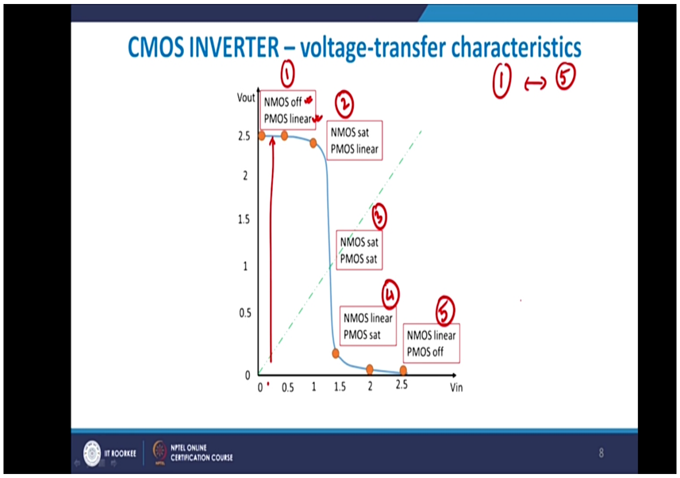
June 11 report

Date:	11/06/2020	Name:	Ankitha c c
Course:	Vlsi	USN:	4al16ec004
Topic:	MOS transistor basics-II and III Source: NPTEL	Semester & Section:	8th & "A" section
Github Repository:	ankitha-c-c		

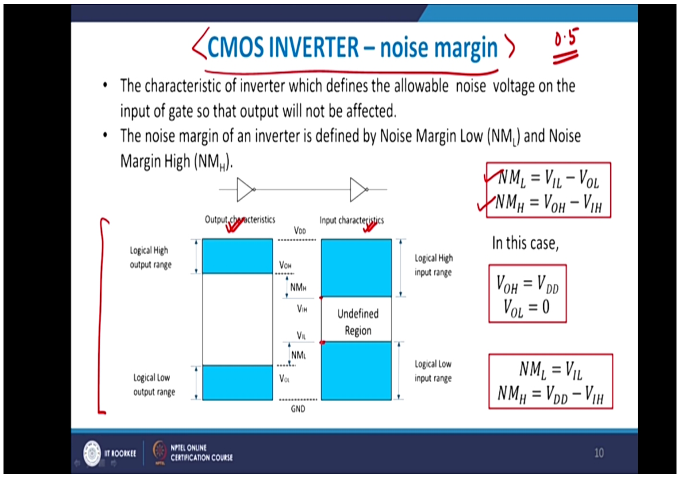
FORENOON SESSION DETAILS
Image of session   CMOS Digital VLSI Design Lecture-06 CMOS INVERTER BASICS-II   CMOS Digital VLSI Design Lecture-07 CMOS INVERTER BASICS-III

Report – Report can be typed or hand written for up to two pages.

1.MOS transistor basics-II and III



CMOS INVERTER – voltage-transfer characteristics



CMOS INVERTER – noise margin

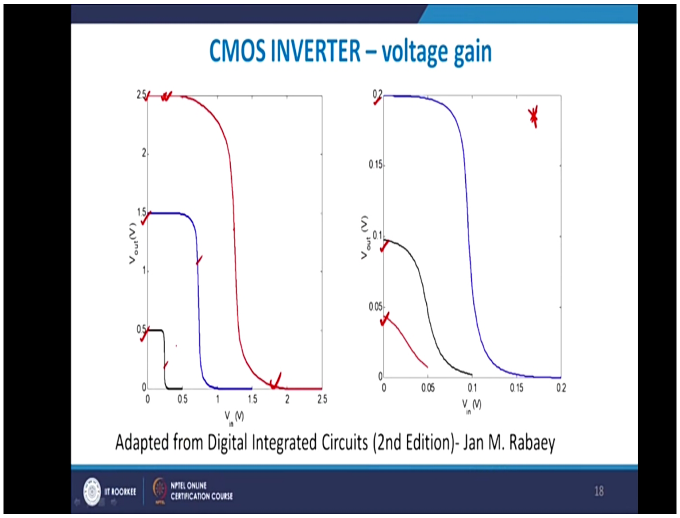
- The characteristic of inverter which defines the allowable noise voltage on the input of gate so that output will not be affected.
- The noise margin of an inverter is defined by Noise Margin Low (NM_L) and Noise Margin High (NM_H).

