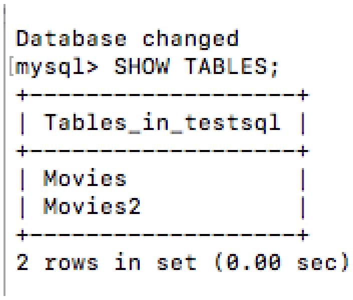
mysql> create database testsql;
Query OK, 1 row affected (0.00 sec)

// AFTER DATABASE IS CREATED
mysql> show databases;
mysql> use testsql;

mysql> show tables;
Empty set (0.00 sec)

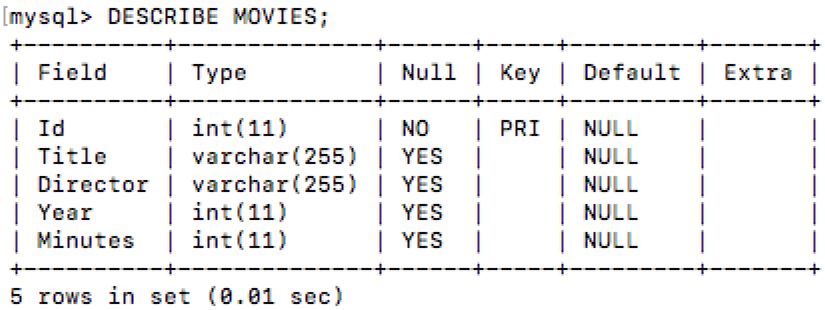
mysql> CREATE TABLE Movies (Id int NOT NULL UNIQUE, Title varchar(255), Director varchar(255), Year int, Minutes int);
Query OK, 0 rows affected (0.05 sec)

mysql> show tables;



```
Database changed
[mysql> SHOW TABLES;
+-------------------+
| Tables_in_testsql |
+-------------------+
| Movies            |
| Movies2           |
+-------------------+
2 rows in set (0.00 sec)
```

mysql> describe Movies;

