# Content from https://www.sqltutorial.org/what-is-sql/

Home » SQL Tutorial » What Is SQLWhat Is SQLSummary: in this tutorial, we will introduce you to the SQL language, discuss standard SQL and some popular SQL dialects.Introduction to SQL languageSQL is a programming language designed to manage data stored in a relational database management system (RDBMS).SQL stands for the structured query language. It is pronounced as /ˈɛs kjuː ˈɛl/ or /ˈsiːkwəl/.SQL consists of a data definition language, data manipulation language, and a data control language.The data definition language deals with the schema creation and modification e.g., CREATE TABLE statement allows you to create a new table in the database and the ALTER TABLE statement changes the structure of an existing table.The data manipulation language provides the constructs to query data such as the SELECT statement and to update the data such as INSERT, UPDATE, and DELETE statements.The data control language consists of the statements that deal with the user authorization and security such as GRANT and REVOKE statements.SQL StandardSQL was one of the first commercial database languages since 1970. Since then different database vendors implemented SQL in their products with some variations. To bring greater conformity between the vendors, the American Standards Institute (ANSI) published the first SQL standard in 1986.ANSI then updated the SQL standard in 1992, known as SQL92 and SQL2, and again in 1999 as SQL99 and SQL3. Every time, ANSI added new features and commands into the SQL language.The SQL Standard is now maintained by both ANSI and International Standards Organization as ISO/IEC 9075 standard. The latest release standard is SQL:2011.The SQL standard formalizes SQL syntax structures and behaviors across database products. It becomes even more important to the open-source databases such as MySQL and PostgreSQL where the RDBMS are developed mainly by the communities rather than big corporations.SQL DialectsThe community constantly requests new features and capabilities that do not exist in the SQL standard yet, therefore, even with the SQL standard in place, there are many SQL dialects in various database products.Because ANSI and ISO have not yet developed these important features, RDBMS vendors (or communities) are free to invent their own new syntax structure.The following are the most popular dialects of SQL:PL/SQL stands for procedural language/SQL. It is developed by Oracle for the Oracle Database.Transact-SQL or T-SQL is developed by Microsoft for Microsoft SQL Server.PL/pgSQL stands for Procedural Language/PostgreSQL that consists of SQL dialect and extensions implemented in PostgreSQLMySQL has its own procedural language since version 5. Note that MySQL was acquired by Oracle.In each tutorial, we will explain the SQL syntax structures and behaviors that are valid across the databases. We also will discuss the exceptions if they exist in a particular database.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL Tutorial Up NextSQL Sample Database

# Content from https://www.sqltutorial.org/sql-syntax/

Home » SQL Tutorial » SQL SyntaxSQL SyntaxSummary: in this tutorial, we will introduce you to the SQL syntax that helps you understand the details of SQL statements.SQL is a declarative language, therefore, its syntax reads like a natural language. An SQL statement begins with a verb that describes the action, for example, SELECT, INSERT, UPDATE or DELETE. Following the verb are the subject and predicate.A predicate specifies conditions that can be evaluated as true, false, or unknown.See the following SQL statement:SELECT   
 first\_name  
FROM  
 employees  
WHERE  
 YEAR(hire\_date) = 2000;Code language: SQL (Structured Query Language) (sql)As you see, it reads like a normal sentence. Get the first names of employees who were hired in 2000.The SELECT first\_name, FROM employees, and WHERE  are clauses in the SQL statement. Some clauses are mandatory e.g., the SELECT and FROM clauses whereas others are optional such as the WHERE clause.Because SQL was designed specifically for the non-technical people in mind, it is very simple and easy to understand. To write an SQL statement, you just need to tell what you want instead of how you want it like other imperative languages such as PHP, Java, and C++.SQL is a user-friendly language because it is mainly for the users who perform ad-hoc queries and generate reports.Nowadays, SQL is used by the highly technical people like data analysts, data scientists, developers, and database administrators.SQL commandsSQL is made up of many commands. Each SQL command is typically terminated with a semicolon (;). For example, the following are two different SQL commands separated by a semicolon (;).SELECT   
 first\_name, last\_name  
FROM  
 employees;  
  
DELETE FROM employees   
WHERE  
 hire\_date < '1990-01-01';Code language: SQL (Structured Query Language) (sql)SQL uses the semicolon (;) to mark the end of a command.Each command is composed of tokens that can be literals, keywords, identifiers, or expressions. Tokens are separated by space, tabs, or newlines.LiteralsLiterals are explicit values which are also known as constants. SQL provides three kinds of literals: string, numeric, and binary.String literal consists of one or more alphanumeric characters surrounded by single quotes, for example:'John'  
'1990-01-01'  
'50'Code language: SQL (Structured Query Language) (sql)50 is a number. However, if you surround it with single quotes e.g., '50', SQL treats it as a string literal.Typically, SQL is case sensitive with respect to string literals, so the value 'John' is not the same as 'JOHN'.Numeric literals are the integer, decimal, or scientific notation, for example:200  
-5  
6.0221415E23Code language: SQL (Structured Query Language) (sql)SQL represents binary value using the notation x'0000', where each digit is hexadecimal value, for example:x'01'  
x'0f0ff'  
Code language: SQL (Structured Query Language) (sql)KeywordsSQL has many keywords that have special meanings such as SELECT, INSERT, UPDATE, DELETE, and DROP. These keywords are the reserved words, therefore, you cannot use them as the name of tables, columns, indexes, views, stored procedures, triggers, or other database objects.IdentifiersIdentifiers refer to specific objects in the database such as tables, columns, indexes, etc. SQL is case-insensitive with respect to keywords and identifiers.The following statements are equivalent.Select  \* From employees;  
  
SELECT \* FROM EMPLOYEES;  
  
select \* from employees;  
  
SELECT \* FROM employees;Code language: SQL (Structured Query Language) (sql)To make the SQL commands more readable and clear, we will use the SQL keywords in uppercase and identifiers in lower case throughout the tutorials.CommentsTo document SQL statements, you use the SQL comments. When parsing SQL statements with comments, the database engine ignores the characters in the comments.A comment is denoted by two consecutive hyphens ( --) that allow you to comment the remaining line.  See the following example.SELECT   
 employee\_id, salary  
FROM  
 employees  
WHERE  
 salary < 3000;-- employees with low salaryCode language: SQL (Structured Query Language) (sql)This is an SQL comment.-- employees with low salaryCode language: SQL (Structured Query Language) (sql)To document the code that can span multiple lines, you use the multiline C-style notation ( /\*\*/) as the shown in the following statement:/\* increase 5% for employees whose salary is less than 3,000 \*/  
UPDATE employees   
SET   
 salary = salary \* 1.05  
WHERE  
 salary < 3000;Code language: SQL (Structured Query Language) (sql)In this tutorial, we have introduced you to the SQL syntax that helps you understand each component of an SQL statement.To learn the SQL language more effectively, you need to have a good sample database to practice with. In the next tutorial, we will introduce you to a simple SQL database.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL Sample Database Up NextSQL SELECT

# Content from https://www.sqltutorial.org/sql-sample-database/

Home » SQL Tutorial » SQL Sample DatabaseSQL Sample DatabaseSummary: in this tutorial, you will learn about a SQL Sample Database called HR that manages the HR data of the small businesses.The following database diagram illustrates the HR sample database:The HR sample database has seven tables:The employees table stores the data of employees.The jobs table stores the job data including job title and salary range.The departments table stores department data.The dependents table stores the employee’s dependents.The locations table stores the location of the departments of the company.The countries table stores the data of countries where the company is doing business.The regions table stores the data of regions such as Asia, Europe, America, and the Middle East and Africa. The countries are grouped into regions.The following picture shows the table names and their records.TableRowsemployees40dependents30departments11jobs11locations7countries25regions4Typically, you need to install a Relational Database Management System (RDBMS) to work with SQL.If you have worked with an RDBMS such as MySQL, PostgreSQL, Oracle Database, and SQL Server, you can use the following script to create the sample database in one of these databases.In case you don’t have a database system to practice, you can quickly use our SQL online tool to execute the SQL statements in your web browser.MySQLThe following SQL script creates the HR sample database in MySQL: Create HR Sample Database in MySQLThe following script allows you to insert data into the tables in MySQL: Load HR Data in MySQLPostgreSQLThe following script creates the HR sample database structure in PostgreSQL. Create HR Sample Database in PostgreSQLThe following script allows you to insert data into the tables in PostgreSQL: Load HR Data in PostgreSQLMicrosoft SQL ServerThe following script creates the HR sample database structure in Microsoft SQL Server. Create HR Sample Database in SQL ServerThe following script allows you to insert data into the tables: Load HR Data in SQL ServerOracle Database (>12c)The following script creates the HR sample database structure in Oracle Database 12c. Create HR Sample Database in OracleThe following script inserts data into the tables in the Oracle database: Load HR Data in Oracle DatabaseSQLiteThe following script creates the HR sample database structure in SQLite. Create HR Sample Database in SQLiteThe following script inserts data into the tables in the SQLite: Create HR Sample Database in SQLiteRemoving tablesThe following is the script that drops all tables in case you want to refresh the sample database. Drop All TablesWas this tutorial helpful ? Yes No Send Cancel PreviouslyWhat Is SQL Up NextSQL Syntax

# Content from https://www.sqltutorial.org/sql-select/

Home » SQL Tutorial » SQL SELECTSQL SELECTSummary: in this tutorial, you will learn how to use the SQL SELECT statement to query data from a single table.Introduction to SQL SELECT statementThe SQL SELECT statement selects data from one or more tables. The following shows the basic syntax of the SELECT statement that selects data from a single table.SELECT   
 select\_list  
FROM  
 table\_name;Code language: SQL (Structured Query Language) (sql)In this syntax:First, specify a list of comma-separated columns from the table in the SELECT clause.Then, specify the table name in the FROM clause.When evaluating the SELECT statement, the database system evaluates the FROM clause first and then the SELECT clause. It’s like from a table, select data from these columns.The semicolon (;) is not part of a query. The database server uses it to separate two SQL statements.For example, if you execute two SQL SELECT statements, you need to separate them using the semicolon (;). Check out the SQL syntax for more information.If you want to query data from all the columns of the table, you can use the asterisk (\*) operator instead if specifying all the column names:SELECT \* FROM table\_name;SQL is case-insensitive. Therefore, the SELECT and select keywords have the same meaning.By convention, we will use the uppercase letters for the SQL keywords, such as SELECT and FROM and the lowercase letters for the identifiers such as table and column names. This convention makes the SQL statements more readable.SQL SELECT statement examplesWe’ll use the employees table in the sample database for demonstration purposes.1) SQL SELECT – selecting data from all columns exampleThe following example uses the SQL SELECT statement to get data from all the rows and columns in the employees table:SELECT \* FROM employees;Code language: SQL (Structured Query Language) (sql)Try ItThe following shows the result sets returned by the database server. It’s like a spreadsheet that contains rows and columns with a heading:+-------------+-------------+-------------+-----------------------------------+--------------+------------+--------+----------+------------+---------------+  
| employee\_id | first\_name | last\_name | email | phone\_number | hire\_date | job\_id | salary | manager\_id | department\_id |  
+-------------+-------------+-------------+-----------------------------------+--------------+------------+--------+----------+------------+---------------+  
| 100 | Steven | King | [email protected] | 515.123.4567 | 1987-06-17 | 4 | 24000.00 | NULL | 9 |  
| 101 | Neena | Kochhar | [email protected] | 515.123.4568 | 1989-09-21 | 5 | 17000.00 | 100 | 9 |  
| 102 | Lex | De Haan | lex.de [email protected] | 515.123.4569 | 1993-01-13 | 5 | 17000.00 | 100 | 9 |  
| 103 | Alexander | Hunold | [email protected] | 590.423.4567 | 1990-01-03 | 9 | 9000.00 | 102 | 6 |  
| 104 | Bruce | Ernst | [email protected] | 590.423.4568 | 1991-05-21 | 9 | 6000.00 | 103 | 6 |  
| 105 | David | Austin | [email protected] | 590.423.4569 | 1997-06-25 | 9 | 4800.00 | 103 | 6 |  
| 106 | Valli | Pataballa | [email protected] | 590.423.4560 | 1998-02-05 | 9 | 4800.00 | 103 | 6 |  
| 107 | Diana | Lorentz | [email protected] | 590.423.5567 | 1999-02-07 | 9 | 4200.00 | 103 | 6 |  
| 108 | Nancy | Greenberg | [email protected] | 515.124.4569 | 1994-08-17 | 7 | 12000.00 | 101 | 10 |  
...Code language: plaintext (plaintext)The SELECT \* is read as the select star. The select star is helpful for ad-hoc queries only.For the application development, you should avoid using the select star for the following reason.The select \* returns data from all columns of a table. Often, the application doesn’t need data from all the columns but one or some columns.If you use the select \*, the database needs more time to read data from the disk and transfer it to the application. This often results in poor performance if the table contains many columns with a lot of data.2) SQL SELECT – selecting data from specific columnsTo select data from specific columns, you can specify the column list after the SELECT clause of the SELECT statement.For example, the following select data from the employee id, first name, last name, and hire date of all rows in the employees table:SELECT   
 employee\_id,   
 first\_name,   
 last\_name,   
 hire\_date  
FROM  
 employees;Code language: SQL (Structured Query Language) (sql)Try ItNow, the result set includes only four columns specified in the SELECT clause:+-------------+-------------+-------------+------------+  
| employee\_id | first\_name | last\_name | hire\_date |  
+-------------+-------------+-------------+------------+  
| 100 | Steven | King | 1987-06-17 |  
| 101 | Neena | Kochhar | 1989-09-21 |  
| 102 | Lex | De Haan | 1993-01-13 |  
| 103 | Alexander | Hunold | 1990-01-03 |  
| 104 | Bruce | Ernst | 1991-05-21 |  
| 105 | David | Austin | 1997-06-25 |  
| 106 | Valli | Pataballa | 1998-02-05 |  
| 107 | Diana | Lorentz | 1999-02-07 |  
| 108 | Nancy | Greenberg | 1994-08-17 |  
| 109 | Daniel | Faviet | 1994-08-16 |  
| 110 | John | Chen | 1997-09-28 |  
...Code language: plaintext (plaintext)3) SQL SELECT – performing a simple calculationThe following example uses the SELECT statement to get the first name, last name, salary, and new salary:SELECT   
 first\_name,   
 last\_name,   
 salary,   
 salary \* 1.05  
FROM  
 employees;Code language: SQL (Structured Query Language) (sql)The expression salary \* 1.05 adds 5% to the salary of every employee. By default, SQL uses the expression as the column heading:+-------------+-------------+----------+---------------+  
| first\_name | last\_name | salary | salary \* 1.05 |  
+-------------+-------------+----------+---------------+  
| Steven | King | 24000.00 | 25200.0000 |  
| Neena | Kochhar | 17000.00 | 17850.0000 |  
| Lex | De Haan | 17000.00 | 17850.0000 |  
| Alexander | Hunold | 9000.00 | 9450.0000 |  
| Bruce | Ernst | 6000.00 | 6300.0000 |  
| David | Austin | 4800.00 | 5040.0000 |  
| Valli | Pataballa | 4800.00 | 5040.0000 |  
| Diana | Lorentz | 4200.00 | 4410.0000 |  
| Nancy | Greenberg | 12000.00 | 12600.0000 |  
...Code language: plaintext (plaintext)To assign an expression or a column an alias, you specify the AS keyword followed by the column alias as follows:expression AS column\_aliasCode language: SQL (Structured Query Language) (sql)For example, the following SELECT statement uses the new\_salary as the column alias for the salary \* 1.05 expression:SELECT   
 first\_name,   
 last\_name,   
 salary,   
 salary \* 1.05 AS new\_salary  
FROM  
 employees;Code language: SQL (Structured Query Language) (sql)Output:+-------------+-------------+----------+------------+  
| first\_name | last\_name | salary | new\_salary |  
+-------------+-------------+----------+------------+  
| Steven | King | 24000.00 | 25200.0000 |  
| Neena | Kochhar | 17000.00 | 17850.0000 |  
| Lex | De Haan | 17000.00 | 17850.0000 |  
| Alexander | Hunold | 9000.00 | 9450.0000 |  
| Bruce | Ernst | 6000.00 | 6300.0000 |  
| David | Austin | 4800.00 | 5040.0000 |  
| Valli | Pataballa | 4800.00 | 5040.0000 |  
| Diana | Lorentz | 4200.00 | 4410.0000 |  
| Nancy | Greenberg | 12000.00 | 12600.0000 |SummaryUse the SQL SELECT statment to select data from a table.To select data from a table, specify the table name in the FROM clause and a list of column in the SELECT clause.The SELECT \* is the shorthand of the SELECT all columns in a table.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL Syntax Up NextSQL ORDER BY

# Content from https://www.sqltutorial.org/sql-order-by/

Home » SQL Tutorial » SQL ORDER BYSQL ORDER BYSummary: This tutorial shows you how to use the SQL ORDER BY clause to sort the result set based on specified criteria in ascending or descending orders.Introduction to SQL ORDER BY clauseThe ORDER BY is an optional clause of the SELECT statement. The ORDER BY clause allows you to sort the rows returned by the SELECT clause by one or more sort expressions in ascending or descending order.The following shows the syntax of the ORDER BY clause:SELECT   
 select\_list  
FROM  
 table\_name  
ORDER BY   
 sort\_expression [ASC | DESC];Code language: SQL (Structured Query Language) (sql)In this syntax:First, place the ORDER BY clause after the FROM clause. The database will evaluate the SELECT statement with the ORDER BY \* clause in the following order: FROM > SELECT > ORDER BY.Second, specify a sort expression after the ORDER BY clause. The sort expression specifies the sort criteria.Third, use ASC option to sort the result set by the sort expression in ascending order and DESC to sort the result set by the sort expression in the descending order.Note that the ORDER BY clause uses the ASC option by default if you don’t either ASC or DESC.The ORDER BY clause also allows you to sort the result set by multiple expressions. In this case, you need to use a comma to separate two sort expressions:SELECT   
 select\_list  
FROM  
 table\_name  
ORDER BY   
 sort\_expression\_1 [ASC | DESC],  
 sort\_expression\_2 [ASC | DESC];Code language: CSS (css)In this syntax, the ORDER BY clause sorts the result set by the sort\_expression\_1 first, and then sorts the sorted result set by the sort\_expression\_2.Note that if you don’t specify the ORDER BY clause, the SELECT statement will not sort the result set. It means that the rows in the result set don’t have a specific order.SQL ORDER BY clause examplesWe’ll use the employees table in the sample database for the demonstration.1) Using SQL ORDER BY clause to sort values in one column exampleThe following SELECT statement returns the data from the employee id, first name, last name, hire date, and salary column of the employees table:SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 hire\_date,  
 salary  
FROM  
 employees;Code language: SQL (Structured Query Language) (sql)Try It+-------------+-------------+-------------+------------+----------+  
| employee\_id | first\_name | last\_name | hire\_date | salary |  
+-------------+-------------+-------------+------------+----------+  
| 100 | Steven | King | 1987-06-17 | 24000.00 |  
| 101 | Neena | Kochhar | 1989-09-21 | 17000.00 |  
| 102 | Lex | De Haan | 1993-01-13 | 17000.00 |  
| 103 | Alexander | Hunold | 1990-01-03 | 9000.00 |  
| 104 | Bruce | Ernst | 1991-05-21 | 6000.00 |  
| 105 | David | Austin | 1997-06-25 | 4800.00 |  
| 106 | Valli | Pataballa | 1998-02-05 | 4800.00 |  
| 107 | Diana | Lorentz | 1999-02-07 | 4200.00 |  
| 108 | Nancy | Greenberg | 1994-08-17 | 12000.00 |  
| 109 | Daniel | Faviet | 1994-08-16 | 9000.00 |  
| 110 | John | Chen | 1997-09-28 | 8200.00 |  
...As you can see clearly from the output, the rows do not have any order.The following example uses the ORDER BY clause to sort employees by first names in alphabetical order:SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 hire\_date,  
 salary  
FROM  
 employees  
ORDER BY  
 first\_name;Code language: SQL (Structured Query Language) (sql)Try It+-------------+-------------+-------------+------------+----------+  
| employee\_id | first\_name | last\_name | hire\_date | salary |  
+-------------+-------------+-------------+------------+----------+  
| 121 | Adam | Fripp | 1997-04-10 | 8200.00 |  
| 115 | Alexander | Khoo | 1995-05-18 | 3100.00 |  
| 103 | Alexander | Hunold | 1990-01-03 | 9000.00 |  
| 193 | Britney | Everett | 1997-03-03 | 3900.00 |  
| 104 | Bruce | Ernst | 1991-05-21 | 6000.00 |  
| 179 | Charles | Johnson | 2000-01-04 | 6200.00 |  
| 109 | Daniel | Faviet | 1994-08-16 | 9000.00 |  
| 105 | David | Austin | 1997-06-25 | 4800.00 |  
| 114 | Den | Raphaely | 1994-12-07 | 11000.00 |  
| 107 | Diana | Lorentz | 1999-02-07 | 4200.00 |  
...The ORDER BY sorts the rows by the values in the first\_name column.2) Using SQL ORDER BY  clause to sort values in multiple columns exampleThe following example uses the ORDER BY clause to sort the employees by the first name in ascending order and the last name in descending order:SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 hire\_date,  
 salary  
FROM  
 employees  
ORDER BY  
 first\_name,  
 last\_name DESC;Code language: SQL (Structured Query Language) (sql)Try It+-------------+-------------+-------------+------------+----------+  
| employee\_id | first\_name | last\_name | hire\_date | salary |  
+-------------+-------------+-------------+------------+----------+  
| 121 | Adam | Fripp | 1997-04-10 | 8200.00 |  
| 115 | Alexander | Khoo | 1995-05-18 | 3100.00 |  
| 103 | Alexander | Hunold | 1990-01-03 | 9000.00 |  
| 193 | Britney | Everett | 1997-03-03 | 3900.00 |  
| 104 | Bruce | Ernst | 1991-05-21 | 6000.00 |  
| 179 | Charles | Johnson | 2000-01-04 | 6200.00 |  
| 109 | Daniel | Faviet | 1994-08-16 | 9000.00 |  
| 105 | David | Austin | 1997-06-25 | 4800.00 |  
| 114 | Den | Raphaely | 1994-12-07 | 11000.00 |  
| 107 | Diana | Lorentz | 1999-02-07 | 4200.00 |  
| 118 | Guy | Himuro | 1998-11-15 | 2600.00 |  
...In this example, the ORDER BY clause sorts the result set by the first name in ascending order, then it sorts the sorted result set by the last name in descending order.Notice the change in position of two employees: Alexander Khoo and Alexander Hunold in the result set.3) Using SQL ORDER BY clause to sort values in a numeric column exampleThe following example uses the ORDER BY clause to sort employees by salary from high to low:SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 hire\_date,  
 salary  
FROM  
 employees  
ORDER BY  
 salary DESC;Code language: SQL (Structured Query Language) (sql)Try It+-------------+-------------+-------------+------------+----------+  
| employee\_id | first\_name | last\_name | hire\_date | salary |  
+-------------+-------------+-------------+------------+----------+  
| 100 | Steven | King | 1987-06-17 | 24000.00 |  
| 101 | Neena | Kochhar | 1989-09-21 | 17000.00 |  
| 102 | Lex | De Haan | 1993-01-13 | 17000.00 |  
| 145 | John | Russell | 1996-10-01 | 14000.00 |  
| 146 | Karen | Partners | 1997-01-05 | 13500.00 |  
| 201 | Michael | Hartstein | 1996-02-17 | 13000.00 |  
| 205 | Shelley | Higgins | 1994-06-07 | 12000.00 |  
| 108 | Nancy | Greenberg | 1994-08-17 | 12000.00 |  
| 114 | Den | Raphaely | 1994-12-07 | 11000.00 |  
...4) Using SQL ORDER BY to sort by dates exampleBesides the character and numeric data, you can use the ORDER BY clause to sort rows by date. For example, the following statement uses the ORDER BY clause to sort the employees by values in the hire\_date column from:SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 hire\_date,  
 salary  
FROM  
 employees  
ORDER BY  
 hire\_date;Code language: SQL (Structured Query Language) (sql)Try It+-------------+-------------+-------------+------------+----------+  
| employee\_id | first\_name | last\_name | hire\_date | salary |  
+-------------+-------------+-------------+------------+----------+  
| 100 | Steven | King | 1987-06-17 | 24000.00 |  
| 200 | Jennifer | Whalen | 1987-09-17 | 4400.00 |  
| 101 | Neena | Kochhar | 1989-09-21 | 17000.00 |  
| 103 | Alexander | Hunold | 1990-01-03 | 9000.00 |  
| 104 | Bruce | Ernst | 1991-05-21 | 6000.00 |  
| 102 | Lex | De Haan | 1993-01-13 | 17000.00 |  
| 203 | Susan | Mavris | 1994-06-07 | 6500.00 |  
| 204 | Hermann | Baer | 1994-06-07 | 10000.00 |  
...To see the employees who joined the company from lastest to earliest, you sort the employees by the hire dates in descending order:SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 hire\_date,  
 salary  
FROM  
 employees  
ORDER BY  
 hire\_date DESC;Code language: SQL (Structured Query Language) (sql)Try It+-------------+-------------+-------------+------------+----------+  
| employee\_id | first\_name | last\_name | hire\_date | salary |  
+-------------+-------------+-------------+------------+----------+  
| 179 | Charles | Johnson | 2000-01-04 | 6200.00 |  
| 113 | Luis | Popp | 1999-12-07 | 6900.00 |  
| 119 | Karen | Colmenares | 1999-08-10 | 2500.00 |  
| 178 | Kimberely | Grant | 1999-05-24 | 7000.00 |  
| 107 | Diana | Lorentz | 1999-02-07 | 4200.00 |  
| 118 | Guy | Himuro | 1998-11-15 | 2600.00 |  
| 126 | Irene | Mikkilineni | 1998-09-28 | 2700.00 |  
| 177 | Jack | Livingston | 1998-04-23 | 8400.00 |  
| 176 | Jonathon | Taylor | 1998-03-24 | 8600.00 |  
...SummaryUse the ORDER BY clause to sort rows returned by the SELECT clause.Use the ASC option to sort rows in ascending order and DESC option to sort rows in descending order.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL SELECT Up NextSQL DISTINCT

# Content from https://www.sqltutorial.org/sql-distinct/

Home » SQL Tutorial » SQL DISTINCTSQL DISTINCTSummary: in this tutorial, you will learn how to use the SQL DISTINCT operator to remove duplicates from a result setIntroduction to SQL DISTINCT operatorTo remove duplicate rows from a result set, you use the DISTINCT operator in the SELECT clause as follows:SELECT DISTINCT  
 column1, column2, ...  
FROM  
 table1;Code language: SQL (Structured Query Language) (sql)If you use one column after the DISTINCT operator, the DISTINCT operator uses values in that column to evaluate duplicates.If you use two or more columns, the DISTINCT will use the combination of values in those columns to evaluate the duplicate.Note that the DISTINCT only removes the duplicate rows from the result set. It doesn’t delete duplicate rows in the table.If you want to select two columns and remove duplicates in one column, you should use the GROUP BY clause instead.SQL DISTINCT examplesWe will use the employees table in the sample database to demonstrate how the DISTINCT operator works.1) Using SQL DISTINCT operator on one column exampleThe following statement selects the salary data from the salary column of the employees table and sorts them from high to low:SELECT   
 salary  
FROM  
 employees  
ORDER BY salary DESC;Code language: SQL (Structured Query Language) (sql)Try It+----------+  
| salary |  
+----------+  
| 24000.00 |  
| 17000.00 |  
| 17000.00 |  
| 14000.00 |  
| 13500.00 |  
| 13000.00 |  
| 12000.00 |  
| 12000.00 |  
| 11000.00 |  
| 10000.00 |  
| 9000.00 |  
| 9000.00 |  
...The result set has some duplicates. For example, 17000, 12000, and 9000.The following statement uses the DISTINCT operator to select unique values from the salary column of the employees table:SELECT   
 DISTINCT salary  
FROM  
 employees  
ORDER BY salary DESC;Code language: SQL (Structured Query Language) (sql)Try It+----------+  
| salary |  
+----------+  
| 24000.00 |  
| 17000.00 |  
| 14000.00 |  
| 13500.00 |  
| 13000.00 |  
| 12000.00 |  
| 11000.00 |  
| 10000.00 |  
| 9000.00 |Code language: plaintext (plaintext)As you can see, the result set doesn’t contain any duplicate salary values.2) Using SQL DISTINCT operator on multiple columns exampleThe following statement selects the job id and salary from the employees table:SELECT  
 job\_id,  
 salary  
FROM  
 employees  
ORDER BY  
 job\_id,  
 salary DESC;Code language: SQL (Structured Query Language) (sql)Try It+--------+----------+  
| job\_id | salary |  
+--------+----------+  
| 1 | 8300.00 |  
| 2 | 12000.00 |  
| 3 | 4400.00 |  
| 4 | 24000.00 |  
| 5 | 17000.00 |  
| 5 | 17000.00 |  
| 6 | 9000.00 |  
| 6 | 8200.00 |  
...Code language: plaintext (plaintext)The result set has some duplicate rows e.g., job id 5 salary 17000. It means that there are two employees with the same job id and salary.The following statement uses the DISTINCT operator to remove the duplicate values in job id and salary:SELECT DISTINCT  
 job\_id,  
 salary  
FROM  
 employees  
ORDER BY  
 job\_id,  
 salary DESC;Code language: SQL (Structured Query Language) (sql)Try It+--------+----------+  
| job\_id | salary |  
+--------+----------+  
| 1 | 8300.00 |  
| 2 | 12000.00 |  
| 3 | 4400.00 |  
| 4 | 24000.00 |  
| 5 | 17000.00 |  
| 6 | 9000.00 |  
| 6 | 8200.00 |  
...Code language: plaintext (plaintext)Note that you still see the duplicate in the job\_id column because the DISTINCT operator uses values from both job\_id and salary to evaluate the duplicate, not just values in the job\_id column.SQL DISTINCT and NULLIn the database, NULL means unknown or missing data.Unlike values like numbers, strings, dates, etc. NULL does not equal anything, even itself. The following expression will return unknown (or NULL):NULL=NULLCode language: PHP (php)Typically, the DISTINCT operator treats all NULL the same. Therefore, the DISTINCT operator keeps only one NULL in the result set.Note that this behavior may be different between database products.For example, the following statement returns the distinct phone numbers of employees:SELECT DISTINCT phone\_number  
FROM employees  
ORDER BY phone\_number;Code language: SQL (Structured Query Language) (sql)Try It+--------------+  
| phone\_number |  
+--------------+  
| NULL |  
| 515.123.4444 |  
| 515.123.4567 |  
| 515.123.4568 |  
| 515.123.4569 |  
| 515.123.5555 |  
...Code language: plaintext (plaintext)Notice that the query returns only one NULL in the result set.SummaryUse DISTINCT operator in the SELECT clause to remove duplicate rows from the result set.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL ORDER BY Up NextSQL LIMIT

