**Bases de datos II**

**Actividad 3**

**MySQL**

Jonny Luna Guerrero

Edwin Marrugo Rodríguez

Alberto Valencia Solis

Universidad de Cartagena

Facultad de Ingeniería

Ingeniería de Software

Bases de Datos II

Profesor Mg. John Arrieta Arrieta

Semestre V

Cartagena, Junio 29 de 2023

# Tabla de Contenido

[1 Tabla de Contenido 2](#_Toc139478257)

[2 Introducción 3](#_Toc139478258)

[3 Objetivos 4](#_Toc139478259)

[3.1 Objetivos Generales: 4](#_Toc139478260)

[3.2 Objetivos específicos 4](#_Toc139478261)

[4 Justificación 6](#_Toc139478262)

[5 Desarrollo 8](#_Toc139478263)

[5.1 Qué es y cuál es el propósito de una Función en SQL 8](#_Toc139478264)

[5.2 Qué es y cual es el propósito de un Procedimiento almacenado en SQL 8](#_Toc139478265)

[5.3 Cuáles son las diferencias más representativas entre Función y Procedimiento almacenado. 9](#_Toc139478266)

[5.4 Investigue cuales son las funciones predefinidas en OracleSQL, PostgreSQL y SQLServer que nos permiten realizar operaciones sobre: 9](#_Toc139478267)

[5.5 Qué es y cuál es el objetivo de una Función personalizada o definida por el usuario 12](#_Toc139478268)

[5.6 Cual es la sintaxis general para crear e invocar una Función definida por el usuario. 13](#_Toc139478269)

[5.7 Crear una función que reciba como parámetros de entrada la cc y la clave de un deudor, entonces la función debe retornar Verdadero si existe un usuario con esa misma CC y Clave registrados en esa tabla, o retornar FALSO en caso contrario. 13](#_Toc139478270)

[5.8 Invocar y probar la función con datos que permitan mostrar resultado Falso y otros que permitan retornar resultado Verdadero. 14](#_Toc139478271)

[5.9 Crear una función que reciba como parámetros la cédula de un deudor y devuelva un solo dato con el nombre del deudor, valor del último, fecha y estado del último crédito que tiene en el sistema. El resultado debe ser una cadena de texto en el siguiente formato: por ejemplo. NOMBRE: FULANITO DE TAL, FECHA: 12-04-2022, VALOR: $100000, ESTADO: ACTIVO 15](#_Toc139478272)

[5.10 Invocar y probar la función 17](#_Toc139478273)

[5.11 Crear una que reciba 2 números y revuelva la sumatoria acumulativa entre ellos. 18](#_Toc139478274)

[5.12 Crear un procedimiento almacenado con las siguientes características: 20](#_Toc139478275)

[5.13 Invocar y probar el procedimiento almacenado. 21](#_Toc139478276)

[6 Bibliografía 23](#_Toc139478277)

# Introducción

En este trabajo, continuamos profundizando en el DML, y exploraremos temas avanzados relacionados con las bases de datos y el lenguaje SQL; En primer lugar, discutiremos las funciones en SQL y cómo se pueden utilizar para realizar cálculos y manipulaciones de datos.

Luego, examinaremos los procedimientos almacenados en SQL y cómo se diferencian de las funciones y haremos un paralelo entre estas dos herramientas del lenguaje SQL; de las funciones haremos una diferenciación entre aquellas definidas por el lenguaje(predefinidas) y las definidas por nosotros mismos (definidas por el usuario). Realizaremos ejemplos que nos ayudaran a poner en práctica el uso y observar su poder y utilidad con varios ejemplos sencillos. Con la práctica y la apropiación de conceptos reflexionaremos sobre la importancia de aprender estos temas para un ingeniero de software. Al final de este trabajo pretendemos adquirir una comprensión más a fondo de estos conceptos y herramientas, dominarlos y saber cómo aplicarlos en el mundo real.

.

# Objetivos

## Objetivos Generales:

* Comprender el concepto y la utilidad de las funciones en SQL.
* Distinguir entre procedimientos almacenados y funciones en SQL.
* Explorar las diferencias entre las funciones predefinidas y las funciones definidas por el usuario.
* Adquirir habilidades prácticas en el uso de funciones y procedimientos almacenados en SQL.
* Reconocer la importancia de aprender y dominar estos temas para un ingeniero de software.

