



**Clase 15.** Curso SQL

# ***FUNCIONES***

***RECUERDA PONER A GRABAR LA  
CLASE***





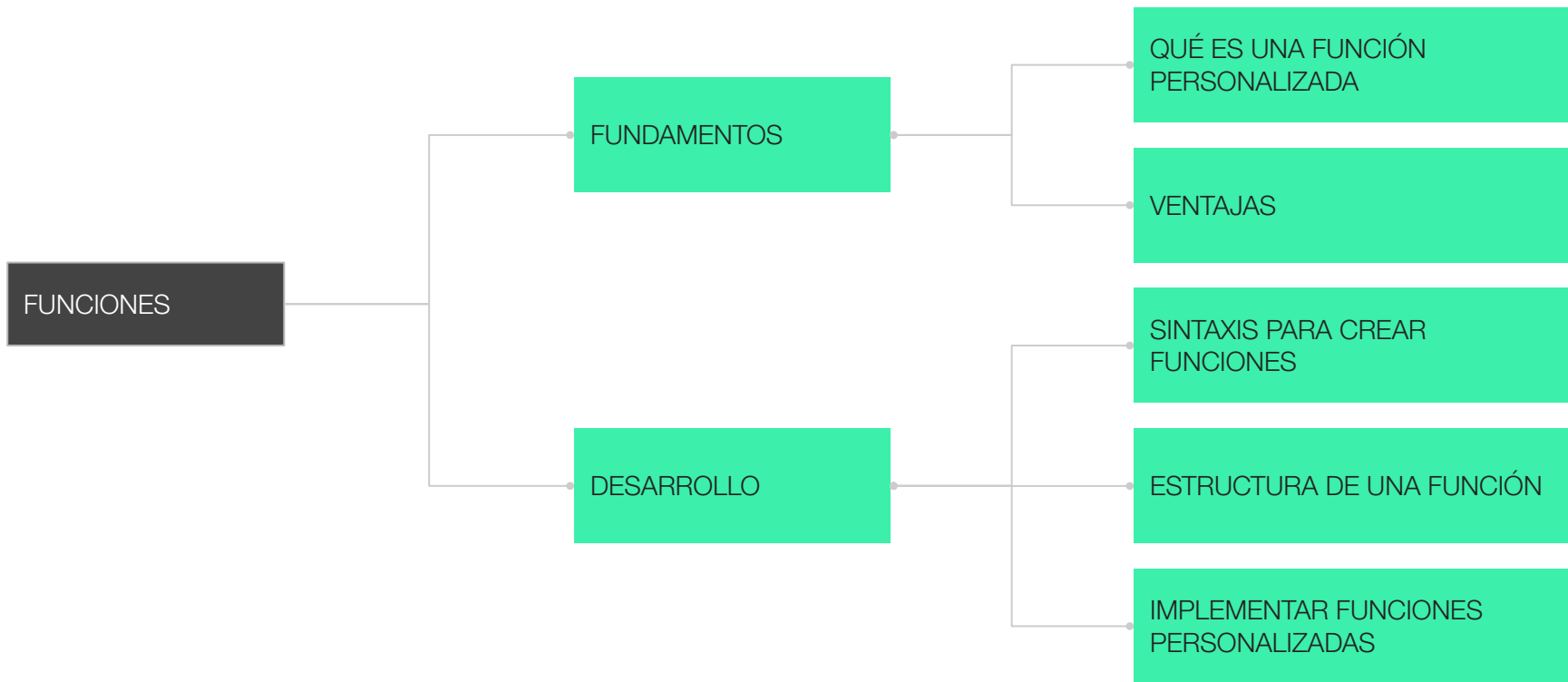
## ***OBJETIVOS DE LA CLASE***

- Definir las funciones personalizadas.
- Presentar la sintaxis para la creación de funciones personalizadas.
- Implementar nuestras propias funciones personalizadas.

# ***MAPA DE CONCEPTOS***

# MAPA DE CONCEPTOS CLASE 15

¡Para  
recordar!



# ***FUNCIONES***

En la **Clase 06** vimos cómo aprovechar las funciones escalares y de transformación, propias del lenguaje SQL.

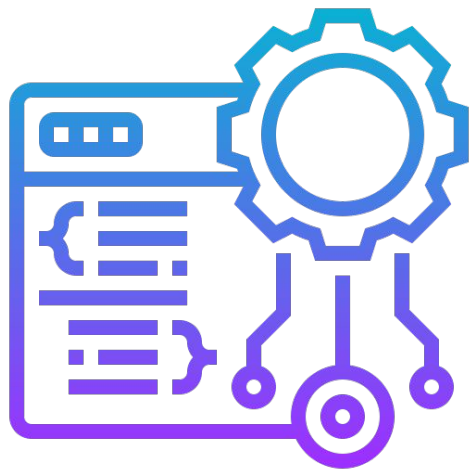
Hoy aprenderemos a crear nuestras funciones customizadas para cubrir diferentes necesidades específicas que las funciones predeterminadas del lenguaje SQL no contemplan.