# Content from https://www.sqltutorial.org/sql-limit/

Home » SQL Tutorial » SQL LIMITSQL LIMITSummary: in this tutorial, you’ll learn to use the SQL LIMIT clause to limit the number of rows returned from a query.Introduction to SQL LIMIT clauseTo limit the number of rows returned by a select statement, you use the LIMIT and OFFSET clauses.The following shows the syntax of LIMIT & OFFSET clauses:SELECT   
 column\_list  
FROM  
 table1  
ORDER BY column\_list  
LIMIT row\_count OFFSET offset;Code language: SQL (Structured Query Language) (sql)In this syntax:The LIMIT row\_count determines the number of rows (row\_count) returned by the query.The OFFSET offset clause skips the offset rows before beginning to return the rows.The OFFSET clause is optional. If you omit it, the query will return the row\_count rows from the first row returned by the SELECT clause.When you use the LIMIT clause, it is important to use an ORDER BY clause to ensure the order of rows in the result set.Not all database systems support the LIMIT clause. Therefore, the LIMIT clause is available only in some database systems only such as MySQL, PostgreSQL, SQLite, Sybase SQL Anywhere, and HSQLDB. If you use SQL Server, you can use the SELECT TOP instead.SQL LIMIT clause examplesWe’ll use the employees table in the sample database to demonstrate the LIMIT & OFFSET clauses.The following statement returns all rows in the employees table sorted by the first\_name column.SELECT   
 employee\_id,   
 first\_name,   
 last\_name  
FROM  
 employees  
ORDER BY   
 first\_name;Code language: SQL (Structured Query Language) (sql)Try ItThe following example uses the LIMIT clause to return the first 5 rows in the result set returned by the SELECT clause:SELECT   
 employee\_id,   
 first\_name,   
 last\_name  
FROM  
 employees  
ORDER BY   
 first\_name  
LIMIT 5; Code language: SQL (Structured Query Language) (sql)Try ItThe following example uses both LIMIT & OFFSET clauses to return five rows starting from the 4th row:SELECT   
 employee\_id, first\_name, last\_name  
FROM  
 employees  
ORDER BY first\_name  
LIMIT 5 OFFSET 3;Code language: SQL (Structured Query Language) (sql)Try ItIn MySQL, you can use the shorter form of the LIMIT & OFFSET clauses like this:SELECT   
 employee\_id,   
 first\_name,   
 last\_name  
FROM  
 employees  
ORDER BY   
 first\_name  
LIMIT 3 , 5;Code language: SQL (Structured Query Language) (sql)Try ItUsing SQL LIMIT to get the top N rows with the highest or lowest valueYou can use the LIMIT clause to get the top N rows with the highest or lowest value. For example, the following statement gets the top five employees with the highest salaries.SELECT   
 employee\_id,   
 first\_name,   
 last\_name,   
 salary  
FROM  
 employees  
ORDER BY   
 salary DESC  
LIMIT 5;Code language: SQL (Structured Query Language) (sql)Try ItFirst, the ORDER BY clause sorts the employees by salary in descending order and then the LIMIT clause restricts five rows returned from the query.To get the top five employees with the lowest salary, you sort the employees by salary in the ascending order instead.Getting the rows with the Nth highest valueSuppose you have to get employees who have the 2nd highest salary in the company. To do so, you use the LIMIT OFFSET clauses as follows.SELECT   
 employee\_id,   
 first\_name,   
 last\_name,   
 salary  
FROM  
 employees  
ORDER BY   
 salary DESC  
LIMIT 1 OFFSET 1;Code language: SQL (Structured Query Language) (sql)Try ItThe ORDER BY clause sorts the employees by salary in descending order. And the LIMIT 1 OFFSET 1 clause gets the second row from the result set.This query works with the assumption that every employee has a different salary. It will fail if there are two employees who have the same highest salary.Also, if you have two or more employees who have the same 2nd highest salary, the query just returns the first one.To fix this issue, you can get the second highest salary first using the following statement.SELECT DISTINCT  
 salary  
FROM  
 employees  
ORDER BY salary DESC  
LIMIT 1 , 1;Code language: SQL (Structured Query Language) (sql)Try ItAnd pass the result to another query:SELECT   
 employee\_id, first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 salary = 17000;Code language: SQL (Structured Query Language) (sql)Try ItIf you know subquery, you can combine both queries into a single query as follows:SELECT   
 employee\_id, first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 salary = (SELECT DISTINCT  
 salary  
 FROM  
 employees  
 ORDER BY salary DESC  
 LIMIT 1 , 1);Code language: SQL (Structured Query Language) (sql)Try ItSummaryUse LIMIT & OFFSET clauses to limit the number of rows returned by a query.LIMIT & OFFSET is not SQL standard.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL DISTINCT Up NextSQL FETCH

# Content from https://www.sqltutorial.org/sql-fetch/

Home » SQL Tutorial » SQL FETCHSQL FETCHSummary: in this tutorial, you will learn how to use the SQL FETCH clause to limit the number of rows returned by a query.Introduction to SQL FETCH clauseTo limit the number of rows returned by a query, you use the LIMIT clause. The LIMIT clause is widely supported by many database systems such as MySQL, H2, and HSQLDB. However, the LIMIT clause is not in SQL standard.SQL:2008 introduced the OFFSET FETCH clause which has a similar function to the LIMIT clause. The OFFSET FETCH clause allows you to skip N first rows in a result set before starting to return any rows.The following shows the syntax of the SQL FETCH clause:OFFSET offset\_rows { ROW | ROWS }  
FETCH { FIRST | NEXT } [ fetch\_rows ] { ROW | ROWS } ONLYCode language: SQL (Structured Query Language) (sql)In this syntax:The ROW and ROWS, FIRST and NEXT are the synonyms. Therefore, you can use them interchangeably.The offset\_rows is an integer number which must be zero or positive. In case the offset\_rows is greater than the number of rows in the result set, no rows will be returned.The fetch\_rows is also an integer number that determines the number of rows to be returned. The value of fetch\_rows is equal to or greater than one.Because rows are stored in the table in an unspecified order, you should always use the FETCH clause with the ORDER BY clause to get consistent output.Many database systems support the OFFSET FETCH clause including Oracle Database 12c+, PostgreSQL 10+, and Microsoft SQL Server 2012+. However, each database system implements the OFFSET FETCH clause differently with some variances.The OFFSET FETCH clause is typically used in the client or web applications that require pagination. For example, if each page has ten rows, to get the rows of the second page, you can skip the first ten rows and returns the next ten rows.SQL FETCH examplesWe will use the employees table in the sample database for the demonstration.The following statement returns the first employee who has the highest salary:SELECT   
 employee\_id,   
 first\_name,   
 last\_name,   
 salary  
FROM employees  
ORDER BY   
 salary DESC  
OFFSET 0 ROWS  
FETCH NEXT 1 ROWS ONLY;Code language: SQL (Structured Query Language) (sql)In this example, first, the ORDER BY clause sort the employees by salary from high to low. The OFFSET clause skips zero rows and the FETCH clause returns the first row.The following statement sorts the employees by salary, skips the first five employees with the highest salary, and fetches the next five ones.SELECT   
 employee\_id,   
 first\_name,   
 last\_name,   
 salary  
FROM employees  
ORDER BY   
 salary DESC  
OFFSET 5 ROWS  
FETCH NEXT 5 ROWS ONLY;Code language: SQL (Structured Query Language) (sql)SummaryUse the SQL FETCH clause to limit the number of rows returned by a query.The SQL FETCH clause skips N rows in a result set before starting to return any rows.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL LIMIT Up NextSQL WHERE

# Content from https://www.sqltutorial.org/sql-where/

Home » SQL Tutorial » SQL WHERESQL WHERESummary: in this tutorial, you will learn how to use the SQL WHERE clause to filter rows based on specified conditions.Introduction to SQL WHERE clauseTo select specific rows from a table, you use a WHERE clause in the SELECT statement. The following illustrates the syntax of the WHERE clause in the SELECT statement:SELECT   
 column1, column2, ...  
FROM  
 table\_name  
WHERE  
 condition;Code language: SQL (Structured Query Language) (sql)The WHERE clause appears immediately after the FROM clause. The WHERE clause contains one or more logical expressions that evaluate each row in the table. If a row that causes the condition evaluates to true, it will be included in the result set; otherwise, it will be excluded.Note that SQL has three-valued logic which are TRUE, FALSE, and UNKNOWN. It means that if a row causes the condition to evaluate to FALSE or NULL, the row will not be returned.Note that the logical expression that follows the WHERE clause is also known as a predicate. You can use various operators to form the row selection criteria used in the WHERE clause.The following table shows the SQL comparison operators:OperatorMeaning=Equal to<> (!=)Not equal to<Less than>Greater than<=Less than or equal>=Greater than or equalTo form a simple expression, you use one of the operators above with two operands that can be either column name on one side and a literal value on the other, for example:salary > 1000Code language: SQL (Structured Query Language) (sql)It asks a question: “Is salary greater than 1000?”.Or you can use column names on both sides of an operator such as:min\_salary < max\_salaryCode language: SQL (Structured Query Language) (sql)This expression asks another question: “Is the min salary less than the max salary?”.The literal values that you use in an expression can be numbers, characters, dates, and times, depending on the format you use:Number: use a number that can be an integer or a decimal without any formatting e.g., 100, 200.5Character: use characters surrounded by either single or double quotes e.g., “100”, “John Doe”.Date: use the format that the database stores. It depends on the database system e.g., MySQL uses 'yyyy-mm-dd' format to store the date data.Time: use the format that the database system uses to store the time. For example, MySQL uses 'HH:MM:SS' to store time data.Besides the SELECT statement, you can use the WHERE clause in the UPDATE or DELETE statement to specify which rows to be updated or deleted.SQL WHERE examplesWe will use the employees table to demonstrate how to select data from the table using the WHERE clause.SQL WHERE clause with numeric comparison examplesThe following query finds employees who have salaries greater than 14,000 and sorts the result set based on the salary in descending order.SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 salary  
FROM  
 employees  
WHERE  
 salary > 14000  
ORDER BY  
 salary DESC;Try ItThe following query finds all employees who work in the department id 5.SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 department\_id  
FROM  
 employees  
WHERE  
 department\_id = 5  
ORDER BY  
 first\_name;Code language: SQL (Structured Query Language) (sql)Try ItSQL WHERE clause with characters exampleSQL is case-insensitive. However, when it comes to the values in the comparisons, it is case-sensitive. For instance, the following query finds the employee whose last name is Chen.SELECT  
 employee\_id,  
 first\_name,  
 last\_name  
FROM  
 employees  
WHERE  
 last\_name = 'Chen';Code language: SQL (Structured Query Language) (sql)Try ItHowever, if you use CHEN or chen, no row will be returned.SQL WHERE clause with dates examplesTo get all employees who joined the company after January 1st, 1999, you use the following query:SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 hire\_date  
FROM  
 employees  
WHERE  
 hire\_date >= '1999-01-01'  
ORDER BY  
 hire\_date DESC;Code language: SQL (Structured Query Language) (sql)Try ItIf you want to find the employees who joined the company in 1999, you have several ways:Use the YEAR function to get the year data from the hire\_date column and use the equal to (=) operator to form the expression.Use two expressions with the AND operator.Use the BETWEEN operator.The following statement illustrates the first way:SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 hire\_date  
FROM  
 employees  
WHERE  
 YEAR (hire\_date) = 1999  
ORDER BY  
 hire\_date DESC;Code language: SQL (Structured Query Language) (sql)Try ItIn this tutorial, we have shown you how to use the SQL WHERE clause to filter data based on a specified condition.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL FETCH Up NextSQL Comparison Operators

# Content from https://www.sqltutorial.org/sql-comparison-operators/

Home » SQL Tutorial » SQL Comparison OperatorsSQL Comparison OperatorsSummary: in this tutorial, you will learn about SQL comparison operators and how to use them to form conditions for filtering data.The SQL comparison operators allow you to test if two expressions are the same. The following table illustrates the comparison operators in SQL:OperatorMeaning=Equal<>Not equal to>Greater than>=Greater than or equal to<Less than<=Less than or equal toThe result of a comparison operator has one of three value true, false, and unknown.Equal to operator(=)The equal to operator compares the equality of two expressions:expression1 = expression2  
Code language: SQL (Structured Query Language) (sql)It returns true if the value of the left expression is equal to the value of the right expression; otherwise, it returns false.For example, the following statement finds the employee whose last name is Himuro:SELECT   
 employee\_id, first\_name, last\_name  
FROM  
 employees  
WHERE  
 last\_name = 'Himuro';   
Code language: SQL (Structured Query Language) (sql)In this example, the query searches for the string Himuro in the last\_name column of the employees table.Note that the equal operator cannot be used to compare null values. For example, the intention of the following query is to find all employees who do not have phone numbers:SELECT   
 employee\_id, first\_name, last\_name, phone\_number  
FROM  
 employees  
WHERE  
 phone\_number = NULL;  
Code language: SQL (Structured Query Language) (sql)However, it returns an empty result set because the following expression always returns false.phone\_number = NULL  
Code language: SQL (Structured Query Language) (sql)To compare null values, you use the IS NULL operator instead:SELECT   
 employee\_id, first\_name, last\_name, phone\_number  
FROM  
 employees  
WHERE  
 phone\_number IS NULL;  
Code language: SQL (Structured Query Language) (sql)Not equal to operator (<>)The not equal to (<>) operator compares two non-null expressions and returns true if the value of the left expression is not equal to the right one; otherwise, it returns false.expression1 <> expression2  
Code language: SQL (Structured Query Language) (sql)For example, the following statement returns all employees whose department id is not 8.SELECT   
 employee\_id, first\_name, last\_name, department\_id  
FROM  
 employees  
WHERE  
 department\_id <> 8  
ORDER BY first\_name , last\_name;  
Code language: SQL (Structured Query Language) (sql)You can use the AND operator to combine multiple expressions that use the not equal to (<>) operator. For example, the following statement finds all employees whose department id is not eight and ten.SELECT   
 employee\_id, first\_name, last\_name, department\_id  
FROM  
 employees  
WHERE  
 department\_id <> 8  
 AND department\_id <> 10  
ORDER BY first\_name , last\_name;   
Code language: SQL (Structured Query Language) (sql)Greater than operator (>)The greater than operator (>) compares two non-null expressions and returns true if the left operand is greater than the right operand; otherwise, the result is false.expression1 > expression2  
Code language: SQL (Structured Query Language) (sql)For example, to find the employees whose salary is greater than 10,000, you use the greater than operator in the WHERE clause as follows:SELECT   
 employee\_id, first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 salary > 10000  
ORDER BY salary DESC;  
Code language: SQL (Structured Query Language) (sql)You can combine expressions that use various comparison operators using the AND or OR operator. For example, the following statement finds employees in department 8 and have the salary greater than 10,000:SELECT   
 employee\_id, first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 salary > 10000 AND department\_id = 8  
ORDER BY salary DESC;  
Code language: SQL (Structured Query Language) (sql)Less than operator (<)The less than operator compares two non-null expressions. The result is true if the left operand evaluates to a value that is lower than the value of the right operand; otherwise, the result is false.The following shows the syntax of the less than operator:expression1 < expression2  
Code language: SQL (Structured Query Language) (sql)For example, the statement below returns all employees whose salaries are less than 10,000:SELECT   
 employee\_id, first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 salary < 10000  
ORDER BY salary DESC;Code language: SQL (Structured Query Language) (sql)Greater than or equal operator (>=)The greater than or equal operator (>=) compares two non-null expressions. The result is true if the left expression evaluates to a value that is greater than the value of the right expression.The following illustrates the syntax of the greater than or equal operator:expression1 >= expression2  
Code language: SQL (Structured Query Language) (sql)For example, the following statement finds employees whose salaries are greater than or equal 9,000:SELECT   
 employee\_id, first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 salary >= 9000  
ORDER BY salary;Code language: SQL (Structured Query Language) (sql)Less than or equal to operator(<=)The less than or equal to operator compares two non-null expressions and returns true if the left expression has a value less than or equal the value of the right expression; otherwise, it returns true.The following shows the syntax of the less than or equal to operator:expression1 <= expression2Code language: SQL (Structured Query Language) (sql)For example, the following statement finds employees whose salaries are less than or equal to 9,000:In this tutorial, you have learned how to use the comparison operators to form comparison expression for filtering data based on a specified condition.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL WHERE Up NextSQL Logical Operators

# Content from https://www.sqltutorial.org/sql-logical-operators/

Home » SQL Tutorial » SQL Logical OperatorsSQL Logical OperatorsSummary: in this tutorial, you will learn about the SQL logical operators and how to use them to test for the truth of a condition.A logical operator allows you to test for the truth of a condition. Similar to a comparison operator, a logical operator returns a value of true, false, or unknown.The following table illustrates the SQL logical operators:OperatorMeaningALLReturn true if all comparisons are trueANDReturn true if both expressions are trueANYReturn true if any one of the comparisons is true.BETWEENReturn true if the operand is within a rangeEXISTSReturn true if a subquery contains any rowsINReturn true if the operand is equal to one of the value in a listLIKEReturn true if the operand matches a patternNOTReverse the result of any other Boolean operator.ORReturn true if either expression is trueSOMEReturn true if some of the expressions are trueANDThe AND operator allows you to construct multiple conditions in the WHERE clause of an SQL statement such as SELECT, UPDATE, and DELETE:expression1 AND expression2Code language: SQL (Structured Query Language) (sql)The AND operator returns true if both expressions evaluate to true.The following example finds all employees whose salaries are greater than 5,000 and less than 7,000:SELECT   
 first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 salary > 5000 AND salary < 7000  
ORDER BY salary;Code language: SQL (Structured Query Language) (sql)ORSimilar to the AND operator, the OR operator combines multiple conditions in an SQL statement’s WHERE clause:expression1 OR expression2Code language: SQL (Structured Query Language) (sql)However, the OR operator returns true if a least one expression evaluates to true.For example, the following statement finds employees whose salary is either 7,000 or 8,000:SELECT   
 first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 salary = 7000 OR salary = 8000  
ORDER BY salary;Code language: SQL (Structured Query Language) (sql)IS NULLThe IS NULL operator compares a value with a null value and returns true if the compared value is null; otherwise, it returns false.For example, the following statement finds all employees who do not have a phone number:SELECT   
 first\_name, last\_name, phone\_number  
FROM  
 employees  
WHERE  
 phone\_number IS NULL  
ORDER BY first\_name , last\_name;  
Code language: SQL (Structured Query Language) (sql)BETWEENThe BETWEEN operator searches for values that are within a set of values, given the minimum value and maximum value. Note that the minimum and maximum values are included as part of the conditional set.For example, the following statement finds all employees whose salaries are between 9,000 and 12,000.SELECT   
 first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 salary BETWEEN 9000 AND 12000  
ORDER BY salary;   
Code language: SQL (Structured Query Language) (sql)Notice that the value 9,000 and 12,000 are included in the output.INThe IN operator compares a value to a list of specified values. The IN operator returns true if the compared value matches at least one value in the list; otherwise, it returns false.The following statement finds all employees who work in the department id 8 or 9.SELECT   
 first\_name, last\_name, department\_id  
FROM  
 employees  
WHERE  
 department\_id IN (8, 9)  
ORDER BY department\_id;  
Code language: SQL (Structured Query Language) (sql)LIKEThe LIKE operator compares a value to similar values using a wildcard operator. SQL provides two wildcards used in conjunction with the LIKE operator:The percent sign ( %) represents zero, one, or multiple characters.The underscore sign ( \_) represents a single character.The following statement finds all employees whose first name starts with the string jo:SELECT   
 employee\_id, first\_name, last\_name  
FROM  
 employees  
WHERE  
 first\_name LIKE 'jo%'  
ORDER BY first\_name;  
Code language: SQL (Structured Query Language) (sql)The following example finds all employees with the first names whose the second character is  h:SELECT   
 employee\_id, first\_name, last\_name  
FROM  
 employees  
WHERE  
 first\_name LIKE '\_h%'  
ORDER BY first\_name;  
Code language: SQL (Structured Query Language) (sql)ALLThe ALL operator compares a value to all values in another value set. The ALL operator must be preceded by a comparison operator and followed by a subquery.The following illustrates the syntax of the ALL operator:comparison\_operator ALL (subquery)  
Code language: SQL (Structured Query Language) (sql)Note that you will learn about the subquery in the subquery tutorial.The following example finds all employees whose salaries are greater than all salaries of employees in the department 8:SELECT   
 first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 salary >= ALL (SELECT   
 salary  
 FROM  
 employees  
 WHERE  
 department\_id = 8)  
ORDER BY salary DESC;Code language: SQL (Structured Query Language) (sql)ANYThe ANY operator compares a value to any value in a set according to the condition as shown below:comparison\_operator ANY(subquery)Code language: SQL (Structured Query Language) (sql)Similar to the ALL operator, the ANY operator must be preceded by a comparison operator and followed by a subquery.For example, the following statement finds all employees whose salaries are greater than the average salary of every department:SELECT   
 first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 salary > ANY(SELECT   
 AVG(salary)  
 FROM  
 employees  
 GROUP BY department\_id)  
ORDER BY first\_name , last\_name;   
Code language: SQL (Structured Query Language) (sql)Note that SOME is an alias for ANY, therefore, you can use them interchangeably.EXISTSThe EXISTS operator tests if a subquery contains any rows:EXISTS (subquery)Code language: SQL (Structured Query Language) (sql)If the subquery returns one or more rows, the result of the EXISTS is true; otherwise, the result is false.For example, the following statement finds all employees who have dependents:SELECT   
 first\_name, last\_name  
FROM  
 employees e  
WHERE  
 EXISTS( SELECT   
 1  
 FROM  
 dependents d  
 WHERE  
 d.employee\_id = e.employee\_id);  
Code language: SQL (Structured Query Language) (sql)Now you should have a brief overview of all SQL logical operators and how to use them to test the truth of a condition. In the next tutorials, you will learn about each logical operator in detail.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL Comparison Operators Up NextSQL AND

# Content from https://www.sqltutorial.org/sql-and/

Home » SQL Tutorial » SQL ANDSQL ANDSummary: this tutorial introduces you to the SQL AND operator and shows you how to apply it to form flexible conditions in the WHERE clause of an SQL statement.Introduction to SQL AND operatorThe AND operator is a logical operator that combines two Boolean expressions in the WHERE clause of the SELECT, UPDATE, or DELETE statement. The following illustrates the syntax of the AND operator:expression1 AND expression2;The AND operator returns true if both expressions evaluate to true. If one of the two expressions is false, then the AND operator returns false even one of the expressions is NULL.The following table illustrates the results of the AND operator when comparing true, false, and NULL values:TRUEFALSENULLTRUETRUEFALSENULLFALSEFALSEFALSEFALSENULLNULLFALSENULLSQL AND operator examplesWe will use the employees table from the sample database to demonstrate the SQL AND operator.The following statement finds all employees who have both job id 9 and salary greater than 5,000:SELECT  
 first\_name,  
 last\_name,  
 job\_id,  
 salary  
FROM  
 employees  
WHERE  
 job\_id = 9  
AND salary > 5000;Try ItTo find all the employees who joined the company between 1997 and 1998, you use the AND operator as follows:SELECT  
 first\_name,  
 last\_name,  
 hire\_date  
FROM  
 employees  
WHERE  
 YEAR (hire\_date) >=1997  
AND YEAR (hire\_date) <= 1998;Code language: SQL (Structured Query Language) (sql)Try ItSQL AND operator and short-circuit evaluationThe short-circuit feature allows the database system to stop evaluating the remaining parts of a logical expression as soon as it can determine the result.Let’s take a look at an example to get a better understanding of how the to short-circuit evaluation feature works.See the following condition:1 = 0 AND 1 = 1;The database system processes the two comparisons first and uses the AND operator to evaluate the two results.However, with the short-circuit evaluation feature, the database system just has to evaluate the left part of the expression because the left part (1 = 0) returns false that causes the whole condition returns false regardless of the result of the right part of the condition.The short-circuit feature, therefore, decreases the CPU computation time and in some cases helps prevent runtime-error. Consider the following condition:1 = 0 AND 1/0;If the database system supports the short-circuit feature, it will not evaluate the right part of the expression (1/0) that causes the division by zero error.Now, you should understand how the SQL AND operator works and how to apply it to form a complex condition in your queries.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL Logical Operators Up NextSQL OR

# Content from https://www.sqltutorial.org/sql-or/

Home » SQL Tutorial » SQL ORSQL ORSummary: in this tutorial, you will learn how to use the SQL OR operator to combine two Boolean expressions.Introduction to SQL OR operatorThe SQL OR is a logical operator that combines two boolean expressions. The SQL OR operator returns either true or false depending on the results of expressions.The OR operator is typically used in the WHERE clause of the SELECT, UPDATE, or DELETE statement to form a flexible condition.The following illustrates the syntax of the SQL OR operator:SELECT  
 column1,  
 column2,  
 ...  
FROM  
 table1  
WHERE  
 expression1  
OR expression2;The following table shows the result of the OR operator when we compare the true, false, and NULL values.TRUEFALSENULLTRUETRUETRUETRUEFALSETRUEFALSENULLNULLTRUENULLNULLNotice that the OR operator always returns true if either expression is true.If the database system supports the short-circuit feature, the OR operator stops evaluating the remaining parts of the condition as soon as one expression is true.Note that the short-circuit feature helps the database system save CPU computation by aborting processing the remaining part of a logical expression as soon as it can determine the result. For more information on the short-circuit feature, check it out the SQL AND operator tutorial.When you use the OR operator with the AND operator, the database system evaluates the OR operator after the AND operator. This is known as the rule of precedence. However, you can use parentheses to change the order of evaluation.SQL OR operator examplesWe will use the  employees table from the sample database for the demonstration of the OR operator.The following statement finds all employees who joined the company in 1997 or 1998.SELECT  
 first\_name,  
 last\_name,  
 hire\_date  
FROM  
 employees  
WHERE  
 YEAR (hire\_date) = 1997 OR YEAR (hire\_date) = 1998  
ORDER BY  
 first\_name,  
 last\_name;  
Try ItTo find all employees who joined the company  in 1997 or 1997 and worked in the department id 3, you use both AND and OR operators as follows:SELECT  
 first\_name,  
 last\_name,  
 hire\_date,  
 department\_id  
FROM  
 employees  
WHERE  
 department\_id = 3  
AND (  
 YEAR (hire\_date) = 1997  
 OR YEAR (hire\_date) = 1998  
)  
ORDER BY  
 first\_name,  
 last\_name;Try ItIf you don’t use the parentheses, the query will retrieve employees who joined the company in 1997 and worked in department id 3 or employees who joined the company in 1998 regardless of departments.This is because the database system evaluates the OR operator after the AND operator.SELECT  
 first\_name,  
 last\_name,  
 hire\_date,  
 department\_id  
FROM  
 employees  
WHERE  
 department\_id = 3  
AND YEAR (hire\_date) = 1997  
OR YEAR (hire\_date) = 1998  
ORDER BY  
 first\_name,  
 last\_name;Code language: SQL (Structured Query Language) (sql)Try ItIf a query uses many OR operators, it will become difficult to read. To make the query more readable, you can use the IN operator instead.For example, the following query finds all employees who joined the company in 1990 or 1999 or 2000.SELECT  
 first\_name,  
 last\_name,  
 hire\_date  
FROM  
 employees  
WHERE  
 YEAR (hire\_date) = 2000  
OR YEAR (hire\_date) = 1999  
OR YEAR (hire\_date) = 1990;Code language: SQL (Structured Query Language) (sql)Try ItYou can replace the OR operators by the IN operator as follows:SELECT  
 first\_name,  
 last\_name,  
 hire\_date  
FROM  
 employees  
WHERE  
 YEAR (hire\_date) IN (1990, 1999, 2000)  
ORDER BY  
 hire\_date;Code language: SQL (Structured Query Language) (sql)Try ItIn this tutorial, you have learned how to use the SQL OR operator to combines two Boolean expressions to form a flexible condition.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL AND Up NextSQL BETWEEN

