Meta Description: Understand the potential of SQL's LIKE operator with a comprehensive guide. Learn syntax, wildcards, multiple value retrieval, and more.

SQL Like

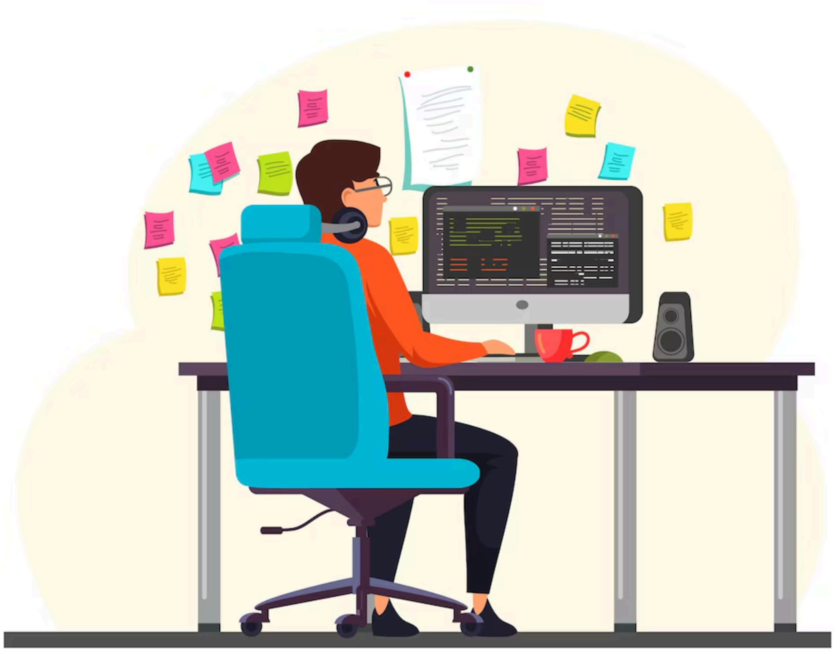


Image Source

Alt Text: A young man coding SQL on his PC

Introduction

Databases are a useful tool. SQL or Structured Query Language is the tool by which these databases can be manipulated or accessed. Personally, I learnt the MySQL database while in 10th grade. Though I was a newbie to programming, SQL syntax was easier than the others. That is the reason I gained interest in this language. One of the most important tasks that can be done using SQL is retrieving data. This brings us to the **SQL LIKE** operator.

The **SQL LIKE** operator is a tool to retrieve data from a table column in any required pattern. It is used with the WHERE clause and in the statements UPDATE, SELECT, and DELETE. One simple example of its use would be searching names starting with S (say Sarah).

A good hold on this operator will make it easier to scan databases for specific data you want to retrieve. This task, which can be daunting if done manually, in turn can be done in a few seconds.

Syntax

Without proper syntax the LIKE operator won’t work, given below is the proper syntax to use the **SQL LIKE** operator:

```
SELECT coloumn_name1, column_name2, …
FROM table_name
WHERE coloumn_name1 LIKE required_pattern;
```

This code will retrieve data that follows the “required_pattern” from the given “coloumn_name”. Another set of code that looks the same with a minor difference is using “=” in place of “LIKE”. Let us spot their differences -

Equals(=)	LIKE

When used, the equal operator compares the whole string with the strings in columns.	On the other hand, LIKE uses wildcards to compare strings.

It is safe to say wildcards are important in understanding how the **SQL LIKE** operator works. We will now learn more about wildcards.

Wildcards in SQL Like operator

Wildcards in **SQL LIKE** operators are special characters that can be used in single or combinations to search for data in the specified pattern. Examples of wildcards are % (per cent sign) and _ (underscore).

1. % (per cent sign)

The % character acts as a substitute to find multiple characters that contain a common set of characters. The position of the % character depends on the location of the common set in those characters.