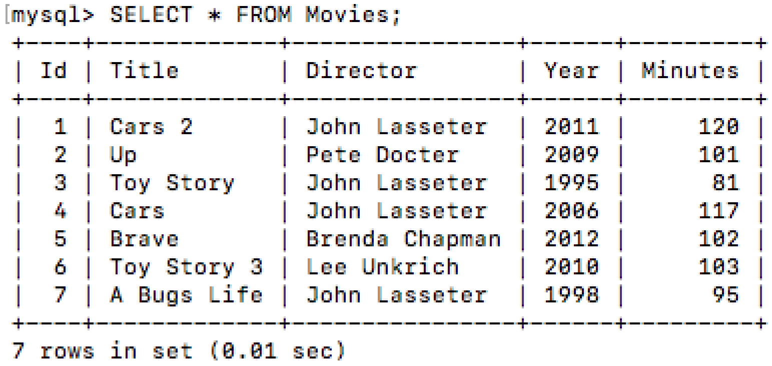


```
[mysql> DESCRIBE MOVIES;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| Id       | int(11)      | NO   | PRI | NULL    |       |
| Title    | varchar(255) | YES  |     | NULL    |       |
| Director | varchar(255) | YES  |     | NULL    |       |
| Year     | int(11)      | YES  |     | NULL    |       |
| Minutes  | int(11)      | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

INSERT INTO Movies2 (Id, Title, Director, Year, Minutes) VALUES ('4', 'Wall-E', 'Andrew Scanton', '2008', '104'), ('5', 'Toy Story 2', 'John Lasseter', '1999', '93'), ('6', 'The Incredibles', 'Brad Bird', '2004', '116'), ('7', 'Finding Nemo', 'Andrew Stanton', '2003', '107');

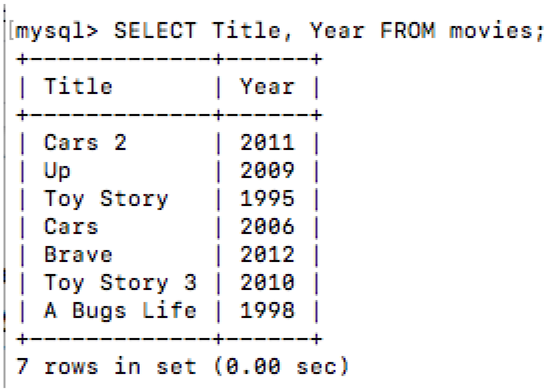
DELETE from Movies2 WHERE Id=2;

SELECT * FROM Movies;



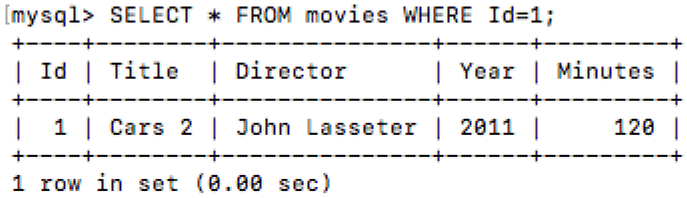
```
mysql> SELECT * FROM Movies;
+----+-------------+----------------+------+---------+
| Id | Title       | Director       | Year | Minutes |
+----+-------------+----------------+------+---------+
|  1 | Cars 2      | John Lasseter  | 2011 |     120 |
|  2 | Up          | Pete Docter    | 2009 |     101 |
|  3 | Toy Story   | John Lasseter  | 1995 |      81 |
|  4 | Cars        | John Lasseter  | 2006 |     117 |
|  5 | Brave       | Brenda Chapman | 2012 |     102 |
|  6 | Toy Story 3 | Lee Unkrich    | 2010 |     103 |
|  7 | A Bugs Life | John Lasseter  | 1998 |      95 |
+----+-------------+----------------+------+---------+
7 rows in set (0.01 sec)
```

mysql> SELECT Title, Year FROM movies;



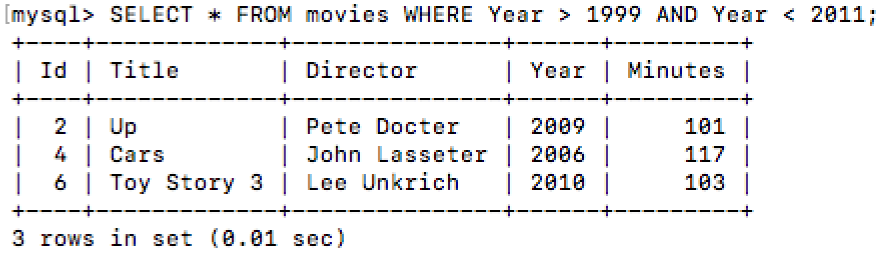
```
mysql> SELECT Title, Year FROM movies;
+-------------+------+
| Title       | Year |
+-------------+------+
| Cars 2      | 2011 |
| Up          | 2009 |
| Toy Story   | 1995 |
| Cars        | 2006 |
| Brave       | 2012 |
| Toy Story 3 | 2010 |
| A Bugs Life | 1998 |
+-------------+------+
7 rows in set (0.00 sec)
```

mysql> SELECT * FROM movies WHERE Id=1;

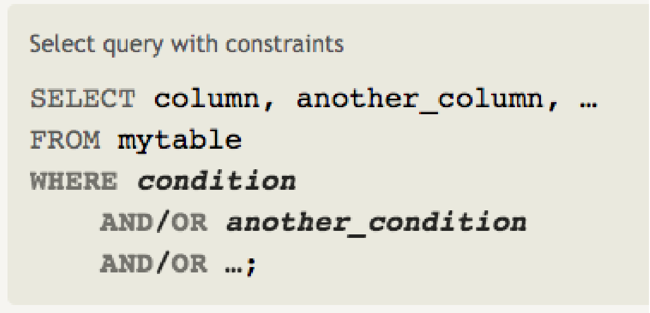


