50 MySQL Query Snippets for Beginners

Writing SQL queries is hard. These 50 snippets should help you to write faster and better.

#1 Show all databases

```
SHOW DATABASES;
```

#2 Use a specific database

```
USE database_name;
```

#3 Show all tables in a database

```
SHOW TABLES;
```

#4 Describe a table's structure

```
DESCRIBE table_name;
```

#5 Create a new database

```
CREATE DATABASE database_name;
```

#6 Create a new table

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    ...
);
```

#7 Insert data into a table

```
INSERT INTO table_name (column1, column2, ...)
VALUES (value1, value2, ...);
```

#8 Update data in a table

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

#9 Delete data from a table

```
DELETE FROM table_name
WHERE condition;
```

#10 Select all records from a table

```
SELECT * FROM table_name;
```

#11 Select specific columns from a table

```
SELECT column1, column2, ...
FROM table_name;
```

#12 Select distinct values from a column

```
SELECT DISTINCT column_name
FROM table_name;
```

#13 Filter rows using WHERE clause

```
SELECT * FROM table_name
WHERE condition;
```

#14 Sort records in ascending order

```
SELECT * FROM table_name
ORDER BY column_name ASC;
```

#15 Sort records in descending order

```
SELECT * FROM table_name
ORDER BY column_name DESC;
```

#16 Limit the number of records returned

```
SELECT * FROM table_name
LIMIT count;
```

#17 Join two tables

```
SELECT *
FROM table1
JOIN table2 ON table1.column_name = table2.column_name;
```

#18 Use the COUNT() function to count rows

```
SELECT COUNT(column_name)
FROM table_name;
```

#19 Use the SUM() function to calculate the sum

```
SELECT SUM(column_name)
FROM table_name;
```

#20 Use the AVG() function to calculate the average

```
SELECT AVG(column_name)
FROM table_name;
```

#21 Use the MIN() function to find the minimum value

```
SELECT MIN(column_name)
FROM table_name;
```

#22 Use the MAX() function to find the maximum value

```
SELECT MAX(column_name)
FROM table_name;
```

#23 Group records by a column

```
SELECT column_name, COUNT(*)
FROM table_name
GROUP BY column_name;
```

#24 Filter groups using HAVING clause

```
SELECT column_name, COUNT(*)
FROM table_name
GROUP BY column_name
HAVING condition;
```

#25 Use the CONCAT() function to concatenate strings

```
SELECT CONCAT(column1, column2)
FROM table_name;
```

#26 Use the IN operator to match any value in a list

```
SELECT *
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

#27 Use the LIKE operator for pattern matching

```
SELECT *
FROM table_name
WHERE column_name LIKE 'pattern';
```

#28 Use the BETWEEN operator to select a range of values

```
SELECT *
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

#29 Use the IS NULL operator to check for null values

```
SELECT *
FROM table_name
WHERE column_name IS NULL;
```

#30 Use the AS keyword to give a column an alias

```
SELECT column_name AS alias_name
FROM table_name;
```

#31 Use the DISTINCT keyword with COUNT() to count distinct values

```
SELECT COUNT(DISTINCT column_name)
FROM table_name;
```

#32 Use the CASE statement for conditional logic

```
SELECT column_name,
    CASE
        WHEN condition1 THEN value1
        WHEN condition2 THEN value2
        ELSE value3
    END
FROM table_name;
```

#33 Use the UNION operator to combine the result of multiple SELECT statements

```
SELECT column1, column2
FROM table1
UNION
SELECT column1, column2
FROM table2;
```

#34 Use the ORDER BY clause with multiple columns

```
SELECT *
FROM table_name
ORDER BY column1, column2;
```

#35 Use the DISTINCT keyword to eliminate duplicate rows

```
SELECT DISTINCT column_name
FROM table_name;
```

#36 Use the GROUP_CONCAT() function to concatenate values into a string

```
SELECT column_name, GROUP_CONCAT(other_column)
FROM table_name
GROUP BY column_name;
```

#37 Use the COALESCE() function to replace NULL values

```
SELECT COALESCE(column_name, 'default_value')
FROM table_name;
```

#38 Use the NOW() function to get the current date and time

```
SELECT NOW();
```

#39 Use the DATE() function to extract the date from a datetime value

```
SELECT DATE(datetime_column)
FROM table_name;
```

#40 Use the SUM() function with GROUP BY to calculate totals per group

```
SELECT category, SUM(quantity)
FROM table_name
GROUP BY category;
```

#41 Use the ORDER BY clause with LIMIT to get the top N records

```
SELECT *
FROM table_name
ORDER BY column_name DESC
LIMIT N;
```

#42 Use the IFNULL() function to replace NULL values with a specific value

```
SELECT column_name, IFNULL(other_column, 'replacement')
FROM table_name;
```

#43 Use the RAND() function to retrieve random rows

```
SELECT column_name
FROM table_name
ORDER BY RAND()
LIMIT N;
```

#44 Use the CONCAT_WS() function to concatenate strings with a separator

```
SELECT CONCAT_WS('-', column1, column2)
FROM table_name;
```

#45 Use the LEFT JOIN to retrieve records from the left table and matching records from the right table

```
SELECT *
FROM table1
LEFT JOIN table2 ON table1.column_name = table2.column_name;
```

#46 Use the RIGHT JOIN to retrieve records from the right table and matching records from the left table

```
SELECT *
FROM table1
RIGHT JOIN table2 ON table1.column_name = table2.column_name;
```

#47 Use the CROSS JOIN to retrieve the Cartesian product of two tables

```
SELECT *
FROM table1
CROSS JOIN table2;
```

#48 Use the EXISTS operator to check for the existence of rows in a subquery

```
SELECT *
FROM table_name
WHERE EXISTS (SELECT * FROM other_table WHERE condition);
```

#49 Use the TRUNCATE TABLE statement to remove all rows from a table

```
TRUNCATE TABLE table_name;
```

#50 Use the DROP TABLE statement to delete a table

```
DROP TABLE table_name;
```