Best javascript cheat sheet pdf

I'm not robot

reCAPTCHA

Continue

Getting to know G Suite Google Keep lets you create notes and make target lists that sync across your computer and phone or tablet. It is remarkably useful in a number of ways. You can tag and search all your notes so they're easy to find later. You can share your notes with other users and collaborate on them. You can record voice memos, and Keep transcribes them as text notes. You can include images in your notes, and if the text appears in an image, the image appears in the search results. You can create time or location trigger reminders based on your notes. Keep is included in the G Suite and is also available for free to individual users. You can use it as a web app in a web browser on your computer (Chrome and Firefox are supported, as is IE/Edge on Windows and Safari on macOS); It's also available as an app for your Android or iOS mobile device. To use Keep, your device must be signed in to your Google Account, such as a G Suite account or Gmail account. This guide guides you through how to quickly start using Keep. The screenshots and descriptions of the user interface refer specifically to the web version, but the same features are in the Keep mobile app – only posted differently. IT Pros: We encourage you to share this story with your G Suite users to help them learn the Keep ropes. In this guide, you'll find a useful cheat sheet that documents some of the more commonly used elements in SQL, and even some of the less common ones. Hopefully it will help developers - both beginner and experienced level - become more skilled in their understanding of the SQL language. Use this as a quick reference in development, a learning aid, or even print it and bind it if you prefer (whatever works!). But before we get to the cheat sheet ourselves, for developers who may not be familiar with SQL, let's start with ... PDF version of SQL Cheat Sheet SQL Cheat Sheet (Download PDF) Infographic Version of SQL Cheat Sheet (PNG) SQL Cheat Sheet (Download PNG) What is SQL SQL stands for Structured Query Language. It's the language you choose on today's web for storing, manipulating, and retrieving data in relational databases. Most, if not all of the sites you visit will use it in any way, including this one. This is what a basic relational database looks like. This example specifically stores e-commerce information, especially the products on sale, the users who purchase them, and registrations of these orders that connect these 2 devices. Using SQL, you can interact with the database by typing queries, which, when performed, returns any results that meet the criteria. Here is an example of query:- SELECT * FROM users; Using this SELECT statement, the query selects all data from all columns in the user's table. It would then data as below, which is usually called a result set:- If we were to replace the star asterisk characters (*) with specific column names instead, only the data from these columns is returned from the query. SELECT first_name, last_name FROM USERS; We can add some complexity to a standard SELECT statement by adding a WHERE statement, which allows you to filter what is returned. SELECT * FROM PRODUCTS WHERE stock_count &lt; = 10 ORDER AT stock_count ASC; This query returns all data from the product table with a stock_count of less than 10 in the result set. The use of the ORDER BY keyword means that the results will be ordered using stock_count, lowest values to the highest. Using the INSERT INTO statement, we can add new data to a table. Here is a basic example of adding a new user to the user table:- INSERT Into users (first_name, last_name, address, email) VALUES ('Tests', 'Jester', '123 Fake Street, Sheffield, United Kingdom', '[email protected]'); So if you were to run the query again to return all data from the user's table, the result set will look like this: Of course, these examples show only a very small sample of what the SQL language is capable of. SQL vs MySQL You may have heard of MySQL before. It is important that you do not confuse this with SQL yourself, as there is a clear difference. SQL is the language. It outlines syntax that allows you to write queries that manage relational databases. Nothing more. Meanwhile, MySQL is a database system that is running on a server. It implements the SQL language so that you can write queries by using the syntax to manage MySQL databases. In addition to MySQL, there are other systems that implement SQL. Some of the more popular include: PostgreSQL SQLite Oracle Database Microsoft SQL Server Installing MySQL Windows The recommended way to install MySQL on Windows is by using the installer you can download from the MySQL website. MacOS On macOS uses the recommended way to install MySQL native packages, which sounds much more complicated