# Web Programming
# PHP + SQL

Presenter
Prof. Henry Louis

# Introduction to Database

**What is DataBase:**
A database is an organized collection of structured information, or data, typically stored electronically in a computer system.

**What is SQL:**
SQL is a programming language used by nearly all relational databases to query, manipulate, and define data, and to provide access control.

# MySQL vs MYSQLi

**MySQL:**

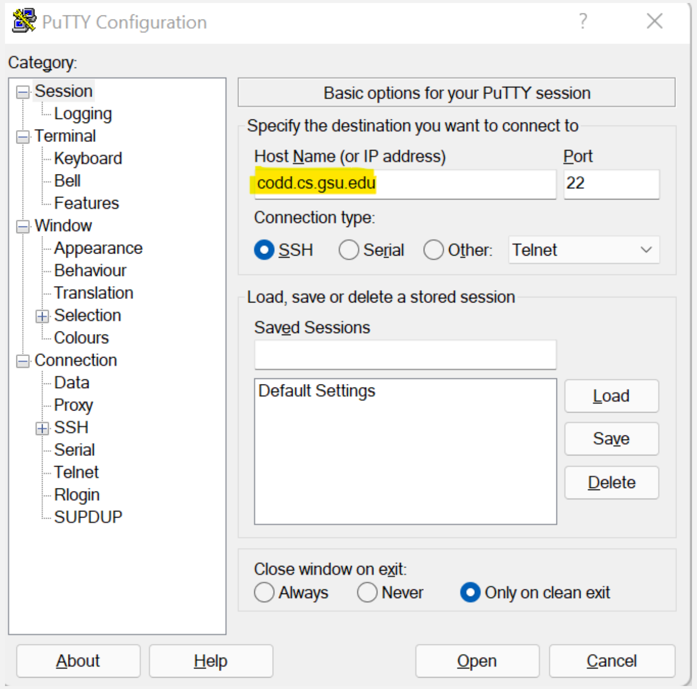
MySQL is a relational database management system based on SQL.

**MySQLi**:

- MySQLi is the Improved driver. The I stands for any one of improved, interface, ingenious, incompatible or incomplete
- To correct the issues of the MySQL extension, a new extension has been created for PHP5. it is called MySQLi.
- It supports all the latest features in MySQL server 4.1 or higher.

# Connecting to MySQLi

- Establish an SSH/PuTTY connection to your server
- Launching SSH/PuTTY - and login with your credentials



PuTTY Configuration

Category:

- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data
  - Proxy
  - SSH
  - Serial
  - Telnet
  - Rlogin
  - SUPDUP

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): codd.cs.gsu.edu

Port: 22

Connection type: ● SSH ○ Serial ○ Other: Telnet

Load, save or delete a stored session

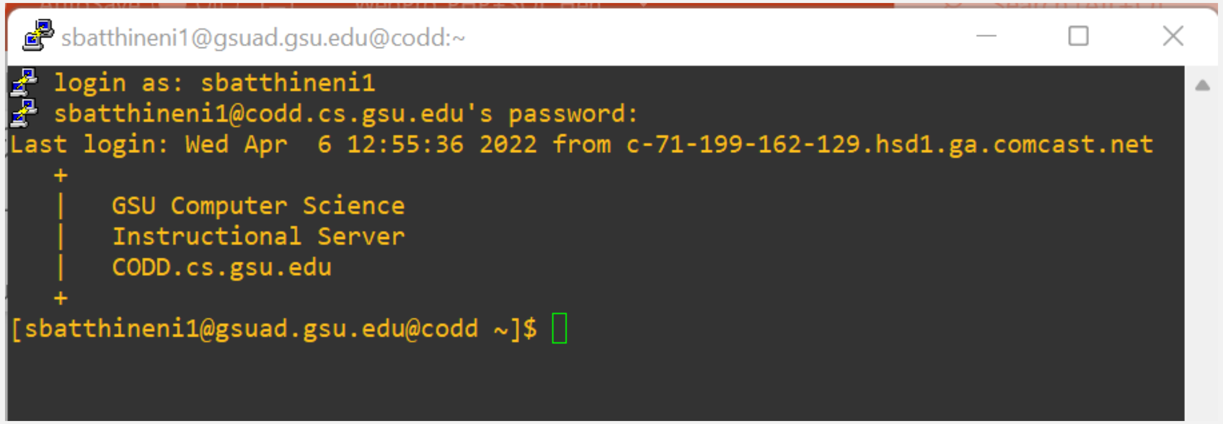
Saved Sessions

Default Settings

Load Save Delete

Close window on exit: ○ Always ○ Never ● Only on clean exit

About Help Open Cancel



sbatthineni1@gsuad.gsu.edu@codd:~

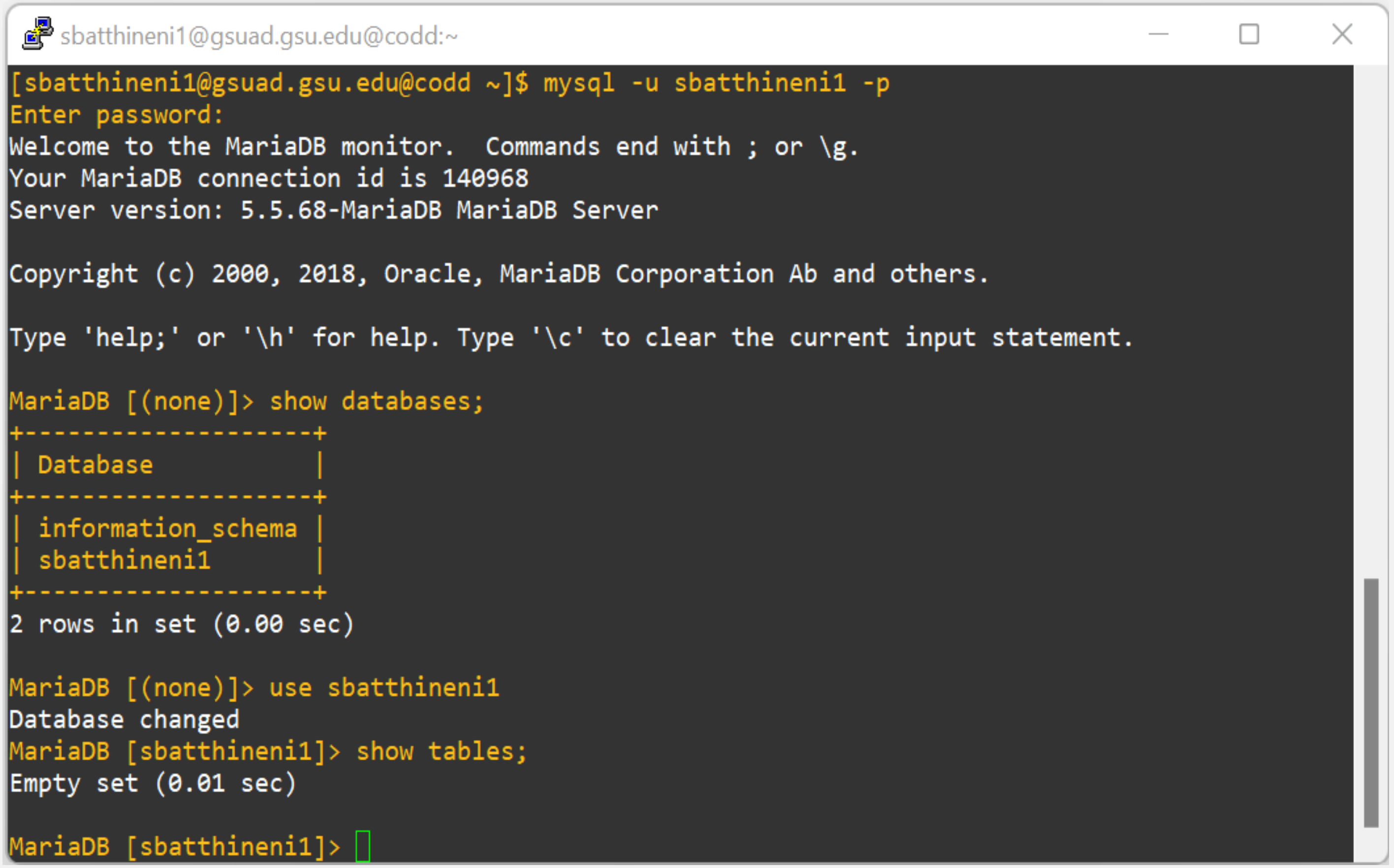
```
login as: sbatthineni1
sbatthineni1@codd.cs.gsu.edu's password:
Last login: Wed Apr  6 12:55:36 2022 from c-71-199-162-129.hsd1.ga.comcast.net
   +
   |   GSU Computer Science
   |   Instructional Server
   |   CODD.cs.gsu.edu
   +
[sbatthineni1@gsuad.gsu.edu@codd ~]$ 
```

We're doing it #thestateway

# Connecting to MySQLi

- Login to MySQL by typing the following commands
  - $ mysql -u username -p
  - Enter password: username

- The database is already created for just need to do is to create tables
  - mysql> SHOW DATABASES;
  - mysql> USE mydb;

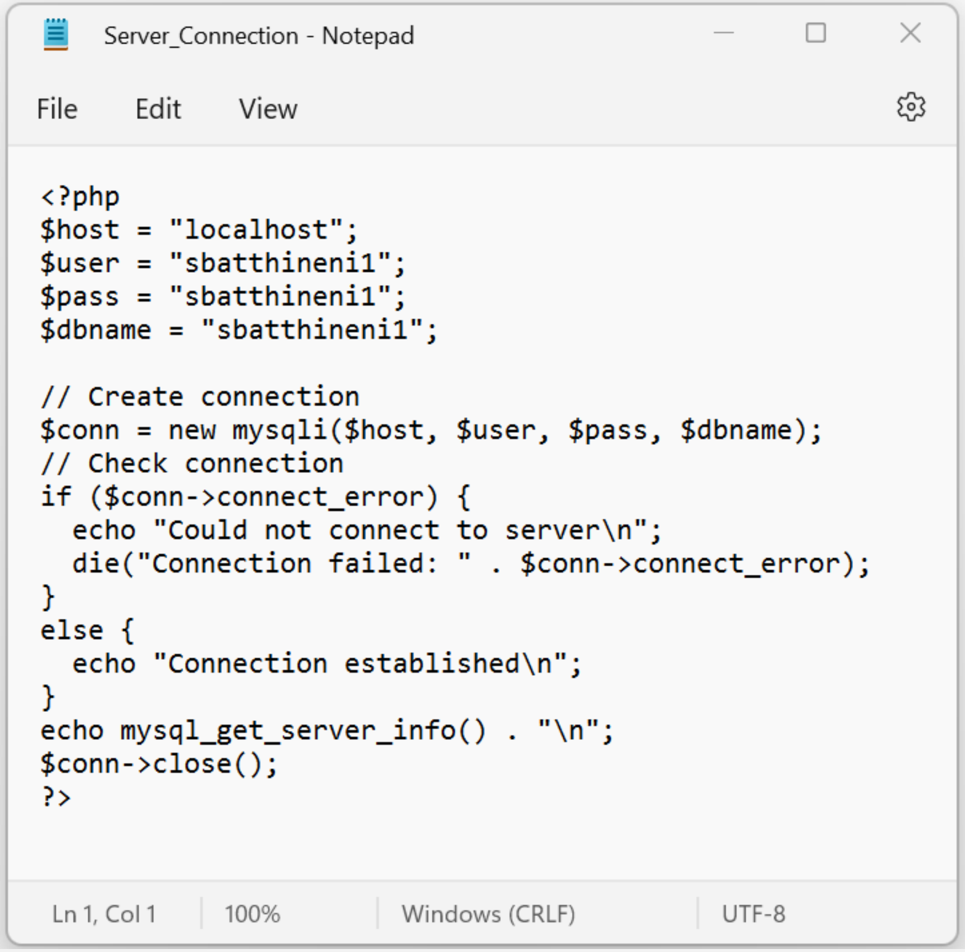


```
[sbatthineni1@gsuad.gsu.edu@codd ~]$ mysql -u sbatthineni1 -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 140968
Server version: 5.5.68-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| sbatthineni1       |
+--------------------+
2 rows in set (0.00 sec)

MariaDB [(none)]> use sbatthineni1
Database changed
MariaDB [sbatthineni1]> show tables;
Empty set (0.01 sec)

MariaDB [sbatthineni1]>
```