## Objetivos específicos

* Comprender cómo se utilizan las funciones en SQL para realizar cálculos y manipulaciones de datos.
* Conocer las características y ventajas de los procedimientos almacenados en SQL.
* Identificar las diferencias clave entre las funciones y los procedimientos almacenados en SQL.
* Explorar las funciones predefinidas disponibles en SQL y comprender cómo utilizarlas en consultas.
* Aprender a crear y utilizar funciones definidas por el usuario para resolver problemas específicos.
* Practicar con ejemplos sencillos para familiarizarse con el uso de funciones y procedimientos almacenados.
* Reflexionar sobre la importancia de estos temas en el contexto del desarrollo de software.
* Reconocer las ventajas y desventajas de utilizar funciones y procedimientos almacenados en diferentes situaciones.
* Obtener una comprensión profunda de los conceptos y herramientas relacionados con las funciones y procedimientos almacenados en SQL.
* Aplicar de manera efectiva las funciones y procedimientos almacenados en el desarrollo de aplicaciones de bases de datos en el mundo real.

# Justificación

El conocimiento de estos conceptos avanzados de Base de Datos juega un papel fundamental en nuestra formación como profesionales de la ingeniería de software, ya que proporciona los conocimientos necesarios para diseñar, implementar y administrar sistemas de bases de datos eficientes y robustos.

Dentro de este contexto, es crucial dominar temas avanzados relacionados con las funciones y los procedimientos almacenados en SQL. La importancia de dominar estos temas radica en la capacidad que adquirimos para aprovechar al máximo el potencial de las bases de datos y mejorar la eficiencia y la calidad de las aplicaciones que desarrollemos; algunas razones que justifican la importancia de adquirir estos conocimientos y habilidades en estos temas son, entre otros son:

Manipulación avanzada de datos:

Las funciones en SQL permiten realizar cálculos, transformaciones y manipulaciones complejas de datos en tiempo real.

Al dominar el concepto de funciones, podemos realizar consultas sofisticadas y obtener información precisa y relevante sobre los datos almacenados.

Optimización del rendimiento de la base de datos: Los procedimientos almacenados en SQL ofrecen la posibilidad de encapsular secuencias de instrucciones SQL en un solo bloque, lo que permite un acceso más rápido y eficiente a los datos.

Al comprender y utilizar adecuadamente los procedimientos almacenados, podemos mejorar significativamente el rendimiento de las aplicaciones al minimizar la carga de la red y reducir la latencia de la comunicación con la base de datos.

Reutilización de código: Tanto las funciones predefinidas como las definidas por el usuario promueven la reutilización de código en el desarrollo de aplicaciones. Al utilizar funciones predefinidas, aprovechamos la funcionalidad incorporada en el sistema de gestión de bases de datos, evitando así la necesidad de desarrollar funciones personalizadas desde cero.

De la misma manera, al crear funciones definidas por el usuario, se puede encapsular la lógica de negocio específica de la aplicación para su uso repetido, lo que agiliza el desarrollo y mejora el mantenimiento del código.

Escalabilidad y mantenibilidad: La utilización adecuada de funciones y procedimientos almacenados contribuye a la escalabilidad y mantenibilidad de las aplicaciones. Al dividir la lógica de negocio en unidades funcionales coherentes, se facilita la gestión y la evolución del sistema a medida que crece en tamaño y complejidad.

Además, el modularidad proporcionado por las funciones y los procedimientos almacenados simplifica las tareas de prueba, depuración y actualización del código, resultando un código más fácil de mantener.

Habilidad profesional: En el entorno altamente competitivo de la ingeniería de software, el dominio de las funciones y los procedimientos almacenados en SQL es una habilidad que necesariamente debemos tener para tener una ventaja competitiva al buscar empleo.

# Desarrollo

Para el desarrollo de esta actividad académica, usted como estudiante del programa de Ingeniería de Software en modalidad a distancia de la Universidad de Cartagena debe desarrollar, presentar y sustentar en tutoría los siguientes puntos, de acuerdo a lo estipulado en tutorías sobre lo concerniente a las actividades académicas evaluativas, a continuación se describe el desarrollo de la primera actividad, donde ustedes como equipo deben tomar como base el mismo ejercicio desarrollado en B-1 (el caso práctico elegido para desarrollar las actividades propuestas en la asignatura BD-1), continuar con el desarrollo de la temática de la asignatura BD-2, por lo que para esta primera actividad (Unidad 1) deben resolver los siguiente puntos sobre Funciones, Procedimientos Almacenados y su aplicación en los motores de bases de datos relacionales OracleSQL, PostgreSQL y SQLServer.