$$NM_L = V_{IL} - V_{OL}$$
$$NM_H = V_{OH} - V_{IH}$$

In this case,

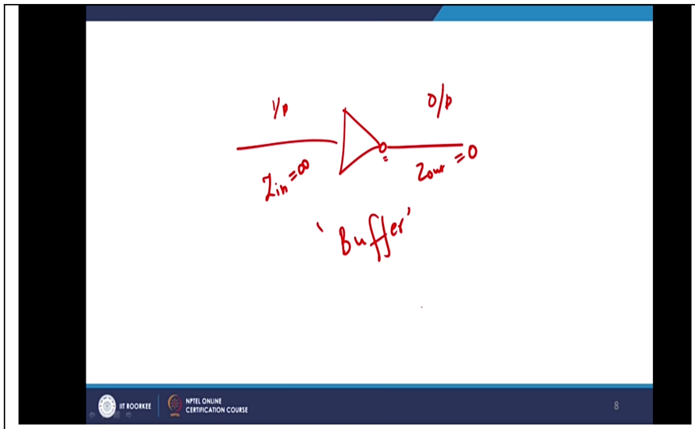
$$V_{OH} = V_{DD}$$
$$V_{OL} = 0$$

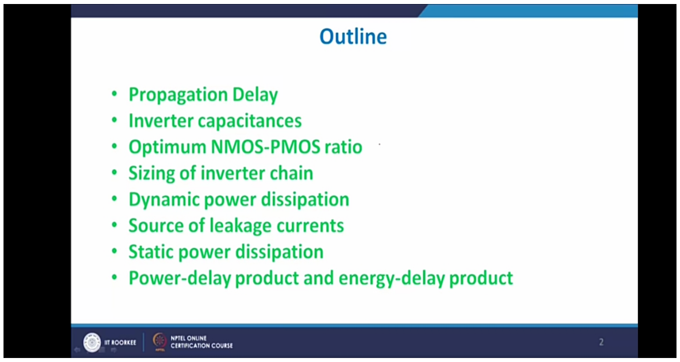
$$NM_L = V_{IL}$$
$$NM_H = V_{DD} - V_{IH}$$



CMOS INVERTER – voltage gain

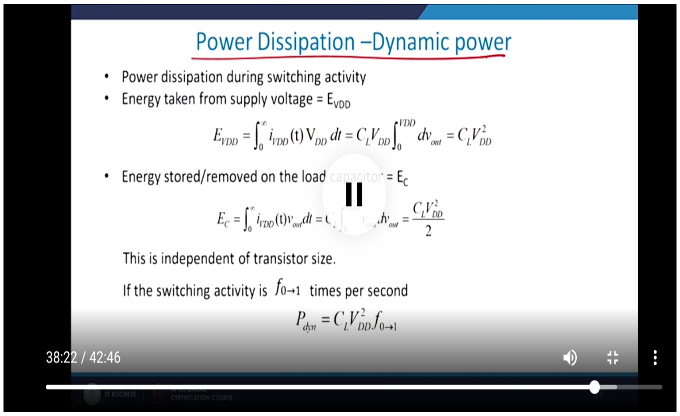
Adapted from Digital Integrated Circuits (2nd Edition)- Jan M. Rabaey





Outline

- Propagation Delay
- Inverter capacitances
- Optimum NMOS-PMOS ratio
- Sizing of inverter chain
- Dynamic power dissipation
- Source of leakage currents
- Static power dissipation
- Power-delay product and energy-delay product



