Select Query

The SELECT query in SQL is used to retrieve data from a database. It allows you to specify exactly which columns and rows you want to fetch, making it a fundamental tool for data analysis.

```
mysql> select * from movies;
+-------+-----------------------------------------------------------------------------------------------------+------+-----------+
| id    | name                                                                                                | year | rankscore |
+-------+-----------------------------------------------------------------------------------------------------+------+-----------+
|     0 | #28                                                                                                 | 2002 |      NULL |
|     1 | #7 Train: An Immigrant Journey, The                                                                 | 2000 |      NULL |
|     2 | $                                                                                                   | 1971 |       6.4 |
|     3 | $1,000 Reward                                                                                       | 1913 |      NULL |
|     4 | $1,000 Reward                                                                                       | 1915 |      NULL |
|     5 | $1,000 Reward                                                                                       | 1923 |      NULL |
|     6 | $1,000,000 Duck                                                                                     | 1971 |         5 |
|     7 | $1,000,000 Reward, The                                                                              | 1920 |      NULL |
|     8 | $10,000 Under a Pillow                                                                              | 1921 |      NULL |
|     9 | $100,000                                                                                            | 1915 |      NULL |
|    10 | $100,000 Pyramid, The                                                                               | 2001 |      NULL |
|    11 | $1000 a Touchdown                                                                                   | 1939 |       6.7 |
```

```
mysql> select name, year from movies;
+-----------------------------------------------------------------------------------------------------+------+
| name                                                                                                | year |
+-----------------------------------------------------------------------------------------------------+------+
| #28                                                                                                 | 2002 |
| #7 Train: An Immigrant Journey, The                                                                 | 2000 |
| $                                                                                                   | 1971 |
| $1,000 Reward                                                                                       | 1913 |
| $1,000 Reward                                                                                       | 1915 |
| $1,000 Reward                                                                                       | 1923 |
| $1,000,000 Duck                                                                                     | 1971 |
| $1,000,000 Reward, The                                                                              | 1920 |
| $10,000 Under a Pillow                                                                              | 1921 |
| $100,000                                                                                            | 1915 |
| $100,000 Pyramid, The                                                                               | 2001 |
| $1000 a Touchdown                                                                                   | 1939 |
```

You can also:

- Retrieve all columns using SELECT *
- Filter results with WHERE
- Sort data using ORDER BY
- Remove duplicates with DISTINCT

The SELECT query is the foundation of working with data in SQL, and mastering it is key to exploring and analyzing data efficiently.

In databases, limit and offset are commonly used to control the number of records returned in a query and the starting point for fetching data. They are often used together in pagination to display results in chunks (like pages of a list).

Limit

Definition: The LIMIT clause is used to specify the maximum number of rows or records to retrieve from the database or result set.

Usage: If you only need a subset of records from a query, you can set a limit to reduce the amount of data returned.

Example SQL query:

```
mysql> select name, rankscore from movies limit 20;
+-------------------------------------+-----------+
| name                                | rankscore |
+-------------------------------------+-----------+
| #28                                 |      NULL |
| #7 Train: An Immigrant Journey, The |      NULL |
| $                                   |       6.4 |
| $1,000 Reward                       |      NULL |
| $1,000 Reward                       |      NULL |
| $1,000 Reward                       |      NULL |
| $1,000,000 Duck                     |         5 |
| $1,000,000 Reward, The              |      NULL |
| $10,000 Under a Pillow              |      NULL |
| $100,000                            |      NULL |
| $100,000 Pyramid, The               |      NULL |
| $1000 a Touchdown                   |       6.7 |
| $20,000 Carat, The                  |      NULL |
| $21 a Day Once a Month              |      NULL |
| $2500 Bride, The                    |      NULL |
| $30                                 |       7.5 |
| $30,000                             |      NULL |
| $300 y tickets                      |      NULL |
| $40,000                             |       9.6 |
| $5,000 Reward                       |      NULL |
+-------------------------------------+-----------+
20 rows in set (0.00 sec)
```

Offset:

Definition: The OFFSET clause is used to specify the number of rows to skip before starting to return records. It is useful for pagination.

Usage: It helps in specifying where to start the result set. For example, in the second page of a list, you might want to skip the first 20 records.

