



Tarea HITO3

09.04.20XX

**─**

Base de Datos II

Unifranz Sede el Alto Hito3

**Objetivos de la tarea.**

1. Mostrar el manejo de **programación a nivel de Base de Datos**.
2. Mostrar el manejo del **lenguaje procedural.**
3. Mostrar el manejo de **funciones en MySQL**.

# Consigna.

Esta tarea se divide en dos partes:

* La primera parte corresponde a la **parte TEÓRICA necesaria**, en donde se encuentra un conglomerado de preguntas relacionadas a BASES DE DATOS RELACIONALES.
* La segunda parte corresponde a la **parte PRÁCTICA necesaria** en donde deberá realizar y crear funciones manejando el motor de base de datos MySQL manejando el concepto del lenguaje procedural.

# Consideraciones sobre la entrega.

La tarea es abierta, usted decide cómo entregar su tarea. Puede agregar imágenes, enlaces, etc. Sin embargo, considerar los siguientes enunciados.

* Deberá generar una presentación hecha en **POWER POINT**. Después este mismo archivo deberá ser convertido a un **archivo PDF**.
* Después de tener disponible la presentación, debe de **GENERAR UN VIDEO**

explicando todo su contenido o lo que considere necesario e importante.

* 1. El video debe de tener como mínimo 15 minutos.
  2. En el video debe estar presente su cámara web.
     + Si no tiene cámara web puede apoyarse en [IRIUM](https://iriun.com/) que básicamente convierte su celular en una cámara web.
* Después de tener disponible el video, el video lo puede subir a **youtube, vimeo** o a drive u otra cualquier plataforma.
* Finalmente, todo lo generado, es decir:

1. El archivo **ppt** (powerpoint).
2. El archivo **pdf** (el powerpoint convertido a PDF).
3. El **video** subido a alguna plataforma.
4. IMPORTANTE:
   * Tiene que ser subido a la plataforma **GITHUB**.
   * Para hacer este proceso tiene que tener sus archivos ya disponibles.
   * Crear la carpet **HITO3/PROCESUA**L en nuestro repositorio de github.

* En la plataforma **MOODLE** solo deberá de subir una carátula referenciando a su tarea que se encuentra en github.

**Ejemplo**:

**UNIVERSIDAD PRIVADA FRANZ TAMAYO DEFENSA HITO 3 - TAREA FINAL**

**Estudiante:** Univ. Nombres Apellidos **Asignatura:** BASE DE DATOS II **Carrera:** INGENIERÍA DE SISTEMAS

**Sede**: El Alto

**Paralelo:** BDA (1)

**Docente:** Lic. William Barra Paredes

**fecha:** xx/xx/2022

**GITHUB**: <https://github.com/dheeyi/base-de-datos-i>i (aqui va el enlace a su cuenta de github)

**A partir de este punto están las preguntas TEÓRICAS y PRÁCTICAS a resolverse.**

**Manejo de conceptos.**

1. Deﬁna que es lenguaje procedural en MySQL.

R.- ¿Qué son los parámetros en MySQL?

Los parámetros de los procedimientos almacenados de MySQL

1. Deﬁna que es una función en MySQL.
2. ¿Qué cosas características debe de tener una función? Explique sobre el nombre, el return, parametros, etc.
3. ¿Cómo crear, modiﬁcar y cómo eliminar una función? Adjunte un ejemplo de su uso.
4. Para qué sirve la funcion CONCAT y como funciona en MYSQL
   * ¿Crear una función que muestre el uso de las función CONCAT?
   * La función debe concatenar 3 cadenas.
5. 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
6. Para qué sirve la funcion STRCMP y como funciona en MYSQL

### ¿Crear una función que muestre el uso de las función STRCMP?

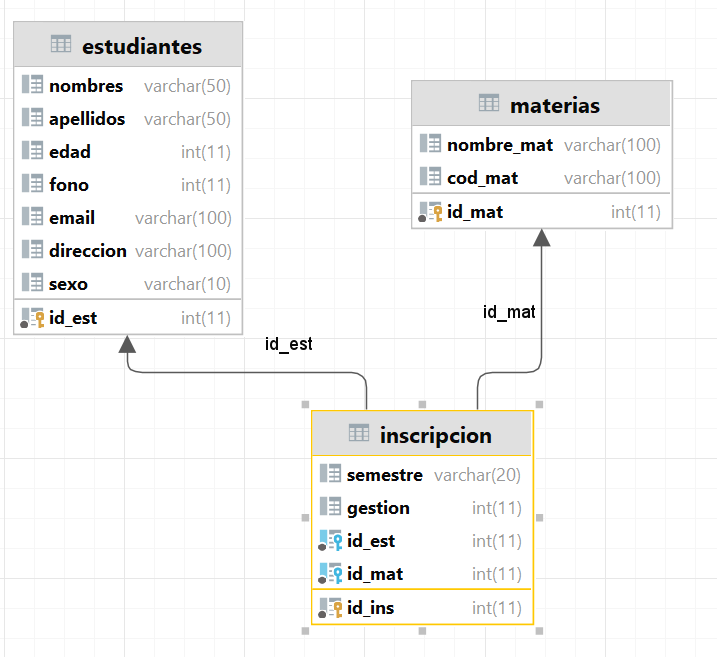
* + La función debe comparar 3 cadenas. Y deberá determinar si dos de ellas son iguales.