## Qué es y cuál es el propósito de una Función en SQL

Las funciones en SQL son bloques de código que realizan cálculos, manipulaciones y operaciones sobre los datos almacenados en una base de datos. Estas funciones pueden recibir parámetros de entrada y devolver un valor o un conjunto de valores como resultado. Las funciones en SQL se utilizan para realizar operaciones repetitivas, simplificar consultas y encapsular la lógica de cálculos complejos. Son una forma de modularidad que permite reutilizar código y facilita el desarrollo, mantenimiento y optimización de las aplicaciones.

## Qué es y cual es el propósito de un Procedimiento almacenado en SQL

Un procedimiento almacenado en SQL es un conjunto de instrucciones SQL predefinidas que se almacenan en la base de datos y se pueden llamar y ejecutar cuando sea necesario. Los procedimientos almacenados se utilizan para realizar tareas específicas y complejas en la base de datos, como manipulación de datos, validaciones y operaciones de negocio. A diferencia de las funciones, los procedimientos almacenados no necesariamente devuelven un valor, pero pueden modificar datos, generar resultados temporales o realizar acciones específicas en el sistema. Los procedimientos almacenados también promueven la modularidad y la reutilización de código, lo que facilita el mantenimiento y la escalabilidad del sistema.

## Cuáles son las diferencias más representativas entre Función y Procedimiento almacenado.

La principal diferencia entre una función y un procedimiento almacenado en SQL radica en su uso y resultado. Una función devuelve un valor o un conjunto de valores como resultado, mientras que un procedimiento almacenado no necesariamente devuelve un valor, sino que puede realizar acciones y modificaciones en la base de datos. Las funciones se utilizan principalmente en consultas y cálculos de datos, mientras que los procedimientos almacenados se utilizan para realizar tareas más complejas y acciones específicas en la base de datos; las funciones se centran en los cálculos y la manipulación de datos, mientras que los procedimientos almacenados se enfocan en realizar tareas y acciones específicas sobre la base de datos.

Además de lo anterior las funciones puede ser funciones predefinidas o función definida por el usuario.

## Investigue cuales son las funciones predefinidas en OracleSQL, PostgreSQL y SQLServer que nos permiten realizar operaciones sobre:

Las funciones predefinidas en SQL son aquellas que vienen incorporadas en el sistema de gestión de bases de datos. Estas funciones están disponibles para ser utilizadas directamente en las consultas y operaciones de SQL sin necesidad de definirlas previamente. Las funciones predefinidas abarcan una amplia gama de operaciones, como funciones matemáticas, funciones de manipulación de cadenas, funciones de fecha y hora, funciones de agregación, entre otras. Estas funciones predefinidas ofrecen funcionalidad estándar y amplían las capacidades de SQL para realizar operaciones comunes de manera eficiente.

Operaciones sobre:

|  |  |  |
| --- | --- | --- |
| NOMBRE FUNCIÓN | ACCIÓN | UN EJEMPLO |
| **a. Cadenas** |  |  |
| CONCAT | Concatena dos o más cadenas | SELECT CONCAT('Hello', 'World'); |
| LENGTH | Devuelve la longitud de una cadena | SELECT LENGTH('Hello'); |
| SUBSTRING | Extrae una parte de una cadena | SELECT SUBSTRING('Hello World', 7, 5); |
| UPPER | Convierte una cadena a mayúsculas | SELECT UPPER('hello'); |
| LOWER | Convierte una cadena a minúsculas | SELECT LOWER('HELLO'); |
| REPLACE | Reemplaza una subcadena en una cadena | SELECT REPLACE('Hello World', 'World', 'Universe'); |
| TRIM | Elimina espacios en blanco de una cadena | SELECT TRIM(' Hello '); |
| LEFT | Devuelve los primeros caracteres de una cadena | SELECT LEFT('Hello World', 5); |
| RIGHT | Devuelve los últimos caracteres de una cadena | SELECT RIGHT('Hello World', 5); |
| INSTR | Devuelve la posición de una subcadena | SELECT INSTR('Hello World', 'World'); |
| **b. Cifrado/Descifrado** |  |  |
| AES\_ENCRYPT | Cifra una cadena utilizando AES | SELECT AES\_ENCRYPT('Secret', 'password'); |
| AES\_DECRYPT | Descifra una cadena cifrada con AES | SELECT AES\_DECRYPT('Encrypted', 'password'); |
| MD5 | Calcula el hash MD5 de una cadena | SELECT MD5('Hello'); |
| SHA1 | Calcula el hash SHA-1 de una cadena | SELECT SHA1('Hello'); |
| ENCRYPT | Cifra una cadena utilizando el algoritmo crypt | SELECT ENCRYPT('Secret', 'salt'); |
| **c. Fechas** |  |  |
| NOW | Devuelve la fecha y hora actuales | SELECT NOW(); |
| CURDATE | Devuelve la fecha actual | SELECT CURDATE(); |
| CURTIME | Devuelve la hora actual | SELECT CURTIME(); |
| DATE | Extrae la parte de fecha de una fecha/hora | SELECT DATE('2023-07-05 10:30:00'); |
| TIME | Extrae la parte de hora de una fecha/hora | SELECT TIME('2023-07-05 10:30:00'); |
| YEAR | Extrae el año de una fecha | SELECT YEAR('2023-07-05'); |
| MONTH | Extrae el mes de una fecha | SELECT MONTH('2023-07-05'); |
| DAY | Extrae el día de una fecha | SELECT DAY('2023-07-05'); |
| **d. Matemáticas** |  |  |
| SUM | Calcula la suma de los valores en una columna | SELECT SUM(column\_name) FROM table\_name; |
| AVG | Calcula el promedio de los valores en una columna | SELECT AVG(column\_name) FROM table\_name; |
| MAX | Devuelve el valor máximo en una columna | SELECT MAX(column\_name) FROM table\_name; |
| MIN | Devuelve el valor mínimo en una columna | SELECT MIN(column\_name) FROM table\_name; |
| COUNT | Cuenta el número de filas o valores no nulos | SELECT COUNT(\*) FROM table\_name; |
| ROUND | Redondea un número | SELECT ROUND(3.14159, 2); |
| **e. Conversiones** |  |  |
| CAST | Convierte un valor a un tipo de datos | SELECT CAST(column\_name AS INT) FROM table\_name; |
| CONVERT | Convierte un valor a un tipo de datos | SELECT CONVERT(column\_name, INT) FROM table\_name; |
| DATE\_FORMAT | Convierte una fecha en una cadena con formato | SELECT DATE\_FORMAT(date\_column, '%Y-%m-%d') FROM table\_name; |
| **f. Manejo de errores** |  |  |
| ERRNO | Devuelve el número de error para la última operación | SELECT ERRNO(); |
| ERROR | Devuelve el mensaje de error para la última operación | SELECT ERROR(); |
| SIGNAL | Genera un error personalizado | SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Error message'; |
| **g. Información del motor de BD** |  |  |
| VERSION | Devuelve la versión del servidor MySQL | SELECT VERSION(); |
| DATABASE | Devuelve el nombre de la base de datos actual | SELECT DATABASE(); |
| USER | Devuelve el nombre de usuario actual | SELECT USER(); |
| CONNECTION\_ID | Devuelve el ID de conexión actual | SELECT CONNECTION\_ID(); |

## Qué es y cuál es el objetivo de una Función personalizada o definida por el usuario

**Función definida por el usuario:**

Una función definida por el usuario en SQL es una función personalizada creada por el usuario para satisfacer necesidades específicas. Estas funciones se crean utilizando el lenguaje SQL y permiten encapsular lógica de cálculo y manipulación de datos que no están disponibles en las funciones predefinidas. Las funciones definidas por el usuario se pueden utilizar en consultas y operaciones de SQL de la misma manera que las funciones predefinidas. Proporcionan flexibilidad y permiten a los usuarios extender la funcionalidad de SQL según sus requerimientos particulares.

## Cual es la sintaxis general para crear e invocar una Función definida por el usuario.

La sintaxis general para crear una función definida por el usuario en MySQL es la siguiente:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Usar la base de datos prueba\_transacciones\_grupoXYZ creada en la actividad de la Unidad 2 (punto 5 al punto 11), entonces deben hacer los siguiente:

## Crear una función que reciba como parámetros de entrada la cc y la clave de un deudor, entonces la función debe retornar Verdadero si existe un usuario con esa misma CC y Clave registrados en esa tabla, o retornar FALSO en caso contrario.

