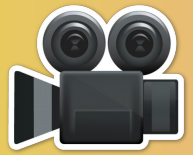
CODER HOUSE

Clase 06. SQL

SUBLENGUAJE DDL

RECUERDA PONER A GRABAR LA CLASE



CODER HOUSE

OBJETIVOS DE LA CLASE

- Reconocer e implementar sentencias de DATA DEFINITION LANGUAGE
- Conocer el significado e implementación de funciones escalares y de transformación

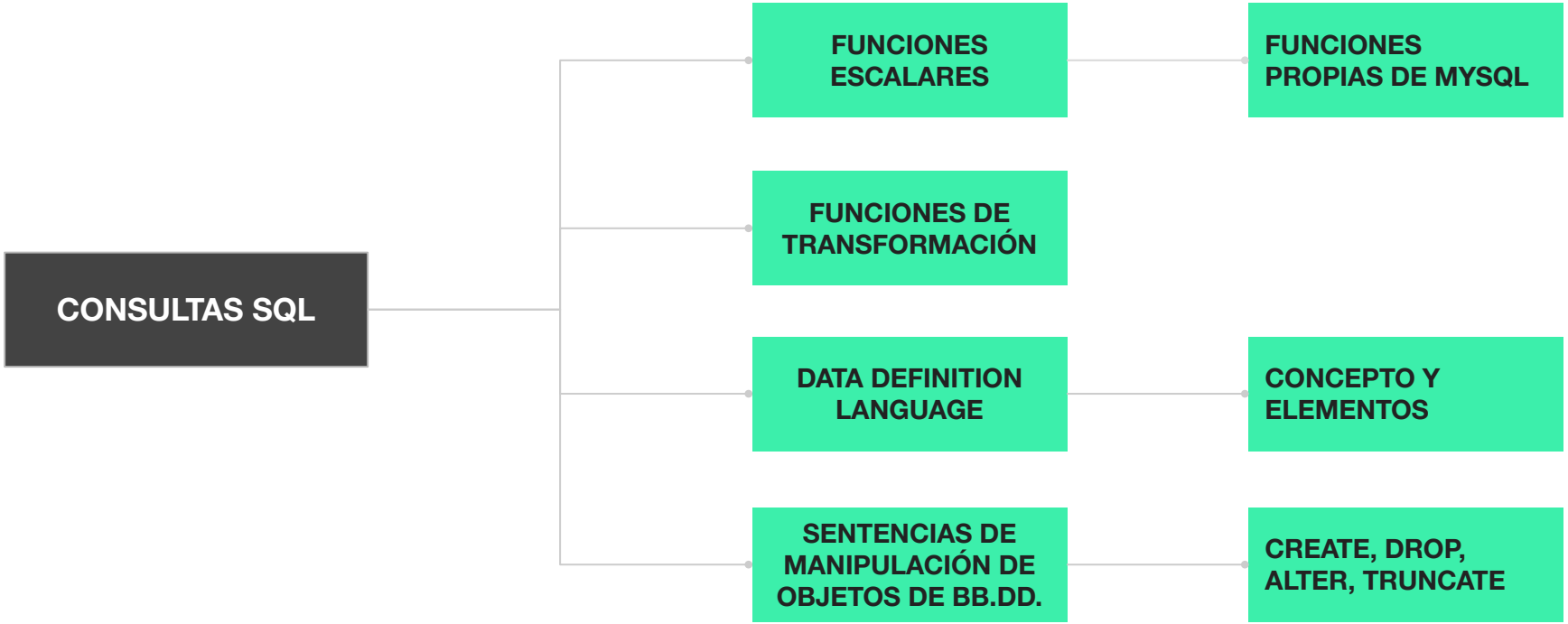
CODER HOUSE

MAPA DE CONCEPTOS

CODER HOUSE

MAPA DE CONCEPTOS CLASE 6

¡Para recordar!



CODER HOUSE

A través de las funciones escalares y de transformación, comenzaremos a manipular la información almacenada en la DB transformando y convirtiendo la misma a diferentes tipos de datos

También profundizaremos en el DDL para entender cómo crear, modificar y eliminar los principales objetos de una DB

SUBLENGUAJE DDL

CODER HOUSE

FUNDAMENTOS DE DATA DEFINITION LANGUAGE

CODER HOUSE

EL LENGUAJE DE DEFINICIÓN DE DATOS (DDL)

En la primera clase aprendimos que SQL está conformado por cuatro sublenguajes aplicables en diferentes ámbitos de uso en la DB: **DDL, DML, DCL, TCL**

Hoy profundizaremos **DATA DEFINITION LANGUAGE - DDL**, para sacar partido de todos sus beneficios.

