**DATABASES**Relational (SQL) V Non-relational (noSQL)  
  
Structured Query Language:

|  |  |  |
| --- | --- | --- |
|  | Data Definition Language (**DDL**) | Data Manipulation Language (**DML**) |
| Create | CREATE | INSERT INTO |
| Read | SHOW | SELECT |
| Update | ALTER | UPDATE |
| Delete | DROP | DELETE |

**Data Types**: Numeric, Text & Date/Time

**Constraints**:   
- UNIQUE: no repeated entry  
- NOT NULL: cannot be left blank  
- DEFAULT: if left blank will be set to a default  
- PRIMARY KEY: record number, usually id field  
- FOREIGN KEY: a primary key in another table. Any table with a FK will be a child of another table

ALTER TABLE table\_name  
ADD column\_name DATATYPE CONSTRAINTS;

ALTER TABLE table\_name  
DROP column\_name;

ALTER TABLE table\_name   
MODIFY column\_name DATATYPE CONSTRAINTS;

CREATE DATABASE kfc;  
USE kfc;

CREATE TABLE customers(  
id INT UNIQUE NOT NULL AUTO\_INCREMENT,  
first\_name VARCHAR(255) DEFAULT "unknown",  
last\_name VARCHAR(255) NOT NULL,  
phone\_no VARCHAR(11) NOT NULL,  
over\_18 BOOLEAN,  
points DECIMAL(6,2),  
PRIMARY KEY(id)  
);

SHOW TABLES;  
DESCRIBE customers;

# HOW TO UPDATE TABLE COLUMN

ALTER TABLE customers ADD age INT NOT NULL;  
ALTER TABLE customers DROP COLUMN over\_18;  
ALTER TABLE customers MODIFY first\_name VARCHAR(255) DEFAULT "N/A";

UPDATE table\_name

SET column1 = value1, column2 = value2...., columnN = valueN

WHERE [condition];

At end of foreign key:  
- FOREIGN KEY (customer\_id) REFERENCES customers(id) ON DELETE CASCADE;

UPDATE orders SET total\_price=10.00 WHERE order\_id =1;

INSERT INTO order\_items(order\_id, item\_id, quantity) VALUES(1,3,1),(2,4,5)(2,2,1);

READ from tables to view records:  
SELECT \* FROM table\_name;  
SELECT col1, col3 FROM table\_name;  
**WHERE** clause –  
SELECT \* FROM table\_name WHERE col\_name=value;  
SELECT \* FROM orders WHERE id=2;

Operators:  
= > < != >= <=

SELECT \* FROM orders WHERE price BETWEEN 1.00 AND 3.00;  
BETWEEN is inclusive (like >= and <=)

**Looking for patterns in text**:   
SELECT \* FROM items WHERE item LIKE “%ash”

WHERE name LIKE ‘\_r%’

SELECT DISTINCT col\_name FROM table;

SELECT \* FROM table ORDER BY column DESC LIMIT 2;

ADDING FOREIGN KEY TO TABLE IF NOT DONE ON CREATION:  
  
ALTER TABLE table\_name   
ADD FOREIGN KEY (column\_name)   
REFERENCES table   
HAVING PRIMARY KEY(column\_name);

**### AGGREGATE FUNCTIONS ###**

# When you have a range of data and want to perform an operation on them to return a single values  
- average: AVG  
- sum: SUM  
- minimum: MIN  
- maximum: MAX  
- count: COUNT

* **[HOME](https://www.makeuseof.com/)**
* [**PROGRAMMING**](https://www.makeuseof.com/category/programming/)

Readers like you help support MUO. When you make a purchase using links on our site, we may earn an affiliate commission.

**The Essential SQL Commands Cheat Sheet for Beginners**

**BY**[**JOE COBURN**](https://www.makeuseof.com/author/joecoburn/)

PUBLISHED FEB 29, 2020

Want to learn more about SQL? Having a handle on the various SQL query commands is a great way to get ahead.

