SQL

SQL

- Often times, in order for us to build the most functional website we can, we depend on a **database** to store information.

- If you've ever used Microsoft Excel or Google Spreadsheets (among others), odds are you're familiar with the notion of a database: a hierarchically organized set of tables, each of which contains a set of rows and columns.

SQL

- SQL (the *Structured Query Language*) is a programming language whose purpose is to **query** a database.

- **MySQL** is an open-source platform on which you can establish the type of relational database that SQL is most adept at working with.

- Many installations of MySQL come with a GUI tool called **phpMyAdmin** which can be used to execute database queries in a more user-friendly way.

SQL

- After you create a database, the next thing you'll most likely want to do is create a **table**.
  - The syntax for doing this is actually a bit awkward to do programmatically, at least at the outset, and so this is where phpMyAdmin will come in handy.

- As part of the process of creating a table, you'll be asked to specify all of the **columns** in that table.

- Thereafter, all your queries will refer to **rows** of the table.

SQL

- Each column of your SQL table is capable of holding data of a particular data type.

SQL

- Each column of your SQL table is capable of holding data of a particular data type.

INT	SMALLINT	TINYINT	MEDIUMINT	BIGINT
DECIMAL	FLOAT	BIT	DATE	TIME
DATETIME	TIMESTAMP	CHAR	VARCHAR	BINARY
BLOB	TEXT	ENUM	GEOMETRY	LINESTRING

SQL

- Each column of your SQL table is capable of holding data of a particular data type.

INT	SMALLINT	TINYINT	MEDIUMINT	BIGINT
DECIMAL	FLOAT	BIT	DATE	TIME
DATETIME	TIMESTAMP	CHAR	VARCHAR	BINARY
BLOB	TEXT	ENUM	GEOMETRY	LINESTRING

SQL

- Each column of your SQL table is capable of holding data of a particular data type.

INT	SMALLINT	TINYINT	MEDIUMINT	BIGINT
DECIMAL	FLOAT	BIT	DATE	TIME
DATETIME	TIMESTAMP	CHAR	VARCHAR	BINARY
BLOB	TEXT	ENUM	GEOMETRY	LINESTRING

SQL

- Each column of your SQL table is capable of holding data of a particular data type.

INT	SMALLINT	TINYINT	MEDIUMINT	BIGINT
DECIMAL	FLOAT	BIT	DATE	TIME
DATETIME	TIMESTAMP	CHAR	VARCHAR	BINARY
BLOB	TEXT	ENUM	GEOMETRY	LINESTRING

SQL

- Each column of your SQL table is capable of holding data of a particular data type.

INT	SMALLINT	TINYINT	MEDIUMINT	BIGINT
DECIMAL	FLOAT	BIT	DATE	TIME
DATETIME	TIMESTAMP	CHAR	VARCHAR	BINARY
BLOB	TEXT	ENUM	GEOMETRY	LINESTRING

SQL

- Each column of your SQL table is capable of holding data of a particular data type.

INT	SMALLINT	TINYINT	MEDIUMINT	BIGINT
DECIMAL	FLOAT	BIT	DATE	TIME
DATETIME	TIMESTAMP	CHAR	VARCHAR	BINARY
BLOB	TEXT	ENUM	GEOMETRY	LINESTRING

SQL

- Each column of your SQL table is capable of holding data of a particular data type.

INT	SMALLINT	TINYINT	MEDIUMINT	BIGINT
DECIMAL	FLOAT	BIT	DATE	TIME
DATETIME	TIMESTAMP	CHAR	VARCHAR	BINARY
BLOB	TEXT	ENUM	GEOMETRY	LINESTRING

SQL

- Each column of your SQL table is capable of holding data of a particular data type.

INT	SMALLINT	TINYINT	MEDIUMINT	BIGINT
DECIMAL	FLOAT	BIT	DATE	TIME
DATETIME	TIMESTAMP	CHAR	VARCHAR	BINARY
BLOB	TEXT	ENUM	GEOMETRY	LINESTRING

SQL

- Each column of your SQL table is capable of holding data of a particular data type.