CODER HOUSE

DATA DEFINITION LANGUAGE

DDL

El **Lenguaje de Definición de Datos** se ocupa de **modificar la estructura de objetos de una DB**

Lo conforman **diferentes sentencias** que nos **permiten crear, modificar, borrar o definir la estructura** de las tablas que almacenan datos

SENTENCIAS DDL

Las sentencias disponibles a través de DDL, son:

- **CREATE**
- **ALTER**
- **DROP**
- **TRUNCATE**

Con ellas creamos, modificamos, alteramos y eliminamos objetos.

SENTENCIAS DDL

Si bien estas tareas se pueden, y suelen realizar a través de una herramienta de gestión como **Mysql Workbench**, el DDL permite en aquellos casos donde la DB oficia de motor de sistemas web o de gestión, generar muchas o todas estas tareas de forma automatizada.

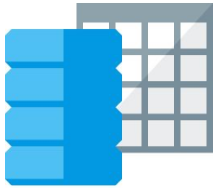
DDL: CREATE

DDL: CREATE

La sentencia **CREATE** cumple la función de **crear nuevos objetos en la base de datos**.

Los tipos de objetos a crear pueden ser: *tablas*, *índices*, *stored procedures* y hasta nuevas *bases de datos*, además, *usuarios* específicos.

SENTENCIA DDL CREATE



Enfoquémonos en la creación de una tabla.

Su sentencia sería 👇

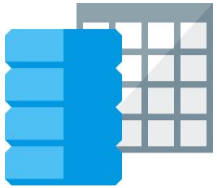
```
CREATE TABLE [nombre de la tabla](

[definiciones de columnas]),

([parámetros de la tabla]);
```

CODER HOUSE

SENTENCIA DDL CREATE



`[nombre de la tabla]`: definimos el nombre distintivo de la tabla a crear. Ej: **friend**.

`[definiciones de columnas]`: definimos las columnas o campos y sus propiedades o tipo de datos.

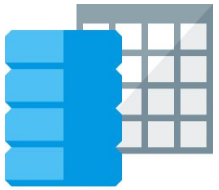
`[parámetros de la tabla]`: definimos otras particularidades de la tabla como por ejemplo, los índices.

DEFINICIÓN DE COLUMNAS O CAMPOS

CODER HOUSE

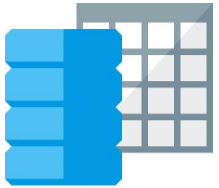
DDL: CREATE DEFINICIÓN DE CAMPOS

Para **definir los campos** o columnas, **debemos indicar el nombre** que queremos para éste, **el tipo de dato** que contendrá (con o sin límite en cantidad de caracteres), y **si acepta o no valores nulos**.



DDL: CREATE DEFINICIÓN DE CAMPOS

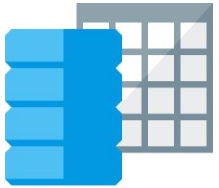
```
CREATE TABLE pay (
id_pay          INT NOT NULL AUTO_INCREMENT,
amount          REAL NOT NULL,
currency        VARCHAR(20) NOT NULL,
date_pay        DATE NOT NULL,
pay_type        VARCHAR(50),
id_system_user  INT NOT NULL,
id_game         INT NOT NULL),
([parámetros de la tabla]));
```



CODER HOUSE

DDL: CREATE PARÁMETROS DE LA TABLA

```
CREATE TABLE pay (
id_pay          INT NOT NULL AUTO_INCREMENT
                PRIMARY KEY,
amount          REAL NOT NULL,
currency        VARCHAR(20) NOT NULL,
date_pay        DATE NOT NULL,
pay_type        VARCHAR(50),
id_system_user  INT NOT NULL,
id_game         INT NOT NULL),
([parámetros de la tabla]));
```