than it actually is. Essentially, it also involves simply downloading an installer. Alternatively, if you prefer to use package managers like Homebrew, you can install MySQL as follows: brew install mysql While if you need to install the older MySQL version 5.7, which is still widely used today online, you can: brew install [email protected] Using MySQL With MySQL now installed on your system, to get up and go as quickly as possible write SQL queries , it is recommended that you use an SQL management application to make managing your databases a much easier and easier process. There are many apps to choose from that largely do the same job, so it's down to your own personal preference that you'll use: MySQL Workbench is developed by Oracle, the owner of MySQL. HeidiSQL (Recommended Windows) is a free open source app for Windows. For macOS and wine is required first as a prerequisite. Prerequisite. is a very popular option that operates in the browser. Sequel Pro (Recommended macOS) is a macOS only option and our favorite thanks to the clear and easy-to-use interface. When you're ready to start writing your own SQL queries, instead of spending time creating your own database, consider importing dummy data instead. The MySQL website contains a number of dummy databases that you can download for free and then import into the SQL app. Our favorite of these is the world database, which provides some interesting data to practice writing SQL queries for. Here's a screenshot of the country table in Sequel Pro. This sample query returns all countries with Queen Elizabeth II as head of state. While this returns all European countries with a population of over 50million along with its capital and population. And this latest returns the average percentage of French speakers in countries where the total number of French speakers is higher than 10%. Cheat sheet keywords A collection of keywords used in SQL statements, a description, and where an example is applicable. Some of the more advanced keywords have their own dedicated section later in the cheat sheet. Where MySQL is mentioned next to an example, this means that this example applies only to MySQL databases (unlike other database systems). Description of SQL KEYWORD KEYWORDS ADDS Adds a new column to an existing table. Example: Adds a new column named email_address to a table named users. CHANGE TABLE users add email_address varchar(255); ADD CONSTRAINT It creates a new constraint in an existing table, which is used to specify rules for data in the table. For example, adds a new PRIMARY KEY constraint called User to column ID and last NAME. CHANGE TABLE USERS ADD RESTRICTION USER PRIMARY KEY (ID, LAST NAME); CHANGE TABLE Adds, deletes, or edits columns in a table. It can also be used to add and delete constraints in a table, according to the above. Example: Adds a new Boolean column called approved to a table named appointments. CHANGE TABLE APPOINTMENTS ADD APPROVED Boolean; Example 2: Deletes the approved column from the Quote table CHANGE TABELL agreements DROP COLUMN approved; CHANGE COLUMN Changes the data type of the column in a table. For example, in the Users table, you make the column incept_date a datetime type. CHANGE TABLE USERS CHANGE COLUMN incept_date datetime; ALL Returns true if all subquery values meet the passed condition. Example: Returns users with a higher number of tasks than the user with the highest number of tasks in the HR department (id 2) SELECT first_name, last name, tasks_no FROM users where tasks_no &gt; ALL (SELECT tasks FROM user where department_id = 2); OG is used to join conditions in a WHERE clause. Example: Returns events in London, United Kingdom SELECT * * events where host_country='United Kingdom' AND host_city='London'; All returns true if any of the subquery values meet the specified condition. Example: Returns products from the product table that have received orders – stored in the order table – with a quantity of more than 5. SELECT NAME FROM PRODUCTS WHERE productId = ANY (SELECT productId FROM orders WHERE quantity &gt; 5); AS Renames a table or column with an alias value that exists only during the query. Example: Aliases north_east_user_subscriptions select north_east_user_subscriptions AS ne_subs FROM users WHERE ne_subs &gt; 5; ASC Used with ORDER BY to return the data in ascending order. Example: Apples, Bananas, Peaches, Radish BETWEEN Selects values within the specified range. Example 1: Selects inventory with a quantity between 100 and 150. SELECT * FROM WAREHOUSE WHERE QUANTITIES BETWEEN 100 and 150; Example 2: Selects inventory with a quantity NOT between 100 and 150. Alternatively, using the NON keyword here reverses the logic and selects values outside the specified range. SELECT * FROM STOCK WHERE QUANTITY NOT BETWEEN 100 AND 150; CASE Change query output