# Content from https://www.sqltutorial.org/sql-between/

Home » SQL Tutorial » SQL BETWEENSQL BETWEENSummary: in this tutorial, you’ll learn how to use the SQL BETWEEN operator to check if a value falls within a specific range.Introduction to SQL BETWEEN operatorThe BETWEEN operator is one of the logical operators in SQL. The BETWEEN operator checks if a value is within a range of values.The syntax of the BETWEEN operator is as follows:expression BETWEEN low AND high;Code language: SQL (Structured Query Language) (sql)The BETWEEN operator returns true if the expression is greater than or equal to ( >=) the low value and less than or equal to ( <=) the high value.Technically, the BETWEEN is the equivalent to the following expression that uses the greater than or equal to (>=) and less than or equal to (<=) operators:expression >= low AND expression <= highCode language: SQL (Structured Query Language) (sql)To compare a value with an exclusive range, you need to use the comparison operators less than (<) and greater than ( >).NOT BETWEENTo negate the result of the BETWEEN operator, you use the NOT operator:expression NOT BETWEEN low AND highCode language: SQL (Structured Query Language) (sql)The NOT BETWEEN returns true if the expression is less than low or greater than (>) high; otherwise, it returns false.Like the BETWEEN operator, you can rewrite the NOT BETWEEN operator using the less than (<) and greater than (>) operators with the OR operator as follows:expression < low OR expression > highCode language: SQL (Structured Query Language) (sql)In practice, you often use the BETWEEN and NOT BETWEEN operator in the WHERE clause of the SELECT to select rows whose value of a column is within a specific range.SQL BETWEEN operator examplesWe’ll use the employees table from the sample database to illustrate how the BETWEEN operator works.1) Using the SQL BETWEEN opeator with numbers exampleThe following statement uses the BETWEEN operator to find all employees whose salaries are between 2,500 and 2,900:SELECT   
 employee\_id,   
 first\_name,   
 last\_name,   
 salary  
FROM  
 employees  
WHERE  
 salary BETWEEN 2500 AND 2900  
ORDER BY   
 salary DESC;Code language: SQL (Structured Query Language) (sql)Try It+-------------+------------+-------------+---------+  
| employee\_id | first\_name | last\_name | salary |  
+-------------+------------+-------------+---------+  
| 116 | Shelli | Baida | 2900.00 |  
| 117 | Sigal | Tobias | 2800.00 |  
| 126 | Irene | Mikkilineni | 2700.00 |  
| 118 | Guy | Himuro | 2600.00 |  
| 119 | Karen | Colmenares | 2500.00 |  
+-------------+------------+-------------+---------+  
Notice that the result set includes the employees whose salaries are 2,500 and 2,900.The following query returns the same result set as the above query. However, it uses comparison operators greater than or equal to (>=) and less than or equal to (<=) instead:SELECT   
 employee\_id,   
 first\_name,   
 last\_name,   
 salary  
FROM  
 employees  
WHERE  
 salary >= 2500 AND salary <= 2900  
ORDER BY   
 salary;Code language: SQL (Structured Query Language) (sql)Try It2) Using SQL NOT BETWEEN exampleThe following example uses the NOT BETWEEN operator to find all employees whose salaries are not in the range of 2,500 and 2,900:SELECT   
 employee\_id,   
 first\_name,   
 last\_name,   
 salary  
FROM  
 employees  
WHERE  
 salary NOT BETWEEN 2500 AND 2900  
ORDER BY salary DESC;Code language: SQL (Structured Query Language) (sql)Try It+-------------+-------------+------------+----------+  
| employee\_id | first\_name | last\_name | salary |  
+-------------+-------------+------------+----------+  
| 100 | Steven | King | 24000.00 |  
| 101 | Neena | Kochhar | 17000.00 |  
| 102 | Lex | De Haan | 17000.00 |  
| 145 | John | Russell | 14000.00 |  
| 146 | Karen | Partners | 13500.00 |  
| 201 | Michael | Hartstein | 13000.00 |  
| 205 | Shelley | Higgins | 12000.00 |  
| 108 | Nancy | Greenberg | 12000.00 |  
| 114 | Den | Raphaely | 11000.00 |  
| 204 | Hermann | Baer | 10000.00 |  
| 109 | Daniel | Faviet | 9000.00 |  
| 103 | Alexander | Hunold | 9000.00 |  
| 176 | Jonathon | Taylor | 8600.00 |  
| 177 | Jack | Livingston | 8400.00 |  
| 206 | William | Gietz | 8300.00 |  
| 121 | Adam | Fripp | 8200.00 |  
| 110 | John | Chen | 8200.00 |  
| 120 | Matthew | Weiss | 8000.00 |  
| 122 | Payam | Kaufling | 7900.00 |  
| 112 | Jose Manuel | Urman | 7800.00 |  
| 111 | Ismael | Sciarra | 7700.00 |  
| 178 | Kimberely | Grant | 7000.00 |  
| 113 | Luis | Popp | 6900.00 |  
| 123 | Shanta | Vollman | 6500.00 |  
| 203 | Susan | Mavris | 6500.00 |  
| 179 | Charles | Johnson | 6200.00 |  
| 202 | Pat | Fay | 6000.00 |  
| 104 | Bruce | Ernst | 6000.00 |  
| 106 | Valli | Pataballa | 4800.00 |  
| 105 | David | Austin | 4800.00 |  
| 200 | Jennifer | Whalen | 4400.00 |  
| 107 | Diana | Lorentz | 4200.00 |  
| 192 | Sarah | Bell | 4000.00 |  
| 193 | Britney | Everett | 3900.00 |  
| 115 | Alexander | Khoo | 3100.00 |  
+-------------+-------------+------------+----------+3) Using SQL BETWEEN operator with a date rangesThe following example uses the BETWEEN operator to find all employees who joined the company between January 1, 1999, and December 31, 2000:SELECT   
 employee\_id,   
 first\_name,   
 last\_name,   
 hire\_date  
FROM  
 employees  
WHERE  
 hire\_date BETWEEN '1999-01-01' AND '2000-12-31'  
ORDER BY   
 hire\_date;Code language: SQL (Structured Query Language) (sql)Try It+-------------+------------+------------+------------+  
| employee\_id | first\_name | last\_name | hire\_date |  
+-------------+------------+------------+------------+  
| 107 | Diana | Lorentz | 1999-02-07 |  
| 178 | Kimberely | Grant | 1999-05-24 |  
| 119 | Karen | Colmenares | 1999-08-10 |  
| 113 | Luis | Popp | 1999-12-07 |  
| 179 | Charles | Johnson | 2000-01-04 |  
+-------------+------------+------------+------------+  
The following example uses the NOT BETWEEN operator to find employees who have not joined the company from January 1, 1989 to December 31, 1999:SELECT   
 employee\_id,   
 first\_name,   
 last\_name,   
 hire\_date  
FROM  
 employees  
WHERE  
 hire\_date NOT BETWEEN '1989-01-01' AND '1992-12-31'  
ORDER BY   
 hire\_date;Code language: SQL (Structured Query Language) (sql)Try It+-------------+------------+-----------+------------+  
| employee\_id | first\_name | last\_name | hire\_date |  
+-------------+------------+-----------+------------+  
| 100 | Steven | King | 1987-06-17 |  
| 200 | Jennifer | Whalen | 1987-09-17 |  
| 179 | Charles | Johnson | 2000-01-04 |  
+-------------+------------+-----------+------------+  
3 rows in set (0.00 sec)Code language: JavaScript (javascript)4) Using SQL BETWEEN operator with a function exampleThe following example uses the BETWEEN operator with the YEAR function to find employees who joined the company between 1990 and 1993:SELECT   
 employee\_id,   
 first\_name,   
 last\_name,   
 year(hire\_date) joined\_year  
FROM  
 employees  
WHERE   
 year(hire\_date) BETWEEN 1990 and 1993   
ORDER BY   
 hire\_date;Output:+-------------+------------+-----------+-------------+  
| employee\_id | first\_name | last\_name | joined\_year |  
+-------------+------------+-----------+-------------+  
| 103 | Alexander | Hunold | 1990 |  
| 104 | Bruce | Ernst | 1991 |  
| 102 | Lex | De Haan | 1993 |  
+-------------+------------+-----------+-------------+In this example:First, the YEAR() function returns the year from the hire date.Second, the BETWEEN operator uses the result of the YEAR() function and check if it is within the range 1990 and 1993.If your database doesn’t support the YEAR() function, you need to use a similar function:DatabaseThe function to extract the year from a datePostgreSQLDATE\_PART('year', hire\_date)OracleEXTRACT(year from hire\_date)SQL ServerYEAR(hire\_date)SummaryThe BETWEEN operator returns true if a value is within a specific range.Use the NOT operator to negate the BETWEEN opeator.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL OR Up NextSQL IN

# Content from https://www.sqltutorial.org/sql-in/

Home » SQL Tutorial » SQL INSQL INSummary: in this tutorial, you will learn how to use the SQL IN operator to check if a value is in a set of values.Introduction to SQL IN OperatorThe IN is a logical operator in SQL. The IN operator returns true if a value is in a set of values or false otherwise.The following illustrates the syntax of the IN operator:expression IN (value1,value2,...)Code language: SQL (Structured Query Language) (sql)Technically, you can substitute the IN operator with the = and OR operators The condition that uses the IN operator can be rewritten using one or more OR operators as follows:expression = value1 OR expression = value2 OR ...Code language: SQL (Structured Query Language) (sql)To negate the IN operator, you use the NOT operator:expression NOT IN (value1, value2,...)Code language: SQL (Structured Query Language) (sql)The NOT IN operator returns true if the expression does not equal any values in the list or false otherwise.To substitute the IN operator, you can use the != and AND operators as follows:expression != value1 AND expression != value2 AND...Code language: SQL (Structured Query Language) (sql)Notice that if any value in the list (value1,value2,...) is null, the IN operator returns no rows.In practice, you often use the IN and NOT IN operators in the  WHERE clause of the SELECT statement to select rows with a value in a set of values. Also, you’ll use the IN operator in subqueries.SQL IN examplesWe will use the  employees table in the sample database to demonstrate the functionality of the IN operator.The following example uses the IN operator to find employees with the job id is 8, 9, or 10:SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 job\_id  
FROM  
 employees  
WHERE  
 job\_id IN (8, 9, 10)  
ORDER BY  
 job\_id;Code language: SQL (Structured Query Language) (sql)Try It+-------------+------------+-----------+--------+  
| employee\_id | first\_name | last\_name | job\_id |  
+-------------+------------+-----------+--------+  
| 203 | Susan | Mavris | 8 |  
| 103 | Alexander | Hunold | 9 |  
| 104 | Bruce | Ernst | 9 |  
| 105 | David | Austin | 9 |  
| 106 | Valli | Pataballa | 9 |  
| 107 | Diana | Lorentz | 9 |  
| 201 | Michael | Hartstein | 10 |  
+-------------+------------+-----------+--------+The following example uses the NOT IN operator to find employees whose job’s id is neither 7, 8, nor 9:SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 job\_id  
FROM  
 employees  
WHERE  
 job\_id NOT IN (7, 8, 9)  
ORDER BY  
 job\_id;Code language: SQL (Structured Query Language) (sql)Try It+-------------+-------------+-------------+--------+  
| employee\_id | first\_name | last\_name | job\_id |  
+-------------+-------------+-------------+--------+  
| 206 | William | Gietz | 1 |  
| 205 | Shelley | Higgins | 2 |  
| 200 | Jennifer | Whalen | 3 |  
| 100 | Steven | King | 4 |  
| 102 | Lex | De Haan | 5 |  
| 101 | Neena | Kochhar | 5 |  
| 109 | Daniel | Faviet | 6 |  
| 113 | Luis | Popp | 6 |  
| 110 | John | Chen | 6 |  
| 111 | Ismael | Sciarra | 6 |  
| 112 | Jose Manuel | Urman | 6 |  
| 201 | Michael | Hartstein | 10 |  
| 202 | Pat | Fay | 11 |  
| 204 | Hermann | Baer | 12 |  
| 118 | Guy | Himuro | 13 |  
...2) Using SQL IN opeator with a subquery exampleA subquery is a query nested inside another query. Let’s take a look at an example:The following query returns the department id of the Marketing and Sales departments:SELECT   
 department\_id  
FROM  
 departments  
WHERE  
 department\_name = 'Marketing'  
 OR department\_name = 'Sales'Code language: SQL (Structured Query Language) (sql)Try It+---------------+  
| department\_id |  
+---------------+  
| 2 |  
| 8 |  
+---------------+The query returns a list of two department ids.And you can pass the id list to the IN operator to find employees who work in the Marketing and Sales departments like this:SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 department\_id  
FROM  
 employees  
WHERE  
 department\_id IN (2, 8);Code language: SQL (Structured Query Language) (sql)Try It+-------------+------------+------------+---------------+  
| employee\_id | first\_name | last\_name | department\_id |  
+-------------+------------+------------+---------------+  
| 145 | John | Russell | 8 |  
| 146 | Karen | Partners | 8 |  
| 176 | Jonathon | Taylor | 8 |  
| 177 | Jack | Livingston | 8 |  
| 178 | Kimberely | Grant | 8 |  
| 179 | Charles | Johnson | 8 |  
| 201 | Michael | Hartstein | 2 |  
| 202 | Pat | Fay | 2 |  
+-------------+------------+------------+---------------+  
To combine two above queries into a single query, you can use the first query in place of the list inside parentheses followed the IN operator:SELECT   
 employee\_id, first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 department\_id IN (SELECT   
 department\_id  
 FROM  
 departments  
 WHERE  
 department\_name = 'Marketing'  
 OR department\_name = 'Sales')Code language: SQL (Structured Query Language) (sql)Try ItSummaryUse the SQL IN operator to check if a value is in a set of values.Use the NOT opeator to negate the IN opeator, NOT IN.Use the `IN` with a subquery to combine two queries into a single query.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL BETWEEN Up NextSQL LIKE

# Content from https://www.sqltutorial.org/sql-like/

Home » SQL Tutorial » SQL LIKESQL LIKESummary: in this tutorial, you will learn how to use the SQL LIKE operator to test whether a value matches a pattern.Introduction to SQL LIKE operatorThe LIKE operator is one of the SQL logical operators. The LIKE operator returns true if a value matches a pattern or false otherwise.The syntax of the LIKE operator is as follows:expression LIKE patternCode language: SQL (Structured Query Language) (sql)In this syntax, the LIKE operator tests whether an expression matches the pattern. The SQL standard provides you with two wildcard characters to make a pattern: % percent wildcard matches zero, one, or more characters \_ underscore wildcard matches a single character.The following show an example of using the % and \_ wildcard characters:ExpressionMeaningLIKE 'Kim%'match a string that starts with KimLIKE '%er'match a string that ends with erLIKE '%ch%'match a string that contains chLIKE 'Le\_'match a string that starts with Le and is followed by one character e.g., Les, Len…LIKE '\_uy'match a string that ends with uy and is preceded by one character e.g., guyLIKE '%are\_'match a string that contains are and ends with one characterLIKE '\_are%'match a string that contains are, starts with one character, and ends with any number of charactersNote that besides the % and \_ wildcards, some database systems may have other wildcard characters that are specific to those databases.NOT LIKETo negate the LIKE operator, you use the NOT operator:expression NOT LIKE patternThe NOT LIKE operator returns true if the expression doesn’t match the pattern or false otherwise.Escape characterTo match a string that contains a wildcard for example 10%, you need to instruct the LIKE operator to treat the % in 10% as a regular character.To do that, you need to explicitly specify an escape character after the ESCAPE clause:expression LIKE pattern ESCAPE escape\_characterCode language: SQL (Structured Query Language) (sql)For example:value LIKE '%10!%%' ESCAPE '!'Code language: JavaScript (javascript)In this example, the ! is an escape character. It instructs the LIKE operator to treat the % in the 10% as a regular character.In practice, you often use the LIKE operator in  WHERE clause of the SELECT, UPDATE, and DELETE statements.SQL LIKE operator examplesWe’ll use the employees table in the sample database for the demonstration.The following example uses the LIKE operator to find all employees whose first names start with Da :SELECT  
 employee\_id,  
 first\_name,  
 last\_name  
FROM  
 employees  
WHERE  
 first\_name LIKE 'Da%';Code language: SQL (Structured Query Language) (sql)Try It+-------------+------------+-----------+  
| employee\_id | first\_name | last\_name |  
+-------------+------------+-----------+  
| 105 | David | Austin |  
| 109 | Daniel | Faviet |  
+-------------+------------+-----------+Code language: plaintext (plaintext)The following example use the LIKE operator to find all employees whose first names end with er:SELECT  
 employee\_id,  
 first\_name,  
 last\_name  
FROM  
 employees  
WHERE  
 first\_name LIKE '%er';Code language: SQL (Structured Query Language) (sql)Try It+-------------+------------+-----------+  
| employee\_id | first\_name | last\_name |  
+-------------+------------+-----------+  
| 103 | Alexander | Hunold |  
| 115 | Alexander | Khoo |  
| 200 | Jennifer | Whalen |  
+-------------+------------+-----------+Code language: plaintext (plaintext)The following example uses the LIKE operator to find employees whose last names contain the word an:SELECT  
 employee\_id,  
 first\_name,  
 last\_name  
FROM  
 employees  
WHERE  
 last\_name LIKE '%an%';Code language: SQL (Structured Query Language) (sql)Try It+-------------+-------------+-----------+  
| employee\_id | first\_name | last\_name |  
+-------------+-------------+-----------+  
| 102 | Lex | De Haan |  
| 112 | Jose Manuel | Urman |  
| 123 | Shanta | Vollman |  
| 178 | Kimberely | Grant |  
+-------------+-------------+-----------+Code language: plaintext (plaintext)The following statement retrieves employees whose first names start with Jo and are followed by at most 2 characters:SELECT  
 employee\_id,  
 first\_name,  
 last\_name  
FROM  
 employees  
WHERE  
 first\_name LIKE 'Jo\_\_';Code language: SQL (Structured Query Language) (sql)Try It+-------------+------------+-----------+  
| employee\_id | first\_name | last\_name |  
+-------------+------------+-----------+  
| 110 | John | Chen |  
| 145 | John | Russell |  
+-------------+------------+-----------+Code language: plaintext (plaintext)The following statement uses the LIKE operator with the % and \_ wildcard to find employees whose first names start with any number of characters and are followed by at most one character:SELECT  
 employee\_id,  
 first\_name,  
 last\_name  
FROM  
 employees  
WHERE  
 first\_name LIKE '%are\_';Code language: SQL (Structured Query Language) (sql)Try It+-------------+------------+------------+  
| employee\_id | first\_name | last\_name |  
+-------------+------------+------------+  
| 119 | Karen | Colmenares |  
| 146 | Karen | Partners |  
+-------------+------------+------------+SQL NOT LIKE operator exampleThe following example uses the NOT LIKE operator to find all employees whose first names start with the letter S but not start with Sh:SELECT  
 employee\_id,  
 first\_name,  
 last\_name  
FROM  
 employees  
WHERE  
 first\_name LIKE 'S%'  
AND first\_name NOT LIKE 'Sh%'  
ORDER BY  
 first\_name;Code language: SQL (Structured Query Language) (sql)Try It+-------------+------------+-----------+  
| employee\_id | first\_name | last\_name |  
+-------------+------------+-----------+  
| 192 | Sarah | Bell |  
| 117 | Sigal | Tobias |  
| 100 | Steven | King |  
| 203 | Susan | Mavris |  
+-------------+------------+-----------+Code language: plaintext (plaintext)SummaryThe LIKE operator returns true if a value matches a pattern or false otherwse.Use the NOT operator to negate the LIK opeator.Use the % wildcard to match one or more charactersUse the \_ wildcard to match a single character.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL IN Up NextSQL IS NULL

# Content from https://www.sqltutorial.org/sql-is-null/

Home » SQL Tutorial » SQL IS NULLSQL IS NULLSummary: in this tutorial, you learn about the NULL and how to use the SQL  IS NULL and IS NOT NULL operators to test if an expression is NULL or not.What is NULLNULL is special in SQL. NULL indicates that the data is unknown, inapplicable, or even does not exist. In other words, NULL represents the missing data in the database.For example, if employees do not have phone numbers, you can store their phone numbers as empty strings.However, if you don’t know their phone numbers when you save the employee records, you need to use the NULL for the unknown phone numbers.The NULL is special because any comparisons with a NULL can never result in true or false, but in a third logical result, unknown.The following statement returns NULL:SELECT NULL = 5;Code language: PHP (php)Try ItThe NULL value is not even equal to itself, as shown in the following statement:SELECT NULL = NULL;Code language: SQL (Structured Query Language) (sql)Try ItIn this example, the result is NULL.You cannot use the comparison operator equal to (=) to compare a value to a NULL value. For example, the following statement will not return the correct result:SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 phone\_number  
FROM  
 employees  
WHERE  
 phone\_number = NULL;Code language: SQL (Structured Query Language) (sql)Try ItThe IS NULL and IS NOT NULL operatorsTo determine whether an expression or column is NULL or not, you use the IS NULL operator as follows:expression IS NULLCode language: SQL (Structured Query Language) (sql)If the result of the expression is NULL, IS NULL operator returns true; otherwise, it returns false.To check if an expression or column is not NULL, you use the IS NOT operator:expression IS NOT NULLCode language: PHP (php)The IS NOT NULL returns false if the value of the expression is NULl; otherwise, it returns true;SQL IS NULL and IS NOT NULL examplesIn these examples, we’ll use the  employees table from the sample database for the demonstration.To find all employees who do not have the phone numbers, you use the IS NULL operator as follows:SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 phone\_number  
FROM  
 employees  
WHERE  
 phone\_number IS NULL;Code language: PHP (php)Try ItTo find all employees who have phone numbers, you use IS NOT NULL as shown in the following statement:SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 phone\_number  
FROM  
 employees  
WHERE  
 phone\_number IS NOT NULL;Code language: SQL (Structured Query Language) (sql)Try ItNow you should understand the NULL concept and know how to use the SQL IS operator to check whether a value is NULL or notWas this tutorial helpful ? Yes No Send Cancel PreviouslySQL LIKE Up NextSQL NOT

# Content from https://www.sqltutorial.org/sql-not/

Home » SQL Tutorial » SQL NOTSQL NOTSummary: in this tutorial, you will learn how to use the SQL NOT operator to negate a Boolean expression in the WHERE clause of the SELECT statement.Introduction to the SQL NOT operatorYou have learned how to use various logical operators such as AND, OR, LIKE, BETWEEN, IN, and EXISTS. These operators help you to form flexible conditions in the WHERE clause.To negate the result of any Boolean expression, you use the NOT operator. The following illustrates how to use the NOT operator:NOT [Boolean\_expression]Code language: SQL (Structured Query Language) (sql)The following table shows the result of the NOT operator.NOTTRUEFALSEFALSETRUENULLNULLSQL NOT operator examplesWe’ll use the employees table to help better you understand the NOT operator.The following statement retrieves all employees who work in the department id 5.SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 salary  
FROM  
 employees  
WHERE  
 department\_id = 5  
ORDER BY  
 salary;  
Code language: SQL (Structured Query Language) (sql)Try ItTo get the employees who work in the department id 5 and with a salary not greater than 5000.SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 salary  
FROM  
 employees  
WHERE  
 department\_id = 5  
AND NOT salary > 5000  
ORDER BY  
 salary;Code language: SQL (Structured Query Language) (sql)Try ItSQL NOT with IN operator exampleTo negate the IN operator, you use the NOT operator. For example, the following statement gets all the employees who are not working in the departments 1, 2, or 3.SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 department\_id  
FROM  
 employees  
WHERE  
 department\_id NOT IN (1, 2, 3)  
ORDER BY  
 first\_name;Code language: SQL (Structured Query Language) (sql)Try ItSQL NOT LIKE operator exampleYou can negate the LIKE operator by using the NOT LIKE. For example, the following statement retrieves all the employees whose first names do not start with the letter D.SELECT  
 first\_name,  
 last\_name  
FROM  
 employees  
WHERE  
 first\_name NOT LIKE 'D%'  
ORDER BY  
 first\_name;Code language: SQL (Structured Query Language) (sql)Try ItSQL NOT BETWEEN exampleThe following example shows you how to use the NOT to negate the BETWEEN operator to get employees whose salaries are not between 5,000 and 1,000.SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 salary  
FROM  
 employees  
WHERE  
 salary NOT BETWEEN 3000  
AND 5000  
ORDER BY  
 salary;Code language: SQL (Structured Query Language) (sql)Try ItSQL NOT EXISTS exampleSee the following employees and dependents tables:The following query uses the NOT EXISTS operator to get the employees who do not have any dependents.SELECT  
 employee\_id,  
 first\_name,  
 last\_name  
FROM  
 employees e  
WHERE  
 NOT EXISTS (  
 SELECT  
 employee\_id  
 FROM  
 dependents d  
 WHERE  
 d.employee\_id = e.employee\_id  
 );Code language: SQL (Structured Query Language) (sql)Try ItNow you should know how to use the NOT operator to negate a Boolean expression.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL IS NULL Up NextSQL Alias

# Content from https://www.sqltutorial.org/sql-case/

Home » SQL Tutorial » SQL CASESQL CASESummary: in this tutorial, you will learn how to use the SQL CASE expression to add the logic to the SQL statements.Introduction to SQL CASE expressionThe SQL CASE expression allows you to evaluate a list of conditions and returns one of the possible results. The CASE expression has two formats: simple CASE and searched CASE.You can use the CASE expression in a clause or statement that allows a valid expression. For example, you can use the CASE expression in statements such as SELECT, DELETE, and UPDATE or in clauses such as SELECT, ORDER BY, and HAVING.Simple CASE expressionThe following illustrates the simple CASE expression:CASE expression  
WHEN when\_expression\_1 THEN  
 result\_1  
WHEN when\_expression\_2 THEN  
 result\_2  
WHEN when\_expression\_3 THEN  
 result\_3  
...  
ELSE  
 else\_result  
ENDCode language: SQL (Structured Query Language) (sql)The CASE expression compares an expression to a set of expression (when\_expression\_1, when\_expression\_2, when\_expression\_3, …) using the equality operator (=). If you want to use other comparison operators such as greater than (>), less than (<), etc., you use the searched CASE expression.The CASE statement returns the result\_1, result\_2, or result\_3 if the expression matches the corresponding expression in the WHEN clause.If the expression does not match any expression in the WHEN clause, it returns the esle\_result in the ELSE clause. The ELSE clause is optional.If you omit the ELSE clause and the expression does not match any expression in the WHEN clause, the CASE expression returns NULL.Simple CASE expression exampleLet’s take a look at the employees table.Suppose the current year is 2000.We can use the simple CASE expression to get the work anniversaries of employees by using the following statement:SELECT   
 first\_name,  
 last\_name,  
 hire\_date,  
 CASE (2000 - YEAR(hire\_date))  
 WHEN 1 THEN '1 year'  
 WHEN 3 THEN '3 years'  
 WHEN 5 THEN '5 years'  
 WHEN 10 THEN '10 years'  
 WHEN 15 THEN '15 years'  
 WHEN 20 THEN '20 years'  
 WHEN 25 THEN '25 years'  
 WHEN 30 THEN '30 years'  
 END aniversary  
FROM  
 employees  
ORDER BY first\_name;Code language: SQL (Structured Query Language) (sql)Try ItThe YEAR function returns the year when the employee joined the company. We get the number of years that the employee has been with the company and by subtracting the year when the employee joined the company from the current year (2000).We get the number of years that the employee has been with the company by subtracting the year when the employee joined the company from the current year (2000).Then we compare the result with 1, 3, 5, 10, 15, 20, 25, 30 If the year of service equals one of these numbers, the CASE expression returns the work anniversary of the employee.If the year of services of the employee does not match these numbers, the CASE expression returns NULL.Searched CASE expressionThe following shows the searched CASE expression.CASE  
WHEN boolean\_expression\_1 THEN  
 result\_1  
WHEN boolean\_expression\_2 THEN  
 result\_2  
WHEN boolean\_expression\_3 THEN  
 result\_3  
ELSE  
 else\_result  
END;Code language: SQL (Structured Query Language) (sql)The database system evaluates the boolean expression for each WHEN clause in the order specified in the CASE expression.If the Boolean expression in each WHEN clause evaluates to true, the searched CASE statement returns the result in the corresponding THEN clause.If no Boolean expression returns true, the CASE expression return the result else\_result in the ELSE clause.Like the simple CASE expression, the END clause is optional. If you omit the ELSE clause and no Boolean expression evaluates to true, the CASE expression returns a NULL value.Search CASE expression exampleThe following illustrates the searched CASE expression example.SELECT   
 first\_name,  
 last\_name,  
 CASE  
 WHEN salary < 3000 THEN 'Low'  
 WHEN salary >= 3000 AND salary <= 5000 THEN 'Average'  
 WHEN salary > 5000 THEN 'High'  
 END evaluation  
FROM  
 employees;Code language: SQL (Structured Query Language) (sql)Try ItIf the salary is less than 3000, the CASE expression returns “Low”. If the salary is between 3000 and 5000, it returns “average”. When the salary is greater than 5000, the CASE expression returns “High”.In this tutorial, we have introduced you to the SQL CASE statement that allows you to add the IF THEN ELSE logic to the SQL statements.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL DELETE Up NextSQL Data Types

# Content from https://www.sqltutorial.org/sql-alias/

Home » SQL Tutorial » SQL AliasSQL AliasSummary: in this tutorial, you will learn about SQL aliases including table and column aliases to make your queries shorter and more understandable.SQL alias allows you to assign a table or a column a temporary name during the execution of a query. SQL has two types of aliases: table and column aliases.SQL column aliasesWhen designing database tables, you may use abbreviations for the column names to keep them short. For example:The so\_no stands for sales order number.The qty stands for quantity.Or you may have to work with legacy systems that aggressively use abbreviations for naming columns and tables.For example, SAP ERP uses abbreviations (in German) to name all the columns and tables such as VBELN for naming sales document number column.When you use SQL to query data from these tables, the output is not obvious. To fix it, you can use column aliases that give columns temporary names during the execution of the query.The following shows the syntax of using column aliases:column\_name AS alias\_nameCode language: SQL (Structured Query Language) (sql)In this syntax, you specify the column alias after the AS keyword followed by the column name. The AS keyword is optional. So you can omit it like this:column\_name alias\_nameCode language: SQL (Structured Query Language) (sql)If the alias contains spaces, you need to put it inside single quotes (or double quotes) like this:column\_name AS 'Alias Name'Code language: SQL (Structured Query Language) (sql)The following example shows how to use the column aliases:SELECT  
 inv\_no AS invoice\_no,  
 amount,  
 due\_date AS 'Due date',  
 cust\_no 'Customer No'  
FROM  
 invoices;Code language: SQL (Structured Query Language) (sql)This query has multiple column aliases:The invoice\_no is the column alias of the inv\_no column.The 'Due date' is the column alias of the due\_date column. Because the alias contains space, you have to place it inside single quote (‘) or double quotes (“) .The 'Customer no' is the alias of the cust\_no column. Note that it doesn’t has the AS keyword.Aliases for expressionsIf a query contains expressions, you can assign column aliases to the expressions. For example:SELECT   
 first\_name,   
 last\_name,   
 salary \* 1.1 AS new\_salary  
FROM  
 employees;Code language: SQL (Structured Query Language) (sql)In this example, the database will use the column aliases as the heading of the expressions in the result set.Common mistakes of column aliasesSince you assign aliases to columns in the SELECT clause, you can only reference the aliases in the clauses that are evaluated after the SELECT clause.The following query will result in an error:SELECT   
 first\_name,   
 last\_name,   
 salary \* 1.1 AS new\_salary  
FROM  
 employees  
WHERE new\_salary > 5000Code language: SQL (Structured Query Language) (sql)Error:Unknown column 'new\_salary' in 'where clause'Code language: JavaScript (javascript)Why?In this SELECT statement, the database evaluates the clauses in the following order:FROM > WHERE > SELECTCode language: SQL (Structured Query Language) (sql)The database evaluates the WHERE clause before the SELECT clause. Therefore, at the time it evaluates the WHERE clause, the database doesn’t have the information of the new\_salary column alias. So it issued an error.However, the following query works correctly:SELECT   
 first\_name,   
 last\_name,   
 salary \* 1.1 AS new\_salary  
FROM  
 employees  
ORDER BY new\_salary;Code language: SQL (Structured Query Language) (sql)In this example, the database evaluates the clauses of the query in the following order:FROM > SELECT > ORDER BYCode language: SQL (Structured Query Language) (sql)The database evaluates the SELECT clause before the ORDER BY clause. Therefore, at the time of evaluating the ORDER BY clause, the database has the information of the new\_salary alias, which is the alias of the expression salary \* 1.1. Hence, it works as expected.SQL table aliasesLike columns, you can assign aliases to tables. And these aliases are called table aliases.To assign an alias to a table, you use the following syntax:table\_name AS table\_aliasCode language: PHP (php)In this syntax, the AS keyword is also optional. So you can omit it like the following:table\_name table\_aliasCode language: SQL (Structured Query Language) (sql)Notice that assigning an alias to a table does not rename the table permanently. It just gives the table another name temporarily during the execution of a query.So why do you need table aliases?When specifying the column names in the SELECT clause, you can use the following syntax:table\_name.column\_nameCode language: CSS (css)In this syntax, the column has a fully qualified name that includes both table and column names. For example:SELECT   
 employees.first\_name,   
 employees.last\_name  
FROM  
 employees;Code language: SQL (Structured Query Language) (sql)In this example, the query selects the data from the first\_name and last\_name columns of the employees table. Unlike a typical query, we specify the fully qualified names for the first\_name and last\_name columns.In the SELECT clause, instead of using the employees table name, you can use the table alias of the employees table. For example:SELECT   
 e.first\_name,   
 e.last\_name  
FROM  
 employees AS e;Code language: SQL (Structured Query Language) (sql)In this example, we assign e table alias to the employees in the FROM clause and reference it in the SELECT clause.But why do you need to use the fully qualified names for columns?See the following employees and departments tables from the sample database:Both employees and departments tables have the columns with the same name: department\_id.When querying data from both tables, you need to specify the exact table that the department\_id column belongs to. Otherwise, you’ll get an error because the database doesn’t know which table it needs to go to select data.If you want to select data from the employees table, you can reference it in the SELECT clause as follows:employees.department\_idCode language: SQL (Structured Query Language) (sql)The same is applied to the department\_id of the departments table:departments.department\_idCode language: SQL (Structured Query Language) (sql)If e and d are table aliases of the employees and departments tables, you can reference the department\_id column in each table using the table aliases as follows:e.department\_id  
d.department\_idCode language: SQL (Structured Query Language) (sql)In the next tutorial, you’ll learn how to use the join clauses to select data from both tables and apply the table aliases. Also, you’ll learn how to reference the same table in a single query twice using the self-join technique. In this case, you need to use the table aliases.SummarySQL has two type of aliases: column and table aliases.Reference the column aliases in the clauses that are evaluated after the SELECT clause.Use table aliases to qualify the column names.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL NOT Up NextSQL INNER JOIN