Example SQL query:

```
mysql> select name, rankscore from movies limit 20 offset 20;
+-------------------------------------+-----------+
| name                                | rankscore |
+-------------------------------------+-----------+
| $5,000,000 Counterfeiting Plot, The |      NULL |
| $5.15/Hr.                           |      NULL |
| $5.20 an Hour Dream, The            |      NULL |
| $50,000 Challenge, The              |      NULL |
| $50,000 Climax Show, The            |       2.6 |
| $50,000 Jewel Theft, The            |      NULL |
| $50,000 Reward                      |      NULL |
| $500 Reward, The                    |      NULL |
| $500,000 Reward                     |      NULL |
| $pent                               |       4.3 |
| $ucces Part One                     |      NULL |
| $windle                             |       5.4 |
| & frres                             |      NULL |
| ''Bear'' Facts, The                 |      NULL |
| '15'                                |       6.8 |
| '24-25' ne vozvrashchayetsya        |      NULL |
| '38                                 |       6.7 |
| '42                                 |      NULL |
| '49, un souffle de colre            |      NULL |
| '49-'17                             |       5.8 |
+-------------------------------------+-----------+
20 rows in set (0.00 sec)
```

Order By:

The ORDER BY clause is used in SQL to sort the result set of a query based on one or more columns. It allows you to arrange the data either in ascending (default) or descending order.

Here's a more detailed explanation:

Syntax:

SELECT column1, column2, ...

FROM table_name

ORDER BY column_name [**ASC**|**DESC**];

Example:

Order by using with ASC(Ascending order)

```
mysql> select name, rankscore, year from movies order by year limit 20;
+-------------------------------------+-----------+------+
| name                                | rankscore | year |
+-------------------------------------+-----------+------+
| Traffic Crossing Leeds Bridge       |      NULL | 1888 |
| Roundhay Garden Scene               |      NULL | 1888 |
| Monkeyshines, No. 1                 |       7.3 | 1890 |
| Monkeyshines, No. 3                 |      NULL | 1890 |
| Monkeyshines, No. 2                 |      NULL | 1890 |
| Duncan Smoking                      |       3.6 | 1891 |
| Duncan or Devonald with Muslin Cloud |       3.5 | 1891 |
| Duncan and Another, Blacksmith Shop |       3.5 | 1891 |
| Newark Athlete                      |       4.3 | 1891 |
| Men Boxing                          |         4 | 1891 |
| Monkey and Another, Boxing          |       3.2 | 1891 |
| Fencing                             |       4.1 | 1892 |
| Hand Shake, A                       |       2.6 | 1892 |
| Boxing                              |       1.9 | 1892 |
| Wrestling                           |         2 | 1892 |
| Clown et ses chiens, Le             |       3.8 | 1892 |
| Un bon bock                         |         2 | 1892 |
| Prince de Galles, Le                |       2.9 | 1892 |
| Pauvre Pierrot                      |       5.1 | 1892 |
| Man on Parallel Bars                |       1.4 | 1892 |
+-------------------------------------+-----------+------+
20 rows in set (0.23 sec)
```

2nd example:

Order by using with DESC(Descending order)

```
mysql> select name, rankscore, year from movies order by year desc limit 20;
+-----------------------------------------+-----------+------+
| name                                    | rankscore | year |
+-----------------------------------------+-----------+------+
| Harry Potter and the Half-Blood Prince  |      NULL | 2008 |
| War of the Red Cliff, The               |      NULL | 2007 |
| Tripoli                                 |      NULL | 2007 |
| Spider-Man 3                            |      NULL | 2007 |
| Rapunzel Unbraided                      |      NULL | 2007 |
| Harry Potter and the Order of the Phoenix |      NULL | 2007 |
| Untitled Star Trek Prequel              |      NULL | 2007 |
| DragonBall Z                            |      NULL | 2007 |
| Back to School                          |      NULL | 2006 |
| 2176                                    |      NULL | 2006 |
| Band on the Run                         |      NULL | 2006 |
| Andrew Henry's Meadow                   |      NULL | 2006 |
| Big Bug Man                             |      NULL | 2006 |
| Bielski Brothers, The                   |      NULL | 2006 |
| Benjamin Button                         |      NULL | 2006 |
| Becoming Jane                           |      NULL | 2006 |
| Astrix et les Vikings                   |      NULL | 2006 |
| Balls of Courage                        |      NULL | 2006 |
| Big Baby                                |      NULL | 2006 |
| Arthur, the Movie                       |      NULL | 2006 |
+-----------------------------------------+-----------+------+
20 rows in set (0.22 sec)
```

Distinct:

The DISTINCT keyword in SQL is used to remove duplicate values from the result set of a query. It ensures that the query returns only unique values for the specified columns. It is helpful when you want to retrieve a list of values without repetition.