1. Para qué sirve la función CHAR\_LENGTH y LOCATE y como funciona en MYSQL
   * ¿Crear una función que muestre el uso de ambas funciones?
2. ¿Cual es la diferencia entre las funciones de agresión y funciones creados por el DBA? Es decir funciones creadas por el usuario.
3. ¿Busque y deﬁna a qué se referirá cuando se habla de parámetros de entrada y salida en MySQL?

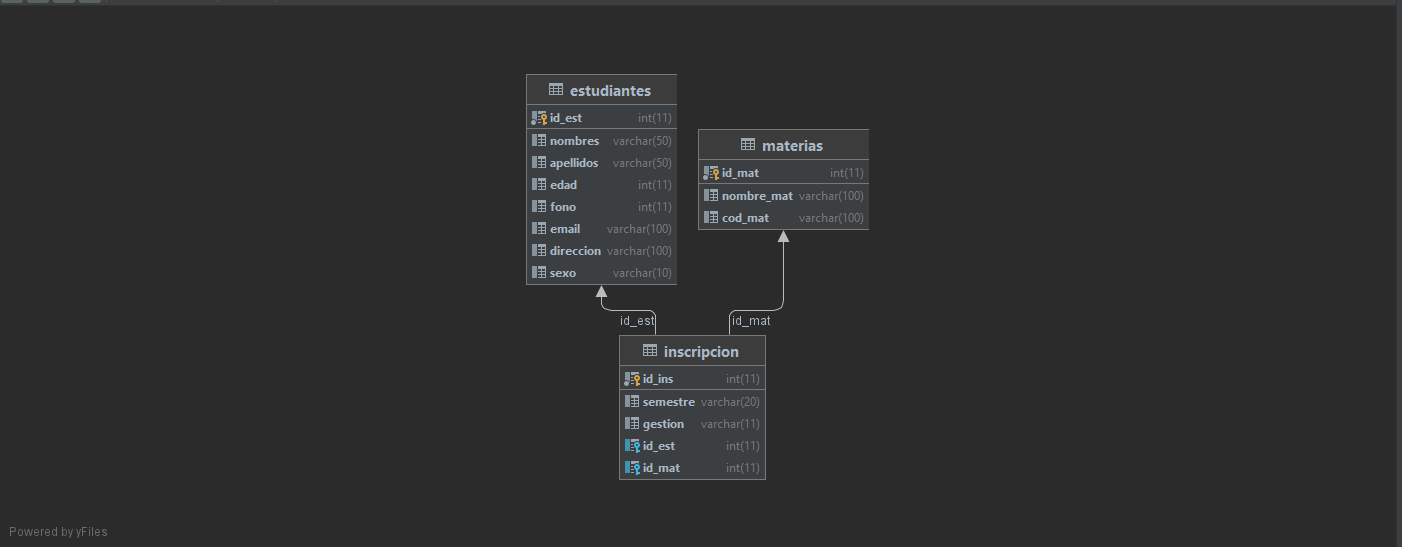
* **Es decir IN INOUT, etc**

**Parte practica**

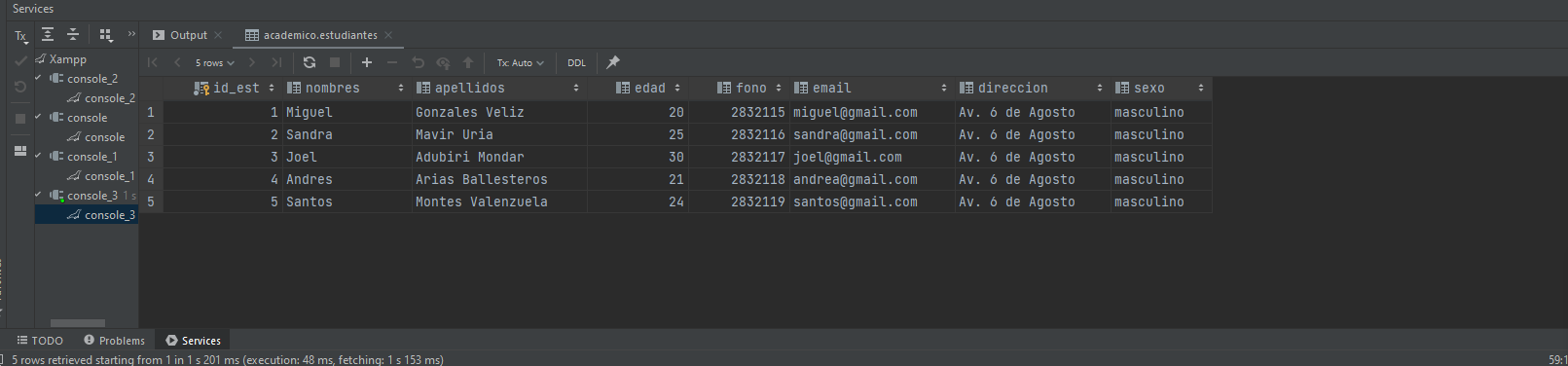
1. Crear la siguiente Base de datos y sus registros.

