SQL Cheat Sheet

Mosh Hamedani

Code with Mosh (codewithmosh.com)

1st Edition

About this Cheat Sheet

This cheat sheet includes the materials I've covered in my SQL tutorial for Beginners on YouTube.

https://youtu.be/7S_tz1z_5bA

Both the YouTube tutorial and this cheat cover the core language constructs and they are not complete by any means.

If you want to learn everything SQL has to offer and become a SQL expert, check out my Complete SQL Mastery Course.

Use the **coupon code CHEATSHEET** upon checkout to get this course with a 90% discount:

https://codewithmosh.com/p/complete-sql-mastery/

About the Author

Hi! My name is Mosh Hamedani. I'm a software engineer with two decades of experience and I've taught over three million how to code or how to become a professional software engineer. It's my mission to make software engineering simple and accessible to everyone.

https://codewithmosh.com

https://youtube.com/user/programmingwithmosh

https://twitter.com/moshhamedani

https://facebook.com/programmingwithmosh/

Basics

```sql
USE sql_store;

SELECT *
FROM customers
WHERE state = 'CA'
ORDER BY first_name
LIMIT 3;
```

- SQL is **not** a case-sensitive language.
- In MySQL, every statement must be terminated with a semicolon.

Comments

We use comments to add notes to our code.

```sql
-- This is a comment and it won't get executed.
```

SELECT Clause

```sql
-- Using expressions

SELECT (points * 10 + 20) AS discount_factor
FROM customers
```

Order of operations:

- Parenthesis
- Multiplication / division
- Addition / subtraction

```sql
-- Removing duplicates

SELECT DISTINCT state
FROM customers
```

WHERE Clause

We use the WHERE clause to filter data.

Comparison operators:

- Greater than: >
- Greater than or equal to: >=
- Less than: <
- Less than or equal to: <=
- Equal: =
- Not equal: <>
- Not equal: !=

Logical Operators

```
-- AND (both conditions must be True)
SELECT *
FROM customers
WHERE birthdate > ‘1990-01-01’ AND points > 1000


-- OR (at least one condition must be True)
SELECT *
FROM customers
WHERE birthdate > ‘1990-01-01’ OR points > 1000

-- NOT (to negate a condition)
SELECT *
FROM customers
WHERE NOT (birthdate > ‘1990-01-01’)
```

IN Operator

```
-- Returns customers in any of these states: VA, NY, CA
SELECT *
FROM customers
WHERE state IN (‘VA’, ‘NY’, ‘CA’)
```

BETWEEN Operator

```
SELECT *
FROM customers
WHERE points BETWEEN 100 AND 200
```

LIKE Operator

```
-- Returns customers whose first name starts with b
SELECT *
FROM customers
WHERE first_name LIKE ‘b%’
```

- %: any number of characters
- _: exactly one character

REGEXP Operator

```
-- Returns customers whose first name starts with a
SELECT *
FROM customers
WHERE first_name REGEXP ‘^a’
```

- ^: beginning of a string
- $: end of a string
- |: logical OR
- [abc]: match any single characters
- [a-d]: any characters from a to d

More Examples

```
-- Returns customers whose first name ends with EY or ON
WHERE first_name REGEXP 'ey$|on$'

-- Returns customers whose first name starts with MY
-- or contains SE
WHERE first_name REGEXP '^my|se'

-- Returns customers whose first name contains B followed by
-- R or U
WHERE first_name REGEXP 'b[ru]'
```

IS NULL Operator

```
-- Returns customers who don't have a phone number
SELECT *
FROM customers
WHERE phone IS NULL
```

ORDER BY Clause

```
-- Sort customers by state (in ascending order), and then
-- by their first name (in descending order)
SELECT *
FROM customers
ORDER BY state, first_name DESC
```

LIMIT Clause

```
-- Return only 3 customers
SELECT *
FROM customers
LIMIT 3
```

```sql
-- Skip 6 customers and return 3
SELECT *
FROM customers
LIMIT 6, 3
```

Inner Joins

```sql
SELECT *
FROM customers c
JOIN orders o
   ON c.customer_id = o.customer_id
```