Structured Query Language (SQL) is the tool of choice for manipulating databases. This language lets you manipulate data like no other, and it costs nothing to start using!

If you're new to SQL commands or need a reminder of some parts of this powerful tool at your disposal, then our SQL commands cheat sheet below is exactly what you need. Scroll down to see the awesomeness it contains, or download the PDF for future use. (The PDF contains examples for the given commands.)

**FREE DOWNLOAD:** This cheat sheet is available as a **downloadable PDF** from our distribution partner, TradePub. You will have to complete a short form to access it for the first time only. Download [**The Essential SQL Commands Cheat Sheet**](https://makeuseof.tradepub.com/c/pubRD.mpl?secure=1&sr=pp&_t=pp:&qf=w_makc87&ch=CSPSQLCS).

**The Essential SQL Commands Cheat Sheet**

| **Command** | **Action** |
| --- | --- |
| **Query Commands** | |
| SELECT | Basic query building block to retrieve data. |
| SELECT \* | Using \* with SELECT returns all columns. |
| SELECT column | Specify exact columns with their name. |
| SELECT table.column | Reference a column from a specific table. |
| FROM | Specify where to find data. |
| AS | Temporarily alias a table name or column to a new name. |
| WHERE | Filter results with a condition. |
| AND | Use multiple conditions with a WHERE clause. Results must match all conditions. |
| OR | Use multiple conditions with a WHERE clause. Results only need to match one condition. |
| ORDER BY | Order the results by a column. The database chooses how to order. |
| ORDER BY column ASC | Order the results by a column in ascending order. |
| ORDER BY column DESC | Order the results by a column in descending order. |
| LIMIT | Restrict the number of results returned. |
| OFFSET | Skip the first OFFSET number of rows. Often used with LIMIT. |
| SUBQUERY | Run a query to retrieve data for another query. |
| **Aggregate Functions¹** | |
| COUNT | Count the number of rows that match the query. |
| MAX | Return the highest value in a numeric column. |
| MIN | Return the lowest value in a numeric column. |
| SUM | Sum the values of a numeric column. |
| AVG | Calculate the average value for a numeric column. |
| HAVING | Used with aggregate functions instead of the WHERE clause. |
| GROUP BY | Used to refine an aggregate result. |
| **Operators** | |
| LIKE | Case-sensitive search for a pattern with a wildcard operator (%). |
| ILIKE | Case-insensitive search for a pattern with a wildcard operator (%). |
| BETWEEN | Search for a value between two values. Works with dates or numbers. |
| > | Search for values greater than a condition. |
| >= | Search for values greater or equal to a condition. |
| < | Search for values less than a condition. |
| <= | Search for values less than or equal to a condition. |
| = | Search for values matching a condition exactly. |
| <> | Search for values not equal to a condition. |
| UNION | Combine two unique queries (with the same columns) into one result. |
| UNION ALL | Combine two queries (with the same columns) into one result. Duplicates allowed. |
| IN | Shorthand for WHERE. Specifies multiple OR conditions. |
| NOT IN | Shorthand for WHERE. Specifies multiple OR conditions (inverted) or not equal to. |
| IS NULL | Check for empty values. |
| IS NOT NULL | Check for no empty values. |
| INTERSECT | Return results which match two queries. |
| MINUS | Return results in one query which are not in another query.¹ |
| **Joins** | |
| ON | Used to specify the column to compare and match results. |
| USING | Shorthand for ON, used when the column name is the same in both tables. |
| LEFT OUTER JOIN | All the results from the left table, with only the matching results from the right table. |
| LEFT OUTER JOIN (WITH NULL) | (With null) All the results from the left table but not in the right table. |
| INNER JOIN | All the results that match in both the left and right tables. |
| FULL OUTER JOIN | All the results from both the left and right tables. |