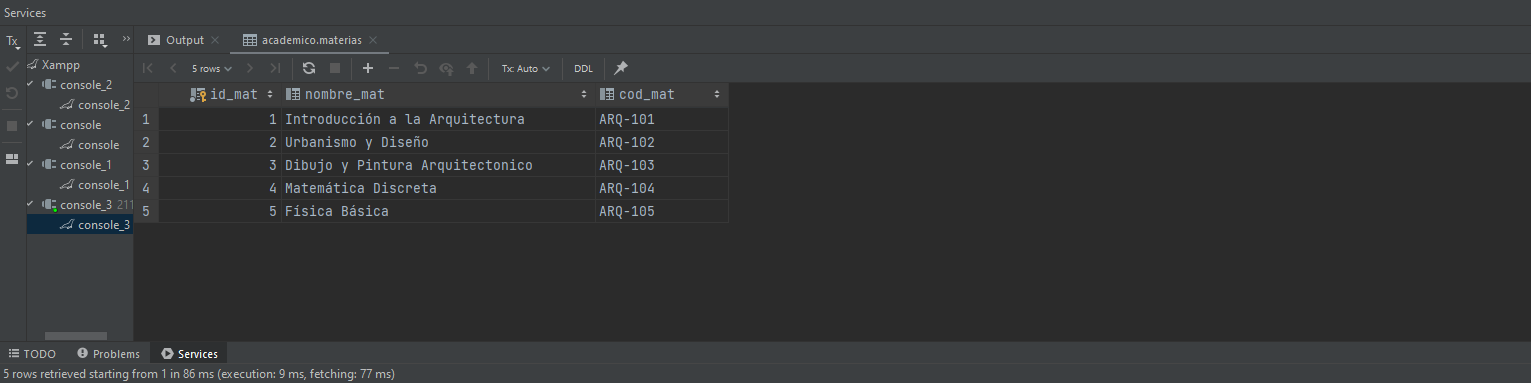


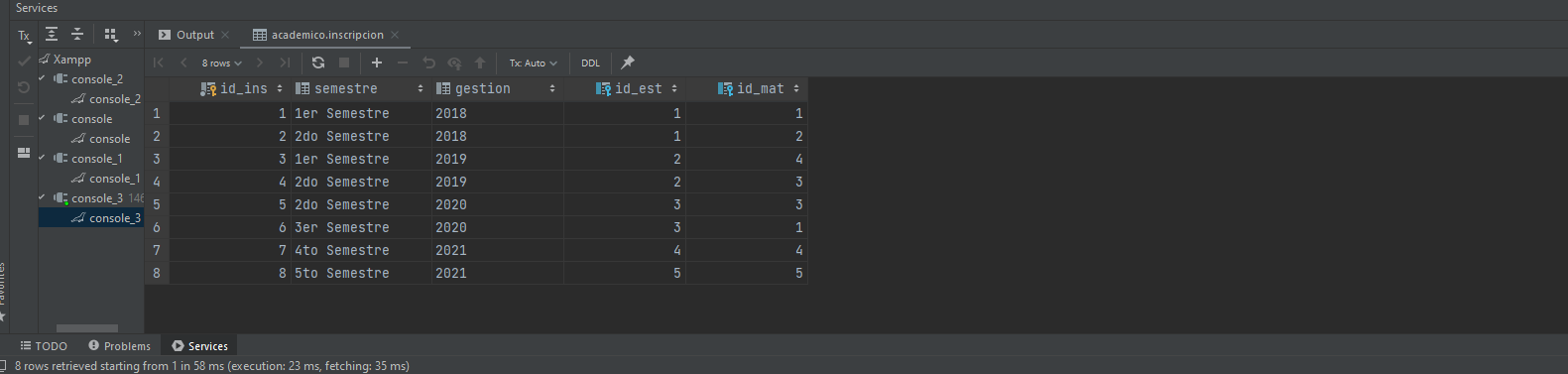
create table estudiantes(  
 id\_est INT(11) NOT NULL AUTO\_INCREMENT,  
 nombres VARCHAR(50),  
 apellidos VARCHAR(50),  
 edad INT(11),  
 fono INT(11),  
 email VARCHAR(100),  
 direccion VARCHAR(100),  
 sexo VARCHAR(10),  
 PRIMARY KEY (id\_est)  
);  
create table materias(  
 id\_mat INT(11) NOT NULL AUTO\_INCREMENT,  
 nombre\_mat VARCHAR(100),  
 cod\_mat VARCHAR(100),  
 PRIMARY KEY (id\_mat)  
);  
create table inscripcion(  
 id\_ins INT(11) NOT NULL AUTO\_INCREMENT,  
 semestre VARCHAR(20),  
 gestion VARCHAR(11),  
 id\_est INT(11),  
 id\_mat INT(11),  
 PRIMARY KEY (id\_ins),  
 FOREIGN KEY (id\_est) REFERENCES estudiantes(id\_est),  
 FOREIGN KEY (id\_mat) REFERENCES materias(id\_mat)  
);



|  |
| --- |
| DATOS TABLA ESTUDIANTES |
| DATOS TABLA MATERIAS |
| DATOS TABLA INSCRIPCION |



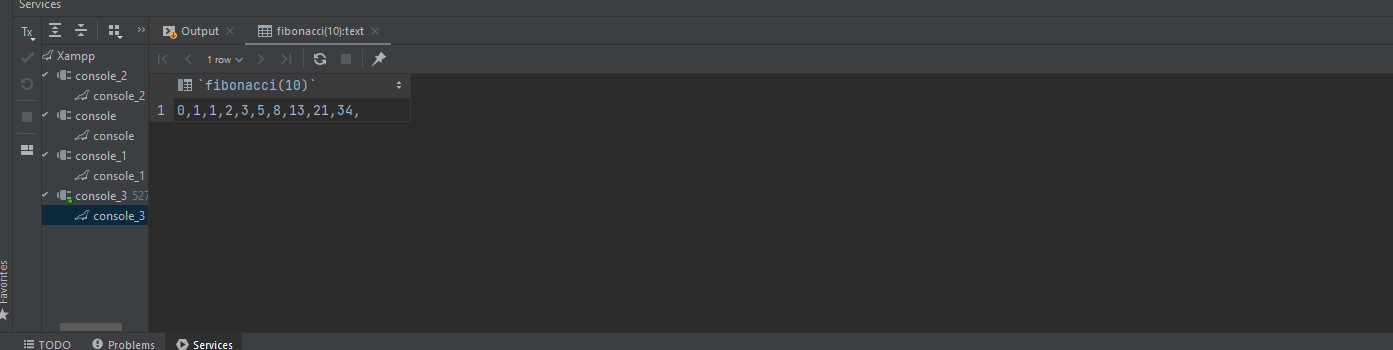




INSERT INTO estudiantes(nombres,apellidos,edad,fono,email,direccion,sexo)  