For example: Say we have this database named “Salary_List” with the given column names-

SLNo.	Name	Salary(Rs.)
1.	Animesh	20000
2.	Rahul	50000
3.	Ayush	33000
4.	Palash	35000
5.	Shantanu	32000

In order to retrieve data of people with salaries in range of 30-40K, we can use the % operator in the following way:

SELECT SLNO., Name, Salary(Rs.) * FROM Salary_List WHERE Salary(Rs.) LIKE ‘3%’ ;

Result:

SLNo.	Name	Salary(Rs.)
3.	Ayush	33000
4.	Palash	35000
5.	Shantanu	32000

Now, in this dataset, we figure out how to match strings starting with specific characters using % Say, we have the task to retrieve data about workers whose name starts with ‘A’.

SELECT SLNO., Name, Salary(Rs.) * FROM Salary_List WHERE Salary(Rs.) LIKE ‘A%’ ;

Result:

SLNo.	Name	Salary(Rs.)
1.	Animesh	20000
3.	Ayush	33000

Thus, using the % wildcard at the end of a string of characters searches for characters in the dataset that start with the string.

Next, from the same dataset, we find a string ending with specific characters using %. Say, we need to search for workers whose names end with ‘sh’.

SELECT * FROM Salary_List WHERE Salary(Rs.) LIKE ‘%sh’ ;

Result:

SLNo.	Name	Salary(Rs.)
1.	Animesh	20000
3.	Ayush	33000
4.	Palash	35000
5.	Shantanu	32000

2. _ (underscore)

The underscore character is a bit more dynamic than %. It can be used to find one, or multiple strings of characters in any arrangement in the dataset. The no. of _ character used can also specify the length of the string of characters to find in the dataset.

Say we have this database named “People”:

Name	City	Age
John	Orlando	33
David	Miami	30
Dave	Texas	42
Devon	California	45
Peter	Washington	26

Now, to sort out details of people whose age is in their 40s, we can use the _ character in the following way:

```
SELECT Name, City, Age FROM People WHERE Age LIKE ‘4_’ ;
```

Result:

Name	City	Age
Dave	Texas	42
Devon	California	45

In a similar manner, we can select names starting with the character ‘D’ and also specify the no. of letters to look out for after ‘D’

```
SELECT Name, City, Age FROM People WHERE Name LIKE ‘D____’ ;
```

Result:

Name	City	Age
David	Miami	30
Devon	California	45

The thing to keep in mind is that there was another Name starting with D which was “Dave”, but it didn’t show. This was because of using four no. of _ (underscores) after D, which instructs to search for Names with just only 4 characters after D.

SQL Like and NOT Like operators

SQL LIKE operator just like the name suggests, searches for data that contains a specified pattern. It uses commonness to its advantage. It works along with the WHERE clause and in statements such as SELECT, DELETE, and UPDATE.

SQL NOT LIKE operator as its name suggests, does the opposite of the LIKE operator. It retrieves all results except the ones having the specified pattern. The **SQL not like** operator works along the WHERE clause and also in statements like SELECT, DELETE, and UPDATE.

An example to understand the difference between LIKE and NOT LIKE operator:

Say we have the following table or database named “Places”:

Landmark	Country	Fee(in $)
Taj Mahal	India	10
Eiffel Tower	France	100
Statue of Liberty	USA	50
Disney Land	USA	70
Great Wall of China	China	50

Example of LIKE operator:

SELECT Landmark FROM Places WHERE Fee(in $) LIKE ‘10%’ ;

Result:

Landmark
Taj Mahal
Eiffel Tower

Due to the use of **SQL LIKE** operator, Landmarks with entry fees starting with 10 are displayed.

Example of NOT LIKE operator:

SELECT Landmark, Country FROM Places WHERE Country NOT LIKE ‘USA’ ;

Result:

Landmark	Country
Taj Mahal	India
Eiffel Tower	France
Great Wall of China	China

On the use of NOT LIKE for Country named USA, the requested data from all other countries except USA was displayed. It ignores the pattern that is specified. This is how the NOT LIKE operator behaves.

SQL Like with multiple values

SQL LIKE is an important operator but till now we have only learnt how to retrieve data with a single string pattern value. Now, let us see how this operator can work for multiple values. This is achieved by the use of AND or OR operator. Below is an example of using **sql like multiple values**.