Outer Joins

```sql
-- Return all customers whether they have any orders or not
SELECT *
FROM customers c
LEFT JOIN orders o
   ON c.customer_id = o.customer_id
```

USING Clause

If column names are exactly the same, you can simplify the join with the USING clause.

```sql
SELECT *
FROM customers c
JOIN orders o
   USING (customer_id)
```

Cross Joins

```sql
-- Combine every color with every size
SELECT *
FROM colors
CROSS JOIN sizes
```

Unions

```
-- Combine records from multiple result sets
SELECT name, address
FROM customers
UNION
SELECT name, address
FROM clients
```

Inserting Data

```
-- Insert a single record
INSERT INTO customers(first_name, phone, points)
VALUES (‘Mosh’, NULL, DEFAULT)
```

```
-- Insert multiple single records
INSERT INTO customers(first_name, phone, points)
VALUES
    (‘Mosh’, NULL, DEFAULT),
    (‘Bob’, ‘1234’, 10)
```

Want to Become a SQL Expert?

If you’re serious about learning SQL and getting a job as a software developer or data scientist, I highly encourage you to enroll in my Complete SQL Mastery Course. Don’t waste your time following disconnected, outdated tutorials. My Complete SQL Mastery Course has everything you need in one place:

- 10 hours of HD video
- Unlimited access - watch it as many times as you want
- Self-paced learning - take your time if you prefer
- Watch it online or download and watch offline
- Certificate of completion - add it to your resume to stand out
- 30-day money-back guarantee - no questions asked

The price for this course is $149 but the first 200 people who have downloaded this cheat sheet can get it for $12.99 using the coupon code **CHEATSHEET**:

https://codewithmosh.com/p/complete-sql-mastery/

SQL COMMANDS
CHEAT SHEET

S Q L   C o m m a n d s

The commands in SQL are called Queries and they are of two types:

- Data Definition Query: The statements which defines the structure of a database, create tables, specify their keys, indexes and so on
- Data manipulation queries: These are the queries which can be edited.

  E.g.: Select, update and insert operation

Command	Syntax	Description
ALTER table	ALTER TABLE table_name ADD column_name datatype;	It is used to add columns to a table in a database
AND	SELECT column_name(s) FROM table_name WHERE column_1 = value_1 AND column_2 = value_2;	It is an operator that is used to combine two conditions
AS	SELECT column_name AS 'Alias' FROM table_name;	It is an keyword in SQL that is used to rename a column or table using an alias name
BETWEEN	SELECT column_name(s) FROM table_name WHERE column_name BETWEEN value_1 AND value_2;	It is an operator used to filter the result within a certain range
CASE	SELECT column_name, CASE WHEN condition THEN 'Result_1' WHEN condition THEN 'Result_2' ELSE 'Result_3' END FROM table_name;	It is a statement used to create different outputs inside a SELECT statement
COUNT	SELECT COUNT(column_name) FROM table_name;	It is a function that takes the name of a column as argument and counts the number of rows when the column is not NULL
Create TABLE	CREATE TABLE table_name (column_1 datatype, column_2 datatype, column_3 datatype);	It is used to create a new table in a database and specify the name of the table and columns inside it
GROUP BY	SELECT column_name, COUNT(*) FROM table_name GROUP BY column_name;	It is an clause in SQL used for aggregate functions in collaboration with the SELECT statement
HAVING	SELECT column_name, COUNT(*) FROM table_name GROUP BY column_name HAVING COUNT(*) > value;	It is used in SQL because the WHERE keyword cannot be used in aggregating functions
INNER JOIN	SELECT column_name(s) FROM table_1 JOIN table_2 ON table_1.column_name = table_2.column_name;	It is used to combine rows from different tables if the Join condition goes TRUE
INSERT	INSERT INTO table_name (column_1, column_2, column_3) VALUES (value_1, 'value_2', value_3);	It is used to add new rows to a table
IS NULL/ IS NOT NULL	SELECT column_name(s) FROM table_name WHERE column_name IS NULL;	It is a operator used with the WHERE clause to check for the empty values
LIKE	SELECT column_name(s) FROM table_name WHERE column_name LIKE pattern;	It is an special operator used with the WHERE clause to search for a specific pattern in a column
LIMIT	SELECT column_name(s) FROM table_name LIMIT number;	It is a clause to specify the maximum number of rows the result set must have
MAX	SELECT MAX(column_name) FROM table_name;	It is a function that takes number of columns as an argument and return the largest value among them
MIN	SELECT MIN(column_name) FROM table_name;	It is a function that takes number of columns as an argument and return the smallest value among them
OR	SELECT column_name FROM table_name WHERE column_name = value_1 OR column_name = value_2;	It is an operator that is used to filter the result set to contain only the rows where either condition is TRUE
ORDER BY	SELECT column_name FROM table_name ORDER BY column_name ASC | DESC;	It is a clause used to sort the result set by a particular column either numerically or alphabetically
OUTER JOIN	SELECT column_name(s) FROM table_1 LEFT JOIN table_2 ON table_1.column_name = table_2.column_name;	It is sued to combine rows from different tables even if the condition is NOT TRUE
ROUND	SELECT ROUND(column_name, integer) FROM table_name;	It is a function that takes the column name and a integer as an argument, and rounds the values in a column to the number of decimal places specified by an integer
SELECT	SELECT column_name FROM table_name;	It is a statement that is used to fetch data from a database
SELECT DISTINCT	SELECT DISTINCT column_name FROM table_name;	It is used to specify that the statement is a query which returns unique values in specified columns
SUM	SELECT SUM(column_name) FROM table_name;	It is function used to return sum of values from a particular column
UPDATE	UPDATE table_name SET some_column = some_value WHERE some_column = some_value;	It is used to edit rows in a table
WHERE	SELECT column_name(s) FROM table_name WHERE column_name operator value;	It is a clause used to filter the result set to include the rows which where the condition is TRUE
WITH	WITH temporary_name AS (SELECT * FROM table_name) SELECT * FROM temporary_name WHERE column_name operator value;	It is used to store the result of a particular query in a temporary table using an alias
DELETE	DELETE FROM table_name WHERE some_column = some_value;	It is used to remove the rows from a table
AVG	SELECT AVG(column_name) FROM table_name;	It is used to aggregate a numeric column and return its average