Syntax:

SELECT DISTINCT column1, column2, ...

FROM table_name;

Example:

```
mysql> select distinct genre from movies_genres;
+-------------+
| genre       |
+-------------+
| Documentary |
| Short       |
| Comedy      |
| Crime       |
| Western     |
| Family      |
| Animation   |
| Drama       |
| Romance     |
| Mystery     |
| Thriller    |
| Adult       |
| Music       |
| Action      |
| Fantasy     |
| Sci-Fi      |
| Horror      |
| War         |
| Musical     |
| Adventure   |
| Film-Noir   |
+-------------+
21 rows in set (0.45 sec)
```

2nd example:

```
mysql> select distinct first_name, last_name from directors order by first_name limit 10;
+---------------+-------------+
| first_name    | last_name   |
+---------------+-------------+
| 'Chico'       | Hernandez   |
| 'Philthy' Phil | Phillips    |
| 'Weird Al'    | Yankovic    |
| A.            | Aleksandrov |
| A.            | Babes       |
| A.            | Balakrishnan |
| A.            | Barr-Smith  |
| A.            | Berry       |
| A.            | Bhimsingh   |
| A.            | Bistritsky  |
+---------------+-------------+
10 rows in set (0.02 sec)
```

WHERE, Comparison operators, NULL

In SQL, the WHERE clause is used to filter records based on specific conditions. It allows you to specify criteria for selecting rows that meet a certain condition. This is useful when you only want to retrieve or manipulate certain rows of data from a table, rather than all of them.

Syntax:

SELECT column1, column2, ...

FROM table_name

WHERE condition;

Key Points:

1. **Condition**: The condition specified after the WHERE clause can include comparisons (e.g., =, >, <, <>, etc.), logical operators (e.g., AND, OR), and other expressions to filter data.
2. **Filtering Rows**: The WHERE clause only returns rows that meet the condition specified. If no rows meet the condition, the result will be empty.
3. **Multiple Conditions**: You can combine multiple conditions using AND, OR, and NOT to create more complex filters.

Examples:

```
mysql> select name, year, rankscore from movies where rankscore > 9 limit 20;
+-------------------------------------------------------+------+-----------+
| name                                                  | year | rankscore |
+-------------------------------------------------------+------+-----------+
| $40,000                                               | 1996 |       9.6 |
| +1 -1                                                 | 1987 |       9.6 |
| 12 (2003/II)                                          | 2003 |       9.8 |
| 12 stulyev                                            | 1971 |       9.3 |
| 14 Million Dreams                                     | 2003 |       9.5 |
| 2+1                                                   | 2004 |       9.2 |
| 2wks, 1yr                                             | 2002 |       9.4 |
| 36K                                                   | 2000 |       9.5 |
| 5 Card Stud                                           | 2002 |       9.4 |
| 8  $                                                  | 1999 |       9.2 |
| Abang                                                 | 1981 |       9.3 |
| Abduction of Figaro                                   | 1984 |       9.1 |
| Abdulladzhan, ili posvyashchayetsya Stivenu Spilbergu | 1991 |       9.1 |
| Able's House Is Green, The                            | 2003 |       9.4 |
| Abrahams Gold                                         | 1990 |       9.3 |
| Accordon                                              | 2004 |       9.7 |
| Aclik                                                 | 1974 |       9.4 |
| Actiunea Autobuzul                                    | 1978 |       9.4 |
| Adamah                                                | 1947 |       9.4 |
| Adaptatziya                                           | 1981 |       9.7 |
+-------------------------------------------------------+------+-----------+
20 rows in set (0.01 sec)
```

```
mysql> select name, year, rankscore from movies where rankscore > 9 order by rankscore desc limit 20;
+----------------------------------------+------+-----------+
| name                                   | year | rankscore |
+----------------------------------------+------+-----------+
| Huttyn                                 | 1996 |       9.9 |
| New Clear Farm                         | 1998 |       9.9 |
| Duminica la ora 6                      | 1965 |       9.9 |
| Blow Job                               | 2002 |       9.9 |
| New World, The                         | 1982 |       9.9 |
| Marche des femmes  Hendaye, La         | 1975 |       9.9 |
| Distinto amanecer                      | 1943 |       9.9 |
| Dawn of the Friend                     | 2004 |       9.9 |
| Himala                                 | 1982 |       9.9 |
| Atunci i-am condamnat pe toti la moarte | 1971 |       9.9 |
| Complex Sessions, The                  | 1994 |       9.9 |
| Clearing, The                          | 2001 |       9.9 |
| Duck Soup                              | 1942 |       9.9 |
| Gong fu qi jie                         | 1979 |       9.9 |
| Dosti                                  | 1964 |       9.9 |
| Devil's Circus, The                    | 1926 |       9.9 |
| Jnos vitz                              | 1973 |       9.9 |
| Genet parle d'Angela Davis             | 1970 |       9.9 |
| Napolon Bonaparte                      | 1934 |       9.9 |
| Ivan Groznyj III                       | 1988 |       9.9 |
+----------------------------------------+------+-----------+
20 rows in set (0.19 sec)
```