# ***FUNCIONES***

# ***CONCEPTO GENERAL***

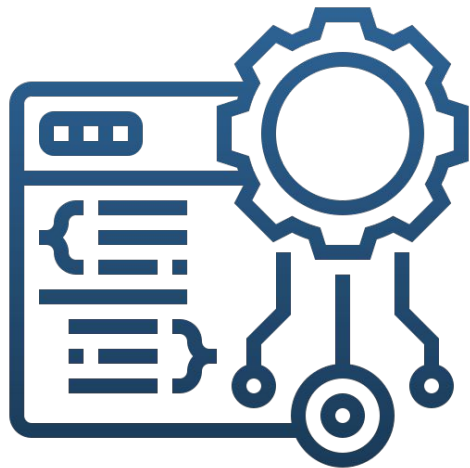


# ***FUNCIONES: DEFINICIÓN***



Las funciones customizadas permiten procesar y manipular datos de forma procedural y eficiente. Dichos datos son enviados a través de uno o más parámetros, al momento de invocar la función y retornando un único resultado.

# ***FUNCIONES: DEFINICIÓN***



Podemos crear funciones, a la medida de nuestra necesidad, combinando las mismas con funciones ya existentes del lenguaje Mysql, para así obtener los resultados deseados tal como necesitamos.

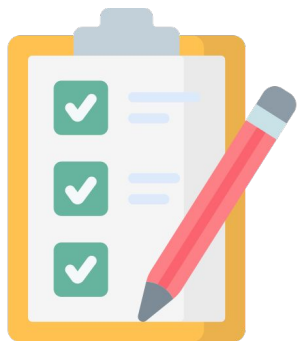
# ***FUNCIONES: DEFINICIÓN***



Para crear funciones personalizadas, necesitamos conocer el lenguaje de programación **SQL**, el cual nos permitirá definir una estructura básica que toda función personalizada debe cumplir para operar como tal.

***BENEFICIOS***

# ***BENEFICIOS***



Algunos de los beneficios a destacar:

- mejoran la integridad y seguridad de los datos
- optimizan el rendimiento de la base de datos
- otorgan una mejor lectura del código

# ***BENEFICIOS***



**Mejoran la integridad y seguridad de los datos:** crear funciones en el servidor, garantiza que, los resultados de los datos procesados serán iguales para cualquier Stack de Programación. Evitando el error humano programando del lado del cliente cuya lógica pueda diferir entre diferentes lenguajes de programación.

# ***BENEFICIOS***



**Optimizan el rendimiento de la base de datos:** para complementar el punto anterior, las funciones customizadas de Mysql envían los datos procesados por el servidor al cliente, minimizando el tráfico de información.

# ***BENEFICIOS***



**Otorgan una mejor lectura del código:** cuando creas tus propias funciones puedes normalizar o personalizar su nombre, de acuerdo a la convención de nombres de tu país o de la empresa en donde trabajas, haciendo así más fácil la lectura e interpretación del código SQL que utiliza las mismas.



# ***TIPOS DE FUNCIONES***

# ***TIPOS DE FUNCIONES***



En el mundo de la programación, existen tres tipos de funciones:

- Funciones simples
- Funciones con parámetro(s)
- *Funciones con parámetro(s) y retorno*

Particularmente, en SQL, se usa siempre el último tipo de función.

***SINTAXIS***

# ***SINTAXIS BÁSICA***



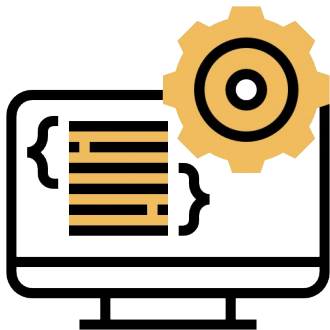
Como mencionamos anteriormente, las funciones personalizadas se crean utilizando el lenguaje *de programación SQL*.

Este es un lenguaje de programación casi tan poderoso como cualquier otro lenguaje, y cuenta con una sintaxis básica común a todas las funciones personalizadas y/o procedimientos almacenados.