# Content from https://www.sqltutorial.org/sql-inner-join/

Home » SQL Tutorial » SQL INNER JOINSQL INNER JOINSummary: in this tutorial, we will show you how to use the SQL INNER JOIN clause to query data from two or more tables.Introduction to the SQL INNER JOIN clauseSo far, you have learned how to use the SELECT statement to query data from a single table. However, the SELECT statement is not limited to query data from a single table. The SELECT statement can link multiple tables together.The process of linking tables is called joining. SQL provides many kinds of joins such as inner join, left join, right join, full outer join, etc. This tutorial focuses on the inner join.Suppose, you have two tables: A and B.Table A has four rows: (1,2,3,4) and table B has four rows: (3,4,5,6)When table A joins with table B using the inner join, you have the result set (3,4) that is the intersection of table A and table B.See the following picture.For each row in table A, the inner join clause finds the matching rows in table B. If a row is matched, it is included in the final result set.Suppose the columns in the A and B tables are a and b. The following statement illustrates the inner join clause:SELECT a  
FROM A  
INNER JOIN B ON b = a;Code language: SQL (Structured Query Language) (sql)The INNER JOIN clause appears after the FROM clause. The condition to match between table A and table B is specified after the ON keyword. This condition is called join condition i.e., B.n = A.nThe INNER JOIN clause can join three or more tables as long as they have relationships, typically foreign key relationships.For example, the following statement illustrates how to join 3 tables: A, B, and C:SELECT  
 A.n  
FROM A  
INNER JOIN B ON B.n = A.n  
INNER JOIN C ON C.n = A.n;Code language: SQL (Structured Query Language) (sql)SQL INNER JOIN examplesLet’s take some practical examples of using the INNER JOIN clause.1) Using SQL INNER JOIN to join two tables exampleWe will use the employees and departments tables from the sample database to demonstrate how the INNER JOIN clause works.Each employee belongs to one and only one department while each department can have more than one employee. The relationship between the departments and employees is one-to-many.The department\_id column in the employees table is the foreign key column that links the employees to the departments table.To get the information of the department id 1,2, and 3, you use the following statement.SELECT  
 department\_id,  
 department\_name  
FROM  
 departments  
WHERE  
 department\_id IN (1, 2, 3);Code language: SQL (Structured Query Language) (sql)Try ItNotice that we used the IN operator in the WHERE clause to get rows with department\_id 1, 2 and 3.To get the information of employees who work in the department id 1, 2 and 3, you use the following query:SELECT  
 first\_name,  
 last\_name,  
 department\_id  
FROM  
 employees  
WHERE  
 department\_id IN (1, 2, 3)  
ORDER BY  
 department\_id;Code language: SQL (Structured Query Language) (sql)Try ItTo combine data from these two tables, you use an inner join clause as the following query:SELECT   
 first\_name,  
 last\_name,  
 employees.department\_id,  
 departments.department\_id,  
 department\_name  
FROM  
 employees  
 INNER JOIN  
 departments ON departments.department\_id = employees.department\_id  
WHERE  
 employees.department\_id IN (1 , 2, 3);Code language: SQL (Structured Query Language) (sql)Try ItFor each row in the employees table, the statement checks if the value of the department\_id column equals the value of the department\_id column in the departments table. If the conditionIf the condition employees.department\_id = departments.department\_id is satisfied, the combined row that includes data from rows in both employees and departments tables are included in the result set.Notice that both employees and departments tables have the same column name department\_id, therefore we had to qualify the department\_id column using the syntax table\_name.column\_name.SQL INNER JOIN 3 tables exampleEach employee holds one job while a job may be held by many employees. The relationship between the jobs table and the employees table is one-to-many.The following database diagram illustrates the relationships between employees, departments and jobs tables:The following query uses the inner join clauses to join 3 tables: employees, departments, and jobs to get the first name, last name, job title, and department name of employees who work in department id 1, 2, and 3.SELECT  
 first\_name,  
 last\_name,  
 job\_title,  
 department\_name  
FROM  
 employees e  
INNER JOIN departments d ON d.department\_id = e.department\_id  
INNER JOIN jobs j ON j.job\_id = e.job\_id  
WHERE  
 e.department\_id IN (1, 2, 3);Code language: SQL (Structured Query Language) (sql)Try ItNow you should understand how the SQL INNER JOIN clause works and know how to apply it to query data from multiple tables.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL Alias Up NextSQL LEFT JOIN

# Content from https://www.sqltutorial.org/sql-left-join/

Home » SQL Tutorial » SQL LEFT JOINSQL LEFT JOINSummary: in this tutorial, we will introduce you another kind of joins called SQL LEFT JOIN that allows you to retrieve data from multiple tables.Introduction to SQL LEFT JOIN clauseIn the previous tutorial, you learned about the inner join that returns rows if there is, at least, one row in both tables that matches the join condition. The inner join clause eliminates the rows that do not match with a row of the other table.The left join, however, returns all rows from the left table whether or not there is a matching row in the right table.Suppose we have two tables A and B. The table A has four rows 1, 2, 3 and 4. The table B also has four rows 3, 4, 5, 6.When we join table A with table B, all the rows in table A (the left table) are included in the result set whether there is a matching row in the table B or not.In SQL, we use the following syntax to join table A with table B.SELECT  
 A.n  
FROM  
 A  
LEFT JOIN B ON B.n = A.n;Code language: SQL (Structured Query Language) (sql)The LEFT JOIN clause appears after the FROM clause. The condition that follows the ON keyword is called the join condition B.n = A.nSQL LEFT JOIN examplesSQL LEFT JOIN two tables examplesLet’s take a look at the countries and locations tables.Each location belongs to one and only one country while each country can have zero or more locations. The relationship between the countries and locations tables is one-to-many.The country\_id column in the locations table is the foreign key that links to the country\_id column in the countries table.To query the country names of US, UK, and China, you use the following statement.SELECT  
 country\_id,  
 country\_name  
FROM  
 countries  
WHERE  
 country\_id IN ('US', 'UK', 'CN');Code language: SQL (Structured Query Language) (sql)Try ItThe following query retrieves the locations located in the US, UK and China:SELECT  
 country\_id,  
 street\_address,  
 city  
FROM  
 locations  
WHERE  
 country\_id IN ('US', 'UK', 'CN');Code language: SQL (Structured Query Language) (sql)Try ItNow, we use the LEFT JOIN clause to join the countries table with the locations table as the following query:SELECT  
 c.country\_name,  
 c.country\_id,  
 l.country\_id,  
 l.street\_address,  
 l.city  
FROM  
 countries c  
LEFT JOIN locations l ON l.country\_id = c.country\_id  
WHERE  
 c.country\_id IN ('US', 'UK', 'CN')Code language: SQL (Structured Query Language) (sql)Try ItThe condition in the WHERE clause is applied so that the statement only retrieves the data from the US, UK, and China rows.Because we use the LEFT JOIN clause, all rows that satisfy the condition in the WHERE clause of the countries table are included in the result set.For each row in the countries table, the LEFT JOIN clause finds the matching rows in the locations table.If at least one matching row found, the database engine combines the data from columns of the matching rows in both tables.In case there is no matching row found e.g., with the country\_id CN, the row in the countries table is included in the result set and the row in the locations table is filled with NULL values.Because non-matching rows in the right table are filled with the NULL values, you can apply the LEFT JOIN clause to miss-match rows between tables.For example, to find the country that does not have any locations in the locations table, you use the following query:SELECT  
 country\_name  
FROM  
 countries c  
LEFT JOIN locations l ON l.country\_id = c.country\_id  
WHERE  
 l.location\_id IS NULL  
ORDER BY  
 country\_name;Code language: SQL (Structured Query Language) (sql)Try ItSQL LEFT JOIN 3 tables exampleSee the following tables: regions, countries, and locations.One region may have zero or many countries while each country is located in the one region. The relationship between countries and regions tables is one-to-many. The region\_id column in the countries table is the link between the countries and regions table.The following statement demonstrates how to join 3 tables: regions, countries, and locations:SELECT  
 r.region\_name,  
 c.country\_name,  
 l.street\_address,  
 l.city  
FROM  
 regions r  
LEFT JOIN countries c ON c.region\_id = r.region\_id  
LEFT JOIN locations l ON l.country\_id = c.country\_id  
WHERE  
 c.country\_id IN ('US', 'UK', 'CN');Code language: SQL (Structured Query Language) (sql)Try ItNow you should have a good understanding of how the SQL LEFT JOIN clause works and know how to apply the LEFT JOIN clause to query data from multiple tables.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL INNER JOIN Up NextSQL SELF JOIN

# Content from https://www.sqltutorial.org/sql-full-outer-join/

Home » SQL Tutorial » SQL FULL OUTER JOINSQL FULL OUTER JOINSummary: in this tutorial, you will learn how to use SQL FULL OUTER JOIN clause to query data from multiple tables.Introduction to SQL FULL OUTER JOIN clauseIn theory, a full outer join is the combination of a left join and a right join. The full outer join includes all rows from the joined tables whether or not the other table has the matching row.If the rows in the joined tables do not match, the result set of the full outer join contains NULL values for every column of the table that lacks a matching row. For the matching rows, a single row that has the columns populated from the joined table is included in the result set.The following statement illustrates the syntax of the full outer join of two tables:SELECT column\_list  
FROM A  
FULL OUTER JOIN B ON B.n = A.n;Code language: SQL (Structured Query Language) (sql)Note that the OUTER keyword is optional.The following Venn diagram illustrates the full outer join of two tables.SQL FULL OUTER JOIN examplesLet’s take an example of using the FULL OUTER JOIN clause to see how it works.First, create two new tables: baskets and fruits for the demonstration. Each basket stores zero or more fruits and each fruit can be stored in zero or one basket.CREATE TABLE fruits (  
 fruit\_id INTEGER PRIMARY KEY,  
 fruit\_name VARCHAR (255) NOT NULL,  
 basket\_id INTEGER  
);Code language: SQL (Structured Query Language) (sql)CREATE TABLE baskets (  
 basket\_id INTEGER PRIMARY KEY,  
 basket\_name VARCHAR (255) NOT NULL  
);Code language: SQL (Structured Query Language) (sql)Second, insert some sample data into the baskets and fruits tables.INSERT INTO baskets (basket\_id, basket\_name)  
VALUES  
 (1, 'A'),  
 (2, 'B'),  
 (3, 'C');  
Code language: SQL (Structured Query Language) (sql)INSERT INTO fruits (  
 fruit\_id,  
 fruit\_name,  
 basket\_id  
)  
VALUES  
 (1, 'Apple', 1),  
 (2, 'Orange', 1),  
 (3, 'Banana', 2),  
 (4, 'Strawberry', NULL);Code language: SQL (Structured Query Language) (sql)Third, the following query returns each fruit that is in a basket and each basket that has a fruit, but also returns each fruit that is not in any basket and each basket that does not have any fruit.SELECT  
 basket\_name,  
 fruit\_name  
FROM  
 fruits  
FULL OUTER JOIN baskets ON baskets.basket\_id = fruits.basket\_id;Code language: SQL (Structured Query Language) (sql) basket\_name | fruit\_name  
-------------+------------  
 A | Apple  
 A | Orange  
 B | Banana  
 (null) | Strawberry  
 C | (null)Code language: SQL (Structured Query Language) (sql)As you see, the basket C does not have any fruit and the Strawberry is not in any basket.You can add a WHERE clause to the statement that uses the FULL OUTER JOIN clause to get more specific information.For example, to find the empty basket, which does not store any fruit, you use the following statement:SELECT  
 basket\_name,  
 fruit\_name  
FROM  
 fruits  
FULL OUTER JOIN baskets ON baskets.basket\_id = fruits.basket\_id  
WHERE  
 fruit\_name IS NULL;Code language: SQL (Structured Query Language) (sql) basket\_name | fruit\_name  
-------------+------------  
 C | (null)  
(1 row)Code language: SQL (Structured Query Language) (sql)Similarly, if you want to see which fruit is not in any basket, you use the following statement:SELECT  
 basket\_name,  
 fruit\_name  
FROM  
 fruits  
FULL OUTER JOIN baskets ON baskets.basket\_id = fruits.basket\_id  
WHERE  
 basket\_name IS NULL;Code language: SQL (Structured Query Language) (sql) basket\_name | fruit\_name  
-------------+------------  
(null) | Strawberry  
(1 row)Code language: SQL (Structured Query Language) (sql)In this tutorial, we have shown you how to use the SQL FULL OUTER JOIN clause to query data from multiple tables.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL SELF JOIN Up NextSQL CROSS JOIN

# Content from https://www.sqltutorial.org/sql-cross-join/

Home » SQL Tutorial » SQL CROSS JOINSQL CROSS JOINSummary: this tutorial shows you how to use the SQL CROSS JOIN to make a Cartesian product of the joined tables.Introduction to SQL CROSS JOIN clauseA cross join is a join operation that produces the Cartesian product of two or more tables.In Math, a Cartesian product is a mathematical operation that returns a product set of multiple sets.For example, with two sets A {x,y,z} and B {1,2,3}, the Cartesian product of A x B is the set of all ordered pairs (x,1), (x,2), (x,3), (y,1) (y,2), (y,3), (z,1), (z,2), (z,3).The following picture illustrates the Cartesian product of A and B:Similarly, in SQL, a Cartesian product of two tables A and B is a result set in which each row in the first table (A) is paired with each row in the second table (B). Suppose the A table has n rows and the B table has m rows, the result of the cross join of the A and B tables have n x m rows.The following illustrates syntax of the CROSS JOIN clause:SELECT column\_list  
FROM A  
CROSS JOIN B;Code language: SQL (Structured Query Language) (sql)The following picture illustrates the result of the cross join between the table A and table B. In this illustration, the table A has three rows 1, 2 and 3 and the table B also has three rows x, y and z. As the result, the Cartesian product has nine rows:Note that unlike the INNER JOIN, LEFT JOIN, and FULL OUTER JOIN, the CROSS JOIN clause does not have a join condition.The following statement is equivalent to the one that uses the CROSS JOIN clause above:SELECT   
 column\_list  
FROM  
 A,  
 B;Code language: SQL (Structured Query Language) (sql)SQL CROSS JOIN exampleWe will create two new tables  for the demonstration of the cross join: sales\_organization table stores the sale organizations. sales\_channel table stores the sales channels.The following statements create the sales\_organization and sales\_channel tables:CREATE TABLE sales\_organization (  
 sales\_org\_id INT PRIMARY KEY,  
 sales\_org VARCHAR (255)  
);Code language: SQL (Structured Query Language) (sql)CREATE TABLE sales\_channel (  
 channel\_id INT PRIMARY KEY,  
 channel VARCHAR (255)  
);Code language: SQL (Structured Query Language) (sql)Suppose the company has two sales organizations that are Domestic and Export, which are in charge of sales in the domestic and international markets.The following statement inserts two sales organizations into the sales\_organization table:INSERT INTO sales\_organization (sales\_org\_id, sales\_org)  
VALUES  
 (1, 'Domestic'),  
 (2, 'Export');Code language: SQL (Structured Query Language) (sql)The company can distribute goods via various channels such as wholesale, retail, eCommerce, and TV shopping. The following statement inserts sales channels into the sales\_channel table:INSERT INTO sales\_channel (channel\_id, channel)  
VALUES  
 (1, 'Wholesale'),  
 (2, 'Retail'),  
 (3, 'eCommerce'),  
 (4, 'TV Shopping');Code language: SQL (Structured Query Language) (sql)To find the all possible sales channels that a sales organization can have, you use the CROSS JOIN to join the sales\_organization table with the sales\_channel table as follows:SELECT  
 sales\_org,  
 channel  
FROM  
 sales\_organization  
CROSS JOIN sales\_channel;   
Code language: SQL (Structured Query Language) (sql)Here is the result set:The result set includes all possible rows in the sales\_organization and sales\_channel tables.The following query is equivalent to the statement that uses the CROSS JOIN clause above:SELECT  
 sales\_org,  
 channel  
FROM  
 sales\_organization,  
 sales\_channel;Code language: SQL (Structured Query Language) (sql)In some database systems such as PostgreSQL and Oracle, you can use the INNER JOIN clause with the condition that always evaluates to true to perform a cross join such as:SELECT  
 sales\_org,  
 channel  
FROM  
 sales\_organization  
INNER JOIN sales\_channel ON 1 = 1;Code language: SQL (Structured Query Language) (sql)In this tutorial, you have learned how to use the SQL CROSS JOIN clause to produce a Cartesian product of two or more tables.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL FULL OUTER JOIN Up NextSQL GROUP BY

# Content from https://www.sqltutorial.org/sql-self-join/

Home » SQL Tutorial » SQL SELF JOINSQL SELF JOINSummary: this tutorial shows you how to join table to itself by using the SQL self-join technique.Introduction to SQL self-joinSometimes, it is useful to join a table to itself. This type of join is known as the self-join.We join a table to itself to evaluate the rows with other rows in the same table. To perform the self-join, we use either an inner join or left join clause.Because the same table appears twice in a single query, we have to use the table aliases. The following statement illustrates how to join a table to itself.SELECT  
 column1,  
 column2,  
 column3,  
 ...  
FROM  
 table1 A  
INNER JOIN table1 B ON B.column1 = A.column2;Code language: SQL (Structured Query Language) (sql)In this statement joins the table1 to itself using an INNER JOIN clause. A and B are the table aliases of the table1. The B.column1 = A.column2 is the join condition.Besides the INNER JOIN clause, you can use the LEFT JOIN clause.Let’s take few examples of using the self-join technique.SQL self-join examplesSee the following employees table.The manager\_id column specifies the manager of an employee. The following statement joins the employees table to itself to query the information of who reports to whom.SELECT   
 e.first\_name || ' ' || e.last\_name AS employee,  
 m.first\_name || ' ' || m.last\_name AS manager  
FROM  
 employees e  
 INNER JOIN  
 employees m ON m.employee\_id = e.manager\_id  
ORDER BY manager;Code language: SQL (Structured Query Language) (sql)Try ItThe president does not have any manager. In the employees table, the manager\_id of the row that contains the president is NULL.Because the inner join clause only includes the rows that have matching rows in the other table, therefore the president did not show up in the result set of the query above.To include the president in the result set, we use the LEFT JOIN clause instead of the INNER JOIN clause as the following query.SELECT   
 e.first\_name || ' ' || e.last\_name AS employee,  
 m.first\_name || ' ' || m.last\_name AS manager  
FROM  
 employees e  
 LEFT JOIN  
 employees m ON m.employee\_id = e.manager\_id  
ORDER BY manager;Code language: SQL (Structured Query Language) (sql)Try ItIn this tutorial, you have learned how to use the INNER JOIN or LEFT JOIN clause to join table to itself.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL LEFT JOIN Up NextSQL FULL OUTER JOIN

# Content from https://www.sqltutorial.org/sql-aggregate-functions/

Home » SQL Aggregate FunctionsSQL Aggregate FunctionsSummary: in this tutorial, you will learn about the SQL aggregate functions including AVG(), COUNT(), MIN(), MAX(), and SUM().An SQL aggregate function calculates on a set of values and returns a single value. For example, the average function ( AVG) takes a list of values and returns the average.Because an aggregate function operates on a set of values, it is often used with the GROUP BY clause of the SELECT statement. The GROUP BY clause divides the result set into groups of values and the aggregate function returns a single value for each group.The following illustrates how the aggregate function is used with the GROUP BY clause:SELECT c1, aggregate\_function(c2)  
FROM table  
GROUP BY c1;Code language: SQL (Structured Query Language) (sql)The following are the commonly used SQL aggregate functions: AVG() – returns the average of a set. COUNT() – returns the number of items in a set. MAX() – returns the maximum value in a set. MIN() – returns the minimum value in a set SUM() – returns the sum of all or distinct values in a setExcept for the COUNT() function, SQL aggregate functions ignore null.You can use aggregate functions as expressions only in the following:The select list of a SELECT statement, either a subquery or an outer query.A HAVING clause AVGThe AVG() function returns the average values in a set. The following illustrates the syntax of the AVG() function:AVG( ALL | DISTINCT)Code language: SQL (Structured Query Language) (sql)The ALL keyword instructs the AVG() function to calculate the average of all values while the DISTINCT keyword forces the function to operate on distinct values only. By default, the ALL option is used.The following example shows how to use the AVG() function to calculate the average salary of each department:SELECT   
 department\_name, ROUND(AVG(salary), 0) avg\_salary  
FROM  
 employees  
 INNER JOIN  
 departments USING (department\_id)  
GROUP BY department\_name  
ORDER BY department\_name;Code language: SQL (Structured Query Language) (sql) MINThe MIN() function returns the minimum value of a set. The following illustrates the syntax of the MIN() function:MIN(column | expression)Code language: SQL (Structured Query Language) (sql)For example, the following statement returns the minimum salary of the employees in each department:SELECT   
 department\_name, MIN(salary) min\_salary  
FROM  
 employees  
 INNER JOIN  
 departments USING (department\_id)  
GROUP BY department\_name  
ORDER BY department\_name;Code language: SQL (Structured Query Language) (sql) MAXThe MAX() function returns the maximum value of a set. The MAX() function has the following syntax:MAX(column | expression)Code language: SQL (Structured Query Language) (sql)For example, the following statement returns the highest salary of employees in each department:SELECT   
 department\_name, MAX(salary) highest\_salary  
FROM  
 employees  
 INNER JOIN  
 departments USING (department\_id)  
GROUP BY department\_name  
ORDER BY department\_name;Code language: SQL (Structured Query Language) (sql) COUNTThe COUNT() function returns the number of items in a set. The following shows the syntax of the COUNT() function:COUNT ( [ALL | DISTINCT] column | expression | \*)Code language: SQL (Structured Query Language) (sql)For example, the following example uses the COUNT(\*) function to return the headcount of each department:SELECT   
 department\_name, COUNT(\*) headcount  
FROM  
 employees  
 INNER JOIN  
 departments USING (department\_id)  
GROUP BY department\_name  
ORDER BY department\_name;Code language: SQL (Structured Query Language) (sql) SUMThe SUM() function returns the sum of all values. The following illustrates the syntax of the SUM() function:SUM(ALL | DISTINCT column)Code language: SQL (Structured Query Language) (sql)For example, the following statement returns the total salary of all employees in each department:SELECT   
 department\_id, SUM(salary)  
FROM  
 employees  
GROUP BY department\_id;  
Code language: SQL (Structured Query Language) (sql)In this tutorial, you have learned the most commonly used SQL aggregate functions including AVG(), COUNT(), MIN(), MAX(), and SUM() functions. PreviouslySQL Functions Up NextSQL AVG

# Content from https://www.sqltutorial.org/sql-aggregate-functions/sql-avg/

Home » SQL Aggregate Functions » SQL AVGSQL AVGSummary: this tutorial, we will show you how to use SQL AVG function to get the average value of a set.Introduction to SQL AVG functionThe SQL AVG function is an aggregate function that calculates the average value of a set. The following illustrates the syntax of the SQL AVG function:AVG([ALL|DISTINCT] expression)Code language: SQL (Structured Query Language) (sql)If we use the ALL keyword, the AVG function takes all values in the calculation. By default, the AVG function uses ALL whether we specify it or not.If we specify the DISTINCT keyword explicitly, the AVG function will take the unique values only in the calculation.For example, we have a set of (1,2,3,3,4) and apply the AVG(ALL) to this set, the AVG function will perform the following calculation:(1+2+3+3+4)/5 = 2.6Code language: SQL (Structured Query Language) (sql)However, the AVG(DISTINCT) will process as follows:(1+2+3+4)/4 = 2.5Code language: SQL (Structured Query Language) (sql)SQL AVG function examplesWe will use the employees table in the sample database to demonstrate how the SQL AVG function works. The following picture illustrates the structure of the employees table:To calculate the average salary of all employees, you apply the AVG function to the salary column as follows:SELECT   
 AVG(salary)  
FROM  
 employees;Code language: SQL (Structured Query Language) (sql)Try ItLet’s apply the DISTINCT operator to see if the result changes:SELECT   
 AVG(DISTINCT salary)  
FROM  
 employees;Code language: SQL (Structured Query Language) (sql)Try ItIt changed because some employees have the same salary.To round the result to 2 decimal places, you use the ROUND function as follows:SELECT   
 ROUND(AVG(DISTINCT salary), 2)  
FROM  
 employees;Code language: SQL (Structured Query Language) (sql)Try ItTo calculate the average value of a subset of values, we add a WHERE clause to the SELECT statement. For instance, to calculate the average salary of employees in the department id 5, we use the following query:SELECT  
 AVG(DISTINCT salary)  
FROM  
 employees  
WHERE  
 department\_id = 5;Code language: SQL (Structured Query Language) (sql)Try ItThe following statement returns the average salary of employees who hold the job id 6:SELECT   
 AVG(salary)  
FROM  
 employees  
WHERE  
 job\_id = 6;Code language: SQL (Structured Query Language) (sql)Try ItSQL AVG with GROUP BY clause exampleTo calculate the average values of groups, we use the AVG function with the GROUP BY clause. For example, the following statement returns the departments and the average salary of employees of each department.SELECT  
 department\_id,  
 AVG(salary)  
FROM  
 employees  
GROUP BY  
 department\_id;Code language: SQL (Structured Query Language) (sql)Try ItWe can use the inner join clause to join the employees table with the departments table to get the department name data:SELECT  
 e.department\_id,  
 department\_name,  
 AVG(salary)  
FROM  
 employees e  
INNER JOIN departments d ON d.department\_id = e.department\_id  
GROUP BY  
 e.department\_id;Code language: SQL (Structured Query Language) (sql)Try ItSQL AVG with ORDER BY clause exampleTo sort the result set that includes the AVG results, you use the AVG function in the ORDER BY clause as follows:SELECT  
 e.department\_id,  
 department\_name,  
 AVG(salary)  
FROM  
 employees e  
INNER JOIN departments d ON d.department\_id = e.department\_id  
GROUP BY  
 e.department\_id  
ORDER BY  
 AVG(salary) DESC;Try ItSQL AVG with HAVING clause exampleTo filter group, you use the AVG function in the HAVING clause. For example, the following statement gets the department that has the average salary less than 5000:SQL AVG with a subqueryWe can apply AVG function multiple times in a single SQL statement to calculate the average value of a set of average values.For example, we can use the AVG function to calculate the average salary of employees in each department, and apply the AVG function one more time to calculate the average salary of departments.The following query illustrates the idea:SELECT  
 AVG(employee\_sal\_avg)  
FROM  
 (  
 SELECT  
 AVG(salary) employee\_sal\_avg  
 FROM  
 employees  
 GROUP BY  
 department\_id  
 ) t;Code language: SQL (Structured Query Language) (sql)Try ItHow the query works.The subquery returns a set of the average salaries of employees for each department.The outer query returns the average salary of departments.In this tutorial, you have learned how to use the SQL AVG function to calculate the average value of a set.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL Aggregate Functions Up NextSQL COUNT

