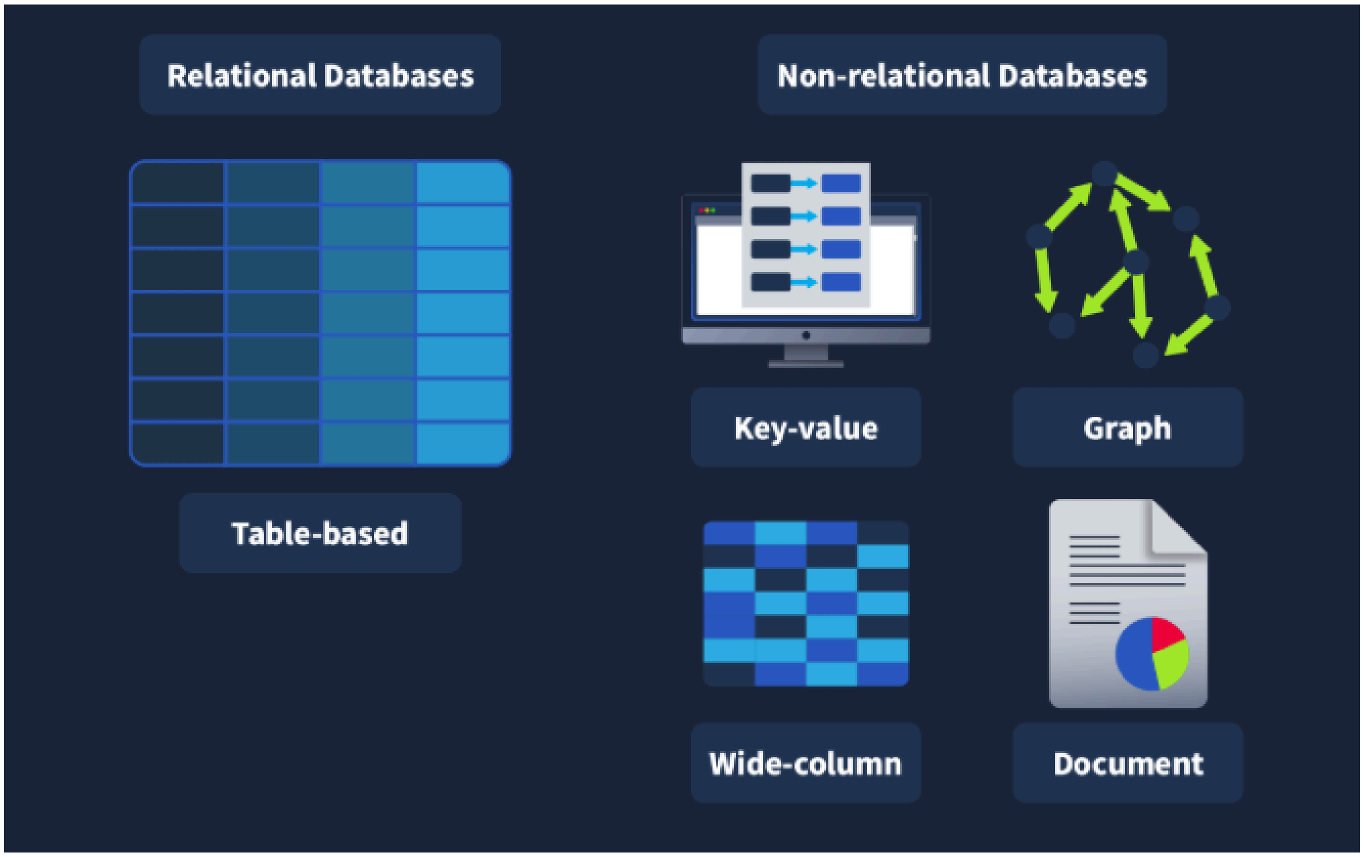
SQL Fundamentals

Databases 101

databases are used extensively and can contain many different things. It's not just massive-scale businesses that use databases. Smaller-scale businesses, when setting up, will almost certainly have to configure a database to store their data