Commands and syntax for querying data from Single Table	Commands and syntax for querying data from Multiple Table

SELECT c1 FROM t
To select the data in Column c1 from table t
	SELECT c1, c2
	FROM t1
	INNER JOIN t2 on condition
	Select column c1 and c2 from table t1 and perform an inner join between t1 and t2

SELECT * FROM t
To select all rows and columns from table t
	SELECT c1, c2
	FROM t1
	LEFT JOIN t2 on condition
	Select column c1 and c2 from table t1 and perform a left join between t1 and t2

SELECT c1 FROM t
WHERE c1 = 'test'
To select data in column c1 from table t, where c1=test
	SELECT c1, c2
	FROM t1
	RIGHT JOIN t2 on condition
	Select column c1 and c2 from table t1 and perform a right join between t1 and t2

SELECT c1 FROM t
ORDER BY c1 ASC (DESC)
To select data in column c1 from table t either in ascending or descending order
	SELECT c1, c2
	FROM t1
	FULL OUTER JOIN t2 on condition
	Select column c1 and c2 from table t1 and perform a full outer join between t1 and t2

SELECT c1 FROM t
ORDER BY c1LIMIT n OFFSET offset
To skip the offset of rows and return the next n rows
	SELECT c1, c2
	FROM t1
	CROSS JOIN t2
	Select column c1 and c2 from table t1 and produce a Cartesian product of rows in a table

SELECT c1, aggregate(c2)
FROM t
GROUP BY c1
To group rows using an aggregate function
	SELECT c1, c2
	FROM t1, t2
	Select column c1 and c2 from table t1 and produce a Cartesian product of rows in a table

SELECT c1, aggregate(c2)
FROM t
GROUP BY c1HAVING condition
Group rows using an aggregate function and filter these groups using 'HAVING' clause
	SELECT c1, c2
	FROM t1 A
	INNER JOIN t2 B on condition
	Select column c1 and c2 from table t1 and join it to itself using INNER JOIN clause

IntelliPaat

FURTHERMORE:

SQL Developer, SQL DBA Training Masters Program