```sql
SELECT * FROM peliculas; /* MUESTRA TODA LA TABLA */
    /* CAMPOS POR SEPARADO */
SELECT peliculas.titulo, peliculas.anio FROM peliculas;
SELECT anio, titulo FROM peliculas;
SELECT titulo FROM peliculas;

        /* ACCEDIENDO A LA INFORMACION */
SELECT * FROM peliculas WHERE anio = 1999;
SELECT * FROM peliculas WHERE anio != 1999;
SELECT * FROM peliculas WHERE anio > 1999;
SELECT * FROM peliculas WHERE anio >= 1999;
SELECT * FROM peliculas WHERE anio < 1999;
SELECT * FROM peliculas WHERE anio <= 1999;
SELECT * FROM peliculas WHERE anio = 1999 AND titulo = "The Matrix";
SELECT * FROM peliculas WHERE anio = 1998 OR anio = 2000;
SELECT * FROM peliculas WHERE anio BETWEEN 1998 AND 2000; /* BETWEEN (Rango) */
           /* LIKE(Que contenga la palabra) */
SELECT * FROM peliculas WHERE titulo LIKE "godfather";
SELECT * FROM peliculas WHERE titulo LIKE "%godfather";
SELECT * FROM peliculas WHERE titulo LIKE "%godfather%";

           /* ORDENAMIENTO  */
SELECT * FROM peliculas ORDER BY anio;
SELECT * FROM peliculas ORDER BY anio DESC;
SELECT * FROM peliculas ORDER BY anio ASC;
SELECT * FROM peliculas ORDER BY anio ASC, titulo DESC;

        /* ORDENAMIENTOS */
SELECT * FROM peliculas LIMIT 10;           /* MUESTRA 10 RESULTADOS */
SELECT * FROM peliculas LIMIT 10 OFFSET 1;  /* OFFSET INDICA # DE PAGINAS O COLUMNAS */
SELECT * FROM peliculas LIMIT 10 OFFSET 0;  /* SIEMPRE EMPIESA EN LA PAGINA O COLUMNA 0 */
SELECT * FROM peliculas LIMIT 10 OFFSET 10;
SELECT * FROM peliculas LIMIT 10 OFFSET 20;
SELECT * FROM peliculas LIMIT 10 OFFSET 250;
SELECT * FROM peliculas LIMIT 20, 10;           /* SE INVIERTE SE MOSTRARA  PAG 20 , 10 RESULTADOS */

SELECT * FROM peliculas LIMIT 0, 1000;
SELECT * FROM actors LIMIT 0, 1000;
           /* CAMPOS NULL */
SELECT * FROM peliculas WHERE anio IS NULL;     /* MUESTRA CAMPO NULL O VACIO */
SELECT * FROM peliculas WHERE anio IS NOT NULL ORDER BY anio;   /* EXCLUYE CAMPOS NULL Y LOS ORDENA */
```

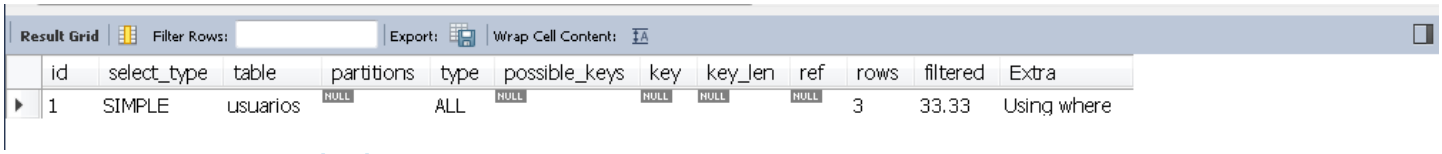
```sql
USE prueba_bd;  /* pociciona en la BD a usar */
/* Documentacion: www.dev.mysql.com  - Documentation */

            /*  CREAR BASE DE DATOS */
CREATE SCHEMA prueba_db_3;
CREATE DATABASE prueba_db_4;
CREATE SCHEMA prueba_bd_3 DEFAULT CHARACTER SET utf8; /* utf8 para caracteres especiales */

            /*  CREAR TABLAS */
CREATE TABLE pelicula(titulo VARCHAR(100), anio INTEGER) ENGINE InnoDB;
CREATE TABLE actores(nombre VARCHAR(50)) ENGINE InnoDB;
CREATE TABLE pelicula(titulo VARCHAR(100) NOT NULL, anio INTEGER NULL) ENGINE InnoDB;
CREATE TABLE actores(nombre VARCHAR(50) NOT NULL) ENGINE InnoDB;        /* ENGINE OPCIONAL SELECCIONA EL MOTOR DE BD A USAR */

SHOW ENGINES;   /* MUESTRA LOS MOTORES DE BASE DE DATOS */

            /* INSERTAR */
INSERT INTO peliculas VALUES ("The Matrix" , 1999);
INSERT INTO peliculas ( titulo , anio ) VALUES ("The Goodfather" , 1999);     /* SE PUEDE INVERTIR EL ORDEN DE LOS CAMPOS */
INSERT INTO peliculas ( titulo , anio ) VALUES ("The Matrix Reloader" , 1999) , ("The Matrix 4" , 2020) ;
INSERT INTO peliculas SET titulo = "the matrix revolution" , anio = 2003;

