MySQL_Commands

Saturday, November 13, 2021 12:09 AM

Table of contents

Commands | **Output/Exp**

```
show databases; #shows available databases
```

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| sys                |
| world              |
+--------------------+
6 rows in set (0.00 sec)
```

```
create database test_database; # create database
```

```
drop database test_database; #delete database
```

```
use test_database; #use the selected database
```

```
select database(); # tells the currently selected database
```

```
+------------+
| database() |
+------------+
| my_pets    |
+------------+
1 row in set (0.00 sec)
```

```
create table cats(
age int,
address varchar(100)
); #creating table with name cats
```

```
show tables; #shows available tables in the selected database.
```

```
+-------------------+
| Tables_in_my_pets |
+-------------------+
| cats              |
+-------------------+
1 row in set (0.01 sec)
```

```
show columns from cats # shows columns from the table cats.
OR

desc cats; #performs the same action as above describes the
table.
```

```
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| name  | varchar(100) | YES  |     | NULL    |       |
| age   | int          | YES  |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
2 rows in set (0.02 sec)
```

```
drop table cats; # deletes the table cats from the database.
```

Inserting data in the tables

```
insert into cats (name, age)
values ("jetson", 7); #this will insert data into already
existing table.
```

```
select * from cats;
```

```
+----------+------+
| name     | age  |
+----------+------+
| jetson   |    8 |
| victoria |    6 |
+----------+------+
2 rows in set (0.01 sec)
```

Multiple insert:

```
insert into cats (name, age) # don't forget to write table
values ("tim", 4), #name.
("john", 5),
("katy", 9),
("lens", 20);
```

```
show warnings; # shows you warnings.
```

```
insert into cats (name)
values ("cluadia");

# NULL is yes in the table. It means its ok to have unknown
value.
# NULL not means its 0.
```

```
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| name  | varchar(50) | YES  |     | NULL    |       |
| age   | int         | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

```
create table cats2 (
name varchar(50) not null,
age int not null ); # this will ensure that name and age
columns don't have null values. Default value is specifies if
nothing is provided.
```

```
create table cats2 (
name varchar(50) default "name not specified",
age int default 20 ); # here if any column entry is null/not
provided then default value is replaced.
```

```
create table cats2 (
name varchar(50) not null default "name is not specifies",
age int not null default 20 ); # here you can't write null
values and if no value provided then replaced by default value.
```

```
+-------+-------------+------+-----+--------------------+-------+
| Field | Type        | Null | Key | Default            | Extra |
+-------+-------------+------+-----+--------------------+-------+
| name  | varchar(50) | NO   |     | name not specified |       |
| age   | int         | NO   |     | 20                 |       |
+-------+-------------+------+-----+--------------------+-------+
2 rows in set (0.01 sec)
```

```
create table unique_cats (
cat_id int not null,
name varchar(50),
age int,
primary key (cat_id)
); # primary key is unique to each entry.
```

```
# auto_increment will increment id as more entries comes #in
automatically.

create table employees(
id int auto_increment not null,
first_name varchar(50),
last_name varchar(50),
middle_name varchar(50),
current_status varchar(50) not null default "employed",
primary key(id)
);

insert into employees(id, first_name, last_name,
current_status)
values(1, "dolly", "devil", "internship");
```

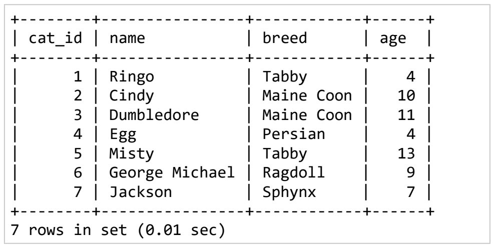
```
+----------------+-------------+------+-----+----------+----------------+
| Field          | Type        | Null | Key | Default  | Extra          |
+----------------+-------------+------+-----+----------+----------------+
| id             | int         | NO   | PRI | NULL     | auto_increment |
| first_name     | varchar(50) | YES  |     | NULL     |                |
| last_name      | varchar(50) | YES  |     | NULL     |                |
| middle_name    | varchar(50) | YES  |     | NULL     |                |
| current_status | varchar(50) | NO   |     | employed |                |
+----------------+-------------+------+-----+----------+----------------+
5 rows in set (0.00 sec)

mysql> select * from employees;
+----+------------+-----------+-------------+----------------+
| id | first_name | last_name | middle_name | current_status |
+----+------------+-----------+-------------+----------------+
|  1 | dolly      | devil     | NULL        | internship     |
+----+------------+-----------+-------------+----------------+
1 row in set (0.01 sec)
```

CRUD Commands(Create, Read, Update, Delete):

```
# inserting data in cats table
insert into cats(name, breed, age)
values('Ringo', 'Tabby', 4),
('Cindy', 'Maine Coon', 10),
('Dumbledore', 'Maine Coon', 11),
('Egg', 'Persian', 4),
('Misty', 'Tabby', 13),
('George Michael', 'Ragdoll', 9),
('Jackson', 'Sphynx', 7);
```

Select statement

```
select * from cats; # gives us all the rows in the cats table.
```



```
+--------+----------------+------------+------+
| cat_id | name           | breed      | age  |
+--------+----------------+------------+------+
|      1 | Ringo          | Tabby      |    4 |
|      2 | Cindy          | Maine Coon |   10 |
|      3 | Dumbledore     | Maine Coon |   11 |
|      4 | Egg            | Persian    |    4 |
|      5 | Misty          | Tabby      |   13 |
|      6 | George Michael | Ragdoll    |    9 |
|      7 | Jackson        | Sphynx     |    7 |
+--------+----------------+------------+------+
7 rows in set (0.01 sec)
```

```
select name from cats; #Accessing specific columns using
#select statement.
```

```
+----------------+
| name           |
+----------------+
| Ringo          |
| Cindy          |
| Dumbledore     |
| Egg            |
| Misty          |
| George Michael |
| Jackson        |
+----------------+
7 rows in set (0.00 sec)
```

select name, age from cats;#selecting multiple columns at once. Here **order matters** as in the next query.

```
+----------------+------+
| name           | age  |
+----------------+------+
| Ringo          |    4 |
| Cindy          |   10 |
| Dumbledore     |   11 |
| Egg            |    4 |
| Misty          |   13 |
| George Michael |    9 |
| Jackson        |    7 |
+----------------+------+
7 rows in set (0.00 sec)
```

select breed, age, name from cats;

```
+------------+------+----------------+
| breed      | age  | name           |
+------------+------+----------------+
| Tabby      |    4 | Ringo          |
| Maine Coon |   10 | Cindy          |
| Maine Coon |   11 | Dumbledore     |
| Persian    |    4 | Egg            |
| Tabby      |   13 | Misty          |
| Ragdoll    |    9 | George Michael |
| Sphynx     |    7 | Jackson        |
+------------+------+----------------+
7 rows in set (0.00 sec)
```

Where clause

select * from cats **where** age=4;

```
+--------+-------+---------+------+
| cat_id | name  | breed   | age  |
+--------+-------+---------+------+
|      1 | Ringo | Tabby   |    4 |
|      4 | Egg   | Persian |    4 |
+--------+-------+---------+------+
2 rows in set (0.00 sec)
```

select * from cats where name='Egg'; # you can also write "egg", capital letter #doesn't affect the query.

```
+--------+------+---------+------+
| cat_id | name | breed   | age  |
+--------+------+---------+------+
|      4 | Egg  | Persian |    4 |
+--------+------+---------+------+
1 row in set (0.00 sec)
```

Some practice queries-Select, Where:
select cat_id from cats;

```
+--------+
| cat_id |
+--------+
|      1 |
|      2 |
|      3 |
|      4 |
|      5 |
|      6 |
|      7 |
+--------+
7 rows in set (0.00 sec)
```

select name, breed from cats;

```
+----------------+------------+
| name           | breed      |
+----------------+------------+
| Ringo          | Tabby      |
| Cindy          | Maine Coon |
| Dumbledore     | Maine Coon |
| Egg            | Persian    |
| Misty          | Tabby      |
| George Michael | Ragdoll    |
| Jackson        | Sphynx     |
+----------------+------------+
7 rows in set (0.00 sec)
```

select name, age from cats where breed='Tabby';

```
+-------+------+
| name  | age  |
+-------+------+
| Ringo |    4 |
| Misty |   13 |
+-------+------+
2 rows in set (0.00 sec)
```

select cat_id, age from cats where cat_id=age;

```
+--------+------+
| cat_id | age  |
+--------+------+
|      4 |    4 |
|      7 |    7 |
+--------+------+
2 rows in set (0.00 sec)
```

Aliases:
select cat_id **as** id, name **as** cats_names from cats; #aliases only changes name of the column for showing original column name are not changed.

```
+----+----------------+
| id | cats_names     |
+----+----------------+
|  1 | Ringo          |
|  2 | Cindy          |
|  3 | Dumbledore     |
|  4 | Egg            |
|  5 | Misty          |
|  6 | George Michael |
|  7 | Jackson        |
+----+----------------+
7 rows in set (0.00 sec)
```

Update statement:
Keep in mind! do check before updating that you are updating the right entries, same goes for delete statement.

update cats **set** breed='Shorthair' where breed='Tabby';#changing breed from 'tabby' to 'shorthair'.

```
+--------+----------------+------------+------+
| cat_id | name           | breed      | age  |
+--------+----------------+------------+------+
|      1 | Ringo          | Shorthair  |    4 |
|      2 | Cindy          | Maine Coon |   10 |
|      3 | Dumbledore     | Maine Coon |   11 |
|      4 | Egg            | Persian    |    4 |
|      5 | Misty          | Shorthair  |   13 |
|      6 | George Michael | Ragdoll    |    9 |
|      7 | Jackson        | Sphynx     |    7 |
+--------+----------------+------------+------+
7 rows in set (0.01 sec)
```

```
update cats set age=14 where name='Misty'; # change age #from 13 to 14.
```

```
+--------+----------------+------------+------+
| cat_id | name           | breed      | age  |
+--------+----------------+------------+------+
|      1 | Ringo          | Shorthair  |    4 |
|      2 | Cindy          | Maine Coon |   10 |
|      3 | Dumbledore     | Maine Coon |   11 |
|      4 | Egg            | Persian    |    4 |
|      5 | Misty          | Shorthair  |   14 |
|      6 | George Michael | Ragdoll    |    9 |
|      7 | Jackson        | Sphynx     |    7 |
+--------+----------------+------------+------+
7 rows in set (0.00 sec)
```

Some practice queries-Update:

```
update cats set name='Jack' where name='jackson'; # update #'jackson' to 'jack'
```

```
+--------+----------------+------------+------+
| cat_id | name           | breed      | age  |
+--------+----------------+------------+------+
|      1 | Ringo          | Shorthair  |    4 |
|      2 | Cindy          | Maine Coon |   10 |
|      3 | Dumbledore     | Maine Coon |   11 |
|      4 | Egg            | Persian    |    4 |
|      5 | Misty          | Shorthair  |   14 |
|      6 | George Michael | Ragdoll    |    9 |
|      7 | Jack           | Sphynx     |    7 |
+--------+----------------+------------+------+
7 rows in set (0.00 sec)
```

```
update cats set breed='British Shorthair' where name='Ringo'; # update 'Ringo' # breed to 'British #Shorthair'.
```

```
+--------+----------------+-------------------+------+
| cat_id | name           | breed             | age  |
+--------+----------------+-------------------+------+
|      1 | Ringo          | British Shorthair |    4 |
|      2 | Cindy          | Maine Coon        |   10 |
|      3 | Dumbledore     | Maine Coon        |   11 |
|      4 | Egg            | Persian           |    4 |
|      5 | Misty          | Shorthair         |   14 |
|      6 | George Michael | Ragdoll           |    9 |
|      7 | Jack           | Sphynx            |    7 |
+--------+----------------+-------------------+------+
7 rows in set (0.00 sec)
```

```
update cats set age=12 where breed='Maine Coon'; # update #'Maine Coon' age to 12.
```

```
+--------+----------------+-------------------+------+
| cat_id | name           | breed             | age  |
+--------+----------------+-------------------+------+
|      1 | Ringo          | British Shorthair |    4 |
|      2 | Cindy          | Maine Coon        |   12 |
|      3 | Dumbledore     | Maine Coon        |   12 |
|      4 | Egg            | Persian           |    4 |
|      5 | Misty          | Shorthair         |   14 |
|      6 | George Michael | Ragdoll           |    9 |
|      7 | Jack           | Sphynx            |    7 |
+--------+----------------+-------------------+------+
7 rows in set (0.00 sec)
```

Delete statement:
Before deleting something it is a good practice that you check what are going to delete by using select statement.

```
delete from cats where name='egg'; # note that the cat_id #4 no longer existing. Deleting row.
```

```
+--------+----------------+-------------------+------+
| cat_id | name           | breed             | age  |
+--------+----------------+-------------------+------+
|      1 | Ringo          | British Shorthair |    4 |
|      2 | Cindy          | Maine Coon        |   12 |
|      3 | Dumbledore     | Maine Coon        |   12 |
|      5 | Misty          | Shorthair         |   14 |
|      6 | George Michael | Ragdoll           |    9 |
|      7 | Jack           | Sphynx            |    7 |
+--------+----------------+-------------------+------+
6 rows in set (0.01 sec)
```

```
delete cats; #this will delete all the data inside the #table but the table structure still exist you can put #data inside it.
# drop table will entirely remove your table.
```

```
drop database <name of database>; #delete database
```

Some practice queries-Delete:

```
delete from cats where age=4;
```

```
+--------+----------------+------------+------+
| cat_id | name           | breed      | age  |
+--------+----------------+------------+------+
|      2 | Cindy          | Maine Coon |   12 |
|      3 | Dumbledore     | Maine Coon |   12 |
|      5 | Misty          | Shorthair  |   14 |
|      6 | George Michael | Ragdoll    |    9 |
|      7 | Jack           | Sphynx     |    7 |
+--------+----------------+------------+------+
5 rows in set (0.00 sec
```

```
delete from cats where age=cat_id; # deletes data where #age and cat_id are same.
```

```
+--------+----------------+------------+------+
| cat_id | name           | breed      | age  |
+--------+----------------+------------+------+
|      2 | Cindy          | Maine Coon |   12 |
|      3 | Dumbledore     | Maine Coon |   12 |
|      5 | Misty          | Shorthair  |   14 |
|      6 | George Michael | Ragdoll    |    9 |
+--------+----------------+------------+------+
4 rows in set (0.00 sec)
```

```
drop table <table-name> # delete table
```

```
delete from cats; # deletes all data from the table. Table still exists.
```

```
Empty set (0.00 sec)
```