INT	SMALLINT	TINYINT	MEDIUMINT	BIGINT
DECIMAL	FLOAT	BIT	DATE	TIME
DATETIME	TIMESTAMP	CHAR	VARCHAR	BINARY
BLOB	TEXT	ENUM	GEOMETRY	LINESTRING

SQL

- Unlike in C, the CHAR data type in SQL does not refer to a single character. Rather, it is a fixed-length string.
  - In most relational databases, including MySQL, you actually specify the fixed-length as part of the type definition, e.g. `CHAR(10)`.

- A VARCHAR refers to a variable-length string.
  - VARCHARs also require you to specify the **maximum** possible length of a string that could be stored in that column, e.g. `VARCHAR(99)`.

SQL

- One other important consideration when constructing a table in SQL is to choose one column to be your **primary key**.

- Primary keys enable rows of a table to be uniquely and quickly identified.
  - Choosing your primary key appropriately can make subsequent operations on the table much easier.

- It is also possible to establish a joint primary key – a combination of two columns that is always guaranteed to be unique.

SQL

- SQL is a programming language, but its vocabulary is fairly limited.

- We will primarily consider just **four** operations that one may perform on a table.

SQL

- SQL is a programming language, but its vocabulary is fairly limited.

- We will primarily consider just **four** operations that one may perform on a table.

`INSERT`

SQL

- SQL is a programming language, but its vocabulary is fairly limited.

- We will primarily consider just **four** operations that one may perform on a table.

```
INSERT
SELECT
```

SQL

- SQL is a programming language, but its vocabulary is fairly limited.

- We will primarily consider just **four** operations that one may perform on a table.

```
INSERT
SELECT
UPDATE
```

SQL

- SQL is a programming language, but its vocabulary is fairly limited.

- We will primarily consider just **four** operations that one may perform on a table.

```
INSERT
SELECT
UPDATE
DELETE
```

SQL

users

idnum	username	password	fullname
10	jerry	fus!ll!	Jerry Seinfeld
11	gcostanza	b0sc0	George Costanza

moms

username	mother
jerry	Helen Seinfeld
gcostanza	Estelle Costanza

SQL

- **`INSERT`**
  - Add information to a table.

SQL

- **INSERT**
  - Add information to a table.

```
INSERT INTO
<table>
(<columns>)
VALUES
(<values>)
```

SQL

- **INSERT**
  - Add information to a table.

```
INSERT INTO
users
(username, password, fullname)
VALUES
('newman', 'USMAIL', 'Newman')
```

SQL

users

idnum	username	password	fullname
10	jerry	fus!ll!	Jerry Seinfeld
11	gcostanza	b0sc0	George Costanza

moms

username	mother
jerry	Helen Seinfeld
gcostanza	Estelle Costanza

SQL

users

idnum	username	password	fullname
10	jerry	fus!ll!	Jerry Seinfeld
11	gcostanza	b0sc0	George Costanza
12	newman	USMAIL	Newman

moms

username	mother
jerry	Helen Seinfeld
gcostanza	Estelle Costanza

SQL

users

idnum	username	password	fullname
10	jerry	fus!ll!	Jerry Seinfeld
11	gcostanza	b0sc0	George Costanza
12	newman	USMAIL	Newman

moms

username	mother
jerry	Helen Seinfeld
gcostanza	Estelle Costanza

SQL

- When defining the column that ultimately ends up being your table's primary key, it's usually a good idea to have that column be an integer.

- Moreover, so as to eliminate the situation where you may accidentally forget to specify a real value for the primary key column, you can configure that column to **autoincrement**, so it will pre-populate that column for you automatically when rows are added to the table.