```
mysql> SELECT * FROM movies WHERE Id=1;
+----+--------+---------------+------+---------+
| Id | Title  | Director      | Year | Minutes |
+----+--------+---------------+------+---------+
|  1 | Cars 2 | John Lasseter | 2011 |     120 |
+----+--------+---------------+------+---------+
1 row in set (0.00 sec)
```

mysql> SELECT * FROM movies WHERE Year > 1999 AND Year < 2011;
OR
mysql> SELECT * FROM movies WHERE Year BETWEEN 2000 AND 2010;

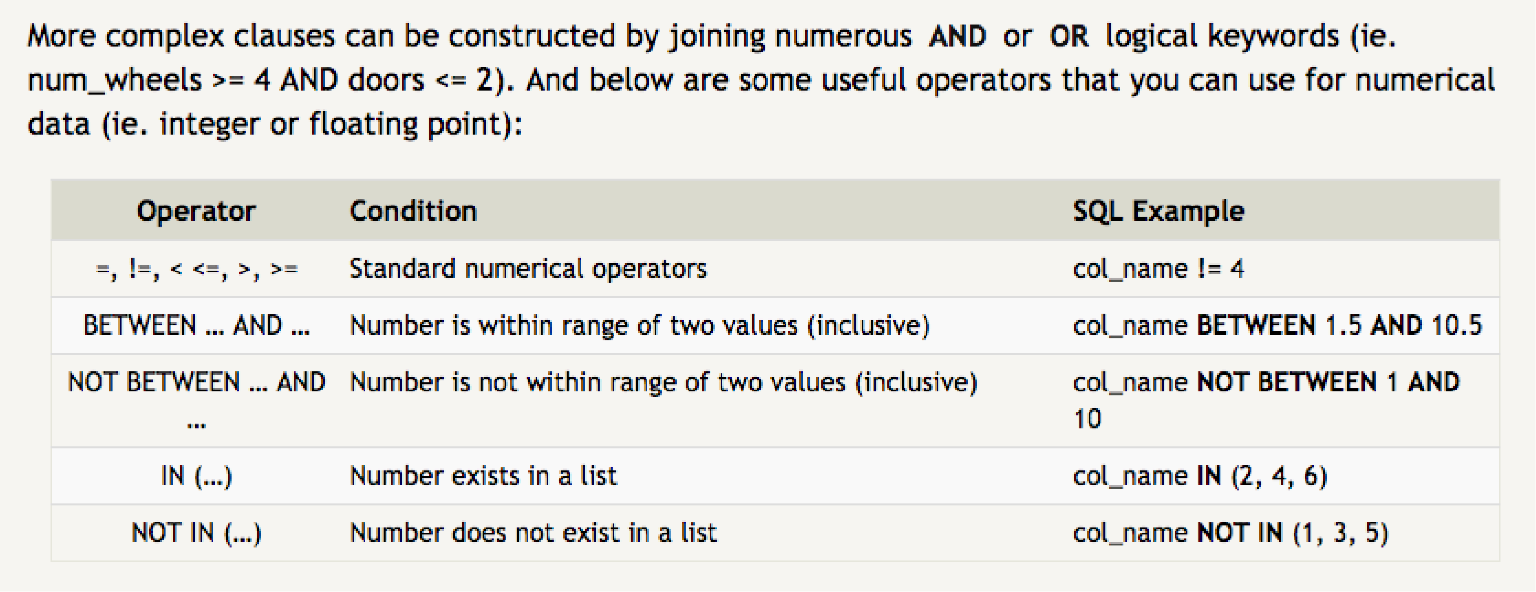


