

DEFENSA HITO 3

ESTUDIANTE:

Karla Belen Diaz Flores

ASIGNATURA:

Base De Datos II

CARRERA:

Ingeniería De Sistemas

DOCENTE:

Ing. William Roddy Barra

GESTIÓN:

2022



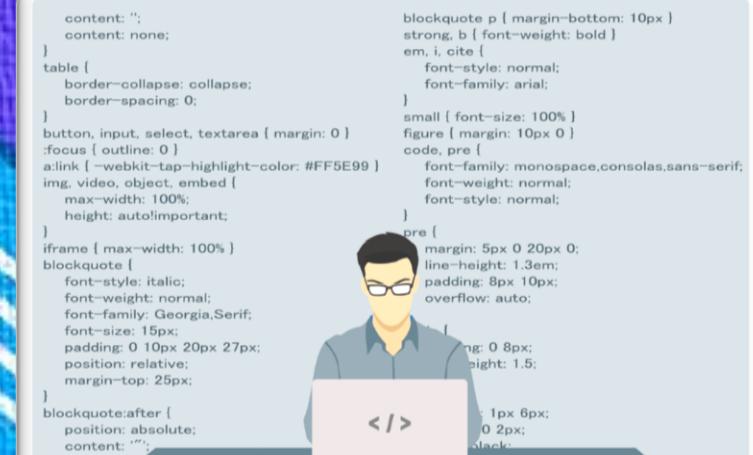
```
content: "";
content: none;
}
table {
border-collapse: collapse;
border-spacing: 0;
}
button, input, select, textarea { margin: 0 }
:focus { outline: 0 }
a:link { -webkit-tap-highlight-color: #FF5E99 }
img, video, object, embed {
max-width: 100%;
height: auto!important;
}
iframe { max-width: 100% }
blockquote {
font-style: italic;
font-weight: normal;
font-family: Georgia,Serif;
font-size: 15px;
padding: 0 10px 20px 27px;
position: relative;
margin-top: 25px;
}
blockquote:after {
position: absolute;
content: " ";
}
blockquote:after {
position: absolute;
content: " ";
}
```



1. Defina que es lenguaje procedural en MySQL.

Los procedimientos almacenados MySQL, también conocidos como Stored Procedure, se presentan como conjuntos de instrucciones escritas en el lenguaje SQL. Su objetivo es realizar una tarea determinada, desde operaciones sencillas hasta tareas muy complejas. Los procedimientos almacenados MySQL contienen una o más instrucciones SQL además de un procesamiento manipulador o lógico.

La característica fundamental de los procedimientos almacenados MySQL es que estos comandos se quedan almacenados y se ejecutan en el servidor o en el motor de bases de datos.



2. Defina que es una función en MySQL.

Las funciones son piezas de código que reciben datos de entrada, realizan operaciones con ellos y luego devuelven un resultado.