Concatenation

```
select
```

```
+----------------------+
| full_name            |
+----------------------+
| Jhumpa Lahiri        |
concat(author_fname, ' ', author_lname)
as full_name from books; #combines two columns/strings.
```

```
| Neil Gaiman          |
| Neil Gaiman          |
| Jhumpa Lahiri        |
| Dave Eggers          |
| Dave Eggers          |
| Michael Chabon       |
| Patti Smith          |
| Dave Eggers          |
| Neil Gaiman          |
| Raymond Carver       |
| Raymond Carver       |
| Don DeLillo          |
| John Steinbeck       |
| David Foster Wallace |
| David Foster Wallace |
+----------------------+
16 rows in set (0.00 sec)
```

```
select author_fname as first, author_lname as last,
concat (author_fname,' ', author_lname) as full
from books; # concatenating two columns and aliasing names as
#full in the 3rd column.
```

```
+---------+----------------+----------------------+
| first   | last           | full                 |
+---------+----------------+----------------------+
| Jhumpa  | Lahiri         | Jhumpa Lahiri        |
| Neil    | Gaiman         | Neil Gaiman          |
| Neil    | Gaiman         | Neil Gaiman          |
| Dave    | Eggers         | Dave Eggers          |
| Neil    | Gaiman         | Neil Gaiman          |
| Raymond | Carver         | Raymond Carver       |
| Raymond | Carver         | Raymond Carver       |
| Don     | DeLillo        | Don DeLillo          |
| John    | Steinbeck      | John Steinbeck       |
| David   | Foster Wallace | David Foster Wallace |
| David   | Foster Wallace | David Foster Wallace |
+---------+----------------+----------------------+
16 rows in set (0.00 sec)
```

Concat with separator

```
select
concat_ws ('-', title, author_fname, author_lname) as full
from books; # it will put the '-' after each concatenation #so
you don't need to put seperator each time.
```

```
+--------------------------------------------------------------------+
| full                                                               |
+--------------------------------------------------------------------+
| The Namesake-Jhumpa-Lahiri                                         |
| Norse Mythology-Neil-Gaiman                                        |
| American Gods-Neil-Gaiman                                          |
| Interpreter of Maladies-Jhumpa-Lahiri                              |
| A Hologram for the King: A Novel-Dave-Eggers                       |
| The Circle-Dave-Eggers                                             |
| The Amazing Adventures of Kavalier & Clay-Michael-Chabon           |
| Just Kids-Patti-Smith                                              |
| A Heartbreaking Work of Staggering Genius-Dave-Eggers              |
| Coraline-Neil-Gaiman                                               |
| What We Talk About When We Talk About Love: Stories-Raymond-Carver |
| Where I'm Calling From: Selected Stories-Raymond-Carver            |
| White Noise-Don-DeLillo                                            |
| Cannery Row-John-Steinbeck                                         |
| Oblivion: Stories-David-Foster Wallace                             |
| Consider the Lobster-David-Foster Wallace                          |
+--------------------------------------------------------------------+
16 rows in set (0.00 sec)
```

Substring

```
select substring('Hello World', 1, 4);  # unlike python index
start from 1 not 0.
```

```
+--------------------------------+
| substring('Hello World', 1, 4) |
+--------------------------------+
| Hell                           |
+--------------------------------+
1 row in set (0.00 sec)
```

```
select substring('Hello World',  -3);
```

```
+-------------------------------+
| substring('Hello World',  -3) |
+-------------------------------+
| rld                           |
+-------------------------------+
1 row in set (0.01 sec)
```

```
select substring('Hello World',  4);   # if we pass one value
it start from that index to end.

OR

Select substring ('Hello World', 4, 8); # these both are same
produces same results.

OR

substr('Hello World', 4, 8) # substr() function.
```

```
+------------------------------+
| substring('Hello World',  4) |
+------------------------------+
| lo World                     |
+------------------------------+
1 row in set (0.00 sec)
```

```
select substring(title, 1, 5) from books;
```

```
+------------------------+
| substring(title, 1, 5) |
+------------------------+
| The N                  |
| Norse                  |
| Ameri                  |
| Inter                  |
| A Hol                  |
| The C                  |
| The A                  |
| Just                   |
| A Hea                  |
| Coral                  |
| What                   |
| Where                  |
| White                  |
| Canne                  |
| Obliv                  |
| Consi                  |
+------------------------+
```

```
select substring(title, 1, 5) short_title from books;
```

```
+-------------+
| short_title |
+-------------+
| The N       |
| Norse       |
| Ameri       |
| Inter       |
| A Hol       |
| The C       |
| The A       |
| Just        |
| A Hea       |
| Coral       |
| What        |
| Where       |
| White       |
| Canne       |
| Obliv       |
| Consi       |
+-------------+
16 rows in set (0.00 sec)
```

```
select concat(substr(title, 1, 5), '...') as short_title
from books;
```

```
+-------------+
| short_title |
+-------------+
| The N...    |
| Norse...    |
| Ameri...    |
| Inter...    |
| A Hol...    |
| The C...    |
| The A...    |
| Just ...    |
| A Hea...    |
| Coral...    |
| What ...    |
| Where...    |
| White...    |
| Canne...    |
| Obliv...    |
| Consi...    |
+-------------+
16 rows in set (0.00 sec)
```

Replace

```
select replace('Hello World', 'l', '0');
```

```
+----------------------------------+
| replace('Hello World', 'l', '0') |
+----------------------------------+
| He00o Wor0d                      |
+----------------------------------+
1 row in set (0.00 sec)
```

```
 select replace('HellO World', 'o', '*'); # replace is case-sensitive only lower case 'o' got replaced.
```

```
+----------------------------------+
| replace('HellO World', 'o', '*') |
+----------------------------------+
| HellO W*rld                      |
+----------------------------------+
1 row in set (0.00 sec)
```

```
select replace('cheese bread coffe milk', ' ', ' and ');
```

```
+-----------------------------------------------+
| replace('cheese bread coffe milk', ' ', ' and ') |
+-----------------------------------------------+
| cheese and bread and coffe and milk           |
+-----------------------------------------------+
1 row in set (0.00 sec)
```

Reverse clause

```
select reverse('Hello World');
```

```
+------------------------+
| reverse('Hello World') |
+------------------------+
| dlroW olleH            |
+------------------------+
1 row in set (0.01 sec)
```

Character length

```
select char_length('Hello World');
```

```
+----------------------------+
| char_length('Hello World') |
+----------------------------+
|                         11 |
+----------------------------+
1 row in set (0.00 sec)
```

```
select author_fname, char_length(author_fname) as length from
books;
```

```
+--------------+--------+
| author_fname | length |
+--------------+--------+
| Jhumpa       |      6 |
| Neil         |      4 |
| Neil         |      4 |
| Jhumpa       |      6 |
| Dave         |      4 |
| Dave         |      4 |
| Michael      |      7 |
| Patti        |      5 |
| Dave         |      4 |
| Neil         |      4 |
| Raymond      |      7 |
| Raymond      |      7 |
| Don          |      3 |
| John         |      4 |
| David        |      5 |
| David        |      5 |
+--------------+--------+
16 rows in set (0.00 sec)
```

```
select upper('hellow world');
```

```
+----------------------+
| upper('hellow world') |
+----------------------+
| HELLOW WORLD          |
+----------------------+
1 row in set (0.00 sec)
```

```
select lower('hellow world');
```

```
+----------------------+
| lower('hellow world') |
+----------------------+
| hellow world          |
+----------------------+
1 row in set (0.00 sec)
```

```
select concat('my favorite book is ', upper(title)) as titles
from books;
```

```
+--------------------------------------------------------------------------+
| titles                                                                   |
+--------------------------------------------------------------------------+
| my favorite book is THE NAMESAKE                                         |
| my favorite book is NORSE MYTHOLOGY                                      |
| my favorite book is AMERICAN GODS                                        |
| my favorite book is INTERPRETER OF MALADIES                              |
| my favorite book is A HOLOGRAM FOR THE KING: A NOVEL                     |
| my favorite book is THE CIRCLE                                           |
| my favorite book is THE AMAZING ADVENTURES OF KAVALIER & CLAY            |
| my favorite book is JUST KIDS                                            |
| my favorite book is A HEARTBREAKING WORK OF STAGGERING GENIUS            |
| my favorite book is CORALINE                                             |
| my favorite book is WHAT WE TALK ABOUT WHEN WE TALK ABOUT LOVE: STORIES |
| my favorite book is WHERE I'M CALLING FROM: SELECTED STORIES             |
| my favorite book is WHITE NOISE                                          |
| my favorite book is CANNERY ROW                                          |
| my favorite book is OBLIVION: STORIES                                    |
| my favorite book is CONSIDER THE LOBSTER                                 |
+--------------------------------------------------------------------------+
16 rows in set (0.00 sec)
```

Some practice String Functions:

```
select reverse(
upper('hello there im playing with sql ')
) as reverse_upper; # reverse the string alongwith #capitalize
it and aliase column name.
```

```
+----------------------------------+
| reverse_upper                    |
+----------------------------------+
|  LQS HTIW GNIYALP MI EREHT OLLEH |
+----------------------------------+
1 row in set (0.00 sec)
```

```
select replace(concat('I', ' ', 'like', ' ', 'cats'), ' ',
'_'); # concatenate the strings and then replace spaces in
#that string with '_'.
```

```
+--------------------------------------------------------+
| replace(concat('I', ' ', 'like', ' ', 'cats'), ' ', '_') |
+--------------------------------------------------------+
| I_like_cats                                              |
+--------------------------------------------------------+
1 row in set (0.00 sec)
```

```
select replace(title, ' ', '->') as title
from books;
```

```
+--------------------------------------------------------------+
| title                                                        |
+--------------------------------------------------------------+
| The->Namesake                                                |
| Norse->Mythology                                             |
| American->Gods                                               |
| Interpreter->of->Maladies                                    |
| A->Hologram->for->the->King:->A->Novel                       |
| The->Circle                                                  |
| The->Amazing->Adventures->of->Kavalier->&->Clay              |
| Just->Kids                                                   |
| A->Heartbreaking->Work->of->Staggering->Genius               |
| Coraline                                                     |
| What->We->Talk->About->When->We->Talk->About->Love:->Stories |
| Where->I'm->Calling->From:->Selected->Stories                |
| White->Noise                                                 |
| Cannery->Row                                                 |
| Oblivion:->Stories                                           |
| Consider->the->Lobster                                       |
+--------------------------------------------------------------+
16 rows in set (0.00 sec)
```

```
select author_fname as forwards, reverse(author_fname) as
backwards from books;
```

```
+----------+-----------+
| forwards | backwards |
+----------+-----------+
| Jhumpa   | apmuhJ    |
| Neil     | lieN      |
| Neil     | lieN      |
| Jhumpa   | apmuhJ    |
| Dave     | evaD      |
| Dave     | evaD      |
| Michael  | leahciM   |
| Patti    | ittaP     |
| Dave     | evaD      |
| Neil     | lieN      |
| Raymond  | dnomyaR   |
| Raymond  | dnomyaR   |
| Don      | noD       |
| John     | nhoJ      |
| David    | divaD     |
| David    | divaD     |
+----------+-----------+
16 rows in set (0.00 sec)
```

```
select upper(concat(author_fname, ' ', author_lname)) as 'full
```

```
+----------------------+
name in caps'
from books;
```

```
| full name in caps    |
+----------------------+
| JHUMPA LAHIRI        |
| NEIL GAIMAN          |
| NEIL GAIMAN          |
| JHUMPA LAHIRI        |
| DAVE EGGERS          |
| DAVE EGGERS          |
| MICHAEL CHABON       |
| PATTI SMITH          |
| DAVE EGGERS          |
| NEIL GAIMAN          |
| RAYMOND CARVER       |
| RAYMOND CARVER       |
| DON DELILLO          |
| JOHN STEINBECK       |
| DAVID FOSTER WALLACE |
| DAVID FOSTER WALLACE |
+----------------------+
16 rows in set (0.00 sec)
```

```
select title,
char_length(title) as 'character count'
from books;
```

```
+-----------------------------------------------------+-----------------+
| title                                               | character count |
+-----------------------------------------------------+-----------------+
| The Namesake                                        |              12 |
| Norse Mythology                                     |              15 |
| American Gods                                       |              13 |
| Interpreter of Maladies                             |              23 |
| A Hologram for the King: A Novel                    |              32 |
| The Circle                                          |              10 |
| The Amazing Adventures of Kavalier & Clay           |              41 |
| Just Kids                                           |               9 |
| A Heartbreaking Work of Staggering Genius           |              41 |
| Coraline                                            |               8 |
| What We Talk About When We Talk About Love: Stories |              51 |
| Where I'm Calling From: Selected Stories            |              40 |
| White Noise                                         |              11 |
| Cannery Row                                         |              11 |
| Oblivion: Stories                                   |              17 |
| Consider the Lobster                                |              20 |
+-----------------------------------------------------+-----------------+
16 rows in set (0.00 sec)
```

```
select concat(substring(title, 1, 10), '...') as "short title",
concat(author_lname, ',', author_fname) as author,
concat(stock_quantity, ' in stock') as quantity
from books;
```

```
+---------------+----------------------+--------------+
| short title   | author               | quantity     |
+---------------+----------------------+--------------+
| The Namesa... | Lahiri,Jhumpa        | 32 in stock  |
| Norse Myth... | Gaiman,Neil          | 43 in stock  |
| American G... | Gaiman,Neil          | 12 in stock  |
| Interprete... | Lahiri,Jhumpa        | 97 in stock  |
| A Hologram... | Eggers,Dave          | 154 in stock |
| The Circle... | Eggers,Dave          | 26 in stock  |
| The Amazin... | Chabon,Michael       | 68 in stock  |
| Just Kids...  | Smith,Patti          | 55 in stock  |
| A Heartbre... | Eggers,Dave          | 104 in stock |
| Coraline...   | Gaiman,Neil          | 100 in stock |
| What We Ta... | Carver,Raymond       | 23 in stock  |
| Where I'm ... | Carver,Raymond       | 12 in stock  |
| White Nois... | DeLillo,Don          | 49 in stock  |
| Cannery Ro... | Steinbeck,John       | 95 in stock  |
| Oblivion: ... | Foster Wallace,David | 172 in stock |
| Consider t... | Foster Wallace,David | 92 in stock  |
+---------------+----------------------+--------------+
16 rows in set (0.00 sec)
```

Distinct

```
select distinct author_fname from books; # selects unique
values from the #author_fname column.
```

```
+--------------+
| author_fname |
+--------------+
| Jhumpa       |
| Neil         |
| Dave         |
| Michael      |
| Patti        |
| Raymond      |
| Don          |
| John         |
| David        |
| Dan          |
| Freida       |
| George       |
+--------------+
12 rows in set (0.00 sec)
```

```
select distinct author_fname, author_lname from books;
#distinct is applied to #both author_fname, author_lname
#columns.
```

```
+--------------+----------------+
| author_fname | author_lname   |
+--------------+----------------+
| Jhumpa       | Lahiri         |
| Neil         | Gaiman         |
| Dave         | Eggers         |
| Michael      | Chabon         |
| Patti        | Smith          |
| Raymond      | Carver         |
| Don          | DeLillo        |
| John         | Steinbeck      |
| David        | Foster Wallace |
| Dan          | Harris         |
| Freida       | Harris         |
| George       | Saunders       |
+--------------+----------------+
12 rows in set (0.00 sec)
```

