—---------- DDL Lenguaje de Definicion de Datos ----------

```
CREATE DATABASE nombreBD  ;
DROP   DATABASE nombreBD;

USE nombreBD;
                        ------------------------------------------------------------------------
CREATE TABLE nombreTable (
          campo1 INT NOT NULL,
          campo2  VARCHAR(45) NOT NULL,
 PRIMARY KEY (campo1)
);
```

ALTER TABLE ... ADD: Agrega una columna:

```
ALTER TABLE nombre_de_tabla
ADD nombre_de_columna tipo de dato;
```

ALTER TABLE ... DROP: Elimina una columna:

```
ALTER TABLE nombre_de_tabla
DROP COLUMN nombre_de_columna;
```

```
CHANGE COLUMN
CHANGE COLUMN `nombre` `nombre` VARCHAR(200) NOT NULL ;
```

```
DROP TABLE nombreTabla;
```

—---------- DML Lenguaje de Modificacion de Datos ----------

```
INSERT INTO nombreTable
 (id, campo2,,,,,campon)
VALUES (1, value2,,,,,valuen);

INSERT INTO nombreTable    /* si el id es Auto incremento*/
 ( campo2,,,,,campon)
VALUES ( value2,,,,,valuen);

INSERT INTO nombreTable  VALUES (value1, value2,,,,,valuen);

UPDATE tabla  SET
        campo1=value1  ,
        campo2=value2
WHERE condicion;

DELETE FROM  nombreTable  WHERE condicion

SELECT    *
```

FROM nombreTable ;

Lo encerrado entre corchetes [ ] es opcional
SELECT [*,] campos [**AS** nuevoNombreColumna]
FROM nombreTable [**AS** nuevoNombreTabla]
[**WHERE** condicion]
[**ORDER BY** campos] [ASC] [DESC}
[**LIMIT** numero]

;—----------

/* clase 3 mysql */

SELECT [*,] campos [**AS** nuevoNombreColumna]
FROM nombreTable [**ALIAS** nuevoNombreTabla]
[**WHERE** condicion]
[**ORDER BY** campos] [ASC] [DESC}
[**INNER/LEFT/RIGTH JOIN** otratabla **ON** condicion] /* clase 3 mysql */
[**GROUP BY** campos] /* clase 3 mysql */
[**LIMIT** numero] **;**

Operador	Descripción
=	Igual
<>	Diferente
!=	Diferente
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que

Operador	Descripción
LIKE	Define un patrón de búsqueda y utiliza % y _

NOT LIKE	Negación de LIKE
IS NULL	Verifica si el Valor es NULL
IS NOT NULL	Verifica si el Valor es diferente de NULL
IN ()	Valores que coinciden en una lista
BETWEEN	Valores en un Rango (incluye los extremos)

```
SELECT codigo FROM productos
WHERE descripción IN ('Harina' ,'Azúcar' ,'Leche')

SELECT * FROM clientes c
WHERE calle LIKE '%San Martín%'    /*like */

SELECT V.precio,(V.precio * 1.21) AS precio_con_iva
FROM ventas AS V   /*alias */
```

```
SELECT campo1, campo2, ...,campoN FROM tabla1
    JOIN tabla2 ON tabla1.campo1 = tabla2.campo2
    JOIN tabla3 ON tabla2.campo3 = tabla3.campo4
```

Las **funciones de agregación** más comunes disponibles en el lenguaje son: **SUM(), AVG(), MAX(), MIN(), COUNT()**.

```
SELECT month(p.fecha) as Mes, SUM(cant) as Cantidad
FROM pedidos_productos pp
        JOIN productos pr ON pp.codproducto=pr.codigo
```

```
        JOIN pedidos p    ON p.nro=pp.codpedido
WHERE year(p.fecha)=2017
GROUP BY month(p.fecha)

SELECT p.Nro, SUM(cant*precio) as total
FROM pedidos_productos pp
        JOIN productos pr ON pp.codproducto=pr.codigo
        JOIN pedidos p ON p.nro=pp.codpedido
GROUP BY p.nro
HAVING SUM(cant*precio)>1000  /*HAVING Cuando tengo
que filtrar sobre funciones agregadas*/
```

Funciones de fecha

YEAR(d): Devuelve el año correspondiente de la fecha “d”.

MONTH(d): Devuelve el mes de la fecha “d”.

DAY(d): Devuelve el día del mes de la fecha “d”.

DATE_ADD(): Agrega valores de tiempo (intervalos) a un valor de fecha.

```
DESCRIBE nombreTabla;
```