Sentencia SQL

DELIMITER //

CREATE FUNCTION verificar\_deudor(d\_cc INT, d\_clave VARCHAR(15))

RETURNS VARCHAR(10)

BEGIN

DECLARE existe BOOLEAN;

SET existe = EXISTS(

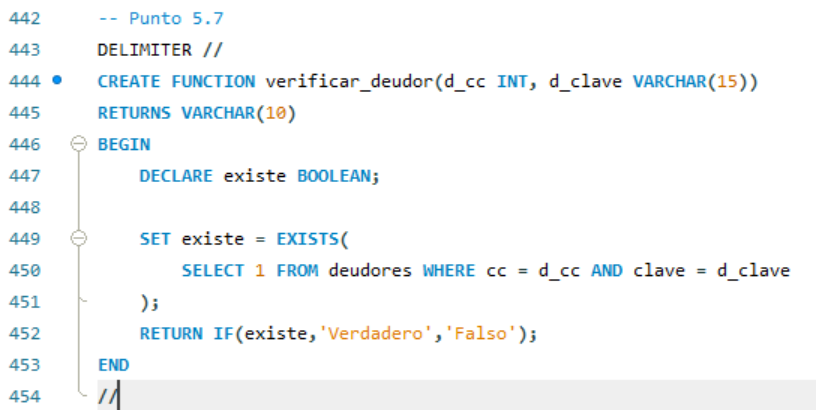
SELECT 1 FROM deudores WHERE cc = d\_cc AND clave = d\_clave

);

RETURN IF(existe,'Verdadero','Falso');

END

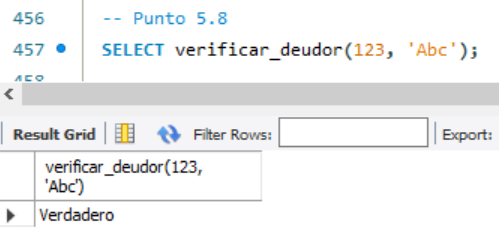
//



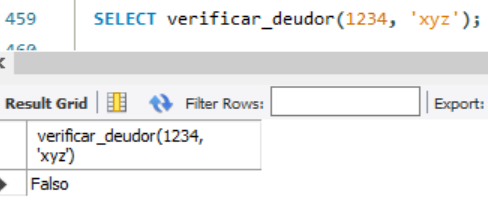
## Invocar y probar la función con datos que permitan mostrar resultado Falso y otros que permitan retornar resultado Verdadero.

Probamos para un deudor existente.

SELECT verificar\_deudor(123, 'Abc');



Ahora para uno no existente.



## Crear una función que reciba como parámetros la cédula de un deudor y devuelva un solo dato con el nombre del deudor, valor del último, fecha y estado del último crédito que tiene en el sistema. El resultado debe ser una cadena de texto en el siguiente formato: por ejemplo. NOMBRE: FULANITO DE TAL, FECHA: 12-04-2022, VALOR: $100000, ESTADO: ACTIVO

DELIMITER //

CREATE FUNCTION obtener\_informacion\_deudor(d\_cc INT)

RETURNS TEXT

BEGIN

DECLARE d\_nombre VARCHAR(30);

DECLARE ultimo\_credito VARCHAR(500);

DECLARE resultado VARCHAR(500);

-- Obtener el nombre del deudor

SELECT nombre INTO d\_nombre

FROM deudores

WHERE cc = d\_cc;

-- Obtener el último crédito del deudor

SELECT CONCAT('NOMBRE: ', d\_nombre, ', ',

'FECHA: ', DATE\_FORMAT(c.fecha, '%d-%m-%Y'), ', ',

'VALOR: $', c.valor, ', ',

'ESTADO: ', c.estado)

INTO ultimo\_credito

FROM creditos c

JOIN deudores d ON c.deudor\_id = d.cc

WHERE d.cc = d\_cc

ORDER BY c.fecha DESC

LIMIT 1;

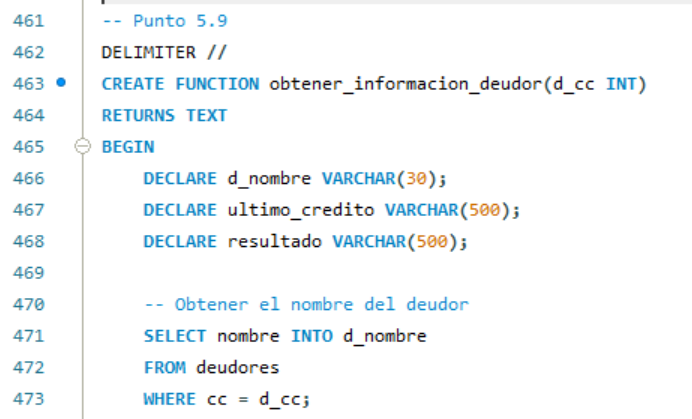
-- Construir el resultado en el formato requerido