```
mysql> SELECT * FROM movies WHERE Year > 1999 AND Year < 2011;
+----+-------------+---------------+------+---------+
| Id | Title       | Director      | Year | Minutes |
+----+-------------+---------------+------+---------+
|  2 | Up          | Pete Docter   | 2009 |     101 |
|  4 | Cars        | John Lasseter | 2006 |     117 |
|  6 | Toy Story 3 | Lee Unkrich   | 2010 |     103 |
+----+-------------+---------------+------+---------+
3 rows in set (0.01 sec)
```



Select query with constraints

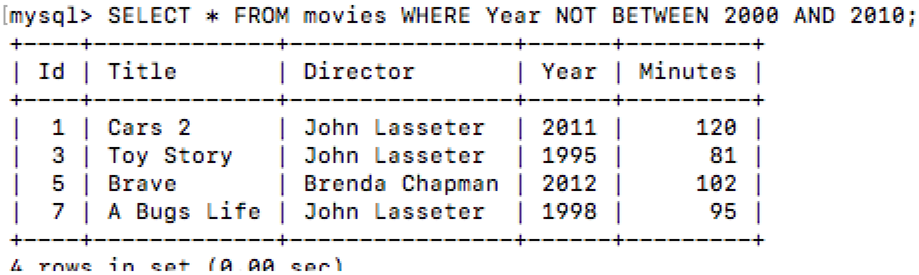
```
SELECT column, another_column, …
FROM mytable
WHERE condition
    AND/OR another_condition
    AND/OR …;
```



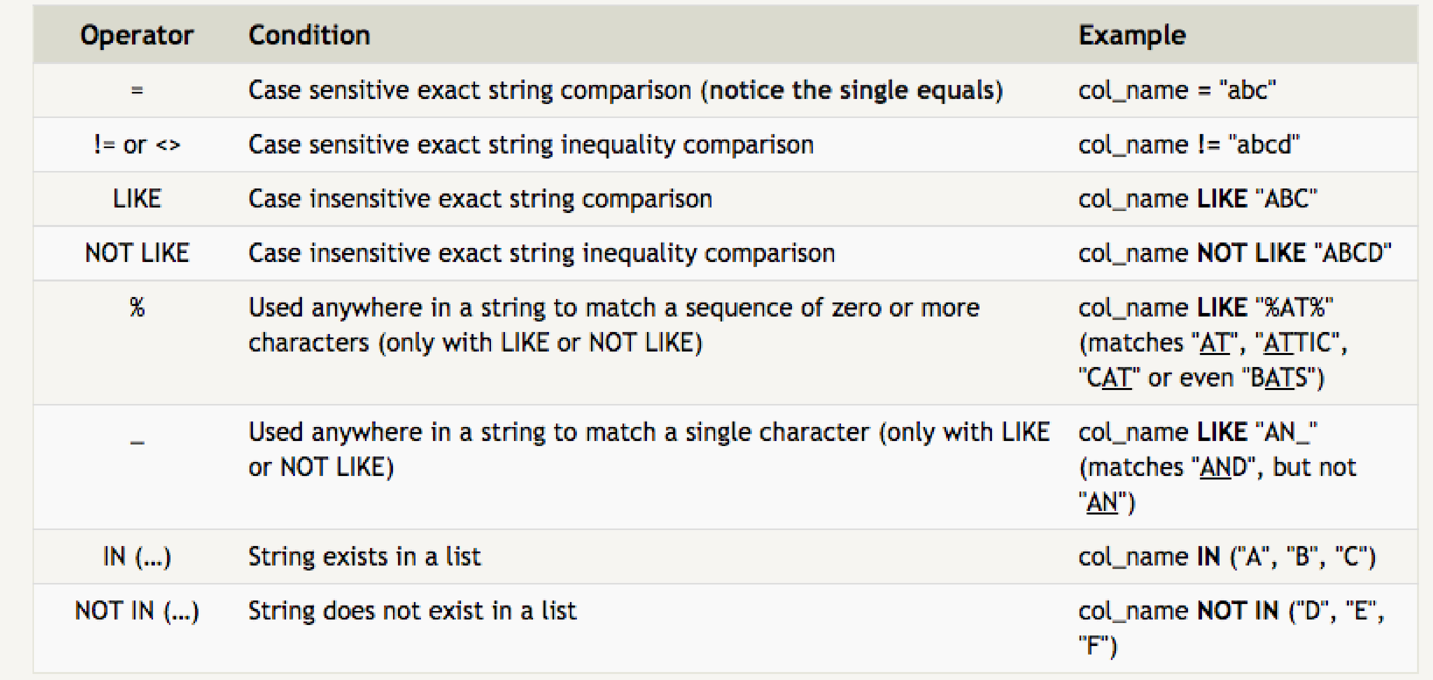
More complex clauses can be constructed by joining numerous **AND** or **OR** logical keywords (ie. num_wheels >= 4 AND doors <= 2). And below are some useful operators that you can use for numerical data (ie. integer or floating point):

Operator	Condition	SQL Example
=, !=, < <=, >, >=	Standard numerical operators	col_name != 4
BETWEEN … AND …	Number is within range of two values (inclusive)	col_name **BETWEEN** 1.5 **AND** 10.5
NOT BETWEEN … AND …	Number is not within range of two values (inclusive)	col_name **NOT BETWEEN** 1 **AND** 10
IN (…)	Number exists in a list	col_name **IN** (2, 4, 6)
NOT IN (…)	Number does not exist in a list	col_name **NOT IN** (1, 3, 5)

SELECT * FROM movies WHERE Year NOT BETWEEN 2000 AND 2010;

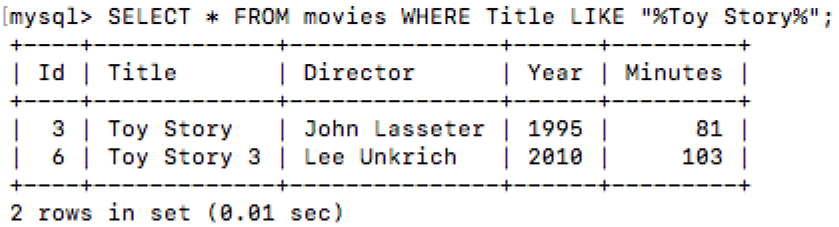


