**MySQL, MongoDB**

**MySQL**

MySQL is used to store data in tables that map to objects. Each table has a schema defining what columns each row of the table will have. Developers can reliably store and retrieve many data types, including text, numbers, dates, times, and even JSON.

Interacting with a MySQL database is done with SQL (Structured Query Language). SQL is not a fully-fledged programming language. But as a querying language, it offers a straightforward syntax to manage your database through:

* Creating, updating, and deleting tables.
* Indexing tables.
* Retrieving, inserting, updating, and deleting data in tables.
* Joining data across multiple tables.
* Running mathematical functions on queried data.
* Partitioning data.

[**Benefits and advantages of MySQL**](https://planetscale.com/learn/articles/what-is-mysql#benefits-and-advantages-of-mysql)

* There aren’t many hurdles to using MySQL because of how much tooling exists to make it plug-and-play with most programming languages and frameworks. Even beginners not familiar with MySQL can use Object-Relationship Mapping (ORM) interfaces to interact with their database with Python, PHP, Ruby, and many other programming languages. MySQL is also compatible with many popular application frameworks, such as Ruby on Rails, Laravel, Django, Spring Boot, and ASP.NET.
* Cost doesn’t need to be an obstacle either. MySQL is freely licensed. It’s also lightweight enough to be run on a laptop for personal projects.
* Additionally, MySQL’s security features are well-tested. MySQL offers a variety of tools enabling granular control of who has access to what, and numerous plugins exist for auditing and data encryption. With a vast ecosystem of MySQL tooling, there are many options available for tools like GUI-based desktop clients that make it easier to interact with your data.

**Common MySQL commands:**

* SELECT - extracts data from a database
* UPDATE - updates data in a database
* DELETE - deletes data from a database
* INSERT INTO - inserts new data into a database
* CREATE DATABASE - creates a new database
* ALTER DATABASE - modifies a database
* CREATE TABLE - creates a new table
* ALTER TABLE - modifies a table
* DROP TABLE - deletes a table
* CREATE INDEX - creates an index (search key)
* DROP INDEX - deletes an index

SELECT:

The SELECT statement is used to select data from a database. The data returned is stored in a result table, called the result-set

Syntax:

* SELECT \* FROM table\_name;
* SELECT column1, column2, ...  
  FROM table\_name;

WHERE:

The WHERE clause is used to filter records. It is used to extract only those records that fulfil a specified condition.

Syntax:

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition;

AND, OR, NOT:

The WHERE clause can be combined with AND, OR, and NOT operators.

The AND and OR operators are used to filter records based on more than one condition:

* The AND operator displays a record if all the conditions separated by AND are TRUE.
* The OR operator displays a record if any of the conditions separated by OR is TRUE.

The NOT operator displays a record if the condition(s) is NOT TRUE.

Syntax:

* SELECT column1, column2, ...  
  FROM table\_name  
  WHERE condition1 AND condition2 AND condition3 ...;
* SELECT column1, column2, ...  
  FROM table\_name  
  WHERE condition1 OR condition2 OR condition3 ...;
* SELECT column1, column2, ...  
  FROM table\_name  
  WHERE NOT condition;

ORDER BY:

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

Syntax:

SELECT column1, column2, ...  
FROM table\_name  
ORDER BY column1, column2, ... ASC|DESC;

INSERT INTO:

It is possible to write the INSERT INTO statement in two ways:

1. Specify both the column names and the values to be inserted:

Syntax:

INSERT INTO table\_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);

2. If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table.

Syntax:

INSERT INTO table\_name  
VALUES (value1, value2, value3, ...);

UPDATE:

The UPDATE statement is used to modify the existing records in a table

Syntax:

UPDATE table\_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;

DELETE:

The DELETE statement is used to delete existing records in a table.

Syntax:

DELETE FROM table\_name WHERE condition;

CREATE DATABASE:

The CREATE DATABASE statement is used to create a new SQL database.

Syntax:

CREATE DATABASE databasename;

DROP DATABASE:

The DROP DATABASE statement is used to drop an existing SQL database.

Syntax:

DROP DATABASE databasename;

CREATE TABLE:

The CREATE TABLE statement is used to create a new table in a database.

Syntax:

CREATE TABLE table\_name (  
   column1 datatype*,*   column2 datatype*,*   column3 datatype,  
   ....  
);

DROP TABLE:

The DROP TABLE statement is used to drop an existing table in a database.

Syntax:

DROP TABLE table\_name;

ALTER TABLE:

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

Syntax:

Add column: ALTER TABLE table\_name  
ADD column\_name datatype;

Drop column: ALTER TABLE table\_name  
DROP COLUMN column\_name;

Modify column: ALTER TABLE table\_name  
MODIFY COLUMN column\_name datatype;

These are few MySQL commands.

**MongoDB**

MongoDB is a document database which is often referred to as a non-relational database. This does not mean that relational data cannot be stored in document databases. It means that relational data is stored differently. A better way to refer to it is as a non-tabular database.

#### 1. How to show the databases present

* show dbs

#### 2. How to create a database

* use [name of database]

##### Example

use users

#### 3. How to check active collection or database

* db

#### 4. How to create a collection

* db.createCollection("[name of collection]")

##### Example

db.createCollection("customer")

#### 5. How to show collections

* show collections

#### 6. How to insert a document into a collection

* db.[name of collection].insert({})

#### 7. How to show all documents in a collection

* db.[collection Name].find()

or

* db.[collection name].find().pretty() for a well indented list.

##### Example

db.customer.find().pretty()

#### 8. How to limit fields when listing all the documents present in a collection

##### Example

db.customer.find({},{id:1}).pretty()

The above will only display \_id the result.

#### 9. How to update a document with $set

* db.[name of collection].update({},{$set: {}})

#### 10. How to rename a field in a document with $rename

##### Example

db.customer.update(

{"name": "sushruth"},

{$rename:

{"gender":"sex"}

}

)

### **11. How to delete or remove a field with $unset**

### **Example**

db.customer.update(

{"name": "sushruth"},

{$unset:

{"sex": 1}

}

)

#### 12. How to remove a document from a collection

* db.[name of collections].remove({})

or

* db.[name of collections].remove({}, {justOne: true}) to remove just one with the unique match.