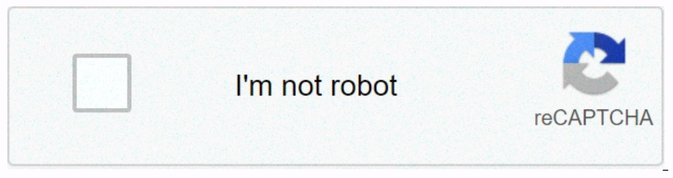

Continue

Swift programming language cheat sheet

In this guide you will find a useful prankster that documents some of the most commonly used elements of SQL and even some of the rarer ones. We hope it will help developers - both beginners and experienced level - become more adept at understanding SQL language. Use this as a quick reference during development, learning help, or even print and bind it if you want (whatever it does!). But before we get to the joker himself, for developers who may not be familiar with SQL, let's start with... PDF version SQL Cheat Sheet SQL Cheat Sheet (Download PDF) Infographic version SQL Cheat Sheet (PNG) SQL Cheat Sheet (Download PNG) Which is SQL SQL indicates structured questionable language. It's the language of choice on today's web to store, manipulate, and retrieve data within relational databases. Most, if not all, websites you visit will use it in some way, including this one. Here's what the basic relational database looks like. This example specifically stores e-commerce information, especially products on sale, customers who buy them, and records of those orders that bind these 2 entities together. You can use SQL to communicate with a database by writing a query, which returns all results that meet its criteria when executed. Here's an example of a query:- SELECT * FROM USERS; Use this select statement to select all data from all columns in the user table. It would then return data such as below, commonly referred to as a result set:- If we replaced the wildcard character character (*) with specific column names instead, only the data from those columns would be returned from the query. SELECT first_name, last_name FROM users; We can add a little complexity to the standard SELECT statement by adding a WHERE clause that allows you to filter what comes back. CHOOSE * FROM PRODUCTS WHERE stock_count < = 10 ORDER STOCK_COUNT ASC; This query would return all data from the product table with stock_count than 10 in the result set. Using the ORDER BY keyword means that results will be ordered using the stock_count, the lowest values to the highest. With INSERTS, we can add new information to the table in the statement. Here is a basic example that adds a new user to the user table:- INSERT U users (first_name, last_name, address, email) VALUES (Tester, Jester, 123 Fake Street, Sheffield, U.K., ); Then, if you were to re-query the return of all data from the user table, the result set would look like this: Of course, these examples show only a very small choice of what SQL language is capable of. SQL vs MySQL You may have already heard of MySQL. It is important not to confuse this with SQL itself, because there is a clear difference. SQL is a language. It describes a syntax that allows you to write queries that control relational databases. Nothing more. MySQL, meanwhile, is a database system that runs on a server. that's SQL language, allowing you to write queries using its 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 using an installer that you can download from the MySQL website. MacOS On macOS, the recommended way to install MySQL is to use native packages, which sounds a lot more complicated than it actually is. Essentially, this also involves only downloading the installer. Alternatively, If you want to use package managers like Homebrew, you can install MySQL as such: brew install mysql While if you need to install an older MySQL version 5.7, which is still widely used on the web today, you can: brew install Using MySQL with MySQL now installed on your system, to get up and go as quickly as possible by writing SQL queries , it is recommended that you use the SQL Management application to make database management a much simpler and simpler process. There are a lot of apps to choose from that largely do the same job, so it comes down to your personal preference on which one to use: MySQL Workbench is being developed by Oracle, which owns MySQL. HeidiSQL (Recommended Windows) is a free open source application for Windows. MacOS and Linux users first need wine as a prerequisite. phpMyAdmin is a very popular alternative that operates in a web browser. The Pro (Recommended macOS) sequel is the only alternative to macOS and our favourite thanks to its clear and simple interface. When you're ready to start writing your own SQL queries, instead of spending time creating your own database, consider importing false data. The MySQL website