```
mysql> SELECT * FROM movies WHERE Year NOT BETWEEN 2000 AND 2010;
+----+-------------+----------------+------+---------+
| Id | Title       | Director       | Year | Minutes |
+----+-------------+----------------+------+---------+
|  1 | Cars 2      | John Lasseter  | 2011 |     120 |
|  3 | Toy Story   | John Lasseter  | 1995 |      81 |
|  5 | Brave       | Brenda Chapman | 2012 |     102 |
|  7 | A Bugs Life | John Lasseter  | 1998 |      95 |
+----+-------------+----------------+------+---------+
4 rows in set (0.00 sec)
```



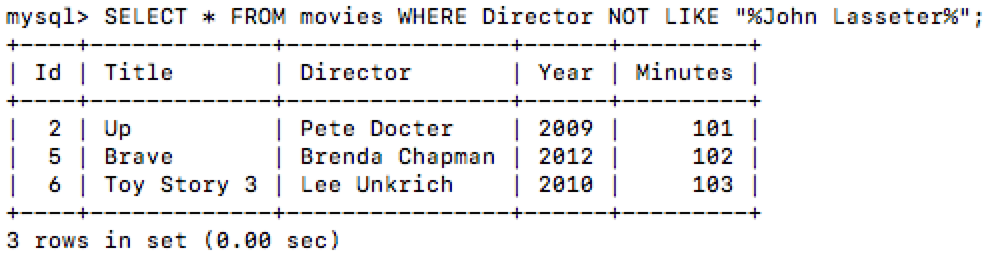
Operator	Condition	Example
=	Case sensitive exact string comparison (**notice the single equals**)	col_name = "abc"
!= or <>	Case sensitive exact string inequality comparison	col_name != "abcd"
LIKE	Case insensitive exact string comparison	col_name **LIKE** "ABC"
NOT LIKE	Case insensitive exact string inequality comparison	col_name **NOT LIKE** "ABCD"
%	Used anywhere in a string to match a sequence of zero or more characters (only with LIKE or NOT LIKE)	col_name **LIKE** "%AT%" (matches "AT", "ATTIC", "CAT" or even "BATS")
_	Used anywhere in a string to match a single character (only with LIKE or NOT LIKE)	col_name **LIKE** "AN_" (matches "AND", but not "AN")
IN (…)	String exists in a list	col_name **IN** ("A", "B", "C")
NOT IN (…)	String does not exist in a list	col_name **NOT IN** ("D", "E", "F")

SELECT * FROM movies WHERE Title LIKE "%Toy Story%";



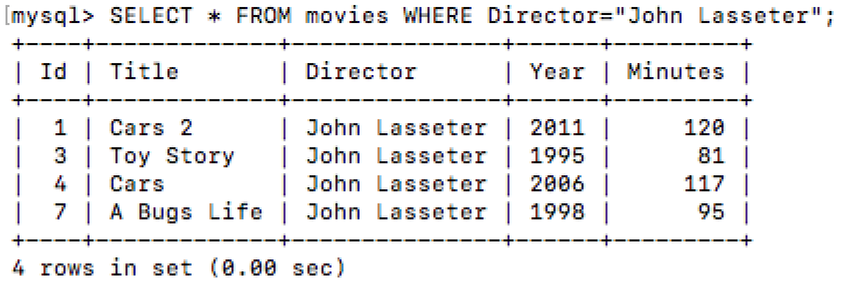
```
[mysql> SELECT * FROM movies WHERE Title LIKE "%Toy Story%";
+----+-------------+---------------+------+---------+