We have a database named “Birthdays” -

Name	BirthdayMonth	Country
Mary	December	USA
Martha	January	Spain
Trevor	June	USA
Jay	August	India

In this database, let us try and apply **SQL LIKE** for multiple values.

```
SELECT Name, Country FROM Birthdays WHERE Name LIKE ‘M%’ AND Country LIKE ‘USA’ ;
```

Result:

Name	Country
Mary	USA

We have the required data from the database in which LIKE operator is used for names starting with M and Country “USA”. This is an example how the LIKE operator for multiple values works.

LIKE Operator with OR

The **SQL LIKE** operator can work with multiple string values by using the OR operator. The OR operator provides independent string values unlike in AND operator.

The syntax for using OR with the LIKE operator is given:

```
SELECT coloumn_name1, column_name2, …
From Table_name
WHERE coloumn_name1 LIKE pattern_1 OR coloumn_name2 LIKE pattern_2 OR … ;
```

Now for an example, we will use a table named “Sport” -

Sports	Sportsman	Country
Cricket	Virat Kohli	India
Soccer	Lionel Messi	Argentina
Golf	Tyler Woods	USA
Tennis	Roger Federer	Switzerland

From this database, we will now retrieve data using OR operator.

```
SELECT Sports, Sportsman, Country FROM Sports LIKE ‘Cricket’ OR Country LIKE ‘USA’ ;
```

Result:

Sports	Sportsman	Country
Cricket	Virat Kohli	India
Golf	Tyler Woods	USA

In the code, two strings were specified, Cricket from the sports column and USA from the Country column. Thus, using of OR makes sure that data satisfying any of the two strings are retrieved from the database.

Takeaways

SQL is a revolutionary language for maintaining databases and retrieving data from such databases is a prime feature. Here is a sum up of what we learnt about the **SQL LIKE** operator.

- LIKE is an important operator as it helps in most of retrieving operations of Data from databases.
- In **sql like multiple values** of strings can be specified and retrieved in any pattern.
- It uses Wildcards for the input of specific patterns.
- It is used along with the WHERE clause and in statements such as UPDATE, DELETE, and SELECT.
- The LIKE operator works better with other operators like OR and AND.
- One of its variations is the NOT LIKE operator, which works oppositely as the LIKE operator.
- The NOT LIKE operator displays data from all columns except the ones that contain the specified string pattern.

The **SQL LIKE** operator is used with simple queries such as WHERE, SELECT, UPDATE, FROM, and DELETE. It is the add-on on these queries as it makes retrieving easy and better. Without LIKE and NOT LIKE operators, filtering of data would be much harder. So, you must learn to use this operator properly to get better at SQL.

SQL makes managing and accessing of databases easier, by reducing time-consuming tasks to a few prompts on the keyboard. **SQL LIKE** operator is such one prompt. Its prime use in retrieving of data is enhanced by the uses of wildcards and other operators like OR, and AND. It provides accurate and fast filtering of data which makes it a significant SQL operator.

FAQs

1. **Is SQL like case sensitive?**

Ans: When you look for a case-sensitive dataset, SQL LIKE operator will be case-sensitive during retrieval. However, you can also append your query with a new collation to select **sql like case sensitive** or insensitive.

2. **What is the syntax of like?**

Ans: The syntax of LIKE is simple, it goes as follows:
SELECT coloumn_name1, column_name2, …
FROM table_name
WHERE coloumn_name1 LIKE required_pattern;

3. **How to use '%' in SQL?**

Ans: The ‘%’ operator is used with the LIKE operator to specify multiple strings of characters. It is used in the way given below:

SELECT column_Name1 * FROM table_name
WHERE column_Name1 LIKE ‘%pattern’ ;

4. **Is SQL like RegEx?**

Ans: SQL uses the % operator for specifying characters. However, RegEx uses regular expressions to compare different values in a column with the help of parameters.

