Lab Exercise #3 - K-Means Clustering

Instruction:
Read the instructions carefully and follow each step.

Provide your best answers for all questions.

The name, email, and photo associated with your Google account will be recorded when you upload files and submit this form

* Indicates required question

Email *

Record my email address with my response

Name *

Example: Dela Cruz, Juan C.

Your answer

Student Number *

Example: 24B1234

Your answer

Section *

- ○ 4A
- ○ 4B
- ○ 4C
- ○ 4D
- ○ 4E
- ○ 4F

Image Compression using K-means Clustering

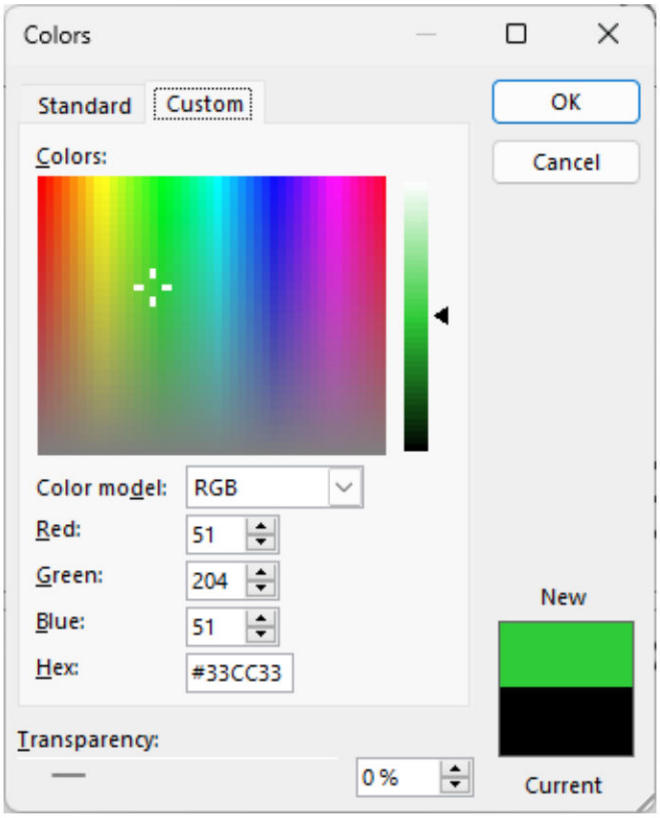
In this lab exercise, we will explore how K-means clustering can be used for image compression, a technique to significantly reduce the file size of an image without losing too much visual quality.

Background

An image is made up of pixels, the smallest units that form the picture, similar to puzzle pieces. *

In a colored image, each pixel has three values representing the intensity of Red, Green, and Blue (RGB), ranging from 0 to 255.

For example, when selecting a custom color in Microsoft Office, you can see the specific RGB values that define that color.



◯ I have read and understood.

A 1280 x 720 image contains 1280 × 720 pixels, with each pixel requiring 24 bits, resulting in about 22 million bits to store the image (e.g., 24 x 1280 x 720). *

◯ I have read and understood.

Compression Concept

To reduce the file size, we aim to reduce the number of bits needed to represent the image. *

Using K-means clustering, we can limit the number of colors in the image.

For instance, instead of using the full range of RGB values, we could cluster the pixels into 30 dominant colors.

This reduces the storage requirement, as we need only 15 bits per pixel (5 bits per color channel) instead of 24.

- [ ] I have read and understood.

By applying K-means clustering, we can reduce the file size of a 1280 x 720 image to about 52% of its original size while preserving most of its visual quality. *

- [ ] I have read and understood.

Objective

In this lab, you will apply K-means clustering to compress an image by reducing the number of colors used. *

You will see how machine learning can efficiently compress image files.

- [ ] I have read and understood.