**We're doing it #thestateway**

# First connection through PHP script

- The following script is a simple PHP script. If this small script runs OK, we have everything needed to be installed.



```
<?php
$host = "localhost";
$user = "sbatthineni1";
$pass = "sbatthineni1";
$dbname = "sbatthineni1";

// Create connection
$conn = new mysqli($host, $user, $pass, $dbname);
// Check connection
if ($conn->connect_error) {
  echo "Could not connect to server\n";
  die("Connection failed: " . $conn->connect_error);
}
else {
  echo "Connection established\n";
}
echo mysql_get_server_info() . "\n";
$conn->close();
?>
```

# First connection through PHP script

- Variables hostname, username, password & Database name are needed for connecting to the MySQL database
- **Mysqli()** is to connect to the database and returns a boolean value.
- If **$Conn** variable has boolean false, the connection to the
- database was not created.
- We call the **connect_error** function to generate an error.
- **mysql_get_server_info()** returns the MySQL server version.
- **mysql_close()** closes the connection to the database.

Closing the connection is not necessary. But it is a good practice.

# First Connection through PHP script

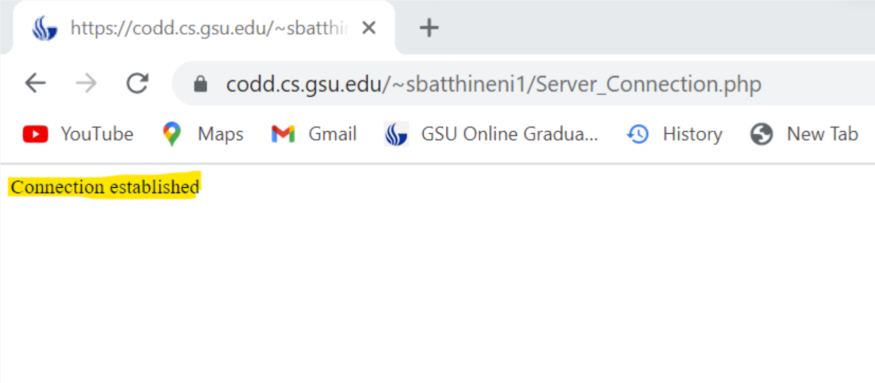
Step 1: Create a .php file with the above code

Step 2: Connect to the Codd server to transfer the files through SCP/SFTP clients such as WinSCP or FileZilla.

Step 3: Place this PHP file in your personal Web directory (public_html) for the files to be visible on the web.

Step 4: Open the URL - https://codd.cs.gsu.edu/~jdoe1/Homework/example.html

Output


https://codd.cs.gsu.edu/~sbatthi
codd.cs.gsu.edu/~sbatthineni1/Server_Connection.php
YouTube
Maps
Gmail
GSU Online Gradua...
History
New Tab

```
Connection established
```

-------Cool! We are now connected to the server 🥱---------

# Relational Database

A relational database organizes data into tables that can be linked—or related—based on data common to each.

A relational database stores data about entities. An entity is any object that an application wants to store information about.

Ex: A university database may need to store data about students, faculty, and course entities.

------- Programmers, lets dig into the query world ---------

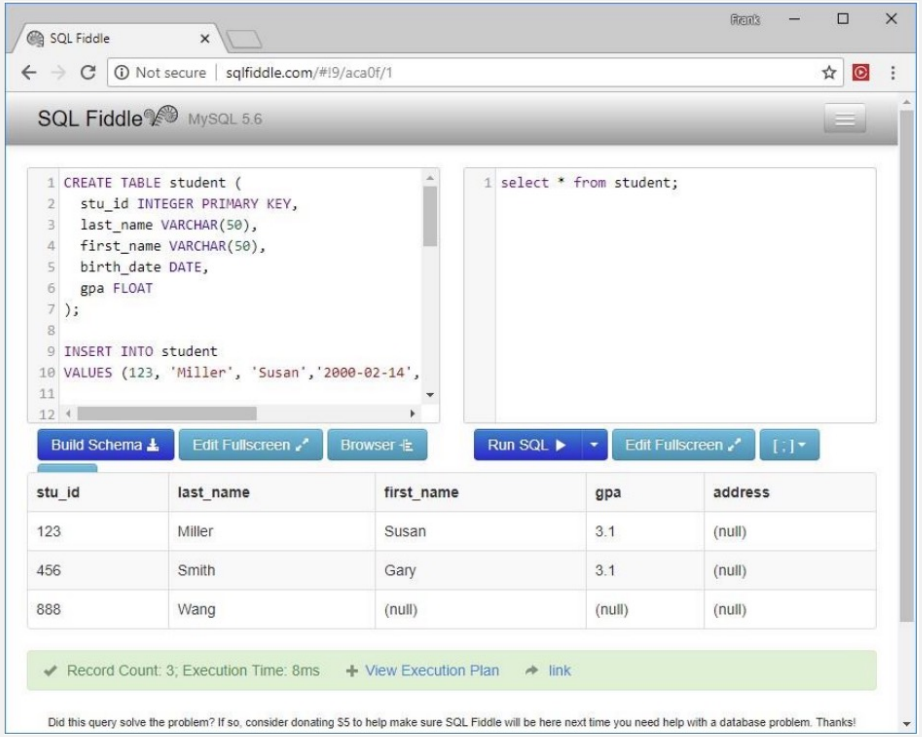
# CRUD Operations & SQL Fiddle

## CRUD operations

The acronym CRUD names the four common operations that are performed on data: Create, Retrieve, Update, and Delete

## SQL Fiddle

Some websites allow developers to run SQL statements without installing an RDBMS.



```
CREATE TABLE student (
  stu_id INTEGER PRIMARY KEY,
  last_name VARCHAR(50),
  first_name VARCHAR(50),
  birth_date DATE,
  gpa FLOAT
);

INSERT INTO student
VALUES (123, 'Miller', 'Susan','2000-02-14',
```

```
select * from student;
```

| stu_id | last_name | first_name | gpa | address |
|---|---|---|---|---|
| 123 | Miller | Susan | 3.1 | (null) |
| 456 | Smith | Gary | 3.1 | (null) |
| 888 | Wang | (null) | (null) | (null) |

✓ Record Count: 3; Execution Time: 8ms

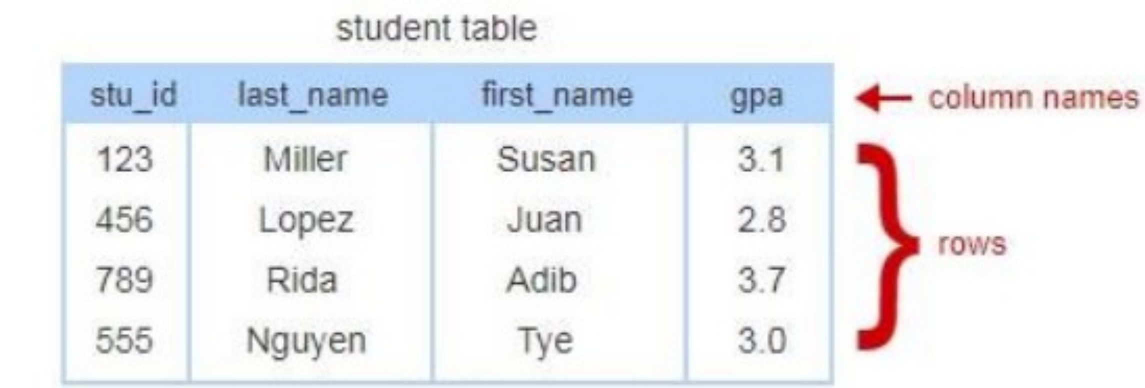
Did this query solve the problem? If so, consider donating $5 to help make sure SQL Fiddle will be here next time you need help with a database problem. Thanks!

We're doing it #thestateway

# Creating Tables in MySQLi

A table is a collection of related entities that is organized into columns and rows, like a spreadsheet.

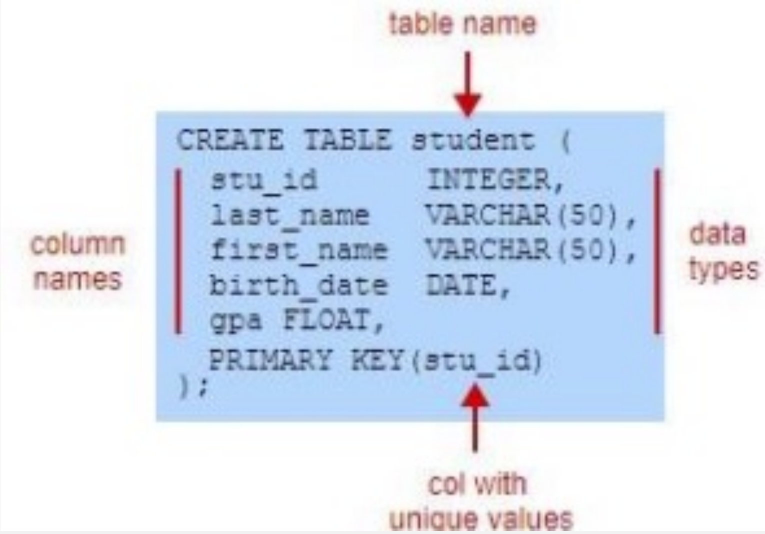
**Example:**



student table

| stu_id | last_name | first_name | gpa |
|---|---|---|---|
| 123 | Miller | Susan | 3.1 |
| 456 | Lopez | Juan | 2.8 |
| 789 | Rida | Adib | 3.7 |
| 555 | Nguyen | Tye | 3.0 |

column names
rows



```
CREATE TABLE student (
  stu_id        INTEGER,
  last_name     VARCHAR(50),
  first_name    VARCHAR(50),
  birth_date    DATE,
  gpa FLOAT,
  PRIMARY KEY(stu_id)
);
```

table name
column names
data types
col with unique values

Each column has a data type. Data types vary between relational databases, but 3 common data types include:

Character: Ex: A student name is a character column

Number: A student ID and GPA

Date:  Calendar dates, A student's birth date is a date column

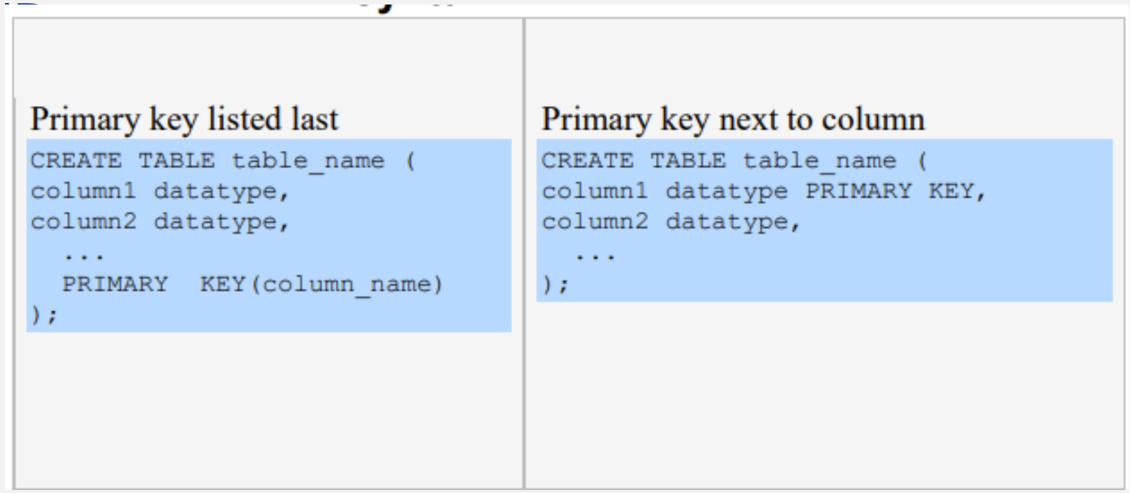
We're doing it #thestateway

# Keys in DB Tables

Every table must have a primary key, a column or combination of columns that uniquely identifies each row.

Two tables that have a relationship are linked together using a foreign key, a primary key from one table that is shared in another table.

**Example:** The student table's primary key is stu_id, because every student has a unique student



Primary key listed last

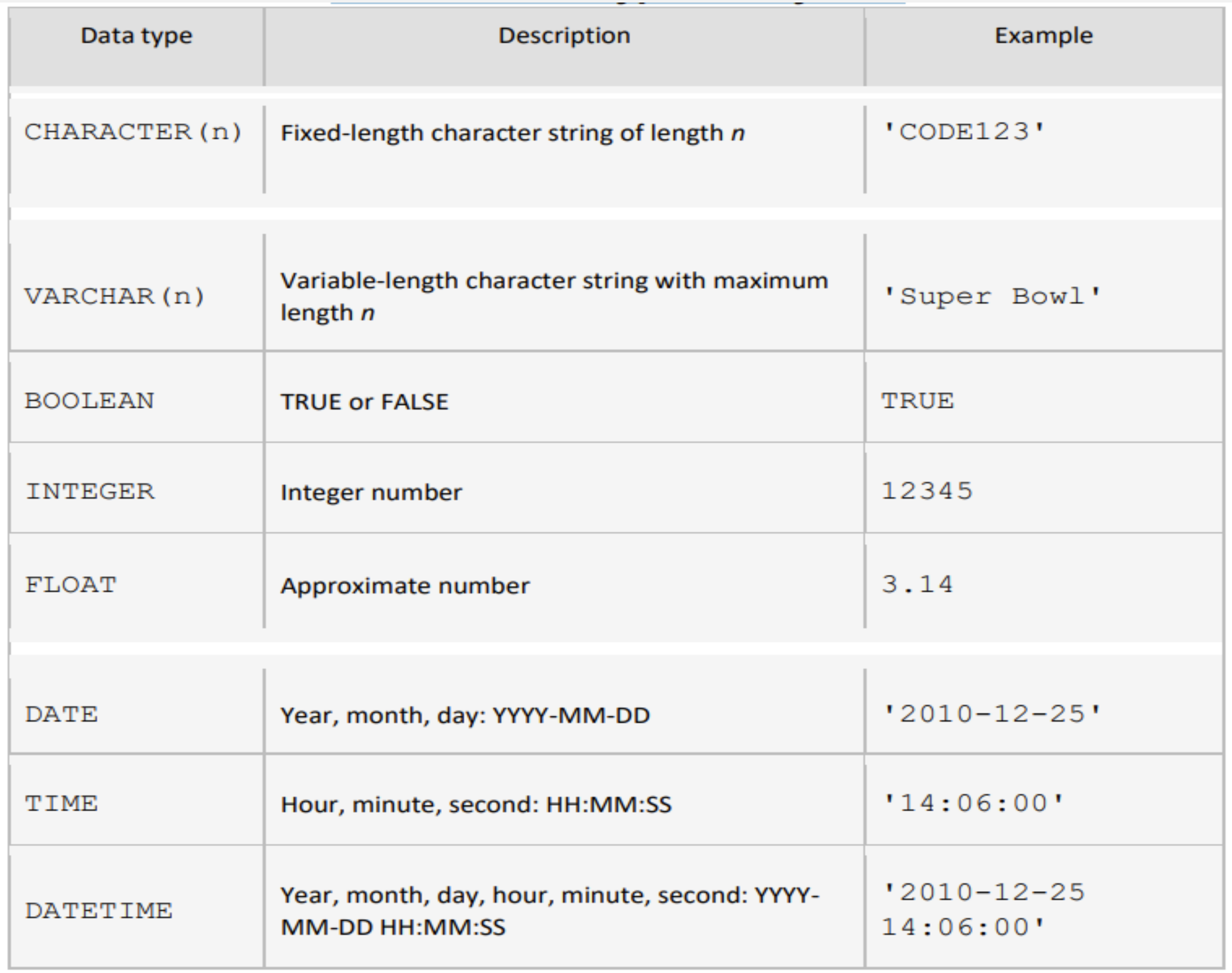
```
CREATE TABLE table_name (
column1 datatype,
column2 datatype,
  ...
  PRIMARY  KEY(column_name)
);
```

Primary key next to column

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

W3 schools Link - https://www.w3schools.com/mysql/mysql_create_table.asp

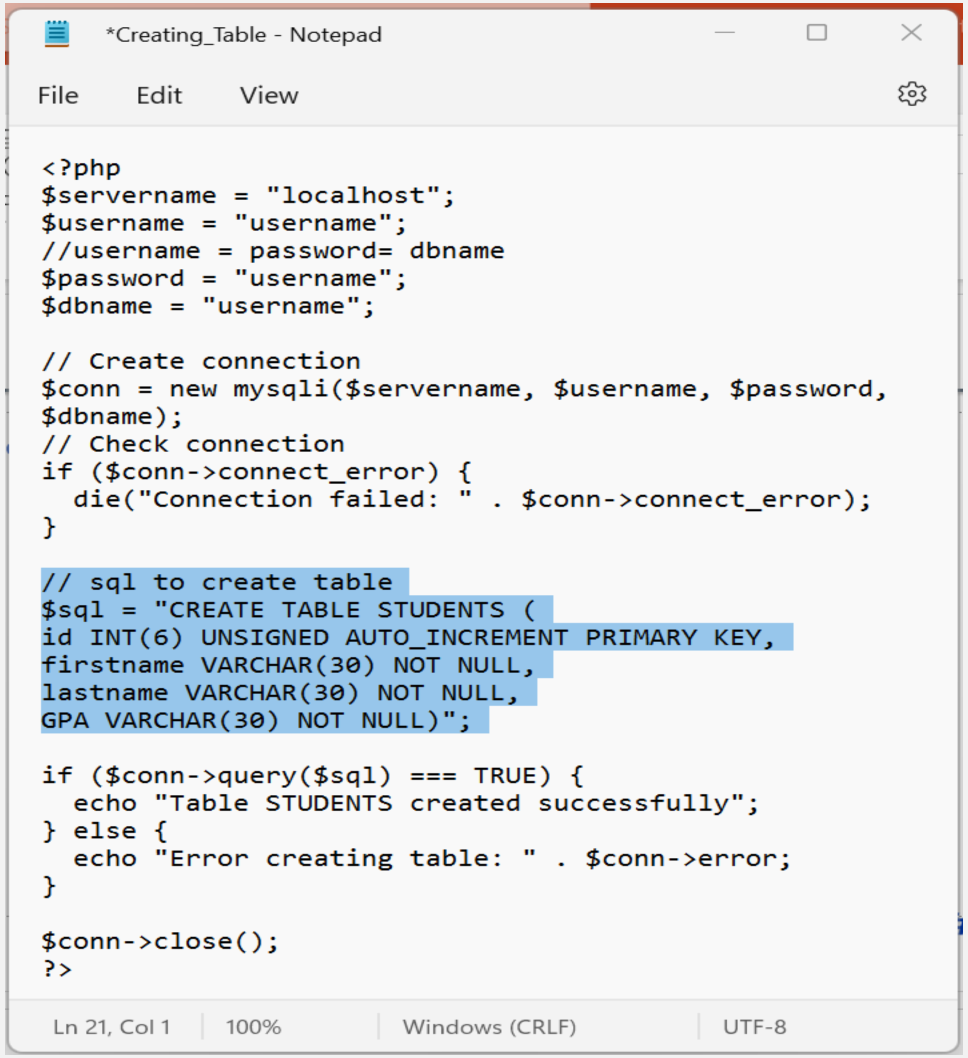
# Common data types in MySQL



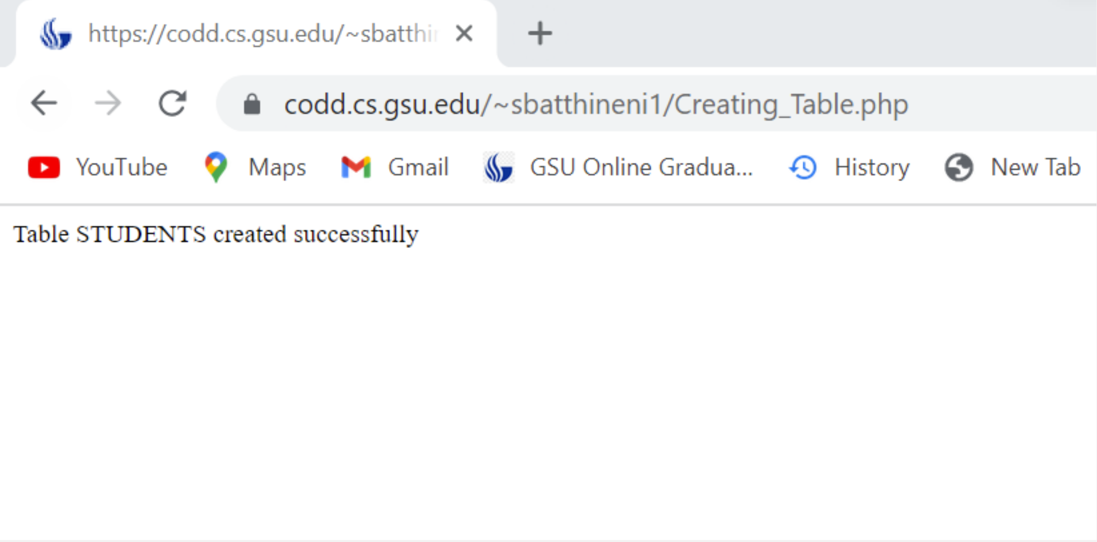
| Data type | Description | Example |
|---|---|---|
| CHARACTER(n) | Fixed-length character string of length *n* | 'CODE123' |
| VARCHAR(n) | Variable-length character string with maximum length *n* | 'Super Bowl' |
| BOOLEAN | TRUE or FALSE | TRUE |
| INTEGER | Integer number | 12345 |
| FLOAT | Approximate number | 3.14 |
| DATE | Year, month, day: YYYY-MM-DD | '2010-12-25' |
| TIME | Hour, minute, second: HH:MM:SS | '14:06:00' |
| DATETIME | Year, month, day, hour, minute, second: YYYY-MM-DD HH:MM:SS | '2010-12-25 14:06:00' |