Different Types of Databases

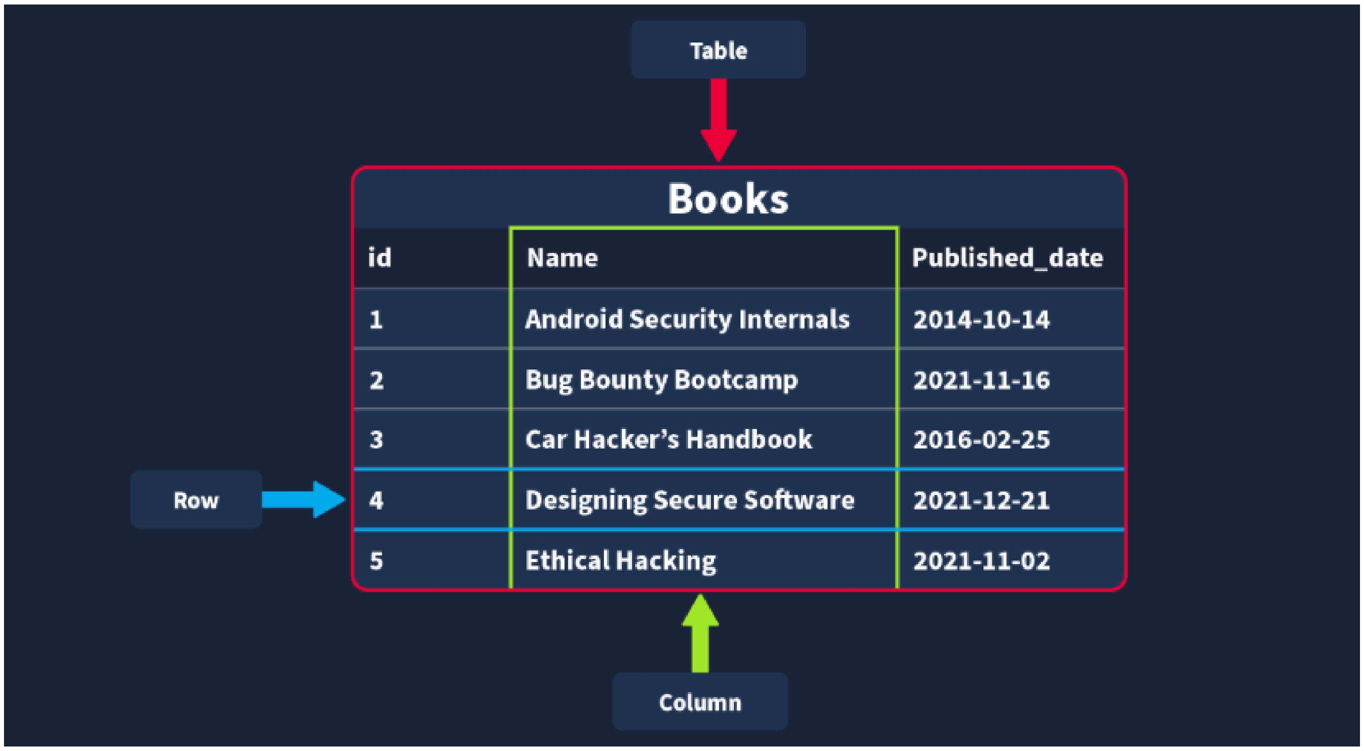


relational databases (aka SQL) vs **non-relational databases** (aka NoSQL).

Relational databases: Store structured data, meaning the data inserted into this database follows a structure. For example, the data collected on a user consists of first_name, last_name, email_address, username and password. When a new user joins, an entry is made in the database following this structure.

Non-relational databases: Instead of storing data the above way, store data in a non-tabular format. For example, if documents are being scanned, which can contain varying types and quantities of data, and are stored in a database that calls for a non-tabular format.

Tables, Rows and Columns

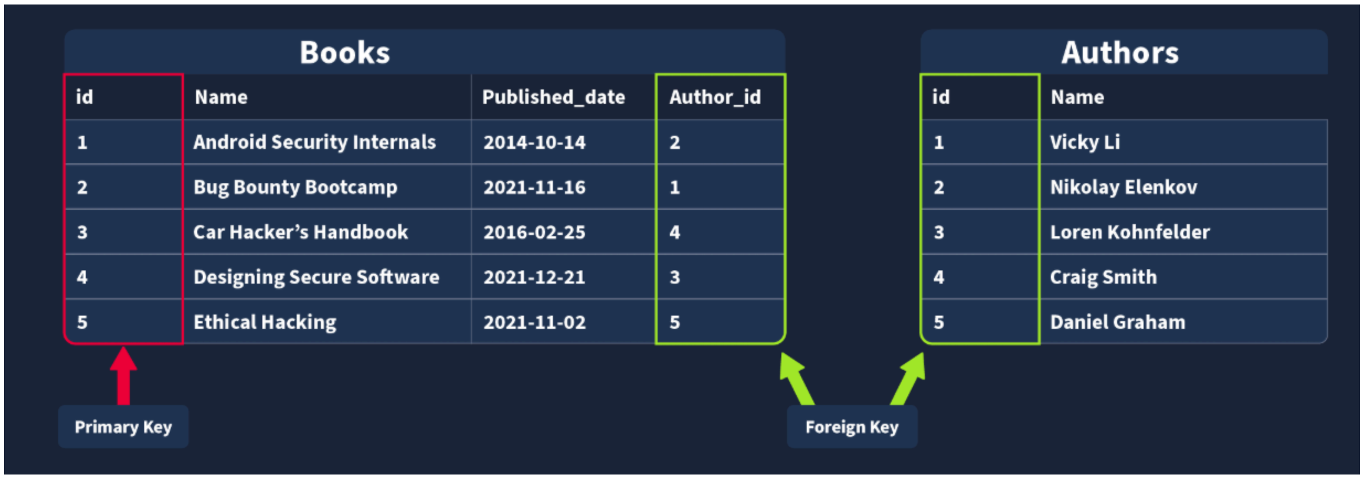


Books

id	Name	Published_date
1	Android Security Internals	2014-10-14
2	Bug Bounty Bootcamp	2021-11-16
3	Car Hacker's Handbook	2016-02-25
4	Designing Secure Software	2021-12-21
5	Ethical Hacking	2021-11-02

Primary and Foreign Keys

If we wanted to query for a book in our story but also have the author of that book returned, our data would need to be related somehow; we do this with keys. There are two types of **keys**:



Books

id	Name	Published_date	Author_id
1	Android Security Internals	2014-10-14	2
2	Bug Bounty Bootcamp	2021-11-16	1
3	Car Hacker's Handbook	2016-02-25	4
4	Designing Secure Software	2021-12-21	3
5	Ethical Hacking	2021-11-02	5

Authors

id	Name
1	Vicky Li
2	Nikolay Elenkov
3	Loren Kohnfelder
4	Craig Smith
5	Daniel Graham

Primary Keys: A primary key is used to ensure that the data collected in a certain column is unique.

Foreign Keys: A foreign key is a column (or columns) in a table that also exists in another table within the database, and therefore provides a link between the two tables.

SQL

The interaction between the end user and the database can be done using SQL (Structured Query Language). SQL is a programming language that can be used to query, define and manipulate the data stored in a relational database.

The Benefits of SQL and Relational Databases

SQL is almost as ubiquitous as databases themselves, and for good reason. Here are some of the benefits that come with learning and using to use SQL:

- **It's *fast***: Relational databases (aka those that SQL is used for) can return massive batches of data almost instantaneously due to how little storage space is used and high processing speeds.
- **Easy to Learn:** Unlike many programming languages, SQL is written in plain English, making it much easier to pick up. The highly readable nature of the language means users can concentrate on learning the functions and syntax.
- **Reliable:** As mentioned before, relational databases can guarantee a level of accuracy when it comes to data by defining a strict structure into which data sets must fall in order to be inserted.
- **Flexible:** SQL provides all kinds of capabilities when it comes to querying a database; this allows users to perform vast data analysis tasks very efficiently.

