MYSQL ---LENGUAJE DE DESCRIPCIÓN DE DATOS

```
CREATE DATABASE Nombre_bd;
```

```
CREATE TABLE Nombre_Tabla
(
Nombre_campo Tipo_campo [NOT NULL][UNIQUE][PRIMARY KEY][CHECK],
…
[CONSTRAINT Nombre] PRIMARY KEY/FOREIGN KEY...REFERENCES...
[ON DELETE CASCADE/SET NULL/SET DEFAULT ON UPDATE CASCADE/SET NULL/…]
)
```

```
ADD [COLUMN] NOMBRE_CAMPO TIPO CAMPO [RESTRICCIONES]          MODIFY
NOMBRE_CAMPO TIPO CAMPO [TYPE] [RESTRICCIONES]
DROP COLUMN NOMBRE_CAMPO
```

BORRAR LA BASE DE DATOS/ TABLAS

```
DROP DATABASE NOMBRE_BD

DROP TABLE NOMBRE_TABLA
```

CREAR/ELIMINAR PK (UNA VEZ HECHA LA TABLA)

```
ALTER TABLE NOMBRE_TABLA
ADD CONSTRAINT NOMBRE_PK PRIMARY KEY(CAMPO)

ALTER TABLE NOMBRE_TABLA
DROP TIPO_CONSTRAINT NOMBRE_PK
```

CREAR/ELIMINAR FK (UNA VEZ HECHA LA TABLA)

AÑADIR FK:

```
ALTER TABLE NOMBRE_TABLA
ADD CONSTRAINT NOMBRE_FK FOREIGN KEY(CAMPO) REFERENCES NOMBRE_TABLA
(CAMPO TABLA REFERENCIADA)
ON UPDATE [OPCIÓN DE MODIFICACIÓN] ON DELETE [OPCIÓN DE BORRADO]
(SI NO PONEMOS NADA SE TOMA LA OPCIÓN POR DEFECTO NO ACTION)
```

BORRAR FK:

```
ALTER TABLE NOMBRE_TABLA
DROP CONSTRAINT NOMBRE de la FK /
DROP FOREIGN KEY NOMBRE de la FK
```

```
INSERT INTO NOMBRE_TABLA [LISTA DE CAMPOS]
VALUES (VALOR_CAMPO1, VALOR_CAMPO2, …),   INSERTA LA PRIMERA FILA
       (VALOR_CAMPO1, VALOR_CAMPO2, …),   INSERTA LA SEGUNDA FILA
       …
       ;
SOLO PONEMOS VALUES UNA VEZ, AUNQUE SE INSERTEN VARIAS FILAS
```

```
UPDATE NOMBRE_TABLA
SET CAMPO=NUEVO_VALOR   MODIFICARÍA TODAS LAS FILAS DE LA TABLA
-------
 UPDATE NOMBRE_TABLA
SET CAMPO=NUEVO_VALOR
WHERE CONDICIÓN;    INSERTA SOLO LAS FILAS QUE CUMPLEN LA CONDICIÓN.
```

```
DELETE FROM NOMBRE_TABLA   BORRARÍA TODA LA TABLA
DELETE FROM NOMBRE_TABLA
WHERE CONDICIÓN   BORRA SOLO LAS FILAS QUE CUMPLEN LA CONDICIÓN
```

CREAR UN ÍNDICE EN UNA TABLA

```
CREATE INDEX NOMBRE_INDEX
ON NOMBRE_TABLA (NOMBRE_CAMPO)   CREARÍA UN ÍNDICE SOBRE UN SOLO
CAMPO
CREATE INDEX NOMBRE_INDEX
ON NOMBRE_TABLA (NOMBRE_C<AMPO1, NOMBRE_CAMPO2)   CREARÍA UN
ÍNDICE SOBRE LA COMBINACIÓN DE ESOS DOS CAMPOS)
```

MYSQL --LENGUAJE DE MANIPULACIÓN DE DATOS

```
SELECT [DISTINCT ] <expr1>, <expr2>, ....
FROM  NOMBRETABLA
WHERE <predicado>
GROUP BY <expr1>, <expr2>, ....
HAVING <expr1>, <expr2>, ....
ORDER BY <expr1>, <expr2>, ....
LIMIT
SELECT CAMPO AS NUEVO_NOMBRE
FROM NOMBRETABLA;   1 PALABRA O DOS SIN ESPACIOS, NO PONEMOS COMILLAS
SELECT CAMPO AS "NUEVO NOMBRE"
FROM NOMBRETABLA ;  2 PALABRAS CON ESPACIO, PONEMOS COMILLAS DOBLES
```

FUNCIONES DE AGREGADO:

```
SELECT MAX (CAMPO) [AS NUEVO_NOMBRE].
FROM NOMBRETABLA;

SELECT MIN (CAMPO) [AS NUEVO_NOMBRE].
FROM NOMBRETABLA;

SELECT COUNT (CAMPO) / SELECT COUNT (CAMPO)
FROM NOMBRETABLA;

SELECT AVG (CAMPO)
FROM <TABLA >;

SELECT SUM (CAMPO)
FROM NOMBRETABLA;
```