**Veamos a continuación cuál es...**

# ***SINTAXIS BÁSICA***

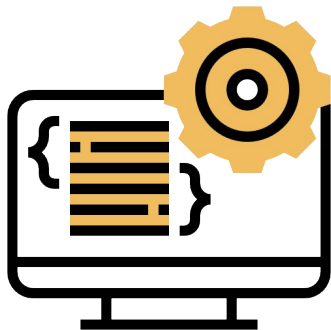
# ***SINTAXIS BÁSICA***



**CREATE FUNCTION `nombre\_de\_la\_funcion`:**

esta es la sentencia DDL que nos permite crear una función partiendo desde una ventana de *Scripting*. El nombre de la función va especificado entre comillas simples del tipo **back tick** o, **acento grave**.

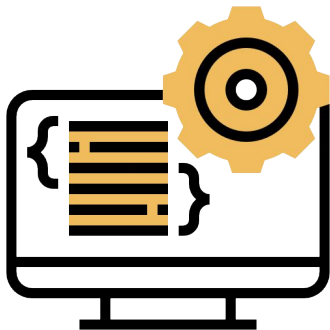
# ***SINTAXIS BÁSICA***



**CREATE FUNCTION `nombre\_de\_la\_funcion`:**

esta es la sentencia DDL que nos permite crear una función desde una ventana de *Scripting*. El nombre de la misma se especifica entre comillas simples del tipo **back tick** o, **acento grave**.

# ***SINTAXIS BÁSICA***



**RETURNS tipoDeDato:** Las funciones devuelven o retornan usualmente algún tipo de dato. Éste puede ser **boolean**, **char** o **number**, entre otros, según lo que estamos procesando.

Junto a la sentencia **CREATE FUNCTION** debemos especificar qué tipo de datos retornará la misma.



# ***SINTAXIS BÁSICA***



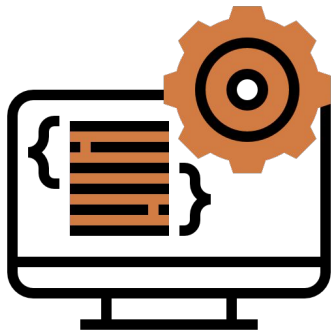
Estructura básica de una función.

```
CREATE FUNCTION `nombre_de_la_funcion` RETURNS CHAR(60)
```

```
...
```

# ***RECEPCIÓN DE PARÁMETROS***

# PARÁMETROS



(**param1 INT, param2 INT**): las funciones suelen recibir uno o más parámetros de entrada, los cuales provienen de las columnas de una tabla.

Debemos especificar el nombre de cada parámetro, junto al tipo de dato de éste y, si son más de uno, debemos separarlos por una coma.

# ***PARÁMETROS***



Estructura básica de una función con más de un parámetro.

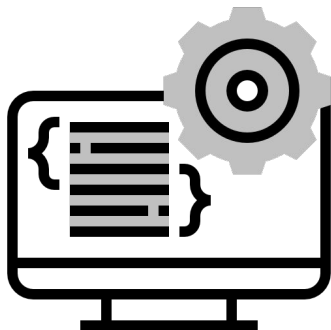
```
CREATE FUNCTION `nombre_de_la_funcion` (param1 INT,  
param2 INT) RETURNS CHAR(60)
```

...

# ***TIPOS DE FUNCIONES***



# ***TIPOS DE FUNCIONES***



Cada función que creamos suele tener diferentes fines. Para optimizar el comportamiento de Mysql al momento de ejecutarla, debemos indicarle qué tipo de función estará ejecutando.

# ***TIPOS DE FUNCIONES***



Las principales opciones que permiten definir el tipo de función, son:

- **DETERMINISTIC**
- **NO SQL**
- **READS SQL DATA**

# ***DETERMINISTIC***



Se considera a una rutina como **DETERMINISTIC** (*determinista*) si esta produce el mismo tipo de resultado que el de sus parámetros de entrada. Se utiliza comúnmente con cadena del texto (*string*) o procesamiento matemáticos, aunque no se limita solo a estos resultados.



# ***NO SQL***



Se considera a una rutina como **NO SQL**, cuando esta no utiliza ningún tipo de llamada o invocación de datos a través del lenguaje SQL. O sea, no se llama una sentencia **SELECT**, **UPDATE**, ni cualquier otra sentencia del tipo DML.

# ***READS SQL DATA***



Se considera a una rutina como **READS SQL DATA**, cuando esta función sólo leerá datos de una base de datos. No modificará datos a través de las cláusulas **INSERT**, **UPDATE** o **DELETE**, solo leerá información a través de la sentencia **SELECT**.