Database and Table Statements

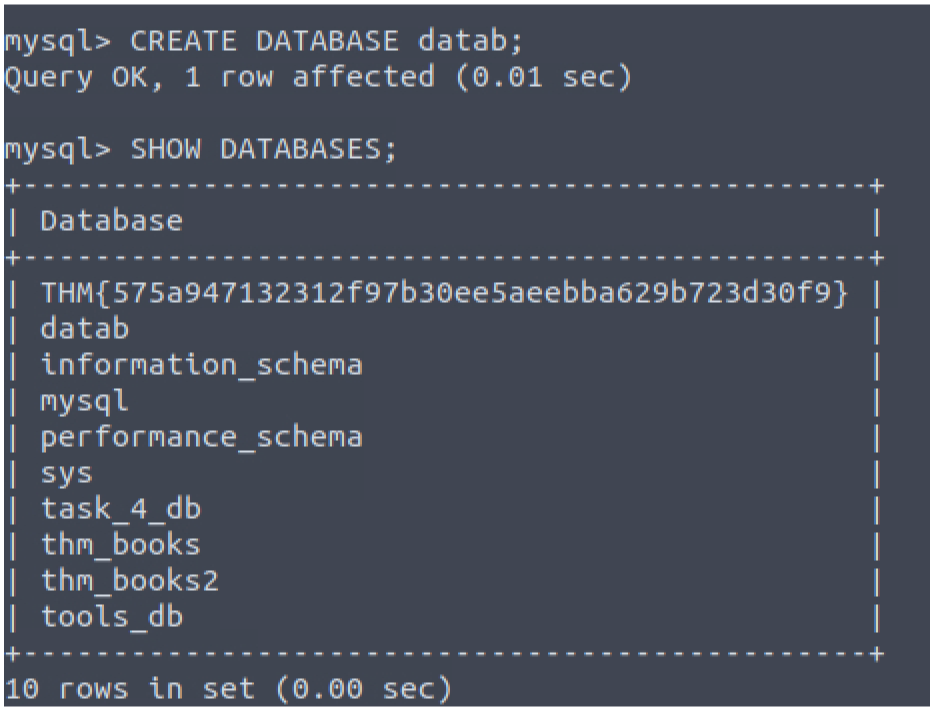
Database Statements

CREATE DATABASE

If a new database is needed, the first step you would take is to create it. This can be done in SQL using the `CREATE DATABASE` statement.

SHOW DATABASES

can view it using the `SHOW DATABASES` statement



```
mysql> CREATE DATABASE datab;
Query OK, 1 row affected (0.01 sec)

mysql> SHOW DATABASES;
+-------------------------------------------------+
| Database                                        |
+-------------------------------------------------+
| THM{575a947132312f97b30ee5aeebba629b723d30f9}   |
| datab                                           |
| information_schema                              |
| mysql                                           |
| performance_schema                              |
| sys                                             |
| task_4_db                                       |
| thm_books                                       |
| thm_books2                                      |
| tools_db                                        |
+-------------------------------------------------+
10 rows in set (0.00 sec)
```

USE DATABASE
to set the database we have just created as the active database, we would run the `USE` statement

DROP DATABASE
Once a database is no longer needed (maybe it was created for test purposes, or is no longer required), it can be removed using the `DROP` statement

Table Statements

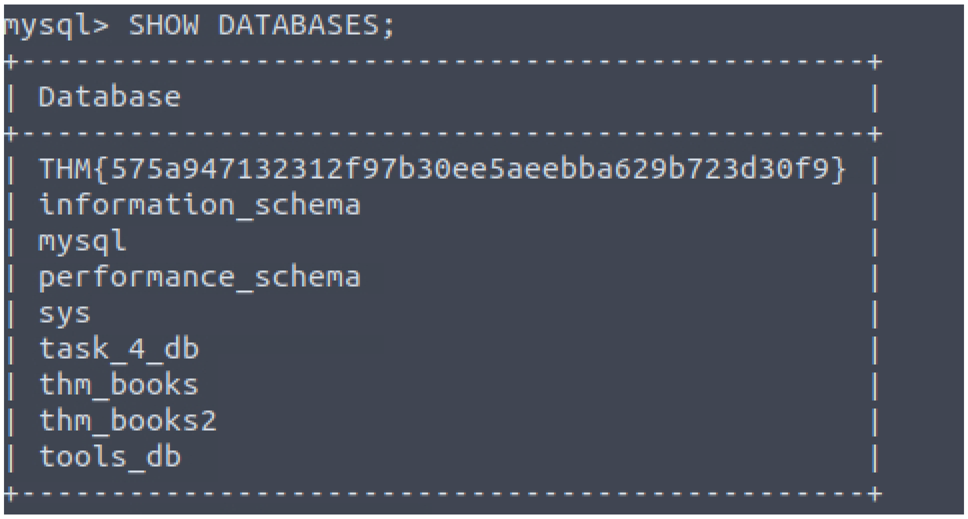
CREATE TABLE
creating tables also uses a `CREATE` statement. but used as `CREATE TABLE`

SHOW TABLES
we can list databases using a SHOW statement, we can also list the tables in our currently active database

DESCRIBE
to know what columns are contained within a table (and their data type), we can describe them using the `DESCRIBE` command

ALTER
there may come a time when your need for the dataset changes, and you need to alter the table. This can be done using the `ALTER` statement.