**We're doing it #thestateway**

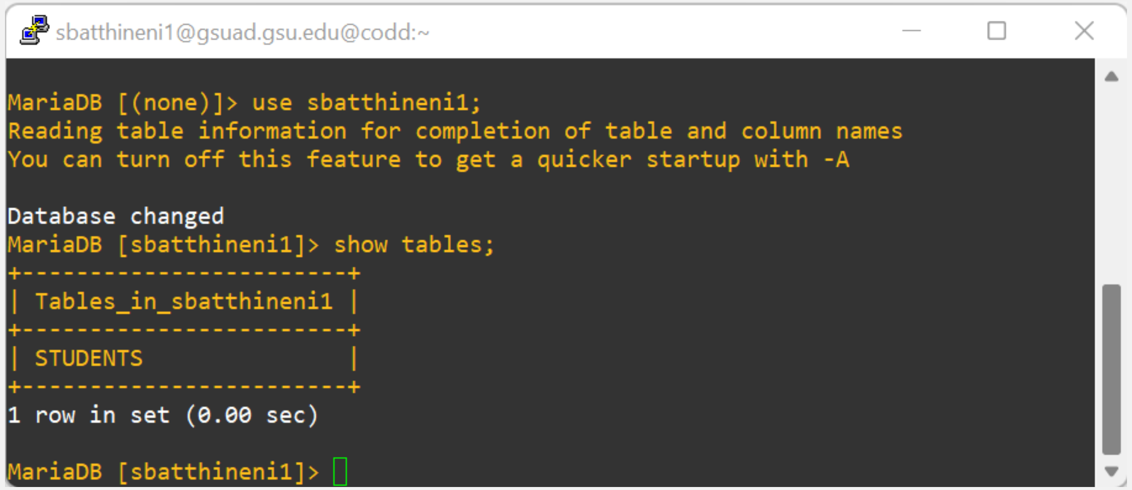
# Creating Tables in MySQLi through PHP



```
<?php
$servername = "localhost";
$username = "username";
//username = password= dbname
$password = "username";
$dbname = "username";

// Create connection
$conn = new mysqli($servername, $username, $password,
$dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// sql to create table
$sql = "CREATE TABLE STUDENTS (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
GPA VARCHAR(30) NOT NULL)";

if ($conn->query($sql) === TRUE) {
  echo "Table STUDENTS created successfully";
} else {
  echo "Error creating table: " . $conn->error;
}

$conn->close();
?>
```


https://codd.cs.gsu.edu/~sbatthi
codd.cs.gsu.edu/~sbatthineni1/Creating_Table.php
YouTube
Maps
Gmail
GSU Online Gradua...
History
New Tab

```
Table STUDENTS created successfully
```



```
MariaDB [(none)]> use sbatthineni1;
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
MariaDB [sbatthineni1]> show tables;
+------------------------+
| Tables_in_sbatthineni1 |
+------------------------+
| STUDENTS               |
+------------------------+
1 row in set (0.00 sec)

MariaDB [sbatthineni1]>
```

We're doing it #thestateway

# NOT NULL & Auto-increment

## NOT NULL

The NOT NULL constraint, listed by a column name in a CREATE TABLE statement prevents a column from having a NULL value.

The primary key column(s) do not need a NOT NULL constraint because a primary key cannot be NULL.

## Auto-increment columns

An autoincrement column is a column that is assigned an automatically incrementing value.

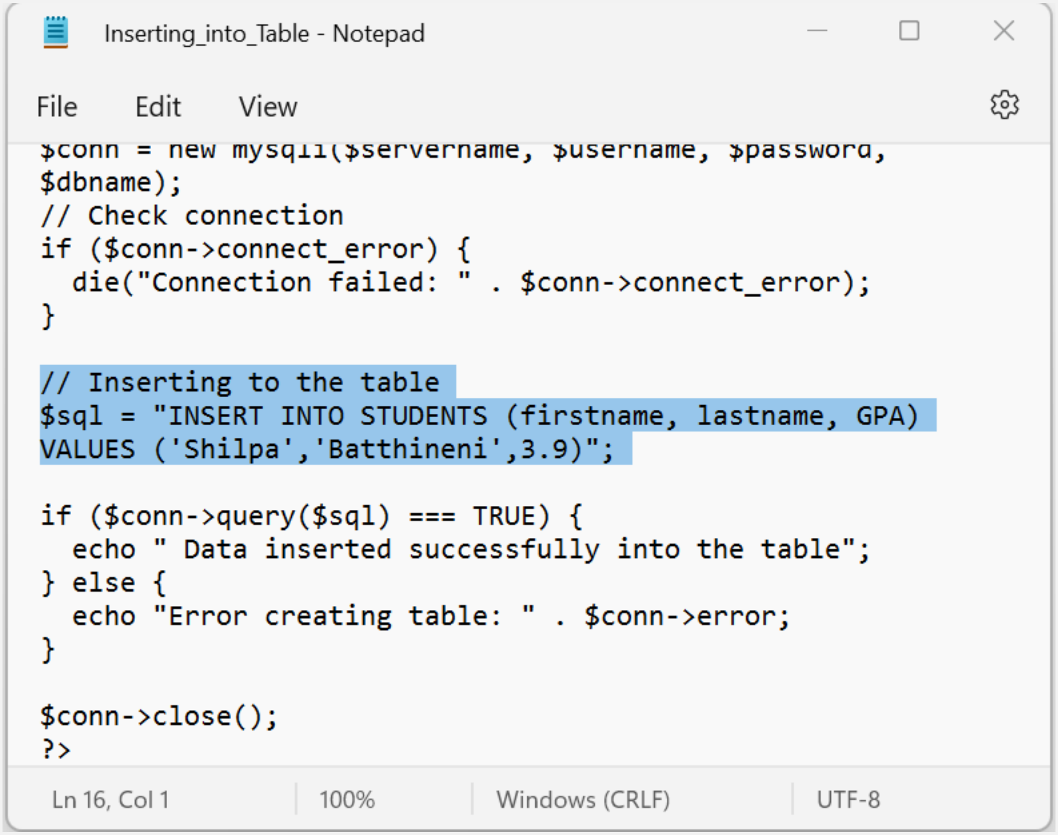
Ex: An auto-incrementing column (e.g., stu_ID column )may be assigned 1, 2,3, etc. for each row that is inserted into the table.

MySQL uses the AUTO_INCREMENT keyword to define an auto-increment column.

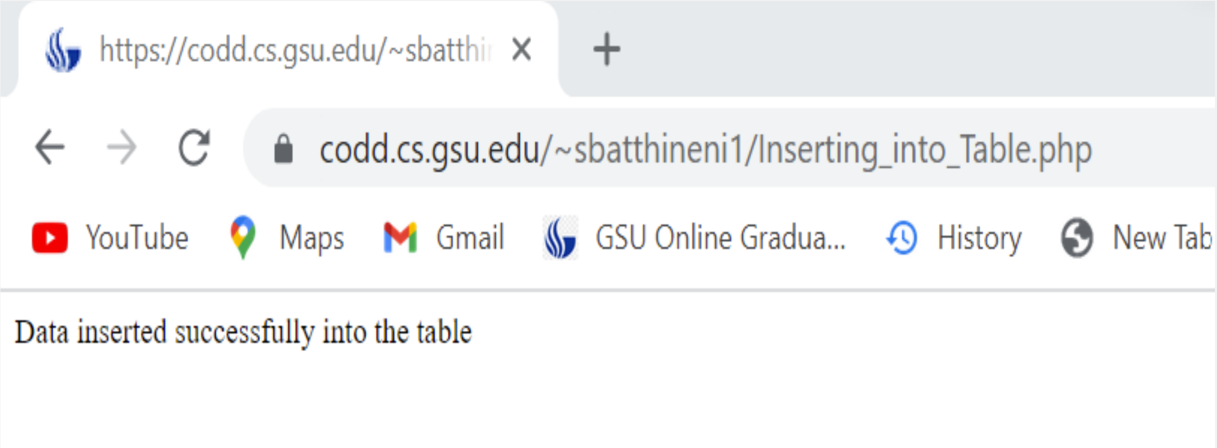
# INSERT STATEMENT

## INSERT STATEMENT:

When a table has an auto-increment column, the INSERT statement should not specify a value for the auto-increment column. Specifying it overwrites the automatically assigned value.



```
$conn = new mysqli($servername, $username, $password,
$dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// Inserting to the table
$sql = "INSERT INTO STUDENTS (firstname, lastname, GPA)
VALUES ('Shilpa','Batthineni',3.9)";

if ($conn->query($sql) === TRUE) {
  echo " Data inserted successfully into the table";
} else {
  echo "Error creating table: " . $conn->error;
}

$conn->close();
?>
```


https://codd.cs.gsu.edu/~sbatthi
codd.cs.gsu.edu/~sbatthineni1/Inserting_into_Table.php
YouTube
Maps
Gmail
GSU Online Gradua...
History
New Tab

