**BackEnd Development:**

* First create two new Product Attribute Programmatically Using Data Patch named Color and Size, but in this case Magento 2 default provide this two attribute so no need to create this Attributes.

Steps:

**Step 1:** Create Color and Size Attributes

* Navigate to Stores -> Attributes -> Product in the admin panel.
* Create two attributes: one for color and one for size.
* For each attribute, set the "Catalog Input Type for Store Owner" to "Visual Swatch" and configure other settings as needed.

**Step 2:** Create Configurable Product

* Navigate to Catalog -> Products in the admin panel.
* Click "Add Product" and select "Configurable Product."
* Choose the attribute set you created in Step 1.
* Configure the basic settings and click "Save and Continue."
* Scroll down to the "Configurations" section.
* Under "Create Configurations," choose the color and size attributes.
* Configure each variation with its associated image, price, and quantity.
* Click "Generate Products" and then "Save and Continue."

Step 3: Configure Swatches

* Navigate to Stores -> Attributes -> Product in the admin panel.
* Edit the color attribute and upload swatch images for each option.
* Repeat the process for the size attribute.
* Step 4: Assign Swatches to Configurable Options
* Go back to the configurable product you created.
* In the "Configurations" section, click "Edit" for each variation.
* Assign the swatch images for color and size.

Step 6: Save and Test

* Save the configurable product.
* Clear the cache: bin/magento cache:clean
* Test the product on the storefront. You should see color and size swatches with associated images, prices, and quantities.
* First we need to create remaining extra attributes using in patch file.

Creating a custom attribute patch file in Magento 2 involves several steps. Patches are a way to make changes to the database schema, data, or code in a safe and version-controlled manner.

Step 1: Create the Patch File

Create a new PHP file in your module directory. For example, create a file named AddCustomAttribute.php in app/code/YourVendor/YourModule/Setup/Patch/Data.

**code is here:**

public function apply()

{

/\*\* @var EavSetup $eavSetup \*/

$eavSetup = $this->\_eavSetupFactory->create(['setup' => $this->\_moduleDataSetup]);

$eavSetup->addAttribute(\Magento\Catalog\Model\Product::ENTITY, 'custom\_product\_attribute', [

'type' => 'int',

'backend' => '',

'frontend' => '',

'label' => 'Custom Product Attribute',

'input' => 'select',

'class' => 'custom\_product\_attribute\_class',

'source' => \Magento\Catalog\Model\Product\Attribute\Source\Boolean::class,

'global' => \Magento\Eav\Model\Entity\Attribute\ScopedAttributeInterface::SCOPE\_GLOBAL,

'visible' => true,

'required' => false,

'user\_defined' => false,

**'default' => '',**

'searchable' => false,

'filterable' => false,

'comparable' => false,

'visible\_on\_front' => false,

'used\_in\_product\_listing' => true,

'unique' => false,

]);

}

Note: Use 'default' => '', so that newly created product attribute is set to “default” attribute set.

Step 2: Register the Patch

* + Create or update the registration.php file in your module directory:Need to check in database ‘patch\_list’ table. New row created or not.

1. Step 3: Apply the Patch : Run the following command to apply the patch:  
    bin/magento setup:upgrade , bin/magento setup:static-content:deploy and bin/magento cache:clean

**FrontEnd Development:**

1. Editing the default Magento theme involves creating a custom theme that inherits from the default theme.
2. Create a directory for your custom theme: In the app/design/frontend directory, create a new folder structure for your theme. For example, if your theme is named CustomTheme and it's based on the default Luma theme, your directory structure might look like this:  
   app/design/frontend/YourVendor/ CustomTheme
3. Create a theme.xml file: Inside your custom theme directory, create a theme.xml file with the following content:

<!-- File: app/design/frontend/YourVendor/CustomTheme/theme.xml -->

<theme xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:Config/etc/theme.xsd">

<title>Your Theme</title>

<parent>Magento/luma</parent> <!-- Specify the parent theme, in this case, Luma -->