provides a number of fake databases that you can download for free and then import into your SQL application. Our favorite of these is the world's database, which provides some interesting data to practice writing SQL queries for. Here's a screenshot of his country table as part of sequel pro. This example inquiry returns all countries with Queen Elizabeth II as their head of state. While this one brings back all European countries with a population of more than 50 million along with its capital and its population. And this final returns the average percentage of French speakers in countries where the total number of French speakers is more than 10%. Cheat Sheet keywords A collection of keywords used in SQL statements, description, and, where appropriate, an example. Some of the more advanced keywords have their own dedicated section later in the jokebahter. If MySQL is mentioned next to an example, this example applies only to MySQL databases (unlike any other database system). SQL Keywords Add a new column to an existing table. Example: Adds a new column named to a table called Users. ALTER TABLE USERS ADD ADD varchar (255); ADD A LIMIT Creates a new restriction on an existing table, which is used to specify rules for all data in a table. Example: Adds a new primary key limit named user to column ID and LAST NAME. USERS OF THE ALTER TABLE ADD THE PRIMARY USER KEY OF THE RESTRICTION (ID, SURNAME); ALTER TABLE Adds, deletes, or edits columns in a table. It can also be used to add or delete restrictions in a table, according to the above. Example: Adds a new boolean column named approved to a table called Quotes. ALTER TABLE deals ADD approved boolean; Example 2: Deletes the column approved from the ALTER TABLE quote table of drop column approved; CHANGE COLUMN The table column data type changes. Example: In the table, users create a column incept_date the date type. ALTER TABLE USERS CHANGE the INCEPT_DATE date; ALL returns true if all subdia values meet the adopted condition. Example: Returns users with more tasks than the most tasked users in hr (id 2) SELECT first_name, last name, tasks_no FROM USERS WHERE tasks_no > EVERYONE (CHOOSE TASKS FROM USERS WHERE department_id = 2); And it is used to join separate conditions within the WHERE clause. Example: Returns events located in London, United Kingdom SELECT * FROM events where host_country='United Kingdom' and host_city='London'; ANY return true if any of the search values meet the specified condition. Example: Returns products from a product table that have received orders — stored in an order table — with a quantity greater than 5. SELECT A NAME FROM THE PRODUCT WHERE PRODUCTID = ANY (SELECT PRODUCTID FROM ORDERS WHERE quantity is > 5); AS renames a table or column with an alias value that exists only for the duration of the query. Example: Aliases north_east_user_subscriptions column SELECT north_east_user_subscriptions AS ne_subs FROM users WHERE ne_subs > 5; ASC is used with ORDER BY to restore data in ascending order. Example: Apples, bananas, peaches, raddish between selecting values within a certain range. Example 1: Select inventory with a quantity between 100 and 150. SELECT * FROM INVENTORY WHERE THE QUANTITY IS BETWEEN 100 AND 150; Example 2: Select inventory with a quantity of NO between 100 and 150. Alternatively, using the NOT keyword here invalidates logic and selects values outside a specific range. SELECT * FROM STOCKS WHERE THE QUANTITY IS NOT BETWEEN 100 AND 150; CASE Change query output depending on conditions. Example: Returns users and their subscriptions, along with a new pillar called activity_levels that makes judgment based on the number of subscriptions. CHOOSE first_name, last name, subscription CASE WHEN subscriptions > 10 THEN 'Very active' WHEN Quantity between 3 and 10 THEN 'Active' OTHER 'inactive' ENDS UP AS ACTIVITY_LEVELS FROM USERS; CHECK Adds a limit that limits the value that can be added Example 1 (MySQL): Make sure all users added to the user table are 18 years old or older. CREATE TABLE USERS (first_name (first_name age int, CHECK (age>=18)); Example 2 (MySQL): Adds a check after the table has already been created. ALTER TABLE USERS ADD CHECK (age>=18); CREATING A DATABASE A new database is created. Example: Creates a new database called a site. CREATE DATABASE SITE SETTINGS; CREATE A TABLE You are creating a new table . Example: Creates a new table named users in a site database. CREATE USER TABLES (id int, first_name varchar (255), surname varchar (255), address varchar (255), contact_number int); DEFAULT Sets the default value for a column; Example 1 (MySQL): Creates a new table called Products that has a name column with the default Placeholder Name value and a column available_from with the default value of today's date. CREATE TABLE PRODUCTS (id int, name varchar(255) DEFAULT Placeholder name, available_from GETDATE() date); Example 2 (MySQL): Same as above, but editing an existing table. ALTER TABLE PRODUCTS CHANGE THE NAME OF THE DEFAULT PLACEHOLDER NAME, ALTERNATE AVAILABLE_FROM DEFAULT GETDATE(); DELETE Delete data from the table. Example: Removes user_id from user_id 674. DELETE FROM USER USER_ID = 674; DESC uses ORDER BY to restore data in descending order. Example: Raddish, Peaches, Bananas, Apples DROP COLUMN Deletes a column from the table. Example: Removes first_name column from the user table. ALTER TABLE USERS DROP A COLUMN first_name DROP DATABASE Deletes the entire database. Example: Deletes a database called a site. SETTING UP DROP DATABASE WEBSITES; DROP DEFAULT Removes the default value for the column. Example (MySQL): Removes the default value from the name column in the products table. ALTER DESKTOP PRODUCTS CHANGE THE DEFAULT COLUMN NAME DROP; DROP TABLE Deletes the table from the database. Example: Removes the user table. DROP TABLE USERS; THERE IS a Check for the existence of any record within the underaduity, returning true if one or more records are returned. Example: Lists all dealerships with a job finance percentage of less than 10. SELECT dealership_name from dealership WHERE THERE IS (SELECT DEAL_NAME OFFERS WHERE dealership_id = deals.dealership_id I finance_percentage < 10); The IZ specifies which table to select or delete the data from. Example: Select data from a user's table. SELECT area_manager OF AREA_MANAGERS WHERE IT EXISTS (CHOOSE ProductName FROM a product where area_manager_id = deals.area_manager_id and price < 20); IN Is used with the WHERE clause as an abbreviation for multiple OR terms. So instead of:- CHOOSE * FROM USERS WHERE COUNTRY = 'USA' OR COUNTRY = 'United Kingdom' OR country = 'Russia' OR country = 'Australia'; You can use:- SELECT * FROM USERS IN THE COUNTRY (USA, United Kingdom, Russia, Australia); INSERT IN Add new rows to the table. Example: Adds a new vehicle. INSERT INTO cars (make, model, mileage, year) VALUES (Audi, 30000, 2016); IS NULL Tests for empty (NULL) values. Example: Returns users who have not provided a contact number SELECT * FROM THE USER WHERE WHERE IS NULL; NOT NULL Reverse to NULL. Tests for non-empty / NULL values. LIKE Returns true if the operand value corresponds to the pattern. Example: Returns true if the user's first_name ends with the son. SELECT * FROM USER FIRST_NAME AS '%sin'; NOT Returns true if the record does not meet the condition. Example: Returns true if the user's first_name is not over with the son. SELECT * FROM FIRST_NAME NOT AS '%sin'; OR used with WHERE to include data when any condition is true. Example: Returns users living in Sheffield or Manchester. CHOOSE * FROM USERS WHERE CITY = 'Sheffield' OR 'Manchester'; ORDER BY Used to sort result data in ascending (default) or descending order using ASC or DESC keywords. Example: Returns countries alphabetically CHOOSE * FROM COUNTRIES ORDER BY NAME; ROWNUM Returns results where the number of rows meets the previous condition. Example: Returns the top 10 countries from a country table SELECT * FROM COUNTRIES WHERE ROWNUM <= 10; SELECT Is used to select data from a database, which is then returned in a result set. Example 1: Select all columns from all users. SELECT * FROM THE USER; Example 2: Selects first_name and surnames from all users.xx SELECT first_name, surname from user; SELECT DISTINCT Sames as SELECT, except duplicate values are turned off. Example: Create a backup table using data from a user's table. SELECT * In usersBackup2020 FROM users; SELECT from Copies of data from one table and inserts it into another. Example: Returns all countries from the user table, removing all duplicate values (which would be very likely) SELECT A SEPARATE country from the user; SELECT TOP Allows you to return a certain number of records to return from the table. Example: Returns the first 3 cars from the car table. CHOOSE TOP 3 * FROM THE CAR; SET Used with UPDATE to update existing data in a table. Example: Updates value value value value values of value value values for an order with an id of 642 in the