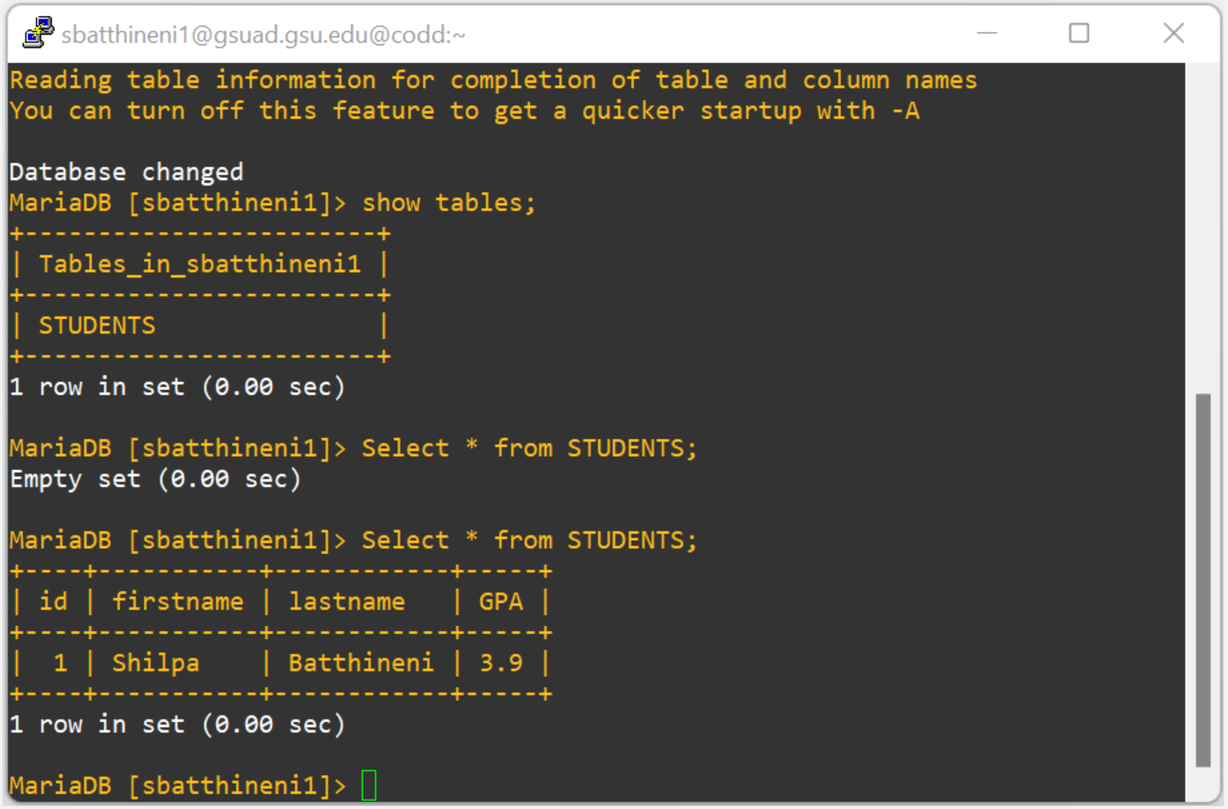
```
Data inserted successfully into the table
```

# SELECT STATEMENT

## SELECT STATEMENT:

The SELECT statement retrieves data from a table. The column names listed after "SELECT" are included in the retrieval result.

If "*" is given as a column name, then all columns are included in the retrieval result.



```
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
MariaDB [sbatthineni1]> show tables;
+------------------------+
| Tables_in_sbatthineni1 |
+------------------------+
| STUDENTS               |
+------------------------+
1 row in set (0.00 sec)

MariaDB [sbatthineni1]> Select * from STUDENTS;
Empty set (0.00 sec)

MariaDB [sbatthineni1]> Select * from STUDENTS;
+----+-----------+------------+-----+
| id | firstname | lastname   | GPA |
+----+-----------+------------+-----+
|  1 | Shilpa    | Batthineni | 3.9 |
+----+-----------+------------+-----+
1 row in set (0.00 sec)

MariaDB [sbatthineni1]> 
```

We're doing it #thestateway

# INDEX in SELECT Statement

Selecting information from large databases can be slow without the use of indexes.

Ex: A database of university faculty information may only have a few hundred rows, but a government database of taxpayer information is likely to contain millions of rows.

**WHERE clause**

WHERE clause in a select statement limits the rows that are retrieved, it works like an if statement in a programming language, specifying conditions that must be true for a row to be selected.

**EXAMPLE** - `select * from STUDENTS WHERE GPA > 3.7;`

**ORDER BY clause**

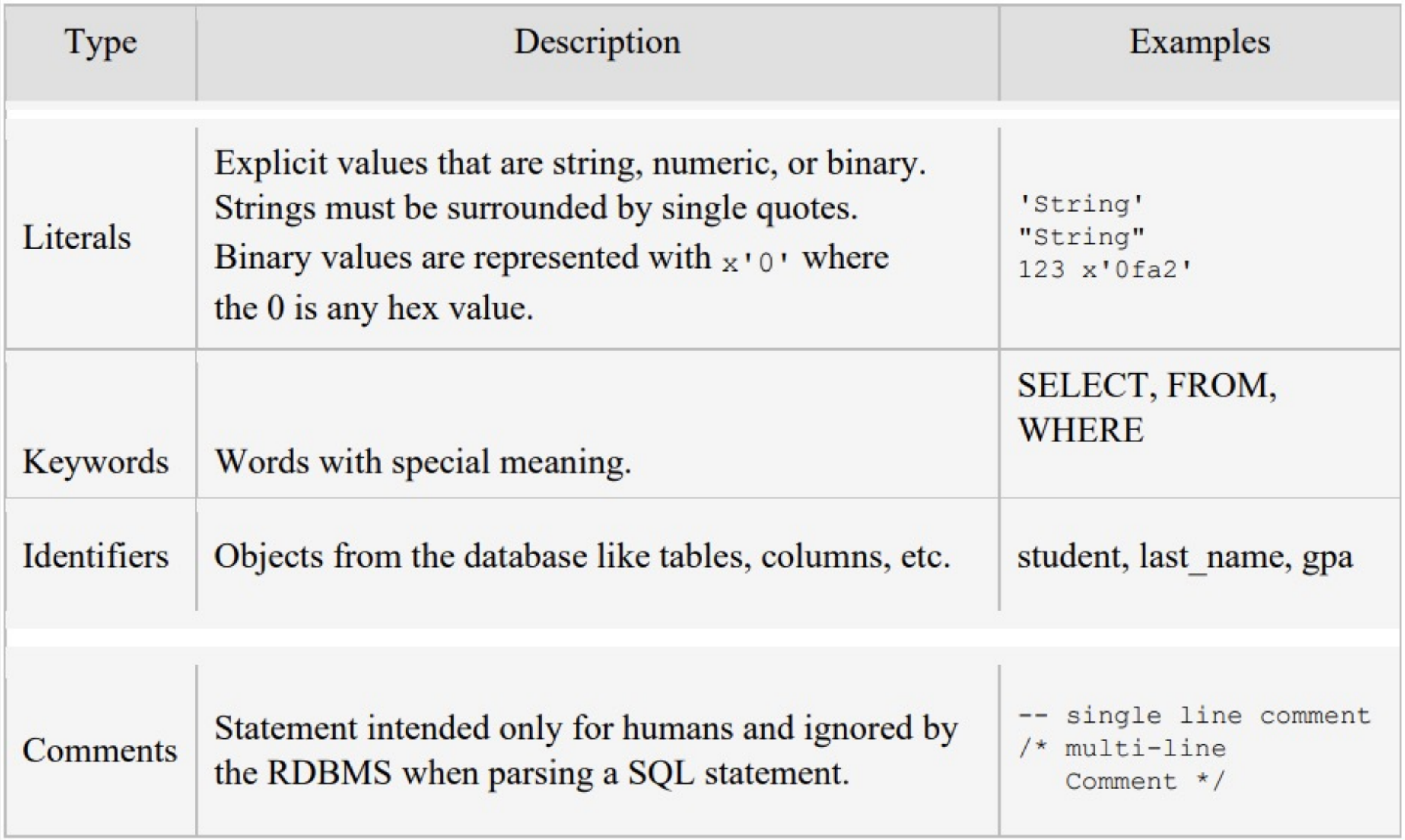
By default, a SELECT statement selects rows in the order the rows appear in the table. The ORDER BY clause sorts the selected rows in ascending order.

The DESC keyword with the ORDER BY clause sorts the rows in descending order.

**EXAMPLE** - `select * from STUDENTS ORDER BY lastname;`

**We're doing it #thestateway**

# Summary of SQL syntax features.



| Type | Description | Examples |
|---|---|---|
| Literals | Explicit values that are string, numeric, or binary. Strings must be surrounded by single quotes. Binary values are represented with `x'0'` where the 0 is any hex value. | `'String'`<br>`"String"`<br>`123 x'0fa2'` |
| Keywords | Words with special meaning. | SELECT, FROM, WHERE |
| Identifiers | Objects from the database like tables, columns, etc. | student, last_name, gpa |
| Comments | Statement intended only for humans and ignored by the RDBMS when parsing a SQL statement. | `-- single line comment`<br>`/* multi-line`<br>`Comment */` |

SELECT last name (SELECT clause)

FROM student (FROM clause)

WHERE GPA > 3.0; (WHERE clause)

The three clauses ending in a semicolon in a statement.

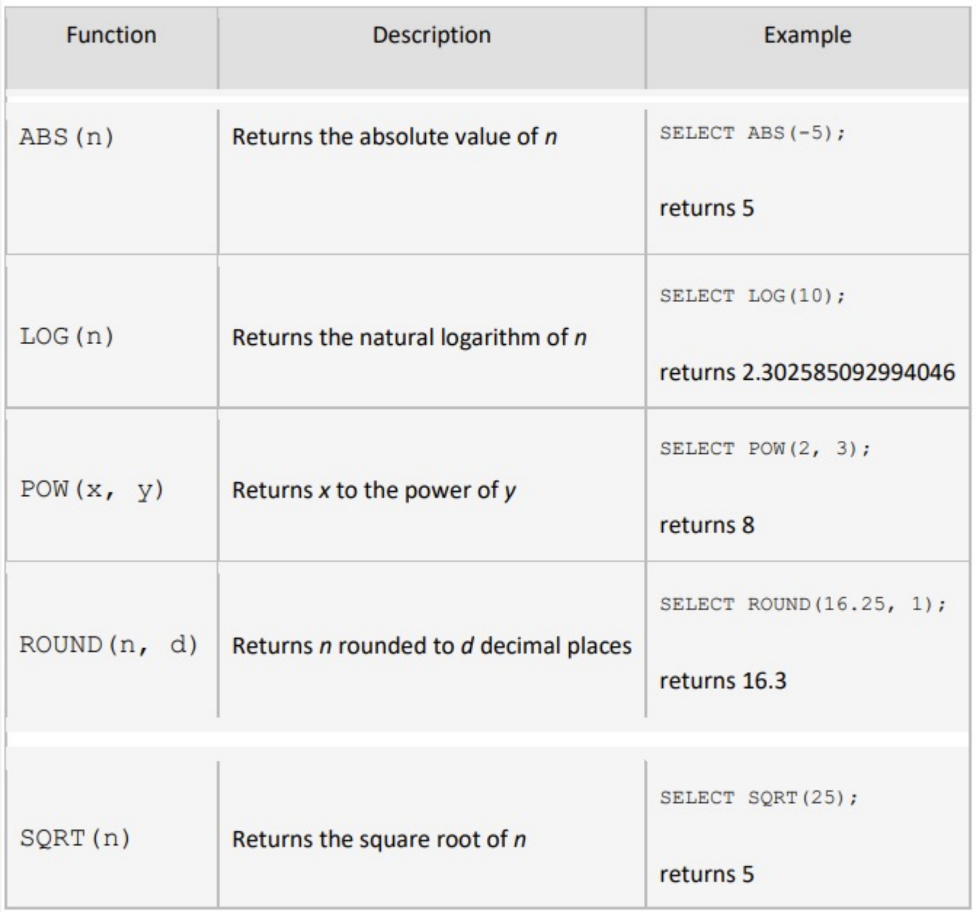
# SQL FUNCTIONS

RDBMS provides a wide range of SQL functions that work with or manipulate numbers, strings, dates, and times. Some common aggregate functions include:

- COUNT() - Counts the number of rows retrieved by a SELECT statement.
- MIN() - Finds the minimum value in a group.
- MAX() - Finds the maximum value in a group.
- SUM() - Sums all the values in a group.
- AVG() - Finds the arithmetic mean (average) of all the values in a group.