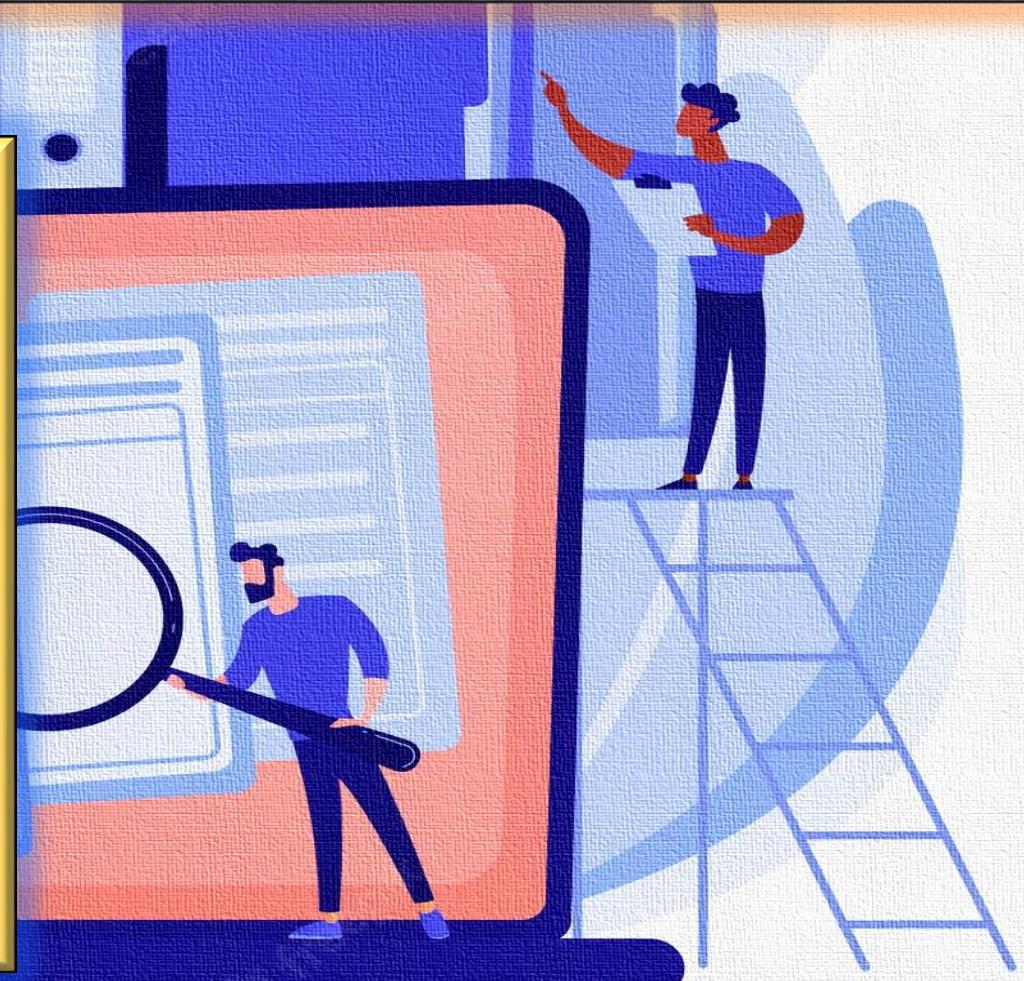
3. ¿Qué cosas características debe de tener una función? Explique sobre el nombre, el return, parametros, etc.

Las siguientes características son las mas importantes en una función:

El nombre es esencial para crear nuestra función con este podemos llamar a la función para mostrar en pantalla.

El return nos sirve para concluir la ejecución de una rutina. Para las funciones, devuelve el resultado de la función o método. Para un procedimiento, devuelve opcionalmente un valor de estado de tipo entero.

Los parámetros se usan para intercambiar datos entre las funciones y los procedimientos almacenados y la aplicación o la herramienta que llamó a la función o al procedimiento almacenados: Los parámetros de entrada permiten a quien realiza la llamada pasar un valor de datos a la función o al procedimiento almacenado.



4. ¿Cómo crear, modificar y cómo eliminar una función? Adjunte un ejemplo de su uso.

create function suma_edades

create or replace function suma_edades

drop function suma_edades;

5. Para qué sirve la función CONCAT y como funciona en MYSQL ○ ¿Crear una función que muestre el uso de las función CONCAT? ○ La función debe concatenar 3 cadenas.

CONCAT es una función de cadena compatible con MySQL para combinar o unir dos o más cadenas y devolverlas como un solo valor. El nombre CONCAT proviene del verbo concatenación, que significa unir 2 o más entidades juntas.

```
create or replace function manejoDeLoop(limite int)
returns text
begin
declare str text default "";

manejo_de_loop: loop
if limite < 0
then
leave manejo_de_loop;
end if;

set str = concat(str, limite, ', ');
set limite = limite - 1;

iterate manejo_de_loop;
end loop;

return str;
end;

select manejoDeLoop(10);
```

6. Para qué sirve la función SUBSTRING y como funciona en MYSQL

- ¿Crear una función que muestre el uso de las función SUBSTRING?
 - La función recibe un nombre completo.
- INPUT: Ximena Condori Mar
- La función solo retorna el nombre.
- OUTPUT: Ximena