# ***TIPOS DE FUNCIONES***

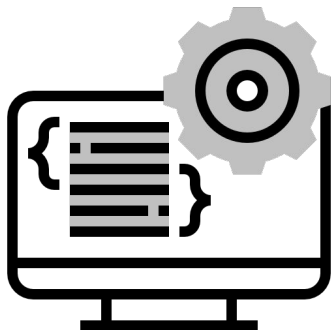


Estructura básica de una función almacenada con más de un parámetro y la implementación del tipo de función que manejará.

```
CREATE FUNCTION `nombre_de_la_funcion` (param1 INT,  
param2 INT) RETURNS CHAR(60)  
DETERMINISTIC  
...
```

# ***CUERPO DE LA FUNCIÓN***

# ***CUERPO DE LA FUNCIÓN***



**BEGIN...END:** El cuerpo de la función se define a través de una estructura de inicio y fin. Y es allí, dentro de este cuerpo donde definiremos, a posteriori, el código que le dará vida a la misma.

# ***CUERPO DE LA FUNCIÓN***



Estructura básica de una función con parámetros y cuerpo principal.

```
CREATE FUNCTION `nombre_de_la_funcion` (param1 INT, param2 INT)
RETURNS CHAR(60)
DETERMINISTIC
BEGIN
...
END
```



***BREAK***

**¡5/10 MINUTOS Y VOLVEMOS!**

# ***DEFINIR UNA VARIABLE***



# DEFINIR UNA VARIABLE



**DECLARE resultado\_id INT:** para manejar los diferentes valores dentro de una función, podemos definir una o más variables. Éstas se definen a través de la palabra reservada **DECLARE** seguido del **nombre de la variable** y el **tipo de dato** que manejará.

# DEFINIR UNA VARIABLE

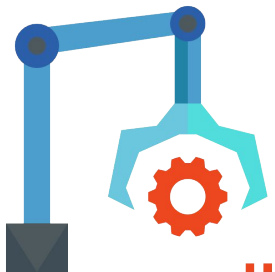


Estructura básica de una función que contiene una variable interna definida.

```
CREATE FUNCTION `nombre_de_la_funcion` (param1 INT, param2 INT) RETURNS  
CHAR(60)  
DETERMINISTIC  
BEGIN  
    DECLARE resultado INT;  
    ...  
END
```

# ***ESTABLECER VALOR DE UNA VARIABLE***

# ***ESTABLECER EL VALOR DE UNA VARIABLE***



**SET resultado\_id = 1 + 1**: cuando deseamos establecer un valor en una variable, debemos utilizar la palabra reservada **SET** seguida de la variable, el operador de definición, y el valor que se le asignará.

# ***ESTABLECER EL VALOR DE UNA VARIABLE***



Estructura básica de una función que contiene una variable interna definida y con un valor asignado.

```
CREATE FUNCTION `nombre_de_la_funcion` (param1 INT, param2 INT) RETURNS  
CHAR(60)  
DETERMINISTIC  
BEGIN  
    DECLARE resultado INT;  
    SET resultado = (param1 * param2);  
    ...  
END
```

***RETORNAR UN RESULTADO***

# ***RETORNAR UN RESULTADO***



**RETURN resultado\_id:** Finalmente, cuando la función realice la o las operaciones pertinentes, devolverá el resultado de las mismas a través de la palabra reservada **RETURN** seguido de la variable que almacena dicho resultado.

# ***RETORNAR UN RESULTADO***



Estructura básica de una función que contiene una variable interna definida, con un valor asignado y retornando un resultado.

```
CREATE FUNCTION `nombre_de_la_funcion` (param1 INT, param2 INT) RETURNS  
CHAR(60)  
DETERMINISTIC  
BEGIN  
    DECLARE resultado INT;  
    SET resultado = (param1 * param2);  
    RETURN resultado;  
END
```



# ***IMPLEMENTACIÓN DE FUNCIONES***



## ***EJEMPLO EN VIVO***

*Veamos cómo implementar nuestras funciones personalizadas en Mysql. Te invitamos a seguir este ejemplo en vivo a la par del docente.*



Vamos a generar una función que realice un cálculo matemático, a partir de una superficie que se requiere pintar. Sabemos que un metro cuadrado de pared requiere 100 cm<sup>3</sup> de pintura.

Al invocar a la función, debemos pasarle:

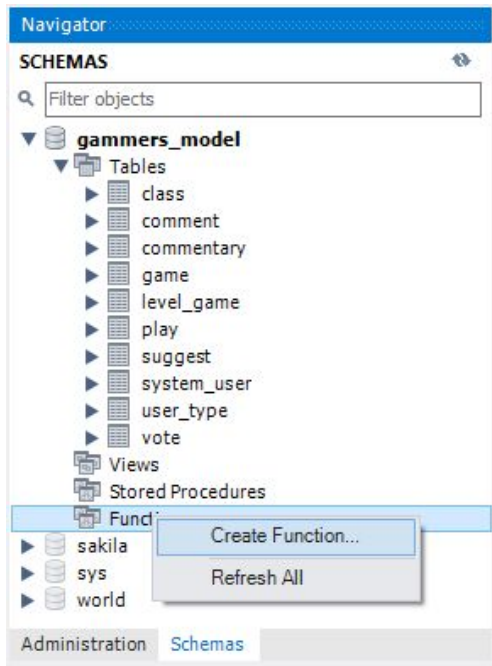
- largo de la pared (*en metros*)
- alto de la pared (*en metros*)
- cantidad de manos de pintura