depending on the conditions. Example: Returns users and their subscriptions, along with a new column called activity_levels that makes a rating based on the number of subscriptions. SELECT first_name, last name, subscriptions CASE WHEN SUBSCRIPTIONS &gt; 10 THEN Very active WHEN Number between 3 and 10 THEN Active other Inactive END AS activity_levels from users; CHECK Adds a constraint that limits the value that can be added to a column. Example 1 (MySQL): Verify that all users added to the user table are 18 years of age or over. CREATE TABELL users (first_name varchar(255), age use, CHECK (age &gt; = 18)); Example 2 (MySQL): Adds a check after the table is already created. CHANGE TABLE USERS ADD CHECK (age&gt; = 18); CREATE DATABASE Creates a new database. Example: Creates a new database named 'websitesetup'. CREATE DATABASE websiteetup; CREATE TABLE Creates a new table . Example: Creates a new table called users in the websitesetup database . CREATE TABELL users (id int, first_name varchar (255), last name varchar (255), address varchar (255), contact_number int); DEFAULT Specifies a default value for a column. Example 1 (MySQL): Creates a new table called Products that has a name column with a default value for Placeholder Name and a available_from column with a default value for today's date. CREATE TABELL products (ID-int, name varchar(255) DEFAULT 'Placeholder name', available_from date STANDARD GETDATE()); Example 2 (MySQL): The same as above, but editing an existing table. CHANGE TABLE PRODUCTS RENAME SET DEFAULT 'PlaceHOLDER Name', CHANGE available_from SET DEFAULT GETDATE(); DELETE Delete data from a table. Example: Removes with a user_id of 674. Delete Delete users where user_id = 674; DESC Is used with ORDER BY to return the data in descending order. Example: Radish, Peach, Bananas, Apple's DROP COLUMN Deletes a column from a table. Example: Removes the first_name column from the user table. CHANGE TABLE USERS DROP COLUMN first_name DROP DATABASE Deletes the entire database. Example: Deletes a database named 'websitesetup'. DROP DATABASE site setup; DROP DEFAULT Removes a default value for a column. Example (MySQL): Removes the default value from the name column in the products table. CHANGE TABLE PRODUCTS CHANGE COLUMN NAME DROP DEFAULT; DROP TABLE Deletes a table from a database. Example: Removes the user table. DROP TABLE users; EXISTS Looks for the existence of an entry in the subquery and returns true if one or more records are returned. Example: Displays all retailers with a deal funding percentage below 10. SELECT dealership_name FROM RETAILERS WHERE AVAILABLE (SELECT deal_name FROM QUOTES where dealership_id = deals.dealership_id AND finance_percentage &lt; 10); FROM Specifies which table you want to select or delete data from. Example: Selects data from the user table. SELECT area_manager FROM area_managers WHERE IT EXISTS (SELECT Product Name from products where area_manager_id = deals.area_manager_id &amp; price &lt; 20); IN Is used with a WHERE clause as a shorthand for multiple OR conditions. So instead of:- SELECT * FROM USERS WHERE COUNTRIES = 'USA' OR COUNTRY = 'UNITED KINGDOM' OR COUNTRY = 'Russia' OR COUNTRY = 'Australia'; You can use:- SELECT * FROM USERS WHERE COUNTRIES IN ('USA', 'United Kingdom', 'Russia', 'Australia'); INSERT IN Add new rows to a table. Example: Adds a new vehicle. INSERT INTO CARS (make, model, mileage, year) VALUES ('Audi', 'A3', 30000, 2016); IS NULL tests for empty (NULL) values. Example: Returns users who have not provided a contact number. SELECT * FROM USERS WHERE contact_number IS ZERO; IS NOT ZERO The opposite of ZERO. Tests for values that are not empty/NULL. LIKE Returns true if the operand value matches a pattern. Example: Returns true if the user's first_name ends with 'son'. CHOOSE * FROM USERS WHERE first_name LIKE '%son'; DOES NOT RETURN TRUE if a record does not meet the condition. Example: Returns true if the user's first_name ends with 'son'. SELECT * FROM USERS WHERE first_name DO NOT LIKE %son; OR Used with WHERE to include data when one of the conditions is met. Example: Returns users who live in either Sheffield or Manchester. SELECT * FROM USERS WHERE THE CITY = 'Sheffield' OR 'Manchester'; ORDER BY Allows you to sort the performance data in ascending (default) or descending order through the use of ASC or DESC keywords. Example: Returns countries in alphabetical order. SELECT * FROM COUNTRY ORDER BY NAME; ROWNUM Returns results where the row number the passed condition. Example: Returns the top 10 countries from Table. SELECT * FROM COUNTRY WHERE ROWNUM &lt; = 10; SELECT Allows you to select data from a database, which is then returned in a result set. Example 1: Selects all columns from all users. SELECT * FROM USERS; Example 2: Selects the columns first_name and last names from all