# Numeric functions and operators

SQL arithmetic operators allow simple calculations to be performed in a SQL statement. RDBMS provide various numeric functions to perform more advanced calculations.

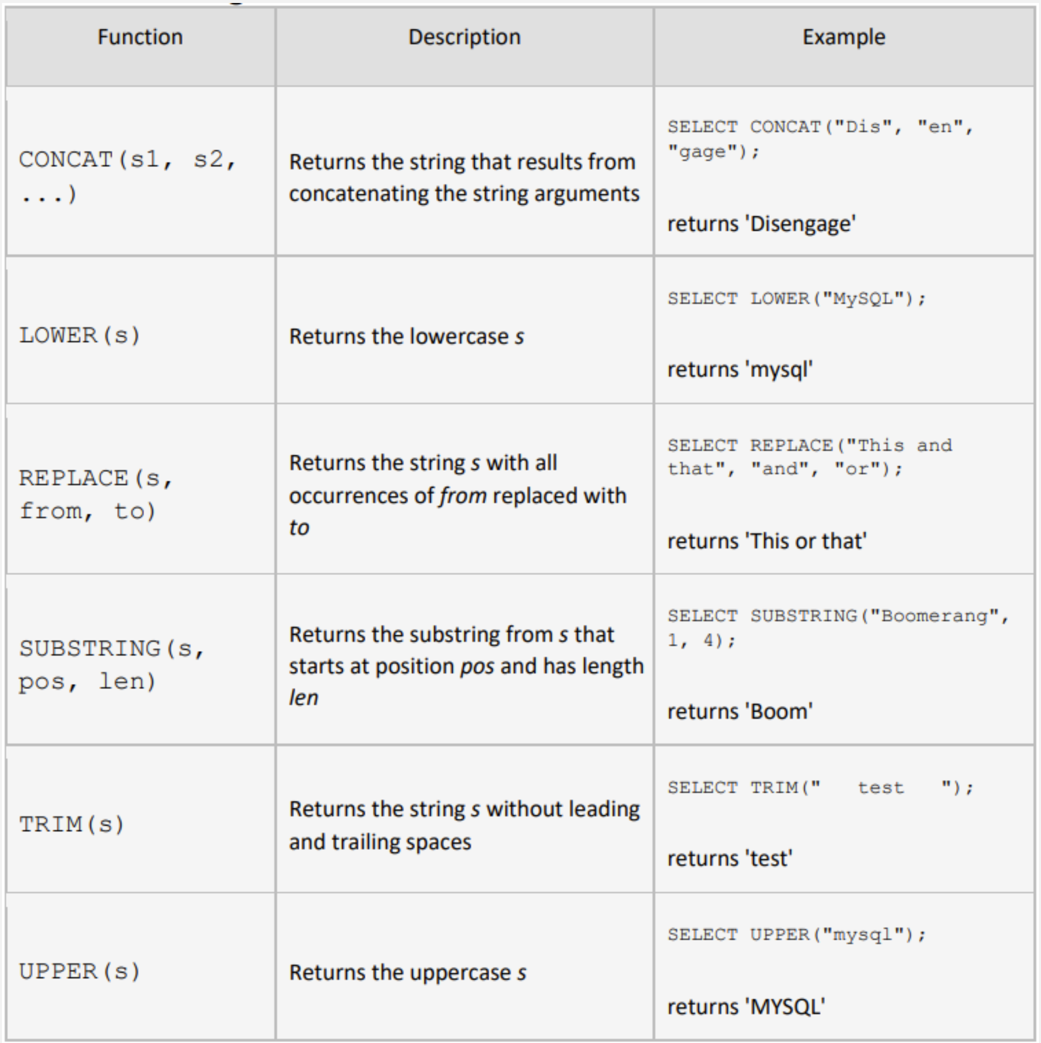


| Function | Description | Example |
|---|---|---|
| `ABS(n)` | Returns the absolute value of *n* | `SELECT ABS(-5);`<br>returns 5 |
| `LOG(n)` | Returns the natural logarithm of *n* | `SELECT LOG(10);`<br>returns 2.302585092994046 |
| `POW(x, y)` | Returns *x* to the power of *y* | `SELECT POW(2, 3);`<br>returns 8 |
| `ROUND(n, d)` | Returns *n* rounded to *d* decimal places | `SELECT ROUND(16.25, 1);`<br>returns 16.3 |
| `SQRT(n)` | Returns the square root of *n* | `SELECT SQRT(25);`<br>returns 5 |

**We're doing it #thestateway**

# String functions

SQL string functions manipulate strings. Some common string functions are summarized in the table below.

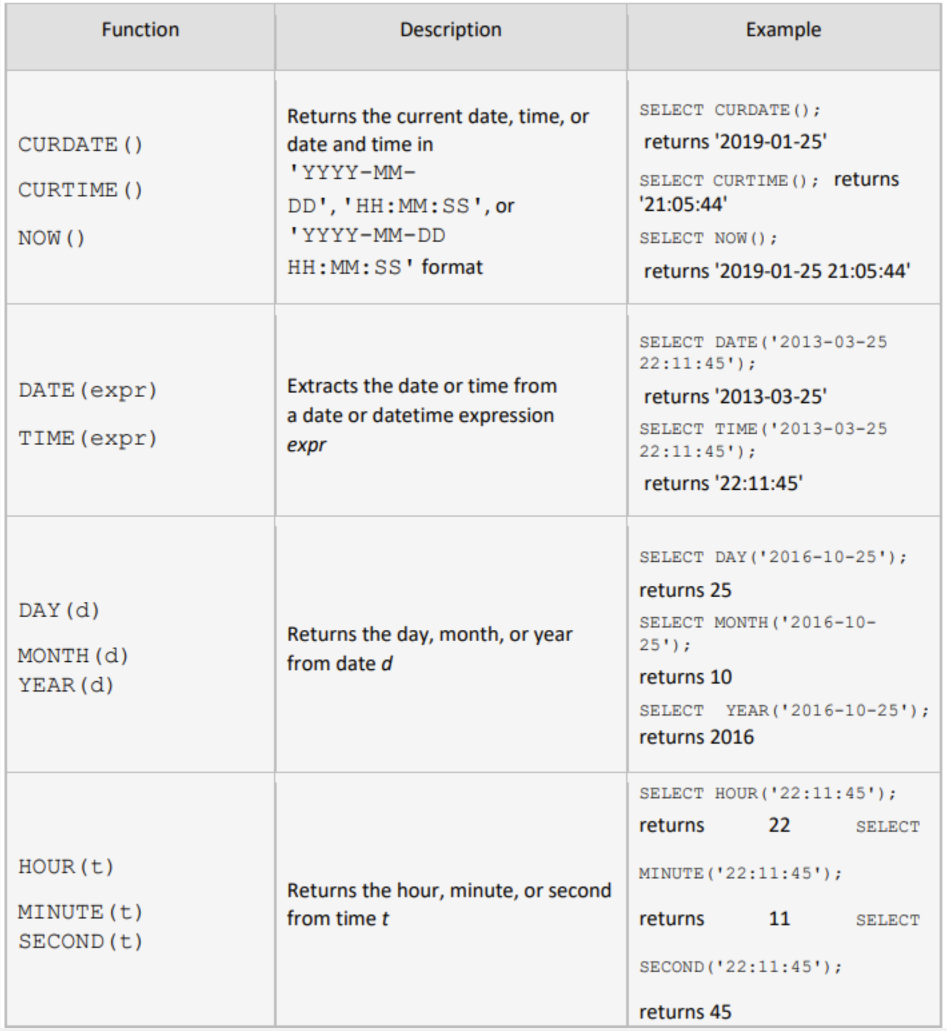


| Function | Description | Example |
|---|---|---|
| `CONCAT(s1, s2, ...)` | Returns the string that results from concatenating the string arguments | `SELECT CONCAT("Dis", "en", "gage");`<br>returns 'Disengage' |
| `LOWER(s)` | Returns the lowercase *s* | `SELECT LOWER("MySQL");`<br>returns 'mysql' |
| `REPLACE(s, from, to)` | Returns the string *s* with all occurrences of *from* replaced with *to* | `SELECT REPLACE("This and that", "and", "or");`<br>returns 'This or that' |
| `SUBSTRING(s, pos, len)` | Returns the substring from *s* that starts at position *pos* and has length *len* | `SELECT SUBSTRING("Boomerang", 1, 4);`<br>returns 'Boom' |
| `TRIM(s)` | Returns the string *s* without leading and trailing spaces | `SELECT TRIM("   test   ");`<br>returns 'test' |
| `UPPER(s)` | Returns the uppercase *s* | `SELECT UPPER("mysql");`<br>returns 'MYSQL' |

**We're doing it #thestateway**

# Common Date & functions

Date and time functions operate on DATE, TIME, and DATETIME columns.



| Function | Description | Example |
|---|---|---|
| CURDATE()<br>CURTIME()<br>NOW() | Returns the current date, time, or date and time in 'YYYY-MM-DD', 'HH:MM:SS', or 'YYYY-MM-DD HH:MM:SS' format | SELECT CURDATE();<br>returns '2019-01-25'<br>SELECT CURTIME(); returns '21:05:44'<br>SELECT NOW();<br>returns '2019-01-25 21:05:44' |
| DATE(expr)<br>TIME(expr) | Extracts the date or time from a date or datetime expression *expr* | SELECT DATE('2013-03-25 22:11:45');<br>returns '2013-03-25'<br>SELECT TIME('2013-03-25 22:11:45');<br>returns '22:11:45' |
| DAY(d)<br>MONTH(d)<br>YEAR(d) | Returns the day, month, or year from date *d* | SELECT DAY('2016-10-25');<br>returns 25<br>SELECT MONTH('2016-10-25');<br>returns 10<br>SELECT YEAR('2016-10-25');<br>returns 2016 |
| HOUR(t)<br>MINUTE(t)<br>SECOND(t) | Returns the hour, minute, or second from time *t* | SELECT HOUR('22:11:45');<br>returns 22 SELECT MINUTE('22:11:45');<br>returns 11 SELECT SECOND('22:11:45');<br>returns 45 |

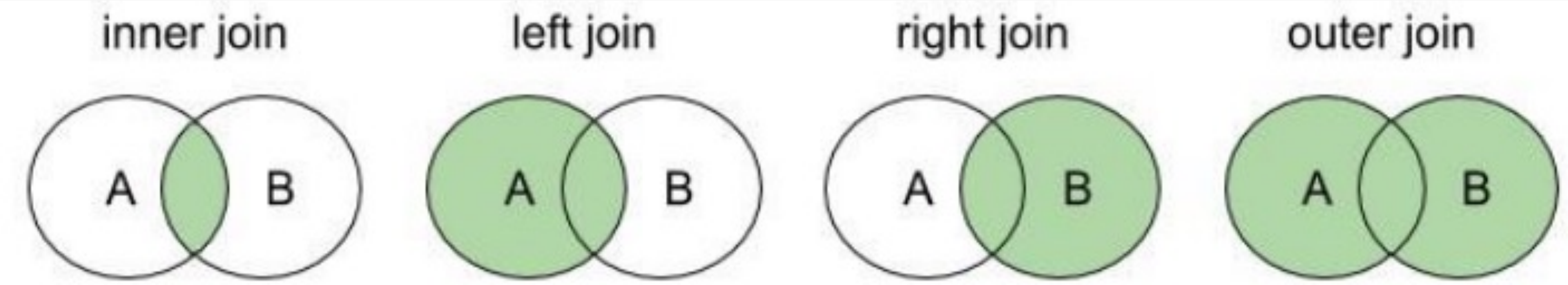
We're doing it #thestateway

# Joining Tables

A SQL JOIN joins information from two or more tables in a SELECT statement. There are four types of joins that exist for two tables A and B: inner, left, right, and outer joins.