Order by

```
+----------------+
select author_lname from books order by author_lname; # by
default the order is in ascending order alphabetical.
```

```
| author_lname   |
+----------------+
| Carver         |
| Carver         |
| Chabon         |
| DeLillo        |
| Eggers         |
| Eggers         |
| Eggers         |
| Foster Wallace |
| Foster Wallace |
| Gaiman         |
| Gaiman         |
| Gaiman         |
| Harris         |
| Harris         |
| Lahiri         |
| Lahiri         |
| Saunders       |
| Smith          |
| Steinbeck      |
+----------------+
19 rows in set (0.01 sec)
```

```
select author_lname from books order by author_lname desc;#desc
will order by author_lname in descending order.
```

```
+----------------+
| author_lname   |
+----------------+
| Steinbeck      |
| Smith          |
| Saunders       |
| Lahiri         |
| Lahiri         |
| Harris         |
| Harris         |
| Gaiman         |
| Gaiman         |
| Gaiman         |
| Foster Wallace |
| Foster Wallace |
| Eggers         |
| Eggers         |
| Eggers         |
| DeLillo        |
| Chabon         |
| Carver         |
| Carver         |
+----------------+
19 rows in set (0.00 sec)
```

```
select title, released_year, pages from books order by pages;
```

```
+-----------------------------------------------------+---------------+-------+
| title                                               | released_year | pages |
+-----------------------------------------------------+---------------+-------+
| What We Talk About When We Talk About Love: Stories |          1981 |   176 |
| Cannery Row                                         |          1945 |   181 |
| Interpreter of Maladies                             |          1996 |   198 |
| Coraline                                            |          2003 |   208 |
| 10% Happier                                         |          2014 |   256 |
| The Namesake                                        |          2003 |   291 |
| Norse Mythology                                     |          2016 |   304 |
| Just Kids                                           |          2010 |   304 |
| White Noise                                         |          1985 |   320 |
| Oblivion: Stories                                   |          2004 |   329 |
| Consider the Lobster                                |          2005 |   343 |
| A Hologram for the King: A Novel                    |          2012 |   352 |
| Lincoln In The Bardo                                |          2017 |   367 |
| fake_book                                           |          2001 |   428 |
| A Heartbreaking Work of Staggering Genius           |          2001 |   437 |
| American Gods                                       |          2001 |   465 |
| The Circle                                          |          2013 |   504 |
| Where I'm Calling From: Selected Stories            |          1989 |   526 |
| The Amazing Adventures of Kavalier & Clay           |          2000 |   634 |
+-----------------------------------------------------+---------------+-------+
19 rows in set (0.00 sec)
```

```
select title, released_year, pages from books order by 2;
# here 2 means the 2nd position element which is #released_year
in this query.
```

```
+-----------------------------------------------------+---------------+-------+
| title                                               | released_year | pages |
+-----------------------------------------------------+---------------+-------+
| Cannery Row                                         |          1945 |   181 |
| What We Talk About When We Talk About Love: Stories |          1981 |   176 |
| White Noise                                         |          1985 |   320 |
| Where I'm Calling From: Selected Stories            |          1989 |   526 |
| Interpreter of Maladies                             |          1996 |   198 |
| The Amazing Adventures of Kavalier & Clay           |          2000 |   634 |
| American Gods                                       |          2001 |   465 |
| A Heartbreaking Work of Staggering Genius           |          2001 |   437 |
| fake_book                                           |          2001 |   428 |
| The Namesake                                        |          2003 |   291 |
| Coraline                                            |          2003 |   208 |
| Oblivion: Stories                                   |          2004 |   329 |
| Consider the Lobster                                |          2005 |   343 |
| Just Kids                                           |          2010 |   304 |
| A Hologram for the King: A Novel                    |          2012 |   352 |
| The Circle                                          |          2013 |   504 |
| 10% Happier                                         |          2014 |   256 |
| Norse Mythology                                     |          2016 |   304 |
| Lincoln In The Bardo                                |          2017 |   367 |
+-----------------------------------------------------+---------------+-------+
19 rows in set (0.00 sec)
```

```
select author_fname, author_lname from books order by
author_lname, author_fname;
```

```
+--------------+----------------+
| author_fname | author_lname   |
+--------------+----------------+
| Raymond      | Carver         |
| Raymond      | Carver         |
| Michael      | Chabon         |
| Don          | DeLillo        |
| Dave         | Eggers         |
| Dave         | Eggers         |
| Dave         | Eggers         |
| David        | Foster Wallace |
| David        | Foster Wallace |
| Neil         | Gaiman         |
| Neil         | Gaiman         |
| Neil         | Gaiman         |
| Dan          | Harris         |
| Freida       | Harris         |
| Jhumpa       | Lahiri         |
| Jhumpa       | Lahiri         |
| George       | Saunders       |
| Patti        | Smith          |
| John         | Steinbeck      |
+--------------+----------------+
19 rows in set (0.00 sec)
```

Limit

```
select title, released_year from books order by released_year
limit 3;
```

```
+-----------------------------------------------------+---------------+
| title                                               | released_year |
+-----------------------------------------------------+---------------+
| Cannery Row                                         |          1945 |
| What We Talk About When We Talk About Love: Stories |          1981 |
| White Noise                                         |          1985 |
+-----------------------------------------------------+---------------+
3 rows in set (0.00 sec)
```

```
select title,
released_year
from books
order by released_year desc
limit 0, 5;
#index start from 0 irrespective of start index of string
methods from 1.
```

```
+---------------------------------+---------------+
| title                           | released_year |
+---------------------------------+---------------+
| Lincoln In The Bardo            |          2017 |
| Norse Mythology                 |          2016 |
| 10% Happier                     |          2014 |
| The Circle                      |          2013 |
| A Hologram for the King: A Novel |          2012 |
+---------------------------------+---------------+
5 rows in set (0.00 sec)
```

```
select title,
released_year
from books
order by released_year desc
limit 1, 5; # start from index 1 and give next 5 values
starting from it.
```

```
+---------------------------------+---------------+
| title                           | released_year |
+---------------------------------+---------------+
| Norse Mythology                 |          2016 |
| 10% Happier                     |          2014 |
| The Circle                      |          2013 |
| A Hologram for the King: A Novel |          2012 |
| Just Kids                       |          2010 |
+---------------------------------+---------------+
5 rows in set (0.00 sec)
```

```
select title,
 released_year
from books
order by released_year desc
limit 5, 3251525;  #if want to get all the entries after some index
then pass a large number on the second parameter.
```

```
+-----------------------------------------------------+---------------+
| title                                               | released_year |
+-----------------------------------------------------+---------------+
| Just Kids                                           |          2010 |
| Consider the Lobster                                |          2005 |
| Oblivion: Stories                                   |          2004 |
| The Namesake                                        |          2003 |
| Coraline                                            |          2003 |
| American Gods                                       |          2001 |
| A Heartbreaking Work of Staggering Genius           |          2001 |
| fake_book                                           |          2001 |
| The Amazing Adventures of Kavalier & Clay           |          2000 |
| Interpreter of Maladies                             |          1996 |
| Where I'm Calling From: Selected Stories            |          1989 |
| White Noise                                         |          1985 |
| What We Talk About When We Talk About Love: Stories |          1981 |
| Cannery Row                                         |          1945 |
+-----------------------------------------------------+---------------+
14 rows in set (0.00 sec)
```

Like clause

```
select title,
    author_fname
from books
where author_fname like '%da%'; #gives author_fname with
#specific 'da' having in it. % is called wildcard meaning
#something before 'da' and something after 'da'.
```

```
+-------------------------------------------+--------------+
| title                                     | author_fname |
+-------------------------------------------+--------------+
| A Hologram for the King: A Novel          | Dave         |
| The Circle                                | Dave         |
| A Heartbreaking Work of Staggering Genius | Dave         |
| Oblivion: Stories                         | David        |
| Consider the Lobster                      | David        |
| 10% Happier                               | Dan          |
| fake_book                                 | Freida       |
+-------------------------------------------+--------------+
7 rows in set (0.00 sec)
```

```
select title,
author_fname
from books
where author_fname like 'da%';   #find words start with 'da' and
#something after it.
```

```
+-------------------------------------------+--------------+
| title                                     | author_fname |
+-------------------------------------------+--------------+
| A Hologram for the King: A Novel          | Dave         |
| The Circle                                | Dave         |
| A Heartbreaking Work of Staggering Genius | Dave         |
| Oblivion: Stories                         | David        |
| Consider the Lobster                      | David        |
| 10% Happier                               | Dan          |
+-------------------------------------------+--------------+
6 rows in set (0.00 sec)
```

```
select title,
    author_fname
from books
where title like '%the%';
```

```
+-------------------------------------------+--------------+
| title                                     | author_fname |
+-------------------------------------------+--------------+
| The Namesake                              | Jhumpa       |
| A Hologram for the King: A Novel          | Dave         |
| The Circle                                | Dave         |
| The Amazing Adventures of Kavalier & Clay | Michael      |
| Consider the Lobster                      | David        |
| Lincoln In The Bardo                      | George       |
+-------------------------------------------+--------------+
6 rows in set (0.00 sec)
```

```
select title,
stock_quantity
from books
where stock_quantity like '____';
#this format has 4 underscores and in like will search for 4
character long values.
```

```
+----------------------+----------------+
| title                | stock_quantity |
+----------------------+----------------+
| Lincoln In The Bardo |           1000 |
+----------------------+----------------+
1 row in set (0.00 sec)
```

```
select title
from books
 where title like '%\%%'; #If we have specific '%' character in the
book title name.
```

```
+-------------+
| title       |
+-------------+
| 10% Happier |
+-------------+
1 row in set (0.00 sec)
```

```
select title
 from books
where title like '%\_%'; #If we have specific '_' character in the
book title name.
```

```
+-----------+
| title     |
+-----------+
| fake_book |
+-----------+
1 row in set (0.00 sec)
```

Some practice refining selections problems:

```
select title from books where title like '%stories%';  #find title
#with 'stories' in it.
```

```
+-----------------------------------------------------+
| title                                               |
+-----------------------------------------------------+
| What We Talk About When We Talk About Love: Stories |
| Where I'm Calling From: Selected Stories            |
| Oblivion: Stories                                   |
+-----------------------------------------------------+
3 rows in set (0.00 sec)
```

```
select title,
    pages
    from books
    order by pages desc
    limit 1;
```

```
+-------------------------------------------+-------+
| title                                     | pages |
+-------------------------------------------+-------+
| The Amazing Adventures of Kavalier & Clay |   634 |
+-------------------------------------------+-------+
1 row in set (0.00 sec)
```

```
select concat(title, ' - ', released_year) as 'summary'
    from books
    order by released_year desc;
```

```
+------------------------------------------------------------+
| summary                                                    |
+------------------------------------------------------------+
| Lincoln In The Bardo - 2017                                |
| Norse Mythology - 2016                                     |
| 10% Happier - 2014                                         |
| The Circle - 2013                                          |
| A Hologram for the King: A Novel - 2012                    |
| Just Kids - 2010                                           |
| Consider the Lobster - 2005                                |
| Oblivion: Stories - 2004                                   |
| The Namesake - 2003                                        |
| Coraline - 2003                                            |
| American Gods - 2001                                       |
| A Heartbreaking Work of Staggering Genius - 2001           |
| fake_book - 2001                                           |
| The Amazing Adventures of Kavalier & Clay - 2000           |
| Interpreter of Maladies - 1996                             |
| Where I'm Calling From: Selected Stories - 1989            |
| White Noise - 1985                                         |
| What We Talk About When We Talk About Love: Stories - 1981 |
| Cannery Row - 1945                                         |
+------------------------------------------------------------+
19 rows in set (0.00 sec)
```

```
select title,
    author_lname
    from books
    where author_lname like '% %'; # author_lname having space.
```

```
+----------------------+----------------+
| title                | author_lname   |
+----------------------+----------------+
| Oblivion: Stories    | Foster Wallace |
| Consider the Lobster | Foster Wallace |
+----------------------+----------------+
2 rows in set (0.00 sec)
```

```
select title,
    released_year,
    stock_quantity
    from books
    order by stock_quantity
    limit 3;
```

```
+-----------------------------------------------------+---------------+----------------+
| title                                               | released_year | stock_quantity |
+-----------------------------------------------------+---------------+----------------+
| Where I'm Calling From: Selected Stories            |          1989 |             12 |
| American Gods                                       |          2001 |             12 |
| What We Talk About When We Talk About Love: Stories |          1981 |             23 |
+-----------------------------------------------------+---------------+----------------+
3 rows in set (0.00 sec)
```

```
select title,
     author_lname
    from books
    order by author_lname,
     title;
```

```
+-----------------------------------------------------+----------------+
| title                                               | author_lname   |
+-----------------------------------------------------+----------------+
| What We Talk About When We Talk About Love: Stories | Carver         |
| Where I'm Calling From: Selected Stories            | Carver         |
| The Amazing Adventures of Kavalier & Clay           | Chabon         |
| White Noise                                         | DeLillo        |
| A Heartbreaking Work of Staggering Genius           | Eggers         |
| A Hologram for the King: A Novel                    | Eggers         |
| The Circle                                          | Eggers         |
| Consider the Lobster                                | Foster Wallace |
| Oblivion: Stories                                   | Foster Wallace |
| American Gods                                       | Gaiman         |
| Coraline                                            | Gaiman         |
| Norse Mythology                                     | Gaiman         |
| 10% Happier                                         | Harris         |
| fake_book                                           | Harris         |
| Interpreter of Maladies                             | Lahiri         |
| The Namesake                                         | Lahiri         |
| Lincoln In The Bardo                                 | Saunders       |
| Just Kids                                            | Smith          |
| Cannery Row                                          | Steinbeck      |
+-----------------------------------------------------+----------------+
19 rows in set (0.00 sec)
```

```
select concat(
     'MY FAVORITE AUTHOR IS ',
     author_fname, ' ', author_lname, '!'
          ) as 'yell'
     from books;
```

```
+---------------------------------------------+
| yell                                        |
+---------------------------------------------+
| MY FAVORITE AUTHOR IS Jhumpa Lahiri!        |
| MY FAVORITE AUTHOR IS Neil Gaiman!          |
| MY FAVORITE AUTHOR IS Neil Gaiman!          |
| MY FAVORITE AUTHOR IS Jhumpa Lahiri!        |
| MY FAVORITE AUTHOR IS Dave Eggers!          |
| MY FAVORITE AUTHOR IS Dave Eggers!          |
| MY FAVORITE AUTHOR IS Michael Chabon!       |
| MY FAVORITE AUTHOR IS Patti Smith!          |
| MY FAVORITE AUTHOR IS Dave Eggers!          |
| MY FAVORITE AUTHOR IS Neil Gaiman!          |
| MY FAVORITE AUTHOR IS Raymond Carver!       |
| MY FAVORITE AUTHOR IS Raymond Carver!       |
| MY FAVORITE AUTHOR IS Don DeLillo!          |
| MY FAVORITE AUTHOR IS John Steinbeck!       |
| MY FAVORITE AUTHOR IS David Foster Wallace! |
| MY FAVORITE AUTHOR IS David Foster Wallace! |
| MY FAVORITE AUTHOR IS Dan Harris!           |
| MY FAVORITE AUTHOR IS Freida Harris!        |
| MY FAVORITE AUTHOR IS George Saunders!      |
+---------------------------------------------+
19 rows in set (0.00 sec)
```

```
select concat(
     'MY FAVORITE AUTHOR IS ',
     upper(author_fname), ' ', upper(author_lname), '!'
     ) as 'yell'
     from books order by author_lname; #get all sorted author full
#Names with the string 'MY FAVORITE AUTHOR IS'.
```

```
+---------------------------------------------+
| yell                                        |
+---------------------------------------------+
| MY FAVORITE AUTHOR IS RAYMOND CARVER!       |
| MY FAVORITE AUTHOR IS RAYMOND CARVER!       |
| MY FAVORITE AUTHOR IS MICHAEL CHABON!       |
| MY FAVORITE AUTHOR IS DON DELILLO!          |
| MY FAVORITE AUTHOR IS DAVE EGGERS!          |
| MY FAVORITE AUTHOR IS DAVE EGGERS!          |
| MY FAVORITE AUTHOR IS DAVE EGGERS!          |
| MY FAVORITE AUTHOR IS DAVID FOSTER WALLACE! |
| MY FAVORITE AUTHOR IS DAVID FOSTER WALLACE! |
| MY FAVORITE AUTHOR IS NEIL GAIMAN!          |
| MY FAVORITE AUTHOR IS NEIL GAIMAN!          |
| MY FAVORITE AUTHOR IS NEIL GAIMAN!          |
| MY FAVORITE AUTHOR IS DAN HARRIS!           |
| MY FAVORITE AUTHOR IS FREIDA HARRIS!        |
| MY FAVORITE AUTHOR IS JHUMPA LAHIRI!        |
| MY FAVORITE AUTHOR IS JHUMPA LAHIRI!        |
| MY FAVORITE AUTHOR IS GEORGE SAUNDERS!      |
| MY FAVORITE AUTHOR IS PATTI SMITH!          |
| MY FAVORITE AUTHOR IS JOHN STEINBECK!       |
+---------------------------------------------+
19 rows in set (0.00 sec)
```