users.xx SELECT first_name, last name FROM users; SELECT DISTINCT Sames as SELECT, except duplicate values are omitted. Example: Creates a backup table by using data from the user table. SELECT * INN usersBackup2020 FROM users; SELECT In Copy data from one table and insert it into another. Example: Returns all countries from the user table and removes any duplicate values (which will be very likely) SELECT DISTINCT countries from users; SELECT TOP Allows you to return a specified number of records to be returned from a table. Example: Returns the top three cars from the car table. CHOOSE THE TOP 3* FROM CARS; SET Used with UPDATE to update existing data in a table. Example: Updates the value and quantity values of an order with an ID of 642 in the order table. UPDATE orders SET value = 19.49, quantity = 2 WHERE id = 642; SOMEONE identical to ANYONE. TOP Is used with SELECT to return a specified number of records from a table. Example: Returns the top five users from the user table. CHOOSE TOP 5 * FROM USERS; TRUNCATE TABLE In the same way as DROP, but instead of deleting the table and data, this only deletes the data. Example: Clears the session table, but leaves the table itself intact. TRUNCING TABLE SESSIONS; UNION Combines the results of 2 or more SELECT statements and returns only distinct values. Example: Returns the cities from the events and subscribers tables. CHOOSE city from events UNION SELECT city from subscribers; UNION ALL The same as UNION, but includes duplicate values. UNIQUE This limitation ensures that all values in a column are unique. Example 1 (MySQL): Adds a unique constraint to the ID column when you create a new user table. CREATE TABELL users (id int NOT NULL, name varchar (255) NOT NULL, UNIQUE (id)); Example 2 (MySQL): Changes an existing column to add a UNIQUE constraint. CHANGE TABLE USERS ADD UNIQUE (id); UPDATE Updates existing data in a table. Example: Updating mileage and serviceDue values for a vehicle with an id of 45 in the car table. UPDATE cars SET mileage = 23500, serviceDue = 0 WHERE id = 45; VALUES used with the INSERT INTO keyword to add new values to a table. Example: Adds a new car to the car table. INSERT INTO CARS (name, model, year) VALUES ('Ford', 'Fiesta', 2010); WHERE Filters results to include only data that meets the specified condition. Example: Returns orders with a quantity of more than 1 item. SELECT * FROM ORDERS WHERE QUANTITY &gt; 1; Comments Comments allow you to explain parts of or to comment on out code and prevent driving. In SQL, there are 2 types of comments, one line, and multiple lines. Simple Comments Single line comments start with -. All text after these 2 characters to the end of the line will be ignored. -- My Select Query SELECT * FROM Users; Multiline comments Multiline comments start with /* and end with */. They span several lines until the closing characters are found. /* This is my selection query. It grabs all rows of data from the user table */ SELECT * FROM users; /* This is another select query, which I will not perform yet SELECT * FROM tasks; */ MySQL data types When you create a new table or edit an existing table, you must specify the type of data each column accepts. In the following example, data sent to the ID column must be an int, while the first_name column has a VARCHAR data type of a maximum of 255 characters. CREATE TABLE users (id-int, first_name varchar(255)); String data types Data types Data type Description CHAR(size) Fixed length string that can contain letters, numbers, and special characters. The size parameter specifies the maximum string length, from 0 to 255 with a standard of 1. VARCHAR(size) Variable longitudinal string similar to CHAR(), but with a maximum string length range from 0 to 65535. BINARY (size) Similar to CHAR (), but stores binary byte strings. VARBINARY (size) Similar to varchar(), but for binary byte strings. TINYBLOB holds binary large objects (BLOBs) with a maximum length of 255 bytes. TINYTEXT Contains a string with a maximum length of 255 characters. Use VARCHAR() instead, as it is extracted much faster. TEXT(size) Contains a string with a maximum length of 65535 bytes. Again, better to use VARCHAR(). BLOB(size) Contains binary large objects (BLOBs) with a maximum length of 65535 bytes. MEDIUMTEXT Contains a string with a maximum length of 16,777,215 characters. MEDIUMBLOB contains binary large objects (BLOBs) with a maximum length of 16,777,215 bytes. LONGTEXT Contains a string with a maximum length of 4,294,967,295 characters. LONGBLOB contains binary large objects (BLOBs) with a maximum length of 4,294,967,295 bytes. ENUM (a, b, c, etc...) A string object that has only one value, selected from a list of values that you define, up to a maximum of 65535 values. If a value is added that is not on this list, it is replaced with an empty