</theme>

1. Register your theme: Register your theme in the app/design/frontend/YourVendor/CustomTheme/registration.php file:

<?php

\Magento\Framework\Component\ComponentRegistrar::register(

\Magento\Framework\Component\ComponentRegistrar::THEME,

'frontend/YourVendor/CustomTheme',

\_\_DIR\_\_

);

1. Apply your theme: After creating your custom theme, apply it by navigating to the Magento admin panel: Content -> Design -> Configuration. Then, under the 'Default Theme' section, select your custom theme.app/design/frontend/YourVendor/ CustomTheme
2. After making changes, deploy static content:
3. bin/magento setup:static-content:deploy and bin/magento cache:clean
   * Modify the product details page (**catalog\_product\_view.xml** or any other necessary file) to display the " color and Size" attribute under the product name.

**Steps:**

1. Copy the Product View Layout XML: Navigate to the default Luma theme or the theme you are using. The default Luma theme's catalog\_product\_view.xml file is located at:  
   vendor/magento/module-catalog/view/frontend/layout/catalog\_product\_view.xml
2. Copy this file to your custom theme directory. For example:  
   app/design/frontend/YourVendor/YourTheme/Magento\_Catalog/layout/catalog\_product\_view.xml
3. Edit the catalog\_product\_view.xml file: Add the below code.

<?xml version="1.0"?>

<page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:View/Layout/etc/page\_configuration.xsd">

<body>

<move element="product.info.options.wrapper" destination="page.main.title" after="-" />

</body>

</page>

1. Deploy Static Content: After making changes, deploy static content:  
   bin/magento setup:static-content:deploy  
   bin/magento cache:clean

Now, when you view a product page, the "Color" and "Size" attributes should be displayed under the product name.

To display custom attributes on the product details page in Magento 2, you need to make modifications to the layout XML file and possibly the template file for the product view.

Step 1: Determine the Block Name

First, you need to determine the block name that renders the product details. The block responsible for the product details page is typically named product.info.main. In your theme or custom module, you may need to override or extend this block.

Step 2: Create or Modify Layout XML

Create a layout XML file in your theme or module. For example, you might create a file named catalog\_product\_view.xml in app/design/frontend/YourVendor/YourTheme/Magento\_Catalog/layout/.

<!-- File: app/design/frontend/YourVendor/YourTheme/Magento\_Catalog/layout/catalog\_product\_view.xml -->

<?xml version="1.0"?>

<page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:View/Layout/etc/page\_configuration.xsd">

<body>

<referenceBlock name="product.info.main">

<block class="Magento\Catalog\Block\Product\View" name="custom.attribute.block" template="Magento\_Catalog::product/view/custom\_attributes.phtml" />

</referenceBlock>

</body>

</page>

Step 3: Create or Modify Template File

Create a template file for your custom attributes. For example, you might create a file named custom\_attributes.phtml in app/design/frontend/YourVendor/YourTheme/Magento\_Catalog/templates/product/view/.

<!-- File: app/design/frontend/YourVendor/YourTheme/Magento\_Catalog/templates/product/view/custom\_attributes.phtml -->

<?php

$\_product = $block->getProduct();

?>

<div class="custom-attributes">

<h2>Custom Attributes</h2>

<dl>

<dt>Custom Attribute 1:</dt>

<dd><?php echo $block->escapeHtml($\_product->getData('custom\_attribute\_1')); ?></dd>

<dt>Custom Attribute 2:</dt>

<dd><?php echo $block->escapeHtml($\_product->getData('custom\_attribute\_2')); ?></dd>

<!-- Add more attributes as needed -->

</dl>

</div>

Step 4: Deploy Static Content

After making changes, deploy static content:

bin/magento setup:static-content:deploy

Step 5: Clear Cache

Clear the cache:

bin/magento cache:clean

Now, your custom attributes should be displayed on the product details page under the specified block. Adjust the code based on your attribute codes, template structure, and styling needs.

**Testing:**

* Create new product and set the value of newly created product attribute.
* Check that this newly created product display on frontend.
* If everything is ok than perform add to cart and proceed to checkout.