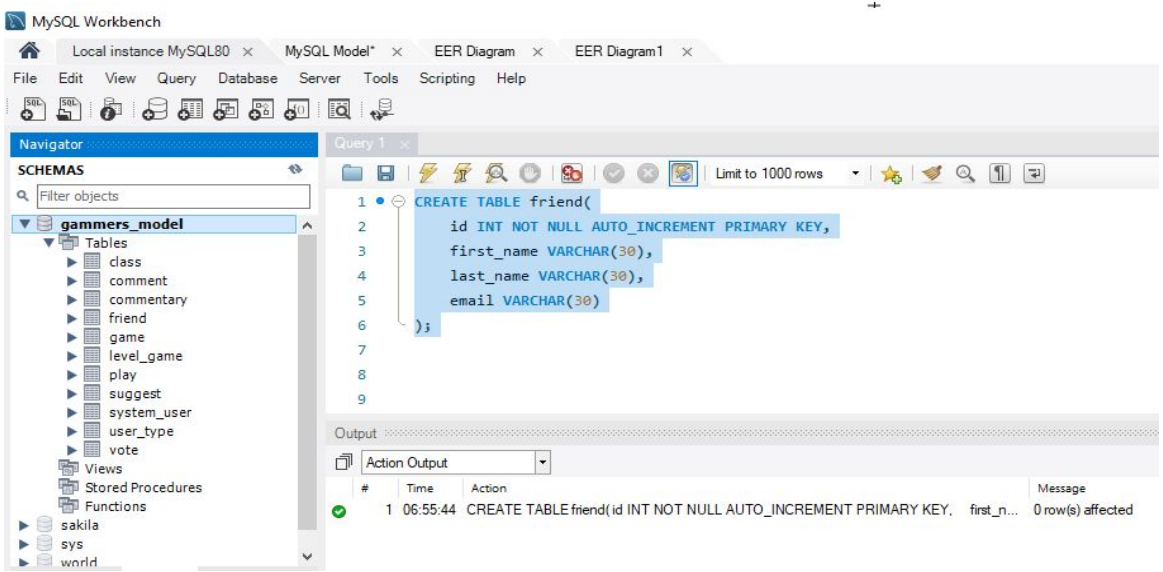
CODER HOUSE

¡VAMOS A PRACTICAR UN POCO!

CODER HOUSE

Ejemplo en vivo

PRUEBA DE CREACIÓN DE LA TABLA



Prueba la sentencia

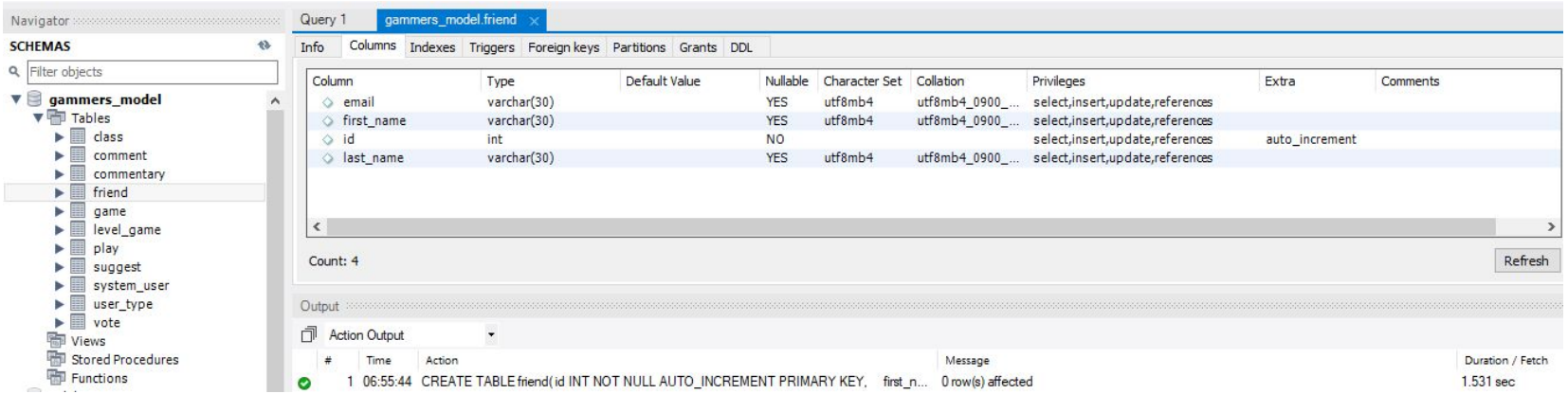
CREATE TABLE, en **Mysql Workbench**.