Ex: A student's name from one table may be joined with a student’s grade on another table.

**Venn diagrams showing which rows are joined from table A and table B.**



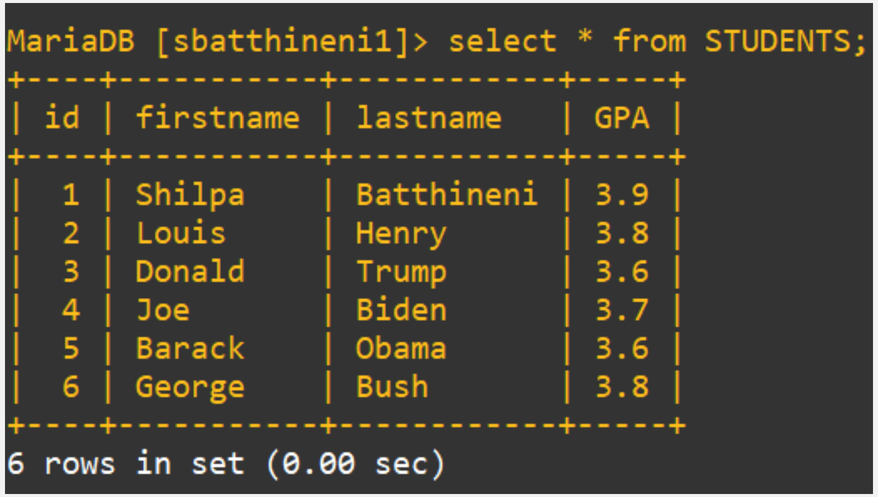
**We’re doing it #thestateway**

# INNER JOIN

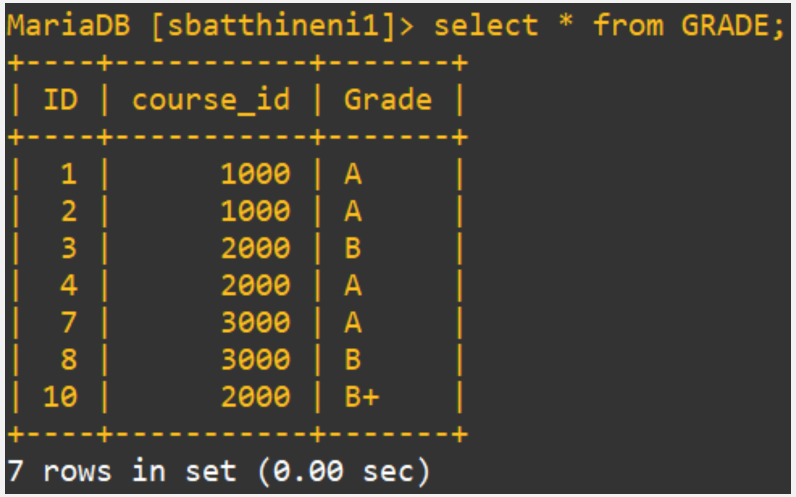
An INNER JOIN clause performs an inner join, joining all rows from the left and right tables where the join condition is met.

**Syntax:** Select * from left table **INNER JOIN** right table on condition;

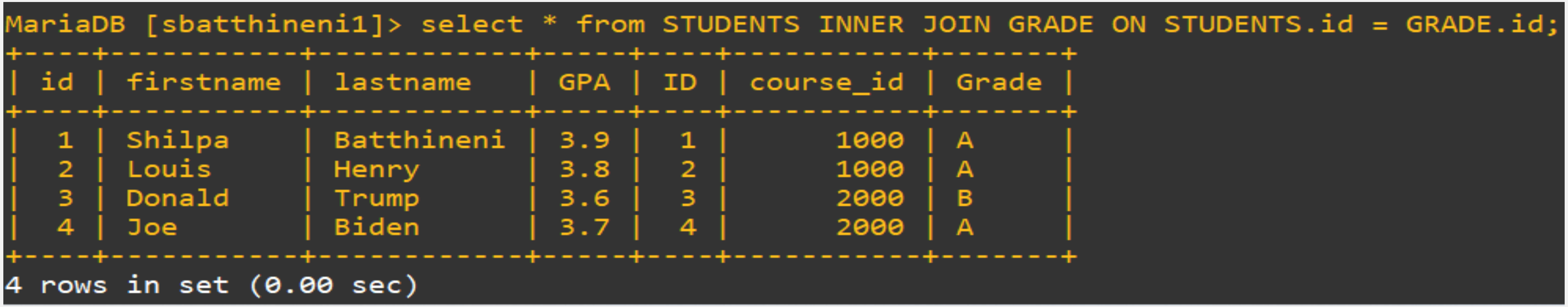
**Example**



```
MariaDB [sbatthineni1]> select * from STUDENTS;
+----+-----------+------------+-----+
| id | firstname | lastname   | GPA |
+----+-----------+------------+-----+
|  1 | Shilpa    | Batthineni | 3.9 |
|  2 | Louis     | Henry      | 3.8 |
|  3 | Donald    | Trump      | 3.6 |
|  4 | Joe       | Biden      | 3.7 |
|  5 | Barack    | Obama      | 3.6 |
|  6 | George    | Bush       | 3.8 |
+----+-----------+------------+-----+
6 rows in set (0.00 sec)
```



```
MariaDB [sbatthineni1]> select * from GRADE;
+----+-----------+-------+
| ID | course_id | Grade |
+----+-----------+-------+
|  1 |      1000 | A     |
|  2 |      1000 | A     |
|  3 |      2000 | B     |
|  4 |      2000 | A     |
|  7 |      3000 | A     |
|  8 |      3000 | B     |
| 10 |      2000 | B+    |
+----+-----------+-------+
7 rows in set (0.00 sec)
```

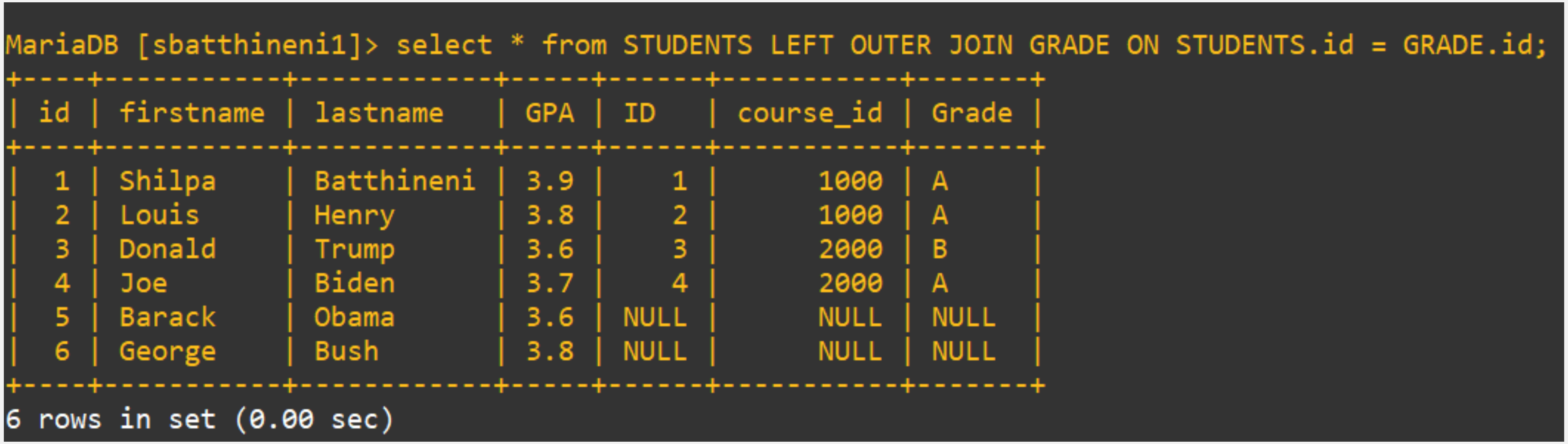


```
MariaDB [sbatthineni1]> select * from STUDENTS INNER JOIN GRADE ON STUDENTS.id = GRADE.id;
+----+-----------+------------+-----+----+-----------+-------+
| id | firstname | lastname   | GPA | ID | course_id | Grade |
+----+-----------+------------+-----+----+-----------+-------+
|  1 | Shilpa    | Batthineni | 3.9 |  1 |      1000 | A     |
|  2 | Louis     | Henry      | 3.8 |  2 |      1000 | A     |
|  3 | Donald    | Trump      | 3.6 |  3 |      2000 | B     |
|  4 | Joe       | Biden      | 3.7 |  4 |      2000 | A     |
+----+-----------+------------+-----+----+-----------+-------+
4 rows in set (0.00 sec)
```

# LEFT OUTER JOIN

LEFT OUTER JOIN clause - Joins all rows from the left table along with rows from the right table where the join condition is met.

The LEFT JOIN includes all rows from STUDENTS and GRADE that have matching student IDs, **plus** rows from the STUDENTS table that have student IDs that do not appear in the GRADE table.