VALUES('Miguel','Gonzales Veliz', 20, 2832115, 'miguel@gmail.com', 'Av. 6 de Agosto', 'masculino'),  
('Sandra','Mavir Uria', 25, 2832116, 'sandra@gmail.com', 'Av. 6 de Agosto', 'masculino'),  
('Joel','Adubiri Mondar', 30, 2832117, 'joel@gmail.com', 'Av. 6 de Agosto', 'masculino'),  
('Andres','Arias Ballesteros', 21, 2832118, 'andrea@gmail.com', 'Av. 6 de Agosto', 'masculino'),  
('Santos','Montes Valenzuela', 24, 2832119, 'santos@gmail.com', 'Av. 6 de Agosto', 'masculino');  
#MATERIAS  
INSERT INTO materias(nombre\_mat,cod\_mat)  
VALUES('Introducción a la Arquitectura', 'ARQ-101'),  
('Urbanismo y Diseño', 'ARQ-102'),  
('Dibujo y Pintura Arquitectonico', 'ARQ-103'),  
('Matemática Discreta', 'ARQ-104'),  
('Física Básica', 'ARQ-105');  
#INSCRIPCION  
INSERT INTO inscripcion(semestre, gestion, id\_est, id\_mat)  
VALUES('1er Semestre', 2018, 1, 1),  
('2do Semestre', 2018, 1, 2),  
('1er Semestre', 2019, 2, 4),  
('2do Semestre', 2019, 2, 3),  
('2do Semestre', 2020, 3, 3),  
('3er Semestre', 2020, 3, 1),  
('4to Semestre', 2021, 4, 4),  
('5to Semestre', 2021, 5, 5);