| Id | Title       | Director      | Year | Minutes |
+----+-------------+---------------+------+---------+
|  3 | Toy Story   | John Lasseter | 1995 |      81 |
|  6 | Toy Story 3 | Lee Unkrich   | 2010 |     103 |
+----+-------------+---------------+------+---------+
2 rows in set (0.01 sec)
```

SELECT * FROM movies WHERE Director NOT LIKE "%John Lasseter%";



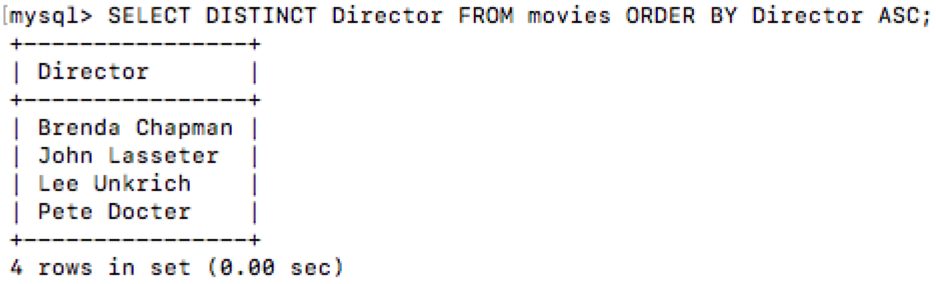
```
mysql> SELECT * FROM movies WHERE Director NOT LIKE "%John Lasseter%";
+----+-------------+----------------+------+---------+
| Id | Title       | Director       | Year | Minutes |
+----+-------------+----------------+------+---------+
|  2 | Up          | Pete Docter    | 2009 |     101 |
|  5 | Brave       | Brenda Chapman | 2012 |     102 |
|  6 | Toy Story 3 | Lee Unkrich    | 2010 |     103 |
+----+-------------+----------------+------+---------+
3 rows in set (0.00 sec)
```

SELECT * FROM movies WHERE Director="John Lasseter";



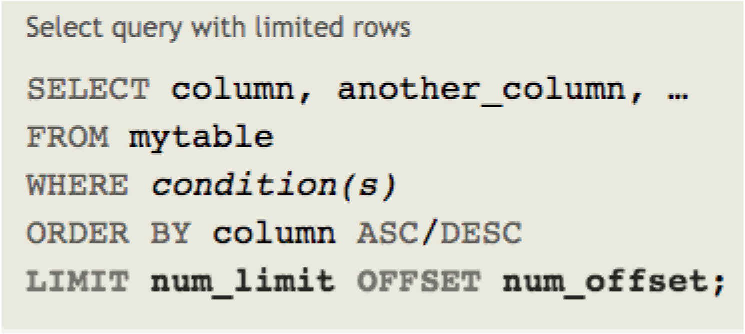
```
[mysql> SELECT * FROM movies WHERE Director="John Lasseter";
+----+-------------+---------------+------+---------+
| Id | Title       | Director      | Year | Minutes |
+----+-------------+---------------+------+---------+
|  1 | Cars 2      | John Lasseter | 2011 |     120 |
|  3 | Toy Story   | John Lasseter | 1995 |      81 |
|  4 | Cars        | John Lasseter | 2006 |     117 |
|  7 | A Bugs Life | John Lasseter | 1998 |      95 |
+----+-------------+---------------+------+---------+
4 rows in set (0.00 sec)
```

// Get all **unique (DISTINCT)** directors list alphabetically by first name
SELECT DISTINCT Director FROM movies ORDER BY Director ASC;