- ubica la tabla en el apartado **tables**, una vez ejecutado el script
- si no la visualizas, pulsa **refresh** para actualizar la vista de objetos

CODER HOUSE

Ejemplo en vivo

VERIFICACIÓN DE DATOS



Finalmente, pulsa el botón secundario del mouse sobre la tabla creada, elige del menú la opción **TABLE INSPECTOR**, y posiciónate en la pestaña **Columns**. Allí podrás verificar la generación de cada campo de la tabla y sus tipos de datos y parámetros definidos

CODER HOUSE

DDL: ALTER

CODER HOUSE

DDL: ALTER

La sentencia **ALTER** , del inglés “*alterar*”, permite modificar la estructura de una tabla u objeto de base de datos.

Con ella podemos agregar/quitar campos, modificar el tipo de datos de un campo y agregar o quitar índices.

SENTENCIA DDL ALTER

Veamos a continuación cómo podemos agregar un campo a la una tabla **ya** creada 👇

```
ALTER TABLE [nombre de la tabla]

ADD [definiciones de columnas];
```

CODER HOUSE

SENTENCIA DDL ALTER

`[nombre de la tabla]`: nombre de la tabla a alterar. Ejemplo: **friend**

ADD: es la acción que realizaremos sobre la tabla. En este caso, agregar un nuevo campo.

`[definición de columna]`: Definimos la o las nuevas columnas, tal como hicimos con la sentencia **CREATE**.

Ejemplo en vivo

DDL: ALTER (PARÁMETROS DE LA TABLA)

Agreguemos a continuación un campo en nuestra tabla **friend**.

El mismo se llamará **age** y será del tipo **INT**.

```
ALTER TABLE friend

ADD age INT;
```

CODER HOUSE

Ejemplo en vivo

VERIFICACIÓN DE DATOS

Verifiquemos a continuación mediante **TABLE INSPECTOR** que el nuevo campo se ha creado satisfactoriamente.

Info | Columns | Indexes | Triggers | Foreign keys | Partitions | Grants | DDL

Column	Type	Default Value	Nullable	Character Set	Collation	Privileges	Extra	Comments
age	int		YES			select,insert,update,references		
email	varchar(30)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references		
first_name	varchar(30)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references		
id	int		NO			select,insert,update,references	auto_increment	
last_name	varchar(30)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references		

- *Pulsemos el botón **Refresh**, para actualizar la visualización de su contenido*

CODER HOUSE

ALTER TABLE [MODIFY]

DDL: ALTER (MODIFY)

Ejemplo en vivo

Si deseamos modificar los valores para un campo existente, debemos reemplazar la instrucción **ADD** por **MODIFY**.

```
ALTER TABLE friend

MODIFY email VARCHAR(50) NOT NULL;
```

CODER HOUSE

VERIFICACIÓN DE DATOS

Ejemplo en vivo

Info | Columns | Indexes | Triggers | Foreign keys | Partitions | Grants | DDL

Column	Type	Default Value	Nullable	Character Set	Collation	Privileges	Extra	Comments
age	int		YES			select,insert,update,references		
email	varchar(50)		NO	utf8mb4	utf8mb4_0900_...	select,insert,update,references		
first_name	varchar(30)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references		
id	int		NO			select,insert,update,references	auto_increment	
last_name	varchar(30)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references		

Nuevamente podemos ver mediante **TABLE INSPECTOR**, que el campo **email** fue modificado satisfactoriamente

CODER HOUSE

OTRAS OPERACIONES CON ALTER TABLE

CHANGE COLUMN: podemos cambiar el nombre de una columna previamente definida.

RENAME TO: podemos cambiar el nombre inicial de una tabla por uno nuevo.

DROP COLUMN: podemos eliminar una columna o campo.

DDL: DROP

CODER HOUSE

DDL: DROP (TABLE)

La sentencia **DROP**, del inglés "*borrar*", nos permite eliminar una tabla

```
DROP TABLE friend;
```

PRECAUCIONES A TENER EN CUENTA

CODER HOUSE

DDL: DROP (TABLE)

Dado que este tipo de actividad es poco frecuente, recomendamos siempre contemplar estas dos opciones:

- utilizar la sentencia USE “schema”.
- clonar la tabla si tiene datos y luego eliminarla.