value instead. Think of the enum similar to HTML radio boxes in this regard. MAKE TABLE T-shirts (color ENUM('red', 'green', 'blue', 'yellow', 'purple')); SET (a, b, c, etc...) A string object that can have 0 or more values, selected from a list of values that you define, up to a maximum of 64 values. Think of set to be equal to HTML check boxes in this regard. Numeric data types String data types Data type Description BIT(size) A bit value type with a default of 1. The allowed number of bits in a value is entered through the size parameter, which can contain values from 1 to 64. A very small integer with a signed selection -128 to 127, and an unsigned range of 0 to 255. Here, the size parameter specifies the maximum allowed display width, which is 255. BOOL Essentially a quick way to set the column to TINYINT with a size of 1. 0 considered false, while 1 is considered true. BOOLSK Same as BOOL. SMALLINT(size) A small integer with a signed range of -32768 to 32767, and an unsigned range from 0 to 65535. Here, the size parameter specifies the maximum allowed display width, which is 255. MEDIUMINT(size) A medium integer with a signed range of -8388608 to 8388607, and an unsigned range from 0 to 16777215. Here, the size parameter specifies the maximum allowed display width, which is 255. INT(size) A medium integer with a signed range of -2147483648 to 2147483647, and an unsigned range from 0 to 4294967295. Here, the size parameter specifies the maximum allowed display width, which is 255. INTE(size) Same as INT. BIGINT(size) A medium integer with a signed range of -9223372036854775808 to 9223372036854775807, and an unsigned range from 0 to 18446744073709551615. Here, the size parameter specifies the maximum allowed display width, which is 255. FLOAT(p) A floating point value. If the precision parameter (p) is between 0 and 24, the data type is set to FLOAT(), while the data type is set to DOUBLE (if the data type is from 25 to 53). This behavior is to make the storage of values more efficient. DOUBLE(size, d) A floating point value where the total digits are entered by the size parameter, and the number of digits after the decimal point is specified by the parameter d. DECIMAL(size, d) An exact fixed point number where the total number of digits is entered by the size parameters, and the total number of digits after the decimal point is specified by the d parameter. For size, the maximum number is 65 and the default value is 10, while for d the maximum number is 30 and the default value is 10. DES(size, d) Same as decimal. Date/Time Data Types Date/Time Data Types Data Type Description DATE A single date in ÅYYY-MM-DD format, with a supported range from '1000-01-01' to '9999-12-31'. DATETIME(fsp) A date in YYYY-MM-DD hh:mm:ss format, with a supported range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. By adding STANDARD and ON REFRESH to the column definition, it is automatically set to the current date/time. TIMESTAMP(fsp) A Unix timestamp, which is a value relative to the number of seconds since the Unix era ('1970-01-01 00:00:00' UTC). This has a supported area from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. By adding CURRENT_TIMESTAMP default and ON UPDATE CURRENT TIMESTAMP in the column definition, it is automatically set to the current date/time. TIME(fsp) A time in hh:mm:ss format, with a supported range from '-838:59:59' to '838:59:59'. YEAR This year, with a supported selection of '1901' to '2155'. Operators Operators Arithmetic Operators Operator Description + Add - Pull * Multiply / Divide % Modulo Bitwise Operator Bitwise Operators Operator Description &amp;amp; Bitwise OG | Bitwise OR ^ Bitwise exclusive OR comparison operators Comparison operators Operator Description = Equal to &gt; Greater than &lt; Less than &gt; = Greater than or equal to &lt; = Less than or equal to &lt;&gt; Not equal to compound operators compound operators operator description += Add equals -= Subtract equals *= Multiplier equals /= Dividing equals %= Modulo equals &amp;= Bitwise AND equals ^-= Bitwise exclusive equals |*= Bitwise OR equals Functions String Functions String Functions Description ASCII correspondingly returns ASCII value for a specific character. CHAR_LENGTH Returns the character length of a string. CHARACTER_LENGTH same as CHAR_LENGTH. CONCAT adds expressions together, with at least 2. CONCAT_WS adds expressions together, but with a delimiter between each value. FIELD Returns an index value relative to the location of a value in a list of values. FIND IN SET Returns the position of a string in a list of strings. FORMAT When a number is sent, this number is formatted to include commas (e.g. 3,400,000). INSERT Used to insert one string into another at a specific point, for a certain number of characters. INSTR Returns the position of the first