order table. UPDATE order SET value = 19.49, quantity = 2 WHERE id = 642; SOME identical to any. TOP Used with SELECT to return a number of records from a table. Example: Returns the first 5 users from the user table. CHOOSE TOP 5 * FROM USERS; TRUNCATE TABLE Similar to a crash, but instead of deleting the table and its data, it deletes only the data. Example: Blanks the session table, but leaves the table itself intact. ROTTING TABLE SESSIONS; The UNION combines the results of 2 or more SELECT statements and returns only different values. Example: Returns cities from event tables and subscribers. CHOOSE A CITY FROM THE UNION SELECT CITY EVENT FROM SUBSCRIBERS; UNION ALL The same as the UNION, but includes duplicate values. UNIQUE This restriction ensures that all values in the column are unique. Example 1 (MySQL): Adds a unique limit id when creating a new user table. CREATE TABLE Users (id int NOT NULL, name varchar(255) IS NOT NULL, UNIQUE (id)); Example 2 (MySQL): (MySQL): existing column to add a UNIQUE limit. USERS OF THE ALTER TABLE ADD UNIQUE (ID); UPDATE Updates existing data in the table. Example: Updates mileage values and Due 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 INSERT In a 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 results filters include only data that meets the default state. Example: Returns orders with a quantity greater than 1 item. CHOOSE * FROM ORDERS WHERE > 1; Comments Comments allow you to explain parts of your SQL statements or to comment on the code and prevent it from being executed. There are 2 types of comments, one line, and multiline in SQL. Comments from one line Comments of one line start with -. Any text after these 2 characters to the end of the line will be ignored. -- My selected SELECT * USER QUERY; Multiline comments Multiline comments start with /* and end with */. Stretch across multiple rows until closing characters are found. ** This is my chosen inquiry. It grabs all rows of data from the user table */ SELECT * FROM users; /* This is another selected query, which I do not yet want to execute * FROM TASKS; */ MySQL Data types When creating a new table or editing an existing one, you must specify the type of data that each column accepts. In the example below, the data passed to the id column must be int, while first_name varchar data type with a maximum of 255 characters. CREATE USERS TABLE (id int, first_name varchar (255)); String Data Types String Data Types Data Type Type Description CHAR(size) A fixed string of length that can contain letters, numbers, and special characters. The size parameter sets the maximum length of the string, from 0-255 with the default 1. VARCHAR(size) Variable length string similar to CHAR(), but with a maximum string length range of 0 to 65535. BINARY (size) Similar to char() but stores binary byte wires. VARBINARY (size) Similar to VARCHAR() but for binary byte wires. TINYBLOB holds binary large objects (BLOBs) with a maximum length of 255 bytes. TINYTEXT Holds the string with a maximum length of 255 characters. Use VARCHAR() instead, as it is reached much faster. TEXT(size) Holds the string with a maximum length of 65535 bytes. Again, it is better to use VARCHAR(). BLOB (size) holds binary large objects (BLOBs) with a maximum length of 65535 bytes. MEDIUMTEXT Holds the string with a maximum length of 16,777,215 characters. MEDIUMBLOB holds binary large objects (BLOBs) with a maximum length of 16,777,215 bytes. LONGTEXT Holds a string with a maximum length of 4,294,967,295 characters. LONGBLOB holds binary large objects (BLOBs) with a maximum length of 4,294,967,295 bytes. ENUM(s, b, c, etc...) A string object that has only one value, selected from a list up to a maximum of 65535 65535 If a value that is not in this list is added, it is replaced instead with a blank value. Consider ENUM to be similar to HTML radio boxes in this regard. CREATE 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 you define, up to a maximum of 64 values. Consider a set that is similar to HTML checkboxes in this regard. Numeric DataSet String Data Types Data Type Description BIT(size) Bit value type with default 1. The allowed number of bits in the value is set through a size parameter, which can hold values from 1 to 64. TINYINT (size) A very small integer with a signed range from -128 to 127 and an unsigned range from 0 to 255. Here, the size parameter determines the maximum permissible screen width, which is 255. BOOL Basically a quick way to place a column on TINYINT size 1. 