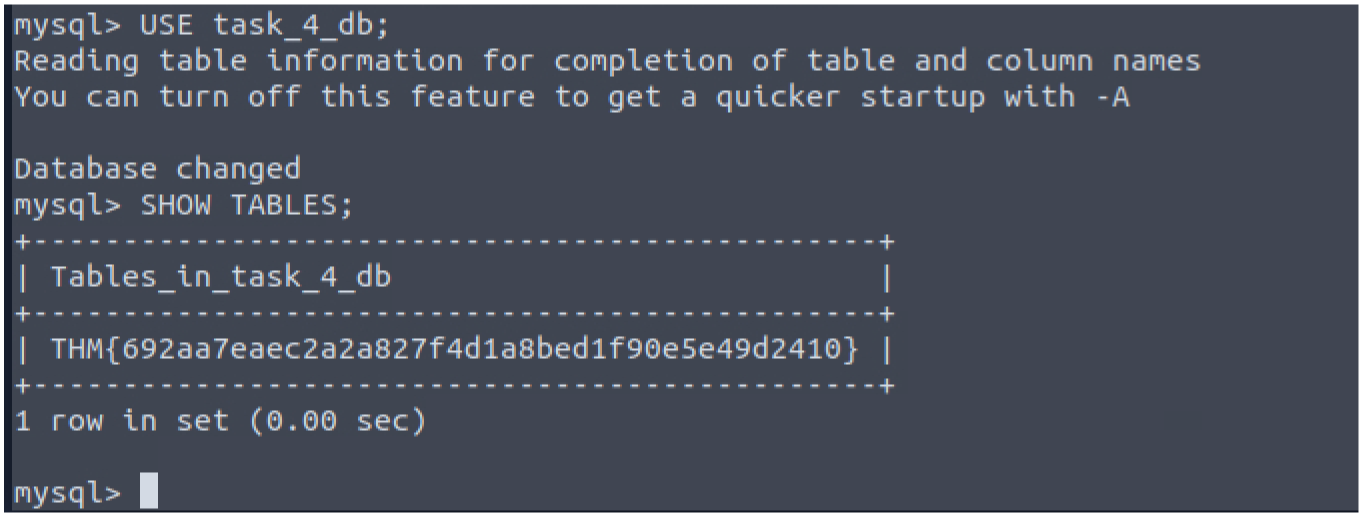
DROP
Similar to removing a database, you can also remove tables using the `DROP` statement `DROP TABLE`.



```
mysql> SHOW DATABASES;
+-----------------------------------------------+
| Database                                      |
+-----------------------------------------------+
| THM{575a947132312f97b30ee5aeebba629b723d30f9} |
| information_schema                            |
| mysql                                         |
| performance_schema                            |
| sys                                           |
| task_4_db                                     |
| thm_books                                     |
| thm_books2                                    |
| tools_db                                      |
+-----------------------------------------------+
```

I did show databases and it gave me a flag

I then used the USE command on task_4_db then SHOW TABLES command and got another flag



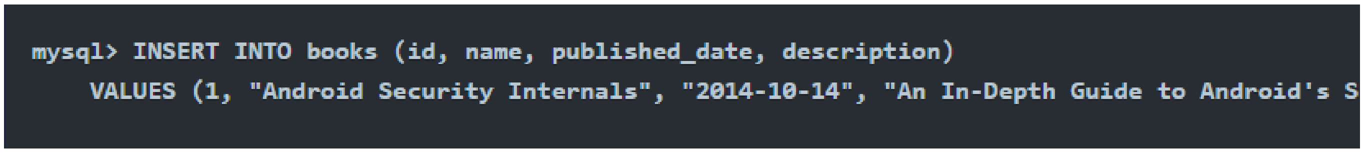
```
mysql> USE task_4_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+-----------------------------------------------+
| Tables_in_task_4_db                           |
+-----------------------------------------------+
| THM{692aa7eaec2a2a827f4d1a8bed1f90e5e49d2410} |
+-----------------------------------------------+
1 row in set (0.00 sec)

mysql> 
```

CRUD Operations

CRUD stands for **C**reate, **R**ead, **U**pdate, and **D**elete, which are considered the basic operations in any system that manages data.

Create Operation (INSERT)

The **Create** operation will create new records in a table. In MySQL, this can be achieved by using the statement `INSERT INTO



```
mysql> INSERT INTO books (id, name, published_date, description)
    VALUES (1, "Android Security Internals", "2014-10-14", "An In-Depth Guide to Android's S
```

Read Operation (SELECT)

The **Read** operation, as the name suggests, is used to read or retrieve information from a table. We can fetch a column or all columns from a table with the `SELECT` statement, as shown in the next example.

```
mysql> SELECT * FROM books;
+----+----------------------------+----------------+-----------------------------------------
| id | name                       | published_date | description
+----+----------------------------+----------------+-----------------------------------------
|  1 | Android Security Internals | 2014-10-14     | An In-Depth Guide to Android's Securit
+----+----------------------------+----------------+-----------------------------------------
```

Update Operation (UPDATE)

The **Update** operation modifies an existing record within a table, and the same statement, `UPDATE`, can be used for this

```
mysql> UPDATE books
    SET description = "An In-Depth Guide to Android's Security Architecture."
    WHERE id = 1;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Delete Operation (DELETE)

The **delete** operation removes records from a table. We can achieve this with the `DELETE` statement.