La función de subcadena de MySQL se utiliza para extraer una subcadena o una parte de la cadena contra la cadena de entrada. Como sugiere el nombre, la función Substring opera en una cadena de entrada y devuelve una subcadena más pequeña contra las opciones especificadas.

```
create function USAR_SUBSTRING(primer  
VARCHAR(30))
```

```
returns TEXT
```

```
begin
```

```
declare str TEXT;
```

```
declare x integer default 1;
```

```
repeat
```

```
set str = SUBSTRING(primer,x,15 );
```

```
set x = x + 1;
```

```
until x <= 20 END REPEAT;
```

```
return str;
```

```
end;
```

```
Select USAR_SUBSTRING('Ximena');
```

7. Para qué sirve la función STRCMP y como funciona en MYSQL

La función STRCMP() en MySQL se usa para comparar dos strings. Si ambas strings son iguales, devuelve 0, si el primer argumento es más pequeño que el segundo según el orden definido, devuelve -1 y devuelve 1 cuando el segundo es más pequeño que el primero.

8. Para qué sirve la función CHAR_LENGTH y LOCATE y como funciona en MYSQL

La función CHAR_LENGTH() en MySQL se usa para encontrar la longitud de una string dada (en caracteres). Cuenta el número de caracteres e ignora si los caracteres son de un solo byte o de varios bytes.

La función LOCATE() en MySQL se usa para encontrar la ubicación de una substring en una string. Devolverá la ubicación de la primera aparición de la substring en la string. Si la substring no está presente en la string, devolverá 0.

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

Las funciones de agregación en SQL nos permiten efectuar operaciones sobre un conjunto de resultados, pero devolviendo un único valor agregado para todos ellos. Es decir, nos permiten obtener medias, máximos, etc... sobre un conjunto de valores.

Una función definida por el usuario (UDF) es un modo de extender MySQL con una nueva función que funciona como una función nativa de MySQL tal como ABS() o CONCAT(). function_name es el nombre que debe usarse en comandos SQL para invocar la función.

10. ¿Busque y defina a qué se referirá cuando se habla de parámetros de entrada y salida en MySQL? o Es decir IN INOUT, etc

Como has visto, los parámetros se definen separados por una coma. Los parámetros de los procedimientos almacenados de MySQL pueden ser de tres tipos:

IN: Es el tipo de parámetro que se usa por defecto. La aplicación o código que invoque al procedimiento tendrá que pasar un argumento para este parámetro. El procedimiento trabajará con una copia de su valor, teniendo el parámetro su valor original al terminar la ejecución del procedimiento.

OUT: El valor de este parámetro puede ser cambiado en el procedimiento, y además su valor modificado será enviado de vuelta al código o programa que invoca el procedimiento.

INOUT: Es una mezcla de los dos conceptos anteriores. La aplicación o código que invoca al procedimiento puede pasarle un valor a éste, devolviendo el valor modificado al terminar la ejecución. En caso de resultarte confuso, echa un ojo al ejemplo que verás más adelante.