1. Crear una función que genere la serie Fibonacci.
   * La función recibe un límite(number)
   * La función debe de retornar una cadena.
   * Ejemplo para **n=7. OUTPUT: 0, 1, 1, 2, 3, 5, 8,**
   * Adjuntar el **código SQL generado y una imagen de su correcto funcionamiento.**

create function *fibonacci*(limite int)  
 returns text  
 begin  
 declare a int default 0;  
 declare b int default 1;  
 declare c int default 0;  
 declare x int default 1;  
 declare response text;  
  
 if limite >= 1  
 then  
 set response = concat(a, ',');  
 end if;  
  
 if limite >= 2  
 then  
 set response = concat(response, b, ',');  
 end if;  
  
 if limite >= 3  
 then  
 while x <= (limite - 2) do  
 set c=a+b;  
 set response = concat(response, c, ',');  
 set a = b;  
 set b = c;  
 set x = x+1;  
 end while;  
 end if;  
 return response;  
 end;

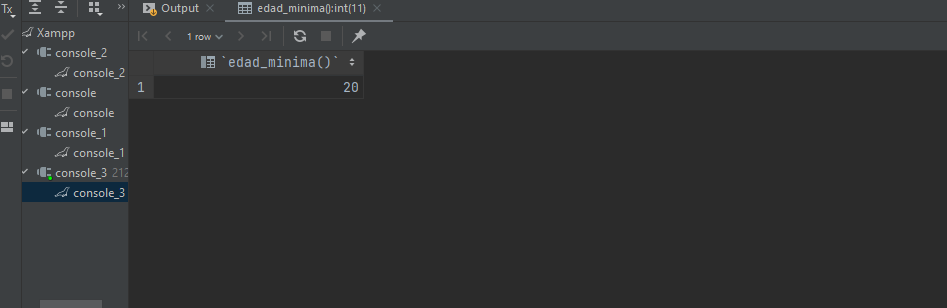


1. Crear una variable global a nivel BASE DE DATOS.
   * Crear una función cualquiera.
   * La función debe retornar la variable global.
   * Adjuntar el código **SQL generado y una imagen de su correcto funcionamiento.**

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

* + Previamente deberá de crear una función que obtenga la edad mínima de los estudiantes
    - La función no recibe ningún parámetro.
    - La función debe de retornar un número.(LA EDAD MÍNIMA).

create function *edad\_minima*()  
returns int  
 begin  
 declare response int default 0;  
 set response = (select min(est.edad)  
 from estudiantes as est);  
 return response;  
 end;



* + Si la edad mínima es **PAR** mostrar todos los pares empezando desde 0 a este ese valor de la edad mínima.

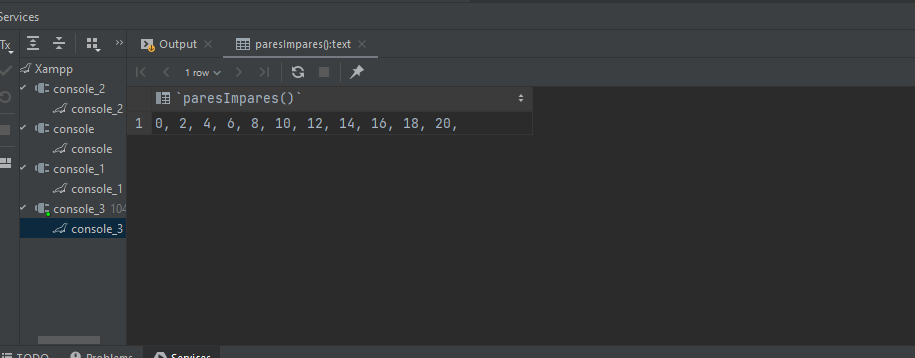


* + Si la edad mínima es **IMPAR** mostrar descendentemente todos los impares hasta el valor 0.