Aggregate functions:

```
select count(*) from books; #count tells the number of rows from the
#selected data
```

```
+----------+
| count(*) |
+----------+
|       19 |
+----------+
1 row in set (0.00 sec)
```

```
select count(author_fname) from books;
```

```
+---------------------+
| count(author_fname) |
+---------------------+
|                  19 |
+---------------------+
1 row in set (0.00 sec)
```

```
select count(distinct author_fname) from books;
#gives count of unique rows.
```

```
+-------------------------------+
| count(distinct(author_fname)) |
+-------------------------------+
|                            12 |
+-------------------------------+
1 row in set (0.01 sec)
```

```
select count(distinct author_fname, author_lname) from books;
# gives unique count of author first and last name combined.
```

```
+--------------------------------------------+
| count(distinct author_fname, author_lname) |
+--------------------------------------------+
|                                         12 |
+--------------------------------------------+
1 row in set (0.00 sec)
```

```
select count(*) from books where title like '%the%';
#Gives count of title having the in it.
```

```
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0.00 sec)
```

Group by clause

Group by applies with some aggregate function.
Aggregate/summarize identical data into single rows.

```
select author_lname, count(*) from books group by author_lname;
```

```
+----------------+----------+
| author_lname   | count(*) |
+----------------+----------+
| Lahiri         |        2 |
| Gaiman         |        3 |
| Eggers         |        3 |
| Chabon         |        1 |
| Smith          |        1 |
| Carver         |        2 |
| DeLillo        |        1 |
| Steinbeck      |        1 |
| Foster Wallace |        2 |
| Harris         |        2 |
| Saunders       |        1 |
+----------------+----------+
11 rows in set (0.00 sec)
```

```
select author_fname, author_lname, count(*) from books group by
author_lname;
#here count refers to the grouped columns after the group by
operation.
```

```
+--------------+----------------+----------+
| author_fname | author_lname   | count(*) |
+--------------+----------------+----------+
| Jhumpa       | Lahiri         |        2 |
| Neil         | Gaiman         |        3 |
| Dave         | Eggers         |        3 |
| Michael      | Chabon         |        1 |
| Patti        | Smith          |        1 |
| Raymond      | Carver         |        2 |
| Don          | DeLillo        |        1 |
| John         | Steinbeck      |        1 |
| David        | Foster Wallace |        2 |
| Dan          | Harris         |        2 |
| George       | Saunders       |        1 |
+--------------+----------------+----------+
11 rows in set (0.00 sec)
```

```
select released_year, count(*) from books group by released_year;
```

```
+---------------+----------+
| released_year | count(*) |
+---------------+----------+
|          2003 |        2 |
|          2016 |        1 |
|          2001 |        3 |
|          1996 |        1 |
|          2012 |        1 |
|          2013 |        1 |
|          2000 |        1 |
|          2010 |        1 |
|          1981 |        1 |
|          1989 |        1 |
|          1985 |        1 |
|          1945 |        1 |
|          2004 |        1 |
|          2005 |        1 |
|          2014 |        1 |
|          2017 |        1 |
+---------------+----------+
16 rows in set (0.00 sec)
```

```
select concat('In ', released_year, ' ', count(*), ' Books
released.') as year from books group by released_year;
```

```
+---------------------------+
| year                      |
+---------------------------+
| In 2003 2 Books released. |
| In 2016 1 Books released. |
| In 2001 3 Books released. |
| In 1996 1 Books released. |
| In 2012 1 Books released. |
| In 2013 1 Books released. |
| In 2000 1 Books released. |
| In 2010 1 Books released. |
| In 1981 1 Books released. |
| In 1989 1 Books released. |
| In 1985 1 Books released. |
| In 1945 1 Books released. |
| In 2004 1 Books released. |
| In 2005 1 Books released. |
| In 2014 1 Books released. |
| In 2017 1 Books released. |
+---------------------------+
16 rows in set (0.00 sec)
```

Min and Max functions

```
select title, max(pages) from books;
```

```
+--------------+------------+
| title        | max(pages) |
+--------------+------------+
| The Namesake |        634 |
+--------------+------------+
1 row in set (0.00 sec)
```

```
select min(pages) from books;
```

```
+------------+
| min(pages) |
+------------+
|        176 |
+------------+
1 row in set (0.00 sec)
```

Sub queries

Sometime the when performing min, max or some other operations two queries are independent and the result of the resultant is not what we are expecting. In this case we use sub queries.

```
select title, max(pages) from books; # the problem is like this the
name should be The 'Amazing Adventures of Kavalier & Clay' because
they have the most pages but we are getting  'The Nameshake'.
```

```
+--------------+------------+
| title        | max(pages) |
+--------------+------------+
| The Namesake |        634 |
+--------------+------------+
1 row in set (0.00 sec)
```

```
select title,
    pages
from books
where pages =(
        select max(pages)
```

```
+-------------------------------------------+-------+
| title                                     | pages |
+-------------------------------------------+-------+
| The Amazing Adventures of Kavalier & Clay |   634 |
+-------------------------------------------+-------+
1 row in set (0.00 sec)
```

```
        from books
    );
```

```
select title,
    pages
from books
where pages =(
        select min(pages)
        from books
    );
```

OR
The above query takes too much time below is efficient.

```
select title,
    pages
from books
order by pages asc
limit 1;
```

```
+-----------------------------------------------------+-------+
| title                                               | pages |
+-----------------------------------------------------+-------+
| What We Talk About When We Talk About Love: Stories |   176 |
+-----------------------------------------------------+-------+
1 row in set (0.00 sec)
```

OR

```
+-----------------------------------------------------+-------+
| title                                               | pages |
+-----------------------------------------------------+-------+
| What We Talk About When We Talk About Love: Stories |   176 |
+-----------------------------------------------------+-------+
1 row in set (0.01 sec)
```

Min/Max with group by

Find the year each author published their first book.

```
select author_fname,
    author_lname,
    min(released_year)
from books
group by author_fname,
    author_lname;
```

```
+-------------+----------------+--------------------+
| Jhumpa      | Lahiri         |               1996 |
| Neil        | Gaiman         |               2001 |
| Dave        | Eggers         |               2001 |
| Michael     | Chabon         |               2000 |
| Patti       | Smith          |               2010 |
| Raymond     | Carver         |               1981 |
| Don         | DeLillo        |               1985 |
| John        | Steinbeck      |               1945 |
| David       | Foster Wallace |               2004 |
| Dan         | Harris         |               2014 |
| Freida      | Harris         |               2001 |
| George      | Saunders       |               2017 |
+-------------+----------------+--------------------+
12 rows in set (0.00 sec)
```

Find the longest page count for each author.

```
select author_fname,
    author_lname,
    max(pages)
from books
group by author_fname,
    author_lname;
```

OR sorted pages

```
select author_fname,
    author_lname,
    max(pages)
from books
group by author_fname,
    author_lname
    order by max(pages) desc;
```

```
+-------------+----------------+------------+
| Jhumpa      | Lahiri         |        291 |
| Neil        | Gaiman         |        465 |
| Dave        | Eggers         |        504 |
| Michael     | Chabon         |        634 |
| Patti       | Smith          |        304 |
| Raymond     | Carver         |        526 |
| Don         | DeLillo        |        320 |
| John        | Steinbeck      |        181 |
| David       | Foster Wallace |        343 |
| Dan         | Harris         |        256 |
| Freida      | Harris         |        428 |
| George      | Saunders       |        367 |
+-------------+----------------+------------+
12 rows in set (0.00 sec)
```

```
+--------------+----------------+------------+
| author_fname | author_lname   | max(pages) |
+--------------+----------------+------------+
| Michael      | Chabon         |        634 |
| Raymond      | Carver         |        526 |
| Dave         | Eggers         |        504 |
| Neil         | Gaiman         |        465 |
| Freida       | Harris         |        428 |
| George       | Saunders       |        367 |
| David        | Foster Wallace |        343 |
| Don          | DeLillo        |        320 |
| Patti        | Smith          |        304 |
| Jhumpa       | Lahiri         |        291 |
| Dan          | Harris         |        256 |
| John         | Steinbeck      |        181 |
+--------------+----------------+------------+
12 rows in set (0.00 sec)
```

```
select concat(author_fname,' '
    ,author_lname) as 'author',
    max(pages) as 'longest book'
from books
group by author_fname,
    author_lname;
```

```
+----------------------+--------------+
| Jhumpa Lahiri        |          291 |
| Neil Gaiman          |          465 |
| Dave Eggers          |          504 |
| Michael Chabon       |          634 |
| Patti Smith          |          304 |
| Raymond Carver       |          526 |
| Don DeLillo          |          320 |
| John Steinbeck       |          181 |
| David Foster Wallace |          343 |
| Dan Harris           |          256 |
| Freida Harris        |          428 |
| George Saunders      |          367 |
+----------------------+--------------+
12 rows in set (0.00 sec)
```

Sum function

```
select concat(author_fname, ' ', author_lname),
    sum(pages)
from books
group by author_fname,
    author_lname;
# each author total book page count.
```

```
+-----------------------------------------+------------+
| Jhumpa Lahiri                           |        489 |
| Neil Gaiman                             |        977 |
| Dave Eggers                             |       1293 |
| Michael Chabon                          |        634 |
| Patti Smith                             |        304 |
| Raymond Carver                          |        702 |
| Don DeLillo                             |        320 |
| John Steinbeck                          |        181 |
| David Foster Wallace                    |        672 |
| Dan Harris                              |        256 |
| Freida Harris                           |        428 |
| George Saunders                         |        367 |
+-----------------------------------------+------------+
12 rows in set (0.00 sec)
```

Average function
Calculate the average released_year across all books

```
select released_year, avg(stock_quantity)
from books
group by released_year;
```

```
+---------------+---------------------+
| released_year | avg(stock_quantity) |
+---------------+---------------------+
|          2003 |             66.0000 |
|          2016 |             43.0000 |
|          2001 |            134.3333 |
|          1996 |             97.0000 |
|          2012 |            154.0000 |
|          2013 |             26.0000 |
|          2000 |             68.0000 |
|          2010 |             55.0000 |
|          1981 |             23.0000 |
|          1989 |             12.0000 |
|          1985 |             49.0000 |
|          1945 |             95.0000 |
|          2004 |            172.0000 |
|          2005 |             92.0000 |
|          2014 |             29.0000 |
|          2017 |           1000.0000 |
+---------------+---------------------+
16 rows in set (0.00 sec)
```

```
select author_fname,
    author_lname,
    avg(pages) as 'Average pages'
from books
group by author_fname,
    author_lname;
```

```
+--------------+----------------+--------------+
| Jhumpa       | Lahiri         |     244.5000 |
| Neil         | Gaiman         |     325.6667 |
| Dave         | Eggers         |     431.0000 |
| Michael      | Chabon         |     634.0000 |
| Patti        | Smith          |     304.0000 |
| Raymond      | Carver         |     351.0000 |
| Don          | DeLillo        |     320.0000 |
| John         | Steinbeck      |     181.0000 |
| David        | Foster Wallace |     336.0000 |
| Dan          | Harris         |     256.0000 |
| Freida       | Harris         |     428.0000 |
| George       | Saunders       |     367.0000 |
+--------------+----------------+--------------+
12 rows in set (0.00 sec)
```