Step 1: Select an Image

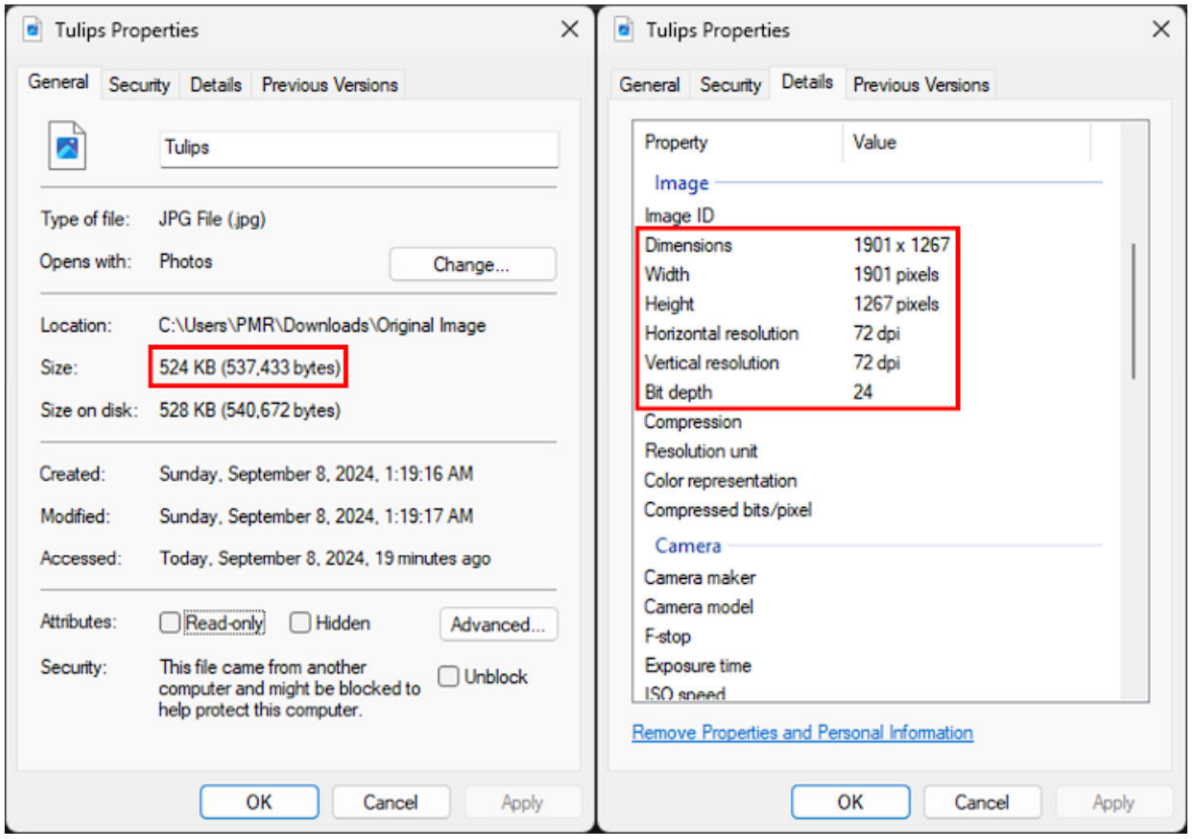
Create a folder named "Original Image" in any directory you prefer. *

- [ ] Done

Download the tulip image from Google Drive and save it in the folder. *

- ◯ Done

Check if the image size and properties match the picture below. *



- ◯ Done

Open Orange and create a new workflow. *

Name it in this format: **[Last_Name]_[Section]_Lab_Exercise_3** (e.g., Rey_4A_Lab_Exercise_3.ows).

- ◯ Done

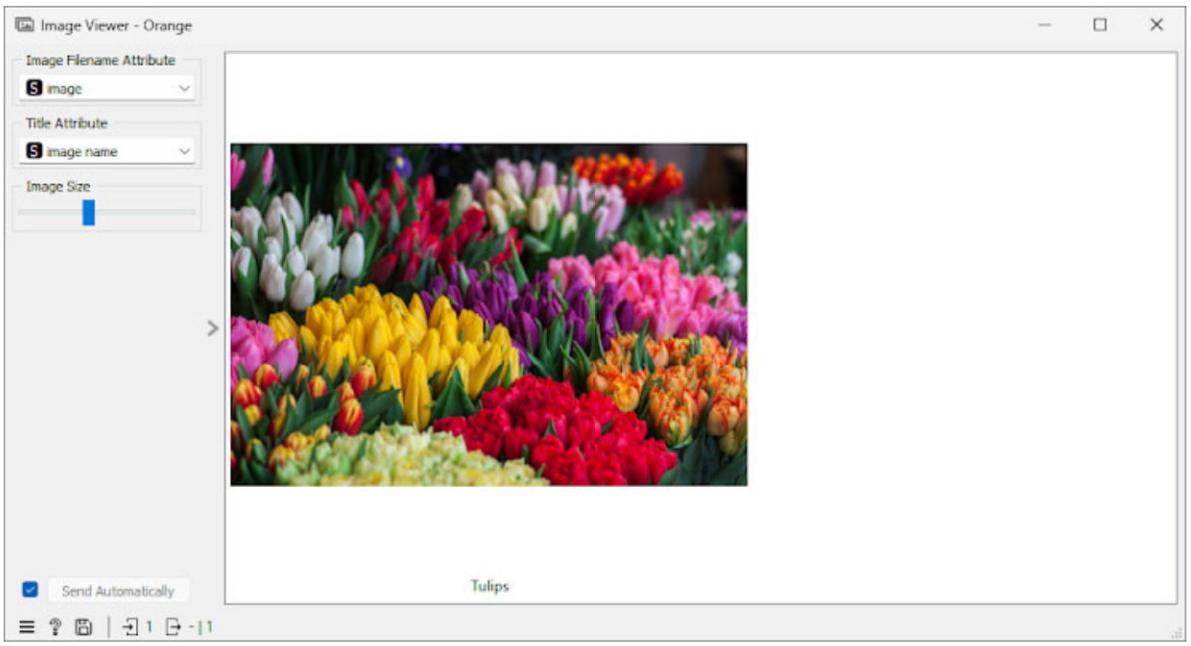
Drag and drop an "Import Image" widget. *

Then, select the "Original Image" folder to load the image.