5. **How to use like in SQL examples?**

Ans: **SQL LIKE** operator is used in the above-mentioned “Salary_List” table as:

SELECT SLNO., Name, Salary(Rs.) * FROM Salary_List WHERE Salary(Rs.) LIKE ‘3%’ ;

Which gives the output:

SLNo.	Name	Salary(Rs.)
3.	Ayush	33000
4.	Palash	35000
5.	Shantanu	32000

Ref Links: https://www.w3schools.com/sql/sql_like.asp

https://www.tutorialspoint.com/sql/sql-like-clause.htm

https://www.programiz.com/sql/like-operator

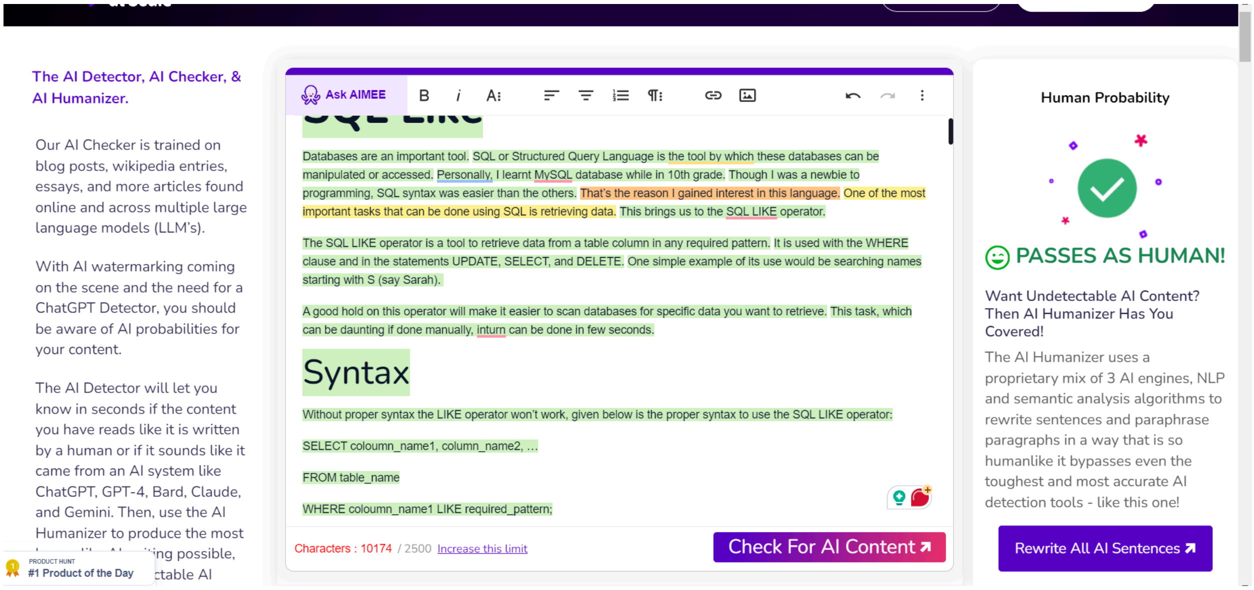
https://www.geeksforgeeks.org/sql-like/

https://www.datacamp.com/tutorial/sql-like-pattern-matching-tutorial

https://learnsql.com/blog/like-sql-not-like/

https://dba.stackexchange.com/questions/209546/how-to-do-a-case-insensitive-like-in-a-case-sensitive-database

https://stackoverflow.com/questions/4078633/difference-between-like-and-regex-operator



The AI Detector, AI Checker, & AI Humanizer.

Our AI Checker is trained on blog posts, wikipedia entries, essays, and more articles found online and across multiple large language models (LLM's).

With AI watermarking coming on the scene and the need for a ChatGPT Detector, you should be aware of AI probabilities for your content.

The AI Detector will let you know in seconds if the content you have reads like it is written by a human or if it sounds like it came from an AI system like ChatGPT, GPT-4, Bard, Claude, and Gemini. Then, use the AI Humanizer to produce the most ing possible, ctable AI

Databases are an important tool. SQL or Structured Query Language is the tool by which these databases can be manipulated or accessed. Personally, I learnt MySQL database while in 10th grade. Though I was a newbie to programming, SQL syntax was easier than the others. That's the reason I gained interest in this language. One of the most important tasks that can be done using SQL is retrieving data. This brings us to the SQL LIKE operator.

The SQL LIKE operator is a tool to retrieve data from a table column in any required pattern. It is used with the WHERE clause and in the statements UPDATE, SELECT, and DELETE. One simple example of its use would be searching names starting with S (say Sarah).

A good hold on this operator will make it easier to scan databases for specific data you want to retrieve. This task, which can be daunting if done manually, inturn can be done in few seconds.