Power Dissipation –Dynamic power

- Power dissipation during switching activity
- Energy taken from supply voltage = E_{VDD}

$$E_{VDD} = \int_0^\infty i_{VDD}(t) V_{DD}\, dt = C_L V_{DD} \int_0^{VDD} dv_{out} = C_L V_{DD}^2$$

- Energy stored/removed on the load capacitor = E_C

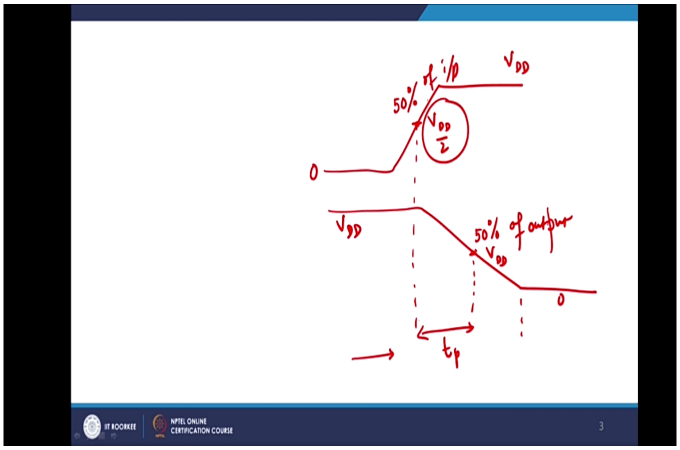
$$E_C = \int_0^\infty i_{VDD}(t) v_{out} dt = C_L \int_0^{VDD} v_{out} dv_{out} = \frac{C_L V_{DD}^2}{2}$$

This is independent of transistor size.

If the switching activity is $f_{0\to1}$ times per second

$$P_{dyn} = C_L V_{DD}^2 f_{0\to1}$$

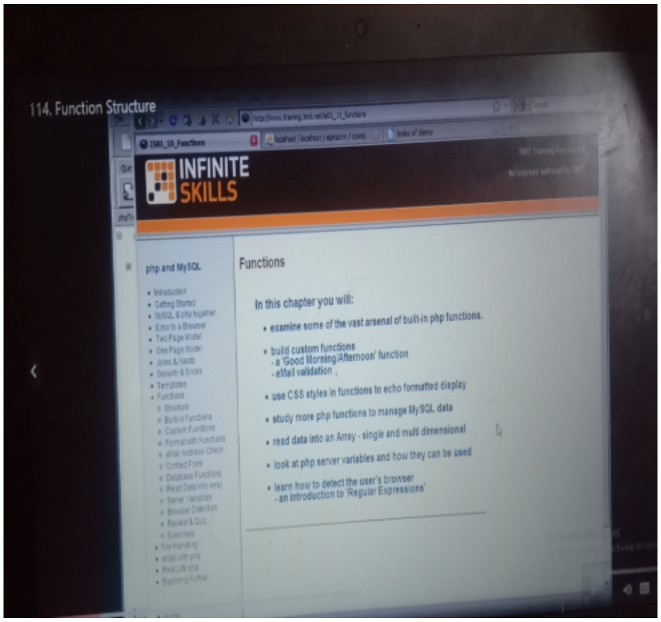
38:22 / 42:46



Date:	11/06/2020	Name:	Ankitha c c
Course:	Mysql	USN:	4al16ec004
Topic:		Semester & Section:	8th & a section

