  
begin  
declare resultado varchar(100) default '';  
set resultado = (nombres, apellidos, edad);  
return resultado;  
end;  
select nombre_apellidos('PEPITO-', 'PEREZ-', 50) as NombreCompleto;
```



```
create function nombre_apellidos(nombres varchar(50), apellidos varchar(50),edad integer)  
returns varchar(100)  
begin  
declare resultado varchar(100) default '';  
set resultado = (nombres, apellidos, edad);  
return resultado;  
end;  
select nombre_apellidos( nombres: 'PEPITO-' , apellidos: 'PEREZ-' , edad: 50) as NombreCompleto;
```

NombreCompleto
1 PEPITO-PEREZ-50

16.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.
- ❖ Debe de crear una función que sume las edades (recibir como parámetro el sexo, y la edad).
- ❖ ■ Ejemplo: sexo='Masculino' y edad=22
- ❖ ■ Note que la función recibe 2 parámetros.
- ❖ La función creada anteriormente debe utilizarse en la consulta SQL. (Cláusula WHERE).

```
create function sum_edades(sexo varchar(10),edad integer)
returns varchar(20)
begin
declare sumaEdades integer default 0;

select SUM(est.edad) INTO sumaEdades
from estudiantes AS est
where est.sexo = sexo and est.edad > est.edad;

return sumaEdades;

end;

select est.nombres, est.apellidos,ins.semestre
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 sum_edades('masculino',22) %2 = 0;
```

17. Crear una función de acuerdo a lo siguiente:

- ❖ 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).
- ❖ ■ La función devuelve un boolean.
- ❖ ■ La función debe recibir el nombre y sus apellidos.

The screenshot shows a database development environment with multiple tabs open. The active tab, 'console_5', contains PL/SQL code for creating a function named 'comparar'. The function takes two parameters: 'nombres_comparar' and 'apellidos_comparar', both of type VARCHAR(20). It declares a boolean variable 'respuesta' and initializes it to false. It then compares the input parameters with the 'nombres' and 'apellidos' columns from the 'estudiantes' table. If they match, it sets 'respuesta' to true and returns it. A select statement is also present to retrieve all student data where the names and surnames match the specified parameters ('Andrea' and 'Arias Ballesteros').

Output window below shows the result of the query, displaying one row with id_est=1, nombres='Andrea', apellidos='Arias Ballesteros', edad=21, gestion=NULL, telefono=2832118, email='andrea@gmail.com', direccion='Av. 6 de Agosto', and sexo='femenino'.

```
create function Materias(materia varchar(15),codigoMat varchar(15))
returns varchar(50)
begin
    return materia = codigoMat;
end;
select est.nombres,est.apellidos,mat.nombre_mat,mat.cod_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 Materias(mat.cod_mat,'ARQ-105');
```



¡GRACIAS POR SU ATENCIÓN!

ENLACE DEL VÍDEO

[HTTPS://DRIVE.GOOGLE.COM/FILE/D/1XMTUD308MRYTXGDOFBH85I18YUXOGH/VIEW?USP=SHARING](https://drive.google.com/file/d/1XMTUD308MRYTXGDOFBH85I18YUXOGH/view?usp=sharing)