```
mysql> DELETE FROM books WHERE id = 1;
```

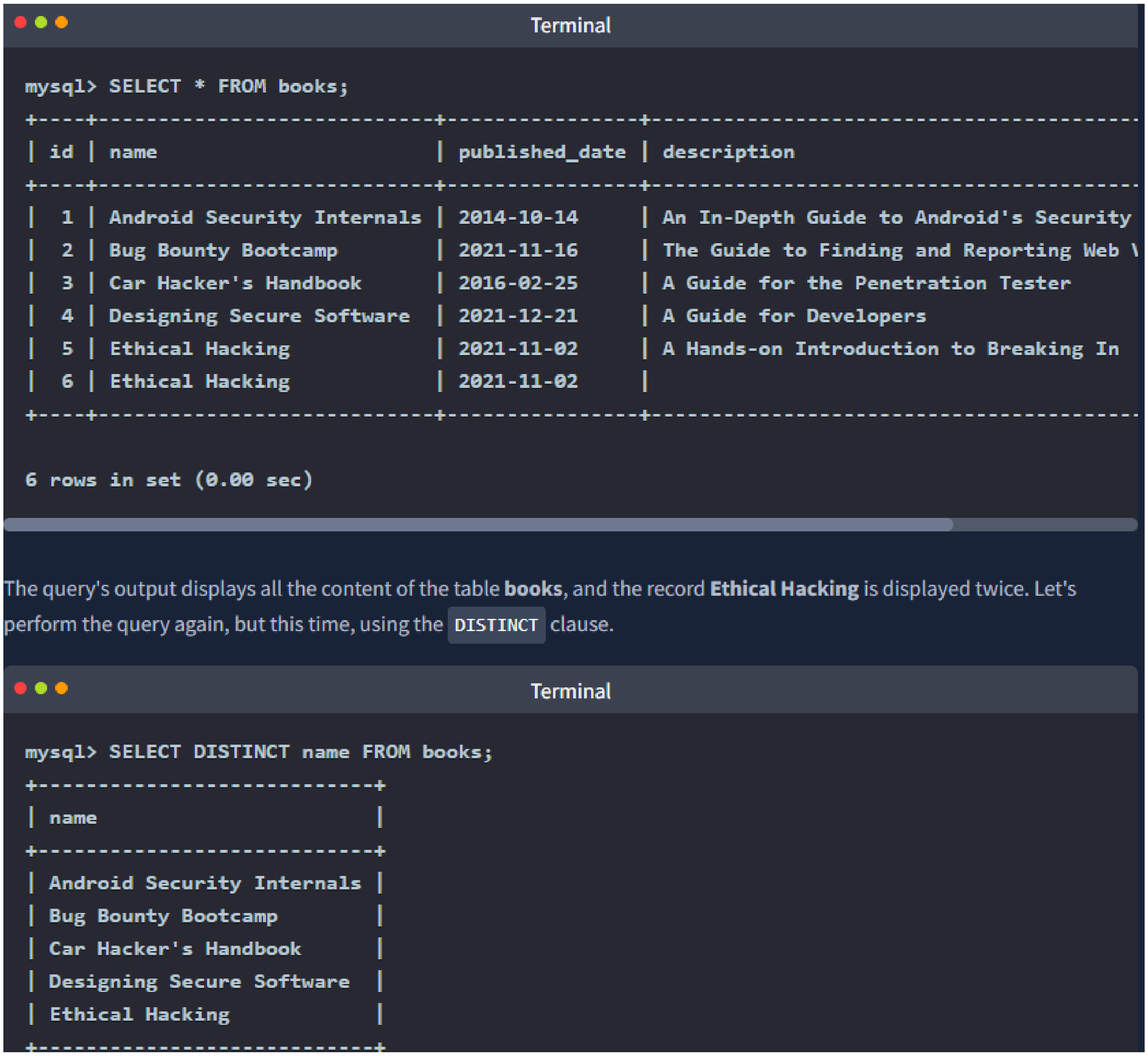
Clauses

A clause is a part of a statement that specifies the criteria of the data being manipulated, usually by an initial statement. Clauses can help us define the type of data and how it should be retrieved or sorted.

on other clauses: `DISTINCT`, `GROUP BY`, `ORDER BY`, and `HAVING`.

DISTINCT Clause

The `DISTINCT` clause is used to avoid duplicate records when doing a query, returning only unique values



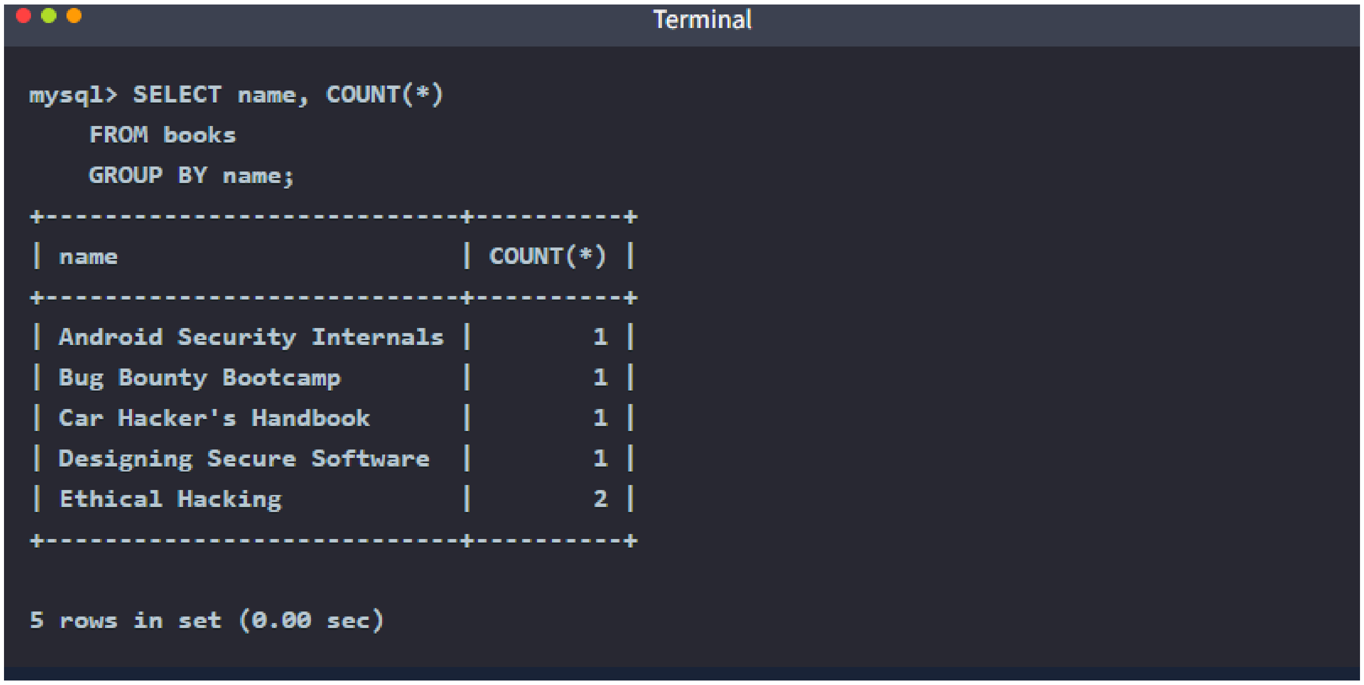
```
mysql> SELECT * FROM books;
+----+---------------------------+----------------+-----------------------------------------
| id | name                      | published_date | description
+----+---------------------------+----------------+-----------------------------------------
|  1 | Android Security Internals | 2014-10-14     | An In-Depth Guide to Android's Security
|  2 | Bug Bounty Bootcamp       | 2021-11-16     | The Guide to Finding and Reporting Web
|  3 | Car Hacker's Handbook     | 2016-02-25     | A Guide for the Penetration Tester
|  4 | Designing Secure Software | 2021-12-21     | A Guide for Developers
|  5 | Ethical Hacking           | 2021-11-02     | A Hands-on Introduction to Breaking In
|  6 | Ethical Hacking           | 2021-11-02     |
+----+---------------------------+----------------+-----------------------------------------

6 rows in set (0.00 sec)
```

The query's output displays all the content of the table **books**, and the record **Ethical Hacking** is displayed twice. Let's perform the query again, but this time, using the `DISTINCT` clause.