| FULL OUTER JOIN (WITH NULL) | (With null) all the results from both the left and right tables excluding results in both tables. |
| RIGHT OUTER JOIN | All the results from the right table, with only the matching results from the left table. |
| RIGHT OUTER JOIN (WITH NULL) | (With null) All the results from the right table but not in the left table. |
| **Creating and Editing Tables** | |
| CREATE TABLE | Create a new table. |
| NULL | Allow empty values for this field. |
| NOT NULL | Don't allow empty values for this field. |
| DEFAULT | A value to populate the field with if one is not supplied. |
| AS | Create a new table based on the structure of an existing table. The new table will contain the data from the old table. |
| ALTER TABLE (ADD COLUMN) | Add a new column to an existing table. |
| ALTER TABLE (DROP COLUMN) | Remove a column from an existing table. |
| ALTER TABLE (ALTER COLUMN) | Change the datatype of an existing column. |
| ALTER TABLE (RENAME COLUMN) | Rename an existing column. |
| ALTER TABLE (RENAME TABLE) | Rename an existing table. |
| ALTER TABLE (MODIFY NULL) | Allow null values for a column. |
| ALTER TABLE (MODIFY NOT NULL) | Prevent null values for a column. |
| DROP TABLE | Delete a table and all its data. |
| TRUNCATE TABLE | Delete all the data in a table, but not the table itself. |
| **Constraints** | |
| PRIMARY KEY | A value that uniquely identifies a record in a table. A combination of NOT NULL and UNIQUE. |
| FOREIGN KEY | References a unique value in another table. Often a primary key in the other table. |
| UNIQUE | Enforce unique values for this column per table. |
| CHECK | Ensure values meet a specific condition. |
| INDEX (CREATE) | Optimise tables and greatly speed up queries by adding an index to a column. |
| INDEX (CREATE UNIQUE) | Create an index that does not allow duplicate values. |
| INDEX (DROP) | Remove an index. |
| **Creating and Editing Data** | |
| INSERT (SINGLE VALUE) | Add a new record to a table. |
| INSERT (MULTIPLE VALUES) | Add several new records to a table. |
| INSERT (SELECT) | Add records to a table, but get the values from an existing table. |
| UPDATE (ALL) | Modify all existing records in a table. |
| UPDATE (WHERE) | Modify existing records in a table which match a condition. |
| DELETE (ALL) | Remove all records from a table. |
| DELETE (WHERE) | Remove records from a table which match a condition. |
| **Creating and Editing Triggers¹** | |
| CREATE TRIGGER | Create a trigger. |
| CREATE TRIGGER (OR MODIFY) | Create a trigger, or update an existing trigger if one is found with the same name. |
| WHEN (BEFORE) | Run the trigger before the event happens. |
| WHEN (AFTER) | Run the trigger after the event happens. |
| EVENT (INSERT) | Run the trigger before or after an insert happens. |
| EVENT (UPDATE) | Run the trigger before or after an update happens. |
| EVENT (DELETE) | Run the trigger before or after a delete happens. |
| ON | Which table to target with this trigger. |
| TRIGGER\_TYPE (FOR EACH ROW) | Execute the trigger for every row changed. |
| TRIGGER\_TYPE (FOR EACH STATEMENT) | Execute the trigger once per SQL statement, regardless of how many rows are altered. |
| EXECUTE | Keyword to indicate the end of the main trigger definition. |
| DROP TRIGGER | Delete a trigger. |
| **Creating and Editing Views** | |
| CREATE VIEW | Create a new view. |
| AS | Define where to retrieve the data for a view. |
| WITH CASCADED CHECK OPTION | Ensure any data modified through a view meets the rules defined by the rule. Apply this to any other views. |
| WITH LOCAL CHECK OPTION | Ensure any data modified through a view meets the rules defined by the rule. Ignore this for any other views. |
| CREATE RECURSIVE VIEW | Create a recursive view (one that refers to a recursive common table expression). |
| CREATE TEMPORARY VIEW | Create a view that exists for the current session only. |
| DROP VIEW | Delete a view. |
| **Common Table Expressions (CTEs)¹** | |
| WITH | Create a new common table expression. |
| AS | Specify the data to use in the CTE. |
| , (COMMA) | Chain multiple CTEs. |