# ***CREAR LA FUNCIÓN ALMACENADA***

# CREAR LA FUNCIÓN

Ejemplo  
en vivo



Desde **Mysql Workbench** podemos crear una nueva función simplemente ubicando el cursor **Objeto Functions** del esquema SQL con el cual deseamos trabajar.

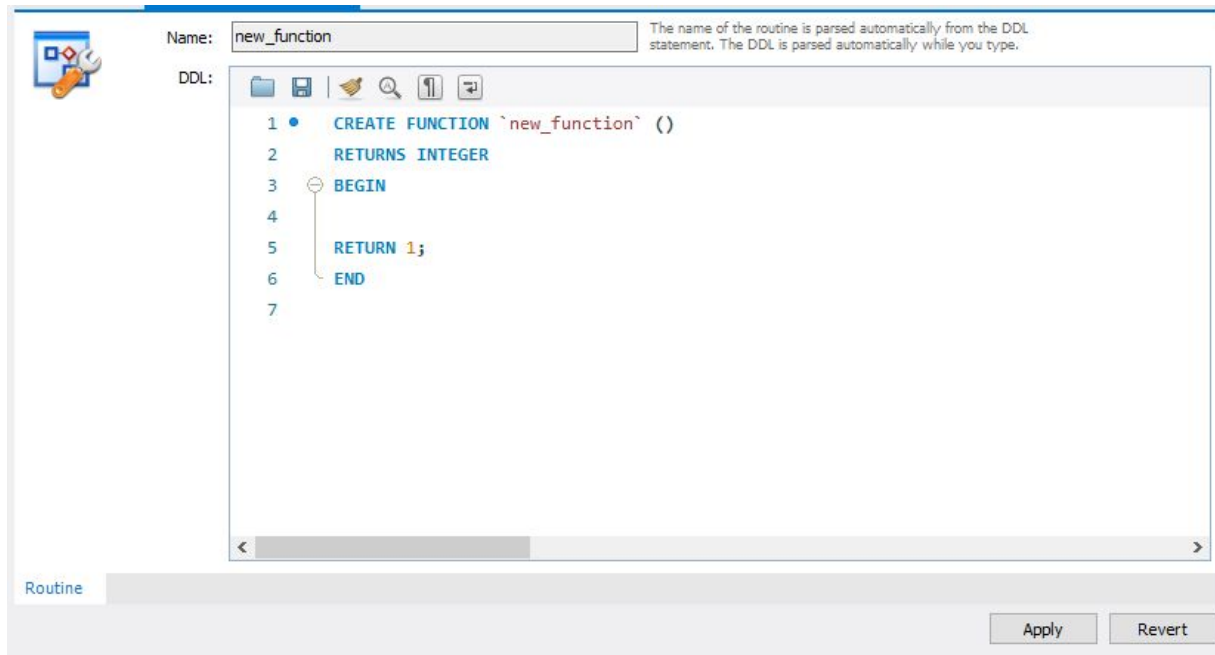
Pulsamos el botón secundario del mouse y seleccionamos la opción **Create Function...** en el menú contextual.