0 is considered false, while 1 is considered true. BOOLEAN Same as BOOL. SMALLINT (size) Small integer with signed range from -32768 to 32767 and unsigned range from 0 to 65535. Here, the size parameter determines the maximum allowed screen width, which is 255. MEDIUMINT (size) Mean integer with a signed range from -8388608 to 8388607 and an unsigned range from 0 to 16777215. Here, the size parameter determines the maximum allowed screen width, which is 255. INT(size) Mean integer with a signed range from -2147483648 to 2147483647 and unsigned range from 0 to 4294967295. Here, the size parameter determines the maximum allowed screen width, which is 255. INTEGER(size) Same as INT. BIGINT (size) Mean integer with signed range from -922337203685475808 to 922337203685475807, and unsigned range from 0 to 18446744073709551615. Here, the size parameter determines the maximum allowed screen width, which is 255. FLOAT(p) Floating point number value. If the precision parameter (p) is between 0 and 24, then the data type is set to FLOAT(), while if it is from 25 to 53, the data type is set to DOUBLE(). This behavior is to make value storage more efficient. DOUBLE(size, d) The value of the floating point where the total digits are set by size parameter and the number of digits after the decimal point is set by d parameter. DECIMAL (size, d) The exact number of a fixed point where the total number of digits is set by size parameters and the total number of digits after the decimal point is set by d parameter. For size, the maximum number is 65, and the default number is 10, while for d, the maximum number is 30, and the default number is 10. DEC(size, d) Same as DECIMAL. Date/Time Data Types Date/Time Data Type Data description DATE Simple date in YYYY-MM-DD format, with supported range from '1000-01-01' to '9999-12-31'. DATETIME(fsp) Date time in YYYY-MM-DD hh:mm:ss format, with supported range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding And ON UPDATE to the definition of column, column, is automatically set to the current date/time. TIMESTAMP(fsp) Unix Timestamp, which is the value relative to the number of seconds from Epoch Unix ('1970-01-01 00:00:00' UTC). This has a supported range from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. Adding default CURRENT_TIMESTAMP AND ON UPDATE CURRENT TIMETAMP to the column definition automatically sets it to the current date/time. TIME(fsp) Time in hh:mm:ss format, with supported range from '-838:59:59' to '838:59:59'. year per year, with a supported range from '1901' to '2155'. Operators Arithmetic Operators Arithmetic Operators Operator Description + Add - Unsug * Multiply / Share % Modulo Bitwise Operator Bitwise Operator Description and Bitwise and | Bitwise OR ^ Bitwise exclusive OR Comparison Operators Comparison Operators Description = Equal > Greater Than < Less than >= Greater than or Equal to < = Less or Equals <> Not Equal to Complex Operators Connection Operator Description += Add Equals -= Subtract Equals *= Multiply Functions Equals /= Divide Equals %= Modulo Equals &= Bitwise AND Equalswise ^-= Bit exclusive equals |*= Bit ORwise equals Functions String Description ASCII Returns the equivalent ASCII value for a specific character. CHAR_LENGTH Returns the length of a string character. CHARACTER_LENGTH s not the same as CHAR_LENGTH. CONCAT adds expressions together, with a minimum of 2.CONCAT_WS adds expressions together, but with a separator between each value. FIELD Returns the value of an index relative to the position of a value within a list of values. FIND IN SET Returns the position of a string in a list of wires. FORMAT When a number passes, returns that number formatted to include commas (e.g. 3,400,000). INSERT allows you to insert one wire into another at a specific time, for a certain number of characters. INSTR Returns the position of the first time one string appears inside another. LCASE turn the string into a little trash. LEFT Starting on the left, remove the specified number of characters from the string and restore them as others. LENGTH Returns the length of a string, but in bytes. LOCATE Returns the first occurrence of one string within the second, LOWER Same as LCASE. LPAD Left pads one string with the other, at a certain length. LTRIM Remove all leading spaces from the specified series. MID pulls one string out of the other, starting from any position. POSITION Returns the position of the first occurrence