```
[mysql> SELECT DISTINCT Director FROM movies ORDER BY Director ASC;
+----------------+
| Director       |
+----------------+
| Brenda Chapman |
| John Lasseter  |
| Lee Unkrich    |
| Pete Docter    |
+----------------+
4 rows in set (0.00 sec)
```

num_offset = start from specific row #, num_limit = # of rows allowed

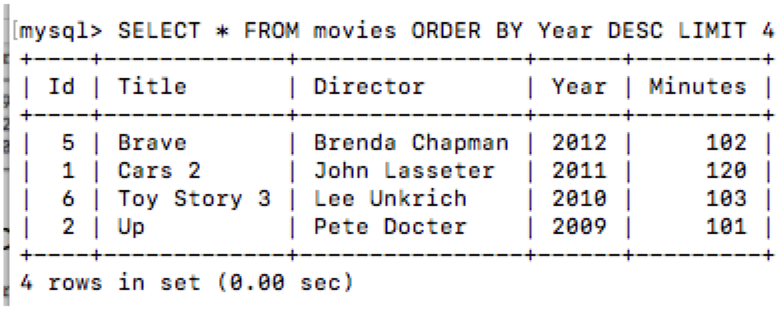


Select query with limited rows

```
SELECT column, another_column, …
FROM mytable
WHERE condition(s)
ORDER BY column ASC/DESC
LIMIT num_limit OFFSET num_offset;
```

// List 4 most recent movies ordered by most recent

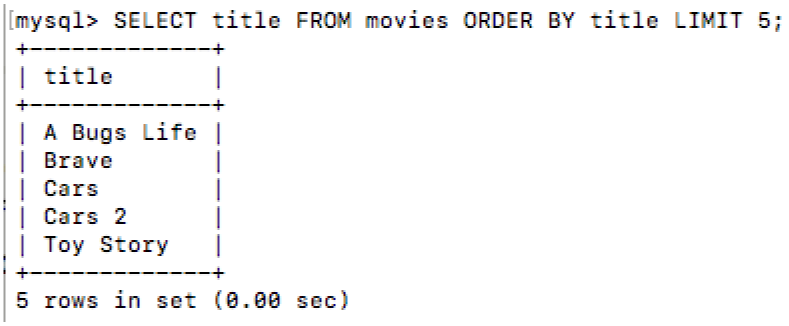
SELECT * FROM movies ORDER BY Year DESC LIMIT 4;



```
mysql> SELECT * FROM movies ORDER BY Year DESC LIMIT 4;
+----+-------------+----------------+------+---------+
| Id | Title       | Director       | Year | Minutes |
+----+-------------+----------------+------+---------+
|  5 | Brave       | Brenda Chapman | 2012 |     102 |
|  1 | Cars 2      | John Lasseter  | 2011 |     120 |
|  6 | Toy Story 3 | Lee Unkrich    | 2010 |     103 |
|  2 | Up          | Pete Docter    | 2009 |     101 |
+----+-------------+----------------+------+---------+
4 rows in set (0.00 sec)
```

Select First Five movies sorted alphabetically

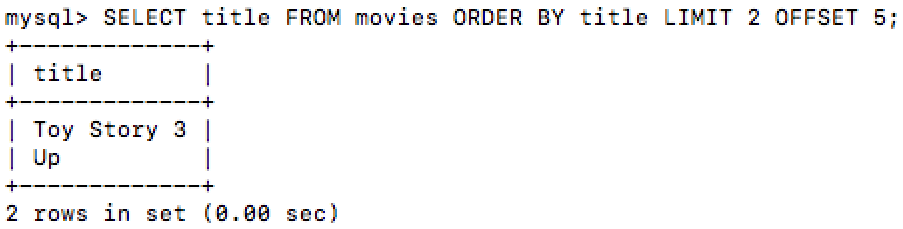
SELECT title FROM movies ORDER BY title LIMIT 5;



```
mysql> SELECT title FROM movies ORDER BY title LIMIT 5;
+-------------+
| title       |
+-------------+
| A Bugs Life |
| Brave       |
| Cars        |
| Cars 2      |
| Toy Story   |
+-------------+
5 rows in set (0.00 sec)
```

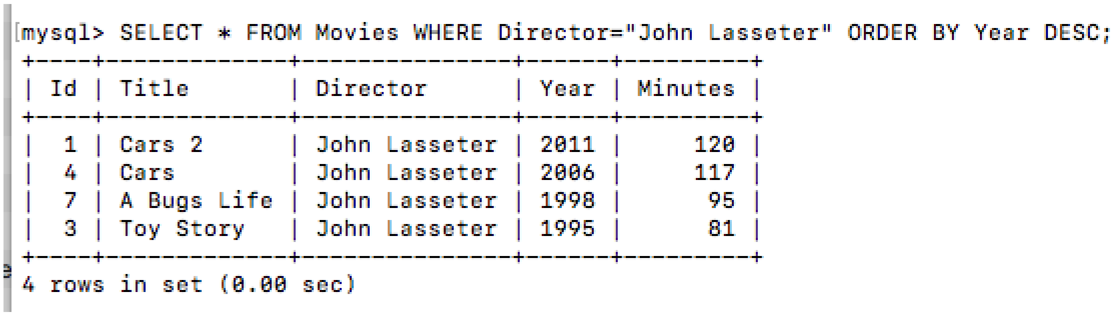
Select the next 2 movies listed alphabetically AFTER the first five

SELECT title FROM movies ORDER BY title LIMIT 2 OFFSET 5;