# CREAR LA FUNCIÓN

Ejemplo  
en vivo



**Mysql Workbench** nos mostrará una ventana para crear una función customizada:



**CODER HOUSE**

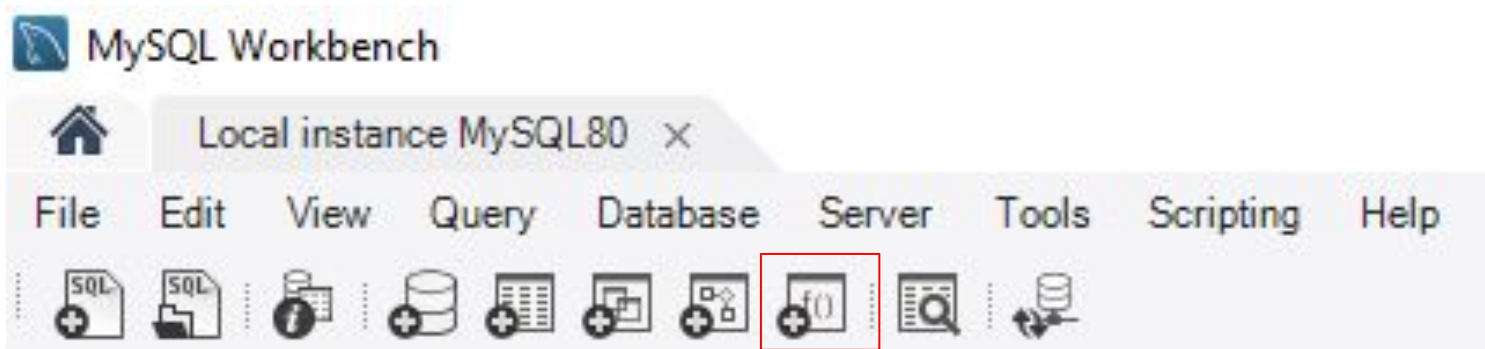
# CREAR LA FUNCIÓN

Ejemplo  
en vivo



Otra opción desde **Mysql Workbench** es identificando el botón **Create a new function...**, ubicado en la Barra de Herramientas superior.

Al pulsarlo, se abrirá una ventana con la estructura base de una función.



# ***DEFINIR ESTRUCTURA DE LA FUNCIÓN***





Creamos la función con su nombre, los parámetros que recibirá, y el valor que retornará.

```
CREATE FUNCTION `calcular_litros_de_pintura` (largo INT, alto INT, total_manos INT)  
RETURNS FLOAT
```



Como no usaremos SQL, definimos el tipo de función como **NO SQL**,  
y le agregamos el cuerpo a través de **BEGIN** y **END**.

```
CREATE FUNCTION `calcular_litros_de_pintura` (largo INT, alto INT, total_manos INT)  
RETURNS FLOAT  
NO SQL  
BEGIN  
...  
END
```



Definimos las variables `resultado` y `litro_x_m2`. A esta última le asignamos el valor `0.10`, correspondiente a los **Lts** de pintura por **M<sup>2</sup>**.

```
CREATE FUNCTION `calcular_litros_de_pintura` (largo INT, alto INT, total_manos INT)
RETURNS FLOAT
NO SQL
BEGIN
    DECLARE resultado FLOAT;
    DECLARE litro_x_m2 FLOAT;
    SET litro_x_m2 = 0.10;
END
```



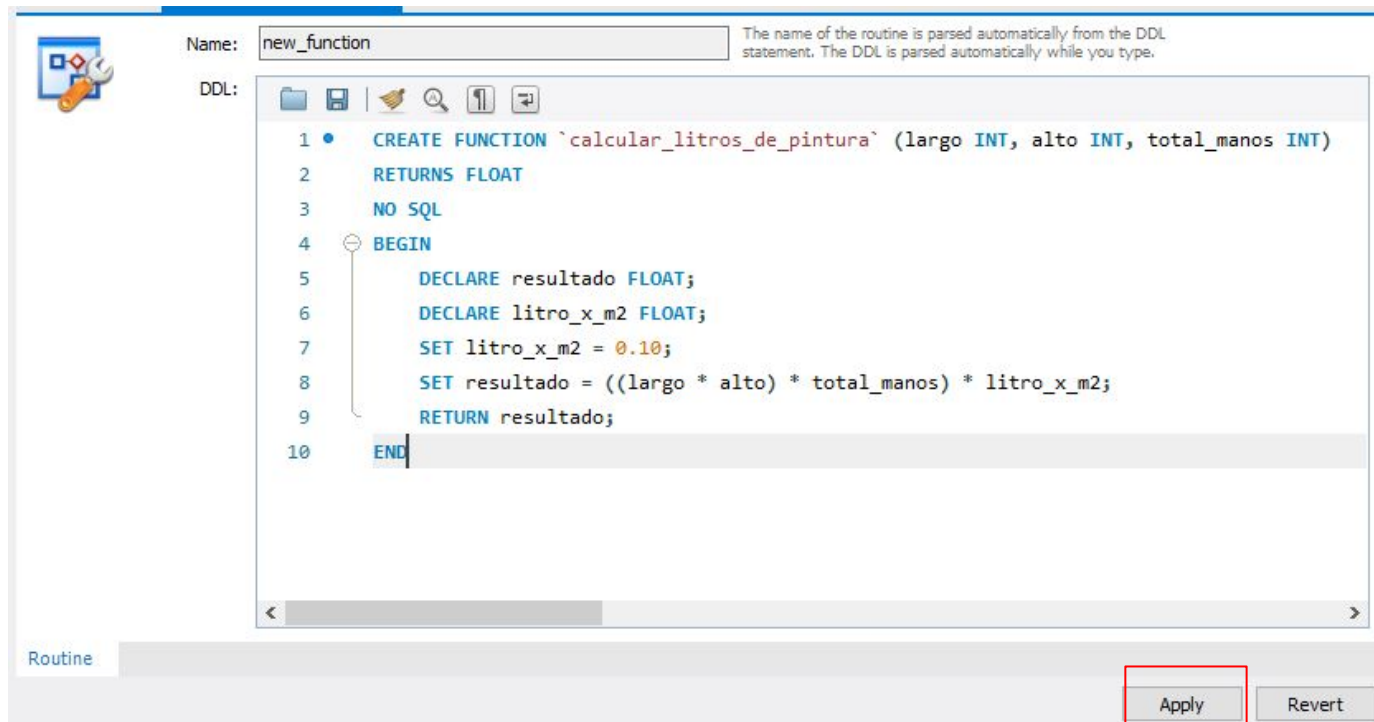
Multiplicamos largo \* alto. Luego, lo multiplicamos por el total de manos de pintura y, finalmente, por el valor del litro x m2. Retornamos el resultado almacenado en la variable homónima, mediante RETURN.

```
CREATE FUNCTION `calcular_litros_de_pintura` (largo INT, alto INT, total_manos INT) RETURNS  
FLOAT  
NO SQL  
BEGIN  
    DECLARE resultado FLOAT;  
    DECLARE litro_x_m2 FLOAT;  
    SET litro_x_m2 = 0.10;  
    SET resultado = ((largo * alto) * total_manos) * litro_x_m2;  
    RETURN resultado;  
END
```