Syntax

Without proper syntax the LIKE operator won't work, given below is the proper syntax to use the SQL LIKE operator:

```
SELECT coloumn_name1, column_name2, …

FROM table_name

WHERE coloumn_name1 LIKE required_pattern;
```

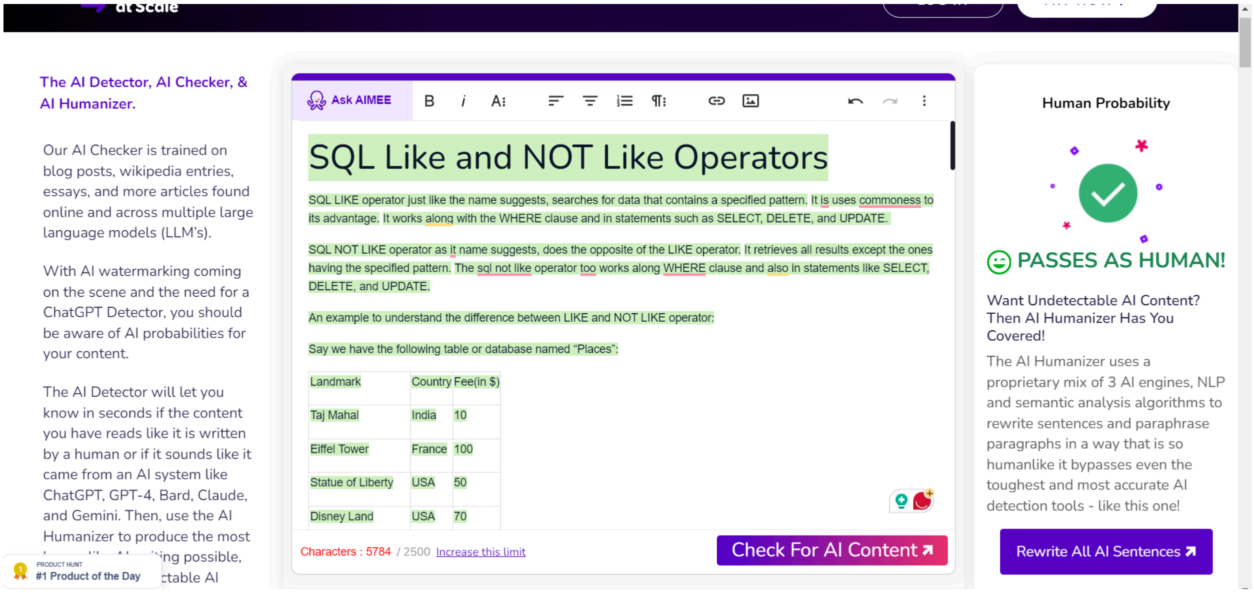
Characters : 10174 / 2500 Increase this limit

Human Probability

PASSES AS HUMAN!

Want Undetectable AI Content? Then AI Humanizer Has You Covered!

The AI Humanizer uses a proprietary mix of 3 AI engines, NLP and semantic analysis algorithms to rewrite sentences and paraphrase paragraphs in a way that is so humanlike it bypasses even the toughest and most accurate AI detection tools - like this one!



The AI Detector, AI Checker, & AI Humanizer.

Our AI Checker is trained on blog posts, wikipedia entries, essays, and more articles found online and across multiple large language models (LLM's).

With AI watermarking coming on the scene and the need for a ChatGPT Detector, you should be aware of AI probabilities for your content.

The AI Detector will let you know in seconds if the content you have reads like it is written by a human or if it sounds like it came from an AI system like ChatGPT, GPT-4, Bard, Claude, and Gemini. Then, use the AI Humanizer to produce the most ing possible, ctable AI

SQL Like and NOT Like Operators

SQL LIKE operator just like the name suggests, searches for data that contains a specified pattern. It is uses commoness to its advantage. It works along with the WHERE clause and in statements such as SELECT, DELETE, and UPDATE.

SQL NOT LIKE operator as it name suggests, does the opposite of the LIKE operator. It retrieves all results except the ones having the specified pattern. The sql not like operator too works along WHERE clause and also in statements like SELECT, DELETE, and UPDATE.