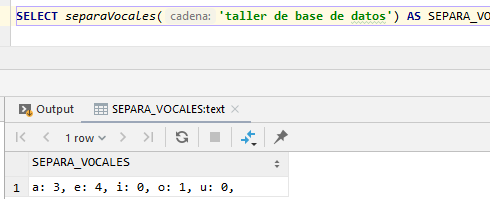
* + Retornar la nueva cadena concatenada.
  + Adjuntar el **código SQL generado y una imagen de su correcto funcionamiento.**
* ***Nota:*** *Esta función está llamando a otra función, considere eso.*

create function *paresImpares*()  
returns text  
 begin  
 declare age int default 0;  
 declare count int default 0;  
 declare response text default '';  
  
 set age = *edad\_minima*();  
 if (age % 2 = 0)  
 then  
 while (count <= age) do  
 # PAR  
 if(count % 2 = 0)  
 then  
 set response = concat(response, count, ', ');  
 end if;  
 set count = count + 1;  
 end while;  
 else  
 set count = age;  
 while (count >= 0) do  
 # IMPAR  
 if(count % 2 = 1)  
 then  
 set response = concat(response, count, ', ');  
 end if;  
 set count = count - 1;  
 end while;  
 end if;  
 return response;  
 end;



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

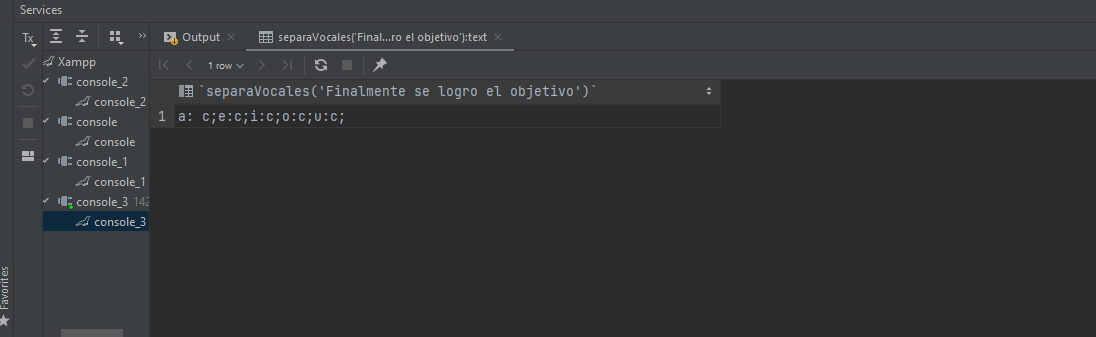
* + La función recibe una cadena y retorna un TEXT.
  + Retornar todas las vocales ordenadas e indicando la cantidad de veces que se repite en la cadena.
  + Resultado esperado.



○

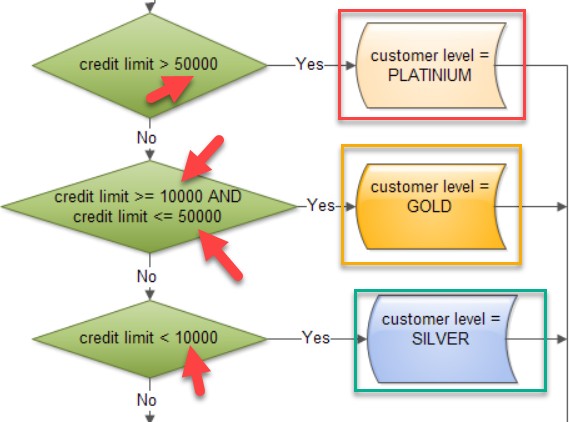
* + Adjuntar el **código SQL generado y una imagen de su correcto**