time one string appears in another. LCASE Convert a string to lowercase. LEFT From left, Extract the specified number of characters from one string and return them as another. LENGTH Returns the length of a string, but in bytes. LOCATE Returns the first instance of a string in another, LOWER same as LCASE. LPAD Left pads one string with another, to a certain length. LTRIM Remove any leading spaces from the specified string. MID Extracts one string from another, from any position. POSITION Returns the position of the first time a substring appears in another. REPEAT Used to repeat a string REPLACE Used to replace all instances of a substring in a string with a new substring. REVERSE Reverses the string. RIGHT From right, extract the specified number of characters from one string and return them as another. RPAD Right pads one string with another, to a certain length. RTRIM Removes any trailing spaces from the specified string. SPACE Returns a string full of spaces that equals the amount you send it. STRCMP compares 2 strings for differences SUBSTR Extracts one substring from another, from any position. SUBSTRING Same as SUBSTR SUBSTRING_INDEX Returns a substring from a string before the sent substring is found the number of times equal to the passed number. TRIM Removes trailing and leading spaces from the specified string. Same as if you were driving and RTRIM together. UCASE Convert a string to uppercase letters. SAME as UCASE. Numeric functions Numeric functions Name Name ABS Returns the absolute value of the specified number. ACOS Returns its arc of the given number. ASIN Returns the arcusus of the given number. ATAN Returns the arc tone of one or two given numbers. ATAN2 Return the arc number of 2 given numbers. AVG Returns the average value of the specified expression. CEIL Returns the closest whole number (integer) up from a given decimal point number. ROOF Same as CEIL. COS Returns the cosine of a given number. COT Returns the cotangent of a given number. COUNT Returns the number of records returned by a SELECT query. DEGREES Converts a radian value to degrees. DIV Allows you to share integer. EXP Returns e to the power of the given number. FLOOR Returns the closest whole number (integer) down from a given decimal point number. GREATEST Returns the highest value in a list of arguments. AT LEAST Returns the smallest value in a list of arguments. LN Returns the natural logarithm of the specified number LOG Returns the natural logarithm of the specified number, or the logarithm of the specified number of the specified base LOG10 Do the same as LOG, but to reason 10. LOG2 Do the same as LOG, but to base 2. MAX Returns the highest value from a set of values. MIN Returns the lowest value from a set of values. MOD Returns the rest of the given number divided by the second given number. PI Returns PI. POW Returns the value of the given number raised to the power of the second given number. POWER Same as POW. RADIANS Converts a degree value to radians. RAND Returns a random number. ROUND Round the given number of the specified number of decimal places. SIGN Returns the character of the given number. SIN Returns the sine of the given number. SQRT Returns the square root of the specified number. SUM Returns the value of the specified value set that is combined. TAN Returns the key to the given number. TRUNCATION Returns a number that is truncated to the specified number of decimal places. Date Functions Date Functions Name Description ADDDATE Add a date range (eg: 10 DAY) to a date (eg: 20/01/20) and return the result (eg: 20/01/30). ADDTIME Add a time interval (e.g.: 2:00) to a time or date time (5:00 a.m.) and return the result (7:00). CURDATE Get the current date. CURRENT_DATE same as CURDATE. CURRENT_TIME Get the current time. CURRENT_TIMESTAMP Get the current date and time. CURTIME Same as CURRENT_TIME. DATE Extracts the date from a datetime expression. DATEDIFF Returns the number of days between the 2 given dates. DATE_ADD same as ADDDATE. DATE_FORMAT Formats the date of the specified pattern. DATE_SUB Draw a date range (e.g.: 10 DAY) to a date (e.g.: 20/01/20) and return the result (e.g.: DAY Returns the day of the specified date. DAYNAME Returns the weekday name for the specified date. DAYOFWEEK Returns the index for the day of the week for the specified date. DAYOFYEAR Returns the day of the year for the given date. EXCERPTS from the given part (e.g. MONTH for 20/01/20 = 01). FROM DAYS Return the date from the specified numeric date value. TIME Return the hour from the given date. LAST DAY Get the last day of the month for the given date. LOCALTIME Retrieves the current local date and time. LOCALTIMESTAMP Same as LOCAL TIME. MAKEDATE Creates a date and returns it, based on the values for given year and number of days. MAKETIME Creates a time and returns