An example to understand the difference between LIKE and NOT LIKE operator:

Say we have the following table or database named "Places":

Landmark	Country	Fee(in $)
Taj Mahal	India	10
Eiffel Tower	France	100
Statue of Liberty	USA	50
Disney Land	USA	70

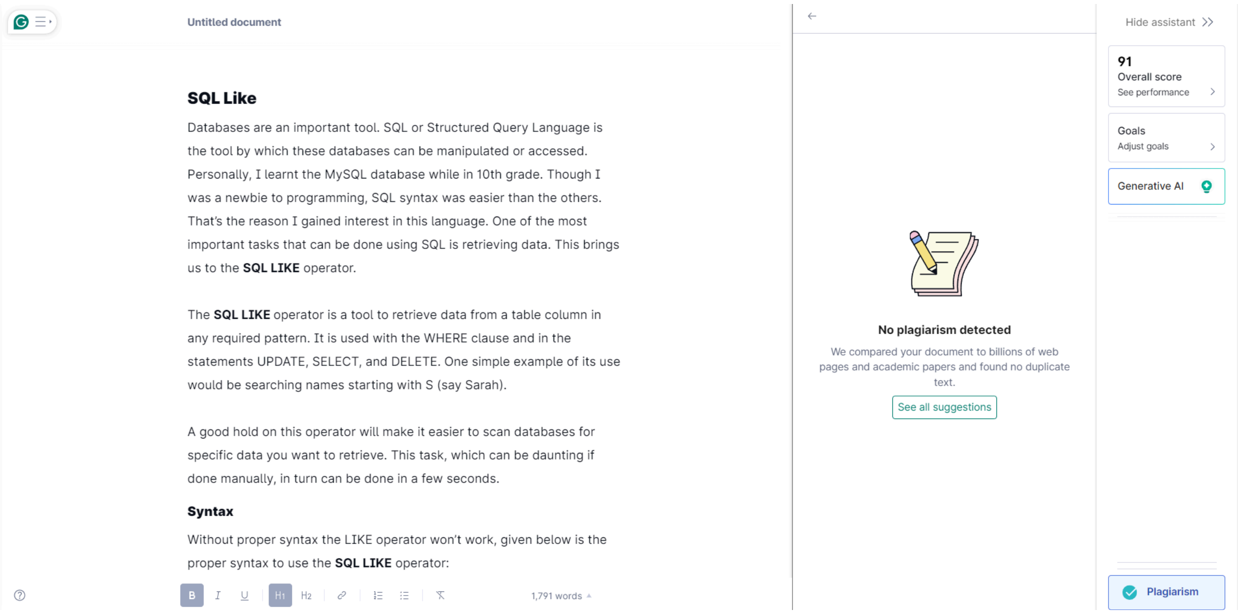
Characters : 5784 / 2500 Increase this limit

Human Probability

PASSES AS HUMAN!

Want Undetectable AI Content? Then AI Humanizer Has You Covered!

The AI Humanizer uses a proprietary mix of 3 AI engines, NLP and semantic analysis algorithms to rewrite sentences and paraphrase paragraphs in a way that is so humanlike it bypasses even the toughest and most accurate AI detection tools - like this one!



SQL Like

Databases are an important tool. SQL or Structured Query Language is the tool by which these databases can be manipulated or accessed. Personally, I learnt the MySQL database while in 10th grade. Though I was a newbie to programming, SQL syntax was easier than the others. That's the reason I gained interest in this language. One of the most important tasks that can be done using SQL is retrieving data. This brings us to the **SQL LIKE** operator.

The **SQL LIKE** operator is a tool to retrieve data from a table column in any required pattern. It is used with the WHERE clause and in the statements UPDATE, SELECT, and DELETE. One simple example of its use would be searching names starting with S (say Sarah).

A good hold on this operator will make it easier to scan databases for specific data you want to retrieve. This task, which can be daunting if done manually, in turn can be done in a few seconds.

Syntax

Without proper syntax the LIKE operator won't work, given below is the proper syntax to use the **SQL LIKE** operator:

No plagiarism detected

We compared your document to billions of web pages and academic papers and found no duplicate text.