PARTE PRACTICA

11. Crear la siguiente Base de datos y sus registros.

```

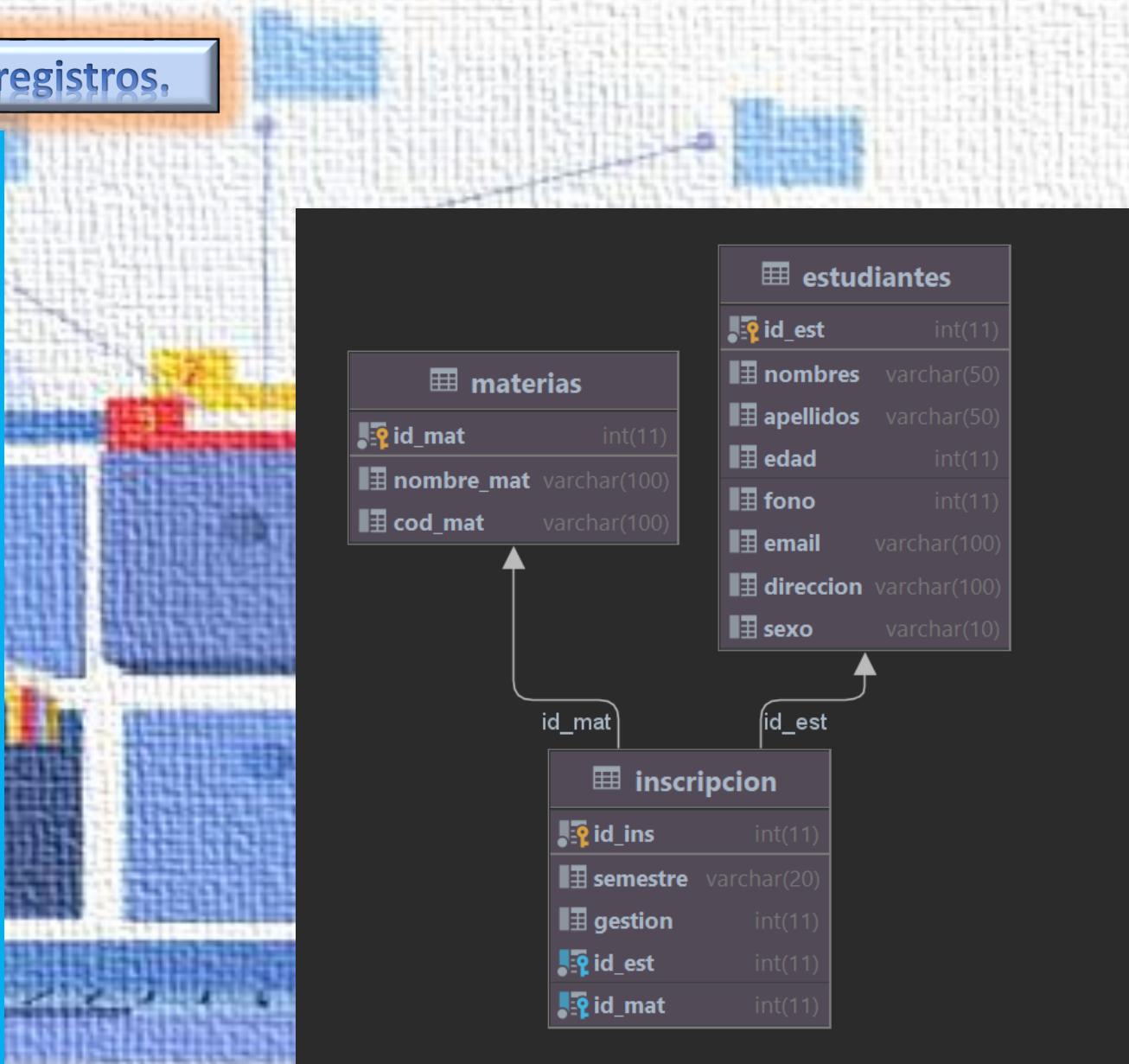
CREATE DATABASE Defensa_Hito3;
USE Defensa_Hito3;

CREATE TABLE estudiantes
(
    id_est INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
    nombres VARCHAR(50),
    apellidos VARCHAR(50),
    edad INTEGER,
    fono INTEGER,
    email VARCHAR(100),
    direccion VARCHAR(100),
    sexo VARCHAR(10)
);
INSERT INTO estudiantes (nombres, apellidos, edad, fono, email, direccion, sexo)
VALUES ('Miguel', 'Gonzales Veliz', 20, 2832115, 'miguel@gmail.com', 'Av. 6 de Agosto', 'masculino');
INSERT INTO estudiantes (nombres, apellidos, edad, fono, email, direccion, sexo)
VALUES ('Sandra', 'Mavri Urias', 25, 2832116, 'sandra@gmail.com', 'Av. 6 de Agosto', 'femenino');
INSERT INTO estudiantes (nombres, apellidos, edad, fono, email, direccion, sexo)
VALUES ('Joel', 'Adubiri Mondar', 30, 2832117, 'joel@gmail.com', 'Av. 6 de Agosto', 'masculino');
INSERT INTO estudiantes (nombres, apellidos, edad, fono, email, direccion, sexo)
VALUES ('Andrea', 'Arias Ballesteros', 21, 2832118, 'andrea@gmail.com', 'Av. 6 de Agosto', 'femenino');
INSERT INTO estudiantes (nombres, apellidos, edad, fono, email, direccion, sexo)
VALUES ('Santos', 'Montes Valenzuela', 24, 2832119, 'santos@gmail.com', 'Av. 6 de Agosto', 'masculino');

CREATE TABLE materias
(
    id_mat INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
    nombre_mat VARCHAR(100),
    cod_mat VARCHAR(100)
);
INSERT INTO materias (nombre_mat, cod_mat) VALUES ('Introducción a la Arquitectura','ARQ-101');
INSERT INTO materias (nombre_mat, cod_mat) VALUES ('Urbanismo y Diseño','ARQ-102');
INSERT INTO materias (nombre_mat, cod_mat) VALUES ('Dibujo y Pintura Arquitectónico','ARQ-103');
INSERT INTO materias (nombre_mat, cod_mat) VALUES ('Matemática discreta','ARQ-104');
INSERT INTO materias (nombre_mat, cod_mat) VALUES ('Física Básica','ARQ-105');

CREATE TABLE inscripcion
(
    id_ins INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
    semestre VARCHAR(20),
    gestion INTEGER,
    id_est INT NOT NULL,
    id_mat INT NOT NULL,
    FOREIGN KEY (id_est) REFERENCES estudiantes (id_est),
    FOREIGN KEY (id_mat) REFERENCES materias (id_mat)
);
INSERT INTO inscripcion (id_est, id_mat, semestre, gestion) VALUES (1, 1, '1er Semestre', 2018);
INSERT INTO inscripcion (id_est, id_mat, semestre, gestion) VALUES (1, 2, '2do Semestre', 2018);
INSERT INTO inscripcion (id_est, id_mat, semestre, gestion) VALUES (2, 4, '1er Semestre', 2019);
INSERT INTO inscripcion (id_est, id_mat, semestre, gestion) VALUES (2, 3, '2do Semestre', 2019);
INSERT INTO inscripcion (id_est, id_mat, semestre, gestion) VALUES (3, 3, '2do Semestre', 2020);
INSERT INTO inscripcion (id_est, id_mat, semestre, gestion) VALUES (3, 1, '3er Semestre', 2020);
INSERT INTO inscripcion (id_est, id_mat, semestre, gestion) VALUES (4, 4, '4to Semestre', 2021);
INSERT INTO inscripcion (id_est, id_mat, semestre, gestion) VALUES (5, 5, '5to Semestre', 2021);