# Content from https://www.sqltutorial.org/sql-aggregate-functions/sql-count/

Home » SQL Aggregate Functions » SQL COUNTSQL COUNTSummary: in this tutorial, you will learn how to use the SQL COUNT function to get the number of items in a group.Introduction to SQL COUNT functionThe SQL COUNT function is an aggregate function that returns the number of rows returned by a query. You can use the COUNT function in the SELECT statement to get the number of employees, the number of employees in each department, the number of employees who hold a specific job, etc.The following illustrates the syntax of the SQL COUNT function:COUNT([ALL | DISTINCT] expression);Code language: SQL (Structured Query Language) (sql)The result of the COUNT function depends on the argument that you pass to it.The ALL keyword will include the duplicate values in the result. For example, if you have a group (1, 2, 3, 3, 4, 4) and apply the COUNT function, the result is 6. By default, the COUNT function uses the ALL keyword whether you specify it or not.The DISTINCT keyword counts only unique values. For example, the COUNT function returns 4 if you apply it to the group (1, 2, 3, 3, 4, 4).The expression is a column of a table where you want to count the value.Another form of the COUNT function that accepts an asterisk (\*) as the argument is as follows:COUNT(\*)Code language: SQL (Structured Query Language) (sql)The COUNT(\*) function returns the number of rows in a table in a query. It counts duplicate rows and rows that contain null values.SQL COUNT function examplesLet’s take some examples to see how the COUNT function works. We will use the employees table in the sample database for demonstration purposes:SQL COUNT(\*) examplesThe following example uses the COUNT(\*) function to get the number of rows from the employees table:SELECT   
 COUNT(\*)  
FROM  
 employees;Code language: SQL (Structured Query Language) (sql)Try ItThe following example uses the COUNT(\*) function to count employees who work in the department with id 6:SELECT   
 COUNT(\*)  
FROM  
 employees  
WHERE  
 department\_id = 6;Code language: SQL (Structured Query Language) (sql)Try ItIn this example:First, the WHERE clause filter department with id 6.Second, the COUNT(\*) function returns the number of rows from the employees table with the value in the department id 6.The following example uses the COUNT(\*) function to get the number of employees with the job id 9:SELECT   
 COUNT(\*)  
FROM  
 employees  
WHERE  
 job\_id = 9;Code language: SQL (Structured Query Language) (sql)Try ItHow it works.First, the WHERE clause includes the rows from the employees table with the job id 9.Second, the COUNT(\*) returns the number of rows from the employees table with the job id 9The following example uses the AS keyword to assign the COUNT(\*) a column alias:SELECT   
 COUNT(\*) as employee\_count  
FROM  
 employees  
WHERE  
 job\_id = 9;Code language: JavaScript (javascript)SQL COUNT with GROUP BY clause examplesThe following example uses the COUNT function with GROUP BY clause to find the number of employees for each department:SELECT  
 department\_id,  
 COUNT(\*)  
FROM  
 employees  
GROUP BY  
 department\_id;Code language: SQL (Structured Query Language) (sql)Try ItHow it works:First, the GROUP BY clause groups the rows in the employees table by the department id.Second, the COUNT(\*) function returns the number of rows for each groupThe following example uses the COUNT(\*) function to get the number of employees by department. Also, it uses an INNER JOIN clause to include the department name in the result set:SELECT  
 e.department\_id,  
 department\_name,  
 COUNT(\*)  
FROM  
 employees e  
INNER JOIN departments d   
 ON d.department\_id = e.department\_id  
GROUP BY  
 e.department\_id,  
 department\_name;Code language: SQL (Structured Query Language) (sql)Try ItSQL COUNT(\*) with ORDER BY clause exampleThe following example uses the ORDER BY clause to sort the number of employees by department:SELECT  
 e.department\_id,  
 department\_name,  
 COUNT(\*)  
FROM  
 employees e  
INNER JOIN departments d ON d.department\_id = e.department\_id  
GROUP BY  
 e.department\_id  
ORDER BY  
 COUNT(\*) DESC;Code language: SQL (Structured Query Language) (sql)Try ItSQL COUNT with HAVING clause exampleThe following example returns the number of employees by department. Also, it uses a HAVING clause to select only departments that have more than five employees:SELECT  
 e.department\_id,  
 department\_name,  
 COUNT(\*)  
FROM  
 employees e  
INNER JOIN departments d ON d.department\_id = e.department\_id  
GROUP BY  
 e.department\_id  
HAVING  
 COUNT(\*) > 5  
ORDER BY  
 COUNT(\*) DESC;Code language: SQL (Structured Query Language) (sql)Try ItSQL COUNT(DISTINCT expression) exampleThe following example uses the COUNT to get the number of values in the job\_id column in the employees table:SELECT   
 COUNT(job\_id)  
FROM  
 employees;Code language: SQL (Structured Query Language) (sql)Try ItThe query returns 40 which includes the duplicate values. To remove the duplicates, you use the DISTINCT keyword to the COUNT function as follows:SELECT   
 COUNT(DISTINCT job\_id)  
FROM  
 employees;The following example uses the COUNT function with the DISTINCT keyword to get the number of managers:SELECT   
 COUNT(DISTINCT manager\_id)  
FROM  
 employees;Code language: SQL (Structured Query Language) (sql)Try ItSummaryThe COUNT function returns the number of rows in a group.The ALL keyword includes duplicate values while the DISTINCT keyword removes the duplicate values in the result.The COUNT(\*) returns the number of rows in a query including duplicate rows and rows that contain null values.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL AVG Up NextSQL MAX

# Content from https://www.sqltutorial.org/sql-aggregate-functions/sql-sum/

Home » SQL Aggregate Functions » SQL SUMSQL SUMSummary: in this tutorial, we will introduce you to the SQL SUM function that calculates the sum of all or distinct values.Introduction to the SQL SUM functionThe SQL SUM function is an aggregate function that returns the sum of all or distinct values. We can apply the SUM function to the numeric column only.The following illustrates the syntax of the SUM function.SUM([ALL|DISTINCT] expression)Code language: SQL (Structured Query Language) (sql)The ALL operator allows you to apply the aggregate to all values. The SUM function uses the ALL operator by default.For example, if you have a set of (1,2,3,3,NULL). The SUM function returns 9. Note that the SUM function ignores NULL values.To calculate the sum of unique values, you use the DISTINCT operator e.g., the SUM(DISTINCT) of the set (1,2,3,3,NULL) is 6.SQL SUM function examplesWe will use the employees table below for the demonstration purposes.To get the sum of salaries of all employees, we apply the SUM function to the salary column as the following query:SELECT   
 SUM(salary)  
FROM  
 employees;Code language: SQL (Structured Query Language) (sql)Try ItTo calculate the sum of salaries of employees who work in department id 5, we add a WHERE clause to the query above as follows:SELECT   
 SUM(salary)  
FROM  
 employees  
WHERE  
 department\_id = 5;Code language: SQL (Structured Query Language) (sql)Try ItSQL SUM with GROUP BY clause exampleWe often use the SUM function in conjunction with the GROUP BY clause to calculate the sums of groups.For example, to calculate the sum of salaries of employees for each department, we apply the SUM function to the salary column and group the rows by department\_id column as the following query:SELECT  
 department\_id,  
 SUM(salary)  
FROM  
 employees  
GROUP BY  
 department\_id;Code language: SQL (Structured Query Language) (sql)Try ItTo include the department name in the result set, we join the employees table with the departments table as follows:SELECT  
 e.department\_id,  
 department\_name,  
 SUM(salary)  
FROM  
 employees e  
INNER JOIN departments d ON d.department\_id = e.department\_id  
GROUP BY  
 e.department\_id;Code language: SQL (Structured Query Language) (sql)Try ItSQL SUM with ORDER BY clause exampleSuppose we want to have the sums of salaries of departments sorted in descending order, in this case, we use the SUM function in the ORDER BY clause as follows:SELECT  
 e.department\_id,  
 department\_name,  
 SUM(salary)  
FROM  
 employees e  
INNER JOIN departments d ON d.department\_id = e.department\_id  
GROUP BY  
 e.department\_id  
ORDER BY  
 SUM(salary) DESC;Code language: SQL (Structured Query Language) (sql)Try ItSQL SUM function with HAVING clause exampleTo filter groups based on condition, you use the HAVING clause. If you want to filter the groups based on the result of the SUM function, you have to place the SUM function in the GROUP BY clause.For example, if you want to get the departments and their sums of salaries greater than 3000, you use the following statement:SELECT  
 e.department\_id,  
 department\_name,  
 SUM(salary)  
FROM  
 employees e  
INNER JOIN departments d ON d.department\_id = e.department\_id  
GROUP BY  
 e.department\_id  
HAVING  
 SUM(salary) > 30000  
ORDER BY  
 SUM(salary) DESC;Code language: SQL (Structured Query Language) (sql)Try ItIn this tutorial, you have learned how to use the SUM function to calculate the sum of values in a set.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL MIN Up NextSQL Comparison Functions

# Content from https://www.sqltutorial.org/sql-max/

Home » SQL Aggregate Functions » SQL MAXSQL MAXSummary: in this tutorial, you will learn how to find the maximum value in a group by using the SQL SUM function.Introduction to SQL MAX functionSQL provides the MAX function that allows you to find the maximum value in a set of values. The following illustrates the syntax of the MAX function.MAX(expression)Code language: SQL (Structured Query Language) (sql)The MAX function ignores NULL values.Unlike the SUM, COUNT, and AVG functions, the DISTINCT option is not applicable to the MAX function.SQL MAX function examplesWe will use the employees table to demonstrate how the MAX function works.The following SELECT statement returns the highest (maximum) salary of employees in the employees table.SELECT   
 MAX(salary)  
FROM  
 employees;Code language: SQL (Structured Query Language) (sql)Try ItTo get the employees who have the highest salary, you use a subquery as follows:SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 salary  
FROM  
 employees  
WHERE  
 salary = (  
 SELECT  
 MAX(salary)  
 FROM  
 employees  
 );Code language: SQL (Structured Query Language) (sql)Try ItThe subquery returns the highest salary. The outer query gets the employees who have the salary that equals the highest salary.SQL MAX with GROUP BY exampleWe usually use the MAX function in conjunction the GROUP BY clause to find the maximum value per group.For example, we can use the MAX function to find the highest salary of employee in each department as follows:SELECT  
 department\_id,  
 MAX(salary)  
FROM  
 employees  
GROUP BY  
 department\_id;Code language: SQL (Structured Query Language) (sql)Try ItTo include the department names in the result, we join the employees table with the departments table as follows:SELECT  
 d.department\_id,  
 department\_name,  
 MAX(salary)  
FROM  
 employees e  
INNER JOIN departments d ON d.department\_id = e.department\_id  
GROUP BY  
 e.department\_id;Code language: SQL (Structured Query Language) (sql)Try ItSQL MAX with ORDER BY exampleLike other aggregate functions, to sort the result set based on the result of the MAX function, we have to place the MAX function in the ORDER BY clause.For example, the following statement returns the highest salaries of employees in each department and sorts the result set based on the highest salaries.SELECT  
 d.department\_id,  
 department\_name,  
 MAX(salary)  
FROM  
 employees e  
INNER JOIN departments d ON d.department\_id = e.department\_id  
GROUP BY  
 e.department\_id  
ORDER BY  
 MAX(salary) DESC;Code language: SQL (Structured Query Language) (sql)Try ItSQL MAX with HAVING exampleWe use the MAX function in the HAVING clause to add the condition to the groups that summarized by the GROUP BY clause.For example, to get the department that has employee whose highest salary is greater than 12000, you use the MAX function in the HAVING clause as follows:SELECT  
 d.department\_id,  
 department\_name,  
 MAX(salary)  
FROM  
 employees e  
INNER JOIN departments d ON d.department\_id = e.department\_id  
GROUP BY  
 e.department\_id  
HAVING  
 MAX(salary) > 12000;Code language: SQL (Structured Query Language) (sql)Try ItIn this tutorial, we have shown you how to use the MAX function to find the maximum value in a group of values.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL COUNT Up NextSQL MIN

# Content from https://www.sqltutorial.org/sql-aggregate-functions/sql-min/

Home » SQL Aggregate Functions » SQL MINSQL MINSummary: this tutorial helps you master the SQL MIN function through practical examples. After the tutorial, you will know how to apply the MIN function effectively to find the minimum value in a set of values.Introduction to SQL MIN functionThe SQL MIN function returns the minimum value in a set of values. The following demonstrates the syntax of the MIN function.MIN(expression)Code language: SQL (Structured Query Language) (sql)Like the MAX function, the MIN function also ignores NULL values and the DISTINCT option is not applicable to the MIN function.SQL MIN function examplesWe will use the employees table to demonstrate the functionality of the MIN function.To find the lowest (minimum) salary of employees, you apply the MIN function to the salary column of the employees table.SELECT   
 MIN(salary)  
FROM  
 employees;Code language: SQL (Structured Query Language) (sql)Try ItTo get the information of the employee who has the lowest salary, you use the following subquery:SELECT  
 employee\_id,  
 first\_name,  
 last\_name,  
 salary  
FROM  
 employees  
WHERE  
 salary = (  
 SELECT  
 MIN(salary)  
 FROM  
 employees  
 );  
Code language: SQL (Structured Query Language) (sql)Try ItFirst, the subquery returns the minimum salary. Then, the outer query retrieves the employee whose salary is equal to the lowest salary returned by the subquery.SQL MIN with GROUP BY exampleWe often use the MIN function together with the GROUP BY clause to find the minimum value in each group.For instance, the following query returns the employees who have the lowest salaries in each department.SELECT  
 department\_id,  
 MIN(salary)  
FROM  
 employees  
GROUP BY  
 department\_id;Code language: SQL (Structured Query Language) (sql)Try ItThe GROUP BY clause groups the employees by department. For each group, the query returns the row with the lowest salary.The result set is not informative as long as we see the names of the department.To combine the department’s name in the result set, we need to join the employees table with the departments using the inner join clause as follows:SELECT  
 d.department\_id,  
 department\_name,  
 MIN(salary)  
FROM  
 employees e  
INNER JOIN departments d ON d.department\_id = e.department\_id  
GROUP BY  
 d.department\_id;Code language: SQL (Structured Query Language) (sql)Try ItSQL MIN with ORDER BY exampleTo sort the groups by the result of the MIN function, you need to put the MIN function in the ORDER BY clause.The following query first retrieves the employee who has the lowest salary in each department, then sorts these departments by the salary in ascending order.SELECT  
 d.department\_id,  
 department\_name,  
 MIN(salary)  
FROM  
 employees e  
INNER JOIN departments d ON d.department\_id = e.department\_id  
GROUP BY  
 d.department\_id  
ORDER BY  
 MIN(salary);Code language: SQL (Structured Query Language) (sql)Try ItSQL MIN with HAVING exampleYou use the HAVING clause to specify the filter condition for groups. To filter the groups based on the result of the MIN function, you place the MIN function in the HAVING clause.For example, the following query retrieves the employees who have the lowest salary in each department. Then, it includes only departments whose salary is less than 3000.SELECT  
 d.department\_id,  
 department\_name,  
 MIN(salary)  
FROM  
 employees e  
INNER JOIN departments d ON d.department\_id = e.department\_id  
GROUP BY  
 d.department\_id  
HAVING  
 MIN(salary) < 3000;Code language: SQL (Structured Query Language) (sql)Try ItSo now you should know how to use the SQL MIN function to find the minimum value in a set of values.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL MAX Up NextSQL SUM

# Content from https://www.sqltutorial.org/sql-group-by/

Home » SQL Tutorial » SQL GROUP BYSQL GROUP BYSummary: in this tutorial, you will learn how to use the SQL GROUP BY clause to group rows based on one or more columns.Introduction to SQL GROUP BY clauseThe GROUP BY is an optional clause of the SELECT statement. The GROUP BY clause allows you to group rows based on values of one or more columns. It returns one row for each group.The following shows the basic syntax of the GROUP BY clause:SELECT  
 column1,  
 column2,  
 aggregate\_function(column3)  
FROM  
 table\_name  
GROUP BY  
 column1,  
 column2;Code language: SQL (Structured Query Language) (sql)The following picture illustrates shows how the GROUP BY clause works:The table on the left side has two columns id and fruit. When you apply the GROUP BY clause to the fruit column, it returns the result set that includes unique values from the fruit column:SELECT   
 fruit  
FROM   
 sample\_table  
GROUP BY   
 fruit;In practice, you often use the GROUP BY clause with an aggregate function such as MIN, MAX, AVG, SUM, or COUNT to calculate a measure that provides the information for each group.For example, the following illustrates how the GROUP BY clause works with the COUNT aggregate function:In this example, we group the rows by the values of the fruit column and apply the COUNT function to the id column. The result set includes the unique values of the fruit columns and the number of the corresponding rows.SELECT   
 fruit, COUNT(id)   
FROM  
 sample\_table  
GROUP BY   
 fruit;The columns that appear in the GROUP BY clause are called grouping columns. If a grouping column contains NULL values, all NULL values are summarized into a single group because the GROUP BY clause considers all NULL values equal.SQL GROUP BY examplesWe will use the employees and departments tables in the sample database to demonstrate how the GROUP BY clause works.The following example uses the GROUP BY clause to group the values in department\_id column of the employees table:SELECT   
 department\_id  
FROM   
 employees  
GROUP BY   
 department\_id;Code language: SQL (Structured Query Language) (sql)Output:Try It+---------------+  
| department\_id |  
+---------------+  
| 1 |  
| 2 |  
| 3 |  
| 4 |  
| 5 |  
| 6 |  
| 7 |  
| 8 |  
| 9 |  
| 10 |  
| 11 |  
+---------------+  
11 rows in set (0.00 sec)Code language: JavaScript (javascript)In this example:First, the SELECT clause returns all values from the department\_id column of employees table.Second, the GROUP BY clause groups all values into groups.The department\_id column of the employees table has 40 rows, including duplicate department\_id values. However, the GROUP BY groups these values into groups.Without an aggregate function, the GROUP BY behaves like the DISTINCT keyword:SELECT   
 DISTINCT department\_id  
FROM   
 employees  
ORDER BY   
 department\_id;Code language: SQL (Structured Query Language) (sql)Try ItThe GROUP BY clause will be more useful when you use it with an aggregate function.For example, the following statement uses the GROUP BY clause with the COUNT function to count the number of employees by department:SELECT  
 department\_id,  
 COUNT(employee\_id) headcount  
FROM  
 employees  
GROUP BY  
 department\_id;Code language: SQL (Structured Query Language) (sql)Try ItOutput:+---------------+-----------+  
| department\_id | headcount |  
+---------------+-----------+  
| 1 | 1 |  
| 2 | 2 |  
| 3 | 6 |  
| 4 | 1 |  
| 5 | 7 |  
| 6 | 5 |  
| 7 | 1 |  
| 8 | 6 |  
| 9 | 3 |  
| 10 | 6 |  
| 11 | 2 |  
+---------------+-----------+  
11 rows in set (0.00 sec)Code language: JavaScript (javascript)How it works.First, the GROUP BY clause groups the rows in the employees table by department id.Second, the COUNT(employee\_id) returns the number of employee id values in each group.SQL GROUP BY with INNER JOIN exampleThe following example returns the number of employees by department. And it uses an INNER JOIN clause to include the department name in the result:SELECT  
 department\_name,  
 COUNT(employee\_id) headcount  
FROM  
 employees e  
INNER JOIN departments d ON d.department\_id = e.department\_id  
GROUP BY  
 department\_name;Code language: SQL (Structured Query Language) (sql)Try ItOutput:+------------------+-----------+  
| department\_name | headcount |  
+------------------+-----------+  
| Accounting | 2 |  
| Administration | 1 |  
| Executive | 3 |  
| Finance | 6 |  
| Human Resources | 1 |  
| IT | 5 |  
| Marketing | 2 |  
| Public Relations | 1 |  
| Purchasing | 6 |  
| Sales | 6 |  
| Shipping | 7 |  
+------------------+-----------+  
11 rows in set (0.01 sec)Code language: JavaScript (javascript)SQL GROUP BY with ORDER BY exampleThe following example uses an ORDER BY clause to sort the departments by headcount:SELECT   
 department\_name,  
 COUNT(employee\_id) headcount  
FROM  
 employees e  
 INNER JOIN  
 departments d ON d.department\_id = e.department\_id  
GROUP BY department\_name  
ORDER BY headcount DESC;Code language: SQL (Structured Query Language) (sql)Try ItOutput:+------------------+-----------+  
| department\_name | headcount |  
+------------------+-----------+  
| Shipping | 7 |  
| Sales | 6 |  
| Finance | 6 |  
| Purchasing | 6 |  
| IT | 5 |  
| Executive | 3 |  
| Marketing | 2 |  
| Accounting | 2 |  
| Human Resources | 1 |  
| Administration | 1 |  
| Public Relations | 1 |  
+------------------+-----------+  
11 rows in set (0.00 sec)Code language: JavaScript (javascript)Note that you can use either the headcount alias or the COUNT(employee\_id) in the ORDER BY clause.SQL GROUP BY with HAVING exampleThe following example uses the HAVING clause to find departments with headcounts are greater than 5:SELECT   
 department\_name,  
 COUNT(employee\_id) headcount  
FROM  
 employees e  
 INNER JOIN  
 departments d ON d.department\_id = e.department\_id  
GROUP BY department\_name  
HAVING headcount > 5  
ORDER BY headcount DESC;Code language: SQL (Structured Query Language) (sql)Try ItOutput:+-----------------+-----------+  
| department\_name | headcount |  
+-----------------+-----------+  
| Shipping | 7 |  
| Sales | 6 |  
| Finance | 6 |  
| Purchasing | 6 |  
+-----------------+-----------+  
4 rows in set (0.00 sec)Code language: JavaScript (javascript)SQL GROUP BY with MIN, MAX, and AVG exampleThe following query returns the minimum, maximum, and average salary of employees in each department.SELECT   
 department\_name,  
 MIN(salary) min\_salary,  
 MAX(salary) max\_salary,  
 ROUND(AVG(salary), 2) average\_salary  
FROM  
 employees e  
 INNER JOIN  
 departments d ON d.department\_id = e.department\_id  
GROUP BY   
 department\_name;Code language: SQL (Structured Query Language) (sql)Try ItOutput:+------------------+------------+------------+----------------+  
| department\_name | min\_salary | max\_salary | average\_salary |  
+------------------+------------+------------+----------------+  
| Accounting | 8300.00 | 12000.00 | 10150.00 |  
| Administration | 4400.00 | 4400.00 | 4400.00 |  
| Executive | 17000.00 | 24000.00 | 19333.33 |  
| Finance | 6900.00 | 12000.00 | 8600.00 |  
| Human Resources | 6500.00 | 6500.00 | 6500.00 |  
| IT | 4200.00 | 9000.00 | 5760.00 |  
| Marketing | 6000.00 | 13000.00 | 9500.00 |  
| Public Relations | 10000.00 | 10000.00 | 10000.00 |  
| Purchasing | 2500.00 | 11000.00 | 4150.00 |  
| Sales | 6200.00 | 14000.00 | 9616.67 |  
| Shipping | 2700.00 | 8200.00 | 5885.71 |  
+------------------+------------+------------+----------------+  
11 rows in set (0.01 sec)Code language: JavaScript (javascript)SQL GROUP BY with SUM function exampleTo get the total salary per department, you apply the SUM function to the salary column and group employees by the department\_id column as follows:SELECT   
 department\_name,  
 SUM(salary) total\_salary  
FROM  
 employees e  
 INNER JOIN  
 departments d ON d.department\_id = e.department\_id  
GROUP BY   
 department\_name;Code language: SQL (Structured Query Language) (sql)Try ItOutput:+------------------+--------------+  
| department\_name | total\_salary |  
+------------------+--------------+  
| Accounting | 20300.00 |  
| Administration | 4400.00 |  
| Executive | 58000.00 |  
| Finance | 51600.00 |  
| Human Resources | 6500.00 |  
| IT | 28800.00 |  
| Marketing | 19000.00 |  
| Public Relations | 10000.00 |  
| Purchasing | 24900.00 |  
| Sales | 57700.00 |  
| Shipping | 41200.00 |  
+------------------+--------------+  
11 rows in set (0.01 sec)Code language: JavaScript (javascript)SQL GROUP BY multiple columnsSo far, you have seen that we have grouped all employees by one column. For example, the following clause places all rows with the same values in the department\_id column in one group.GROUP BY department\_idCode language: SQL (Structured Query Language) (sql)How about grouping employees by values in both department\_id and job\_id columns?GROUP BY department\_id, job\_idCode language: SQL (Structured Query Language) (sql)This clause will group all employees with the same values in both department\_id and job\_id columns in one group.The following statement groups rows with the same values in both department\_id and job\_id columns in the same group then return the rows for each of these groups.SELECT   
 department\_name,  
 job\_title,  
 COUNT(employee\_id)  
FROM  
 employees e  
 INNER JOIN  
 departments d ON d.department\_id = e.department\_id  
 INNER JOIN  
 jobs j ON j.job\_id = e.job\_id  
GROUP BY department\_name ,   
 job\_title;Code language: SQL (Structured Query Language) (sql)Try It+------------------+---------------------------------+--------------------+  
| department\_name | job\_title | COUNT(employee\_id) |  
+------------------+---------------------------------+--------------------+  
| Accounting | Accounting Manager | 1 |  
| Accounting | Public Accountant | 1 |  
| Administration | Administration Assistant | 1 |  
| Executive | Administration Vice President | 2 |  
| Executive | President | 1 |  
| Finance | Accountant | 5 |  
| Finance | Finance Manager | 1 |  
| Human Resources | Human Resources Representative | 1 |  
| IT | Programmer | 5 |  
| Marketing | Marketing Manager | 1 |  
| Marketing | Marketing Representative | 1 |  
| Public Relations | Public Relations Representative | 1 |  
| Purchasing | Purchasing Clerk | 5 |  
| Purchasing | Purchasing Manager | 1 |  
| Sales | Sales Manager | 2 |  
| Sales | Sales Representative | 4 |  
| Shipping | Shipping Clerk | 2 |  
| Shipping | Stock Clerk | 1 |  
| Shipping | Stock Manager | 4 |  
+------------------+---------------------------------+--------------------+  
19 rows in set (0.00 sec)Code language: PHP (php)SummaryThe GROUP BY clause groups the rows into groups based on the values of one or more columns.Use an aggregate function with the GROUP BY clause to calculate the summarized value for each group.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL CROSS JOIN Up NextSQL HAVING

# Content from https://www.sqltutorial.org/sql-having/

Home » SQL Tutorial » SQL HAVINGSQL HAVINGSummary: this tutorial introduces you to the SQL HAVING clause that allows you to specify a condition for the groups summarized by the GROUP BY clause.Introduction to SQL HAVING clauseIn the previous tutorial, you have learned how to use the GROUP BY clause to summarize rows into groups and apply the aggregate function such as MIN, MAX, SUM, COUNT, AVG to each group.To specify a condition for groups, you use the HAVING clause.The HAVING clause is often used with the GROUP BY clause in the SELECT statement. If you use a HAVING clause without a GROUP BY clause, the HAVING clause behaves like the WHERE clause.The following illustrates the syntax of the HAVING clause:SELECT  
 column1,  
 column2,  
 AGGREGATE\_FUNCTION (column3)  
FROM  
 table1  
GROUP BY  
 column1,  
 column2  
HAVING  
 group\_condition;Code language: SQL (Structured Query Language) (sql)Note that the HAVING clause appears immediately after the GROUP BY clause.HAVING vs. WHEREThe WHERE clause applies the condition to individual rows before the rows are summarized into groups by the GROUP BY clause. However, the HAVING clause applies the condition to the groups after the rows are grouped into groups.Therefore, it is important to note that the HAVING clause is applied after whereas the WHERE clause is applied before the GROUP BY clause.SQL HAVING clause examplesWe will take the employees and departments tables in the sample database for the demonstration.To get the managers and their direct reports, you use the GROUP BY clause to group employees by the managers and use the COUNT function to count the direct reports.The following query illustrates the idea:SELECT   
 manager\_id,  
 first\_name,  
 last\_name,  
 COUNT(employee\_id) direct\_reports  
FROM  
 employees  
WHERE  
 manager\_id IS NOT NULL  
GROUP BY manager\_id;Code language: SQL (Structured Query Language) (sql)Try ItTo find the managers who have at least five direct reports, you add a HAVING clause to the query above as the following:SELECT   
 manager\_id,  
 first\_name,  
 last\_name,  
 COUNT(employee\_id) direct\_reports  
FROM  
 employees  
WHERE  
 manager\_id IS NOT NULL  
GROUP BY manager\_id  
HAVING direct\_reports >= 5;Code language: SQL (Structured Query Language) (sql)Try ItSQL HAVING with SUM function exampleThe following statement calculates the sum of salary that the company pays for each department and selects only the departments with the sum of salary between 20000 and 30000.SELECT   
 department\_id, SUM(salary)  
FROM  
 employees  
GROUP BY department\_id  
HAVING SUM(salary) BETWEEN 20000 AND 30000  
ORDER BY SUM(salary);Code language: SQL (Structured Query Language) (sql)Try ItSQL HAVING with MIN function exampleTo find the department that has employees with the lowest salary greater than 10000, you use the following query:SELECT  
 e.department\_id,  
 department\_name,  
 MIN(salary)  
FROM  
 employees e  
INNER JOIN departments d ON d.department\_id = e.department\_id  
GROUP BY  
 e.department\_id  
HAVING  
 MIN(salary) >= 10000  
ORDER BY  
 MIN(salary);Code language: SQL (Structured Query Language) (sql)Try ItHow the query works.First, use the GROUP BY clause to groups employees by department.Second, use the MIN function to find the lowest salary per group.Third, apply the condition to the HAVING clause.SQL HAVING clause with AVG function exampleTo find the departments that have the average salaries of employees between 5000 and 7000, you use the AVG function as the following query:SELECT  
 e.department\_id,  
 department\_name,  
 ROUND(AVG(salary), 2)  
FROM  
 employees e  
INNER JOIN departments d ON d.department\_id = e.department\_id  
GROUP BY  
 e.department\_id  
HAVING  
 AVG(salary) BETWEEN 5000  
AND 7000  
ORDER BY  
 AVG(salary);Code language: SQL (Structured Query Language) (sql)Try ItIn this tutorial, you have learned how to use the SQL HAVING clause to apply the condition to groups.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL GROUP BY Up NextSQL GROUPING SETS