***GUARDAR LA FUNCIÓN CREADA***

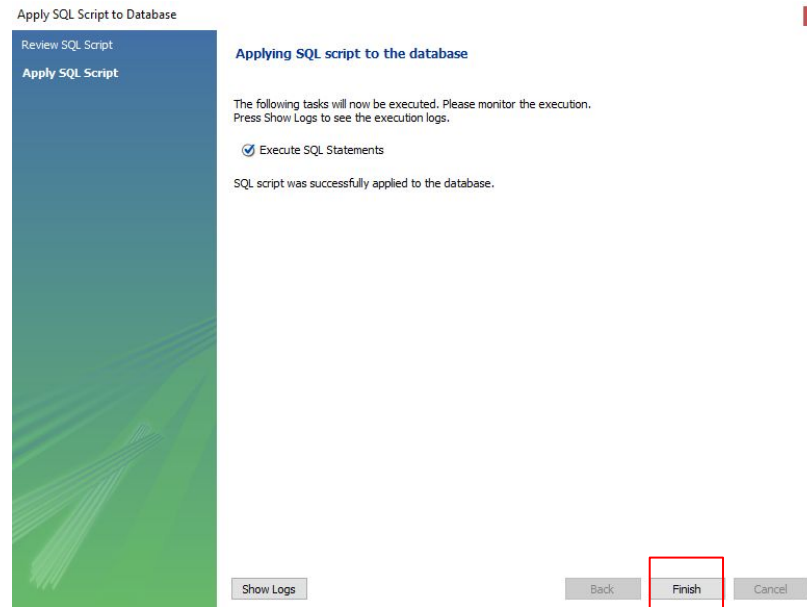
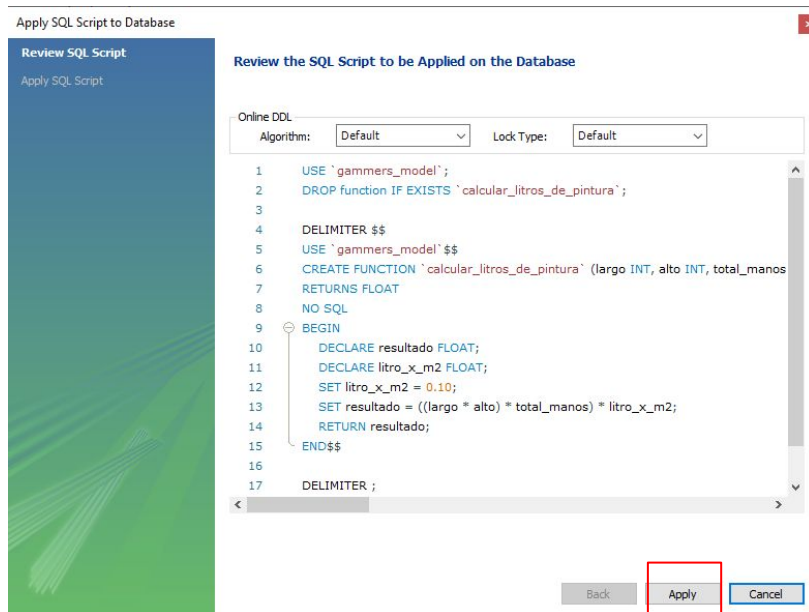