                /* ACTUALIZAR */
SET SQL_SAFE_UPDATES = 0;     /* DESACTIVA LA PROTECCION PARA MODIFICAR */
UPDATE peliculas SET titulo = "The Matrix is Back" , anio = 2014 WHERE titulo = "The Matrix 4";

DELETE FROM peliculas WHERE titulo = "The Matrix is Back" and anio = 2014;      /* PUEDE SER SOLO UN CAMPO */
```

```
            /* RENOMBRAR TABLAS */
RENAME TABLE peliculas TO peliculas_tabla , actores TO actores_tabla;
RENAME TABLE peliculas TO peliculas_tabla;

DROP TABLE peliculas_tabla;  /* ELIMINA COMPLETAMENTE LA TABLA */

TRUNCATE TABLE actores_tabla;   /* ELIMINA EL CONTENIDO DE UNA TABLA */

                    /*  ALTERAR  */
        /* AGREGAR COLUMNAS */
ALTER TABLE peliculas_tabla ADD COLUMN genero varchar (100);
ALTER TABLE actores_tabla ADD (ldn VARCHAR(100) , fdn DATE );

        /* MODIFICAR COLUMNAS */
ALTER TABLE actores_tabla CHANGE COLUMN ldn  lugar_de_nacimiento VARCHAR(100);
ALTER TABLE actores_tabla CHANGE fdn  fecha_de_nacimiento DATE;
ALTER TABLE peliculas_tabla CHANGE COLUMN anio  anio_de_estreno YEAR;

            /* ELIMINAR COLUMNAS */
ALTER TABLE actores_tabla DROP fecha_de_nacimiento;

        /* ELIMINAR BASE DE DATOS */
DROP DATABASE IF EXISTS prueba_bd_3; #IF EXISTS deja correr sistema si BD no existe
DROP SCHEMA IF EXISTS prueba_bd2;  /* SCHEMA = DATABASE */
```

```sql
/*
NORMALIZACION: Es la optimisacion de los datos mara manejar la informacion de manera
mas optima evitando la redundancia y ahorrando espacio  */

/* CREA TABLA CON LLABE PRIMARIA Y LLABE UNICA */
CREATE TABLE genero (
id INTEGER AUTO_INCREMENT PRIMARY KEY ,
nombre VARCHAR(50) NOT NULL UNIQUE KEY); /* CLAVE UNICA PARA QUE NO SE REPITA*/
INSERT INTO genero  /*INCERTANDO VALORES*/
(nombre) VALUES ("Drama");

ALTER TABLE peliculas /* AGREGANDO COLUMNA id*/
ADD COLUMN id INTEGER AUTO_INCREMENT PRIMARY KEY FIRST; #FIRST PONE LA COLUMNA EN PRIMER LUGAR

        /*LLABE FORANEA PERMITE ENLAAR A LAS COLUMNAS DE DISTINTAS TABLAS*/
ALTER TABLE peliculas
ADD COLUMN genero_id INTEGER NULL ,
ADD CONSTRAINT FOREIGN KEY (genero_id) REFERENCES genero(id); #CONSTRAINT RESTRINGE A LO QUE HAY EN CONTENIDO DE genero(id)

UPDATE peliculas SET genero_id = 1 WHERE id = 8 OR id = 9;  /*SE AGREGA GENERO A PELICULAS CON id 8 o 9*/

            /*UNIMOS LAS DOS TABLAS peliculas y gener unidas con JOIN*/
SELECT * FROM peliculas JOIN genero ON peliculas.genero_id = genero.id; /*ON donde se van a unir */
SELECT * FROM peliculas INNER JOIN genero ON peliculas.genero_id = genero.id;
SELECT * FROM peliculas
LEFT OUTER JOIN genero ON peliculas.genero_id = genero.id; /* CANBIA EL ORDEN DE LA INFORMACION QUE VAMOS A MOSTRAR*/
SELECT * FROM peliculas
RIGHT OUTER JOIN genero ON peliculas.genero_id = genero.id; #EN ESTE CASO SE MUESTRA TABLA GENERO CON LAS PELIS ASOCIADAS

        /*ALIAS AS*/
SELECT peliculas.titulo AS titulo_de_pelicula , genero.nombre AS genero /*PONE UN ALIAS A LOS TITULOS DE LAS COLUMNAS*/
FROM peliculas LEFT OUTER JOIN genero
on peliculas.genero_id = genero.id
WHERE genero.id IS NOT NULL ; /* SE MUESTRAN LAS PELICULAS QUE TIENEN UN GENERO*/
```

```sql
            /*  FUNCIONES  */