- ◯ Done

Use the "Image Viewer" widget connected to "Import Image" to view the image. *

You should see something like this:



- ◯ Done

Question: If the image width is 1901 pixels and the height is 1267 pixels, what is the total number of pixels in the image? *

Your answer

Download a script named "get_pixels_rgb.py" from Google Drive and save it in the "Original Image" folder. *

- ◯ Done

Drag and drop a "Python Script" widget and connect it to "Import Image". *

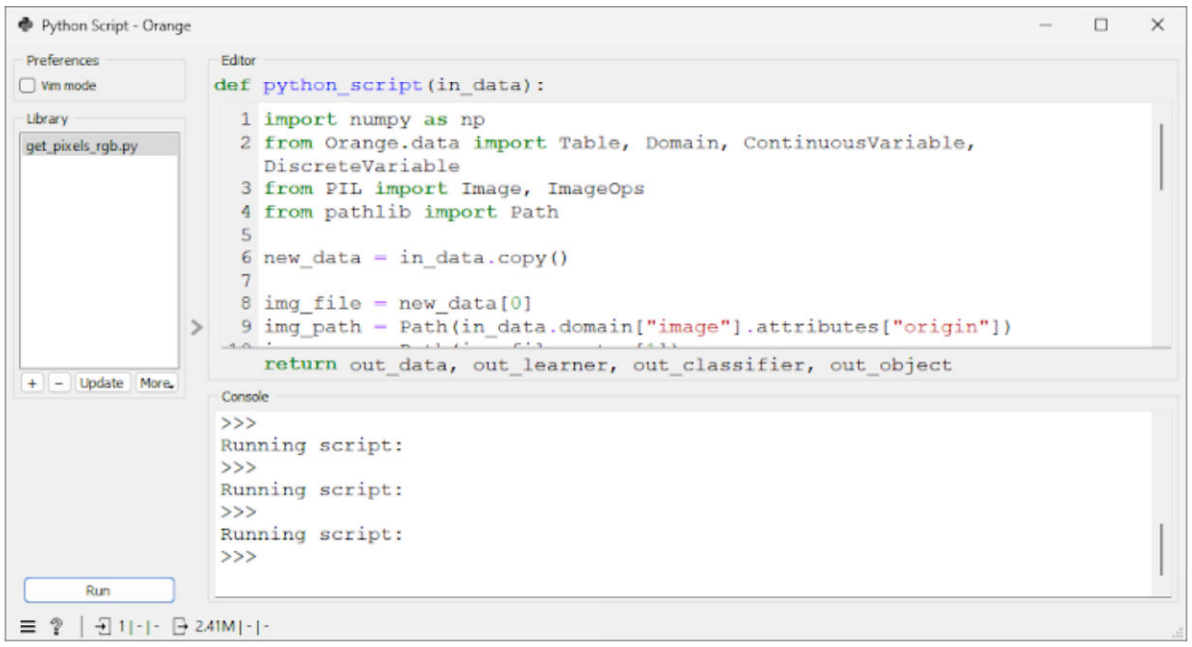
- ◯ Done

In the "Python Script" window, click the "More" button under the "Library" section and choose "Import Script from File." *

- ◯ Done

Import the downloaded script and click the "Run" button. This process may take some time. *

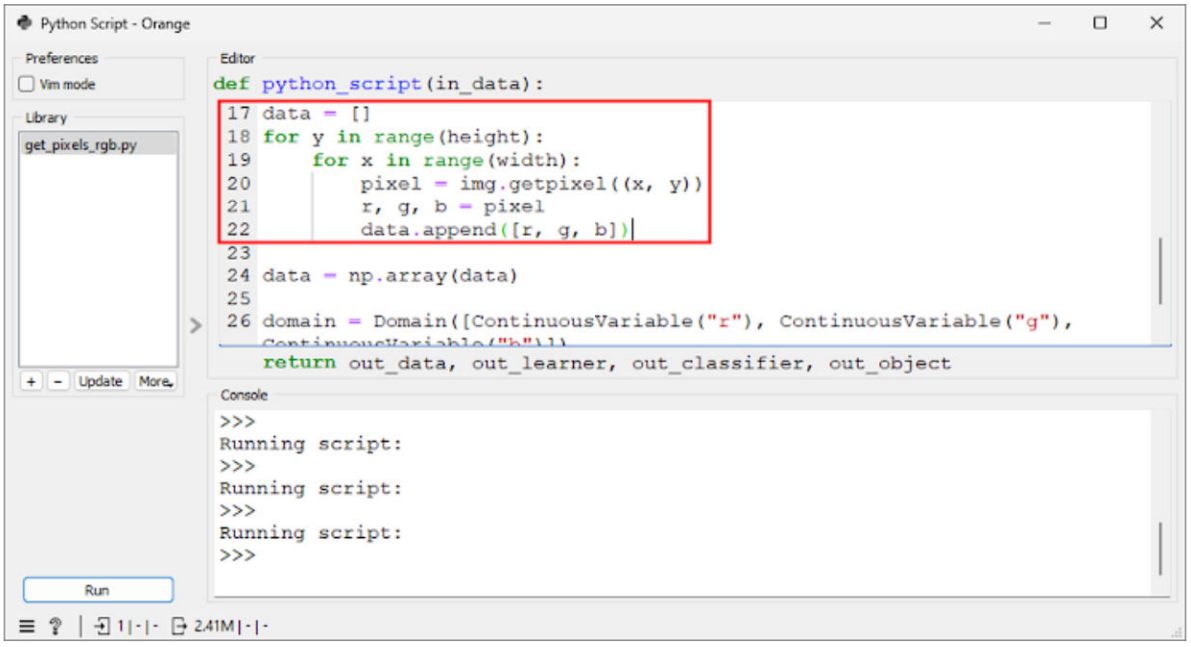
You should then see something like this:



- ◯ Done

Analyze the script. *

Question: From lines 17-22, in what order are the RGB values retrieved from the image?

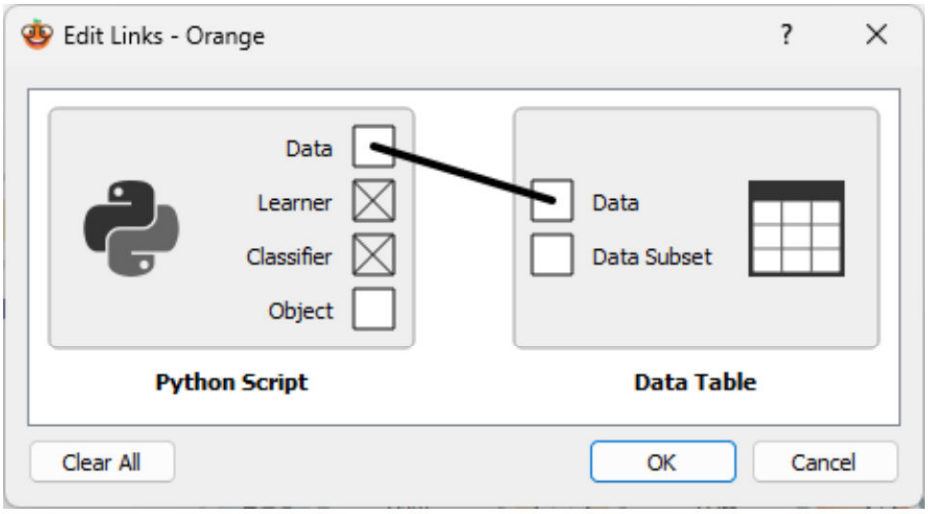


- ◯ Moving top to bottom, column by column, from left to right.
- ◯ Moving left to right, row by row, from top to bottom.

Drag and drop a "Data Table" widget and connect it to the "Python Script" to get the data that the script outputs. *

- ◯ Done

Make sure that "Data Table" is connected to "Python Script" like this: *



- ◯ Done

Check the "Info" section in "Data Table". *

Question: What does "2408567 instances" refer to?

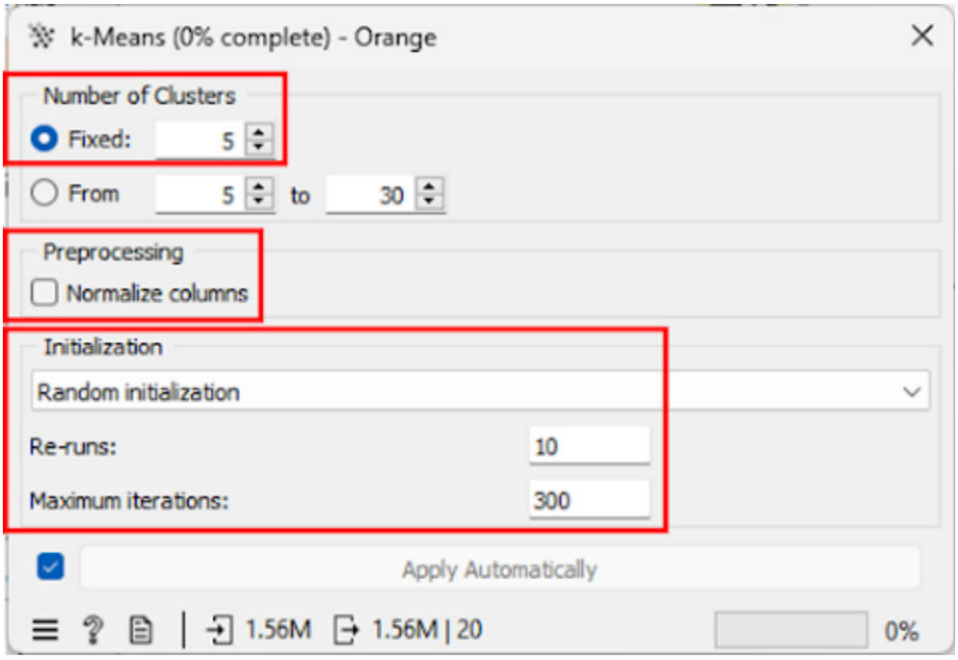
- ◯ The total number of pixels in the image.
- ◯ The width of the image.
- ◯ The height of the image.