```
mysql> SELECT DISTINCT name FROM books;
+----------------------------+
| name                       |
+----------------------------+
| Android Security Internals |
| Bug Bounty Bootcamp        |
| Car Hacker's Handbook      |
| Designing Secure Software  |
| Ethical Hacking            |
+----------------------------+
```

GROUP BY Clause

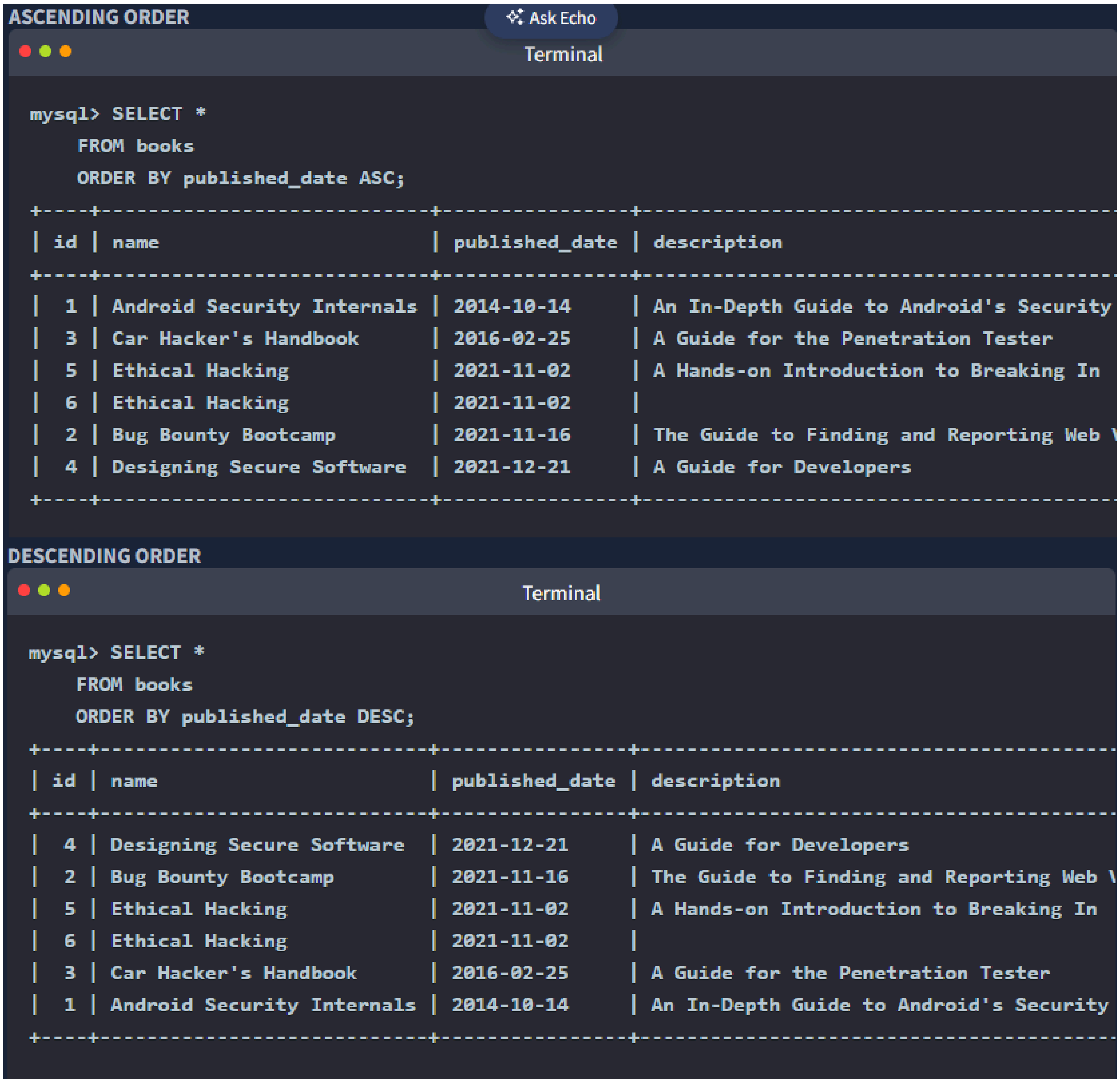
The `GROUP BY` clause aggregates data from multiple records and **groups** the query results in columns.



```
mysql> SELECT name, COUNT(*)
    FROM books
    GROUP BY name;
+---------------------------+----------+
| name                      | COUNT(*) |
+---------------------------+----------+
| Android Security Internals |        1 |
| Bug Bounty Bootcamp       |        1 |
| Car Hacker's Handbook     |        1 |
| Designing Secure Software |        1 |
| Ethical Hacking           |        2 |
+---------------------------+----------+