# Content from https://www.sqltutorial.org/sql-grouping-sets/

Home » SQL Tutorial » SQL GROUPING SETSSQL GROUPING SETSSummary: in this tutorial, you will learn how to use the SQL GROUPING SETS operator to generate multiple grouping sets.Setup a sample tableLet’s set up a new table named inventory to demonstrate the functionality of the GROUPING SETS.First, create a new table named inventory :CREATE TABLE inventory (  
 warehouse VARCHAR(255),  
 product VARCHAR(255) NOT NULL,  
 model VARCHAR(50) NOT NULL,  
 quantity INT,  
 PRIMARY KEY (warehouse,product,model)  
);  
Code language: SQL (Structured Query Language) (sql)Second, insert data into the inventory table:INSERT INTO inventory(warehouse, product, model, quantity)  
VALUES('San Jose', 'iPhone','6s',100);  
INSERT INTO inventory(warehouse, product, model, quantity)  
VALUES('San Fransisco', 'iPhone','6s',50);  
INSERT INTO inventory(warehouse, product, model, quantity)  
VALUES('San Jose','iPhone','7',50);  
INSERT INTO inventory(warehouse, product, model, quantity)  
VALUES('San Fransisco', 'iPhone','7',10);  
INSERT INTO inventory(warehouse, product, model, quantity)  
VALUES('San Jose','iPhone','X',150);  
INSERT INTO inventory(warehouse, product, model, quantity)  
VALUES('San Fransisco', 'iPhone','X',200);  
INSERT INTO inventory(warehouse, product, model, quantity)  
VALUES('San Jose','Samsung','Galaxy S',200);  
INSERT INTO inventory(warehouse, product, model, quantity)  
VALUES('San Fransisco','Samsung','Galaxy S',200);  
INSERT INTO inventory(warehouse, product, model, quantity)  
VALUES('San Fransisco','Samsung','Note 8',100);  
INSERT INTO inventory(warehouse, product, model, quantity)  
VALUES('San Jose','Samsung','Note 8',150);Code language: JavaScript (javascript)Third, query data from the inventory table:SELECT   
 \*  
FROM  
 inventory;  
Code language: SQL (Structured Query Language) (sql)Introduction to SQL GROUPING SETSA grouping set is a set of columns by which you group using the GROUP BY clause. Normally, a single aggregate query defines a single grouping set.The following example defines a grouping set (warehouse, product). It returns the number of stock keeping units (SKUs) stored in the inventory by warehouse and product.SELECT  
 warehouse,  
 product,   
 SUM (quantity) qty  
FROM  
 inventory  
GROUP BY  
 warehouse,  
 product;  
Code language: SQL (Structured Query Language) (sql)The following query finds the number of SKUs by the warehouse. It defines the grouping set (warehouse):SELECT  
 warehouse,   
 SUM (quantity) qty  
FROM  
 inventory  
GROUP BY  
 warehouse;  
Code language: SQL (Structured Query Language) (sql)The following query returns the number of SKUs by the product. It defines the grouping set (product):SELECT  
 product,   
 SUM (quantity) qty  
FROM  
 inventory  
GROUP BY  
 product;  
Code language: SQL (Structured Query Language) (sql)The following query finds the number of SKUs for all warehouses and products. It defines an empty grouping set ().SELECT  
 SUM(quantity) qty  
FROM  
 inventory;  
Code language: SQL (Structured Query Language) (sql)So far, we have four grouping sets: (warehouse, product), (warehouse), (product), and (). To return all grouping sets using a single query, you can use the UNION ALL operator to combine all the queries above.The UNION ALL requires all result sets to have the same number of columns, therefore, you need to add NULL to the select list to of each query as shown below:SELECT  
 warehouse,  
 product,   
 SUM (quantity) qty  
FROM  
 inventory  
GROUP BY  
 warehouse,  
 product  
UNION ALL  
SELECT  
 warehouse,   
 null,  
 SUM (quantity) qty  
FROM  
 inventory  
GROUP BY  
 warehouse  
UNION ALL  
SELECT  
 null,  
 product,   
 SUM (quantity) qty  
FROM  
 inventory  
GROUP BY  
 product  
UNION ALL  
SELECT  
 null,  
 null,  
 SUM(quantity) qty  
FROM  
 inventory;  
Code language: SQL (Structured Query Language) (sql)Here is the output:As you can see clearly from the output, the query generated a single result set with the aggregates for all grouping sets.Even though the query works as expected, it has two main issues:First, the query is difficult to read because it is lengthy.Second, it has a performance issue because the database system has to scan the inventory table multiple times.To resolve these issues, SQL provides us with the GROUPING SETS.The GROUPING SETS is an option of the GROUP BY clause. The GROUPING SETS defines multiple grouping sets within the same query.The following illustrates the general syntax of the GROUPING SETS option:SELECT  
 c1,  
 c2,  
 aggregate (c3)  
FROM  
 table  
GROUP BY  
 GROUPING SETS (  
 (c1, c2),  
 (c1),  
 (c2),  
 ()  
);  
Code language: SQL (Structured Query Language) (sql)This query defines four grouping sets (c1,c2), (c1), (c2), and ().You can apply the GROUPING SETS to rewrite the query with the UNION ALL clauses above:SELECT  
 warehouse,  
 product,   
 SUM (quantity) qty  
FROM  
 inventory  
GROUP BY  
 GROUPING SETS(  
 (warehouse,product),  
 (warehouse),  
 (product),  
 ()  
 );Code language: SQL (Structured Query Language) (sql)Here is the output:This query is more readable and performed faster than the query above because the database system does not have to read the inventory table multiple times.Now, you should know how to use the SQL GROUPING SETS to generate multiple grouping sets using a single query.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL HAVING Up NextSQL ROLLUP

# Content from https://www.sqltutorial.org/sql-rollup/

Home » SQL Tutorial » SQL ROLLUPSQL ROLLUPSummary: in this tutorial, you will learn how to use the SQL ROLLUP to generate multiple grouping sets.Introduction to SQL ROLLUPThe ROLLUP is an extension of the GROUP BY clause. The ROLLUP option allows you to include extra rows that represent the subtotals, which are commonly referred to as super-aggregate rows, along with the grand total row. By using the ROLLUP option, you can use a single query to generate multiple grouping sets.Note that a grouping set is a set of columns by which you group. For example, a query that returns the inventory by the warehouse,  the grouping set is (warehouse).SELECT  
 warehouse,   
 SUM (quantity) qty  
FROM  
 inventory  
GROUP BY  
 warehouse;Code language: SQL (Structured Query Language) (sql)For more information about the GROUPING SETS, check it out the grouping sets tutorial.The following illustrates the basic syntax of the SQL ROLLUP:SELECT   
 c1, c2, aggregate\_function(c3)  
FROM  
 table  
GROUP BY ROLLUP (c1, c2);  
Code language: SQL (Structured Query Language) (sql)The ROLLUP assumes a hierarchy among the input columns. For example, if the input column is (c1,c2), the hierarchy c1 > c2. The ROLLUP generates all grouping sets that make sense considering this hierarchy. This is why we often use ROLLUP to generate the subtotals and the grand total for reporting purposes.In the syntax above, ROLLUP(c1,c2) generates three following grouping sets:(c1,c2)  
(c1)  
()Code language: SQL (Structured Query Language) (sql)This syntax is supported by Oracle, Microsoft SQL Server, and PostgreSQL. However, MySQL has a slightly different syntax as shown below:SELECT   
 c1, c2, aggregate\_function(c3)  
FROM  
 table\_name  
GROUP BY c1, c2 WITH ROLLUP;  
Code language: SQL (Structured Query Language) (sql)SQL ROLLUP examplesWe will use the inventory table that we set up in the GROUPING SETS tutorial for the demonstration.SQL ROLLUP with one column exampleThe following statement uses the GROUP BY clause and the SUM() function to find the total inventory by warehouse:SELECT   
 warehouse, SUM(quantity)  
FROM  
 inventory  
GROUP BY warehouse;  
Code language: SQL (Structured Query Language) (sql)To retrieve the total products in all warehouses, you add the ROLLUP to the GROUP BY clause as follows:SELECT   
 warehouse, SUM(quantity)  
FROM  
 inventory  
GROUP BY ROLLUP (warehouse);  
Code language: SQL (Structured Query Language) (sql)As you can see in the result, the NULL value in the warehouse column specifies the grand total super-aggregate line. In this example, the ROLLUP option causes the query to produce another row that shows the total products in all warehouses.To make the output more readable, you can use the COALESCE() function to substitute the NULL value by the All warehouses as follows:SELECT   
 COALESCE(warehouse, 'All warehouses') AS warehouse,  
 SUM(quantity)  
FROM  
 inventory  
GROUP BY ROLLUP (warehouse);  
Code language: SQL (Structured Query Language) (sql)SQL ROLLUP with multiple columns exampleThe following statement calculates the inventory by warehouse and product:SELECT   
 warehouse, product, SUM(quantity)  
FROM  
 inventory  
GROUP BY warehouse, product;  
Code language: SQL (Structured Query Language) (sql)Let’s add the ROLLUP to the GROUP BY clause:SELECT   
 warehouse, product, SUM(quantity)  
FROM  
 inventory  
GROUP BY ROLLUP (warehouse , product);  
Code language: SQL (Structured Query Language) (sql)Note that the output consists of summary information at two levels of analysis, not just one:Following each set of product rows for a specified warehouse, an extra summary row appears displaying the total inventory. In these rows, values in the productcolumn set to NULL.Following all rows, an extra summary row appears showing the total inventory of all warehouses and products. In these rows, the values in the  warehouse and product columns set to NULL.SQL ROLLUP with partial rollup exampleYou can use ROLLUP to perform a partial roll-up that reduces the number of subtotals calculated as shown in the following example:SELECT   
 warehouse, product, SUM(quantity)  
FROM  
 inventory  
GROUP BY warehouse, ROLLUP (product);  
Code language: SQL (Structured Query Language) (sql)In this example, the ROLLUP only makes a supper-aggregate summary for the product column, not the warehouse column.In this tutorial, you have learned how to use the SQL ROLLUP to perform multiple levels of analysis with a single query.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL GROUPING SETS Up NextSQL CUBE

# Content from https://www.sqltutorial.org/sql-cube/

Home » SQL Tutorial » SQL CUBESQL CUBESummary: in this tutorial, you will learn how to use the SQL CUBE to generate subtotals for the output of a query.Introduction to SQL CUBESimilar to the ROLLUP, CUBE is an extension of the GROUP BY clause. CUBE allows you to generate subtotals like the ROLLUP extension. In addition, the CUBE extension will generate subtotals for all combinations of grouping columns specified in the GROUP BY clause.The following illustrates the syntax of CUBE extension:SELECT   
 c1, c2, AGGREGATE\_FUNCTION(c3)  
FROM  
 table\_name  
GROUP BY CUBE(c1 , c2);  
Code language: SQL (Structured Query Language) (sql)In this syntax, we have two columns specified in the CUBE. The statement creates two subtotal combinations. Generally, if you have n number of columns listed in the CUBE, the statement will create 2n subtotal combinations.SQL CUBE examplesWe will reuse the inventory table created in the ROLLUP tutorial.SQL CUBE with one column exampleThe following statement uses the SUM() function and the GROUP BY clause to find the total inventory of every warehouse:SELECT  
 warehouse,  
 SUM(quantity)  
FROM  
 inventory  
GROUP BY  
 warehouse;  
Code language: SQL (Structured Query Language) (sql)If you want to know the total inventory in all warehouses, you use the CUBE extension in the GROUP BY clause as follows:SELECT  
 warehouse,  
 SUM(quantity)  
FROM  
 inventory  
GROUP BY  
 CUBE(warehouse)  
ORDER BY  
 warehouse;   
Code language: SQL (Structured Query Language) (sql)In this example, the CUBE extension adds a total inventory row with a null value in the warehouse column. The effect is the same as the ROLLUP function. To make the output more readable, you can use the COALESCE() function as shown below:SELECT  
 COALESCE(warehouse,'All warehouses'),  
 SUM(quantity)  
FROM  
 inventory  
GROUP BY  
 CUBE(warehouse)  
ORDER BY  
 warehouse;  
Code language: SQL (Structured Query Language) (sql)SQL CUBE with multiple columns exampleThe following statement finds the total inventory by warehouse and product:SELECT  
 warehouse,  
 product,  
 SUM(quantity)  
FROM  
 inventory  
GROUP BY  
 warehouse,product  
ORDER BY  
 warehouse,  
 product;  
Code language: SQL (Structured Query Language) (sql)When you use the CUBE function, the query makes four subtotals:SELECT  
 warehouse,  
 product,  
 SUM(quantity)  
FROM  
 inventory  
GROUP BY  
 CUBE(warehouse,product)  
ORDER BY  
 warehouse,  
 product;  
Code language: SQL (Structured Query Language) (sql)As you can see in the output, we have four subtotal rows:The third and sixth rows show the total inventory of all products in the San Francisco and San Jose warehouses. The values in the product column are null.The seventh and eighth rows display the total inventory by products which are Samsung and iPhone in all warehouses. Hence, the values in the warehouse columns are null.The last column is the grand total that shows the total inventory in all warehouses.The following statement uses the COALESCE() function to substitute null values by more meaningful data:SELECT  
 COALESCE(warehouse, '...All Warehouses') warehouse,  
 COALESCE(product, '...All Products') product,  
 SUM(quantity)   
FROM  
 inventory  
GROUP BY  
 CUBE(warehouse,product)  
ORDER BY  
 warehouse,  
 product;   
Code language: SQL (Structured Query Language) (sql)Creating cross-tabular reportsThe following query creates a cross-tabular report by retrieving data from the  employees table in the sample database using the CUBE extension:SELECT  
 COALESCE(department\_name, '-') department,  
 COALESCE(job\_title,'-') job,  
 COUNT(\*) ,  
 SUM(salary) salary  
FROM  
 employees  
INNER JOIN departments USING (department\_id)  
INNER JOIN jobs USING (job\_id)  
GROUP BY  
 CUBE(department\_name,job\_title)  
ORDER BY  
 department\_name ASC NULLS LAST;Code language: SQL (Structured Query Language) (sql)Here is the result:DepartmentJobCOUNT(\*)SALARYAccountingAccounting Manager112000AccountingPublic Accountant18300Accounting–220300AdministrationAdministration Assistant14400Administration–14400ExecutiveAdministration Vice President234000ExecutivePresident124000Executive–358000FinanceAccountant539600FinanceFinance Manager112000Finance–651600Human ResourcesHuman Resources Representative16500Human Resources–16500ITProgrammer528800IT–528800MarketingMarketing Manager113000MarketingMarketing Representative16000Marketing–219000Public RelationsPublic Relations Representative110000Public Relations–110000PurchasingPurchasing Clerk513900PurchasingPurchasing Manager111000Purchasing–624900SalesSales Manager227500SalesSales Representative324000Sales–551500ShippingShipping Clerk27900ShippingStock Clerk12700ShippingStock Manager430600Shipping–741200–Accountant539600–Accounting Manager112000–Administration Assistant14400–Administration Vice President234000–Finance Manager112000–Human Resources Representative16500–Marketing Manager113000–Marketing Representative16000–President124000–Programmer528800–Public Accountant18300–Public Relations Representative110000–Purchasing Clerk513900–Purchasing Manager111000–Sales Manager227500–Sales Representative324000–Shipping Clerk27900–Stock Clerk12700–Stock Manager430600––39316200In this tutorial, you have learned how to use the SQL CUBE extensions to generate the subtotal in the output of a query.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL ROLLUP Up NextSQL Subquery

# Content from https://www.sqltutorial.org/sql-union/

Home » SQL Tutorial » SQL UNIONSQL UNIONSummary: this tutorial shows you how to use the SQL UNION to combine two or more result sets from multiple queries and explains the difference between UNION and UNION ALL.Introduction to SQL UNION operatorThe UNION operator combines result sets of two or more SELECT statements into a single result set. The following statement illustrates how to use the UNION operator to combine result sets of two queries:SELECT   
 column1, column2  
FROM  
 table1   
UNION [ALL]  
SELECT   
 column3, column4  
FROM  
 table2;Code language: SQL (Structured Query Language) (sql)To use the UNION operator, you write the dividual SELECT statements and join them by the keyword UNION.The columns returned by the SELECT statements must have the same or convertible data type, size, and be the same order.The database system processes the query by executing two SELECT statements first. Then, it combines two individual result sets into one and eliminates duplicate rows. To eliminate the duplicate rows, the database system sorts the combined result set by every column and scans it for the matching rows located next to one another.To retain the duplicate rows in the result set, you use the UNION ALL operator.Suppose, we have two result sets A(1,2) and B(2,3). The following picture illustrates A UNION B:And the following picture illustrates A UNION ALL BThe union is different from the join that the join combines columns of multiple tables while the union combines rows of the tables.The SQL UNION examplesSQL UNION exampleTo get the data from the A table, you use the following SELECT statement:SELECT   
 id  
FROM  
 A;Code language: SQL (Structured Query Language) (sql)To retrieve the data from the B table, you use the following statement:SELECT   
 id  
FROM  
 B;Code language: SQL (Structured Query Language) (sql)To combine result sets of these two queries, you use the UNION operator as follows:SELECT  
 id  
FROM  
 a  
UNION  
SELECT  
 id  
FROM  
 b;Code language: SQL (Structured Query Language) (sql)The result set includes only 3 rows because the UNION operator removes one duplicate row.SQL UNION ALL exampleTo retain the duplicate row, you use the UNION ALL operator as follows:SQL UNION with ORDER BY exampleTo sort the result set, you place the ORDER BY clause after all the SELECT statements as follows:SELECT  
 id  
FROM  
 a  
UNION  
SELECT  
 id  
FROM  
 b  
ORDER BY id DESC;Code language: SQL (Structured Query Language) (sql)The database system performs the following steps:First, execute each SELECT statement individually.Second, combine result sets and remove duplicate rows to create the combined result set.Third, sort the combined result set by the column specified in the ORDER BY clause.In practice, we often use the UNION operator to combine data from different tables. See the following employees and dependents tables:The following statement uses the UNION operator to combine the first name and last name of employees and dependents.SELECT  
 first\_name,  
 last\_name  
FROM  
 employees  
UNION  
SELECT  
 first\_name,  
 last\_name  
FROM  
 dependents  
ORDER BY  
 last\_name;Code language: SQL (Structured Query Language) (sql)Try ItIn this tutorial, you have learned how to use the UNION operator to combine two or more result sets from multiple queries.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL EXISTS Up NextSQL INTERSECT

# Content from https://www.sqltutorial.org/sql-intersect/

Home » SQL Tutorial » SQL INTERSECTSQL INTERSECTSummary: this tutorial explains the SQL INTERSECT operator and shows you how to apply it to get the intersection of two or more queries.Introduction to SQL INTERSECT operatorThe INTERSECT operator is a set operator that returns distinct rows of two or more result sets from SELECT statements.Suppose, we have two tables: A(1,2) and B(2,3).The following picture illustrates the intersection of A & B tables.The purple section is the intersection of the green and blue result sets.Like the UNION operator, the INTERSECT operator removes the duplicate rows from the final result set.The following statement illustrates how to use the INTERSECT operator to find the intersection of two result sets.SELECT  
 id  
FROM  
 a   
INTERSECT  
SELECT  
 id  
FROM  
 b;Code language: SQL (Structured Query Language) (sql)To use the INTERSECT operator, the columns of the SELECT statements must follow the rules:The data types of columns must be compatible.The number of columns and their orders in the SELECT statements must be the same.SQL INTERSECT operator exampleThe following SELECT statement returns rows from the table A:SELECT  
 id  
FROM  
 A;Code language: SQL (Structured Query Language) (sql)And the following statement retrieves the data from the table B:SELECT  
 id  
FROM  
 B;Code language: SQL (Structured Query Language) (sql)The following statement uses the INTERSECT operator to get the intersection of both queries.SELECT  
 id  
FROM  
 a   
INTERSECT  
SELECT  
 id  
FROM  
 b;Code language: SQL (Structured Query Language) (sql)SQL INTERSECT with ORDER BY exampleTo sort the result set returned by the INTERSECT operator, you place the ORDER BY clause at the end of all statements.For example, the following statement applies the INTERSECT operator to the A and B tables and sorts the combined result set by the id column in descending order.SELECT  
 id  
FROM  
 a   
INTERSECT  
SELECT  
 id  
FROM  
 b  
ORDER BY id DESC;Code language: SQL (Structured Query Language) (sql)Emulate SQL INTERSECT operator using INNER JOIN clauseMost relational database system supports the INTERSECT operator such as Oracle Database, Microsoft SQL Server, PostgreSQL, etc. However, some database systems do not provide the INTERSECT operator like MySQL.To emulate the SQL INTERSECT operator, you can use the INNER JOIN clause as follows:SELECT  
 a.id  
FROM  
 a  
INNER JOIN b ON b.id = a.idCode language: SQL (Structured Query Language) (sql)It returns the rows in the A table that have matching rows in the B table, which produces the same result as the INTERSECT operator.Now you should have a good understanding of the SQL INTERSECT operator and know how to use it to find the intersections of multiple queries.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL UNION Up NextSQL MINUS

# Content from https://www.sqltutorial.org/sql-minus/

Home » SQL Tutorial » SQL MINUSSQL MINUSSummary: in this tutorial, you will learn how to use the SQL MINUS operator to subtract one result set from another.Introduction to SQL MINUS operatorBesides the UNION, UNION ALL, and INTERSECT operators, SQL provides us with the MINUS operator that allows you to subtract one result set from another result set.The following illustrates the syntax of the MINUS operator.SELECT  
 id  
FROM  
 A   
MINUS   
SELECT  
 id  
FROM  
 B;Code language: SQL (Structured Query Language) (sql)To use the MINUS operator, you write individual SELECT statements and place the MINUS operator between them. The MINUS operator returns the unique rows produced by the first query but not by the second one.The following picture illustrates the MINUS operator.To make the result set, the database system performs two queries and subtracts the result set of the first query from the second one.In order to use the MINUS operator, the columns in the SELECT clauses must match in number and must have the same or, at least, convertible data type.We often use the MINUS operator in ETL. An ETL is a software component in data warehouse system. ETL stands for Extract, Transform, and Load. ETL is responsible for loading data from the source systems into the data warehouse system.After complete loading data, we can use the MINUS operator to make sure that the data has been loaded fully by subtracting data in target system from the data in the source system.SQL MINUS examplesConsider the following employees and dependents tables in the sample database.Each employee has zero or more dependents while each dependent depends on one and only one employees. The relationship between the dependents and employees is the one-to-many relationship.The employee\_id column in the dependents table references to the employee\_id column in the  employees table.You can use the MINUS operator to find the employees who do not have any dependents. To do this, you subtract the employee\_id result set in the  employees table from the employee\_id result set in the dependents table.The following query illustrates the idea:SELECT   
 employee\_id  
FROM  
 employees   
MINUS   
SELECT   
 employee\_id  
FROM  
 dependents;Code language: SQL (Structured Query Language) (sql)SQL  MINUS with ORDER BY exampleTo sort the result set returned by the MINUS operator, you place the ORDER BY clause at the end of the last SELECT statement.For example, to sort the employees who do not have any dependents, you use the following query:SELECT   
 employee\_id  
FROM  
 employees   
MINUS   
SELECT   
 employee\_id  
FROM  
 dependents  
ORDER BY employee\_id;Code language: SQL (Structured Query Language) (sql)Now you should have a good understanding of the SQL MINUS operator and know how to apply it to compare two result sets.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL INTERSECT Up NextSQL INSERT

# Content from https://www.sqltutorial.org/sql-subquery/

Home » SQL Tutorial » SQL SubquerySQL SubquerySummary: in this tutorial, you will learn about the SQL subquery and how to use the subqueries to form flexible SQL statements.SQL subquery basicConsider the following employees and departments tables from the sample database:Suppose you have to find all employees who locate in the location with the id 1700. You might come up with the following solution.First, find all departments located at the location whose id is 1700:SELECT   
 \*  
FROM  
 departments  
WHERE  
 location\_id = 1700;  
Code language: SQL (Structured Query Language) (sql)Second, find all employees that belong to the location 1700 by using the department id list of the previous query:SELECT   
 employee\_id, first\_name, last\_name  
FROM  
 employees  
WHERE  
 department\_id IN (1 , 3, 8, 10, 11)  
ORDER BY first\_name , last\_name;Code language: SQL (Structured Query Language) (sql)This solution has two problems. To start with, you have looked at the departments table to check which department belongs to the location 1700. However, the original question was not referring to any specific departments; it referred to the location 1700.Because of the small data volume, you can get a list of department easily. However, in the real system with high volume data, it might be problematic.Another problem was that you have to revise the queries whenever you want to find employees who locate in a different location.A much better solution to this problem is to use a subquery. By definition, a subquery is a query nested inside another query such as SELECT, INSERT, UPDATE, or DELETE statement. In this tutorial, we are focusing on the subquery used with the SELECT statement.In this example, you can rewrite combine the two queries above as follows:SELECT   
 employee\_id, first\_name, last\_name  
FROM  
 employees  
WHERE  
 department\_id IN (SELECT   
 department\_id  
 FROM  
 departments  
 WHERE  
 location\_id = 1700)  
ORDER BY first\_name , last\_name;Code language: SQL (Structured Query Language) (sql)The query placed within the parentheses is called a subquery. It is also known as an inner query or inner select. The query that contains the subquery is called an outer query or an outer select.To execute the query, first, the database system has to execute the subquery and substitute the subquery between the parentheses with its result – a number of department id located at the location 1700 – and then executes the outer query.You can use a subquery in many places such as:With the IN or NOT IN operatorWith comparison operatorsWith the EXISTS or NOT EXISTS operatorWith the ANY or ALL operatorIn the FROM clauseIn the SELECT clauseSQL subquery examplesLet’s take some examples of using the subqueries to understand how they work.SQL subquery with the IN or NOT IN operatorIn the previous example, you have seen how the subquery was used with the IN operator. The following example uses a subquery with the NOT IN operator to find all employees who do not locate at the location 1700:SELECT   
 employee\_id, first\_name, last\_name  
FROM  
 employees  
WHERE  
 department\_id NOT IN (SELECT   
 department\_id  
 FROM  
 departments  
 WHERE  
 location\_id = 1700)  
ORDER BY first\_name , last\_name;Code language: SQL (Structured Query Language) (sql)SQL subquery with the comparison operatorThe following syntax illustrates how a subquery is used with a comparison operator:comparison\_operator (subquery)  
Code language: SQL (Structured Query Language) (sql)where the comparison operator is one of these operators:Equal (=)Greater than (>)Less than (<)Greater than or equal ( >=)Less than or equal (<=)Not equal ( !=) or (<>)The following example finds the employees who have the highest salary:SELECT   
 employee\_id, first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 salary = (SELECT   
 MAX(salary)  
 FROM  
 employees)  
ORDER BY first\_name , last\_name;Code language: SQL (Structured Query Language) (sql)In this example, the subquery returns the highest salary of all employees and the outer query finds the employees whose salary is equal to the highest one.The following statement finds all employees who salaries are greater than the average salary of all employees:SELECT   
 employee\_id, first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 salary > (SELECT   
 AVG(salary)  
 FROM  
 employees);   
Code language: SQL (Structured Query Language) (sql)In this example, first, the subquery returns the average salary of all employees. Then, the outer query uses the greater than operator to find all employees whose salaries are greater than the average.SQL subquery with the EXISTS or NOT EXISTS operatorThe EXISTS operator checks for the existence of rows returned from the subquery. It returns true if the subquery contains any rows. Otherwise, it returns false.The syntax of the EXISTS operator is as follows:EXISTS (subquery )  
Code language: SQL (Structured Query Language) (sql)The NOT EXISTS operator is opposite to the EXISTS operator.NOT EXISTS (subquery)  
Code language: SQL (Structured Query Language) (sql)The following example finds all departments which have at least one employee with the salary is greater than 10,000:SELECT   
 department\_name  
FROM  
 departments d  
WHERE  
 EXISTS( SELECT   
 1  
 FROM  
 employees e  
 WHERE  
 salary > 10000  
 AND e.department\_id = d.department\_id)  
ORDER BY department\_name;   
Code language: SQL (Structured Query Language) (sql)Similarly, the following statement finds all departments that do not have any employee with the salary greater than 10,000:SELECT   
 department\_name  
FROM  
 departments d  
WHERE  
 NOT EXISTS( SELECT   
 1  
 FROM  
 employees e  
 WHERE  
 salary > 10000  
 AND e.department\_id = d.department\_id)  
ORDER BY department\_name;   
Code language: SQL (Structured Query Language) (sql)SQL subquery with the ALL operatorThe syntax of the subquery when it is used with the ALL operator is as follows:comparison\_operator ALL (subquery)  
Code language: SQL (Structured Query Language) (sql)The following condition evaluates to true if x is greater than every value returned by the subquery.x > ALL (subquery)  
Code language: SQL (Structured Query Language) (sql)For example, suppose the subquery returns three value one, two, and three. The following condition evaluates to true if x is greater than 3.x > ALL (1,2,3)  
Code language: SQL (Structured Query Language) (sql)The following query uses the GROUP BY clause and MIN() function to find the lowest salary by department:SELECT   
 MIN(salary)  
FROM  
 employees  
GROUP BY department\_id  
ORDER BY MIN(salary) DESC;  
Code language: SQL (Structured Query Language) (sql)The following example finds all employees whose salaries are greater than the lowest salary of every department:SELECT   
 employee\_id, first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 salary >= ALL (SELECT   
 MIN(salary)  
 FROM  
 employees  
 GROUP BY department\_id)  
ORDER BY first\_name , last\_name;Code language: SQL (Structured Query Language) (sql)SQL subquery with the ANY operatorThe following shows the syntax of a subquery with the ANY operator:comparison\_operator ANY (subquery)  
Code language: SQL (Structured Query Language) (sql)For example, the following condition evaluates to true if x is greater than any value returned by the subquery. So the condition x > SOME (1,2,3) evaluates to true if x is greater than 1.x > ANY (subquery)Code language: SQL (Structured Query Language) (sql)Note that the SOME operator is a synonym for the ANY operator so you can use them interchangeably.The following query finds all employees whose salaries are greater than or equal to the highest salary of every department.SELECT   
 employee\_id, first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 salary >= SOME (SELECT   
 MAX(salary)  
 FROM  
 employees  
 GROUP BY department\_id);  
Code language: SQL (Structured Query Language) (sql)In this example, the subquery finds the highest salary of employees in each department. The outer query looks at these values and determines which employee’s salaries are greater than or equal to any highest salary by department.SQL subquery in the FROM clauseYou can use a subquery in the FROM clause of the SELECT statement as follows:SELECT   
 \*  
FROM  
 (subquery) AS table\_name  
Code language: SQL (Structured Query Language) (sql)In this syntax, the table alias is mandatory because all tables in the FROM clause must have a name.Note that the subquery specified in the FROM clause is called a derived table in MySQL or inline view in Oracle.The following statement returns the average salary of every department:SELECT   
 AVG(salary) average\_salary  
FROM  
 employees  
GROUP BY department\_id;  
Code language: SQL (Structured Query Language) (sql)You can use this query as a subquery in the FROM clause to calculate the average of average salary of departments as follows:SELECT   
 ROUND(AVG(average\_salary), 0)  
FROM  
 (SELECT   
 AVG(salary) average\_salary  
 FROM  
 employees  
 GROUP BY department\_id) department\_salary;  
Code language: SQL (Structured Query Language) (sql)SQL Subquery in the SELECT clauseA subquery can be used anywhere an expression can be used in the SELECT clause. The following example finds the salaries of all employees, their average salary, and the difference between the salary of each employee and the average salary.SELECT   
 employee\_id,  
 first\_name,  
 last\_name,  
 salary,  
 (SELECT   
 ROUND(AVG(salary), 0)  
 FROM  
 employees) average\_salary,  
 salary - (SELECT   
 ROUND(AVG(salary), 0)  
 FROM  
 employees) difference  
FROM  
 employees  
ORDER BY first\_name , last\_name;Code language: SQL (Structured Query Language) (sql)Now you should understand what an SQL subquery is and how to use subqueries to form flexible SQL statements.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL CUBE Up NextSQL Correlated Subquery