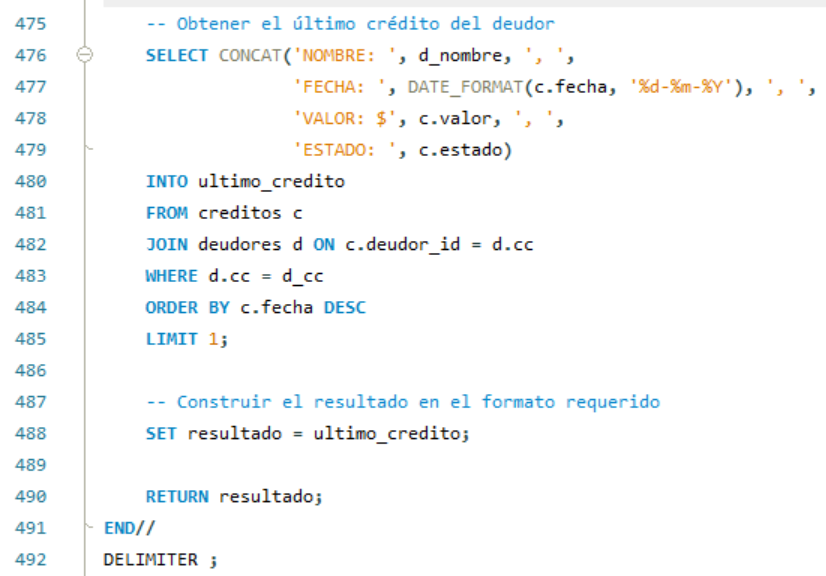
SET resultado = ultimo\_credito;

RETURN resultado;

END//

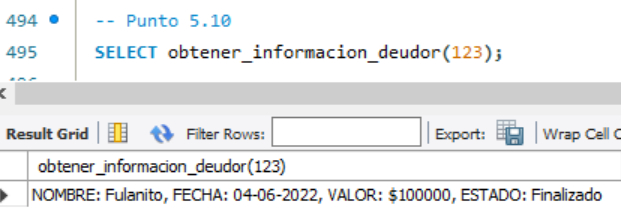
DELIMITER ;





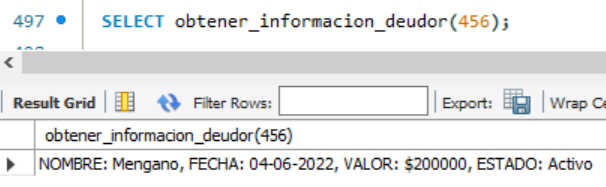
## Invocar y probar la función

SELECT obtener\_informacion\_deudor(123);



Se Prueba con otro deudor

SELECT obtener\_informacion\_deudor(456);



## Crear una que reciba 2 números y revuelva la sumatoria acumulativa entre ellos.

La siguiente función suma todos los números entre dos números dados.

DELIMITER //

CREATE FUNCTION sumatoria\_acumulativa(n\_1 INT, n\_2 INT)

RETURNS INT

BEGIN

DECLARE v\_suma INT DEFAULT 0;

DECLARE v\_min INT;

DECLARE v\_max INT;

IF n\_1 < n\_2 THEN

SET v\_min = n\_1;

SET v\_max = n\_2;

ELSE

SET v\_min = n\_2;

SET v\_max = n\_1;

END IF;

WHILE v\_min <= v\_max DO

SET v\_suma = v\_suma + v\_min;

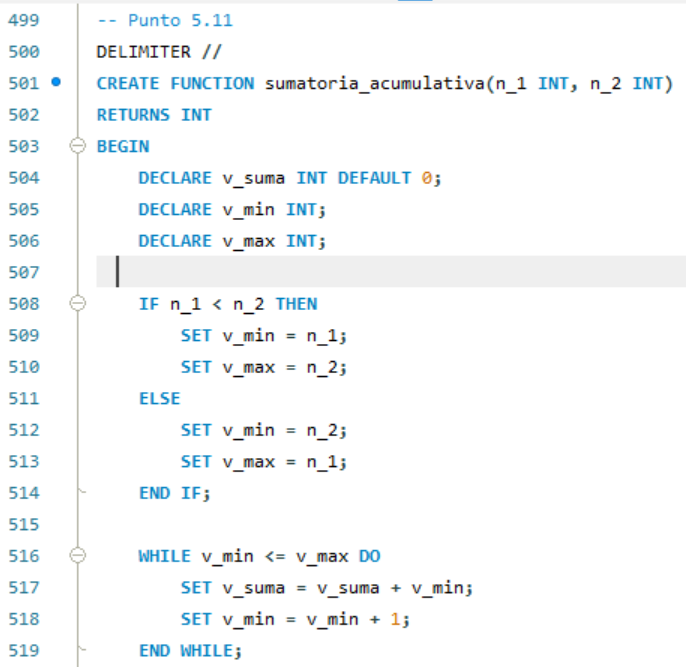
SET v\_min = v\_min + 1;