AFTERNOON SESSION DETAILS

Image of session

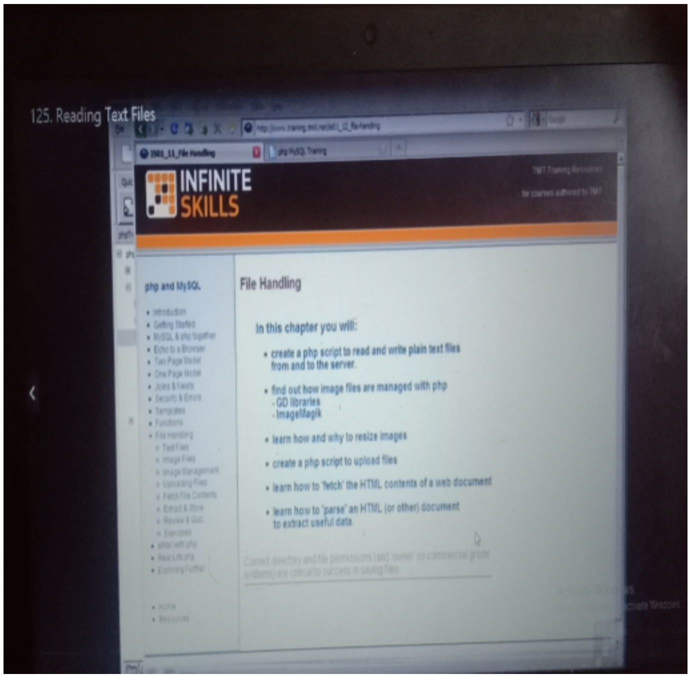


114. Function Structure

Functions

In this chapter you will:

- examine some of the vast arsenal of built-in php functions.
- build custom functions
  - a 'Good Morning/Afternoon' function
  - eMail validation
- use CSS styles in functions to echo formatted display
- study more php functions to manage MySQL data
- read data into an Array - single and multi dimensional
- look at php server variables and how they can be used
- learn how to detect the user's browser
  - an introduction to 'Regular Expressions'



125. Reading Text Files

File Handling

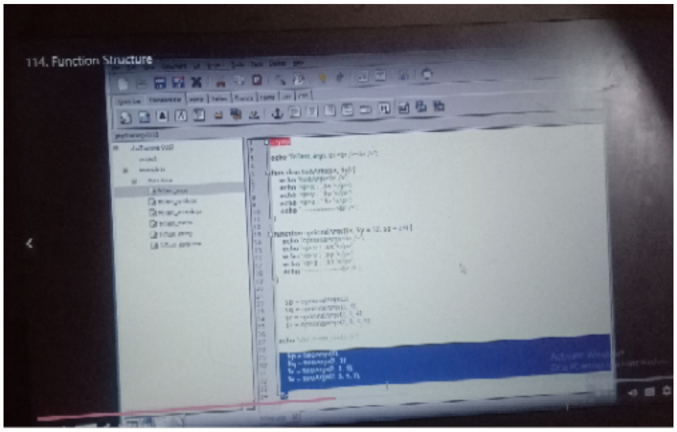
In this chapter you will:

- create a php script to read and write plain text files from and to the server.
- find out how image files are managed with php
  - GD libraries
  - ImageMagik
- learn how and why to resize images
- create a php script to upload files
- learn how to 'fetch' the HTML contents of a web document
- learn how to 'parse' an HTML (or other) document to extract useful data.

Correct directory and file permissions (and 'owner' on commercial grade systems) are critical to success in saving files

Report – Report can be typed or hand written for up to two pages.

1. Php functions



PHP MySQLi Functions

PHP MySQLi Introduction

The MySQLi functions allows you to access MySQL database servers.

Note: The MySQLi extension is designed to work with MySQL version 4.1.13 or newer.

Installation / Runtime Configuration

For the MySQLi functions to be available, you must compile PHP with support for the MySQLi extension.

The MySQLi extension was introduced with PHP version 5.0.0. The MySQL Native Driver was included in PHP version 5.3.0.