Step 2: Apply K-Means to Find the Main Colors

Drag and drop a "k-Means" widget, but do not connect it to "Data Table" yet. *

- ◯ Done

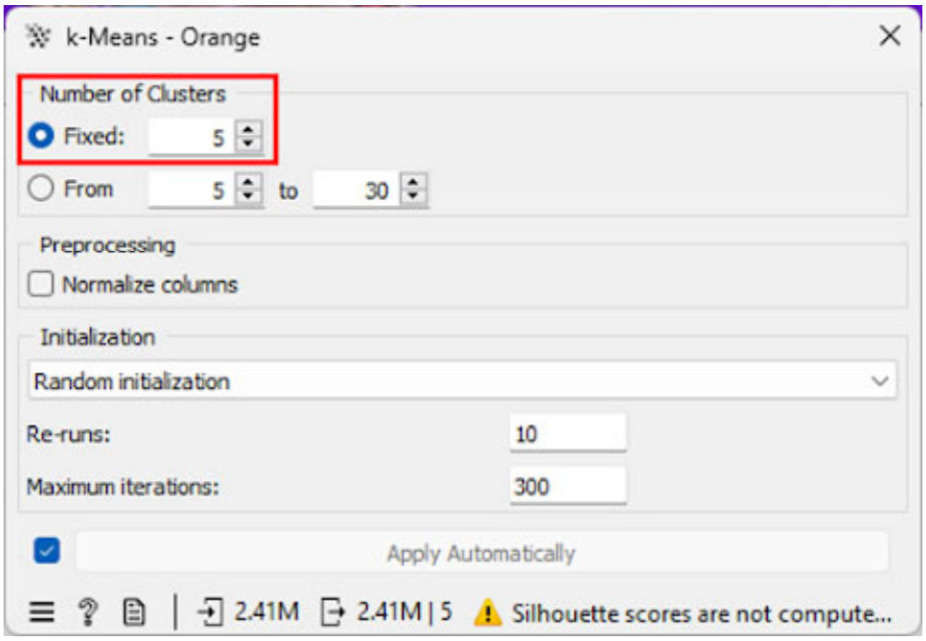
Set "k-Means" to these settings: *



- [ ] Done

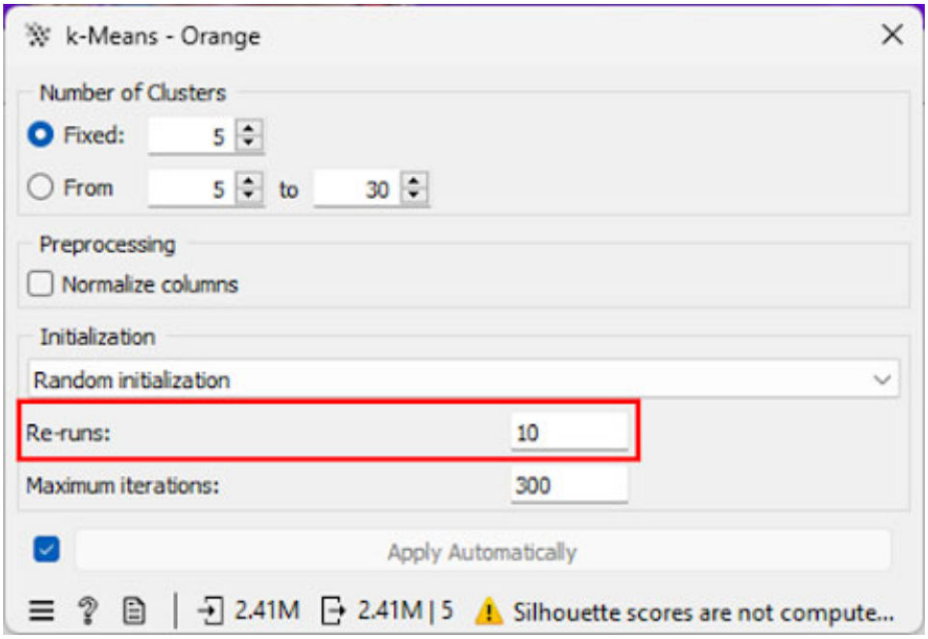
Question: What does the red box in the "k-Means" window image indicate? *

You may refer to this documentation.