Some practice Aggregate functions problems:

```
select count(*) from books; #print number of books in database.
```

```
+----------+
| count(*) |
+----------+
|       19 |
+----------+
1 row in set (0.01 sec)
```

```
select released_year, count(*) from books group by
released_year;
```

```
+---------------+----------+
| released_year | count(*) |
+---------------+----------+
|          2003 |        2 |
|          2016 |        1 |
|          2001 |        3 |
|          1996 |        1 |
|          2012 |        1 |
|          2013 |        1 |
|          2000 |        1 |
|          2010 |        1 |
|          1981 |        1 |
|          1989 |        1 |
|          1985 |        1 |
|          1945 |        1 |
|          2004 |        1 |
|          2005 |        1 |
|          2014 |        1 |
|          2017 |        1 |
+---------------+----------+
16 rows in set (0.00 sec)
```

```
select sum(stock_quantity) from books;
```

```
+---------------------+
| sum(stock_quantity) |
+---------------------+
|                2450 |
+---------------------+
1 row in set (0.00 sec)
```

```
select concat(author_fname, ' ', author_lname) as 'authors',
    avg(released_year)
from books
group by author_fname,
    author_lname;
```

```
+----------------------+--------------------+
| authors              | avg(released_year) |
+----------------------+--------------------+
| Jhumpa Lahiri        |          1999.5000 |
| Neil Gaiman          |          2006.6667 |
| Dave Eggers          |          2008.6667 |
| Michael Chabon       |          2000.0000 |
| Patti Smith          |          2010.0000 |
| Raymond Carver       |          1985.0000 |
| Don DeLillo          |          1985.0000 |
| John Steinbeck       |          1945.0000 |
| David Foster Wallace |          2004.5000 |
| Dan Harris           |          2014.0000 |
| Freida Harris        |          2001.0000 |
| George Saunders      |          2017.0000 |
+----------------------+--------------------+
12 rows in set (0.00 sec)
```

```
select concat(author_fname, ' ', author_lname) as 'authors',
    max(pages)
from books
group by author_fname,
    author_lname;
```

```
+----------------------+------------+
| authors              | max(pages) |
+----------------------+------------+
| Jhumpa Lahiri        |        291 |
| Neil Gaiman          |        465 |
| Dave Eggers          |        504 |
| Michael Chabon       |        634 |
| Patti Smith          |        304 |
| Raymond Carver       |        526 |
| Don DeLillo          |        320 |
| John Steinbeck       |        181 |
| David Foster Wallace |        343 |
| Dan Harris           |        256 |
| Freida Harris        |        428 |
| George Saunders      |        367 |
+----------------------+------------+
12 rows in set (0.00 sec)
```

```
select concat(author_fname, ' ', author_lname) as 'authors',
    max(pages)
from books
group by author_fname,
    author_lname
order by max(pages) desc;
```

```
+----------------------+------------+
| authors              | max(pages) |
+----------------------+------------+
| Michael Chabon       |        634 |
| Raymond Carver       |        526 |
| Dave Eggers          |        504 |
| Neil Gaiman          |        465 |
| Freida Harris        |        428 |
| George Saunders      |        367 |
| David Foster Wallace |        343 |
| Don DeLillo          |        320 |
| Patti Smith          |        304 |
| Jhumpa Lahiri        |        291 |
| Dan Harris           |        256 |
| John Steinbeck       |        181 |
+----------------------+------------+
12 rows in set (0.00 sec)
```

```
select released_year as 'year',
    count(*) as '# books',
    avg(pages) as 'avg pages'
from books
group by released_year;
```

```
+------+---------+-----------+
| year | # books | avg pages |
+------+---------+-----------+
| 2003 |       2 |  249.5000 |
| 2016 |       1 |  304.0000 |
| 2001 |       3 |  443.3333 |
| 1996 |       1 |  198.0000 |
| 2012 |       1 |  352.0000 |
| 2013 |       1 |  504.0000 |
| 2000 |       1 |  634.0000 |
| 2010 |       1 |  304.0000 |
| 1981 |       1 |  176.0000 |
| 1989 |       1 |  526.0000 |
| 1985 |       1 |  320.0000 |
| 1945 |       1 |  181.0000 |
| 2004 |       1 |  329.0000 |
| 2005 |       1 |  343.0000 |
| 2014 |       1 |  256.0000 |
| 2017 |       1 |  367.0000 |
+------+---------+-----------+
16 rows in set (0.00 sec)
```

Data Types

- **Char(4)**
- **Varchar(variable bytes)**
- **Decimal**(total number of digits, digits after decimal) - fixed point. Calculations are exact.

- **Float(4 bytes)** - floating point calculations are approximate: Takes more storage.

- **Double(8 bytes):** For more storage but it's not accurate.

Dates and time

- **Date:** 'YYYY-MM-DD'
- **Time**: 'HH:MM:SS'
- **Datetime**: 'YYYY-MM-DD HH:MM:SS'
- **timestamp**

curdate()-gives current data.
curtime()-gives current time.
now()-gives current datetime.

```
select curdate();
```

```
+------------+
| curdate()  |
+------------+
| 2021-11-21 |
+------------+
1 row in set (0.00 sec)
```

```
select now();
```

```
+---------------------+
| now()               |
+---------------------+
| 2021-11-21 22:58:45 |
+---------------------+
1 row in set (0.00 sec)
```

```
select curtime();
```

```
+-----------+
| curtime() |
+-----------+
| 22:59:10  |
+-----------+
1 row in set (0.00 sec)
```

```
SELECT name, day(birthdate) FROM people;
```

```
+---------+----------------+
| name    | day(birthdate) |
+---------+----------------+
| Padma   |             11 |
| Larry   |             25 |
| Toaster |             21 |
+---------+----------------+
3 rows in set (0.00 sec)
```

Formatting Dates

```
SELECT name, birthdate, day(birthdate) FROM people; #gives the day of that date.
```

```
+---------+------------+----------------+
| name    | birthdate  | day(birthdate) |
+---------+------------+----------------+
| Padma   | 1983-11-11 |             11 |
| Larry   | 1943-12-25 |             25 |
| Toaster | 2021-11-21 |             21 |
+---------+------------+----------------+
3 rows in set (0.00 sec)
```

```
SELECT name, birthdate, dayname(birthdate) FROM people; # gives the dayname of the date.
```

```
+---------+------------+--------------------+
| name    | birthdate  | dayname(birthdate) |
+---------+------------+--------------------+
| Padma   | 1983-11-11 | Friday             |
| Larry   | 1943-12-25 | Saturday           |
| Toaster | 2021-11-21 | Sunday             |
+---------+------------+--------------------+
3 rows in set (0.01 sec)
```

```
SELECT name, birthdate, dayofweek(birthdate) FROM people;
```

```
+---------+------------+----------------------+
| name    | birthdate  | dayofweek(birthdate) |
+---------+------------+----------------------+
| Padma   | 1983-11-11 |                    6 |
| Larry   | 1943-12-25 |                    7 |
| Toaster | 2021-11-21 |                    1 |
+---------+------------+----------------------+
3 rows in set (0.00 sec)
```

```
SELECT name, birthdate, dayofyear(birthdate) FROM people;
```

```
+---------+------------+----------------------+
| name    | birthdate  | dayofyear(birthdate) |
+---------+------------+----------------------+
| Padma   | 1983-11-11 |                  315 |
| Larry   | 1943-12-25 |                  359 |
| Toaster | 2021-11-21 |                  325 |
+---------+------------+----------------------+
3 rows in set (0.01 sec)
```

```
SELECT name, birthtime, dayofyear(birthtime) FROM people;
```

```
+---------+-----------+----------------------+
| name    | birthtime | dayofyear(birthtime) |
+---------+-----------+----------------------+
| Padma   | 10:07:35  |                  325 |
| Larry   | 04:10:42  |                  325 |
| Toaster | 23:01:34  |                  325 |
+---------+-----------+----------------------+
3 rows in set (0.00 sec)
```

```
SELECT name, birthdt, dayofyear(birthdt) FROM people;
```

```
+---------+---------------------+--------------------+
| name    | birthdt             | dayofyear(birthdt) |
+---------+---------------------+--------------------+
| Padma   | 1983-11-11 10:07:35 |                315 |
| Larry   | 1943-12-25 04:10:42 |                359 |
| Toaster | 2021-11-21 23:01:34 |                325 |
+---------+---------------------+--------------------+
3 rows in set (0.00 sec)
```

```
SELECT name, birthdt, month(birthdt) FROM people;
```

```
+---------+---------------------+----------------+
| name    | birthdt             | month(birthdt) |
+---------+---------------------+----------------+
| Padma   | 1983-11-11 10:07:35 |             11 |
| Larry   | 1943-12-25 04:10:42 |             12 |
| Toaster | 2021-11-21 23:01:34 |             11 |
+---------+---------------------+----------------+
3 rows in set (0.01 sec)
```

```
SELECT name, birthdt, monthname(birthdt) FROM people;
```

```
+---------+---------------------+--------------------+
| name    | birthdt             | monthname(birthdt) |
+---------+---------------------+--------------------+
| Padma   | 1983-11-11 10:07:35 | November           |
| Larry   | 1943-12-25 04:10:42 | December           |
| Toaster | 2021-11-21 23:01:34 | November           |
+---------+---------------------+--------------------+
3 rows in set (0.01 sec)
```

```
SELECT name, birthdt, hour(birthdt) FROM people;
```

```
+---------+---------------------+---------------+
| name    | birthdt             | hour(birthdt) |
+---------+---------------------+---------------+
| Padma   | 1983-11-11 10:07:35 |            10 |
| Larry   | 1943-12-25 04:10:42 |             4 |
| Toaster | 2021-11-21 23:01:34 |            23 |
+---------+---------------------+---------------+
3 rows in set (0.00 sec)
```

```
SELECT name, birthdt, minute(birthdt) FROM people;
```

```
+---------+---------------------+-----------------+
| name    | birthdt             | minute(birthdt) |
+---------+---------------------+-----------------+
| Padma   | 1983-11-11 10:07:35 |               7 |
| Larry   | 1943-12-25 04:10:42 |              10 |
| Toaster | 2021-11-21 23:01:34 |               1 |
+---------+---------------------+-----------------+
3 rows in set (0.00 sec)
```

```
SELECT DATE_FORMAT(birthdt, 'Was born on a %W') FROM people;
#format dates specified in the function. See documentation
table for #more info.
```

```
+-----------------------------------------+
| DATE_FORMAT(birthdt, 'Was born on a %W') |
+-----------------------------------------+
| Was born on a Friday                    |
| Was born on a Saturday                  |
| Was born on a Sunday                    |
+-----------------------------------------+
3 rows in set (0.01 sec)
```

```
SELECT DATE_FORMAT(birthdt, '%m/%d/%Y') FROM people;
```

```
+----------------------------------+
| DATE_FORMAT(birthdt, '%m/%d/%Y') |
+----------------------------------+
| 11/11/1983                       |
| 12/25/1943                       |
| 11/21/2021                       |
+----------------------------------+
3 rows in set (0.00 sec)
```

```
SELECT DATE_FORMAT(birthdt, '%m/%d/%Y at %h:%i') FROM people;
```

```
+-------------------------------------------+
| DATE_FORMAT(birthdt, '%m/%d/%Y at %h:%i') |
+-------------------------------------------+
| 11/11/1983 at 10:07                       |
| 12/25/1943 at 04:10                       |
| 11/21/2021 at 11:01                       |
+-------------------------------------------+
3 rows in set (0.00 sec)
```

Date math
Adding days, months, minutes, etc to dates.

select name, birthdate, **datediff**(now(), birthdate) from people;

```
+---------+------------+---------------------------+
| name    | birthdate  | datediff(now(), birthdate) |
+---------+------------+---------------------------+
| Padma   | 1983-11-11 |                     13890 |
| Larry   | 1943-12-25 |                     28456 |
| Toaster | 2021-11-21 |                         0 |
+---------+------------+---------------------------+
3 rows in set (0.01 sec)
```

select birthdt, **date_add**(birthdt, interval 1 month) from people;

```
+---------------------+-------------------------------------+
| birthdt             | date_add(birthdt, interval 1 month) |
+---------------------+-------------------------------------+
| 1983-11-11 10:07:35 | 1983-12-11 10:07:35                 |
| 1943-12-25 04:10:42 | 1944-01-25 04:10:42                 |
| 2021-11-21 23:01:34 | 2021-12-21 23:01:34                 |
+---------------------+-------------------------------------+
3 rows in set (0.00 sec)
```

select birthdt, birthdt **+ interval 1 month** from people;

```
+---------------------+----------------------------+
| birthdt             | birthdt + interval 1 month |
+---------------------+----------------------------+
| 1983-11-11 10:07:35 | 1983-12-11 10:07:35        |
| 1943-12-25 04:10:42 | 1944-01-25 04:10:42        |
| 2021-11-21 23:01:34 | 2021-12-21 23:01:34        |
+---------------------+----------------------------+
3 rows in set (0.00 sec)
```

select birthdt, birthdt **+ interval 1 day** from people;

```
+---------------------+--------------------------+
| birthdt             | birthdt + interval 1 day |
+---------------------+--------------------------+
| 1983-11-11 10:07:35 | 1983-11-12 10:07:35      |
| 1943-12-25 04:10:42 | 1943-12-26 04:10:42      |
| 2021-11-21 23:01:34 | 2021-11-22 23:01:34      |
+---------------------+--------------------------+
3 rows in set (0.00 sec)
```

select birthdt, birthdt **- interval 1 day** from people;

```
+---------------------+--------------------------+
| birthdt             | birthdt - interval 1 day |
+---------------------+--------------------------+
| 1983-11-11 10:07:35 | 1983-11-10 10:07:35      |
| 1943-12-25 04:10:42 | 1943-12-24 04:10:42      |
| 2021-11-21 23:01:34 | 2021-11-20 23:01:34      |
+---------------------+--------------------------+
3 rows in set (0.00 sec)
```

select birthdt, birthdt **+ interval 1 month + interval 1 day** from people;

```
+---------------------+-------------------------------------------+
| birthdt             | birthdt + interval 1 month + interval 1 day |
+---------------------+-------------------------------------------+
| 1983-11-11 10:07:35 | 1983-12-12 10:07:35                       |
| 1943-12-25 04:10:42 | 1944-01-26 04:10:42                       |
| 2021-11-21 23:01:34 | 2021-12-22 23:01:34                       |
+---------------------+-------------------------------------------+
3 rows in set (0.00 sec)
```

Creating a table that stores timestamp for updated content

```
create table comments2 (
    content varchar(100),
    created_at timestamp default now() on update now()
);
# here on update will change created_at value to now every
time created_at is changed.
#To update date whenever the content field is updated we used
this logic.
```

```
insert into comments2 (content)
values('asdassrv');

insert into comments2 (content)
values('jrtsbvf');
```

```
select * from comments2;
```

```
+----------+---------------------+
| content  | created_at          |
+----------+---------------------+
| jrtsbvf  | 2021-11-22 00:33:13 |
| asdassrv | 2021-11-22 00:33:23 |
+----------+---------------------+
2 rows in set (0.00 sec)
```

```
update comments2
set content = 'This is totally great !!!'
where content = 'jrtsbvf';
```

```
update comments2
set content = 'Enjoy this sweet moment!'
where content = 'asdassrv';
```

```
select * from comments2;
```

```
+---------------------------+---------------------+
| content                   | created_at          |
+---------------------------+---------------------+
| This is totally great !!! | 2021-11-22 00:36:02 |
| Enjoy this sweet moment!  | 2021-11-22 00:37:19 |
| I like bannana pie!       | 2021-11-22 00:34:28 |
+---------------------------+---------------------+
3 rows in set (0.00 sec)
```

```
select * from comments2 order by created_at desc;
```

```
+---------------------------+---------------------+
| content                   | created_at          |
+---------------------------+---------------------+
| Enjoy this sweet moment!  | 2021-11-22 00:37:19 |
| This is totally great !!! | 2021-11-22 00:36:02 |
| I like bannana pie!       | 2021-11-22 00:34:28 |
+---------------------------+---------------------+
3 rows in set (0.00 sec)
```