Examples with operators:

Comparison Operators:

```
mysql> select * from movies_genres where genre = 'comedy' limit 20;
+----------+--------+
| movie_id | genre  |
+----------+--------+
|        2 | Comedy |
|        6 | Comedy |
|        8 | Comedy |
|       11 | Comedy |
|       15 | Comedy |
|       18 | Comedy |
|       21 | Comedy |
|       28 | Comedy |
|       29 | Comedy |
|       35 | Comedy |
|       47 | Comedy |
|       50 | Comedy |
|       52 | Comedy |
|       53 | Comedy |
|       59 | Comedy |
|       73 | Comedy |
|       74 | Comedy |
|       76 | Comedy |
|       94 | Comedy |
|      100 | Comedy |
+----------+--------+
20 rows in set (0.00 sec)
```

```
mysql> select * from movies_genres where genre <> 'horror' limit 20;
+----------+-------------+
| movie_id | genre       |
+----------+-------------+
|        1 | Documentary |
|        1 | Short       |
|        2 | Comedy      |
|        2 | Crime       |
|        5 | Western     |
|        6 | Comedy      |
|        6 | Family      |
|        8 | Animation   |
|        8 | Comedy      |
|        8 | Short       |
|        9 | Drama       |
|       10 | Family      |
|       11 | Comedy      |
|       12 | Crime       |
|       12 | Drama       |
|       12 | Short       |
|       13 | Animation   |
|       13 | Short       |
|       14 | Drama       |
|       14 | Romance     |
+----------+-------------+
20 rows in set (0.00 sec)
```

NULL value never work with “=” operator

You can simply write “IS NULL” or “IS NOT NULL”

```
mysql> select name, year, rankscore from movies where rankscore is null limit 20;
+-----------------------------------+------+-----------+
| name                              | year | rankscore |
+-----------------------------------+------+-----------+
| #28                               | 2002 |      NULL |
| #7 Train: An Immigrant Journey, The | 2000 |      NULL |
| $1,000 Reward                     | 1913 |      NULL |
| $1,000 Reward                     | 1915 |      NULL |
| $1,000 Reward                     | 1923 |      NULL |
| $1,000,000 Reward, The            | 1920 |      NULL |
| $10,000 Under a Pillow            | 1921 |      NULL |
| $100,000                          | 1915 |      NULL |
| $100,000 Pyramid, The             | 2001 |      NULL |
| $20,000 Carat, The                | 1913 |      NULL |
| $21 a Day Once a Month            | 1941 |      NULL |
| $2500 Bride, The                  | 1912 |      NULL |
| $30,000                           | 1920 |      NULL |
| $300 y tickets                    | 2002 |      NULL |
| $5,000 Reward                     | 1918 |      NULL |
| $5,000,000 Counterfeiting Plot, The | 1914 |      NULL |
| $5.15/Hr.                         | 2004 |      NULL |
| $5.20 an Hour Dream, The          | 1980 |      NULL |
| $50,000 Challenge, The            | 1989 |      NULL |
| $50,000 Jewel Theft, The          | 1915 |      NULL |
+-----------------------------------+------+-----------+
20 rows in set (0.00 sec)
```

```
mysql> select name, year, rankscore from movies where rankscore is not null limit 20;
+---------------------------+------+-----------+
| name                      | year | rankscore |
+---------------------------+------+-----------+
| $                         | 1971 |       6.4 |
| $1,000,000 Duck           | 1971 |         5 |
| $1000 a Touchdown         | 1939 |       6.7 |
| $30                       | 1999 |       7.5 |
| $40,000                   | 1996 |       9.6 |
| $50,000 Climax Show, The  | 1975 |       2.6 |
| $pent                     | 2000 |       4.3 |
| $windle                   | 2002 |       5.4 |
| '15'                      | 2002 |       6.8 |
| '38                       | 1987 |       6.7 |
| '49-'17                   | 1917 |       5.8 |
| '68                       | 1988 |       5.5 |
| '94 du bi dao zhi qing    | 1994 |       5.7 |
| '?' Motorist, The         | 1906 |       6.8 |
| 'A'                       | 1965 |       7.1 |
| 'A' gai waak              | 1983 |       7.2 |
| 'A' gai waak juk jaap     | 1987 |       7.2 |
| 'Babicky dobjejte presne!' | 1983 |       5.6 |
| 'Breaker' Morant          | 1980 |       7.9 |
| 'Broadway Melody of 1940' | 1940 |       7.1 |
+---------------------------+------+-----------+
20 rows in set (0.00 sec)
```

