In order to connect an HTML page to a PHPMyAdmin database, you will need to use a scripting language such as PHP to send and receive data from the database. Here is an example of how to connect to a PHPMyAdmin database using PHP:

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

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

In this example, the PHP script connects to a MySQL database with the following details:

- `$servername` is the server name or IP address of the database. This is typically "localhost" if the database is running on the same server as the website.
- `$username` is the username used to connect to the database.
- `$password` is the password associated with the above username.
- `$dbname` is the name of the database you want to connect to.

You will need to replace these placeholder values with the appropriate details for your database and database user.

Once you have a successful connection you can use the connection to do database operations like selecting, inserting, and updating data. You can use SQL (Structured Query Language) to interact with the database .

It is a good practice to handle errors and to close the database after you finish operations.

SQL:

SQL, or Structured Query Language, is a programming language used to manage and manipulate relational databases. There are several common SQL commands that you can use to interact with a database:

1. SELECT - This command is used to retrieve data from a database table. For example, the following SQL query would retrieve all the rows from a table named "users":

```
SELECT * FROM users;
```

2. INSERT - This command is used to insert new data into a database table. For example, the following SQL query would insert a new row into a table named "users" with a name of "John Doe" and an age of 30:

```
INSERT INTO users (name, age) VALUES ('John Doe', 30);
```

3. UPDATE - This command is used to modify existing data in a database table. For example, the following SQL query would change the age of a user with the name "John Doe" to 35:

```
UPDATE users SET age = 35 WHERE name = 'John Doe';
```

4. DELETE - This command is used to delete data from a database table. For example, the following SQL query would delete the user with the name "John Doe" from the "users" table:

```
DELETE FROM users WHERE name = 'John Doe';
```

5. CREATE - This command is used to create a new table. For example, the following SQL query would create a table named "users" with columns for "id", "name", and "age":

```
CREATE TABLE users (id INT, name VARCHAR(255), age INT);
```

6. DROP - This command is used to delete an entire table. For example, the following SQL query would delete the table named "users":

```
DROP TABLE users;
```

These are the most common SQL commands but there are other commands like ALTER, INDEX etc which serves different purposes.

You can use the mysqli_query() function in PHP to execute a SQL query and return the results. Once you have established a connection to a database using PHP, you can use the connection object to execute a query and retrieve the results like this:

```
$result = mysqli_query($conn, $sql);
```

Where $conn is the connection object returned by the mysqli_connect() function, and $sql is the SQL query you want to execute.

It is a good practice to use Prepared statements instead of concatenating user input to SQL queries, because it's more secure.

Inserting into phpmyadmin database:
In order to insert and retrieve information from a PHPMyAdmin database, you will need to use a scripting language such as PHP to send and receive data from the database. Here is an example of how to insert and retrieve data from a PHPMyAdmin database using PHP:

```
Inserting Data:

<?php

$servername = "localhost";

$username = "username";

$password = "password";

$dbname = "myDB";


// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);


// Check connection

if ($conn->connect_error) {

    die("Connection failed: " . $conn->connect_error);

}


// Insert data into the database

$name = "John Doe";

$age = 30;

$sql = "INSERT INTO users (name, age) VALUES ('$name', '$age')";


if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";

} else {

    echo "Error: " . $sql . "<br>" . $conn->error;

}



$conn->close();

?>
```

In this example, a PHP script connects to a MySQL database with the appropriate details, then it constructs an SQL query with the INSERT command to insert data into the 'users' table with 2 columns 'name' and 'age' respectively. The $name and $age variables represent the values being inserted. `$conn->query($sql)` is used to execute the SQL command, which returns a Boolean value indicating if the query was successful or not.

Retrieving Data:

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

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Retrieve data from the database
$sql = "SELECT name, age FROM users";
```

```
$result = $conn->query($sql);

if ($result->num_rows > 0) {
   // output data of each row
   while($row = $result->fetch_assoc()) {
       echo "Name: " . $row["name"]. " - Age: " . $row["age"]. "<br>";
   }
} else {
   echo "0 results";
}

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

In this example, a PHP script connects to the MySQL database and retrieves the data from the 'users' table with a SELECT command, it uses a query() function to execute the SQL command, which returns a result object. The result object has a method num_rows to check if there are any rows in the result set, and also a method fetch_assoc() which is used to retrieve the rows one by one. The while loop is used to print the name and age of each row in the result set.

It is important to note that the above examples are for demonstration purposes and may require adjustments to match your database schema and desired functionality. Also, using prepared statement instead of using the query function with concatenated values from user input is more safe.

Input data into textbox:
In order to input data from an HTML textbox into a PHPMyAdmin database, you will need to use a combination of HTML and PHP. Here is an example of how to create an HTML form with a textbox and a submit button, and then use PHP to insert the data into a PHPMyAdmin database:

HTML Form:

```
<form action="insert.php" method="post">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name">
  <br>
  <label for="age">Age:</label>
  <input type="text" id="age" name="age">
  <br><br>
  <input type="submit" value="Submit">
</form>
```

This form creates an HTML page that has two text boxes, one for the name and another for the age, and a submit button. The form's action attribute is set to "insert.php", which means that when the form is submitted, the browser will send a request to that URL, and the PHP script at that URL will handle the form data. The method attribute is set to "post", meaning that the data is sent in the body of the HTTP request and will not be visible in the URL. The input tags have `name` attribute, with this attribute when the form is submitted the data is sent as name-value pairs.

PHP Script (insert.php):

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

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Retrieve data from the form
$name = $_POST["name"];
$age = $_POST["age"];

// Insert data into the database
$sql = "INSERT INTO users (name, age) VALUES ('$name', '$age')";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

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

In this example, the PHP script connects to a MySQL database and retrieves the data from the form using the `$_POST` superglobal. This superglobal is an associative array that contains the data sent in the HTTP request body, so the values sent by the form will be stored in the `$_POST` array with keys that match the `name` attributes of the form's input elements. Then the script construct an SQL query with the INSERT command to insert data into the 'users' table with 2 columns 'name' and 'age' respectively. The $name and $age variables represent the values being inserted. `$conn->query($sql)` is used to execute the SQL command, which returns a Boolean value indicating if the query was successful or not.

It is important to note that this is just one way to insert data from an HTML form into a PHPMyAdmin database, and you may need to adjust it to match your specific requirements. In addition, for more secure way to handle user input, you should use prepared statements.