5 rows in set (0.00 sec)
```

ORDER BY Clause

The `ORDER BY` clause can be used to sort the records returned by a query in ascending or descending order



ASCENDING ORDER

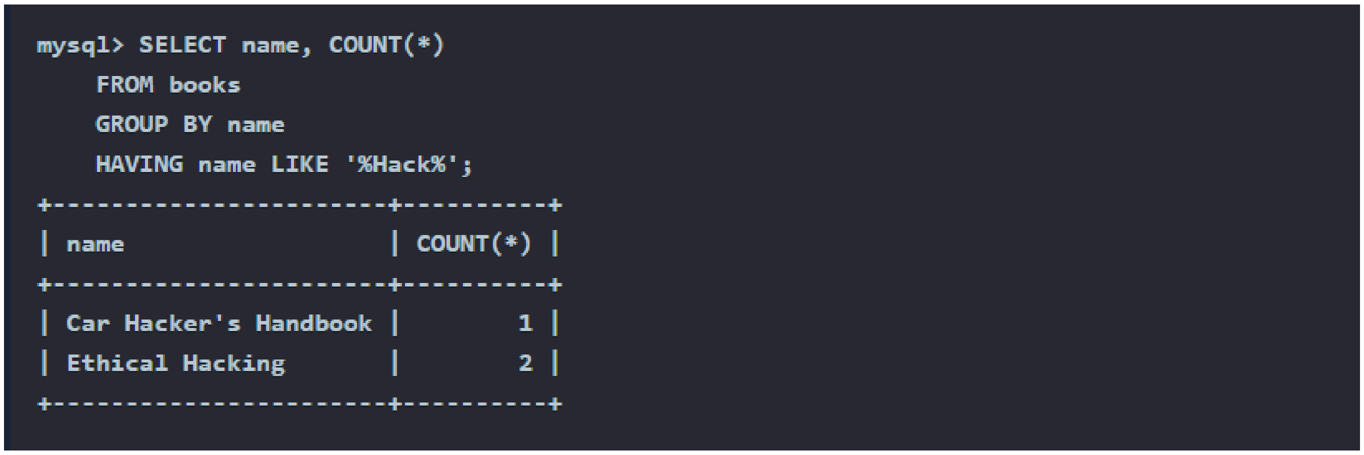
```
mysql> SELECT *
    FROM books
    ORDER BY published_date ASC;
+----+---------------------------+----------------+-----------------------------------------
| id | name                      | published_date | description
+----+---------------------------+----------------+-----------------------------------------
|  1 | Android Security Internals | 2014-10-14     | An In-Depth Guide to Android's Security
|  3 | Car Hacker's Handbook     | 2016-02-25     | A Guide for the Penetration Tester
|  5 | Ethical Hacking           | 2021-11-02     | A Hands-on Introduction to Breaking In
|  6 | Ethical Hacking           | 2021-11-02     |
|  2 | Bug Bounty Bootcamp       | 2021-11-16     | The Guide to Finding and Reporting Web \
|  4 | Designing Secure Software | 2021-12-21     | A Guide for Developers
+----+---------------------------+----------------+-----------------------------------------
```

DESCENDING ORDER

```
mysql> SELECT *
    FROM books
    ORDER BY published_date DESC;
+----+---------------------------+----------------+-----------------------------------------
| id | name                      | published_date | description
+----+---------------------------+----------------+-----------------------------------------
|  4 | Designing Secure Software | 2021-12-21     | A Guide for Developers
|  2 | Bug Bounty Bootcamp       | 2021-11-16     | The Guide to Finding and Reporting Web \
|  5 | Ethical Hacking           | 2021-11-02     | A Hands-on Introduction to Breaking In
|  6 | Ethical Hacking           | 2021-11-02     |
|  3 | Car Hacker's Handbook     | 2016-02-25     | A Guide for the Penetration Tester
|  1 | Android Security Internals | 2014-10-14     | An In-Depth Guide to Android's Security
+----+---------------------------+----------------+-----------------------------------------
```

HAVING Clause

The `HAVING` clause is used with other clauses to filter groups or results of records based on a condition



```
mysql> SELECT name, COUNT(*)
    FROM books
    GROUP BY name
    HAVING name LIKE '%Hack%';
+----------------------+----------+
| name                 | COUNT(*) |
+----------------------+----------+
| Car Hacker's Handbook |        1 |
| Ethical Hacking      |        2 |
+----------------------+----------+
```

Operators

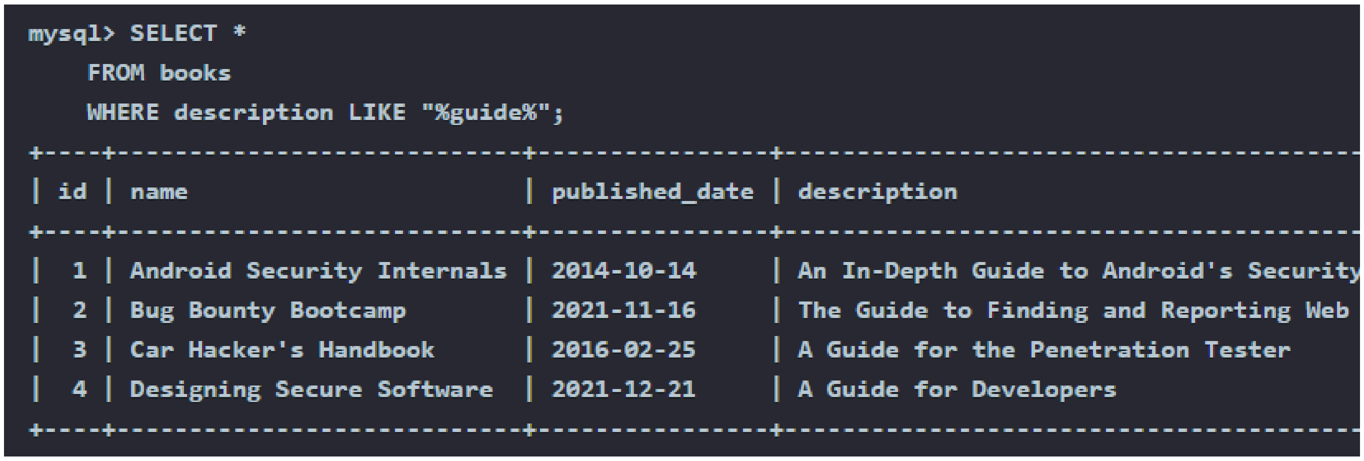
operators are our way to filter and manipulate data effectively

Logical Operators

These operators test the truth of a condition and return a boolean value of `TRUE` or `FALSE`

LIKE Operator

The `LIKE` operator is commonly used in conjunction with clauses like `WHERE` in order to filter for specific patterns within a column