Logical Operators

(and, or,!=,not like, greater than, less than, between, in and not in)

```
select title from books where released_year = 2017;
```

```
+----------------------+
| title                |
+----------------------+
| Lincoln In The Bardo |
+----------------------+
1 row in set (0.00 sec)
```

```
select title from books where released_year != 2017;
```

```
+-----------------------------------------------------+
| title                                               |
+-----------------------------------------------------+
| The Namesake                                        |
| Norse Mythology                                     |
| American Gods                                       |
| Interpreter of Maladies                             |
| A Hologram for the King: A Novel                    |
| The Circle                                          |
| The Amazing Adventures of Kavalier & Clay           |
| Just Kids                                           |
| A Heartbreaking Work of Staggering Genius           |
| Coraline                                            |
| What We Talk About When We Talk About Love: Stories |
| Where I'm Calling From: Selected Stories            |
| White Noise                                         |
| Cannery Row                                         |
| Oblivion: Stories                                   |
| Consider the Lobster                                |
| 10% Happier                                         |
| fake_book                                           |
+-----------------------------------------------------+
18 rows in set (0.00 sec)
```

```
select title from books where title like '%w%';
```

```
+-----------------------------------------------------+
| title                                               |
+-----------------------------------------------------+
| A Heartbreaking Work of Staggering Genius           |
| What We Talk About When We Talk About Love: Stories |
| Where I'm Calling From: Selected Stories            |
| White Noise                                         |
| Cannery Row                                         |
+-----------------------------------------------------+
5 rows in set (0.00 sec)
```

```
select title from books where title not like '%w%';  # not like
opposite of 'like' clause.
```

```
+-------------------------------------------+
| title                                     |
+-------------------------------------------+
| The Namesake                              |
| Norse Mythology                           |
| American Gods                             |
| Interpreter of Maladies                   |
| A Hologram for the King: A Novel          |
| The Circle                                |
| The Amazing Adventures of Kavalier & Clay |
| Just Kids                                 |
| Coraline                                  |
| Oblivion: Stories                         |
| Consider the Lobster                      |
| 10% Happier                               |
| fake_book                                 |
| Lincoln In The Bardo                      |
+-------------------------------------------+
14 rows in set (0.00 sec)
```

```
select title, released_year from books where released_year < 2000;
```

```
+-----------------------------------------------------+---------------+
| title                                               | released_year |
+-----------------------------------------------------+---------------+
| Interpreter of Maladies                             |          1996 |
| What We Talk About When We Talk About Love: Stories |          1981 |
| Where I'm Calling From: Selected Stories            |          1989 |
| White Noise                                         |          1985 |
| Cannery Row                                         |          1945 |
+-----------------------------------------------------+---------------+
5 rows in set (0.00 sec)
```

```
select title, released_year from books where released_year <= 2000;
```

```
+-----------------------------------------------------+---------------+
| title                                               | released_year |
+-----------------------------------------------------+---------------+
| Interpreter of Maladies                             |          1996 |
| The Amazing Adventures of Kavalier & Clay           |          2000 |
| What We Talk About When We Talk About Love: Stories |          1981 |
| Where I'm Calling From: Selected Stories            |          1989 |
| White Noise                                         |          1985 |
| Cannery Row                                         |          1945 |
+-----------------------------------------------------+---------------+
6 rows in set (0.00 sec)
```

Relationships
There always some relationship between tables in the real world.

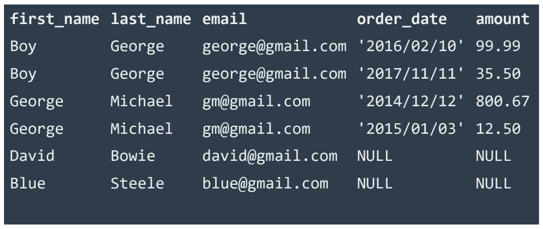
Types of relations:
1. One to one.
2. One to many.
3. Many to one.

One to many relationship(1:Many):

Types of key:
1. Primary key: Its always unique for each entry in a table.
2. Foreign key: References to other table within the current table.
3. Composite key:

```
CREATE TABLE customers(
    id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    email VARCHAR(100)
);
CREATE TABLE orders(
    id INT AUTO_INCREMENT PRIMARY KEY,
    order_date DATE,
    amount DECIMAL(8,2),
    customer_id INT,
    FOREIGN KEY(customer_id) REFERENCES customers(id)
);
```

After using relationship we use foreign key that points to the other table.customer_id is a foreign key that refers to customer table.



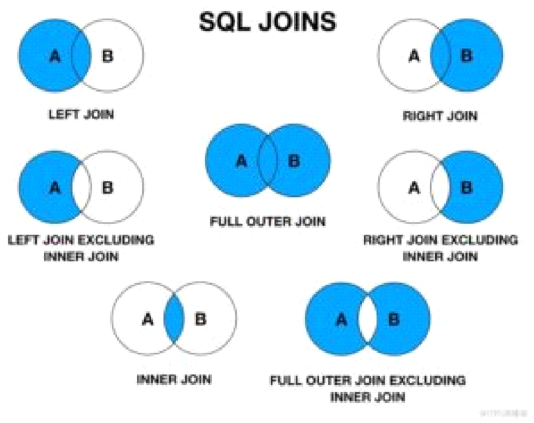
first_name	last_name	email	order_date	amount
Boy	George	george@gmail.com	'2016/02/10'	99.99
Boy	George	george@gmail.com	'2017/11/11'	35.50
George	Michael	gm@gmail.com	'2014/12/12'	800.67
George	Michael	gm@gmail.com	'2015/01/03'	12.50
David	Bowie	david@gmail.com	NULL	NULL
Blue	Steele	blue@gmail.com	NULL	NULL

customer_id	first_name	last_name	email
1	Boy	George	george@gmail.com
2	George	Michael	gm@gmail.com
3	David	Bowie	david@gmail.com
4	Blue	Steele	blue@gmail.com

order_id	order_date	amount	customer_id
1	'2016/02/10'	99.99	1
2	'2017/11/11'	35.50	1
3	'2014/12/12'	800.67	2
4	'2015/01/03'	12.50	2

SQL JOINS

1. Left join
2. Right join
3. Inner join.
4. full outer join.


SQL JOINS
A
B
LEFT JOIN
RIGHT JOIN
FULL OUTER JOIN
LEFT JOIN EXCLUDING INNER JOIN
RIGHT JOIN EXCLUDING INNER JOIN
INNER JOIN
FULL OUTER JOIN EXCLUDING INNER JOIN

Cross join

Select the orders given by the "boy george". This is a 2 step procesfirst fetch the id of the person from customers tables then use that id to get the orders from the orders tables.

```
select * from orders where customer_id =(
select id from customers where last_name='George');

select * from customers, orders; # this is also a join but not used very much.
```

```
+----+------------+--------+-------------+
| id | order_date | amount | customer_id |
+----+------------+--------+-------------+
|  1 | 2016-02-10 |  99.99 |           1 |
|  2 | 2017-11-11 |  35.50 |           1 |
+----+------------+--------+-------------+
2 rows in set (0.01 sec)
```

Inner join

Implicit inner join
```
select * from customers, orders
where customers.id = orders.customer_id;
```

```
+----+------------+-----------+------------------+----+------------+--------+------------
-+
| id | first_name | last_name | email            | id | order_date | amount | customer_id
|
+----+------------+-----------+------------------+----+------------+--------+------------
-+
|  1 | Boy        | George    | george@gmail.com |  1 | 2016-02-10 |  99.99 |           1
|
|  1 | Boy        | George    | george@gmail.com |  2 | 2017-11-11 |  35.50 |           1
|
|  2 | George     | Michael   | gm@gmail.com     |  3 | 2014-12-12 | 800.67 |           2
|
|  2 | George     | Michael   | gm@gmail.com     |  4 | 2015-01-03 |  12.50 |           2
|
|  5 | Bette      | Davis     | bette@aol.com    |  5 | 1999-04-11 | 450.25 |           5
|
+----+------------+-----------+------------------+----+------------+--------+------------
-+
```

For some columns instead of all of them.

```
select first_name, last_name, order_date, amount from
customers, orders
where customers.id = orders.customer_id;
```

```
+------------+-----------+------------+--------+
| first_name | last_name | order_date | amount |
+------------+-----------+------------+--------+
| Boy        | George    | 2016-02-10 |  99.99 |
| Boy        | George    | 2017-11-11 |  35.50 |
| George     | Michael   | 2014-12-12 | 800.67 |
| George     | Michael   | 2015-01-03 |  12.50 |
| Bette      | Davis     | 1999-04-11 | 450.25 |
+------------+-----------+------------+--------+
```

Explicit Inner join(Better option)

It gives the common entries from the two tables.

```
select
    first_name,
    last_name,
    order_date,
    amount
from
    customers
    inner join orders on customers.id = orders.customer_id;
```

```
+------------+-----------+------------+--------+
| first_name | last_name | order_date | amount |
+------------+-----------+------------+--------+
| Boy        | George    | 2016-02-10 |  99.99 |
| Boy        | George    | 2017-11-11 |  35.50 |
| George     | Michael   | 2014-12-12 | 800.67 |
| George     | Michael   | 2015-01-03 |  12.50 |
| Bette      | Davis     | 1999-04-11 | 450.25 |
+------------+-----------+------------+--------+
5 rows in set (0.00 sec)
```

Left join

Takes all the data from the lef table and the common entries of the right table only.

```
select
    first_name,
    last_name,
    order_date,
    sum(amount) as total_amount
from
    customers
    left join orders on customers.id = orders.customer_id
group by
    orders.customer_id
order by
    total_amount desc;
```

```
+------------+-----------+------------+--------------+
| first_name | last_name | order_date | total_amount |
+------------+-----------+------------+--------------+
| George     | Michael   | 2014-12-12 |       813.17 |
| Bette      | Davis     | 1999-04-11 |       450.25 |
| Boy        | George    | 2016-02-10 |       135.49 |
| David      | Bowie     | NULL       |         NULL |
+------------+-----------+------------+--------------+
4 rows in set (0.00 sec)
```

Without order by.

```
select
    first_name,
    last_name,
    sum(amount) as total_amount
from
    customers
    left join orders on customers.id = orders.customer_id
group by
    orders.customer_id;
```

```
+------------+-----------+--------------+
| first_name | last_name | total_amount |
+------------+-----------+--------------+
| Boy        | George    |       135.49 |
| George     | Michael   |       813.17 |
| David      | Bowie     |         NULL |
| Bette      | Davis     |       450.25 |
+------------+-----------+--------------+
4 rows in set (0.00 sec)
```

To set NULL values to 0 use **IFNULL()** method.

```
select
    first_name,
    last_name,
    ifnull(sum(amount), 0) as total_spent
from
    customers
    left join orders on customers.id = orders.customer_id
group by
    orders.customer_id;
```

```
+------------+-----------+-------------+
| first_name | last_name | total_spent |
+------------+-----------+-------------+
| Boy        | George    |      135.49 |
| George     | Michael   |      813.17 |
| David      | Bowie     |        0.00 |
| Bette      | Davis     |      450.25 |
+------------+-----------+-------------+
4 rows in set (0.00 sec)
```

Right join

```
select
    first_name,
    last_name,
    order_date,
    amount
from
    customers
    right join orders on customers.id = orders.customer_id;
```

```
+------------+-----------+------------+--------+
| first_name | last_name | order_date | amount |
+------------+-----------+------------+--------+
| Boy        | George    | 2016-02-10 |  99.99 |
| Boy        | George    | 2017-11-11 |  35.50 |
| George     | Michael   | 2014-12-12 | 800.67 |
| George     | Michael   | 2015-01-03 |  12.50 |
| Bette      | Davis     | 1999-04-11 | 450.25 |
+------------+-----------+------------+--------+
5 rows in set (0.00 sec)
```

Exercise questions:

Problem #1:On delete cascade

When you've a foreign key, if you delete some entry depending on it then the corresponding entry in the other table should also be deleted beacasue it can cause problems. So for this use this "**on delete cascade**" in your sql command.

```
CREATE TABLE orders(
    id INT AUTO_INCREMENT PRIMARY KEY,
    order_date DATE,
    amount DECIMAL(8, 2),
    customer_id INT,
    FOREIGN KEY(customer_id) REFERENCES customers(id)
    on delete cascade
);
```

For example:
If you delete "boy george" then the two orders having customer_id=1 should also be deleted because they depend on "boy george".

Inner join with order by:

```
select first_name,title,grade from students
inner join papers on students.id=papers.student_id
order by grade desc;
```

```
+------------+---------------------------------------+-------+
| first_name | title                                 | grade |
+------------+---------------------------------------+-------+
| Samantha   | De Montaigne and The Art of The Essay |    98 |
| Samantha   | Russian Lit Through The Ages          |    94 |
| Carlos     | Borges and Magical Realism            |    89 |
| Caleb      | My Second Book Report                 |    75 |
| Caleb      | My First Book Report                  |    60 |
+------------+---------------------------------------+-------+
5 rows in set (0.00 sec)
```

Problem# 2:

Select students who didn’t write some papers so this can be done using **left join.**

```
select first_name,title,grade from students
left join papers on students.id=papers.student_id;
```

```
+------------+---------------------------------------+-------+
| first_name | title                                 | grade |
+------------+---------------------------------------+-------+
| Caleb      | My First Book Report                  |    60 |
| Caleb      | My Second Book Report                 |    75 |
| Samantha   | Russian Lit Through The Ages          |    94 |
| Samantha   | De Montaigne and The Art of The Essay |    98 |
| Raj        | NULL                                  |  NULL |
| Carlos     | Borges and Magical Realism            |    89 |
| Lisa       | NULL                                  |  NULL |
+------------+---------------------------------------+-------+
7 rows in set (0.00 sec)
```

Problem #3:

Fill the **'NULL'** with some values.

```
select
    first_name,
    ifnull(title, "MISSING") as title,
    ifnull(grade, 0) as grade
from
    students
    left join papers on students.id = papers.student_id;
```

```
+------------+---------------------------------------+-------+
| first_name | title                                 | grade |
+------------+---------------------------------------+-------+
| Caleb      | My First Book Report                  |    60 |
| Caleb      | My Second Book Report                 |    75 |
| Samantha   | Russian Lit Through The Ages          |    94 |
| Samantha   | De Montaigne and The Art of The Essay |    98 |
| Raj        | MISSING                               |     0 |
| Carlos     | Borges and Magical Realism            |    89 |
| Lisa       | MISSING                               |     0 |
+------------+---------------------------------------+-------+
7 rows in set (0.00 sec)
```

Problem #4:

Select students who didn’t write a paper.