Logical Operators:

In SQL, **logical operators** are used to combine multiple conditions in a WHERE clause. These operators return a Boolean value (TRUE, FALSE, or UNKNOWN) and are essential for filtering data based on multiple criteria.

Here are the main logical operators in SQL:

1. AND

- Combines two or more conditions.
- Returns TRUE only if **all** conditions are true.

```
mysql> select name, year, rankscore from movies where rankscore>9 AND year>2000 limit 20;
+---------------------------+------+-----------+
| name                      | year | rankscore |
+---------------------------+------+-----------+
| 12 (2003/II)              | 2003 |       9.8 |
| 14 Million Dreams         | 2003 |       9.5 |
| 2+1                       | 2004 |       9.2 |
| 2wks, 1yr                 | 2002 |       9.4 |
| 5 Card Stud               | 2002 |       9.4 |
| Able's House Is Green, The | 2003 |       9.4 |
| Accordon                  | 2004 |       9.7 |
| Ai-Fak                    | 2004 |       9.4 |
| American Beer             | 2004 |       9.1 |
| American Exile            | 2001 |       9.1 |
| American Reunion, An      | 2003 |       9.7 |
| And So It Goes...         | 2002 |       9.1 |
| Aparte                    | 2002 |       9.1 |
| Aquarium, L'              | 2001 |       9.5 |
| Arregui, la noticia del da | 2001 |       9.5 |
| Aura                      | 2002 |       9.8 |
| Autorequiem               | 2002 |       9.2 |
| Autumn Heart              | 2003 |       9.2 |
| Baan sau chuk dak hin dui | 2002 |       9.5 |
| Barrio Murders, The       | 2001 |       9.2 |
+---------------------------+------+-----------+
20 rows in set (0.01 sec)
```

2. OR

- Combines two or more conditions.
- Returns TRUE if **any** one of the conditions is true.

```
mysql> select name, year, rankscore from movies where rankscore>9 OR year>2007 limit 20;
+-----------------------------------------------------+------+-----------+
| name                                                | year | rankscore |
+-----------------------------------------------------+------+-----------+
| $40,000                                             | 1996 |       9.6 |
| +1 -1                                               | 1987 |       9.6 |
| 12 (2003/II)                                        | 2003 |       9.8 |
| 12 stulyev                                          | 1971 |       9.3 |
| 14 Million Dreams                                   | 2003 |       9.5 |
| 2+1                                                 | 2004 |       9.2 |
| 2wks, 1yr                                           | 2002 |       9.4 |
| 36K                                                 | 2000 |       9.5 |
| 5 Card Stud                                         | 2002 |       9.4 |
| 8  $                                                | 1999 |       9.2 |
| Abang                                               | 1981 |       9.3 |
| Abduction of Figaro                                 | 1984 |       9.1 |
| Abdulladzhan, ili posvyashchayetsya Stivenu Spilbergu | 1991 |       9.1 |
| Able's House Is Green, The                          | 2003 |       9.4 |
| Abrahams Gold                                       | 1990 |       9.3 |
| Accordon                                            | 2004 |       9.7 |
| Aclik                                               | 1974 |       9.4 |
| Actiunea Autobuzul                                  | 1978 |       9.4 |
| Adamah                                              | 1947 |       9.4 |
| Adaptatziya                                         | 1981 |       9.7 |
+-----------------------------------------------------+------+-----------+
20 rows in set (0.00 sec)
```