SQL

users

idnum	username	password	fullname
10	jerry	fus!ll!	Jerry Seinfeld
11	gcostanza	b0sc0	George Costanza
12	newman	USMAIL	Newman

moms

username	mother
jerry	Helen Seinfeld
gcostanza	Estelle Costanza

SQL

- **INSERT**
  - Add information to a table.

```
INSERT INTO
moms
(username, mother)
VALUES
('kramer', 'Babs Kramer')
```

SQL

users

idnum	username	password	fullname
10	jerry	fus!ll!	Jerry Seinfeld
11	gcostanza	b0sc0	George Costanza
12	newman	USMAIL	Newman

moms

username	mother
jerry	Helen Seinfeld
gcostanza	Estelle Costanza

SQL

users

idnum	username	password	fullname
10	jerry	fus!ll!	Jerry Seinfeld
11	gcostanza	b0sc0	George Costanza
12	newman	USMAIL	Newman

moms

username	mother
jerry	Helen Seinfeld
gcostanza	Estelle Costanza
kramer	Babs Kramer

SQL

- **SELECT**
  - Extract information from a table.

SQL

- **SELECT**
  - Extract information from a table.

```
SELECT
<columns>
FROM
<table>
WHERE
<condition>
ORDER BY
<column>
```

SQL

- **SELECT**
  - Extract information from a table.

```
SELECT
<columns>
FROM
<table>
WHERE
<predicate>
ORDER BY
<column>
```

SQL

- **SELECT**
  - Extract information from a table.

```
SELECT
idnum, fullname
FROM
users
```

SQL

users

idnum	username	password	fullname
10	jerry	fus!ll!	Jerry Seinfeld
11	gcostanza	b0sc0	George Costanza
12	newman	USMAIL	Newman

moms

username	mother
jerry	Helen Seinfeld
gcostanza	Estelle Costanza
kramer	Babs Kramer

SQL

users

idnum	username	password	fullname
10	jerry	fus!ll!	Jerry Seinfeld
11	gcostanza	b0sc0	George Costanza
12	newman	USMAIL	Newman

moms

username	mother
jerry	Helen Seinfeld
gcostanza	Estelle Costanza
kramer	Babs Kramer

SQL

- **SELECT**
  - Extract information from a table.

```
SELECT
password
FROM
users
WHERE
idnum < 12
```

SQL

users

idnum	username	password	fullname
10	jerry	fus!ll!	Jerry Seinfeld
11	gcostanza	b0sc0	George Costanza
12	newman	USMAIL	Newman

moms

username	mother
jerry	Helen Seinfeld
gcostanza	Estelle Costanza
kramer	Babs Kramer

SQL

users

idnum	username	password	fullname
10	jerry	fus!ll!	Jerry Seinfeld
11	gcostanza	b0sc0	George Costanza
12	newman	USMAIL	Newman

moms

username	mother
jerry	Helen Seinfeld
gcostanza	Estelle Costanza
kramer	Babs Kramer

SQL

- **SELECT**
  - Extract information from a table.

```
SELECT
*
FROM
moms
WHERE
username = ‘jerry’
```

SQL

users

idnum	username	password	fullname
10	jerry	fus!ll!	Jerry Seinfeld
11	gcostanza	b0sc0	George Costanza
12	newman	USMAIL	Newman

moms

username	mother
jerry	Helen Seinfeld
gcostanza	Estelle Costanza
kramer	Babs Kramer

SQL

users

idnum	username	password	fullname
10	jerry	fus!ll!	Jerry Seinfeld
11	gcostanza	b0sc0	George Costanza
12	newman	USMAIL	Newman

moms

username	mother
jerry	Helen Seinfeld
gcostanza	Estelle Costanza
kramer	Babs Kramer

SQL

- Databases empower us to organize information into tables efficiently.
  - We don’t always need to store every possible relevant piece of information in the same table, but can use relationships across the tables to let us pull information from where we need it.