Take the average of the grade and fill **'NULL'** with **0**.

```
select
    first_name,
    ifnull(avg(grade), 0) as average
from
    students
    left join papers on students.id = papers.student_id
    group by students.id order by grade desc;
```

```
+------------+---------+
| first_name | average |
+------------+---------+
| Samantha   | 96.0000 |
| Carlos     | 89.0000 |
| Caleb      | 67.5000 |
| Raj        |  0.0000 |
| Lisa       |  0.0000 |
+------------+---------+
5 rows in set (0.00 sec)
```

Problem #5:

Use if statement that if grade is greater than **75%** then **'PASSING'** otherwise **'FAILING'** status.

```
select
    first_name,
    ifnull(avg(grade), 0) as average,
    case
        when avg(grade) >= 75 then "PASSING"
        else "FAILING"
    end as passing_status
from
    students
```

```
+------------+---------+----------------+
| first_name | average | passing_status |
+------------+---------+----------------+
| Samantha   | 96.0000 | PASSING        |
| Carlos     | 89.0000 | PASSING        |
| Caleb      | 67.5000 | FAILING        |
| Raj        |  0.0000 | FAILING        |
| Lisa       |  0.0000 | FAILING        |
+------------+---------+----------------+
5 rows in set (0.00 sec)
```

```
    left join papers on students.id = papers.student_id
group by
    students.id
order by
    average desc;
```

Many to Many joins:

Examples:
Books<->Authors
Blog post<->Tags
Students<->Classes

We've two tables **Reviewers, Series** and 3rd reviews table that have 2 foreign keys. But in the review table you can see that reviewer_id and series_id are int and we can't see what exactly the name of reviwer or series for that we use join as our convenience.

Table1:
Reviewers

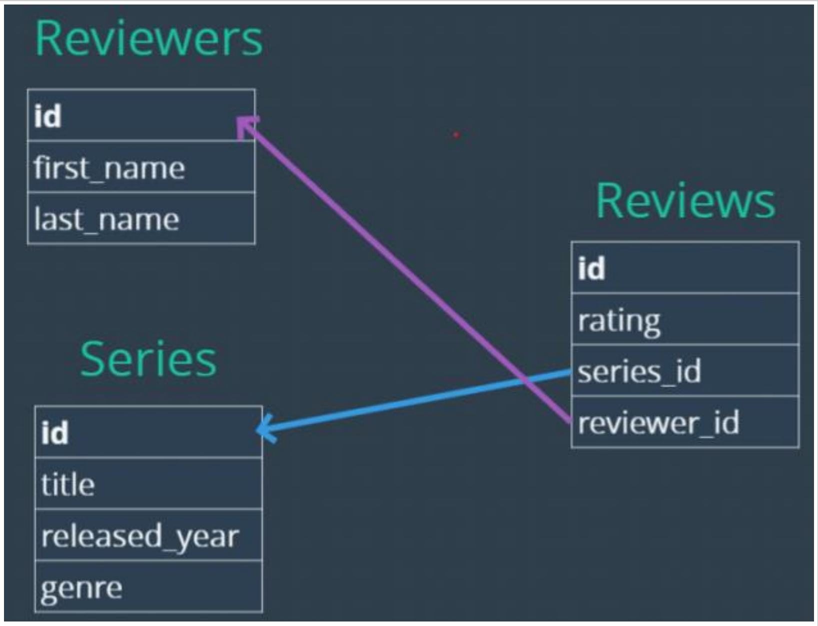
id	first_name	last_name
1	Blue	Steele
2	Wyatt	Earp

Table2:
Series

id	title	released_year	genre
1	Archer	2009	Animation
2	Fargo	2014	Drama

Table3:
Reviews

id	rating	reviewer_id	series_id
1	8.9	1	2
2	9.5	2	2



Creating tables

```
create table reviewers(
    id int auto_increment primary key,
    first_name varchar(100),
    last_name varchar(100)
);

create table series(
    id int auto_increment primary key,
    title varchar(100),
    released_year year(4),
    genre varchar(100)
);
```

Data Insertion:

```
INSERT INTO series (title, released_year, genre) VALUES
    ('Archer', 2009, 'Animation'),
    ('Arrested Development', 2003, 'Comedy'),
    ("Bob's Burgers", 2011, 'Animation'),
    ('Bojack Horseman', 2014, 'Animation'),
    ("Breaking Bad", 2008, 'Drama'),
    ('Curb Your Enthusiasm', 2000, 'Comedy'),
    ("Fargo", 2014, 'Drama'),
    ('Freaks and Geeks', 1999, 'Comedy'),
    ('General Hospital', 1963, 'Drama'),
    ('Halt and Catch Fire', 2014, 'Drama'),
    ('Malcolm In The Middle', 2000, 'Comedy'),
    ('Pushing Daisies', 2007, 'Comedy'),
    ('Seinfeld', 1989, 'Comedy'),
    ('Stranger Things', 2016, 'Drama');

INSERT INTO reviewers (first_name, last_name) VALUES
    ('Thomas', 'Stoneman'),
    ('Wyatt', 'Skaggs'),
    ('Kimbra', 'Masters'),
    ('Domingo', 'Cortes'),
    ('Colt', 'Steele'),
    ('Pinkie', 'Petit'),
    ('Marlon', 'Crafford');
```

```
+----+-----------------------+---------------+-----------+
| id | title                 | released_year | genre     |
+----+-----------------------+---------------+-----------+
|  1 | Archer                |          2009 | Animation |
|  2 | Arrested Development  |          2003 | Comedy    |
|  3 | Bob's Burgers         |          2011 | Animation |
|  4 | Bojack Horseman       |          2014 | Animation |
|  5 | Breaking Bad          |          2008 | Drama     |
|  6 | Curb Your Enthusiasm  |          2000 | Comedy    |
|  7 | Fargo                 |          2014 | Drama     |
|  8 | Freaks and Geeks      |          1999 | Comedy    |
|  9 | General Hospital      |          1963 | Drama     |
| 10 | Halt and Catch Fire   |          2014 | Drama     |
| 11 | Malcolm In The Middle |          2000 | Comedy    |
| 12 | Pushing Daisies       |          2007 | Comedy    |
| 13 | Seinfeld              |          1989 | Comedy    |
| 14 | Stranger Things       |          2016 | Drama     |
+----+-----------------------+---------------+-----------+

+----+------------+-----------+
| id | first_name | last_name |
+----+------------+-----------+
|  1 | Thomas     | Stoneman  |
|  2 | Wyatt      | Skaggs    |
|  3 | Kimbra     | Masters   |
|  4 | Domingo    | Cortes    |
|  5 | Colt       | Steele    |
|  6 | Pinkie     | Petit     |
|  7 | Marlon     | Crafford  |
+----+------------+-----------+
7 rows in set (0.01 sec)
```

Challenge #1:
Select title,rating from the series and review tables.

```
+----------------------+--------+
| title                | rating |
```

Create this table:

```
+----------------------+--------+
| title                | rating |
+----------------------+--------+
| Archer               |    8.0 |
| Archer               |    7.5 |
| Archer               |    8.5 |
| Archer               |    7.7 |
| Archer               |    8.9 |
| Arrested Development |    8.1 |
| Arrested Development |    6.0 |
| Arrested Development |    8.0 |
| Arrested Development |    8.4 |
| Arrested Development |    9.9 |
| Bob's Burgers        |    7.0 |
| Bob's Burgers        |    7.5 |
| Bob's Burgers        |    8.0 |
| Bob's Burgers        |    7.1 |
| Bob's Burgers        |    8.0 |
+----------------------+--------+
```

```
select
    title,
    rating
from
    series
    join reviews on series.id = reviews.series_id;
```

```
+----------------------+--------+
| Archer               |    8.0 |
| Archer               |    7.5 |
| Archer               |    8.5 |
| Archer               |    7.7 |
| Archer               |    8.9 |
| Arrested Development |    8.1 |
| Arrested Development |    6.0 |
| Arrested Development |    8.0 |
| Arrested Development |    8.4 |
| Arrested Development |    9.9 |
| Bob's Burgers        |    7.0 |
| Bob's Burgers        |    7.5 |
| Bob's Burgers        |    8.0 |
| Bob's Burgers        |    7.1 |
| Bob's Burgers        |    8.0 |
| Bojack Horseman      |    7.5 |
| Bojack Horseman      |    7.8 |
| Bojack Horseman      |    8.3 |
| Bojack Horseman      |    7.6 |
| Bojack Horseman      |    8.5 |
| Breaking Bad         |    9.5 |
| Breaking Bad         |    9.0 |
| Breaking Bad         |    9.1 |
| Breaking Bad         |    9.3 |
| Breaking Bad         |    9.9 |
| Curb Your Enthusiasm |    6.5 |
| Curb Your Enthusiasm |    7.8 |
| Curb Your Enthusiasm |    8.8 |
| Curb Your Enthusiasm |    8.4 |
| Curb Your Enthusiasm |    9.1 |
| Fargo                |    9.1 |
| Fargo                |    9.7 |
| Freaks and Geeks     |    8.5 |
| Freaks and Geeks     |    7.8 |
| Freaks and Geeks     |    8.8 |
| Freaks and Geeks     |    9.3 |
| General Hospital     |    5.5 |
| General Hospital     |    6.8 |
| General Hospital     |    5.8 |
| General Hospital     |    4.3 |
| General Hospital     |    4.5 |
| Halt and Catch Fire  |    9.9 |
| Seinfeld             |    8.0 |
| Seinfeld             |    7.2 |
| Stranger Things      |    8.5 |
| Stranger Things      |    8.9 |
| Stranger Things      |    8.9 |
+----------------------+--------+
47 rows in set (0.02 sec)
```

Challenge #2:

Create this table:

```
+----------------------+------------+
| title                | avg_rating |
+----------------------+------------+
| General Hospital     |    5.38000 |
| Bob's Burgers        |    7.52000 |
| Seinfeld             |    7.60000 |
| Bojack Horseman      |    7.94000 |
| Arrested Development |    8.08000 |
| Curb Your Enthusiasm |    8.12000 |
| Archer               |    8.12000 |
| Freaks and Geeks     |    8.60000 |
| Stranger Things      |    8.76667 |
| Breaking Bad         |    9.36000 |
| Fargo                |    9.40000 |
| Halt and Catch Fire  |    9.90000 |
+----------------------+------------+
```

```
select
    title,
    avg(rating) as avg_rating
from
    series
    join reviews on series.id = reviews.series_id
group by
    series.id
order by
    avg_rating desc;
```

```
+----------------------+------------+
| title                | avg_rating |
+----------------------+------------+
| Halt and Catch Fire  |    9.90000 |
| Fargo                |    9.40000 |
| Breaking Bad         |    9.36000 |
| Stranger Things      |    8.76667 |
| Freaks and Geeks     |    8.60000 |
| Archer               |    8.12000 |
| Curb Your Enthusiasm |    8.12000 |
| Arrested Development |    8.08000 |
| Bojack Horseman      |    7.94000 |
| Seinfeld             |    7.60000 |
| Bob's Burgers        |    7.52000 |
| General Hospital     |    5.38000 |
+----------------------+------------+
12 rows in set (0.00 sec)
```

Challenge #3:

Create this table:

```
+------------+-----------+--------+
| first_name | last_name | rating |
+------------+-----------+--------+
| Thomas     | Stoneman  |    8.0 |
| Thomas     | Stoneman  |    8.1 |
| Thomas     | Stoneman  |    7.0 |
| Thomas     | Stoneman  |    7.5 |
| Thomas     | Stoneman  |    9.5 |
| Wyatt      | Skaggs    |    7.5 |
| Wyatt      | Skaggs    |    7.6 |
| Wyatt      | Skaggs    |    9.3 |
```

```
+------------+-----------+--------+
| first_name | last_name | rating |
+------------+-----------+--------+
| Colt       | Steele    |    9.9 |
| Colt       | Steele    |    9.9 |
| Colt       | Steele    |    9.9 |
| Colt       | Steele    |    9.7 |
| Thomas     | Stoneman  |    9.5 |
| Colt       | Steele    |    9.3 |
| Wyatt      | Skaggs    |    9.3 |
| Wyatt      | Skaggs    |    9.1 |
| Colt       | Steele    |    9.1 |
| Domingo    | Cortes    |    9.1 |
| Kimbra     | Masters   |    9.0 |
| Colt       | Steele    |    8.9 |
| Wyatt      | Skaggs    |    6.5 |
| Wyatt      | Skaggs    |    8.4 |
| Wyatt      | Skaggs    |    9.1 |
| Wyatt      | Skaggs    |    7.8 |
| Wyatt      | Skaggs    |    5.5 |
| Wyatt      | Skaggs    |    8.5 |
| Kimbra     | Masters   |    8.5 |
| Kimbra     | Masters   |    8.0 |
| Kimbra     | Masters   |    7.1 |
| Kimbra     | Masters   |    7.8 |
| Kimbra     | Masters   |    9.0 |
| Kimbra     | Masters   |    7.8 |
+------------+-----------+--------+
```

```
select
    first_name,
    last_name,
    rating
from
    reviewers
    inner join reviews on reviewers.id = reviews.reviewer_id
order by
    rating desc;
```

```
| Kimbra     | Masters   |    8.9 |
| Domingo    | Cortes    |    8.9 |
| Domingo    | Cortes    |    8.8 |
| Pinkie     | Petit     |    8.8 |
| Colt       | Steele    |    8.5 |
| Wyatt      | Skaggs    |    8.5 |
| Kimbra     | Masters   |    8.5 |
| Domingo    | Cortes    |    8.5 |
| Wyatt      | Skaggs    |    8.4 |
| Pinkie     | Petit     |    8.4 |
| Domingo    | Cortes    |    8.3 |
| Thomas     | Stoneman  |    8.1 |
| Domingo    | Cortes    |    8.0 |
| Colt       | Steele    |    8.0 |
| Kimbra     | Masters   |    8.0 |
| Kimbra     | Masters   |    8.0 |
| Thomas     | Stoneman  |    8.0 |
| Kimbra     | Masters   |    7.8 |
| Wyatt      | Skaggs    |    7.8 |
| Kimbra     | Masters   |    7.8 |
| Domingo    | Cortes    |    7.7 |
| Wyatt      | Skaggs    |    7.6 |
| Wyatt      | Skaggs    |    7.5 |
| Thomas     | Stoneman  |    7.5 |
| Pinkie     | Petit     |    7.5 |
| Domingo    | Cortes    |    7.2 |
| Kimbra     | Masters   |    7.1 |
| Thomas     | Stoneman  |    7.0 |
| Kimbra     | Masters   |    6.8 |
| Wyatt      | Skaggs    |    6.5 |
| Domingo    | Cortes    |    6.0 |
| Domingo    | Cortes    |    5.8 |
| Wyatt      | Skaggs    |    5.5 |
| Colt       | Steele    |    4.5 |
| Pinkie     | Petit     |    4.3 |
+------------+-----------+--------+
47 rows in set (0.02 sec)
```

Challenge #4:
Using "LEFT JOIN" will join tables in a manner that you can see that which entries of Table B is not in Table A.

