```sql
use sakila;


-- find out all the film which contains "Love" in the title
-- %love%

-- case insensitive : like operator by default
select * from film
where title like '%love%';

-- case sensitive
select * from film
where title like binary '%LOVE%';

-- HW : collate

update film
set title = 'Moh%hit'
where film_id = 458;

select * from film where film_id =458;

-- escape characters : '\'
select * from film
where title like '%\%%';

select distinct release_year from film;

-- find all of those films where release_year is null

select * from film where release_year = null;

SELECT 3 = 3;

SELECT NULL = NULL;

-- null != anything
-- empty basket = empty brain ?

-- IS NULL
select * from film where release_year IS NULL;
```

```sql
select * from film where release_year IS NOT NULL;

-- find all the films where the release_year is not 2006

select * from film where release_year != 2006;

-- select * from film where release_year is not 2006 ;

select * from film where release_year != 2006 or release_year is NULL;


-- find films but give me only 100 films
-- limit : restricts the number of rows we are returning
select * from film limit 100;

-- offset : no of rows to skip

select * from film limit 100 offset 10;

select * from film order by rental_rate desc limit 10;

-- UPDATE table_name
-- SET column_name = value
-- WHERE condition;

select distinct rental_rate from film;

-- update the release_year to 2010 for all of the movies which has
rental_rate = 2.99

select distinct release_year from film
where rental_rate = 2.99;

update film
set release_year = 2010
where rental_rate = 2.99;

select count(*) from film where rental_rate = 2.99;


-- Error Code: 1175. You are using safe update mode and you tried to update
a table without a WHERE that uses a KEY column.  To disable safe mode,
toggle the option in Preferences -> SQL Editor and reconnect.
```

```sql
SHOW VARIABLES LIKE 'sql_safe_updates';

set sql_safe_updates = 0;

-- delete from table_name
-- where condition;

select * from film;

select count(*) from film where release_year = 2012;
select * from film where release_year = 2012;

delete from film
where release_year = 2012;

-- DELETE VS TRUNCATE VS DROP

SELECT * FROM BATCHES;

DELETE FROM BATCHES;

INSERT INTO BATCHES (BATCH_NAME)
VALUES ('BATCH X1'), ('BATCH X2');

-- delete goes row by row and just delete the rows
-- it doesn't change table attributes
-- it doesn't rest auto-increment
-- delete allows you to roll back


START TRANSACTION;

SELECT * FROM BATCHES;

DELETE FROM BATCHES;

ROLLBACK;

COMMIT;


-- TRUNCATE
```

```sql
TRUNCATE BATCHES;
-- it basically removes all of the rows at once
-- auto-increment also resets
-- no rollback is allowed


START TRANSACTION;

TRUNCATE BATCHES;

SELECT * FROM BATCHES;

ROLLBACK;

-- DROP

drop table students;

drop table batches;

select * from batches;
select * from students;

-- Error Code: 1146. Table 'sakila.batches' doesn't exist

commit;

delete from batches;

rollback;

drop table batches;
truncate batches;
delete from batches;
drop table students;
```