ORDENAR REGISTROS:

```
SELECT *
FROM NOMBRETABLA
ORDER BY CAMPO [ASC/DESC];
```

SELECCIONAR REGISTROS CON CARACTERES COMODÍN

```
SELECT *
FROM NOMBRETABLA
WHERE CAMPO LIKE '%LETRA' / 'LETRA%' / '%LETRA%'
```

FUNCIONES DE FECHA:
SELECT YEAR/MONTH/DAY (CAMPO_FECHA)

FUNCIONES CURRENT_DATE, ROUND, TRUNC, MOD

SELECT CURRENT_DATE() Muestra la fecha de hoy
ROUND: redondea al número más próximo.
SELECT ROUND(PRECIO, 2) Redondea precio al número más próximo con 2 decimales.
TRUNC: Trunca el número sin redondear.
SELECT TRUNC(PRECIO, 0) Trunca precio sin decimales.
MOD: Devuelve el resto de una división.
SELECT MOD(DIVIDENDO,DIVISOR) FROM…

COMPOSICIONES DE TABLAS:

```
SELECT CAMPO1, CAMPO2,…CAMPO N
FROM TABLA1 INNER JOIN TABLA2 ON TABLA1.CAMPO=TABLA2.CAMPO
                INNER JOIN TABLA3 ON TABLA2.CAMPO=TABLA3.CAMPO
WHERE CONDICIÓN1=….. AND/OR CONDICIÓN2=….
GROUP BY (….)
HAVING CONDICIÓN (…….)
ORDER BY …. ASC/DESC
```

INNER JOIN CON LA MISMA TABLA (RELACIONES REFLEXIVAS) Es necesario RENOMBRAR la tabla para hacer el INNER JOIN.

```
SELECT TABLA.CAMPO1, CAMPO2, ALIAS.CAMPO3…CAMPO N
FROM TABLA INNER JOIN ALIAS ON TABLA.CAMPO=ALIAS.CAMPO
```

COMPOSICIONES EXTERNAS:

LEFT JOIN (Aparecen todos los registros de la tabla izquierda, aunque no tengan correspondencia con ningún registro de la tabla de la derecha)

RIGHT JOIN (Aparecen todos los registros de la tabla derecha, aunque no tengan correspondencia con ningún registro de la tabla de la izquierda)

```
SELECT CAMPO1,..CAMPO N
FROM TABLA1 RIGTH JOIN TABLA2 ON TABLA1.CAMPO=TABLA2.CAMPO
```

PROGRAMACIÓN DE BASES DE DATOS-MySQL:

BORRAR una foreign key (FK) en MySQL:
DROP FOREIGN KEY Nombre

Estructura condicional: las instrucciones se ejecutan dependiendo de si se cumple o no una determinada condición. Puede utilizarse dentro de un procedimiento almacenado o de una función:

```
IF condicion  THEN
        instrucciones SQL
[ ELSE
        instrucciones SQL
```

Estructura condicional MÚLTIPLE: SENTENCIA CASE

```
CASE VALOR
    WHEN VALOR1 THEN  instrucciones SQL
    [WHEN VALOR2 THEN instrucciones SQL ...
[ELSE instrucciones SQL]
END CASE
```

DECLARAR UNA VARIABLE LOCAL en MySQL:
DECLARE Nombre TIPO

CAMBIO DE DELIMITADORES

```
DELIMITER    //   -- NO OLVIDAR UN ESPACIO ANTES DE //---
//   CERRAR EL DELIMITADOR
DELIMITER    ;    -- activar el delimitador por defecto. ---
                                - NO OLVIDAR UN ESPACIO ANTES DE ;---
```

TIPOS DE PARÁMETROS EN PROCEDIMIENTOS Y FUNCIONES MySQL:
En los procedimientos se pueden utilizar parámetros IN (entrada), OUT (salida) o INOUT (entrada/salida). Es necesario especificarlo.
En las funciones solo se utilizan parámetros IN y no hace falta especificarlo

Algunas FUNCIONES ÚTILES PARA USAR EN PROCEDIMIENTOS Y FUNCIONES:

- upper(CAMPO) convierte una cadena a mayúsculas.
- lower (CAMPO) convierte una cadena a minúsculas.
- length (CAMPO) cuenta los caracteres del campo.
- sqrt(4) calcula la raíz cuadrada del parámetro
- DATEDIFF(fecha1,fecha2) devuelve la diferencia en días entre dos fechas.
- CURRENT_DATE(fecha) o CURDATE(FECHA) devuelve la fecha actual
- ROUND(X) redondea sin decimales
- ROUND(X,D) redondea con d decimales

PROCEDIMIENTOS ALMACENADOS

Crear un procedimiento almacenado:

```
CREATE PROC (O PROCEDURE) Nombre ([parámetros])
instrucciones SQL
```

Llamar al procedimiento:
CALL NOMBRE_PROCEDIMIENTO ([parámetros])

BORRAR un procedimiento almacenado:
DROP PROCEDURE Nombre

FUNCIONES

```
CREATE FUNCTION Nombre (parámetros de la función)
RETURNS tipo_dato_valor_retorno
BEGIN
--instrucciones SQL que se ejecutarán en la función—
RETURN valor_retorno
END
```

Llamar a una función: Se llamará a la función en una cláusula SELECT igual que si se tratase de un campo de una tabla. También puede llamarse en un WHERE.
SELECT NOMBRE_FUNCTION ([parámetros])

BORRAR una función:
DROP FUNCTION Nombre