For installation details, go to: http://php.net/manual/en/mysqli.installation.php

For runtime configuration details, go to: http://php.net/manual/en/mysqli.configuration.php

PHP MySQLi Functions

Function Description

affected_rows()   Returns the number of affected rows in the previous MySQL operation

autocommit()	Turns on or off auto-committing database modifications
begin_transaction()	Starts a transaction
change_user()	Changes the user of the specified database connection
character_set_name()	Returns the default character set for the database connection
close()	Closes a previously opened database connection
commit()	Commits the current transaction
connect()	Opens a new connection to the MySQL server
connect_errno()	Returns the error code from the last connection error
connect_error()	Returns the error description from the last connection error
data_seek()	Adjusts the result pointer to an arbitrary row in the result-set
debug()	Performs debugging operations
dump_debug_info()	Dumps debugging info into the log
errno()	Returns the last error code for the most recent function call
error()	Returns the last error description for the most recent function call
error_list()	Returns a list of errors for the most recent function call
fetch_all()	Fetches all result rows as an associative array, a numeric array, or both
fetch_array()	Fetches a result row as an associative, a numeric array, or both
fetch_assoc()	Fetches a result row as an associative array
fetch_field()	Returns the next field in the result-set, as an object
fetch_field_direct()	Returns meta-data for a single field in the result-set, as an object
fetch_fields()	Returns an array of objects that represent the fields in a result-set
fetch_lengths()	Returns the lengths of the columns of the current row in the result-set
fetch_object()	Returns the current row of a result-set, as an object
fetch_row()	Fetches one row from a result-set and returns it as an enumerated array
field_count()	Returns the number of columns for the most recent query
field_seek()	Sets the field cursor to the given field offset
get_charset()	Returns a character set object
get_client_info()	Returns the MySQL client library version
get_client_stats()	Returns statistics about client per-process

get_client_version()	Returns the MySQL client library version as an integer
get_connection_stats()	Returns statistics about the client connection
get_host_info()	Returns the MySQL server hostname and the connection type
get_proto_info()	Returns the MySQL protocol version
get_server_info()	Returns the MySQL server version
get_server_version()	Returns the MySQL server version as an integer
info()	Returns information about the last executed query
init()	Initializes MySQLi and returns a resource for use with real_connect()
insert_id()	Returns the auto-generated id from the last query
kill()	Asks the server to kill a MySQL thread
more_results()	Checks if there are more results from a multi query
multi_query()	Performs one or more queries on the database

PHP mysqli_connect function

The PHP mysql connect function is used to connect to a MySQL database server.

It has the following syntax.

```
<?php;

$db_handle = mysqli_connect($db_server_name, $db_user_name, $db_password);

?>
```

HERE,

“$db_handle” is the database connection resource variable.

“mysqli_connect(…)” is the function for php database connection

“$server_name” is the name or IP address of the server hosting MySQL server.

“$user_name” is a valid user name in MySQL server.

“$password” is a valid password associated with a user name in MySQL server.

PHP mysqli_query function

The mysqli_query function is used to execute SQL queries.

The function can be used to execute the following query types;

Insert

Select

Update

delete

It has the following syntax.

```
<?php
mysqli_query($db_handle,$query) ;
?>
```

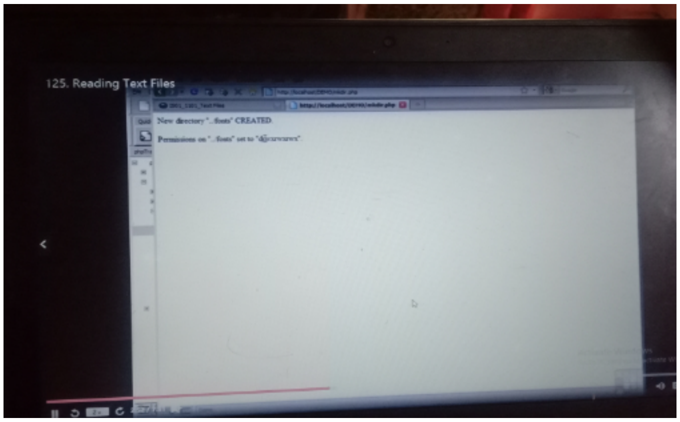
HERE,

"mysqli_query(…)" is the function that executes the SQL queries.