it, based on the specified values for hour, minute, and second. MICROSECOND Returns the microsecond of a given time or date/time. MINUTE Returns the time or date/time minute. MONTH Returns the month of the specified date. MONTHNAME Returns the name of the month for the specified date. NOW Same as LOCALTIME. PERIOD_ADD Adds the specified number of months in the specified period. PERIOD_DIFF Returns the difference between 2 given periods. QUARTER Returns the year quarter of the given date. SECOND Returns the second of a given time or date/time. SEC_TO_TIME Returns a time based on the specified seconds. STR_TO_DATE Creates a date and returns it based on the specified string and format. SUBDATE Same as DATE_SUB. SUBTIME Draws a time interval (e.g.: 2:00) to a time or date time (5:00) and returns the result (3:00). SYSDATE Same as LOCALTIME. TIME Returns the time from a given time or date/time. TIME_FORMAT Returns given time in the specified format. TIME_TO_SEC Converts and returns a time in seconds. TIMEDIFF Returns the difference between 2 given time/date/time expressions. TIMESTAMP Returns the datetime value for the specified date or date time. TO_DAYS Returns the total number of days that have gone from '00-00-0000' to the specified date. WEEK Returns the week number of the specified date. WEEKDAY Returns the weekday number for the specified date. UKEOFYEAR Returns the week number of the given date. YEAR Returns the year from the given date. YEAR WEEK Returns the year and week number of the specified date. Miscellaneous functions Miscellaneous Functions Name Description BIN Returns the specified number in binary. BINARY Returns the specified value as a binary string. CAST Convert one type to another. COALESCE From a list of values, return the first non-null value. CONNECTION_ID For the current connection, return the unique connection ID. CONV Convert the given number from one numeric base system to another. CONVERT Convert the specified value to the specified data type or character set. CURRENT_USER Return the user and the host name that were used to authenticate with the server. DATABASE Get the name of the current database. GROUP BY Used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group Example: Displays the number of users with active orders. SELECT COUNT(user_id), active_orders FROM USERS GROUPS BY ACTIVE_ORDERS; HAVING It is used instead of WHERE with aggregated functions. Example: Displays the number of users with active orders but includes only users with more than 3 3 Orders. SELECT COUNT(user_id), active_orders FROM USERS GROUP BY ACTIVE_ORDERS HAVE COUNT(user_id) &gt; 3; IF the condition is true return a value, otherwise return another value. IFNULL If the given expression corresponds to zero, return the specified value. ISNULL If the expression is zero, return 1, otherwise return 0. LAST_INSERT_ID For the last row that was added or updated in a table, automatically return the increment ID. NULLIF compares the two given expressions. If they are similar, NULL is returned, otherwise the first expression is returned. SESSION_USER Return the current user and host names. SYSTEM_USER Same as SESSION_USER. USER Same as SESSION_USER. VERSION Returns the current version of MySQL that powers the database. Wildcards In SQL, wildcards are special characters used with the like and NOT LIKE keywords, which allows us to search for data with advanced patterns much more efficiently Wildcards Name Description % equals zero or more characters. Example 1: Find all users whose last names end in son. SELECT * FROM USERS WHOSE last name as '%son'; Example 2: Find all users who live in cities that contain the pattern 'che' SELECT * FROM users where the city LIKEs '%che%'; _ Corresponds to any single character. Example: Find all users who live in cities that start with 3 characters, followed by chester. CHOOSE * FROM USERS WHERE THE CITY LIKES ___chester; [charlatan] Corresponds to a single character in the list. Example 1: Find all users whose first names begin with J, H, or M. SELECT * FROM users where first_name LIKE '[jhm]%'; Example 2: Find all users whose first names begin letters between A – L. SELECT * FROM users where first_name LIKE '[a-l]%'; Example 3: Find all users whose first names do not end with letters between n – s. SELECT * FROM users where first_name LIKE '%[!n-s]'; Keys In relational databases, there is a concept of primary and foreign keys. In SQL tables, these are included as constraints, where a table can have a primary key, a foreign key, or both. Primary key A primary key allows each record in a table to be uniquely identified. There can only be one primary key per table, and you can assign this constraint to a single or combination of columns. However, this means that each value in this column(s) must be unique. Typically, in a table, the primary