```



12.Crear una función que genere la serie Fibonacci.

```
CREATE FUNCTION seriefibonacci(limite INT)
RETURNS TEXT
BEGIN
DECLARE J TEXT DEFAULT '';
DECLARE suma INT DEFAULT 0;
DECLARE a INT DEFAULT 0;
DECLARE b INT DEFAULT 1;
DECLARE cont INT DEFAULT 0;
WHILE cont<limite DO
SET J = CONCAT(J,a,',');
SET b=a+b;
SET a=b-a;
SET cont=cont+1;
END WHILE ;
RETURN J;
END;

SELECT seriefibonacci(7);
```

The screenshot shows the MySQL Workbench interface with the following details:

- Code Editor:** Displays the SQL code for creating a function named `seriefibonacci` that takes an integer parameter `limite` and returns a text string representing the Fibonacci sequence up to that limit. The code uses a cursorless WHILE loop to build the sequence in variable `J`.
- Output Window:** Shows the results of executing the function with the argument `7`. The output is:

```
1 0,1,1,2,3,5,8,
```

with a total execution time of `1 s 259 ms`.

13.Crear una variable global a nivel BASE DE DATOS.

```
SET @userAll='ADMIN';
```

```
CREATE FUNCTION  
variable_global(DATA text)  
RETURNS TEXT  
BEGIN  
SET @userAll=data;  
RETURN @userAll;  
END;
```

```
SELECT  
variable_global('Claymore');
```

```
80  
81     SET @userAll='ADMIN';  
82  
83     CREATE FUNCTION variable_global(DATA text)  
84         RETURNS TEXT  
85         BEGIN  
86             SET @userAll=data;  
87             RETURN @userAll;  
88         END;  
89  
90 ✓  SELECT variable_global( DATA: 'Claymore');
```