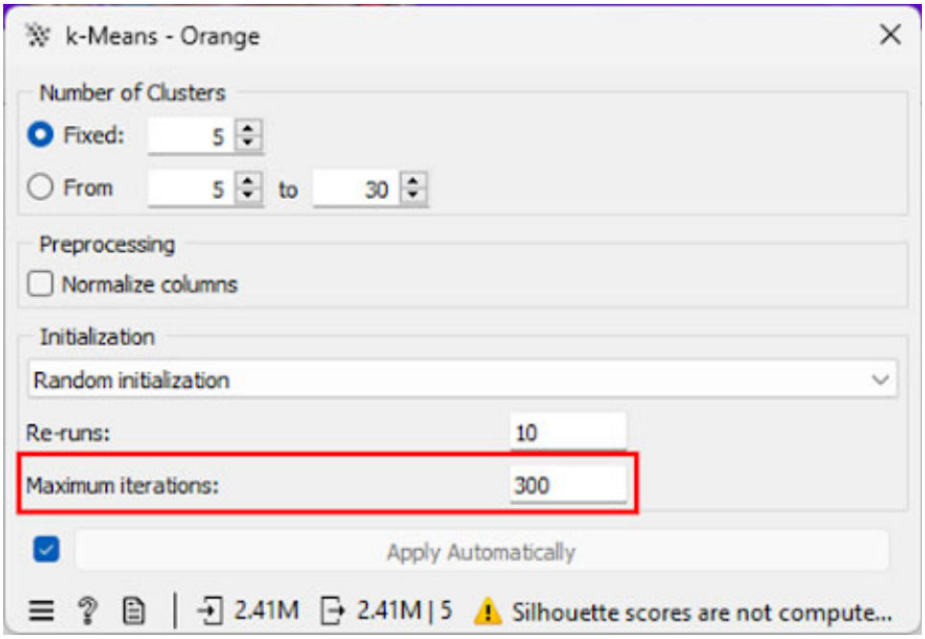
- K-means will produce 5 clusters.
- K-means will run 5 times, choosing the best result.
- The maximum number of centroid iterations is 5.

Question: What does the red box in the "k-Means" window image indicate? *



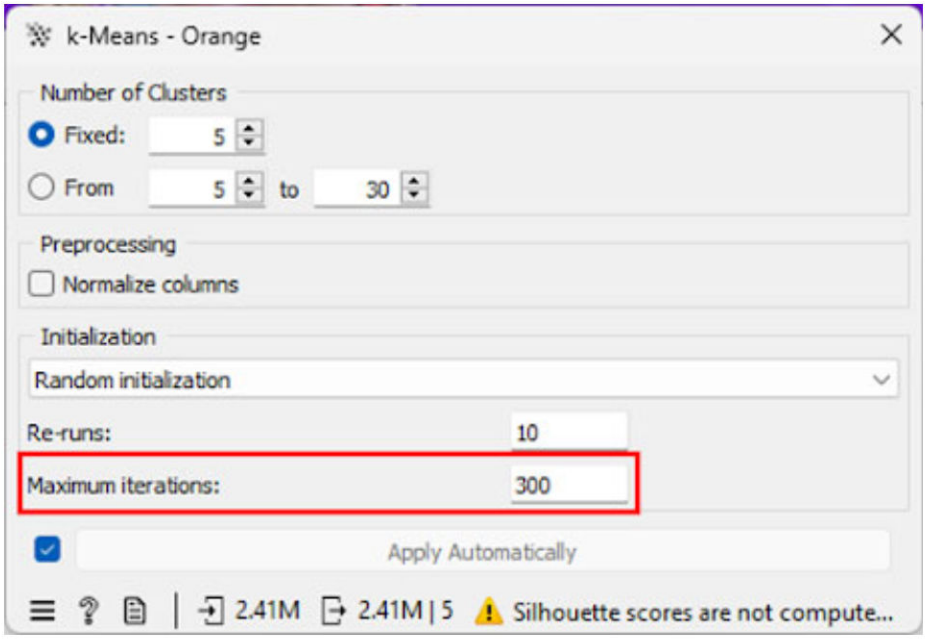
- ◯ K-means will run 10 times, choosing the best result.
- ◯ K-means will produce 10 clusters.
- ◯ The maximum number of centroid iterations is 10.

Question: What does the red box in the "k-Means" window image indicate? *



- ◯ The maximum number of centroid iterations is 300.
- ◯ K-means will produce 300 clusters.
- ◯ K-means will run 300 times, choosing the best result.