# Content from https://www.sqltutorial.org/sql-correlated-subquery/

Home » SQL Tutorial » SQL Correlated SubquerySQL Correlated SubquerySummary: in this tutorial, you will learn about the SQL correlated subquery which is a subquery that uses values from the outer query.Introduction to SQL correlated subqueryLet’s start with an example.See the following employees table in the sample database:The following query finds employees whose salary is greater than the average salary of all employees:SELECT   
 employee\_id,   
 first\_name,   
 last\_name,   
 salary  
FROM  
 employees  
WHERE  
 salary > (SELECT   
 AVG(salary)  
 FROM  
 employees);  
Code language: SQL (Structured Query Language) (sql)In this example, the subquery is used in the WHERE clause. There are some points that you can see from this query:First, you can execute the subquery that returns the average salary of all employees independently.SELECT   
 AVG(salary)  
FROM  
 employees;  
Code language: SQL (Structured Query Language) (sql)Second, the database system needs to evaluate the subquery only once.Third, the outer query makes use of the result returned from the subquery. The outer query depends on the subquery for its value. However, the subquery does not depend on the outer query. Sometimes, we call this subquery is a plain subquery.Unlike a plain subquery, a correlated subquery is a subquery that uses the values from the outer query. Also, a correlated subquery may be evaluated once for each row selected by the outer query. Because of this, a query that uses a correlated subquery may be slow.A correlated subquery is also known as a repeating subquery or a synchronized subquery.SQL correlated subquery examplesLet’s see few more examples of the correlated subqueries to understand them better.SQL correlated subquery in the WHERE clause exampleThe following query finds all employees whose salary is higher than the average salary of the employees in their departments:SELECT   
 employee\_id,   
 first\_name,   
 last\_name,   
 salary,   
 department\_id  
FROM  
 employees e  
WHERE  
 salary > (SELECT   
 AVG(salary)  
 FROM  
 employees  
 WHERE  
 department\_id = e.department\_id)  
ORDER BY   
 department\_id ,   
 first\_name ,   
 last\_name;  
Code language: SQL (Structured Query Language) (sql)Here is the output:In this example, the outer query is:SELECT   
 employee\_id,   
 first\_name,   
 last\_name,   
 salary,   
 department\_id  
FROM  
 employees e  
WHERE  
 salary >  
...  
Code language: SQL (Structured Query Language) (sql)and the correlated subquery is:SELECT  
 AVG( list\_price )  
FROM  
 products  
WHERE  
 category\_id = p.category\_id  
Code language: SQL (Structured Query Language) (sql)For each employee, the database system has to execute the correlated subquery once to calculate the average salary of the employees in the department of the current employee.SQL correlated subquery in the SELECT clause exampleThe following query returns the employees and the average salary of all employees in their departments:SELECT   
 employee\_id,  
 first\_name,  
 last\_name,  
 department\_name,  
 salary,  
 (SELECT   
 ROUND(AVG(salary),0)  
 FROM  
 employees  
 WHERE  
 department\_id = e.department\_id) avg\_salary\_in\_department  
FROM  
 employees e  
 INNER JOIN  
 departments d ON d.department\_id = e.department\_id  
ORDER BY   
 department\_name,   
 first\_name,   
 last\_name;  
Code language: SQL (Structured Query Language) (sql)The output is:For each employee, the database system has to execute the correlated subquery once to calculate the average salary by the employee’s department.SQL correlated subquery with EXISTS operator exampleWe often use a correlated subquery with the EXISTS operator. For example, the following query returns all employees who have no dependents:SELECT   
 employee\_id,   
 first\_name,   
 last\_name  
FROM  
 employees e  
WHERE  
 NOT EXISTS( SELECT   
 \*  
 FROM  
 dependents d  
 WHERE  
 d.employee\_id = e.employee\_id)  
ORDER BY first\_name ,   
 last\_name;  
Code language: SQL (Structured Query Language) (sql)The following picture shows the output:In this tutorial, you have learned about the SQL correlated subquery and how to apply it to form a complex query.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL Subquery Up NextSQL ALL

# Content from https://www.sqltutorial.org/sql-exists/

Home » SQL Tutorial » SQL EXISTSSQL EXISTSSummary: in this tutorial, you will learn how to use the SQL EXISTS operator to test if a subquery contains any rows.Introduction to the SQL EXISTS operatorThe EXISTS operator allows you to specify a subquery to test for the existence of rows. The following illustrates the syntax of the EXISTS operator:EXISTS (subquery)  
Code language: SQL (Structured Query Language) (sql)The EXISTS operator returns true if the subquery contains any rows. Otherwise, it returns false.The EXISTS operator terminates the query processing immediately once it finds a row, therefore, you can leverage this feature of the EXISTS operator to improve the query performance.SQL EXISTS operator exampleWe will use the  employees and dependents tables in the sample database for the demonstration.The following statement finds all employees who have at least one dependent:SELECT   
 employee\_id, first\_name, last\_name  
FROM  
 employees  
WHERE  
 EXISTS( SELECT   
 1  
 FROM  
 dependents  
 WHERE  
 dependents.employee\_id = employees.employee\_id);  
Code language: SQL (Structured Query Language) (sql)The subquery is correlated. For each row in the  employees table, the subquery checks if there is a corresponding row in the dependents table. If yes, then the subquery returns one which makes the outer query to include the current row in the  employees table. If there is no corresponding row, then the subquery returns no row that causes the outer query to not include the current row in the  employees table in the result set.SQL NOT EXISTSTo negate the EXISTS operator, you use the NOT operator as follows:NOT EXISTS (subquery)  
Code language: SQL (Structured Query Language) (sql)For example, the following query finds employees who do not have any dependents:SELECT   
 employee\_id, first\_name, last\_name  
FROM  
 employees  
WHERE  
 NOT EXISTS( SELECT   
 1  
 FROM  
 dependents  
 WHERE  
 dependents.employee\_id = employees.employee\_id);  
Code language: SQL (Structured Query Language) (sql)The following screenshot illustrates the result:SQL EXISTS and NULLIf the subquery returns NULL, the EXISTS operator still returns the result set. This is because the EXISTS operator only checks for the existence of row returned by the subquery. It does not matter if the row is NULL or not.In the following example, the subquery returns NULL but the EXISTS operator still evaluates to true:SELECT   
 employee\_id, first\_name, last\_name  
FROM  
 employees  
WHERE  
 EXISTS( SELECT NULL)  
ORDER BY first\_name , last\_name;   
Code language: SQL (Structured Query Language) (sql)The query returns all rows in the  employees table.In this tutorial, you have learned how to use the SQL EXISTS operator to test for the existence of rows returned by a subquery.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL ANY Up NextSQL UNION

# Content from https://www.sqltutorial.org/sql-all/

Home » SQL Tutorial » SQL ALLSQL ALLSummary: in this tutorial, you will learn about the SQL ALL operator and how to use it to compare a value with a set of values.Introduction to the SQL ALL operatorThe SQL ALL operator is a logical operator that compares a single value with a single-column set of values returned by a subquery.The following illustrates the syntax of the SQL ALL operator:WHERE column\_name comparison\_operator ALL (subquery)  
Code language: SQL (Structured Query Language) (sql)The SQL ALL operator must be preceded by a comparison operator such as >, >=, <, <=, <>, = and followed by a subquery. Some database systems such as Oracle allow a list of literal values instead of a subquery.Note that if the subquery returns no row, the condition in the WHERE clause is always true. Assuming that the subquery returns one or more rows, the following table illustrates the meaning of the SQL ALL operator:ConditionMeaning c > ALL(…) The values in column c must greater than the biggest value in the set to evaluate to true. c >= ALL(…) The values in column c must greater than or equal to the biggest value in the set to evaluate to true. c < ALL(…) The values in column c must be less than the lowest value in the set to evaluate to true. c >= ALL(…) The values in column c must be less than or equal to the lowest value in the set to evaluate to true. c <> ALL(…) The values in column c must not be equal to any value in the set to evaluate to true. c = ALL(…) The values in column c must be equal to any value in the set to evaluate to true.SQL ALL examplesWe will use the employees table from the sample database for the demonstration:SQL ALL with the greater than operatorThe following query finds rows whose values in the column\_name are greater than the biggest values returned by the subquery:SELECT   
 \*  
FROM  
 table\_name  
WHERE  
 column\_name > ALL (subquery);  
Code language: SQL (Structured Query Language) (sql)For example, the following statement finds all employees whose salaries are greater than the highest salary of employees in the Marketing department whose id is 2:SELECT   
 first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 salary > ALL (SELECT   
 salary  
 FROM  
 employees  
 WHERE  
 department\_id = 2)  
ORDER BY salary;Code language: SQL (Structured Query Language) (sql)Let’s verify it by querying the highest salary of employees in department 2:SELECT   
 MAX(salary)  
FROM  
 employees  
WHERE  
 department\_id = 2;   
Code language: SQL (Structured Query Language) (sql)This query returned 13,000 which is lower than any salary that returned by the query which used the ALL operator above.SQL ALL with the greater than or equal to operatorThe following shows the syntax of the SQL ALL operator with the greater than or equal to operator:SELECT   
 \*  
FROM  
 table\_name  
WHERE  
 column\_name >= ALL (subquery);  
Code language: SQL (Structured Query Language) (sql)The query returns all rows whose values in the column\_name are greater than or equal to all the values returned by the subquery.For example, the following query finds all employees whose salaries are greater than or equal to the highest salary of employees in the Marketing department:SELECT   
 first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 salary >= ALL (SELECT   
 salary  
 FROM  
 employees  
 WHERE  
 department\_id = 2)  
ORDER BY salary;   
Code language: SQL (Structured Query Language) (sql)As shown clearly in the screenshot, the salary of Michael is 13,000 which is equal to the highest salary of employees in the Marketing department is included in the result set.SQL ALL with the less than operatorThe following illustrates the ALL operator used with the less than operator:SELECT   
 \*  
FROM  
 table\_name  
WHERE  
 column\_name < ALL (subquery);  
Code language: SQL (Structured Query Language) (sql)This query returns all rows whose values in the column\_name are smaller than the smallest values returned by the subquery.The following statement finds the lowest salary of employees in the Marketing department:SELECT   
 MIN(salary)  
FROM  
 employees  
WHERE  
 department\_id = 2;  
Code language: SQL (Structured Query Language) (sql)To find all employees whose salaries are less than the lowest salary of employees in the Marketing department, you use the ALL operator with the less than operator as follows:SELECT   
 first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 salary < ALL (SELECT   
 salary  
 FROM  
 employees  
 WHERE  
 department\_id = 2)  
ORDER BY salary DESC;  
Code language: SQL (Structured Query Language) (sql)The result is:SQL ALL with the less than or equal to operatorThe following shows the syntax of the ALL operator used with the less than or equal to operator:SELECT   
 \*  
FROM  
 table\_name  
WHERE  
 column\_name <= ALL (subquery);  
Code language: SQL (Structured Query Language) (sql)For example, the following statement finds all employees whose salaries are less than or equal to the lowest salary of employees in the Marketing department:SELECT   
 first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 salary <= ALL (SELECT   
 salary  
 FROM  
 employees  
 WHERE  
 department\_id = 2)  
ORDER BY salary DESC;   
Code language: SQL (Structured Query Language) (sql)SQL ALL with the not equal to operatorThe following query returns all rows whose values in the column\_name are not equal to any values returned by the subquery:SELECT   
 \*  
FROM  
 table\_name  
WHERE  
 column\_name <> ALL (subquery);  
Code language: SQL (Structured Query Language) (sql)For example, to find employees whose salaries are not equal to the average salary of every department, you use the query below:SELECT   
 first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 salary <> ALL (SELECT   
 AVG(salary)  
 FROM  
 employees  
 GROUP BY department\_id)  
ORDER BY salary DESC;   
Code language: SQL (Structured Query Language) (sql)Notice that the subquery finds the average salary of employees by the department by using the AVG() function and the GROUP BY clause.SQL ALL with the equal to operatorWhen you use the ALL operator with the equal to operator, the query finds all rows whose values in the column\_name are equal to any values returned by the subquery:SELECT   
 \*  
FROM  
 table\_name  
WHERE  
 column\_name = ALL (subquery);  
Code language: SQL (Structured Query Language) (sql)The following example finds all employees whose salaries are equal to the highest salary of employees in the Marketing department:SELECT   
 first\_name, last\_name, salary  
FROM  
 employees  
WHERE  
 salary = ALL (SELECT   
 MAX(salary)  
 FROM  
 employees  
 WHERE  
 department\_id = 2);  
Code language: SQL (Structured Query Language) (sql)In this tutorial, you have learned how to use the SQL ALL operator to test whether a value matches a set of values returned by a subquery.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL Correlated Subquery Up NextSQL ANY

# Content from https://www.sqltutorial.org/sql-any/

Home » SQL Tutorial » SQL ANYSQL ANYSummary: in this tutorial, you will learn about the SQL ANY operator and how to use it to compare a value with a set of values.Introduction to the SQL ANY operatorThe ANY operator is a logical operator that compares a value with a set of values returned by a subquery. The ANY operator must be preceded by a comparison operator >, >=, <, <=, =, <> and followed by a subquery.The following illustrates the syntax of the ANY operator:WHERE column\_name comparison\_operator ANY (subquery)  
Code language: SQL (Structured Query Language) (sql)If the subquery returns no row, the condition evaluates to false. Suppose the subquery does not return zero rows, the following illustrates the meaning of the ANY operator when it is used with each comparison operator:ConditionMeaningx = ANY (…) The values in column c must match one or more values in the set to evaluate to true.x != ANY (…) The values in column c must not match one or more values in the set to evaluate to true.x > ANY (…) The values in column c must be greater than the smallest value in the set to evaluate to true.x < ANY (…) The values in column c must be smaller than the biggest value in the set to evaluate to true.x >= ANY (…) The values in column c must be greater than or equal to the smallest value in the set to evaluate to true.x <= ANY (…) The values in column c must be smaller than or equal to the biggest value in the set to evaluate to true.SQL ANY examplesFor the demonstration, we will use the employees table from the sample database:SQL ANY with equal to operator exampleThe following statement uses the AVG() function and GROUP BY clause to find the average salary of each department:SELECT   
 ROUND(AVG(salary), 2)  
FROM  
 employees  
GROUP BY   
 department\_id  
ORDER BY   
 AVG(salary) DESC;  
Code language: SQL (Structured Query Language) (sql)To find all employees whose salaries are equal to the average salary of their department, you use the following query:SELECT   
 first\_name,   
 last\_name,   
 salary  
FROM  
 employees  
WHERE  
 salary = ANY (  
 SELECT   
 AVG(salary)  
 FROM  
 employees  
 GROUP BY   
 department\_id)  
ORDER BY   
 first\_name,   
 last\_name,  
 salary;   
Code language: SQL (Structured Query Language) (sql)Using SQL ANY with the not equal to operator exampleSimilarly, the following query finds all employees whose salaries are not equal to the average salary of every department:SELECT   
 first\_name,   
 last\_name,   
 salary  
FROM  
 employees  
WHERE  
 salary <> ANY (SELECT   
 AVG(salary)  
 FROM  
 employees  
 GROUP BY department\_id)  
ORDER BY   
 first\_name,   
 last\_name,   
 salary;   
Code language: SQL (Structured Query Language) (sql)Using SQL ANY with the greater than operator exampleThe following query finds all employees whose salaries are greater than the average salary in every department:SELECT   
 first\_name,   
 last\_name,   
 salary  
FROM  
 employees  
WHERE  
 salary > ANY (SELECT   
 AVG(salary)  
 FROM  
 employees  
 GROUP BY department\_id)  
ORDER BY   
 salary;   
Code language: SQL (Structured Query Language) (sql)Note that the lowest average salary is 4,150. The query above returns all employees whose salaries are greater than the lowest salary.Using SQL ANY with the greater than or equal to operator exampleThe following statement returns all employees whose salaries are greater than or equal to the average salary in every department:SELECT   
 first\_name,   
 last\_name,   
 salary  
FROM  
 employees  
WHERE  
 salary >= ANY (SELECT   
 AVG(salary)  
 FROM  
 employees  
 GROUP BY department\_id)  
ORDER BY first\_name , last\_name , salary;   
Code language: SQL (Structured Query Language) (sql)Using SQL ANY with the less than operator exampleThe following query finds all employees whose salaries are less than the average salary in every department:SELECT   
 first\_name,   
 last\_name,   
 salary  
FROM  
 employees  
WHERE  
 salary < ANY (SELECT   
 AVG(salary)  
 FROM  
 employees  
 GROUP BY department\_id)  
ORDER BY salary DESC;   
Code language: SQL (Structured Query Language) (sql)In this example, employees whose salaries are smaller than the highest average salary in every department:Using SQL ANY with the less than or equal to operator exampleTo find employees whose salaries are less than or equal to the average salary in every department, you use the following query:SELECT   
 first\_name,   
 last\_name,   
 salary  
FROM  
 employees  
WHERE  
 salary <= ANY (SELECT   
 AVG(salary)  
 FROM  
 employees  
 GROUP BY department\_id)  
ORDER BY salary DESC;   
Code language: SQL (Structured Query Language) (sql)As shown in the screenshot, the result set includes the employees whose salaries are lower than or equal to the highest average salary in every department.Now you should know how to use the SQL ANY operator to form condition by comparing a value with a set of values.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL ALL Up NextSQL EXISTS

# Content from https://www.sqltutorial.org/sql-insert/

Home » SQL Tutorial » SQL INSERTSQL INSERTSummary: in this tutorial, you will learn how to insert data into a table using the SQL INSERT statement.Introduction to the SQL INSERT statementSQL provides the INSERT statement that allows you to insert one or more rows into a table. The INSERT statement allows you to:Insert a single row into a tableInsert multiple rows into a tableCopy rows from a table to another table.We will examine each function of the INSERT statement in the following sections.Insert one row into a tableTo insert one row into a table, you use the following syntax of the INSERT statement.INSERT INTO table1 (column1, column2,...)  
VALUES  
 (value1, value2,...);Code language: SQL (Structured Query Language) (sql)There are some points that you should pay attention to when you insert a new row into a table:First, the number of values must be the same as the number of columns. In addition, the columns and values must be the correspondent because the database system will match them by their relative positions in the lists.Second, before adding a new row, the database system checks for all integrity constraints e.g., foreign key constraint, primary key constraint, check constraint and not null constraint. If one of these constraints is violated, the database system will issue an error and terminate the statement without inserting any new row into the table.It is not necessary to specify the columns if the sequence of values matches the order of the columns in the table. See the following INSERT statement that omits the column list in the INSERT INTO clause.INSERT INTO table1  
VALUES  
 (value1, value2,...);Code language: SQL (Structured Query Language) (sql)However, this is not considering as a good practice.If you don’t specify a column and its value in the INSERT statement when you insert a new row, that column will take a default value specified in the table structure. The default value could be 0, a next integer value in a sequence, the current time, a NULL value, etc. See the following statement:INSERT INTO (column1, column3)  
VALUES  
 (column1, column3);Code language: SQL (Structured Query Language) (sql)In this syntax, the column2 will take a default value.Insert one row into a table exampleWe will use the employees and dependents tables in the sample database to show you how to insert one row into the table.To insert a new row into the dependents table.INSERT INTO dependents (  
 first\_name,  
 last\_name,  
 relationship,  
 employee\_id  
)  
VALUES  
 (  
 'Dustin',  
 'Johnson',  
 'Child',  
 178  
 );Code language: SQL (Structured Query Language) (sql)Try ItWe did not use the department\_id column in the INSERT statement because the dependent\_id column is an auto-increment column, therefore, the database system uses the next integer number as the default value when you insert a new row.The employee\_id column is a foreign key that links the dependents table to the  employees table. Before adding the new rows, the database system checks if the value 178 exists in the employee\_id column of the  employees table to make sure that the foreign key constraint is not violated.If the row is inserted successfully, the database system returned the number of the affected rows.Affected rows: 1Code language: SQL (Structured Query Language) (sql)You can check whether the row has been inserted successfully or not by using the following SELECT statement.SELECT   
 \*  
FROM  
 dependents  
WHERE  
 employee\_id = 178;Code language: SQL (Structured Query Language) (sql)Try ItInsert multiple rows into a tableTo insert multiple rows using a single INSERT statement, you use the following construct:INSERT INTO table1  
VALUES  
 (value1, value2,...),  
 (value1, value2,...),  
 (value1, value2,...),  
 ...;Code language: SQL (Structured Query Language) (sql)For example, to insert two rows into the dependents table, you use the following query.INSERT INTO dependents (  
 first\_name,  
 last\_name,  
 relationship,  
 employee\_id  
)  
VALUES  
 (  
 'Cameron',  
 'Bell',  
 'Child',  
 192  
 ),  
 (  
 'Michelle',  
 'Bell',  
 'Child',  
 192  
 );Code language: SQL (Structured Query Language) (sql)Try ItThe database system returns 2 rows affected. You can verify the result using the following statement.SELECT   
 \*  
FROM  
 dependents  
WHERE  
 employee\_id = 192;Code language: SQL (Structured Query Language) (sql)Try ItCopy rows from other tablesYou can use the INSERT statement to query data from one or more tables and insert it into another table as follows:INSERT INTO table1 (column1, column2)   
SELECT  
 column1,  
 column2  
FROM  
 table2  
WHERE  
 condition1;Code language: SQL (Structured Query Language) (sql)In this syntax, you use a SELECT which is called a subselect instead of the  VALUES clause . The subselect can contain the joins so that you can combine data from multiple tables. When executing the statement, the database system evaluates the subselect first before inserting data.Suppose, you have a table named dependents\_archive that has the same structure as the dependents table. The following statement copies all rows from the dependents table to the dependents\_archive table.INSERT INTO dependents\_archive   
SELECT  
 \*  
FROM  
 dependents;Code language: SQL (Structured Query Language) (sql)Try ItYou can verify the insert operation by using the following statement.SELECT  
 \*  
FROM  
 dependents\_archive;Code language: SQL (Structured Query Language) (sql)Try ItNow you should know how to use the SQL INSERT statement to insert one or more rows into a table.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL MINUS Up NextSQL UPDATE

# Content from https://www.sqltutorial.org/sql-update/

Home » SQL Tutorial » SQL UPDATESQL UPDATESummary: in this tutorial, you will learn how to use the SQL UPDATE statement to modify data of the existing rows a table.Introduction to the SQL UPDATE statementTo change existing data in a table, you use the UPDATE statement. The following shows the syntax of the UPDATE statement:UPDATE table\_name  
SET column1 = value1,  
 column2 = value2  
WHERE  
 condition;Code language: SQL (Structured Query Language) (sql)In this syntax:First, indicate the table that you want to update in the UPDATE clause.Second, specify the columns that you want to modify in the SET clause. The columns that are not listed in the SET clause will retain their original values.Third, specify which rows to update in the WHERE clause.The UPDATE statement affects one or more rows in a table based on the condition in the WHERE clause.  For example, if the WHERE clause contains a primary key expression, the UPDATE statement changes one row only.However, any row that causes the condition in the WHERE to evaluate to true will be modified. Because the WHERE clause is optional, therefore, if you omit it, the all the rows in the table will be affected.SQL UPDATE statement examplesWe will use the employees and dependents table to demonstrate the UPDATE statement.SQL UPDATE one row exampleSuppose the employee id 192 Sarah Bell changed her last name from Bell to Lopez and you need to update her record in the  employees table.To update Sarah’s last name from  Bell to Lopez, you use the following UPDATE statement:UPDATE employees   
SET   
 last\_name = 'Lopez'  
WHERE  
 employee\_id = 192;Code language: SQL (Structured Query Language) (sql)Try ItThe database system updated value in the last\_name column and the row with  employee\_id 192.You can verify it by using the following SELECT statement.SELECT  
 employee\_id,  
 first\_name,  
 last\_name  
FROM  
 employees  
WHERE  
 employee\_id = 192;Code language: SQL (Structured Query Language) (sql)Try ItSQL UPDATE multiple rows exampleNow, Nancy wants to change all her children’s last names from Bell to Lopez. In this case, you need to update all Nancy’s dependents in the dependents table.Before updating the data, let’s check the dependents of Nancy.SELECT  
 \*  
FROM  
 dependents  
WHERE  
 employee\_id = 192;Code language: SQL (Structured Query Language) (sql)Try ItTo update the last names of Nancy’s dependents, you use the following UPDATE statement.UPDATE dependents   
SET   
 last\_name = 'Lopez'  
WHERE  
 employee\_id = 192;Code language: SQL (Structured Query Language) (sql)Try ItSQL UPDATE with subquery exampleSometimes when employees change their last names, you update the  employees table only without updating the dependents table.To make sure that the last names of children are always matched with the last name of parents in the  employees table, you use the following statement:UPDATE dependents  
SET last\_name = (  
 SELECT  
 last\_name  
 FROM  
 employees  
 WHERE  
 employee\_id = dependents.employee\_id  
);Code language: SQL (Structured Query Language) (sql)Try ItBecause the WHERE clause is omitted, the UPDATE statement updated all rows in the dependents table.In the SET clause, instead of using the literal values, we used a subquery to get the corresponding last name value from the  employees table.In this tutorial, we have shown you how to use the SQL UPDATE statement to modify existing data in a table.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL INSERT Up NextSQL DELETE

# Content from https://www.sqltutorial.org/sql-delete/

Home » SQL Tutorial » SQL DELETESQL DELETESummary: in this tutorial, you will learn how to use the SQL DELETE statement to delete one or more rows in a table.Introduction to SQL DELETE statementTo remove one or more rows from a table, you use the DELETE statement. The general syntax for the DELETE statement is as follows:DELETE  
FROM  
 table\_name  
WHERE  
 condition;Code language: SQL (Structured Query Language) (sql)First, provide the name of the table where you want to remove rows.Second, specify the condition in the WHERE clause to identify the rows that need to be deleted. If you omit the WHERE clause all rows in the table will be deleted. Therefore, you should always use the DELETE statement with caution.Generally speaking, the DELETE statement does not return a result set as the SELECT statement. However, it does return the number of rows deleted.SQL DELETE statement examplesWe will use the employees and dependents tables to demonstrate the DELETE statement.SQL DELETE one row in a tableSuppose David, who has employee id 105, wants to remove Fred from his dependent list. We know that Fred has the dependent id 16, so we use the following DELETE statement to remove Fred from the dependents table.DELETE FROM dependents   
WHERE  
 dependent\_id = 16;Code language: SQL (Structured Query Language) (sql)Try ItBecause the WHERE clause contains the primary key expression that identifies Fred, the DELETE statement removes just one row.You can verify that the row with the dependent id 16 has been deleted by using the following statement:SELECT   
 COUNT(\*)  
FROM  
 dependents  
WHERE  
 dependent\_id = 16;Code language: SQL (Structured Query Language) (sql)Try ItSQL DELETE multiple rows exampleTo delete multiple rows in a table, you use the condition in the WHERE clause to identify the rows that should be deleted. For example, the following statement uses the IN operator to include the dependents of the employees with the id is 100, 101, or 102.DELETE FROM dependents   
WHERE  
 employee\_id IN (100 , 101, 102);Code language: SQL (Structured Query Language) (sql)Try ItSQL DELETE rows from related tablesOne employee may have zero or many dependents while one dependent belongs to only one employee. The employee\_id column in the dependents table links to the employee\_id column in the employees table.The relationship between the employees and dependents tables is one-to-many.Logically, a dependent cannot exist without referring to an employee. In other words, when you delete an employee, his or her dependents must be deleted as well.For example, to remove the employee id 192 and all  the employee’s dependents, you need to execute two DELETE statements as follows:DELETE  
FROM  
 employees  
WHERE  
 employee\_id = 192;  
  
DELETE  
FROM  
 dependents  
WHERE  
 employee\_id = 192;Code language: SQL (Structured Query Language) (sql)Try ItMost database systems support the foreign key constraint so that when one row from a table is deleted, the rows in the foreign key tables are also removed automatically.Therefore, when the following DELETE statement is executed:DELETE  
FROM  
 employees  
WHERE  
 employee\_id = 192;Code language: SQL (Structured Query Language) (sql)Try ItAll the rows with employee\_id 192 are also removed automatically.To remove all rows from a table more efficiently, you use the TRUNCATE TABLE statement instead of using the DELETE statement without a WHERE clause.Now you should understand the SQL DELETE statement and how to apply it to remove one or more rows in a table.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL UPDATE Up NextSQL CASE

# Content from https://www.sqltutorial.org/sql-create-table/

Home » SQL Tutorial » SQL CREATE TABLESQL CREATE TABLESummary: in this tutorial, you will learn how to use the SQL CREATE TABLE statement to create new tables.Introduction to SQL CREATE TABLE statementSo far, you have learned various ways to query data from one or more table in the sample database. It is time to learn how to create your own tables.A table is a collection of data stored in a database. A table consists of columns and rows. To create a new table, you use the CREATE TABLE statement with the following syntax:CREATE TABLE table\_name(  
 column\_name\_1 data\_type default value column\_constraint,  
 column\_name\_2 data\_type default value column\_constraint,  
 ...,  
 table\_constraint  
);Code language: SQL (Structured Query Language) (sql)The minimum required information for creating a new table is a table name and a column name.The name of the table, given by the table\_name, must be unique within the database. If you create a table whose name is the same as the one that already exists, the database system will issue an error.In the CREATE TABLE statement, you specify a comma-separated list of column definitions. Each column definition is composed of a column name, column’s data type, a default value, and one or more column constraints.The data type of a column specifies the type of data that column can store. The data type of the column can be the numeric, characters, date, etc.The column constraint controls what kind of value that can be stored in the column. For example, the NOT NULL constraint ensures that the column does not contain any NULL value.A column may have multiple column constraints. For example, the username column of the  users table can have both NOT NULL and UNIQUE constraints.In case a constraint contains multiple columns, you use the table constraint. For example, if a table has the primary key that consists of two columns, in this case, you have to use the PRIMARY KEY table constraint.SQL CREATE TABLE examplesSuppose you have to store the training data of employees in the database with a requirement that each employee may take zero or many training courses, and each training course may be taken by zero or many employees.You looked at the current database and found no place to store this information, therefore, you decided to create new tables.The following statement creates the courses table:CREATE TABLE courses (  
 course\_id INT AUTO\_INCREMENT PRIMARY KEY,  
 course\_name VARCHAR(50) NOT NULL  
);Code language: SQL (Structured Query Language) (sql)Try ItThe courses table has two columns: course\_id and course\_name.The course\_id is the primary key column of the courses table. Each table has one and only one primary key that uniquely identifies each row in the table. It is a good practice to define a primary key for every table.The data type of the course\_id is integer denoted by the INT keyword. In addition, the value of the course\_id column is AUTO\_INCREMENT. It means that when you insert a new row into the courses table without providing the value for the course\_id column, the database system will generate an integer value for the column.The course\_name stores the names of courses. Its data type is the character string ( VARCHAR) with maximum length is 50. The NOT NULL constraint ensures that there is no NULL values stored in the course\_name column.Now you have the table to store the course data. To store the training data, you create a new table named training as follows.CREATE TABLE trainings (  
 employee\_id INT,  
 course\_id INT,  
 taken\_date DATE,  
 PRIMARY KEY (employee\_id , course\_id)  
);Code language: SQL (Structured Query Language) (sql)Try ItThe trainings table consists of three columns:The employee\_id column store the id of employees who took the course.The course\_id column store the course that employee took.The taken\_date column stores the date when the employee took the course.Because the primary key of the trainings table consists of two columns: employee\_id and course\_id, we had to use the PRIMARY KEY table constraint.In this tutorial, you have learned how to use the SQL CREATE TABLE statement to create new a new table in the database.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL Data Types Up NextSQL Identity