Output × variable_global('Claymore'):text

	variable_global('Claymore')
1	Claymore

variable_global('Claymore'): longtext

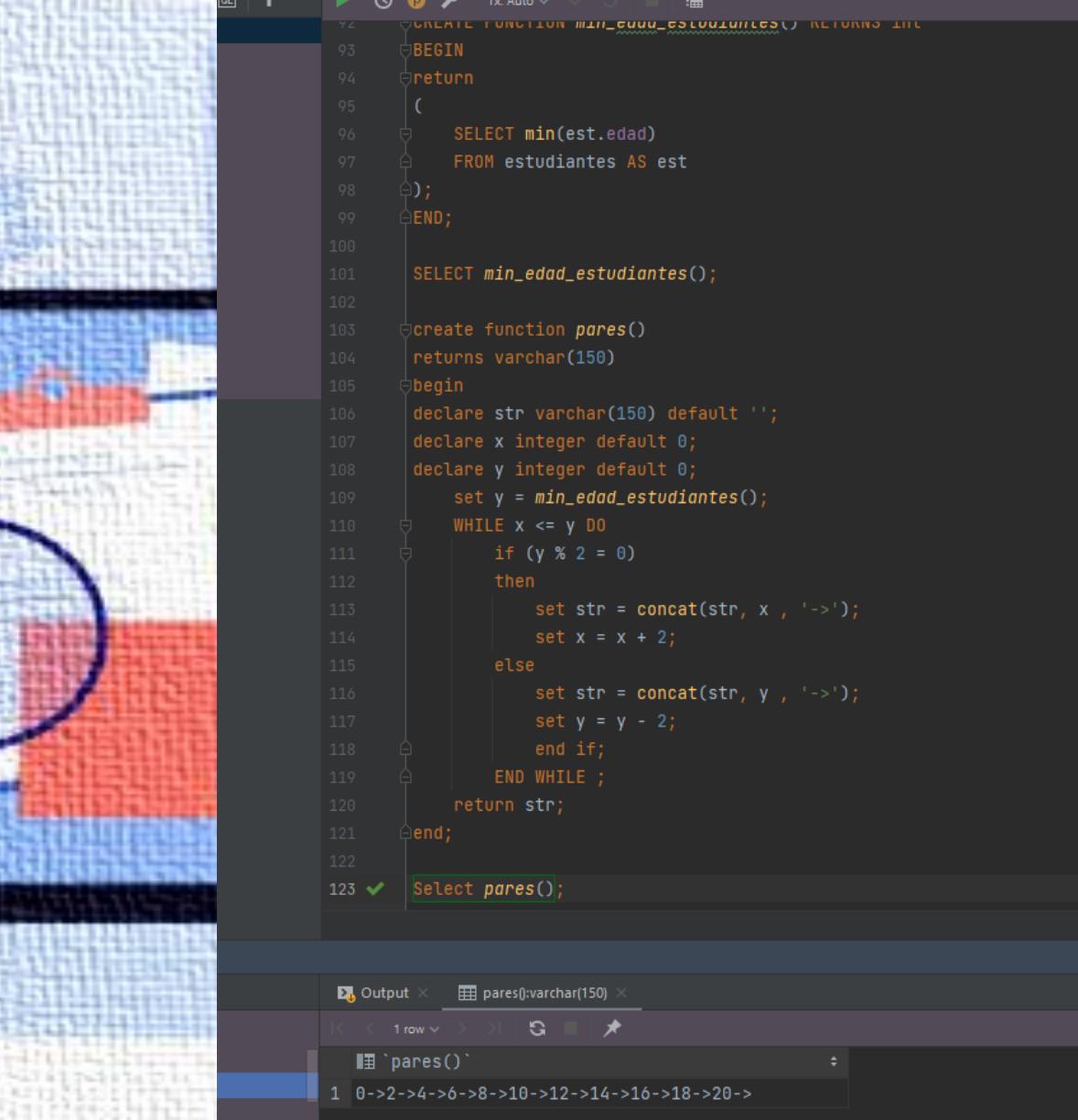
14.Crear una función no recibe parámetros (Utilizar WHILE, REPEAT o LOOP).

```
CREATE FUNCTION min_edad_estudiantes() RETURNS int
BEGIN
return
(
    SELECT min(est.edad)
    FROM estudiantes AS est
);
END;

SELECT min_edad_estudiantes();

create function pares()
returns varchar(150)
begin
declare str varchar(150) default '';
declare x integer default 0;
declare y integer default 0;
set y = min_edad_estudiantes();
WHILE x <= y DO
if (y % 2 = 0)
then
    set str = concat(str, x , '->');
    set x = x + 2;
else
    set str = concat(str, y , '->');
    set y = y - 2;
end if;
END WHILE ;
return str;
end;

Select pares();
```



```
CREATE FUNCTION min_edad_estudiantes() RETURNS int
BEGIN
return
(
    SELECT min(est.edad)
    FROM estudiantes AS est
);
END;

SELECT min_edad_estudiantes();

create function pares()
returns varchar(150)
begin
declare str varchar(150) default '';
declare x integer default 0;
declare y integer default 0;
set y = min_edad_estudiantes();
WHILE x <= y DO
if (y % 2 = 0)
then
    set str = concat(str, x , '->');
    set x = x + 2;
else
    set str = concat(str, y , '->');
    set y = y - 2;
end if;
END WHILE ;
return str;
end;

Select pares();
```