DDL: USE SCHEMA

En entornos de trabajo muchas veces saltamos entre diferentes ambientes de DB de forma constante, lo que puede ocasionar que nos encontremos en el ambiente erróneo al momento de eliminar la tabla.

```
USE GAMER;

DROP TABLE [tabla a eliminar];
```

Ejemplo en vivo

DDL: CLONAR TABLA

Si no estamos 100% seguros de si eliminar la tabla no afectará la necesidad de su información, podemos clonar, previamente, usando la sentencia:

CREATE TABLE ... LIKE ...

```
CREATE TABLE friend_backup

LIKE friend;
```

CODER HOUSE

TABLAS Y RELACIONES

Las relaciones entre tablas son un factor dominante al momento del uso de **DROP TABLE**. Si la tabla tiene definida una o más relaciones con otras, primero deberemos eliminarlas para luego proceder con DROP.

El motor de base de datos impedirá que podamos borrarla a menos que utilicemos la cláusula de DROP CASCADE

DDL: TRUNCATE

SENTENCIA DDL TRUNCATE

La sentencia TRUNCATE TABLE elimina todos los datos que estén almacenados dentro de la tabla definida.

Si bien existe la sentencia DELETE, propia de DML, TRUNCATE garantiza una mayor velocidad de borrado de datos.

SENTENCIA DDL TRUNCATE

La sentencia de esta es la siguiente

👇

```
USE “schema”;

TRUNCATE TABLE friend;
```

TABLAS Y RELACIONES

Al igual que lo visto con la sentencia **DROP**, las relaciones entre tablas son un factor dominante al momento del uso de **TRUNCATE TABLE.**

No podremos borrar datos que estén vinculados a otros datos de otra tabla

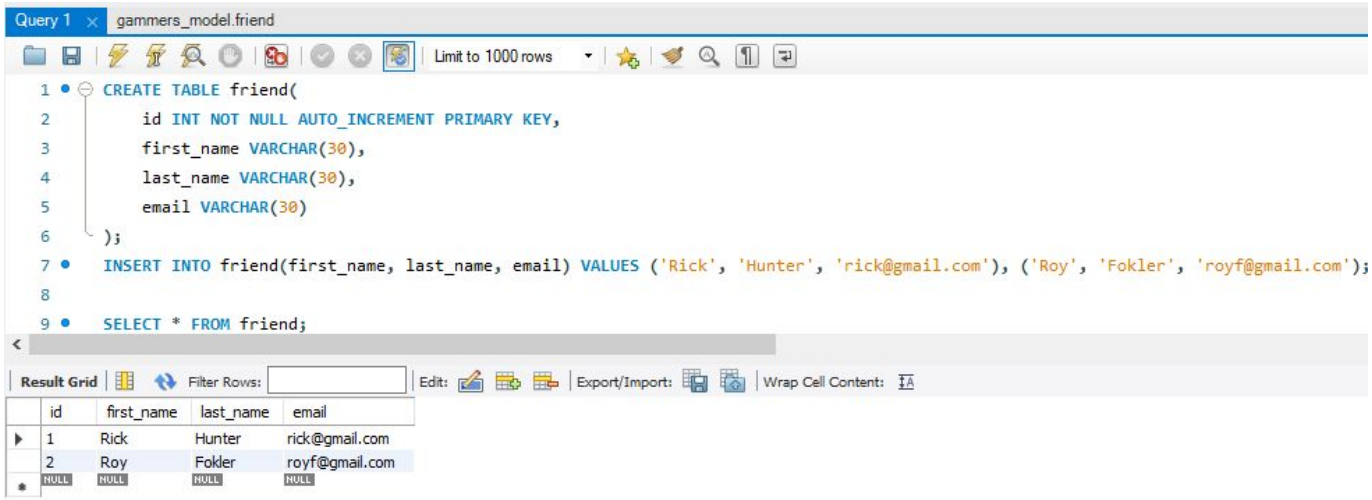
¡VAMOS A PRACTICAR UN POCO!

CODER HOUSE

SENTENCIA TRUNCATE EN ACCIÓN

Ejemplo en vivo

Creemos nuevamente la tabla friend e ingresemos datos a la tabla (también se puede utilizar la opción **insert new row**):



Luego, ejecutamos en una pestaña de script la sentencia **`TRUNCATE TABLE friend`**, para eliminarlos y así probar su efectividad.

CODER HOUSE

BREAK

¡5/10 MINUTOS Y VOLVEMOS!

FUNCIONES ESCALARES

CODER HOUSE

FUNCIONES ESCALARES

Al igual que los lenguajes de programación en general, SQL incluye una serie de funciones denominadas **Funciones Escalares.**

Permiten manipular datos cuando los recuperamos o antes de guardarlos, mediante operaciones predeterminadas, devolviendo un resultado específico acorde a lo esperado.