/*  COUNT -> CUENTA NUMERO DE REJISTROS
    MIN -> NUMERO MINIMO
    MAX -> NUMERO MAXIMO
    SUMA -> SUMA TODA LA COLUMNA
    AVG -> PROMEDIO
*/
SELECT COUNT(*) AS Num_de_Opiniones FROM opiniones where pelicula_id =1;

SELECT MIN(puntuacion) AS min_puntaje ,
MAX(puntuacion) AS max_puntaje ,
SUM(puntuacion) / COUNT(puntuacion) AS puntaje
FROM opiniones where pelicula_id =1;

SELECT AVG(puntuacion) FROM opiniones where pelicula_id =1; /*AVG saca el promedio directo*/

SELECT titulo , MIN(puntuacion) AS min_puntaje ,
MAX(puntuacion) AS max_puntaje ,
IFNULL(AVG(puntuacion) , 0) AS Puntaje   /*SI PUNTUACION ES NULL PROMEDIO = 0 */
FROM peliculas LEFT OUTER JOIN opiniones  /* LEFT OUTER PARA QUE APARESCAN CAMPOS NULL */
ON peliculas.id = opiniones.pelicula_id
WHERE anio_estreno > 2000  /* WHERE ANTES DE GRUP BY HAVING DESPUES DE GRUP BY */
GROUP BY pelicula_id HAVING Puntaje > 3; /*GRUP BY AGRUPA LA INFORMACION Y HAVING CLASIFICA/*

/*
    LOWER -> TEXTO EN MINUSCULAS
    UPPER -> TEXTO EN MAYUSCULAS
    LENGTH -> CUENTA CARACTERES
    CONCAT -> CONCATENA
    SUBSTRING -> CONTROLA QUE CANTIDAD DE CARACTERES SE MOSTRARAN
*/
SELECT UPPER(primer_nombre), LOWER(apellido), email, LENGTH(nombre_usuario) AS Nombre_Usuario
FROM usuarios HAVING Nombre_de_usuario < 7;

    /* CONCATENANDO NOMBRE COMPLETO*/
SELECT CONCAT(primer_nombre , " " , apellido) FROM usuarios;
SELECT SUBSTRING(email, 1,3)FROM usuarios;  /* SUBSTRING */
    /* SUBSTRING CONCATENADO */
SELECT CONCAT(SUBSTRING(email, 1,3) , "*******") AS E_mail FROM usuarios;
```

```sql
/* INDEXACION DE COLUMNAS */
/*
Al utilisar indices se relentisa un poco el proseso por lo que hay que usarlos solo en las
columnas necesarias dende se tiene una crga mayor de trabajo.   */

/* EXPLAIN MUESTRA LOS PROSESOS QUE REALISA EN SEGUNDO PLANO*/
EXPLAIN SELECT * FROM usuarios WHERE apellido = "Rodriguez";

/* En la imagen vemos que se estan recorriendo todas las columnas */
```



id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	usuarios	NULL	ALL	NULL	NULL	NULL	NULL	3	33.33	Using where

```sql
/* Creando indice*/
CREATE INDEX Apellido_Index on usuarios(apellido);
/*Donde usuario es la tabla y apellido la columna a la que se le hace indice*/

EXPLAIN SELECT * FROM usuarios WHERE apellido = "Rodriguez";

SELECT * FROM usuarios WHERE apellido = "Rodriguez";
/*Se muestra la indexacion y ya no se recorren todas las columnas*/

# Creando usuarios

# Usuario 1 solo lectura
GRANT SELECT    /*LOS PRIVILEJIOS QUE VA A TENER*/
ON peliculas_bd_1.* /*LA BD A LA QUE AXEDE EL " * " PARA QUE ACCEDA A TODAS LKAS TABLAS*
TO usuariox'%'  /*EL '%' PARA QUE ACCEDA DE CUALQUIER TIPO DE CONEXION*/
IDENTIFIED BY 'contrasena'; /*CLAVE CON LA QUE INGRESARA*/

FLUSH PRIVILEGES;  /*COMO ESTANDAR PARA GARANTISAR SE APLIQUEN LOS PRIVILEJIOS*/

# CREAR USUARIO2 LECTURA Y ESCRITURA
GRANT SELECT , INSERT , UPDATE , DELETE
ON peliculas_bd_1.*
TO usuario2@'%'
IDENTIFIED BY 'contrasena2';

FLUSH PRIVILEGES;

# usuario 3 DDL PUEDE MODIFICAR LA BASE DE DATOS
GRANT ALTER , CREATE , DROP
ON peliculas_bd_1.*
TO usuario3@'%'
IDENTIFIED BY 'CONTRASENA4';

FLUSH PRIVILEGES;
```