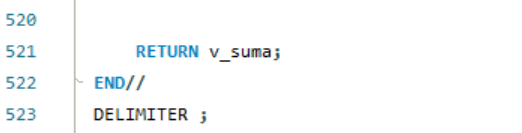
END WHILE;

RETURN v\_suma;

END//

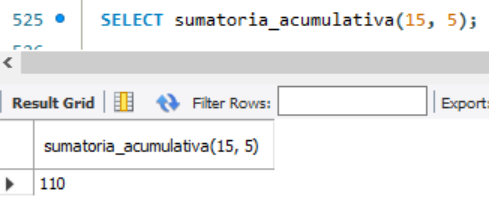
DELIMITER ;



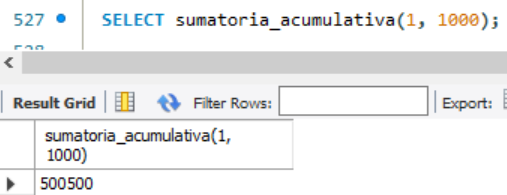


Al darle parámetros nos queda:

SELECT sumatoria\_acumulativa(15, 5);



SELECT sumatoria\_acumulativa(1, 1000);



## Crear un procedimiento almacenado con las siguientes características:

* 1. debe tener un parámetro de **entrada** que reciba el código de un deudor
  2. debe tener un parámetro de **salida** que devuelva la cantidad (cuantos) de créditos Activos que tiene ese deudor

DELIMITER //

CREATE PROCEDURE creditos\_activos(

IN d\_cc INT,

INOUT cantidad INT

)

BEGIN

DECLARE cant\_creditos\_activos INT;

SELECT COUNT(\*) INTO cant\_creditos\_activos

FROM creditos

WHERE deudor\_id = d\_cc

AND estado = 'Activo';

SET cantidad = cant\_creditos\_activos;

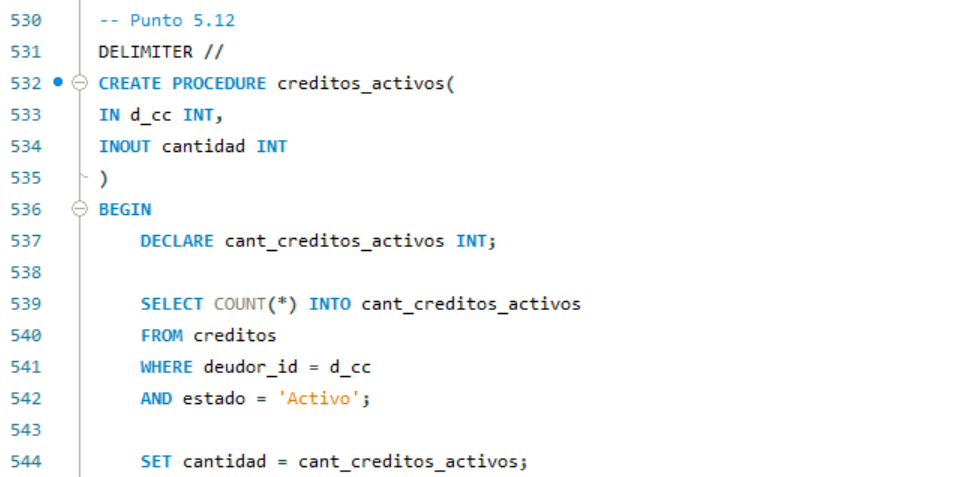
SELECT CONCAT("La cantidad de créditos activos que tiene el deudor con cc= ",