3. NOT

- Reverses the result of a condition.
- Returns TRUE if the condition is false.

```
mysql> select name, year, rankscore from movies where not year<=2000 limit 20;
+-----------------------------------------------------------------------+------+-----------+
| name                                                                  | year | rankscore |
+-----------------------------------------------------------------------+------+-----------+
| #28                                                                   | 2002 |      NULL |
| $100,000 Pyramid, The                                                 | 2001 |      NULL |
| $300 y tickets                                                        | 2002 |      NULL |
| $5.15/Hr.                                                             | 2004 |      NULL |
| $windle                                                               | 2002 |       5.4 |
| '15'                                                                  | 2002 |       6.8 |
| '60s Pop Rock Reunion                                                 | 2004 |      NULL |
| '88 Dodge Aries                                                       | 2002 |      NULL |
| 'Ang Galing galing mo, Babes'                                         | 2002 |      NULL |
| 'As se hizo' - Torremolinos 73                                        | 2003 |      NULL |
| 'Billy Elliot' Boy, The                                               | 2001 |      NULL |
| 'Catch Me If You Can': Behind the Camera                              | 2003 |      NULL |
| 'Elf' Jukebox                                                         | 2004 |      NULL |
| 'Halloween 4' Final Cut                                               | 2001 |      NULL |
| 'Hellboy': The Seeds of Creation                                      | 2004 |      NULL |
| 'Metal Gear Solid 2: Sons of Liberty' - Making of the Hollywood Game  | 2002 |      NULL |
| 'Moonlight Mile': A Journey to Screen                                 | 2003 |      NULL |
| 'n stukje humor                                                       | 2002 |      NULL |
| 'N Sync: PopOdyssey Live                                              | 2002 |      NULL |
| 'N Sync: The Atlantis Concert                                         | 2001 |      NULL |
+-----------------------------------------------------------------------+------+-----------+
```

4. BETWEEN

- Checks if a value is within a range (inclusive).
- Works with numbers, dates, and text.

```
mysql> select name, year, rankscore from movies where year between 1999 AND 2000 limit 20;
+-------------------------------------------------------------+------+-----------+
| name                                                        | year | rankscore |
+-------------------------------------------------------------+------+-----------+
| #7 Train: An Immigrant Journey, The                         | 2000 |      NULL |
| $30                                                         | 1999 |       7.5 |
| $pent                                                       | 2000 |       4.3 |
| & frres                                                     | 2000 |      NULL |
| '60s, The                                                   | 1999 |      NULL |
| '70s: The Decade That Changed Television, The               | 2000 |      NULL |
| 'Bats' Abound                                               | 1999 |      NULL |
| 'Betty Bee' (sopravvivere d'arte)                           | 1999 |      NULL |
| 'Halloween' Unmasked 2000                                   | 1999 |      NULL |
| 'N Sync & Britney Spears: Your #1 Video Requests... And More! | 2000 |      NULL |
| 'N Sync: 'Ntimate Holiday Special                           | 2000 |      NULL |
| 'N Sync: Live From Madison Square Garden                    | 2000 |      NULL |
| 'N Sync: Making the Tour                                    | 2000 |      NULL |
| 'N Sync: No Strings Attached                                | 1999 |      NULL |
| 'Ne gnstige Gelegenheit                                     | 1999 |       4.6 |
| 'On the Inside: Catching Bank Robbers'                      | 2000 |      NULL |
| 'Romeo Must Die' Movie Special                              | 2000 |      NULL |
| 'Weird Al' Yankovic Live!                                   | 1999 |      NULL |
| 'When It Rains'                                             | 2000 |      NULL |
| (Brief Inquiry Into)                                        | 2000 |      NULL |
+-------------------------------------------------------------+------+-----------+
20 rows in set (0.00 sec)
```