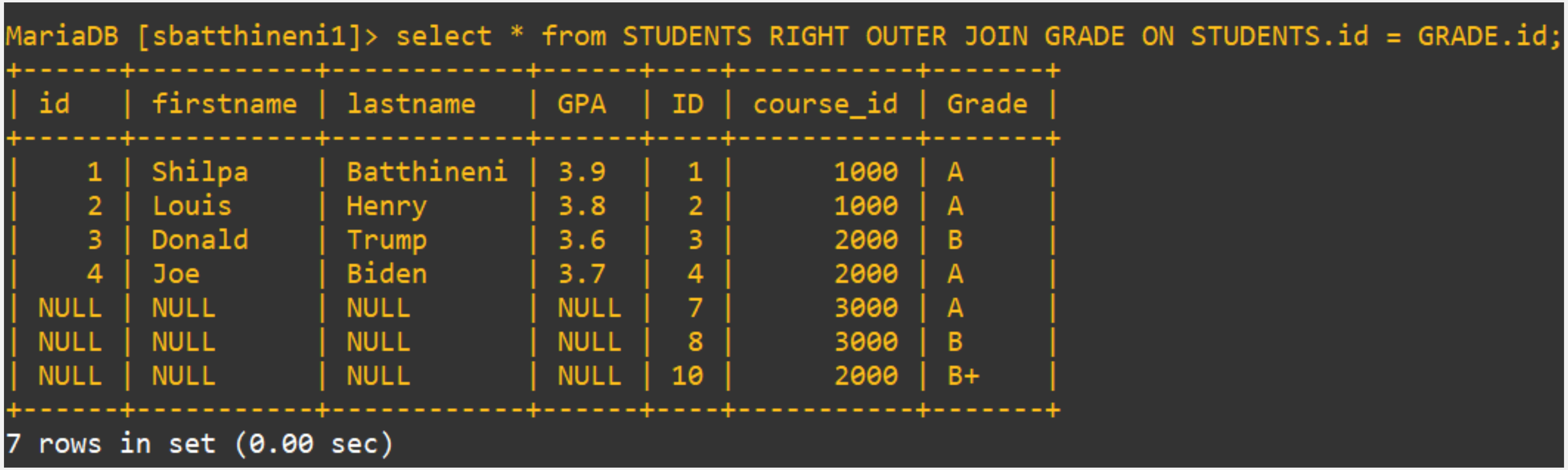
```
MariaDB [sbatthineni1]> select * from STUDENTS LEFT OUTER JOIN GRADE ON STUDENTS.id = GRADE.id;
+----+-----------+------------+-----+------+-----------+-------+
| id | firstname | lastname   | GPA | ID   | course_id | Grade |
+----+-----------+------------+-----+------+-----------+-------+
|  1 | Shilpa    | Batthineni | 3.9 |    1 |      1000 | A     |
|  2 | Louis     | Henry      | 3.8 |    2 |      1000 | A     |
|  3 | Donald    | Trump      | 3.6 |    3 |      2000 | B     |
|  4 | Joe       | Biden      | 3.7 |    4 |      2000 | A     |
|  5 | Barack    | Obama      | 3.6 | NULL |      NULL | NULL  |
|  6 | George    | Bush       | 3.8 | NULL |      NULL | NULL  |
+----+-----------+------------+-----+------+-----------+-------+
6 rows in set (0.00 sec)
```

We're doing it #thestateway

# RIGHT OUTER JOIN

RIGHT OUTER JOIN clause - Joins all rows from the right table along with rows from the left table where the join condition is met

The RIGHT JOIN includes all rows from STUDENT and GRADE that have matching student IDs, plus rows from the GRADE table that have student IDs that do not appear in the student table.

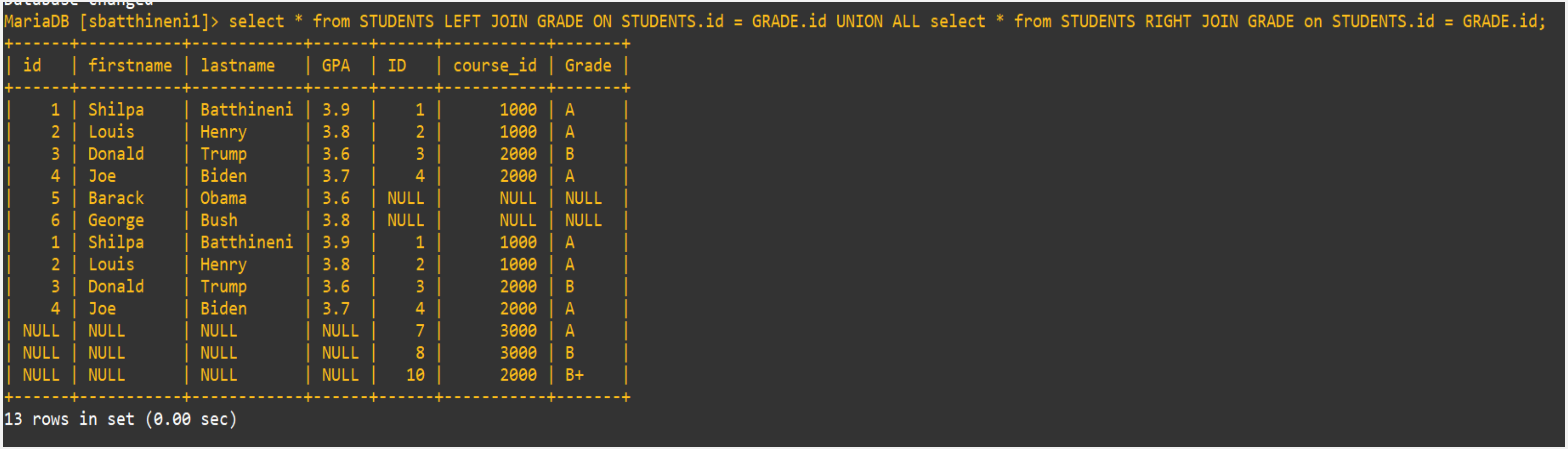


```
MariaDB [sbatthineni1]> select * from STUDENTS RIGHT OUTER JOIN GRADE ON STUDENTS.id = GRADE.id;
+------+-----------+------------+------+----+-----------+-------+
| id   | firstname | lastname   | GPA  | ID | course_id | Grade |
+------+-----------+------------+------+----+-----------+-------+
|    1 | Shilpa    | Batthineni | 3.9  |  1 |      1000 | A     |
|    2 | Louis     | Henry      | 3.8  |  2 |      1000 | A     |
|    3 | Donald    | Trump      | 3.6  |  3 |      2000 | B     |
|    4 | Joe       | Biden      | 3.7  |  4 |      2000 | A     |
| NULL | NULL      | NULL       | NULL |  7 |      3000 | A     |
| NULL | NULL      | NULL       | NULL |  8 |      3000 | B     |
| NULL | NULL      | NULL       | NULL | 10 |      2000 | B+    |
+------+-----------+------------+------+----+-----------+-------+
7 rows in set (0.00 sec)
```

# FULL OUTER JOIN

FULL OUTER JOIN clause - Joins all rows from the right table regardless join condition is met.

We can't do a full outer join in MySQL, so just run it as the UNION ALL of a LEFT JOIN and a RIGHT JOIN:



```
Database changed
MariaDB [sbatthineni1]> select * from STUDENTS LEFT JOIN GRADE ON STUDENTS.id = GRADE.id UNION ALL select * from STUDENTS RIGHT JOIN GRADE on STUDENTS.id = GRADE.id;
+------+-----------+------------+------+------+-----------+-------+
| id   | firstname | lastname   | GPA  | ID   | course_id | Grade |
+------+-----------+------------+------+------+-----------+-------+
|    1 | Shilpa    | Batthineni | 3.9  |    1 |      1000 | A     |
|    2 | Louis     | Henry      | 3.8  |    2 |      1000 | A     |
|    3 | Donald    | Trump      | 3.6  |    3 |      2000 | B     |
|    4 | Joe       | Biden      | 3.7  |    4 |      2000 | A     |
|    5 | Barack    | Obama      | 3.6  | NULL |      NULL | NULL  |
|    6 | George    | Bush       | 3.8  | NULL |      NULL | NULL  |
|    1 | Shilpa    | Batthineni | 3.9  |    1 |      1000 | A     |
|    2 | Louis     | Henry      | 3.8  |    2 |      1000 | A     |
|    3 | Donald    | Trump      | 3.6  |    3 |      2000 | B     |
|    4 | Joe       | Biden      | 3.7  |    4 |      2000 | A     |
| NULL | NULL      | NULL       | NULL |    7 |      3000 | A     |
| NULL | NULL      | NULL       | NULL |    8 |      3000 | B     |
| NULL | NULL      | NULL       | NULL |   10 |      2000 | B+    |
+------+-----------+------------+------+------+-----------+-------+
13 rows in set (0.00 sec)
```

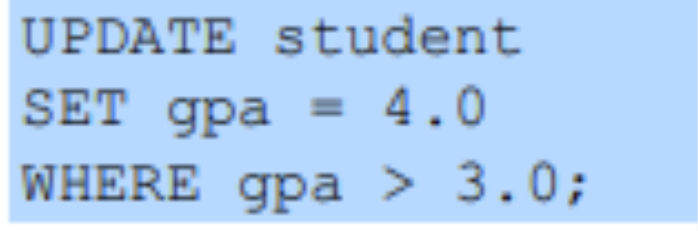
# UPDATE & DELETE Statements

## UPDATE statement

An UPDATE statement's WHERE clause can target more than one row at a time. In the figure below, every student with a GPA > 3.0 is assigned a 4.0 GPA.

If no WHERE clause is specified, every row in the table is updated.

**Syntax:**

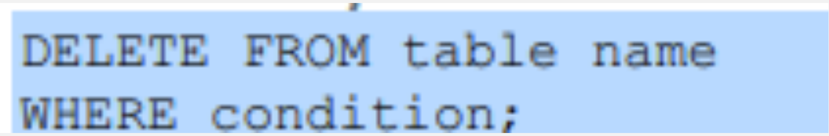


```
UPDATE student
SET gpa = 4.0
WHERE gpa > 3.0;
```

## DELETE statement

The DELETE statement removes rows from a table. The WHERE clause determines which rows are removed. If no WHERE clause is specified, all rows are removed.

**Syntax:**



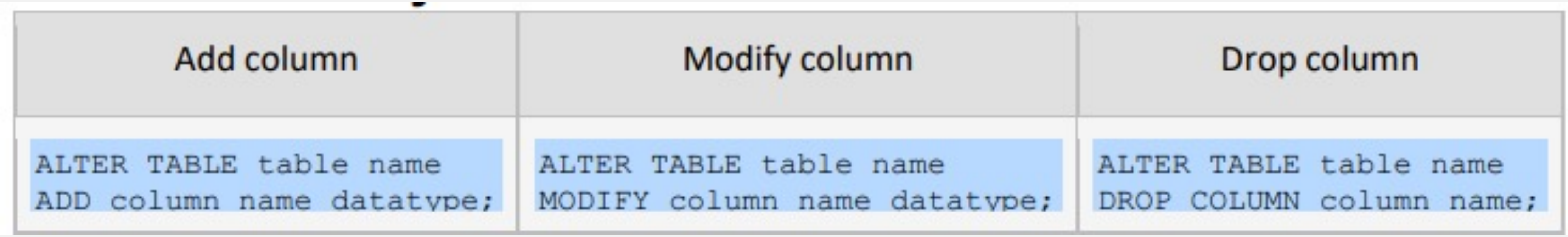
```
DELETE FROM table name
WHERE condition;
```

# ALTER & DROP TABLE

**ALTER TABLE:**

After creating a table, the table can be altered by adding or removing columns or changing a column's data type by using the ALTER TABLE statement.

**Syntax:**



| Add column | Modify column | Drop column |
|---|---|---|
| `ALTER TABLE table name`<br>`ADD column name datatype;` | `ALTER TABLE table name`<br>`MODIFY column name datatype;` | `ALTER TABLE table name`<br>`DROP COLUMN column name;` |

**DROP TABLE:**

The DROP TABLE statement deletes a table along with all the table's rows.

**Syntax:** DROP TABLE table_name;

We're doing it #thestateway

# Assignment

**Now as you are familiar of all the CRED operations:**

- Create an employee database with these fields:

emp_name

job_name

hire_date

salary

department:{

id,name,Location:}

And create a PHP page to save data from HTML form to your MySQL database.

# Learning Objective

You will learn about databases.

A relational database, SQL, MySQLi, and  differences between them.

You will be good at working with MySQL

Run CRED operations.