"$query" is the SQL query to be executed

"$link_identifier" is optional, it can be used to pass in the server connection link

2.Using external files and images



```
New directory "../fonts" CREATED.
Permissions on "../fonts" set to "drwxrwxrwx".
```

Introduction

A Binary Large Object (BLOB) is a MySQL data type that can store binary data such as images, multimedia, and PDF files.

When creating applications that require a tightly-coupled database where images should be in sync with related data (for example, an employee portal, a student database, or a financial application), you might find it convenient to store images such as students' passport photos and signatures in a MySQL database alongside other related information.

This is where the MySQL BLOB data type comes in. This programming approach eliminates the need for creating a separate file system for storing images. The scheme also centralizes the database, making it more portable and secure because the data is isolated from the file system. Creating backups is also more seamless since you can create a single MySQL dump file that contains all your data.

Retrieving data is faster, and when creating records you can be sure that data validation rules and referential integrity are maintained especially when using MySQL transactions.

In this tutorial, you will use the MySQL BLOB data type to store images with PHP on Ubuntu 18.04.

Prerequisites

To follow along with this guide, you will need the following:

An Ubuntu 18.04 server configured using the Initial Server Setup with Ubuntu 18.04 and a non-root user with sudo privileges.

Apache, MySQL, and PHP set up by following the guide on How To Install Linux, Apache, MySQL, PHP (LAMP) stack on Ubuntu 18.04. For this tutorial, it isn’t necessary to create virtual hosts, so you can skip Step 4.

Step 1 — Creating a Database

You’ll start off by creating a sample database for your project. To do this, SSH in to your server and then run the following command to log in to your MySQL server as root:

```
sudo mysql -u root -p
```

Enter the root password of your MySQL database and hit ENTER to continue.

Then, run the following command to create a database. In this tutorial we’ll name it test_company:

```
CREATE DATABASE test_company;
```

Once the database is created, you will see the following output:

Output

```
Query OK, 1 row affected (0.01 sec)
```

Next, create a test_user account on the MySQL server and remember to replace PASSWORD with a strong password:

```
CREATE USER ‘test_user’@‘localhost’ IDENTIFIED BY ‘PASSWORD’;
```

You’ll see the following output:

Output

```
Query OK, 0 rows affected (0.01 sec)
```

To grant test_user full privileges on the test_company database, run:

```
GRANT ALL PRIVILEGES ON test_company.* TO ‘test_user’@‘localhost’;
```

Make sure you get the following output:

Output

```
Query OK, 0 rows affected (0.01 sec)
```

Finally, flush the privileges table in order for MySQL to reload the permissions:

```
FLUSH PRIVILEGES;
```

Ensure you see the following output:

Output

```
Query OK, 0 rows affected (0.01 sec)
```

Now that the test_company database and test_user are ready, you’ll move on to creating a products table for storing sample products.

You'll use this table later to insert and retrieve records to demonstrate how MySQL BLOB works.

Log out from the MySQL server:

```
QUIT;
```

Then, log back in again with the credentials of the test_user that you created:

```
mysql -u test_user -p
```

When prompted, enter the password for the test_user and hit ENTER to continue. Next, switch to the test_company database by typing the following:

```
USE test_company;
```

Once the test_company database is selected, MySQL will display:

Output

```
Database changed
```

Next, create a products table by running:

```
CREATE TABLE `products` (product_id BIGINT PRIMARY KEY AUTO_INCREMENT, product_name VARCHAR(50), price DOUBLE, product_image BLOB) ENGINE = InnoDB;
```

This command creates a table named products. The table has four columns:

product_id: This column uses a BIGINT data type in order to accommodate a large list of products up to a maximum of $2^{63}-1$ items. You've marked the column as PRIMARY KEY to uniquely identify products. In order for MySQL to handle the generation of new identifiers for inserted columns, you have used the keyword AUTO_INCREMENT.