5. IN

- Checks if a value matches any in a list.
- Alternative to multiple OR conditions.

```
mysql> select director_id, genre from directors_genres where genre IN ('Comedy','Horror') limit 20;
+-------------+--------+
| director_id | genre  |
+-------------+--------+
|           8 | Comedy |
|          10 | Comedy |
|          12 | Comedy |
|          18 | Comedy |
|          22 | Comedy |
|          23 | Horror |
|          31 | Comedy |
|          37 | Comedy |
|          41 | Comedy |
|          42 | Comedy |
|          51 | Comedy |
|          52 | Comedy |
|          53 | Comedy |
|          55 | Comedy |
|          67 | Comedy |
|          68 | Comedy |
|          71 | Horror |
|          74 | Comedy |
|          75 | Comedy |
|          80 | Comedy |
+-------------+--------+
20 rows in set (0.04 sec)
```

6. LIKE

- **Searches for a pattern in text (case-sensitive in some databases).**
- Uses wildcards:
    - % = any sequence of characters.
    - _ = single character.

```
mysql> select name, year, rankscore from movies where name LIKE 'Tis%' limit 20;
+-----------------------------------------------------------+------+-----------+
| name                                                      | year | rankscore |
+-----------------------------------------------------------+------+-----------+
| Tis a Gift to Be Simple                                   | 1994 |       8.2 |
| Tis an Ill Wind That Blows No Good                        | 1909 |      NULL |
| Tis an Till Wind That Blows No Good                       | 1912 |      NULL |
| Tis kakomoiras                                            | 1963 |       9.5 |
| Tis mias drakmis ta giasemia                              | 1960 |      NULL |
| Tis moiras t' apopaidi                                    | 1925 |      NULL |
| Tis nyhtas ta kamomata                                    | 1957 |      NULL |
| Tis the Season                                            | 1994 |      NULL |
| Tis the Season                                            | 1998 |         9 |
| Tis tyhis ta grammena                                     | 1957 |      NULL |
| Tis zilias ta kamomata                                    | 1971 |      NULL |
| Tis' the Season to be Jolly                               | 1993 |      NULL |
| Tisane des sarments, La                                   | 1980 |      NULL |
| Tisc za jednu noc                                         | 1932 |      NULL |
| Tisch No. 6                                               | 1998 |      NULL |
| Tisch, Der                                                | 1981 |      NULL |
| Tisch, Der                                                | 1998 |      NULL |
| Tischlein deck Dich, Esel streck Dich, Knppel aus dem Sack | 1938 |      NULL |
| Tischlein, deck dich (1956/I)                             | 1956 |         6 |
| Tischlein, deck dich (1956/II)                            | 1956 |      NULL |
+-----------------------------------------------------------+------+-----------+
20 rows in set (0.01 sec)
```

When we want a name from first characters.

```
mysql> select first_name, last_name from actors where first_name LIKE '%es' limit 20;
+--------------+------------+
| first_name   | last_name  |
+--------------+------------+
| James        | 52X        |
| Tørres       | Aadland    |
| Charles      | Aaron      |
| Reyes        | Abades     |
| Jean-Jacques | Abadie     |
| James        | Abbott     |
| Georges      | Abe        |
| Jacques      | Abellira   |
| Artashes     | Abelyan    |
| Hovhannes    | Abelyan    |
| Georges      | Aber       |
| Andrés       | Aberasturi |
| Georges      | Abiad      |
| Charles      | Able       |
| Myles        | Abney      |
| Charles      | Abou       |
| Charles      | Abraham    |
| James        | Abucewicz  |
| Jacques      | Achilles   |
| Johannes     | Achtelik   |
+--------------+------------+
20 rows in set (0.01 sec)
```

If we want a name from the last characters.

```
mysql> select first_name, last_name from actors where first_name LIKE '%es%' limit 20;
+--------------+--------------+
| first_name   | last_name    |
+--------------+--------------+
| Néstor       | 'Kick Boxer' |
| James        | 52X          |
| Tørres       | Aadland      |
| Jesper       | Aagaard      |
| Vesa         | Aaltonen     |
| Charles      | Aaron        |
| James (I)    | Aaron        |
| James (II)   | Aaron        |
| César        | Abades       |
| Reyes        | Abades       |
| Jean-Jacques | Abadie       |
| Boleslaw     | Abart        |
| César        | Abarzua      |
| Charles S.   | Abbe         |
| James E.     | Abbe         |
| Ernest       | Abbeyquaye   |
| Charles (II) | Abbott       |
| James        | Abbott       |
| Jesse        | Abbott       |
| Georges      | Abe          |
+--------------+--------------+
20 rows in set (0.00 sec)
```