SQL

users

idnum	username	password	fullname
10	jerry	fus!ll!	Jerry Seinfeld
11	gcostanza	b0sc0	George Costanza
12	newman	USMAIL	Newman

moms

username	mother
jerry	Helen Seinfeld
gcostanza	Estelle Costanza
kramer	Babs Kramer

SQL

- Databases empower us to organize information into tables efficiently.
  - We don’t always need to store every possible relevant piece of information in the same table, but can use relationships across the tables to let us pull information from where we need it.

- What if we now find ourselves in a situation where we need to get a user’s full name (from the *users* table) and their mother’s name (from the *mother* table).

SQL

- **`SELECT (JOIN)`**
  - Extract information from multiple tables.

SQL

- **SELECT (JOIN)**
  - Extract information from multiple tables.

```
SELECT
<columns>
FROM
<table1>
JOIN
<table2>
ON
<predicate>
```

SQL

- **SELECT (JOIN)**
  - Extract information from multiple tables.

```
SELECT
users.fullname, moms.mother
FROM
users
JOIN
moms
ON
users.username = moms.username
```

SQL

- **SELECT (JOIN)**
  - Extract information from multiple tables.

```
SELECT
users.fullname, moms.mother
FROM
users
JOIN
moms
ON
users.username = moms.username
```

SQL

users

idnum	username	password	fullname
10	jerry	fus!ll!	Jerry Seinfeld
11	gcostanza	b0sc0	George Costanza
12	newman	USMAIL	Newman

moms

username	mother
jerry	Helen Seinfeld
gcostanza	Estelle Costanza
kramer	Babs Kramer

SQL

users

idnum	username	password	fullname
10	jerry	fus!ll!	Jerry Seinfeld
11	gcostanza	b0sc0	George Costanza
12	newman	USMAIL	Newman

moms

username	mother
jerry	Helen Seinfeld
gcostanza	Estelle Costanza
kramer	Babs Kramer

SQL

users

idnum	username	password	fullname
10	jerry	fus!ll!	Jerry Seinfeld
11	gcostanza	b0sc0	George Costanza
12	newman	USMAIL	Newman

moms

username	mother
jerry	Helen Seinfeld
gcostanza	Estelle Costanza
kramer	Babs Kramer

SQL

users & moms

users.idnum	users.username moms.username	users.password	users.fullname	moms.mother
10	jerry	fus!ll!	Jerry Seinfeld	Helen Seinfeld
11	gcostanza	b0sc0	George Costanza	Estelle Costanza

SQL

users & moms

users.idnum	users.username moms.username	users.password	users.fullname	moms.mother
10	jerry	fus!ll!	Jerry Seinfeld	Helen Seinfeld
11	gcostanza	b0sc0	George Costanza	Estelle Costanza

SQL

- **UPDATE**
  - Modify information in a table.

SQL

- **UPDATE**
  - Modify information in a table.

```
UPDATE
<table>
SET
<column> = <value>
WHERE
<predicate>
```

SQL

- **UPDATE**
  - Modify information in a table.

```
UPDATE
users
SET
password = ‘yadayada’
WHERE
idnum = 10
```

SQL

users

idnum	username	password	fullname
10	jerry	fus!ll!	Jerry Seinfeld
11	gcostanza	b0sc0	George Costanza
12	newman	USMAIL	Newman

moms

username	mother
jerry	Helen Seinfeld
gcostanza	Estelle Costanza
kramer	Babs Kramer

SQL

users

idnum	username	password	fullname
10	jerry	yadayada	Jerry Seinfeld
11	gcostanza	b0sc0	George Costanza
12	newman	USMAIL	Newman

moms

username	mother
jerry	Helen Seinfeld
gcostanza	Estelle Costanza
kramer	Babs Kramer

SQL

- **DELETE**
  - Remove information from a table.