Create this table:

```
+-----------------------+
| unreviewed_series     |
+-----------------------+
| Malcolm In The Middle |
| Pushing Daisies       |
+-----------------------+
```

```
select
    title,
    rating
from
    series
    left join reviews on series.id = reviews.series_id
where
    rating is NULL;
```

OR

```
select
    title as unreviewed_series
from
    series
    left join reviews on series.id = reviews.series_id
where
    rating is NULL;
```

```
+-----------------------+--------+
| title                 | rating |
+-----------------------+--------+
| Malcolm In The Middle |   NULL |
| Pushing Daisies       |   NULL |
+-----------------------+--------+
2 rows in set (0.01 sec)
```

```
+-----------------------+
| unreviewed_series     |
+-----------------------+
| Malcolm In The Middle |
| Pushing Daisies       |
+-----------------------+
2 rows in set (0.00 sec)
```

Challenge #5(GENRE AVG RATINGS):
Create this table:

```
+-----------+------------+
| genre     | avg_rating |
+-----------+------------+
| Animation |    7.86000 |
| Comedy    |    8.16250 |
| Drama     |    8.04375 |
+-----------+------------+
```

```
select
    genre,
    avg(rating) as avg_rating
from
    series
    join reviews on series.id = reviews.series_id
    group by
        genre;
```

OR

```
+-----------+------------+
| genre     | avg_rating |
+-----------+------------+
| Animation |    7.86000 |
| Comedy    |    8.16250 |
| Drama     |    8.04375 |
+-----------+------------+
3 rows in set (0.00 sec)
```

```
+-----------+------------+
| genre     | avg_rating |
+-----------+------------+
| Animation |       7.86 |
| Comedy    |       8.16 |
| Drama     |       8.04 |
+-----------+------------+
3 rows in set (0.00 sec)
```

Round off the rating to **2 decimals.**

```
select
    genre,
    round(avg(rating), 2) as avg_rating
from
    series
    join reviews on series.id = reviews.series_id
    group by
        genre;
```

Challenge #6(REVIEWER STATS):
Create this table:

```
+------------+-----------+-------+-----+-----+---------+----------+
| first_name | last_name | COUNT | MIN | MAX | AVG     | STATUS   |
+------------+-----------+-------+-----+-----+---------+----------+
| Thomas     | Stoneman  |     5 | 7.0 | 9.5 | 8.02000 | ACTIVE   |
| Wyatt      | Skaggs    |     9 | 5.5 | 9.3 | 7.80000 | ACTIVE   |
| Kimbra     | Masters   |     9 | 6.8 | 9.0 | 7.98889 | ACTIVE   |
| Domingo    | Cortes    |    10 | 5.8 | 9.1 | 7.83000 | ACTIVE   |
| Colt       | Steele    |    10 | 4.5 | 9.9 | 8.77000 | ACTIVE   |
| Pinkie     | Petit     |     4 | 4.3 | 8.8 | 7.25000 | ACTIVE   |
| Marlon     | Crafford  |     0 | 0.0 | 0.0 | 0.00000 | INACTIVE |
+------------+-----------+-------+-----+-----+---------+----------+
```

```
select
    first_name,
    last_name,
    count(rating) as count,
    ifnull(min(rating), 0) as MIN,
    ifnull(max(rating), 0) as MAX,
    ifnull(avg(rating), 0) as AVG,
    case
        when count(rating) >= 1 then "ACTIVE"
        else "INACTIVE"
    end as STATUS
from
    reviewers
    left join reviews on reviewers.id = reviews.reviewer_id
group by
    reviewers.id;
```

```
+------------+-----------+-------+-----+-----+---------+----------+
| first_name | last_name | count | MIN | MAX | AVG     | STATUS   |
+------------+-----------+-------+-----+-----+---------+----------+
| Thomas     | Stoneman  |     5 | 7.0 | 9.5 | 8.02000 | ACTIVE   |
| Wyatt      | Skaggs    |     9 | 5.5 | 9.3 | 7.80000 | ACTIVE   |
| Kimbra     | Masters   |     9 | 6.8 | 9.0 | 7.98889 | ACTIVE   |
| Domingo    | Cortes    |    10 | 5.8 | 9.1 | 7.83000 | ACTIVE   |
| Colt       | Steele    |    10 | 4.5 | 9.9 | 8.77000 | ACTIVE   |
| Pinkie     | Petit     |     4 | 4.3 | 8.8 | 7.25000 | ACTIVE   |
| Marlon     | Crafford  |     0 | 0.0 | 0.0 | 0.00000 | INACTIVE |
+------------+-----------+-------+-----+-----+---------+----------+
7 rows in set (0.00 sec)
```

Using IF statement:
You can also use **if statement** instead of case statement it is simpler but if you want something **more powerful then case statement** are prefered. Below query using if statement produces the same result as above.

```
select
    first_name,
    last_name,
    count(rating) as count,
    ifnull(min(rating), 0) as MIN,
    ifnull(max(rating), 0) as MAX,
    ifnull(avg(rating), 0) as AVG,
    if (count(rating)>=1, "ACTIVE","INACTIVE") as STATUS
from
    reviewers
    left join reviews on reviewers.id = reviews.reviewer_id
group by
    reviewers.id;
```

```
+------------+-----------+-------+-----+-----+---------+----------+
| first_name | last_name | count | MIN | MAX | AVG     | STATUS   |
+------------+-----------+-------+-----+-----+---------+----------+
| Thomas     | Stoneman  |     5 | 7.0 | 9.5 | 8.02000 | ACTIVE   |
| Wyatt      | Skaggs    |     9 | 5.5 | 9.3 | 7.80000 | ACTIVE   |
| Kimbra     | Masters   |     9 | 6.8 | 9.0 | 7.98889 | ACTIVE   |
| Domingo    | Cortes    |    10 | 5.8 | 9.1 | 7.83000 | ACTIVE   |
| Colt       | Steele    |    10 | 4.5 | 9.9 | 8.77000 | ACTIVE   |
| Pinkie     | Petit     |     4 | 4.3 | 8.8 | 7.25000 | ACTIVE   |
| Marlon     | Crafford  |     0 | 0.0 | 0.0 | 0.00000 | INACTIVE |
+------------+-----------+-------+-----+-----+---------+----------+
7 rows in set (0.00 sec)
```

Also if you want to add more than one conditions case statement comes in handy as:

```
select
    first_name,
    last_name,
    count(rating) as count,
    ifnull(min(rating), 0) as MIN,
    ifnull(max(rating), 0) as MAX,
    ifnull(avg(rating), 0) as AVG,
    case
        when count(rating) >= 10 then "POWER USER"
        when count(rating) >0 then "ACTIVE"
        else "INACTIVE"
    end as STATUS
from
    reviewers
    left join reviews on reviewers.id = reviews.reviewer_id
group by
    reviewers.id;
```

```
+------------+-----------+-------+-----+-----+---------+------------+
| first_name | last_name | count | MIN | MAX | AVG     | STATUS     |
+------------+-----------+-------+-----+-----+---------+------------+
| Thomas     | Stoneman  |     5 | 7.0 | 9.5 | 8.02000 | ACTIVE     |
| Wyatt      | Skaggs    |     9 | 5.5 | 9.3 | 7.80000 | ACTIVE     |
| Kimbra     | Masters   |     9 | 6.8 | 9.0 | 7.98889 | ACTIVE     |
| Domingo    | Cortes    |    10 | 5.8 | 9.1 | 7.83000 | POWER USER |
| Colt       | Steele    |    10 | 4.5 | 9.9 | 8.77000 | POWER USER |
| Pinkie     | Petit     |     4 | 4.3 | 8.8 | 7.25000 | ACTIVE     |
| Marlon     | Crafford  |     0 | 0.0 | 0.0 | 0.00000 | INACTIVE   |
+------------+-----------+-------+-----+-----+---------+------------+
7 rows in set (0.00 sec)
```

Challenge #7(JOINING 3 TABLES):
Inner Join reviewer and reviews tables and then inner join the resultant with the series table.

Create this table:

```
+----------------------+--------+-----------------+
| title                | rating | reviewer        |
+----------------------+--------+-----------------+
| Archer               |    8.0 | Thomas Stoneman |
| Archer               |    7.7 | Domingo Cortes  |
| Archer               |    8.5 | Kimbra Masters  |
| Archer               |    7.5 | Wyatt Skaggs    |
| Archer               |    8.9 | Colt Steele     |
| Arrested Development |    8.4 | Pinkie Petit    |
| Arrested Development |    9.9 | Colt Steele     |
| Arrested Development |    8.1 | Thomas Stoneman |
| Arrested Development |    6.0 | Domingo Cortes  |
| Arrested Development |    8.0 | Kimbra Masters  |
| Bob's Burgers        |    7.0 | Thomas Stoneman |
| Bob's Burgers        |    8.0 | Domingo Cortes  |
| Bob's Burgers        |    7.1 | Kimbra Masters  |
| Bob's Burgers        |    7.5 | Pinkie Petit    |
| Bob's Burgers        |    8.0 | Colt Steele     |
+----------------------+--------+-----------------+
```

```
select
    title,
    rating,
    concat(first_name, ' ', last_name) as reviewer
from
    reviewers
    inner join reviews on reviewers.id = reviews.reviewer_id
    inner join series on series.id = reviews.series_id
order by title;
```

```
+----------------------+--------+-----------------+
| title                | rating | reviewer        |
+----------------------+--------+-----------------+
| Archer               |    8.0 | Thomas Stoneman |
| Archer               |    7.5 | Wyatt Skaggs    |
| Archer               |    8.5 | Kimbra Masters  |
| Archer               |    7.7 | Domingo Cortes  |
| Archer               |    8.9 | Colt Steele     |
| Arrested Development |    8.1 | Thomas Stoneman |
| Arrested Development |    6.0 | Domingo Cortes  |
| Arrested Development |    8.4 | Pinkie Petit    |
| Arrested Development |    9.9 | Colt Steele     |
| Arrested Development |    8.0 | Kimbra Masters  |
| Bob's Burgers        |    7.0 | Thomas Stoneman |
| Bob's Burgers        |    8.0 | Domingo Cortes  |
| Bob's Burgers        |    7.5 | Pinkie Petit    |
| Bob's Burgers        |    8.0 | Colt Steele     |
| Bob's Burgers        |    7.1 | Kimbra Masters  |
| Bojack Horseman      |    7.5 | Thomas Stoneman |
| Bojack Horseman      |    8.3 | Domingo Cortes  |
| Bojack Horseman      |    7.6 | Wyatt Skaggs    |
| Bojack Horseman      |    8.5 | Colt Steele     |
| Bojack Horseman      |    7.8 | Kimbra Masters  |
| Breaking Bad         |    9.5 | Thomas Stoneman |
| Breaking Bad         |    9.3 | Wyatt Skaggs    |
| Breaking Bad         |    9.9 | Colt Steele     |
| Breaking Bad         |    9.0 | Kimbra Masters  |
| Breaking Bad         |    9.1 | Domingo Cortes  |
| Curb Your Enthusiasm |    6.5 | Wyatt Skaggs    |
| Curb Your Enthusiasm |    8.8 | Domingo Cortes  |
| Curb Your Enthusiasm |    7.8 | Kimbra Masters  |
| Curb Your Enthusiasm |    9.1 | Colt Steele     |
| Curb Your Enthusiasm |    8.4 | Wyatt Skaggs    |
| Fargo                |    9.7 | Colt Steele     |
| Fargo                |    9.1 | Wyatt Skaggs    |
| Freaks and Geeks     |    8.5 | Domingo Cortes  |
| Freaks and Geeks     |    7.8 | Wyatt Skaggs    |
| Freaks and Geeks     |    9.3 | Colt Steele     |
| Freaks and Geeks     |    8.8 | Pinkie Petit    |
| General Hospital     |    6.8 | Kimbra Masters  |
| General Hospital     |    5.8 | Domingo Cortes  |
| General Hospital     |    5.5 | Wyatt Skaggs    |
| General Hospital     |    4.5 | Colt Steele     |
| General Hospital     |    4.3 | Pinkie Petit    |
| Halt and Catch Fire  |    9.9 | Colt Steele     |
| Seinfeld             |    7.2 | Domingo Cortes  |
| Seinfeld             |    8.0 | Kimbra Masters  |
| Stranger Things      |    8.9 | Kimbra Masters  |
| Stranger Things      |    8.9 | Domingo Cortes  |
| Stranger Things      |    8.5 | Wyatt Skaggs    |
+----------------------+--------+-----------------+
47 rows in set (0.00 sec)
```

Database Triggers

Below trigger prevents to add user having age below 18.
It will first check the condition then allow for insertion.
You've a table(users) with a column(age).

- **DELIMITER $$** line sets temporarily delimiter as **'$$'** because we have multiple lines of sql instead of one sql line in BEGIN and END block.
- **SET MESSAGE_TEXT** will throws a message.
- **SIGNAL SQLSTATE** "user defined exception."
- For an INSERT trigger, **OLD contains no values, and NEW contains the new values**. For an UPDATE trigger, **OLD** contains the old values, and NEW contains the new values. For a DELETE trigger, OLD contains the old values, and NEW contains no values.

```
DELIMITER $$
CREATE TRIGGER must_be_adult
     BEFORE INSERT ON people FOR EACH ROW
     BEGIN
          IF NEW.age < 18
          THEN
              SIGNAL SQLSTATE '45000'
                    SET MESSAGE_TEXT = 'Must be an adult!';
          END IF;
     END;
$$
DELIMITER ;
```

Preventing Self-Follows

```
DELIMITER $$
CREATE TRIGGER example_cannot_follow_self
     BEFORE INSERT ON follows FOR EACH ROW
     BEGIN
          IF NEW.follower_id = NEW.following_id
          THEN
               SIGNAL SQLSTATE '45000'
                    SET MESSAGE_TEXT = 'Cannot follow yourself,
silly';
          END IF;
     END;
$$
DELIMITER ;
```

Logging Unfollows

```
DELIMITER $$

CREATE TRIGGER create_unfollow
AFTER
    DELETE ON follows FOR EACH ROW

BEGIN
INSERT INTO
    unfollows
SET
    follower_id = OLD.follower_id,
    followee_id = OLD.followee_id;
END;

$$
DELIMITER;
```

Listing Triggers

```
SHOW TRIGGERS;
```

Removing triggers

```
DROP TRIGGER trigger_name;
```

Sqltools Vscode integration:
 	
Create new MySQL user with old authentication method:
 	
`CREATE USER 'sqluser'@'%' IDENTIFIED WITH mysql_native_password BY 'password';`
`GRANT ALL PRIVILEGES ON *.* TO 'sqluser'@'%';`
`FLUSH PRIVILEGES;`	

Run in terminal: `mysql.exe -u root -p`	
Work in Progress…	