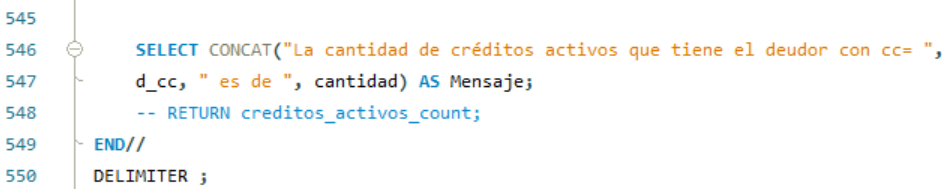
d\_cc, " es de ", cantidad) AS Mensaje;

-- RETURN creditos\_activos\_count;

END//

DELIMITER ;



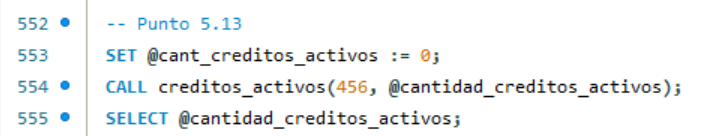


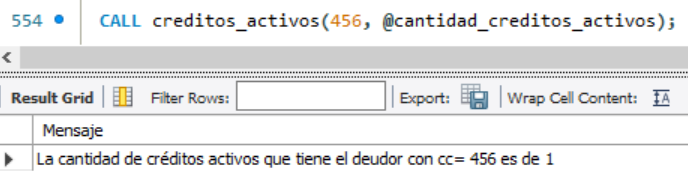
## Invocar y probar el procedimiento almacenado.

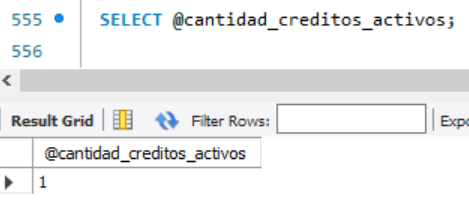
SET @cant\_creditos\_activos := 0;

CALL creditos\_activos(456, @cantidad\_creditos\_activos);

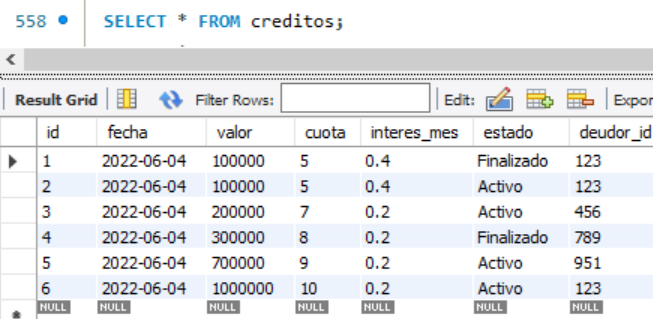
SELECT @cantidad\_creditos\_activos;







Verificamos en la tabla créditos.



.

# Bibliografía

*ChickenCode*. (12 de 06 de 2023). Obtenido de https://youtu.be/1MhU2-LB\_Z8

Mediacloud. (29 de 05 de 2023). *https://blog.mdcloud.es*. Obtenido de https://blog.mdcloud.es/crear-vistas-en-mysql-que-es-y-ventajas-de-hacerlo/#:~:text=Las%20vistas%20en%20MySQL%20(VIEWS)%20son%20tablas%20virtuales.&text=Solo%20muestran%20los%20datos%20que,a%20la%20fuente%20en%20s%C3%AD.

Microsoft. (29 de 05 de 2023). *https://learn.microsoft.com/*. Obtenido de https://learn.microsoft.com/es-es/sql/relational-databases/views/views?view=sql-server-ver16

Montes, A. (18 de 12 de 2022). *YouTube*. Obtenido de Restaurando backup de base de datos en PostgreSQL en formato .SQL desde consola (CMD).: https://youtu.be/0KvzfH2j9nQ

Nuggets, S. (15 de 12 de 2022). *YouTube*. Obtenido de NO ACTION Foreign Keys in PostgreSQL, learn how the NO ACTION behavior is enforced.: https://youtu.be/L1lpGsg5PbI

PostgreSql. (10 de 11 de 2022). *https://www.postgresql.org/*. Obtenido de https://www.postgresql.org/docs/current/sql-alterrole.html

Silberschatz, A., Korth, H., & Sudarshan, S. (2006). *Fundamentos de bases de datos.* Madrid: McGraw-Hill.

*tutorialesprogramacionya*. (15 de 04 de 2023). Obtenido de https://www.tutorialesprogramacionya.com/postgresqlya/index.php?inicio=50

VM, D. (18 de 12 de 2022). *YouTube*. Obtenido de Denisse VM: https://youtu.be/KIGSxLRPHw4