FUNCIONES ESCALARES: VENTAJAS

Algunas ventajas de implementarlas, son:

- Reducir el re-trabajo de la lógica comercial.
- Evitar la inconsistencia de datos que provenga de un software.
- Ayudar a reducir el tráfico de red de aplicaciones cliente/servidor.
- Mejorar en gran medida el rendimiento de los sistemas.

TIPOS DE FUNCIONES ESCALARES

Existen dos tipos de funciones escalares en Mysql:

- funciones integradas
- funciones almacenadas

Trabajaremos en esta instancia con las **funciones integradas.**

FUNCIONES INTEGRADAS

Se clasifican bajo las siguientes categorías:

- Funciones de cadenas.
- Funciones numéricas.
- Funciones de fecha.
- Funciones agregadas.

FUNCIONES DE CADENA

CODER HOUSE

FUNCIONES DE CADENA

Nos **permiten operar** con cualquier tipo de **cadena de caracteres** almacenada en una tabla (*o por almacenarse*).

👉 Podemos, entre otras cosas: *convertir el texto a mayúsculas, minúsculas, concatenar strings, cortar una porción del texto, eliminar espacios, revertir el texto, contar caracteres*, entre decenas de más funciones

EJEMPLOS: CONCAT()

Fusiona cadenas de caracteres en un único bloque de datos.

```
SELECT concat(first_name, last_name)
       AS complete_name
FROM system_user;
```

Podemos, por ejemplo, unificar en un campo llamado **complete_name**, los campos **first_name** y **last_name** de la tabla **SYSTEM_USER.**

EJEMPLOS: UCASE() / LCASE()

Convierte a mayúsculas o minúsculas (*respectivamente*) una cadena de texto.

```
SELECT UCASE(description) FROM class;

-- devolverá, por ejemplo: “ACTION”

SELECT LCASE(description) FROM class;

-- devolverá, por ejemplo: “action”
```

EJEMPLOS: REVERSE()

Revierte el orden de los caracteres de una cadena de texto.

```
SELECT REVERSE(description) FROM class;


-- devolverá, por ejemplo: “noitcA”
```

CODER HOUSE

FUNCIONES DE CADENA

Aquí tienes otras opciones para el manejo de caracteres:

- **`TRIM()`**: elimina los espacios vacíos en los extremos de un texto.
- **`SPACE()`**: cuenta la cantidad de espacios en un bloque de texto.
- **`CHAR_LENGTH()`**: cuenta los caracteres de un bloque de texto.
- **`SUBSTRING()`**: extrae uno o más caracteres de un bloque de texto.
- Y muchas más...

Puedes ver todas las opciones en la web oficial de Mysql.

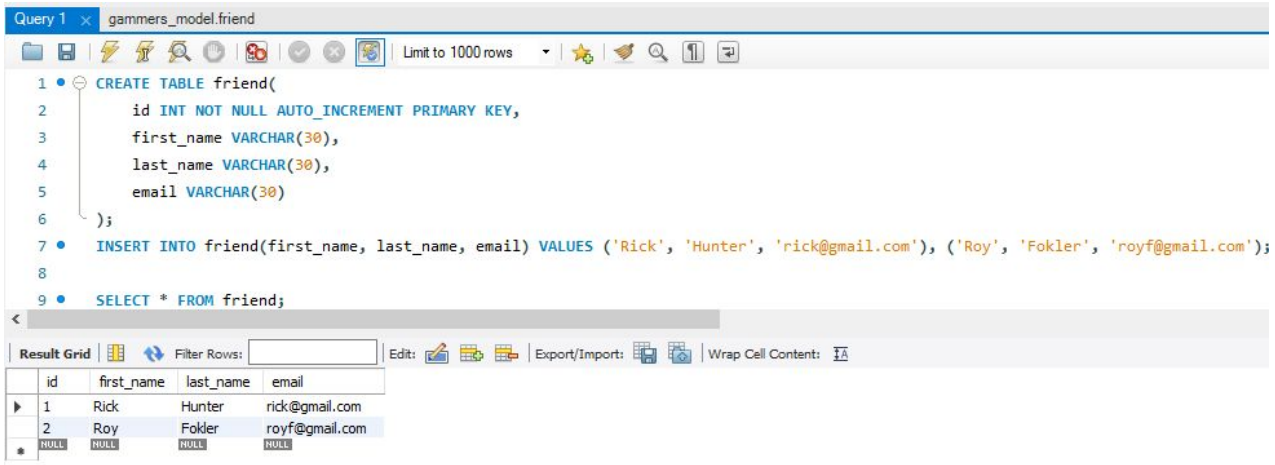
¡VAMOS A PRACTICAR UN POCO!