```
mysql> SELECT *
    FROM books
    WHERE description LIKE "%guide%";
+----+----------------------------+----------------+--------------------------------------
| id | name                       | published_date | description
+----+----------------------------+----------------+--------------------------------------
|  1 | Android Security Internals | 2014-10-14     | An In-Depth Guide to Android's Security
|  2 | Bug Bounty Bootcamp        | 2021-11-16     | The Guide to Finding and Reporting Web
|  3 | Car Hacker's Handbook      | 2016-02-25     | A Guide for the Penetration Tester
|  4 | Designing Secure Software  | 2021-12-21     | A Guide for Developers
+----+----------------------------+----------------+--------------------------------------
```

AND Operator

The `AND` operator uses multiple conditions within a query and returns `TRUE` if all of them are true

OR Operator

The `OR` operator combines multiple conditions within queries and returns `TRUE` if at least one of these conditions is true

NOT Operator

The `NOT` operator reverses the value of a boolean operator, allowing us to exclude a specific condition.

BETWEEN Operator

The `BETWEEN` operator allows us to test if a value exists within a defined **range**

Comparison Operators

The comparison operators are used to compare values and check if they meet specified criteria.

Equal To Operator

The `=` (Equal) operator compares two expressions and determines if they are equal, or it can check if a value matches another one in a specific column.

Not Equal To Operator

The `!=` (not equal) operator compares expressions and tests if they are not equal; it also checks if a value differs from the one within a column.

Less Than Operator

Less Than Operator

The `<` (less than) operator compares if the expression with a given value is lesser than the provided one.

Greater Than Operator

The `>` (greater than) operator compares if the expression with a given value is greater than the provided one.

Less Than or Equal To and Greater Than or Equal To Operators

The `<=` (Less than or equal) operator compares if the expression with a given value is less than or equal to the provided one. On the other hand, The `>=` (Greater than or Equal) operator compares if the expression with a given value is greater than or equal to the provided one

Functions

String Functions

Strings functions perform operations on a string, returning a value associated with it.

CONCAT() Function

This function is used to add two or more strings together. It is useful to combine text from different columns.

GROUP_CONCAT() Function

This function can help us to concatenate data from multiple rows into one field

SUBSTRING() Function

This function will retrieve a substring from a string within a query, starting at a determined position

LENGTH() Function

This function returns the number of characters in a string

Aggregate Functions

These functions aggregate the value of multiple rows within one specified criteria in the query; It can combine multiple values into one result.

COUNT() Function

This function returns the number of records within an expression, as the example below shows

SUM() Function

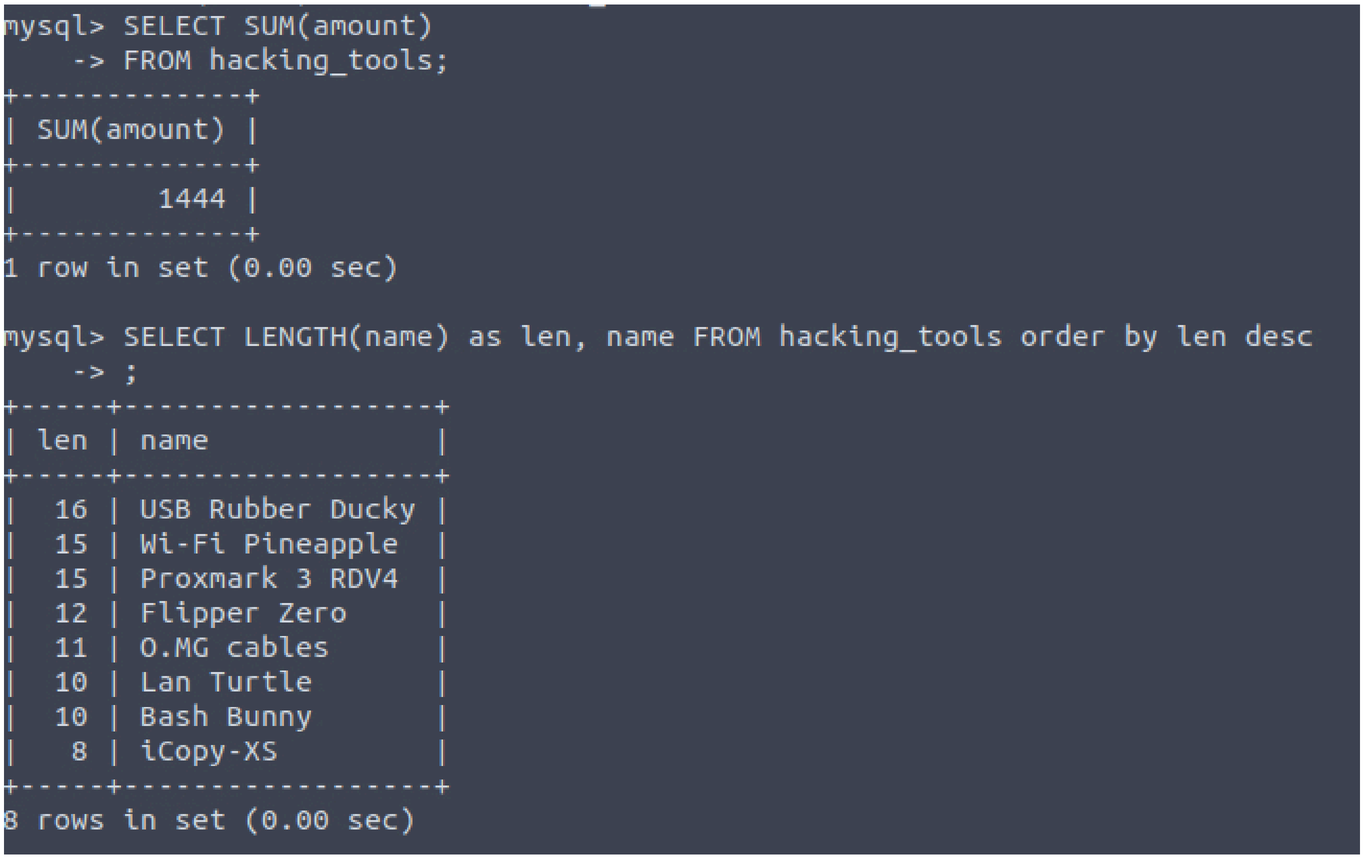
This function sums all values (not NULL) of a determined column.

MAX() Function

This function calculates the maximum value within a provided column in an expression.

MIN() Function

This function calculates the minimum value within a provided column in an expression.



```
mysql> SELECT SUM(amount)
    -> FROM hacking_tools;
+-------------+
| SUM(amount) |
+-------------+
|        1444 |
+-------------+
1 row in set (0.00 sec)

mysql> SELECT LENGTH(name) as len, name FROM hacking_tools order by len desc
    -> ;
+-----+------------------+
| len | name             |
+-----+------------------+
|  16 | USB Rubber Ducky |
|  15 | Wi-Fi Pineapple  |
|  15 | Proxmark 3 RDV4  |
|  12 | Flipper Zero     |
|  11 | O.MG cables      |
|  10 | Lan Turtle       |
|  10 | Bash Bunny       |
|   8 | iCopy-XS         |
+-----+------------------+
8 rows in set (0.00 sec)
```