product_name: This column holds the names of the products. You've used the VARCHAR data type since this field will generally handle alphanumerics up to a maximum of 50 characters—the limit of 50 is just a hypothetical value used for the purpose of this tutorial.

price: For demonstration purposes, your products table contains the price column to store the retail price of products. Since some products may have floating values (for example, 23.69, 45.36, 102.99), you've used the DOUBLE data type.

product_image: This column uses a BLOB data type to store the actual binary data of the products' images.

You've used the InnoDB storage ENGINE for the table to support a wide range of features including MySQL transactions. After executing this for creating the products table, you'll see the following output:

Output

```
Query OK, 0 rows affected (0.03 sec)
```

Log out from your MySQL server:

```
QUIT;
```

You will get the following output

Output

```
Bye
```

The products table is now ready to store some records including products' images and you'll populate it with some products in the next step.

Step 2 – Creating PHP Scripts for Connecting and Populating the Database

In this step, you’ll create a PHP script that will connect to the MySQL database that you created in Step 1. The script will prepare three sample products and insert them into the products table.

To create the PHP code, open a new file with your text editor:

```
sudo nano /var/www/html/config.php
```

Then, enter the following information into the file and replace PASSWORD with the test_user password that you created in Step 1:

/var/www/html/config.php

```
<?php

define(‘DB_NAME’, ‘test_company’);

define(‘DB_USER’, ‘test_user’);

define(‘DB_PASSWORD’, ‘PASSWORD’);

define(‘DB_HOST’, ‘localhost’);

$pdo = new PDO(“mysql:host=” . DB_HOST . “; dbname=” . DB_NAME, DB_USER, DB_PASSWORD);

$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

$pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
```

Save and close the file.

In this file, you’ve used four PHP constants to connect to the MySQL database that you created in Step 1:

DB_NAME : This constant holds the name of the test_company database.

DB_USER : This variable holds the test_user username.

DB_PASSWORD : This constant stores the MySQL PASSWORD of the test_user account.

DB_HOST: This represents the server where the database resides. In this case, you are using the localhost server.

The following line in your file initiates a PHP Data Object (PDO) and connects to the MySQL database:

Step 3 – Displaying Products’ Information From the MySQL Database

With the products’ information and images in the database, you’re now going to code another PHP script that queries and displays the products’ information in an HTML table on your browser.

To create the file, type the following:

```
sudo nano /var/www/html/display_products.php
```

Then, enter the following information into the file:

/var/www/html/display_products.php

```
<html>
```

```
<title>Using BLOB and MySQL</title>
<body>
<?php
require_once 'config.php';
$sql = "SELECT * FROM products";
$stmt = $pdo->prepare($sql);
$stmt->execute();
?>
<table border = '1' align = 'center'> <caption>Products Database</caption>
  <tr>
    <th>Product Id</th>
    <th>Product Name</th>
    <th>Price</th>
    <th>Product Image</th>
  </tr>

<?php
while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
    echo '<tr>';
    echo '<td>' . $row['product_id'] . '</td>';
    echo '<td>' . $row['product_name'] . '</td>';
    echo '<td>' . $row['price'] . '</td>';
    echo '<td>' .
    '<img src = "data:image/png;base64,' . base64_encode($row['product_image']) . '" width = "50px" height = "50px"/>'
    . '</td>';
    echo '</tr>';
}
?>
</table>
```

```
  </body>

</html>
```

Save the changes to the file and close it.

Conclusion

In this guide, you used the MySQL BLOB data type to store and display images with PHP on Ubuntu 18.04. You've also seen the basic advantages of storing images in a database as opposed to storing them in a file system. These include portability, security, and ease of backup. If you are building an application such as a students' portal or employees' database that requires information and related images to be stored together, then this technology can be of great use to you.