SQL

- **DELETE**
  - Remove information from a table.

```
DELETE FROM
<table>
WHERE
<predicate>
```

SQL

- **DELETE**
  - Remove information from a table.

```
DELETE FROM
users
WHERE
username = ‘newman’
```

SQL

users

idnum	username	password	fullname
10	jerry	yadayada	Jerry Seinfeld
11	gcostanza	b0sc0	George Costanza
12	newman	USMAIL	Newman

moms

username	mother
jerry	Helen Seinfeld
gcostanza	Estelle Costanza
kramer	Babs Kramer

SQL

users

idnum	username	password	fullname
10	jerry	yadayada	Jerry Seinfeld
11	gcostanza	b0sc0	George Costanza

moms

username	mother
jerry	Helen Seinfeld
gcostanza	Estelle Costanza
kramer	Babs Kramer

SQL

- All of these operations are pretty easy to do in the graphical interface of phpMyAdmin.

- We want a way to do this programmatically, not just typing SQL commands into the “SQL” tab of phpMyAdmin.

- Fortunately, SQL integrates with PHP very well, by way of functions like `query()`.

SQL

- After you've connected to your database with PHP (using a process called MySQLi or PDO), you can make pass query strings as arguments to functions in PHP and store the result set in an associative array.

SQL

- After you've connected to your database with PHP (using a process called MySQLi or PDO), you can make pass query strings as arguments to functions in PHP and store the result set in an associative array.

```
$results = query("SELECT fullname FROM users WHERE idnum = 10");
```

SQL

- After you've connected to your database with PHP (using a process called MySQLi or PDO), you can make pass query strings as arguments to functions in PHP and store the result set in an associative array.

```
$results = query("SELECT fullname FROM users WHERE idnum = 10");
```

SQL

- After you've connected to your database with PHP (using a process called MySQLi or PDO), you can make pass query strings as arguments to functions in PHP and store the result set in an associative array.

```
$results = query("SELECT fullname FROM users WHERE idnum = 10");
```

SQL

- After you've connected to your database with PHP (using a process called MySQLi or PDO), you can make pass query strings as arguments to functions in PHP and store the result set in an associative array.

```
$results = query("SELECT fullname FROM users WHERE idnum = 10");

print("Thanks for logging in, {$results['fullname']}!");
```

SQL

- After you've connected to your database with PHP (using a process called MySQLi or PDO), you can make pass query strings as arguments to functions in PHP and store the result set in an associative array.

```
$results = query("SELECT fullname FROM users WHERE idnum = ?",
                 $_SESSION["id"]);

print("Thanks for logging in, {$results['fullname']}!");
```

SQL

- After you've connected to your database with PHP (using a process called MySQLi or PDO), you can make pass query strings as arguments to functions in PHP and store the result set in an associative array.

- It's also possible your result set might consist of multiple rows, in which case the result set would be an array of associative arrays, so just need to iterate through it!

SQL

```
<p>The moms of TV’s Seinfeld:</p>
<table>

<?php
    $results = query(“SELECT mothers FROM moms”);

    if($results !== false)
    {
        foreach($results as $result)
        {
            print(“<tr><td>” . $result[‘mothers’] . “</td></tr>”);
        }
    }
?>

</table>
```

SQL

```
<p>The moms of TV’s Seinfeld:</p>
<table>

<?php
    $results = query(“SELECT mothers FROM moms”);

    if($results !== false)
    {
        foreach($results as $result)
        {
            print(“<tr><td>” . $result[‘mothers’] . “</td></tr>”);
        }
    }
?>

</table>
```

SQL

```
<p>The moms of TV's Seinfeld:</p>
<table>

<?php
    $results = query("SELECT mothers FROM moms");

    if($results !== false)
    {
        foreach($results as $result)
        {
            print("<tr><td>" . $result['mothers'] . "</td></tr>");
        }
    }
?>

</table>
```

SQL

```
<p>The moms of TV’s Seinfeld:</p>
<table>

<?php
    $results = query(“SELECT mothers FROM moms”);

    if($results !== false)
    {
        foreach($results as $result)
        {
            print(“<tr><td>” . $result[‘mothers’] . “</td></tr>”);
        }
    }
?>

</table>
```