# Content from https://www.sqltutorial.org/sql-alter-table/

Home » SQL Tutorial » SQL ALTER TABLESQL ALTER TABLESummary: this tutorial shows you how to use the SQL ALTER TABLE to change the structure of existing tables in the database.Once you create a new table, you may want to change its structure because business requirements change. To modify the structure of a table, you use the ALTER TABLE statement. The ALTER TABLE statement allows you to perform the following operations on an existing table:Add a new column using the ADD clause.Modify attribute of a column such as constraint, default value, etc. using the MODIFY clause.Remove columns using the DROP clause.We will examine each operation in detail in the following sections.SQL ALTER TABLE ADD columnThe following statement illustrates the ALTER TABLE with the ADD clause that allows you to add one or more columns to a table.ALTER TABLE table\_name  
ADD new\_colum data\_type column\_constraint [AFTER existing\_column];Code language: SQL (Structured Query Language) (sql)To add one or more columns to a table, you need to perform the following steps:First, specify the table that you want to add column denoted by the table\_name after the ALTER TABLE clause.Second, place the new column definition after the ADD clause. If you want to specify the order of the new column in the table, you can use the optional clause AFTER existing\_column.Note that if you omit the AFTER clause, all the new columns will be added after the last column of the table.Let’s look at some examples of adding new columns to the courses table that we created in the create table tutorialThe following statement adds a new column named credit\_hours to the courses table.ALTER TABLE courses ADD credit\_hours INT NOT NULL;Code language: SQL (Structured Query Language) (sql)You can add multiple columns to a table using a single ALTER TABLE statement. For example, The following statement adds the fee and max\_limit columns to the courses table and places these columns after the course\_name column.ALTER TABLE courses   
ADD fee NUMERIC (10, 2) AFTER course\_name,  
ADD max\_limit INT AFTER course\_name;Code language: SQL (Structured Query Language) (sql)SQL ALTER TABLE MODIFY columnThe MODIFY clause allows you to change some attributes of the existing column e.g., NOT NULL ,UNIQUE, and data type.The following statement shows you the syntax of the ALTER TABLE statement with the DROP clause.ALTER TABLE table\_name  
MODIFY column\_definition;Code language: SQL (Structured Query Language) (sql)Notice that you should modify the attributes of columns of a table that has no data. Because changing the attributes of a column in a table that already has data may result in permanent data loss.For example, if the data type of the column is VARCHAR, and you change it to INT, the database system has to convert the data from VARCHAR to INT. If the conversion fails, the database system may use the default value of the column, which may not be what you expected.The following ALTER TABLE MODIFY statement changes the attribute of the fee column to NOT NULL.ALTER TABLE courses   
MODIFY fee NUMERIC (10, 2) NOT NULL;Code language: SQL (Structured Query Language) (sql)SQL ALTER TABLE DROP columnsWhen a column of a table is obsolete and not used by any other database objects such as triggers, views, stored and stored procedures, you need to remove it from the table.To remove one or more columns, you use the following syntax:ALTER TABLE table\_name  
DROP column\_name,  
DROP colum\_name,  
...Code language: SQL (Structured Query Language) (sql)For example, to remove the fee column of the courses table, you use the following statement.ALTER TABLE courses DROP COLUMN fee;Code language: SQL (Structured Query Language) (sql)To remove more than one column at the same time, you use multiple DROP COLUMN clauses separated by a comma (,).For example, the following statement removes the max\_limit and credit\_hours of the courses table.ALTER TABLE courses   
DROP COLUMN max\_limit,  
DROP COLUMN credit\_hours;Code language: SQL (Structured Query Language) (sql)In this tutorial, you have learned step by step how to use the SQL ALTER TABLE statement to add, modify, and drop one or more columns in a table.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL Auto Increment Up NextSQL ADD COLUMN

# Content from https://www.sqltutorial.org/sql-drop-table/

Home » SQL Tutorial » SQL DROP TABLESQL DROP TABLESummary: in this tutorial, you will learn how to use the SQL DROP TABLE statement to remove one or more tables in a database.Introduction to SQL DROP TABLE statementAs the database evolves, we will need to remove obsolete or redundant tables from the database. To delete a table, we use the DROP TABLE statement.The following illustrates the syntax of the DROP TABLE statement.DROP TABLE [IF EXISTS] table\_name;Code language: SQL (Structured Query Language) (sql)To drop an existing table, you specify the name of the table after the DROP TABLE clause. If the table that is being dropped does not exist, the database system issues an error.To prevent the error of removing a nonexistent table, we use the optional clause IF EXISTS. If we use the IF EXISTS option, the database system will not throw any error if we remove a non-existent table. Some database systems throw a warning or a notice instead.Notice that not all database systems support the IF EXISTS option. The ones that do support the IF EXISTS option are MySQL, PostgreSQL, and SQL Server 2016.The DROP TABLE statement removes both data and structure of a table permanently. Some database systems require the table must be empty before it can be removed from the database. This helps you prevent accidentally deleting a table that is still in use.To delete all data in a table, you can use either the DELETE or TRUNCATE TABLE statement.To drop a table that is referenced by a foreign key constraint of another table, you must disable or remove the foreign constraint before removing the table.SQL DROP TABLE examplesLet’s create a new table for practicing the DROP TABLE statement.The following statement creates a new table named emergency\_contacts that stores the emergency contacts of employees.CREATE TABLE emergency\_contacts (  
 id INT AUTO\_INCREMENT PRIMARY KEY,  
 first\_name VARCHAR(50) NOT NULL,  
 last\_name VARCHAR(50) NOT NULL,  
 relationship VARCHAR(50) NOT NULL,  
 employee\_id INT NOT NULL  
);Code language: SQL (Structured Query Language) (sql)The following statement drops the emergency\_contacts table:DROP TABLE emergency\_contacts;Code language: SQL (Structured Query Language) (sql)SQL DROP TABLE – removing multiples tablesThe DROP TABLE statement allows you to remove multiple tables at the same time. To do this, you need to specify a list of comma-separated tables after the DROP TABLE clause as follows:DROP TABLE table\_name1,table\_name2,...;Code language: SQL (Structured Query Language) (sql)The database system then deletes all tables one by one.SummaryUse SQL DROP TABLE statement to delete one or more tables from the database.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL DROP COLUMN Up NextSQL TRUNCATE TABLE

# Content from https://www.sqltutorial.org/sql-truncate-table/

Home » SQL Tutorial » SQL TRUNCATE TABLESQL TRUNCATE TABLESummary: in this tutorial, you will learn how to use the SQL TRUNCATE TABLE statement to remove all data in a table efficiently and fast.Introduction to the SQL TRUNCATE TABLE statementTo delete all data from a table, you use the DELETE statement without a WHERE clause. For a big table that has few million rows, the DELETE statement is slow and not efficient.To delete all rows from a big table fast, you use the following TRUNCATE TABLE statement:TRUNCATE TABLE table\_name;Code language: SQL (Structured Query Language) (sql)In this syntax, you specify the table\_name that you want to delete data after the TRUNCATE TABLE clause.Some database systems such as MySQL and  PostgreSQL allow you to skip the TABLE keyword so the TRUNCATE TABLE statement is as simple as follows:TRUNCATE table\_name;Code language: SQL (Structured Query Language) (sql)When you issue the TRUNCATE TABLE statement, the database system deletes all rows from the table by deallocating the data pages allocated by the table. By doing this, the RDBMS can reduce the resources for logging and the number of locks that need to acquire.To truncate multiple tables at a time, you can use a list of comma-separated table names after the TRUNCATE TABLE clause as follows:TRUNCATE TABLE table\_name1, table\_name2, ...;Code language: SQL (Structured Query Language) (sql)Not all database systems support this form of the TRUNCATE TABLE statement. If you are using the one that does not, you must issue multiple TRUNCATE TABLE statements to truncate multiple tables.SQL TRUNCATE TABLE vs. DELETELogically the TRUNCATE TABLE statement and the DELETE statement without the WHERE clause gives the same effect that removes all data from a table. However, they do have some differences:When you use the DELETE statement, the database system logs the operations. And with some efforts, you can roll back the data that was deleted. However, when you use the TRUNCATE TABLE statement, you have no chance to roll back except you use it in a transaction that has not been committed.To delete data from a table referenced by a foreign key constraint, you cannot use the TRUNCATE TABLE statement. In this case, you must use the DELETE statement instead.The TRUNCATE TABLE statement does not fire the delete trigger if the table has the triggers associated with it.Some database systems reset the value of an auto-increment column (or identity, sequence, etc.) to its starting value after you execute the TRUNCATE TABLE statement. It is not the case for the DELETE statement.The DELETE statement with a WHERE clause deletes partial data from a table while the TRUNCATE TABLE statement always removes all data from the table.SQL TRUNCATE TABLE examplesLet’s take a look at an example of truncating a table.First, create a new table named big\_table as follows:CREATE TABLE big\_table (  
 id INT AUTO\_INCREMENT PRIMARY KEY,  
 val INT  
);Code language: SQL (Structured Query Language) (sql)Second, execute the following statement as many times as you want to insert sample data into the big\_table table:INSERT INTO big\_table (val)  
VALUES  
 (RAND(100000));Code language: SQL (Structured Query Language) (sql)Note that if you use a database system that supports stored procedure, you can put this statement inside a loop. For example, the following stored procedure in MySQL loads data into the big\_table table with the number of rows specified by the num parameter.DELIMITER $$  
CREATE PROCEDURE load\_big\_table\_data(IN num int)  
BEGIN  
 DECLARE counter int default 0;  
   
 WHILE counter < num DO  
 INSERT INTO big\_table(val)  
 VALUES(RAND(1000000));  
 END WHILE;  
   
END$$Code language: SQL (Structured Query Language) (sql)The following statement calls the load\_big\_table\_data stored procedure to insert 10,000 rows into the big\_table table.CALL load\_big\_table\_data(10000);Code language: SQL (Structured Query Language) (sql)Third, to remove all data from the big\_table, you use the following statement:TRUNCATE TABLE big\_table;Code language: SQL (Structured Query Language) (sql)As you can see, how fast the TRUNCATE TABLE statement is.Now you should know how to use the TRUNCATE TABLE statement to delete all data from a big table fast and understand the differences between the TRUNCATE TABLE and DELETE statements.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL DROP TABLE Up NextSQL Primary Key

# Content from https://www.sqltutorial.org/sql-primary-key/

Home » SQL Tutorial » SQL Primary KeySQL Primary KeySummary: in this tutorial, you will learn about the primary key and how to use the SQL PRIMARY KEY constraint to add a primary key to the table.What is the primary key in SQL?A table consists of columns and rows. Typically, a table has a column or set of columns whose values uniquely identify each row in the table. This column or the set of columns is called the primary key.The primary key that consists of two or more columns is also known as the composite primary key.Each table has one and only one primary key. The primary key does not accept NULL or duplicate values.See the following courses table:Because the values in the course\_id column uniquely identify the rows in the courses table, the course\_id column is the primary key of the courses table.If the primary key consists of two or more columns, the values may be duplicated in one column, but the combination of values from all columns in the primary key must be unique.See the following training table.The primary key of the training table consists of the employee\_id and course\_id columns.The values in the course\_id column are duplicated. However, the combination of values in the employee\_id and course\_id columns is not.Creating table with primary keyGenerally, you define the primary key when creating the table. If the primary key consists of one column, you can use the PRIMARY KEY constraint as a column or table constraint. If the primary key consists of two or more columns, you must use the PRIMARY KEY constraint as the table constraint.Suppose you want to manage the projects and project assignments of the company in the database. Therefore, you need to create two tables: projects and project\_assignments.The following statement creates the projects table:CREATE TABLE projects (  
 project\_id INT PRIMARY KEY,  
 project\_name VARCHAR(255),  
 start\_date DATE NOT NULL,  
 end\_date DATE NOT NULL  
);Code language: SQL (Structured Query Language) (sql)You add the PRIMARY KEY in the column definition to make the project\_id column as the primary key of the projects table.The following statement is equivalent to the above statement but instead of using the PRIMARY KEY constraint as the column constraint, it uses the table constraint:CREATE TABLE projects (  
 project\_id INT,  
 project\_name VARCHAR(255),  
 start\_date DATE NOT NULL,  
 end\_date DATE NOT NULL,  
 CONSTRAINT pk\_id PRIMARY KEY (project\_id)  
);Code language: SQL (Structured Query Language) (sql)You use the CONSTRAINT clause at the end of the CREATE TABLE statement to promote the project\_id column to the primary key.To store the project assignments which represents who was assigned to which project, you need to create the project\_assignments table using the following statement:CREATE TABLE project\_assignments (  
 project\_id INT,  
 employee\_id INT,  
 join\_date DATE NOT NULL,  
 CONSTRAINT pk\_assgn PRIMARY KEY (project\_id , employee\_id)  
);Code language: SQL (Structured Query Language) (sql)Because the primary key consists of two columns: project\_id and employee\_id, you must use the PRIMARY KEY as the table constraint.Adding the primary key with ALTER TABLE statementFirst, you can define a table without a primary key using the CREATE TABLE statement though it is not a good practice. Then, you add the primary key to the table using the ALTER TABLE statement.For example, the following statement creates the project\_milestones table without the primary key. The project\_milesones stores milestones of projects.CREATE TABLE project\_milestones(  
 milestone\_id INT,  
 project\_id INT,  
 milestone\_name VARCHAR(100)  
);Code language: SQL (Structured Query Language) (sql)Now, you can use the following ALTER TABLE statement to promote the milestone\_id column as the primary key.ALTER TABLE project\_milestones  
ADD CONSTRAINT pk\_milestone\_id PRIMARY KEY (milestone\_id);Code language: SQL (Structured Query Language) (sql)You can skip the CONSTRAINT clause as follows:ALTER TABLE project\_milestones  
ADD PRIMARY KEY (milestone\_id);Code language: SQL (Structured Query Language) (sql)Removing the primary key constraintYou will rarely remove the primary key of a table. However, in case you must do it, you can use the ALTER TABLE statement as follows:ALTER TABLE table\_name  
DROP CONSTRAINT primary\_key\_constraint;Code language: SQL (Structured Query Language) (sql)If you are using MySQL, the syntax for removing the primary key is simpler as follows:ALTER TABLE table\_name  
DROP PRIMARY KEY;Code language: SQL (Structured Query Language) (sql)For example, to remove the primary key constraint of the project\_milestones table, you use the following statement.ALTER TABLE project\_milestones   
DROP CONSTRAINT pk\_milestone\_id;Code language: SQL (Structured Query Language) (sql)In this tutorial, you have learned about the primary key concept and how to manage a table’s primary key, including adding and removing the primary key.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL TRUNCATE TABLE Up NextSQL Foreign Key Constraint

# Content from https://www.sqltutorial.org/sql-foreign-key/

Home » SQL Tutorial » SQL Foreign Key ConstraintSQL Foreign Key ConstraintSummary: in this tutorial, you  will learn about the SQL foreign key and how to create a FOREIGN KEY constraint to enforce the relationship between tables.Introduction to SQL foreign key constraintA foreign key is a column or a group of columns that enforces a link between the data in two tables. In a foreign key reference, the primary key column (or columns) of the first table is referenced by the column (or columns) of the second table. The column (or columns) of the second table becomes the foreign key.You use the FOREIGN KEY constraint to create a foreign key when you create or alter table. Let’s take a simple example to get a better understanding.SQL FOREIGN KEY constraint examplesSee the following projects and project\_assignments tables:CREATE TABLE projects (  
 project\_id INT AUTO\_INCREMENT PRIMARY KEY,  
 project\_name VARCHAR(255),  
 start\_date DATE NOT NULL,  
 end\_date DATE NOT NULL  
);  
  
CREATE TABLE project\_milestones(  
 milestone\_id INT AUTO\_INCREMENT PRIMARY KEY,  
 project\_id INT,  
 milestone\_name VARCHAR(100)  
);Code language: SQL (Structured Query Language) (sql)Each project may have zero or more milestones while one milestone must belong to one and only one project. The application that uses these tables must ensure that for each row in the project\_milestones table there exists the corresponding row in the projects table. In other words, a milestone cannot exist without a project.Unfortunately, users may edit the database using client tool or if there is a bug in the application, a row might be added to the project\_milestones table that does not correspond to any row in the projects table. Or user may delete a row in the projects table, leaving orphaned rows in the project\_milestones table. This causes the application not to work properly.The solution is to add an SQL FOREIGN KEY constraint to the project\_milestones table to enforce the relationship between the projects and project\_milestones tables.You can create the FOREIGN KEY constraint when you create the table as follows:CREATE TABLE project\_milestones (  
 milestone\_id INT AUTO\_INCREMENT PRIMARY KEY,  
 project\_id INT,  
 milestone\_name VARCHAR(100),  
 FOREIGN KEY (project\_id)  
 REFERENCES projects (project\_id)  
);Code language: SQL (Structured Query Language) (sql)The FOREIGN KEY clause promotes the project\_id of the project\_milestones table to become the foreign key that is referenced to the project\_id of the projects table.FOREIGN KEY (project\_id)  
 REFERENCES projects (project\_id)Code language: SQL (Structured Query Language) (sql)You can assign a name to a FOREIGN KEY constraint as follows:CREATE TABLE project\_milestones (  
 milestone\_id INT AUTO\_INCREMENT PRIMARY KEY,  
 project\_id INT,  
 milestone\_name VARCHAR(100),  
 CONSTRAINT fk\_project FOREIGN KEY (project\_id)  
 REFERENCES projects (project\_id)  
);Code language: SQL (Structured Query Language) (sql)fk\_project is the name of the FOREIGN KEY constraint.Adding FOREIGN KEY contraints to existing tablesTo add a FOREIGN KEY constraint to existing table, you use the ALTER TABLE statement.ALTER TABLE table\_1  
ADD CONSTRAINT fk\_name FOREIGN KEY (fk\_key\_column)  
 REFERENCES table\_2(pk\_key\_column)Code language: SQL (Structured Query Language) (sql)Suppose the project\_milestones already exists without any predefined foreign key and you want to define a FOREIGN KEY constraint for the project\_id column. To do so, you use the following ALTER TABLE statement:ALTER TABLE project\_milestones  
ADD CONSTRAINT fk\_project FOREIGN KEY(project\_id)  
 REFERENCES projects(project\_id);Code language: SQL (Structured Query Language) (sql)Removing foreign key constraintsTo remove a foreign key constraint, you also use the ALTER TABLE statement as follows:ALTER TABLE table\_name  
DROP CONSTRAINT fk\_name;Code language: SQL (Structured Query Language) (sql)If you are using MySQL, you can use a cleaner syntax as follows:ALTER TABLE table\_name  
DROP FOREIGN KEY fk\_name;Code language: SQL (Structured Query Language) (sql)For example, to remove the fk\_project foreign key constraint, you use the following statement:ALTER TABLE project\_milestones  
DROP CONSTRAINT fk\_project;Code language: SQL (Structured Query Language) (sql)In this tutorial, we have introduced you to the foreign key concept and shown you how to create foreign key using SQL FOREIGN KEY constraint.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL Primary Key Up NextSQL UNIQUE Constraint

# Content from https://www.sqltutorial.org/sql-unique-constraint/

Home » SQL Tutorial » SQL UNIQUE ConstraintSQL UNIQUE ConstraintSummary: in this tutorial, you will learn how to enforce the uniqueness of values in a column or a set of columns using SQL UNIQUE constraint.What is SQL UNIQUE constraintSometimes, you want to make sure that the values in a column or a set of columns are not duplicate. For example, duplicate emails in the employees table are not acceptable.Since the email column is not the part of the primary key, the only way to prevent duplicate values in the email column is to use a UNIQUE constraint.By definition, an SQL UNIQUE constraint defines a rule that prevents duplicate values stored in specific columns that do not participate a primary key.UNIQUE vs. PRIMARY KEY constraintsYou can have at most one PRIMARY KEY constraint whereas you can have multiple UNIQUE constraints in a table. In case you have multiple UNIQUE constraints in a table, all UNIQUE constraints must have a different set of columns.Different from the PRIMARY KEY constraint, the UNIQUE constraint allows NULL values. It depends on the RDBMS to consider NULL values are unique or not.For example, MySQL treats the NULL values as distinct values, therefore, you can store multiple NULL values in the columns participated in the UNIQUE constraint. However, it is not the case for Microsoft SQL Server or Oracle Database.The following table illustrates the differences between UNIQUE constraint and PRIMARY KEY constraint:Not AllowedPRIMARY KEY constraintUNIQUE constraintThe number of constraintsOneManyNULL valuesDo not allowAllowCreating UNIQUE constraintsTypically, you create UNIQUE constraint when creating the table. The following CREATE TABLE statement defines the users table with the username column is unique.CREATE TABLE users (  
 user\_id INT AUTO\_INCREMENT PRIMARY KEY,  
 username VARCHAR(255) NOT NULL UNIQUE,  
 password VARCHAR(255) NOT NULL  
);Code language: SQL (Structured Query Language) (sql)To create a UNIQUE constraint for a column, you need to add the UNIQUE keyword in the column definition. In this case, we created the UNIQUE constraint as the column constraint.If you insert or update the value that is the same as the one which already exists in the username column, the RDBMS will reject the change and return an error.The following statement is equivalent to the above statement with the UNIQUE constraint created using the table constraint syntax.CREATE TABLE users (  
 user\_id INT AUTO\_INCREMENT PRIMARY KEY,  
 username VARCHAR(255) NOT NULL,  
 password VARCHAR(255) NOT NULL,  
 CONSTRAINT uc\_username UNIQUE (username)  
);Code language: SQL (Structured Query Language) (sql)In this case, you put the CONSTRAINT clause at the end of the CREATE TABLE statement.Adding UNIQUE constraints to existing tableIn case the table already exists, you can add a UNIQUE constraint for columns with the prerequisite that the column or the combination of columns which participates in the UNIQUE constraint must contain unique values.Suppose the users table was created without the UNIQUE constraint defined for the username column. To add the UNIQUE constraint to the username column, you use the ALTER TABLE statement as follows:ALTER TABLE users  
ADD CONSTRAINT uc\_username UNIQUE(username);Code language: SQL (Structured Query Language) (sql)If you want to add a new column and create a UNIQUE constraint for it, you use the following form of the ALTER TABLE statement.ALTER TABLE users  
ADD new\_column data\_type UNIQUE;Code language: SQL (Structured Query Language) (sql)For example, the following statement adds the email column with the UNIQUE constraint to the user table.ALTER TABLE users  
ADD email VARCHAR(255) UNIQUE;Code language: SQL (Structured Query Language) (sql)Removing UNIQUE constraintTo remove a UNIQUE constraint, you use the ALTER TABLE statement as follows:ALTER TABLE table\_name  
DROP CONSTRAINT unique\_constraint\_name;Code language: SQL (Structured Query Language) (sql)For example, to remove the uc\_username unique constraint in the users table, you use the following statement.ALTER TABLE users  
DROP CONSTRAINT uc\_username;  
Code language: SQL (Structured Query Language) (sql)In this tutorial, you have learned about the UNIQUE constraint and how to apply it to enforce the uniqueness of values in a column or a set of columns.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL Foreign Key Constraint Up NextSQL CHECK Constraint

# Content from https://www.sqltutorial.org/sql-not-null-constraint/

Home » SQL Tutorial » SQL NOT NULL ConstraintSQL NOT NULL ConstraintSummary: this tutorial shows you how to use the SQL NOT NULL constraint to enforce a column from storing NULL values.Introduction to SQL NOT NULL constraintThe NOT NULL constraint is a column constraint that defines the rule which constrains a column to have non-NULL values only.It means that when we use the INSERT statement to insert a new row into the table, we have to specify  the values for the NOT NULL columns.The following statement illustrates the NOT NULL constraint syntax. It enforces the column\_name to not accept any NULL values.CREATE TABLE table\_name(  
 ...  
 column\_name data\_type NOT NULL,  
 ...  
);Code language: SQL (Structured Query Language) (sql)Logically, an NOT NULL constraint is equivalent to a CHECK constraint, therefore, the above statement is equivalent to the following statement.CREATE TABLE table\_name (   
 ...  
 column\_name data\_type,  
 ...  
 CHECK (column\_name IS NOT NULL)  
);Code language: SQL (Structured Query Language) (sql)For example, the following statement creates the training table that has the taken\_date column with the NOT NULL constraint.CREATE TABLE training (  
 employee\_id INT,  
 course\_id INT,  
 taken\_date DATE NOT NULL,  
 PRIMARY KEY (employee\_id , course\_id)  
);Code language: SQL (Structured Query Language) (sql)Most relational database management systems add the NOT NULL constraint automatically by default to the primary key columns, therefore, we don’t have to specify it explicitly.The following INSERT statement violates the NOT NULL constraint.INSERT INTO training(employee\_id,course\_id)  
VALUES(1,1);Code language: SQL (Structured Query Language) (sql)ALTER TABLE NOT NULL statementTypically, we define the NOT NULL constraints for columns when we create the table. However, sometimes, we want to change the constraint of a column that accepts a NULL value to not accept a NULL value.To carry the change, we use these two steps:First, update all current NULL values to non-NULL values using the UPDATE statement.UPDATE table\_name  
SET column\_name = 0  
WHERE  
 column\_name IS NULL;Code language: SQL (Structured Query Language) (sql)Note that we use the IS NULL operator in the WHERE clause to find the rows whose the  column\_name is NULL.Second, add the NOT NULL constraint to the column using the ALTER TABLE statementALTER TABLE table\_name  
MODIFY column\_name data\_type NOT NULL;Code language: SQL (Structured Query Language) (sql)Suppose the taken\_date column of the training table is NULL and we want to change it to NOT NULL.First, we update all NULL values in the taken\_date column to a specific date e.g., the current date.UPDATE training  
SET taken\_date = CURRENT\_DATE ()  
WHERE  
 taken\_date IS NULL;Code language: SQL (Structured Query Language) (sql)Second, we change the take\_date column to NOT NULL constraint.ALTER TABLE training   
MODIFY taken\_date date NOT NULL;Code language: SQL (Structured Query Language) (sql)In this tutorial, we have shown you how to use the NOT NULL constraint to constrain a column to accept only non-NULL values.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL CHECK Constraint Up NextSQL Views

# Content from https://www.sqltutorial.org/sql-check-constraint/

Home » SQL Tutorial » SQL CHECK ConstraintSQL CHECK ConstraintSummary: in this tutorial, you will learn how to use the SQL CHECK constraint to validate data in a column or a set of columns based on a Boolean expression.Introduction to SQL CHECK constraintA CHECK constraint is an integrity constraint in SQL that allows you to specify that a value in a column or set of columns must satisfy a Boolean expression.You can define a CHECK constraint on a single column or the whole table. If you define the CHECK constraint on a single column, the CHECK constraint checks value for this column only. However, if you define a CHECK constraint on a table, it limits value in a column based on values in other columns of the same row.The CHECK constraint consists of the keyword CHECK followed by a Boolean expression in parentheses:CHECK(Boolean\_expression)Code language: SQL (Structured Query Language) (sql)To assign a CHECK constraint a name, you use the following syntax:CONSTRAINT constraint\_name CHECK(Boolean\_expression)Code language: SQL (Structured Query Language) (sql)It is important to note that the CHECK constraint is satisfied when the Boolean expression returns true or the NULL value. Most Boolean expressions evaluate to NULL if one of the operands is NULL, they will not prevent null values in the constrained column. To make sure that the column does not contain the NULL values, you use the NOT NULL constraint.SQL CHECK constraint examplesLet’s take some examples of creating CHECK constraints.To create the products table with the value in the selling\_price column positive, you use the following CREATE TABLE statement:CREATE TABLE products (  
 product\_id INT PRIMARY KEY,  
 product\_name VARCHAR(255) NOT NULL,  
 selling\_price NUMERIC(10,2) CHECK (selling\_price > 0)  
);Code language: SQL (Structured Query Language) (sql)The CHECK constraint comes after the data type of the column. If you insert or update the selling price with a negative value, the expression selling\_price >= 0 returns false and the RDMBS returns an error.You can assign the CHECK constraint a separate name. The constraint name helps you clarify the error message returned by the RDBMS and know exactly which constraint the value has been violating.To assign a name to a constraint, you use the keyword CONSTRAINT followed by the constraint’s name.For example, the following statement assigns positive\_selling\_price as the name of CHECK constraint on the selling\_price column.The syntax for assigning a CHECK constraint is as follows:CREATE TABLE products (  
 product\_id INT PRIMARY KEY,  
 product\_name VARCHAR(255) NOT NULL,  
 selling\_price NUMERIC(10,2) CONSTRAINT positive\_selling\_price CHECK (selling\_price > 0)  
);Code language: SQL (Structured Query Language) (sql)You can define a CHECK constraint that refers to multiple columns. Suppose you store both selling prices and costs in the products table, and you want to ensure that the cost is always lower than the selling price.CREATE TABLE products (  
 product\_id INT PRIMARY KEY,  
 product\_name VARCHAR (255) NOT NULL,  
 selling\_price NUMERIC (10, 2) CHECK (selling\_price > 0),  
 cost NUMERIC (10, 2) CHECK (cost > 0),  
 CHECK (selling\_price > cost)  
);Code language: SQL (Structured Query Language) (sql)First, you have the two CHECK constraints associated with the selling\_price and cost columns to make sure that values in each column positive.Second, you have another CHECK constraint that is not attached to any column, instead, it appears as the last clause in the CREATE TABLE statement.The first two constraints are column constraints, whereas the third one is a table constraint. The table constraint does not associate with any column.You can assign the table constraint a name using the following syntax.CREATE TABLE table\_name (  
 …,  
 CONSTRAINT check\_constraint\_name CHECK (Boolean\_expression)  
);Code language: SQL (Structured Query Language) (sql)For example, the following statement assigns a name to the CHECK constraint above.CREATE TABLE products (  
 product\_id INT PRIMARY KEY,  
 product\_name VARCHAR (255) NOT NULL,  
 selling\_price NUMERIC (10, 2) CHECK (selling\_price > 0),  
 cost NUMERIC (10, 2) CHECK (cost > 0),  
 CONSTRAINT valid\_selling\_price CHECK (selling\_price > cost)  
);Code language: SQL (Structured Query Language) (sql)In this tutorial, you have learned how to use the CHECK constraint to validate data based on a Boolean expression.Was this tutorial helpful ? Yes No Send Cancel PreviouslySQL UNIQUE Constraint Up NextSQL NOT NULL Constraint