**funcionamiento.**



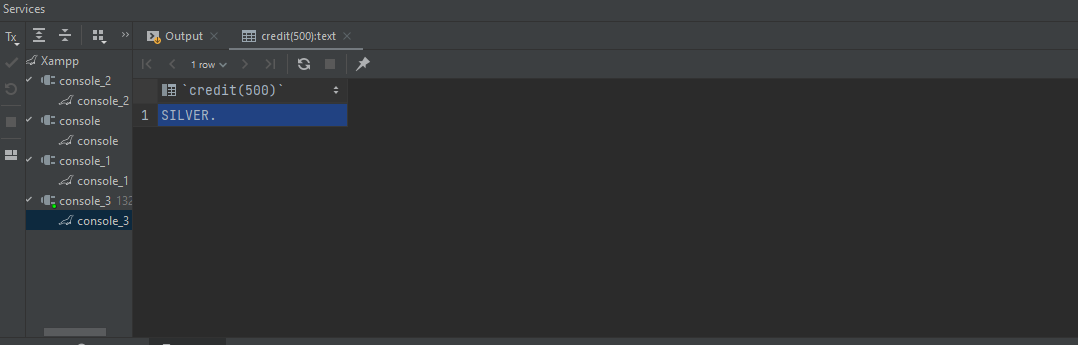
create function *separaVocales*(cadena text)  
returns text  
 begin  
 declare c int default 0;  
 declare count int default 1;  
 declare response text default '';  
 # VOCAL: A  
 while (count <= char\_length(cadena)) do  
 if('a' = SUBSTR(cadena, count, 1) || 'A' = SUBSTR(cadena, count, 1))  
 then  
 set c = c + 1;  
 end if;  
 set count = count + 1;  
 end while;  
 set response = concat(response, 'a: ', 'c',';');  
 set c = 0;  
 set count = 1;  
 # VOCAL: E  
 while (count <= char\_length(cadena)) do  
 if('e' = SUBSTR(cadena, count, 1) || 'E' = SUBSTR(cadena, count, 1))  
 then  
 set c = c + 1;  
 end if;  
 set count = count + 1;  
 end while;  
 set response = concat(response, 'e:', 'c',';');  
 set c = 0;  
 set count = 1;  
 # VOCAL: I  
 while (count <= char\_length(cadena)) do  
 if('i' = SUBSTR(cadena, count, 1) || 'I' = SUBSTR(cadena, count, 1))  
 then  
 set c = c + 1;  
 end if;  
 set count = count + 1;  
 end while;  
 set response = concat(response, 'i:', 'c',';');  
 set c = 0;  
 set count = 1;  
 # VOCAL: O  
 while (count <= char\_length(cadena)) do  
 if('o' = SUBSTR(cadena, count, 1) || 'O' = SUBSTR(cadena, count, 1))  
 then  
 set c = c + 1;  
 end if;  
 set count = count + 1;  
 end while;  
 set response = concat(response, 'o:' , 'c',';' );  
 set c = 0;  
 set count = 1;  
 # VOCAL: O  
 while (count <= char\_length(cadena)) do  
 if('u' = SUBSTR(cadena, count, 1) || 'U' = SUBSTR(cadena, count, 1))  
 then  
 set c = c + 1;  
 end if;  
 set count = count + 1;  
 end while;  
 set response = concat(response, 'u:' , 'c',';');  
  
 return response;  
 end;

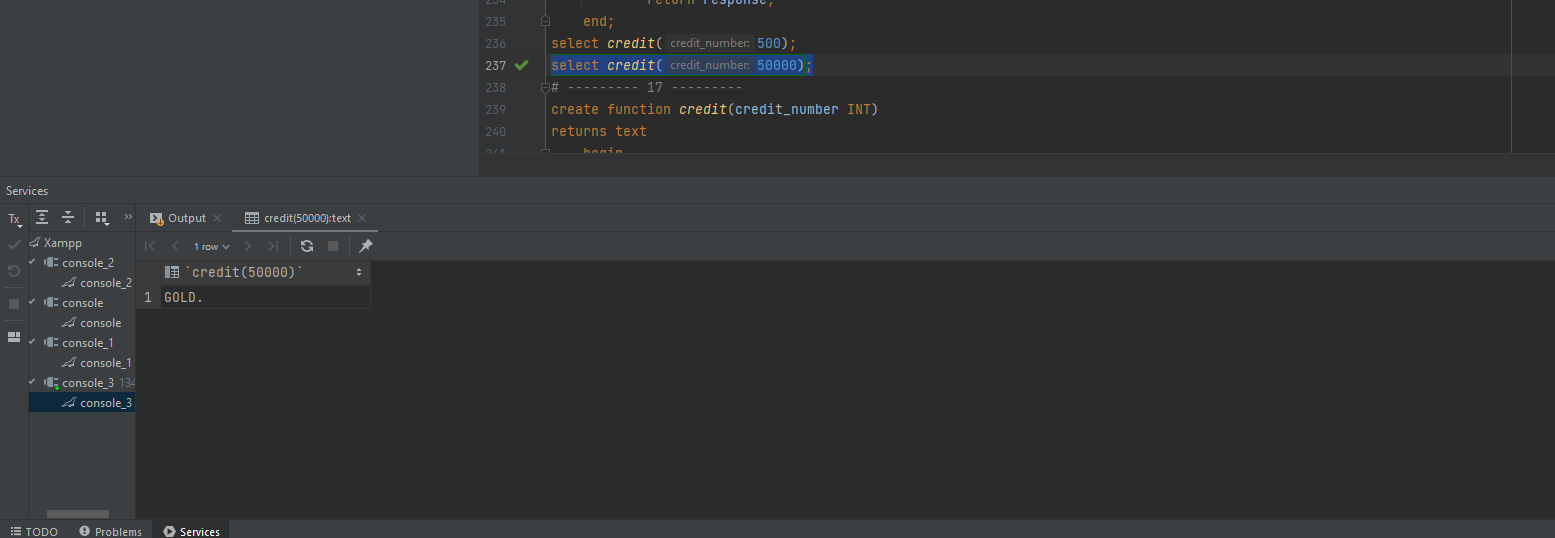
1. Crear una función que recibe un parámetro INTEGER.
   * La función debe de retornar un texto(**TEXT**) como respuesta.
   * El parámetro es un valor numérico **credit\_number**.
   * Si es mayor a 50000 es **PLATINIUM**.
   * Si es mayor igual a 10000 y menor igual a 50000 es **GOLD**.
   * Si es menor a 10000 es **SILVER**
   * La función debe retornar indicando si ese cliente es PLATINUM, GOLD o SILVER en base al valor del credit\_number.
   * Considere la imagen siguiente:



* + Para resolver debe de utilizar la instrucción **CASE - WHEN**.
  + Adjuntar el **código SQL generado y una imagen de su correcto funcionamiento.**

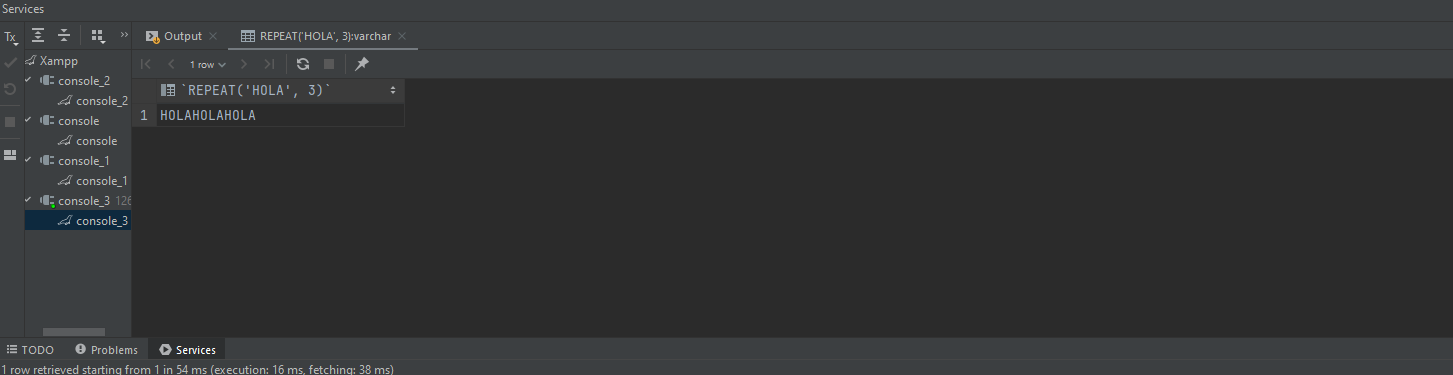
create function *credit*(credit\_number INT)  
returns text  
 begin  
 declare response text default '';  
 set response = (select CASE  
 WHEN credit\_number > 50000 THEN 'PLATINIUM.'  
 WHEN credit\_number >= 10000 && credit\_number <= 50000 THEN 'GOLD.'  
 WHEN credit\_number < 10000 THEN 'SILVER.'  
 END);  
 return response;  
 end;  
select *credit*(500);





1. Crear una función que reciba un parámetro TEXT
   * En donde este parámetro deberá de recibir una **cadena** cualquiera y retorna un **TEXT** de respuesta.

select REPEAT('HOLA', 3);



* + Concatenar **N** veces la misma cadena reduciendo en uno en cada iteración hasta llegar a una sola letra.
  + Utilizar REPEAT y retornar la nueva cadena concatenada.
  + Considerar la siguiente imagen:



* + Adjuntar el **código SQL generado y una imagen de su correcto funcionamiento.**

create function *cadenaRepeat*(cadena TEXT)  
returns text  
 begin  
 declare response text default '';  
 declare newChart text default '';  
 declare count int default 0;  
  
 set newChart = cadena;  
  
 REPEAT  
 SET response = CONCAT(response, newChart,', ');  
 SET count = count + 1;  
 SET newChart = SUBSTR(newChart, 2);  
 UNTIL count >= char\_length(cadena)  
 END REPEAT;  
  
 return response;  
 end;