key is an ID column, and is usually paired with AUTO_INCREMENT keyword. This means that the value increases automatically as new records are created. Example 1 (MySQL) Create a new table and set the primary key to the ID column. CREATE TABELL users (id int NOT NULL AUTO_INCREMENT, first_name varchar(255), last_name varchar(255) NOT NULL, address varchar (255), email varchar (255), PRIMARY KEY (id)); Example 2 (MySQL) Modify an existing table and set the primary key to first_name CHANGE TABLE users add primary key (first_name); Foreign key A foreign key can be used on one column or many and is used to link 2 tables together in a relational database. As shown in the following chart, the table that contains the foreign key is called the child key, while the table that contains the referenced key, or the candidate key, is called the parent table. This essentially means that the column data is split between 2 tables, as a foreign key also prevents invalid data from being inserted, which does not also exist in the parent table. Example 1 (MySQL) Create a new table and turn all columns that refer to IDs in other tables into foreign keys. CREATE TABELL orders (ID-int NOT NULL, user_id int, product_id int, PRIMARY KEY (iD), FOREIGN KEY (user_id) REFERENCES users (id), FOREIGN KEY (product_id) REFERENCES products (id)); Example 2 (MySQL) Modify an existing table and create a foreign key. CHANGE TABLE ORDERS ADD FOREIGN KEY (user_id) REFERENCES USERS (id); Indexes Indexes are attributes that can be mapped to columns that are often sought against to make data retrieval a faster and more efficient process. This does not mean that each column should be turned into an index, as it takes longer for a column with an index to be updated than a column without. This is because when indexed columns are updated, the index itself must also be updated. Indexes Name Description CREATE INDEX Creates an index named idx_test in the first_name and last name columns in the user table. In this situation, duplicate values are allowed. CREATE INDEX idx_test ON users (first_name, last name); CREATE UNIQUE INDEX The same as above, but no duplicate values. CREATE UNIQUE INDEX idx_test ON users (first_name, last name); DROP INDEX Removes an index. CHANGE TABLE USERS DROP INDEX idx_test; Joins In SQL, a JOIN clause is used to return a result set that combines data from multiple tables, based on a common column discussed in both, There are a number of different joins available for you to use:- Inner join (default): Returns all records that have matching values in both tables. Left Join: Returns all records from the first table, along with matching records from the second table. Right join: Returns all records from the second table, along with matching records from the first one. Full Join: Returns all records from both tables when there is a match. A common way to visualize how joins work is like this: In the following example, an inner join will be used to create a new unifying view that combines the order table and then 3 different tables We replace the columns user_id and product_id with the columns first_name and last names of the user who placed the purchase order, along with the name of the item that was purchased. SELECT orders.id, users.surname, products.name as product name FROM orders INNER JOIN users on orders.user_id = users.id INNER JOIN products on orders.product_id = = Returns a result set that looks like: View A view is essentially an SQL result set that is stored in the database under a label, so you can return to it later without having to run the query again. These are especially useful when you have an expensive SQL query that may be required a number of times, so instead of running it over and over again to generate the same result set, you can only do it once and save it as a view. Create views To create a view, you can do so: CREATE VIEW priority_users AS SELECT * FROM USERS DER LAND = 'United Kingdom'; So in the future, if you need access to the saved result set, you can do it like this: SELECT * FROM [priority_users]; Replacing views with the CREATE OR REPLACE command, a view can be updated. CREATE OR REPLACE VIEW [priority_users] AS SELECT * FROM USERS WHOSE COUNTRIES = 'UNITED KINGDOM' OR COUNTRY='USA'; Delete views To delete a view, simply use the DROP VIEW COMMAND. DROP VIEW priority_users; Conclusion Most sites on today's web usage relationship databases in one way or another. This makes SQL a valuable language to know, as it allows you to create more complex, functional websites and systems. Be sure to bookmark this page, so in the future, if you are working with SQL and do not quite remember a specific operator, how to write a specific query, or are just confused about how joins work, then you will have a cheat sheet on hand that is ready, willing and able to help. Help.