If we want a name from middle, last, first and so on.

```
mysql> select first_name, last_name from actors where first_name LIKE 'Agn_s' limit 20;
+------------+-------------+
| first_name | last_name   |
+------------+-------------+
| Agnès      | Bouloche    |
| Agnes      | Wilke       |
| Agnes      | Adams       |
| Agnes      | Aker        |
| Agnès      | Akopian     |
| Agnès      | Alberny     |
| Ágnes      | Almády      |
| Agnès      | Andersen    |
| Agnes      | Anderson    |
| Agnès      | Aranis      |
| Agnès      | Arlet       |
| Agnes      | Aurelio     |
| Agnes      | Ayres       |
| Agnès      | B.          |
| Agnes      | Babette     |
| Agnes      | Baltsa      |
| Ágnes      | Balázs      |
| Agnes      | Bartholomew |
| Agnes      | Becsei      |
| Agnes      | Bernelle    |
+------------+-------------+
20 rows in set (0.03 sec)
```

“ _” implies only atmost one character.

```
mysql> select first_name, last_name from actors where first_name LIKE 'L%' AND first_name NOT LIKE 'Li%' limit 30;
+------------+----------------+
| first_name | last_name      |
+------------+----------------+
| L          | Lover          |
| L'abbé     | Bernès         |
| L'Abbé     | Simon          |
| L'Ami      | Francis        |
| L'Carole   | Caffery        |
| L'nelle    | Hamanaka       |
| L'Oreal    | Bibbs          |
| L'Ronald   | Smith          |
| L'Tanya    | Van Hamersveld |
| L.         | Alaverdyan     |
| L.         | Aleksandrov    |
| L.         | Alekseev       |
| L.         | Alibert        |
| L.         | Alvaryan       |
| L.         | Antonovich     |
| L.         | Arakelyan      |
| L.         | Areshidze      |
| L.         | Arzhanov       |
| L.         | As             |
| L.         | Aspanidze      |
| L.         | Avetisyan      |
| L.         | Azarashvili    |
| L.         | Babenko        |
| L.         | Badalyan       |
| L.         | Badko          |
| L.         | Bajtalsky      |
| L.         | Baranchik      |
| L.         | Baratashvili   |
| L.         | Bartosik       |
| L.         | Batikyan       |
+------------+----------------+
30 rows in set (0.00 sec)
```

Aggregate Functions: COUNT, MIN, MAX, AVG, SUM

SQL aggregate functions are special functions that perform a calculation on a set of values and return a single value. These functions are often used with the GROUP BY clause but can also be used without it.

Here are the most common SQL aggregate functions:

1. MIN()

- **Purpose**: Finds the smallest value in a column.

Example:

```
mysql> select MIN(year) from movies;
+-----------+
| MIN(year) |
+-----------+
|      1888 |
+-----------+
1 row in set (0.17 sec)
```

2. MAX()

- **Purpose**: Finds the largest value in a column.

Example:

```
mysql> select MAX(year) from movies;
+-----------+
| MAX(year) |
+-----------+
|      2008 |
+-----------+
1 row in set (0.12 sec)
```

3. **COUNT()**

- **Purpose**: Counts the number of rows.

Example:

```
mysql> select count(*) from movies;
+----------+
| count(*) |
+----------+
|   388269 |
+----------+
1 row in set (1.05 sec)
```

```
mysql> select COUNT(year) from movies;
+-------------+
| COUNT(year) |
+-------------+
|      388269 |
+-------------+
1 row in set (0.12 sec)
```

```
mysql> select COUNT(*) from movies where year>2000;
+----------+
| COUNT(*) |
+----------+
|    46006 |
+----------+
1 row in set (0.14 sec)
```

4. **SUM()**

- **Purpose**: Adds up all the values in a numeric column.

Example:

```
mysql> select SUM(year) from movies;
+------------+
| SUM(year) |
+------------+
| 764807736 |
+------------+
1 row in set (0.20 sec)
```

5. **AVG()**

- **Purpose**: Calculates the average (mean) of values in a numeric column.

Example:

```
mysql> select AVG(year) from movies;
+------------+
| AVG(year) |
+------------+
| 1969.7883 |
+------------+
1 row in set (0.16 sec)
```