of the substring within the second. REPETITION Allows you to repeat the REPLACE string allows you to replace all cases of substring within a string, with a new substator. REVERSE reverses the string. RIGHT Starting from the right, extract the specified number of characters from the string and restore them as others. RPAD Right pads one string with the other, at a certain length. RTRIM Removes all supporting spaces from the specified string. SPACE Returns a series full of space equal to the amount you pass it. STRCMP compares wires for differences SUBSTR Extracts one substring from another, starting from any position. SUBSTRING The same as substr SUBSTRING_INDEX Returns the substring from a string before being stated past the substring has been found a number of times equal to the number that has passed. TRIM Removes supporting and leading spaces from the specified series. It's the same as if you run LTRIM and RTRIM together. UCASE convert the string to the upper part. TOP Same as UCASE. Numeric functions Of the NUMEric function, the DESCRIPTION of the name ABS returns the absolute value of a specified number. ACOS Returns the arc of a cosine specified number. ASIN returns the arc sine of a specified number. ATAN Returns the arc of the tangent of one or two numbers. ATAN2 Return the bow to the tangent of 2 with respect to the numbers. AVG Returns the average value of a specific expression. CEIL Returns the closest integer (integer) upwards of a specified decimal point number. CEILING Same as CEIL. COS Returns a mazin of a specified number. COT returns the co-agent of a specified number. COUNT Returns the quantity of records returned by select query. DEGREES converts radian value to degrees DIV allows you to share integers. EXP Returns e to the strength of a specified number. FLOOR Returns the nearest integer (integer) down from a specified decimal point number. GREATEST Returns the largest value in the list of arguments. AT LEAST returns the smallest value in the list of arguments. LN Returns the natural logarithm of a specified number LOG Returns the natural logarithm of a specified number, or the logarithm of a specified number to a specific LOG10 base It does the same as LOG, but for base 2.MAX returns the highest value from a set of values. MIN Returns the lowest value from a set of values. The MOD returns the rest of a given number divided by another number. PI returns pi. POW Returns the value of a number raised to the strength of another specified number. POWER Same as POW. RADIANS converts a degree value to radians. RAND returns a random number. ROUND Round days number to a certain amount of decimal places. SIGN Returns the character of a number. SIN returns the sine of a specified number. SQRT Returns the square root of a given number. SUM Returns the value of a specific set of values together. TAN returns the tangent of a number. TRUNCATE Returns a number shortened to a specified number of decimal places. Date of the ADDDATE Name Description function Date Add a date interval (e.g.: 10 DAYS) to the date (e.g. ADDTIME Add a time interval (e.g. 02:00) at a time or date (05:00) and return the result (07:00). CURDATE Get the current date. CURRENT_DATE like CURDATE. CURRENT_TIME Get the current time. CURRENT_TIMESTAMP Get the current date and time. CURTIME Same as CURRENT_TIME. DATE Extracts a date from a date expression. DATEDIFF Returns the number of days between 2 days. DATE_ADD same as ADDDATE. DATE_FORMAT date of a particular sample. DATE_SUB a date interval (e.g. day 10) to date (e.g. DAY Returns the day for a given date. DAYNAME Returns the workday name for a given date. DAYOFWEEK Returns the index for the working day for a given date. DAYOFYEAR Returns the day of the year for a given date. Extract of extract from the date of the given part (e.g. MONTH for 20/01/20 = 01). FROM DAY Return the date from the default numeric date of the value. SAT Return the clock from the specified date. LAST DAY Get the last day of the month for the given date. LOCAL TIME Gets the current local date and time. LOCALTIMESTAMP IS THE SAME AS LOCAL TIME. MAKEDATE Creates a date and returns it, based on a specific year value and the number of days. MAKETIME Creates time and returns it, based on the default values of the watch, minute, and second. MICROSECOND Returns the microsecond of a specific time or date. MINUTE Returns a minute of a specific time or date. The moon returns a month. MONTHNAME Returns the month name of a specific