The screenshot shows the MySQL Workbench interface with the SQL editor open. The code for the 'pares()' function is displayed, which generates a string of even numbers from 0 to 20. The output tab shows the result of executing the 'Select pares();' statement, which is '0->2->4->6->8->10->12->14->16->18->20->'. A magnifying glass icon is visible in the background.

15. Crear una función que determina cuantas veces se repite las vocales.

```
create function vocales(par1 text)
returns text
begin
declare x int default 1;
declare aVeces int default 0;
declare eVeces int default 0;
declare iVeces int default 0;
declare oVeces int default 0;
declare uVeces int default 0;
declare response text default '';
declare letra char default '';
declare limite int default char_length(par1);
declare a varchar(5) default 'a';
declare e varchar(5) default 'e';
declare i varchar(5) default 'i';
declare o varchar(5) default 'o';
declare u varchar(5) default 'u';
```

```
while x <= limite do
    set letra = substring(par1, x, 1);
    if letra = a
    then
        set aVeces = aVeces +1;
    else if letra = e
    then
        set eVeces = eVeces +1;
        else if letra = i
    then
        set iVeces = iVeces +1;
        else if letra = o
    then
        set oVeces = oVeces +1;
        else if letra = u
    then
        set uVeces = uVeces +1;
    end if;
    end if;
    end if;
    end if;
    set x = x + 1;
end while;
set response = concat(a, ':', aVeces, ', ', e, ':', eVeces, ', ', i, ':', iVeces, ', ',
,o, ':', oVeces, ', ', u, ':', uVeces);
return response;
end;

select vocales('taller de base de datos');
```

The screenshot shows the MySQL Workbench interface with the following details:

- Code Editor:** Shows the stored procedure code with line numbers from 151 to 173. Lines 151-170 contain the main logic, and line 172 shows the call to the function.
- Output Window:** Shows the result of executing the function: "vocales('taller de base de datos')". The output is:

```
1 a:3,e:4,i:0,o:1,u:0
```

17. Crear una función que reciba un parámetro TEXT

```
create function paramettex(cadena varchar(20), position integer)
returns text
begin
declare subCadena text default '';
declare limite int default char_length(cadena);
set position = 1;
repeat
    set subCadena = concat(subCadena,substring(cadena , position,
limite),',');
    set position = position + 1;
until position -1 = limite END REPEAT;
return subCadena;
end;

select paramettex('dbaii', 1);
```

```
170 end;
171
172 select vocales( par1: 'taller de base de datos');
173
174 create function paramettex(cadena varchar(20), position integer)
175 returns text
176 begin
177 declare subCadena text default '';
178 declare limite int default char_length(cadena);
179 set position = 1;
180 repeat
181
182     set subCadena = concat(subCadena,substring(cadena , position, limite),',');
183     set position = position + 1;
184
185 until position -1 = limite END REPEAT;
186
187 return subCadena;
188 end;
189
190 select paramettex( cadena: 'dbaii', position: 1);
```

Output × paramettex('dbaii', 1):text ×

paramettex('dbaii', 1)
1 dbaii,baii,aii,ii,i,

¡GRACIAS POR SU ATENCIÓN!



<https://drive.google.com/file/d/1EUI0Hb9beySL2GW462k0Use7593HYnLH/view?usp=sharing>