# Welcome

Got It!

1. Welcome to the course!

Hi, and welcome to Introduction to SQL for Data Science! Most of the world's data live in databases, so learning how to access and unlock insights from these data is an essential skill for every data scientist. SQL, or ess-que-el, is the native language for interacting with databases and is designed for exactly this purpose. This course will give you a basic introduction to SQL. We hope you enjoy it.

2. Let's practice!

Now let's get started!

# Words

|  |  |  |
| --- | --- | --- |
| Command - primary | Command - supplement | Command - Variable |
| SELECT | DISTINCT  COUNT()  AVG  MAX()  SUM()  MIN()  AS | \* [wild card]SELCT (4\*3);  +  -  \* [divide]  / |
| FROM |  |  |
| ; |  |  |
| WHERE | AND  OR  BETWEEN  IN  IS NULL  IS NOT NULL  LIKE  NOT LIKE | ‘ ‘  = equal  <> not equal OR !=  < less than  > greater than  <= less than or equal to  >= greater than or equal to  % |
| ORDER BY | Ascending by default  DESC | = equal  <> not equal OR !=  < less than  > greater than  <= less than or equal to  >= greater than or equal to |
| HAVING |  | = equal  <> not equal OR !=  < less than  > greater than  <= less than or equal to  >= greater than or equal to |
| LIMIT |  | = equal  <> not equal OR !=  < less than  > greater than  <= less than or equal to  >= greater than or equal to |

# Code examples

SELECT COUNT(\*)

FROM films

WHERE language IS NULL;

SELECT title, release\_year

FROM films

WHERE release\_year BETWEEN '1990' AND '2000'

AND budget > 100000000

AND (language = 'Spanish' OR language = 'French');

SELECT title, certification

FROM films

WHERE certification IN ('NC-17', 'R');

SELECT AVG(gross)

FROM films

WHERE title LIKE('A%');

SELECT MAX(gross)

FROM films

WHERE release\_year BETWEEN '2000' AND '2012';

SELECT title, gross

FROM films

WHERE title LIKE('M%')

ORDER BY title;

SELECT sex, count(\*)

FROM employees

GROUP BY sex

ORDER BY count DESC;

SELECT release\_year, COUNT(\*)

From films

GROUP BY release\_year;

SELECT release\_year, AVG(duration)

FROM films

GROUP BY release\_year;

SELECT release\_year, MAX(budget)

FROM films

GROUP BY release\_year;

SELECT imdb\_score, COUNT(\*)

FROM reviews

GROUP BY imdb\_score;

SELECT release\_year, MIN(gross)

FROM films

GROUP BY release\_year;

SELECT language, SUM(gross)

FROM films

GROUP BY language;

SELECT release\_year, country, MAX(budget)

FROM films

GROUP BY release\_year, country

ORDER BY release\_year, country;

SELECT release\_year

FROM films

GROUP BY release\_year

HAVING COUNT(title) > 200;

SELECT release\_year, AVG(budget) AS avg\_budget, AVG(gross) AS avg\_gross

FROM films

WHERE release\_year > 1990

GROUP BY release\_year

HAVING AVG(budget) > 60000000

ORDER BY AVG(gross) DESC;

SELECT country, AVG(budget) AS avg\_budget, AVG(gross) AS avg\_gross

FROM films

GROUP BY country

HAVING COUNT(title) > 10

ORDER BY country

LIMIT 5;

SELECT title, imdb\_score

FROM films

JOIN reviews

ON films.id = reviews.film\_id

WHERE title = 'To Kill a Mockingbird';

SELECT title, (gross-budget) AS net\_profit

FROM films;

SELECT AVG(duration) / 60.0 AS avg\_duration\_hours

FROM films;

-- get the count(deathdate) and multiply by 100.0

SELECT COUNT(deathdate) \* 100.0 / COUNT(\*) AS percentage\_dead

FROM people;

-- then divide by count(\*)

SELECT \*

FROM films

WHERE release\_year <> '2015'

ORDER BY duration;

SELECT imdb\_score, film\_id

FROM reviews

ORDER BY imdb\_score DESC;

SELECT title, duration

FROM films

ORDER BY duration DESC;

SELECT release\_year, duration, title

FROM films

ORDER BY release\_year, duration;

SELECT release\_year

FROM films

GROUP BY release\_year

HAVING COUNT(title) > 200;