date. NOW THE SAME AS LOCAL TIME. PERIOD_ADD the specified number of months to a specific period. PERIOD_DIFF Returns the difference between 2 specific periods. QUARTER Returns the quarter of the year for a given date. ANOTHER returns another specific time or date. SEC_TO_TIME Returns the time based on the specified seconds. STR_TO_DATE Creates a date and returns it based on the default string and format. SUBDATE Same as DATE_SUB. SUBTIME Subima subtracts the time interval (e.g. SYSDATE THE SAME AS LOCAL TIME. TIME returns a time from a specific time or date. TIME_FORMAT Returns a specific time in the specified format. TIME_TO_SEC converts and returns time per second. TIMEDIFF Returns the difference between 2 days of expression time/date. TIMESTAMP Returns a date to a value of a specific date or date. TO_DAYS Returns the total number of days that have passed from '00-00-0000' to the given date. WEEK Returns the number of weeks for a specific date. WORKDAY Returns the number on a weekday for a specific date. WEEKOFYEAR Returns the week number for a specific date. YEAR Returns a year from a given date. YEARWEEK Returns the year and week number for a given date. Misc function misc function of bin name description Returns a specific number in binary. BINARY Returns a specific value as a binary string. CAST convert one type to another. COALESCE Return the first non-null value from the list of values. CONNECTION_ID Convert a specific number from one numeric base system to another for the current connection. Convert the default value to the default data set or characters. CURRENT_USER Restore the user and hostname that was used to authenticate with the server. DATABASE Get the name of the current database. GROUP PO It is used in addition to aggregate functions (COUNT, MAX, MIN, SUM, AVG) for the result group. Example: number of users with active orders. SELECT COUNT (user_id), ACTIVE_ORDERS FROM GROUP USERS PER ACTIVE_ORDERS; ONCE used in a place where with aggregate functions. Example: Specifies the number of users with active orders, but includes only users with more than 3 active orders. SELECT A NUMBER (user_id), ACTIVE_ORDERS USER GROUP BY ACTIVE_ORDERS NUMBER (user_id) > 3; If the status is true, return the value, otherwise return another value. IFNULL If the default expression is equal to null and null, return the default value. ISNULL If the expression is null and void, return 1, otherwise return 0. LAST_INSERT_ID For the last row that has been added or updated in the table, return the Auto Magnification ID. NULLIF Compares 2 default expressions. If they are equal, NULL is returned, otherwise the first expression is returned. SESSION_USER Restore current user and hostname. SYSTEM_USER s not the same as SESSION_USER. USER Same as SESSION_USER. VERSION Restores the current version of mysql power to the database. Wildcard Characters In SQL, Wildcard Characters are special characters used with LIKE and NOT LIKE keywords that allow us to search data with sophisticated patterns much more efficientLy The description of the Wildcards % name equates to zero or more characters. Example 1: Find all

users with surnames ending with a son. SELECT * FROM USERS WHERE A LAST NAME LIKE %sin; Example 2: Find all users living in cities that contain a sample 'che' SELECT * FROM USERS WHERE CITY LIKE '%che%'; _ Equates to any single character. Example: Find all users living in cities starting with any 3 characters, followed by Chester. CHOOSE * FROM USERS WHERE A CITY LIKE ___CHESTER; [charlah] Equals any single character in the list. Example 1: Find all users with names starting with J, H, or M. SELECT * FROM users where first_name LIKE '[jhm]%'; Example 2: Find all users with names starting in letters between A- L. SELECT * OD users first_name like '[a-l]%'; Example 3: Find all users with names that do not end in letters between n-s. SELECT * FROM user first_name LIKE '%[!n-s]'; Keys In relational databases there is a concept of primary and foreign keys. In SQL tables, this is included as a constraint, where a table can have a primary key, a foreign key, or both. The primary key Primary key allows each record in the table to be uniquely identified. There can only be one primary key per table, and you can assign this restriction to any single or combination of columns. However, this means that each value within this column 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 automatically increases as new records are created. Example 1 (MySQL) Create a new table and set the primary key for the column ID. CREATE TABLES Users (id int NOT NULL first_name varchar (255), last_name varchar (255) NE NULL, adresa varchar(255), e-mail varchar (255), PRIMARNI KLJUČ (id));); 2 (MySQL) Modify an existing table and set the primary key first_name column. ALTER TABLE USERS ADD A PRIMARY KEY (FIRST_NAME); A foreign key A foreign key can be applied to one column or many and is used to connect 2 tables together in a database relationship. As seen in the diagram below, the table containing the foreign key is called a children's key, while the table containing the candidate's reference key or key is called the parent table. This basically means that column data is divided between 2 tables because the foreign key also prevents invalid data from being inserted that is also not present in the parent table. Example 1 (MySQL) Create a new table and convert all columns related to IDs in other tables into foreign keys. CREATE TABLE 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. ALTER ORDER TABLE ADD FOREIGN KEY (user_id) REFERENCE USERS(ID); Index indices are attributes that can be assigned to columns that are frequently searched to make retrieving data a faster and more efficient process. This does not mean that each column should be created into an index because it takes longer for the index column to be updated than the column without. This is because when indexes are indexed columns are updated, the index itself must also be updated. Create INDEX Name Description Index Creates an index called 'idx_test' on first_name and last name columns of the user table. Duplicate values are allowed in this case. CREATE INDEX idx_test ON users (first_name, surname); CREATE A UNIQUE INDEX The same as the above, but without duplicate values. CREATE A UNIQUE INDEX idx_test users (first_name, surname); INDEX CRASH The index is removed. ALTER TABLE USERS CANCEL IDX_TEST; Joined to SQL, the JOIN clause is used to return a result set that combines data from multiple tables, based on a common column located in both numbers, there are a number of different joins available for use:- Internal join(Default): Returns all records that have corresponding values in both tables. Left Join: Returns all records from the first table, along with any corresponding records from the second table Right Join: Returns all records from another table, along with any corresponding records from the first. Full Join: Returns all records from both tables when a match exists. The usual way to visualize how to connect work is this: In the following example, an internal port will be used to create a new ununited view that combines the order table, and then 3 different tables We will replace user_id and product_id with the first_name and last names columns of the user who ordered, along with the name of the purchased item. SELECT orders.id, users.first_name, users.last name, as 'product name' from INNER JOIN orders to users orders.user_id = users.id users.id JOIN products on orders.product_id = products.id; It would return a result set that looks like: View view is basically a set of SQL results that are stored in a database under a label, so you can return to it later without having to repute the query. They are especially useful when you have a set of SQL queries that can be needed several times, so instead of running it over and over again to generate the same set results, you can just do it once and save it as a view. To create a view view, you can do this like this: CREATE PRIORITY_USERS SELECT * FROM USERS WHERE COUNTRY = 'United Kingdom'; Then in the future, if you need to access the stored set of results, you can do it like this: CHOOSE * FROM [priority_users]; By replacing the view with create or replace, the view can be updated. CREATE OR REPLACE A VIEW [PRIORITY_USERS] ACCORDING TO THE SELECTED * FROM THE USER IN WHICH COUNTRY = UNITED KINGDOM OR COUNTRY=USA; Delete a view To delete a view, simply use the DROP VIEW command. DROP VIEW priority_users; Conclusion Most websites on today's web some way use relational databases. This makes SQL a valuable language to know, as it allows you to create more complex, functional websites and systems. Be sure to mark this page, so in the future, if you work with SQL and you can not think of a specific operator, how to write a particular query or you are just confused about how it joins the work, then you will have a cheater at hand who is ready, willing and able to help. Help.