## Dentro del editor de funciones del **MySQL Workbench**:



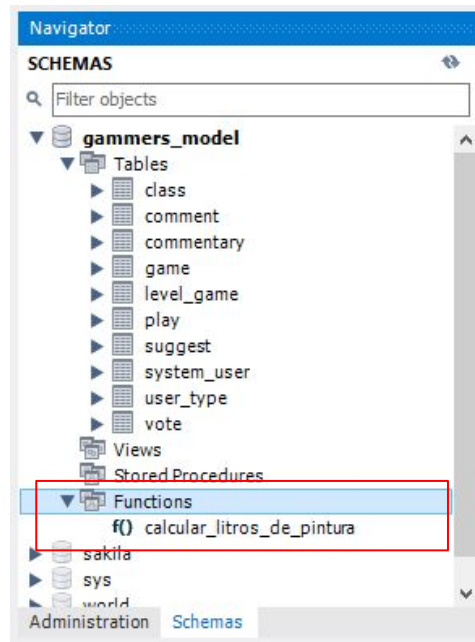


Al pulsar el botón **Apply**, **Mysql Workbench** nos pedirá confirmar la estructura del algoritmo, revisando por última vez los parámetros y código de éste.





Nueva función creada 🖱️





***PROBAR LA FUNCIÓN***

# ***PROBAR LA FUNCIÓN***

Ejemplo  
en vivo



Abramos a continuación una ventana de Script en **Mysql Workbench**.

Luego, escribimos la siguiente sentencia:

```
SELECT calcular_litros_de_pintura(22, 5, 3) AS  
total_pintura;
```

# PROBAR LA FUNCIÓN

Ejemplo  
en vivo



The screenshot shows a SQL IDE window titled "SQL File 5\* x". The query editor contains the following SQL statement:

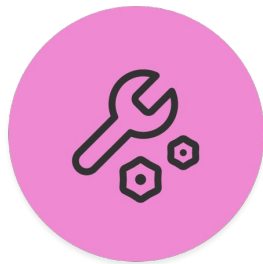
```
1 • SELECT calcular_litros_de_pintura(22, 5, 3) AS total_pintura;  
2  
3  
4  
5  
6
```

Below the query editor, the "Result Grid" is displayed, showing the output of the query:

total_pintura
33

Finalmente, vemos nuestra función personalizada en acción.

Probemos de cambiar los parámetros numéricos, para ir obteniendo otros resultados.



# ***PRÁCTICA DE FUNCIONES SQL***

Practicarás la creación de funciones sobre la base de datos Gamers.

*Tiempo estimado: 15 minutos*

# PRÁCTICA DE FUNCIONES SQL

Desafío  
generico



Crea una nueva función en la **DB GAMERS**, llamada **get\_game()** para obtener el nombre del videojuego, pasándole a dicha función el parámetro **id\_game**.

- 👉 Los nombres de los videojuegos se encuentran en la tabla game.
- 👉 Luego, debes crear una consulta del tipo SELECT sobre la tabla game, obteniendo sólo el name del videojuego.



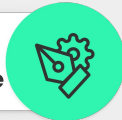
# ***PRESENTAR EN FORMATO SQL***

Presentar en formato .sql el script de inserción de datos de la base de datos del proyecto final.

# PRESENTAR EN FORMATO SQL

**Formato:** El archivo a presentar debe ser del tipo **.sql** nombrado como “Formato + Apellido”.

Desafío  
entregable



**>> Consigna:** Presentar el script de creación de 2 funciones almacenadas con base en los datos de la base de datos del proyecto final.

**>>Aspectos a incluir en el entregable:**

Puedes incluir una función que haga uso interno de funciones propias de SQL, y una segunda función que permita obtener valores de otras tablas, reemplazando a JOIN o de una subconsulta. Procura que cada función reciba, al menos, un parámetro de entrada. Los datos que retorne pueden ser cualquier tipo de dato.

***¿PREGUNTAS?***

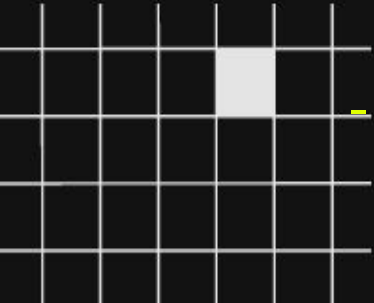






# ***¡MUCHAS GRACIAS!***

Resumen de lo visto en clase hoy:

- 
- Fundamentos de funciones personalizadas.
  - Sintaxis del código para la creación de funciones.
  - Desarrollo de una función personalizada.



***OPINA Y VALORA ESTA CLASE***

***RECUERDA PONER A GRABAR LA  
CLASE***