```
mysql> SELECT title FROM movies ORDER BY title LIMIT 2 OFFSET 5;
+-------------+
| title       |
+-------------+
| Toy Story 3 |
| Up          |
+-------------+
2 rows in set (0.00 sec)
```

mysql> SELECT * FROM Movies WHERE Director="John Lasseter" ORDER BY Year DESC;



```
mysql> SELECT * FROM Movies WHERE Director="John Lasseter" ORDER BY Year DESC;
+----+-------------+---------------+------+---------+
| Id | Title       | Director      | Year | Minutes |
+----+-------------+---------------+------+---------+
|  1 | Cars 2      | John Lasseter | 2011 |     120 |
|  4 | Cars        | John Lasseter | 2006 |     117 |
|  7 | A Bugs Life | John Lasseter | 1998 |      95 |
|  3 | Toy Story   | John Lasseter | 1995 |      81 |
+----+-------------+---------------+------+---------+
4 rows in set (0.00 sec)
```

List all the cities west of Chicago, ordered from west to east

```
SELECT city
FROM north_american_cities
WHERE longitude < -87.629798 ORDER BY Longitude ASC;
```

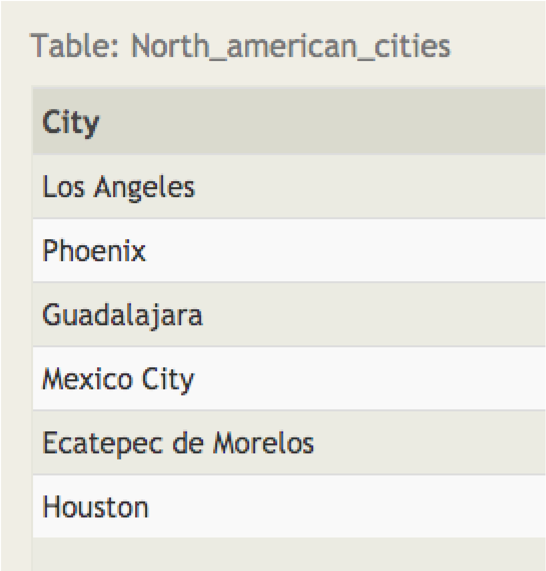


Table: North_american_cities

City
Los Angeles
Phoenix
Guadalajara
Mexico City
Ecatepec de Morelos
Houston

```
// THIS IS SAME, BUT ONLY FOR UNITED STATES
SELECT * FROM north_american_cities WHERE Country="United States" AND longitude <
-87.629798 ORDER BY longitude ASC;
```

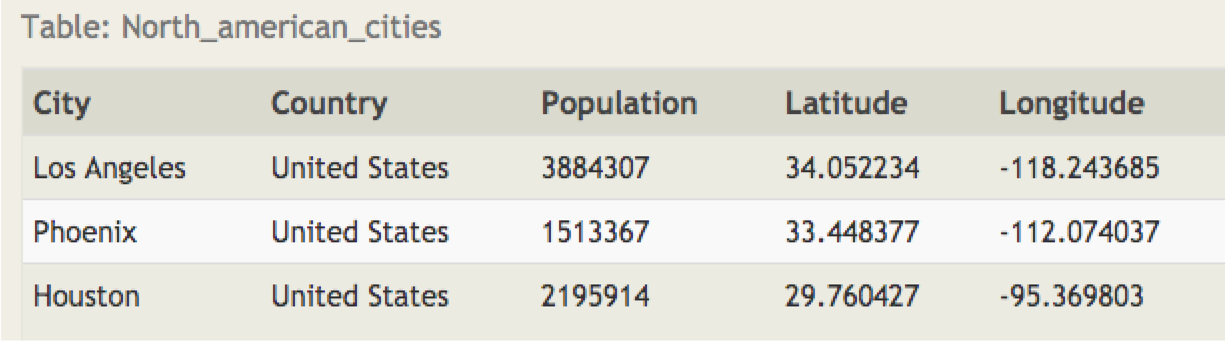


Table: North_american_cities

City	Country	Population	Latitude	Longitude
Los Angeles	United States	3884307	34.052234	-118.243685
Phoenix	United States	1513367	33.448377	-112.074037
Houston	United States	2195914	29.760427	-95.369803