CODER HOUSE

FUNCIONES EN STRINGS

Ejemplo en vivo

Utilizamos nuevamente la tabla friend e insertamos registros:



Ahora probemos cómo accionan algunas de las diferentes funciones escalares sobre los datos cargados.

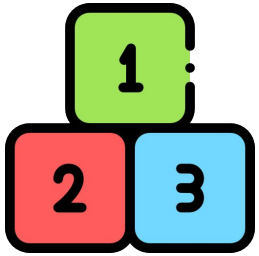
CODER HOUSE

FUNCIONES NUMÉRICAS

CODER HOUSE

FUNCIONES NUMÉRICAS

Nos **permiten operar** con cualquier tipo de **número**, a su vez, se subdividen en dos segmentos:



- **operadores aritméticos:** para realizar operaciones matemáticas básicas.
- **funciones matemáticas:** para realizar conversiones y otras operaciones con números de mayor complejidad.

EJEMPLO: OPERACIONES ARITMÉTICAS

Podemos realizar operaciones aritméticas, utilizando la simbología común, a través de la estructura:

(**número *operador* número**).

El resultado se verá en un campo calculado.

```
-- División

SELECT (21 / 3) AS resultado;

-- Multiplicación

SELECT (7 * 3) AS resultado;

-- Suma

SELECT (18 + 3) AS resultado;

-- Resta

SELECT (30 - 9) AS resultado;
```

EJEMPLO: FUNCIONES MATEMÁTICAS

- **`log()`**, **`log2()`**, **`log10()`**: cálculo de logaritmos, base 2 y base 10.
- **`round()`**: redondeo estándar de un número.
- **`floor()`**: redondeo de un número hacia abajo.
- **`ceiling()`**: redondeo de un número hacia arriba.
- **`truncate()`**: elimina los decimales de un número.

Más otras tantas funciones más. Puedes consultarlas aquí.

FUNCIONES DE FECHA

CODER HOUSE

FUNCIONES DE FECHAS

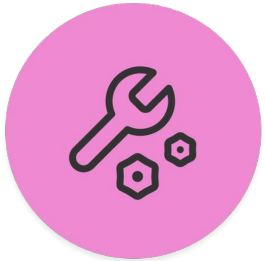
Podemos manipular cualquier tipo de cálculo con fechas:

- Obtener los días ocurridos entre determinadas fechas.
- El número del día de un año.
- Extraer el mes, el año, o día de la fecha actual.
- Saber qué día de la semana fue una determinada fecha

ALGUNAS FUNCIONES DE FECHA

- **`curdate()`**: devuelve la fecha actual.
- **`curtime()`**: devuelve la hora actual.
- **`now()`**: combina los dos anteriores en un resultado.
- **`datediff()`**: obtiene la diferencia de tiempo entre dos fechas.
- **`dayname()`**: Retorna el nombre del día de semana de una fecha determinada.

Más otras tantas funciones más que puedes consultarlas aquí.



IMPLEMENTAR FUNCIONES ESCALARES

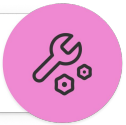
Trabajamos con algunas funciones.

Tiempo estimado: 15 minutos

CODER HOUSE

IMPLEMENTAR FUNCIONES ESCALERAS

Desafío genérico



Abrir una pestaña de consulta (*query tab*), y ejecuta las siguientes funciones:

- concatena tu nombre completo (*respetando los espacios*)
- convierte tu nombre completo a minúsculas, luego a mayúsculas
- Divide tu año de nacimiento por tu día y mes (*ej: 1975 / 2103*)
- Convierte en un entero absoluto el resultado anterior
- Calcula los días que pasaron desde tu nacimiento hasta hoy
- Averiguar qué día de semana era cuando naciste

¿PREGUNTAS? 

¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Lenguaje DDL.
- Funciones escalares .
- Funciones de transformación.

OPINA Y VALORA ESTA CLASE

#DEMOCRATIZANDOLAEDUCACIÓN

CODER HOUSE