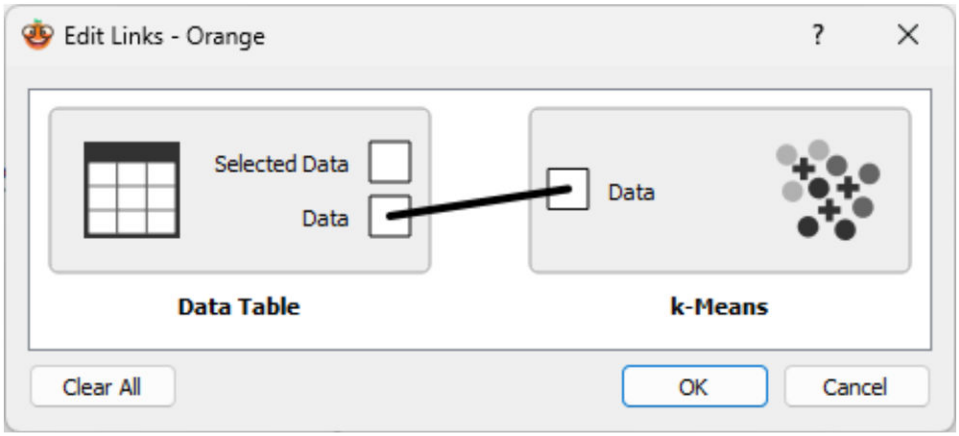
Question: What does the red box in the "k-Means" window image indicate? *



- ◯ The maximum number of centroid iterations is 300.
- ◯ K-means will produce 300 clusters.
- ◯ K-means will run 300 times, choosing the best result.

Connect "k-Means" to "Data Table". *

Make sure it is connected to "Data Table" like this:

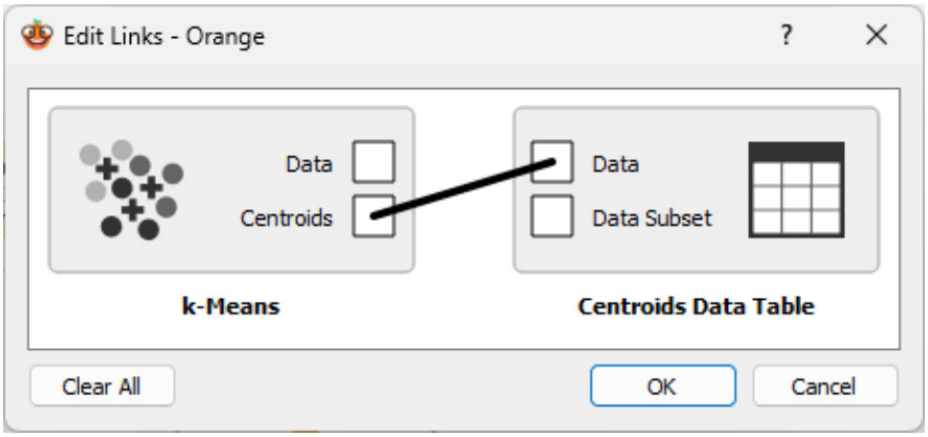


- ◯ Done

Drag and drop a "Data Table" widget and name it "Centroids Data Table". *

- ◯ Done

Make sure "Centroids Data Table" is connected to "k-Means" like this: *

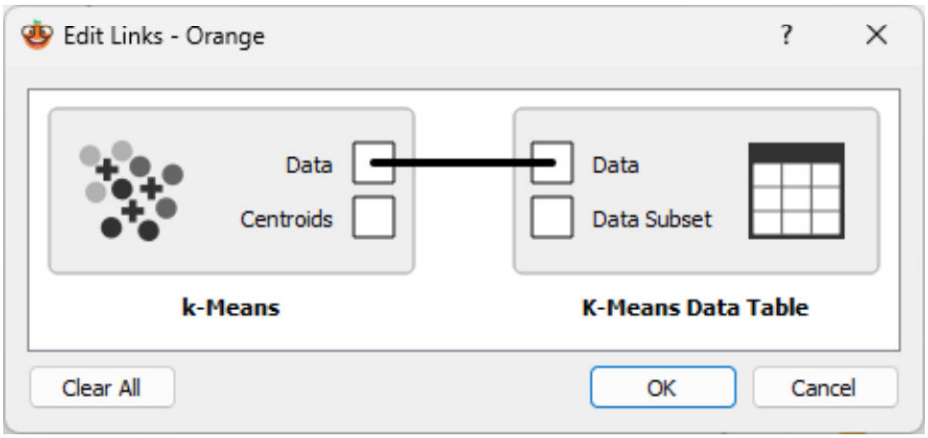


- ◯ Done

Drag and drop a "Data Table" widget and name it "K-Means Data Table". *

- ◯ Done

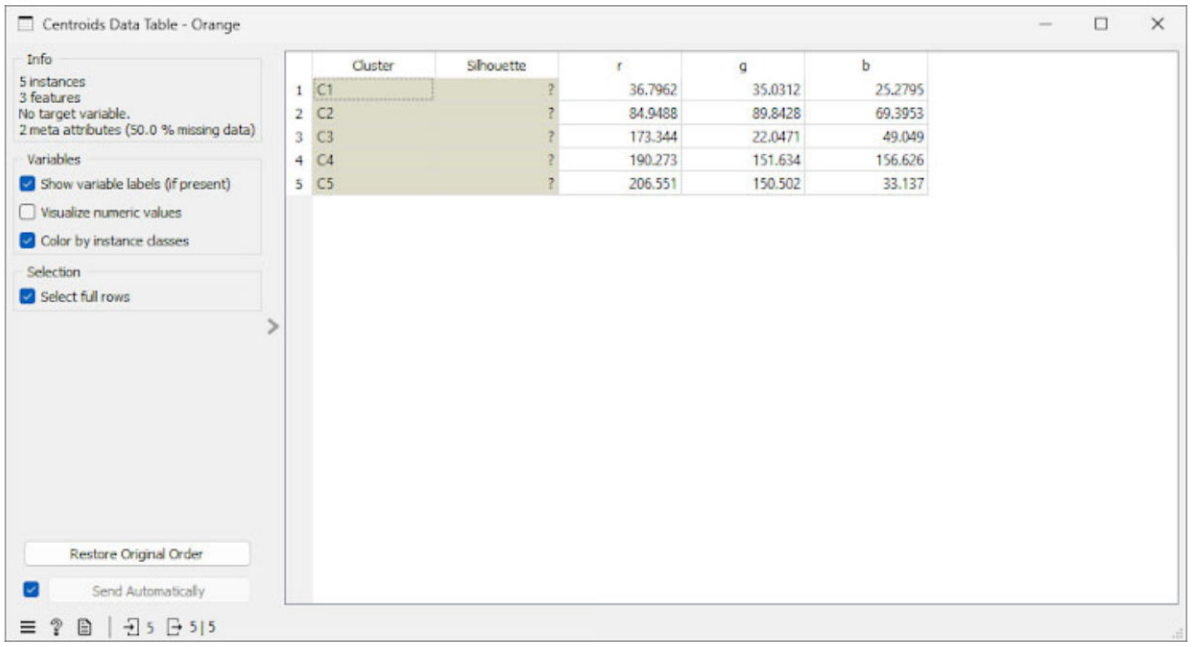
Make sure "K-Means Data Table" is connected to "k-Means" like this: *



- ◯ Done

Check "Centroids Data Table". *

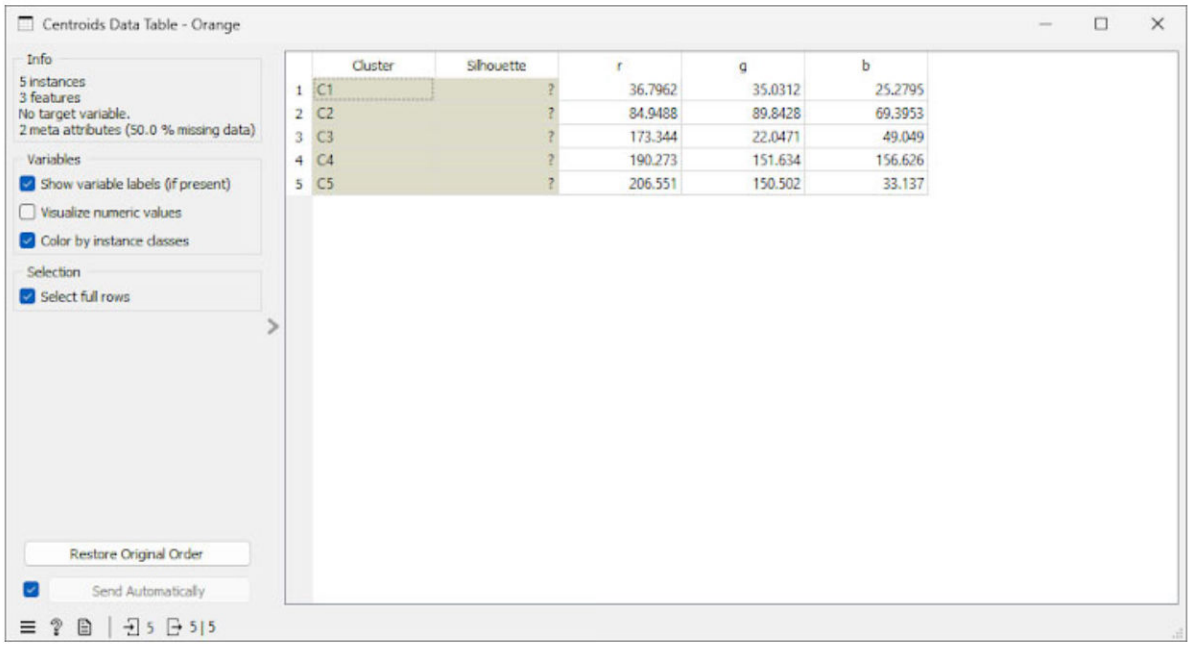
You should see something like this:



◯ Done

Question: Did you get the same RGB values for all 5 clusters? *

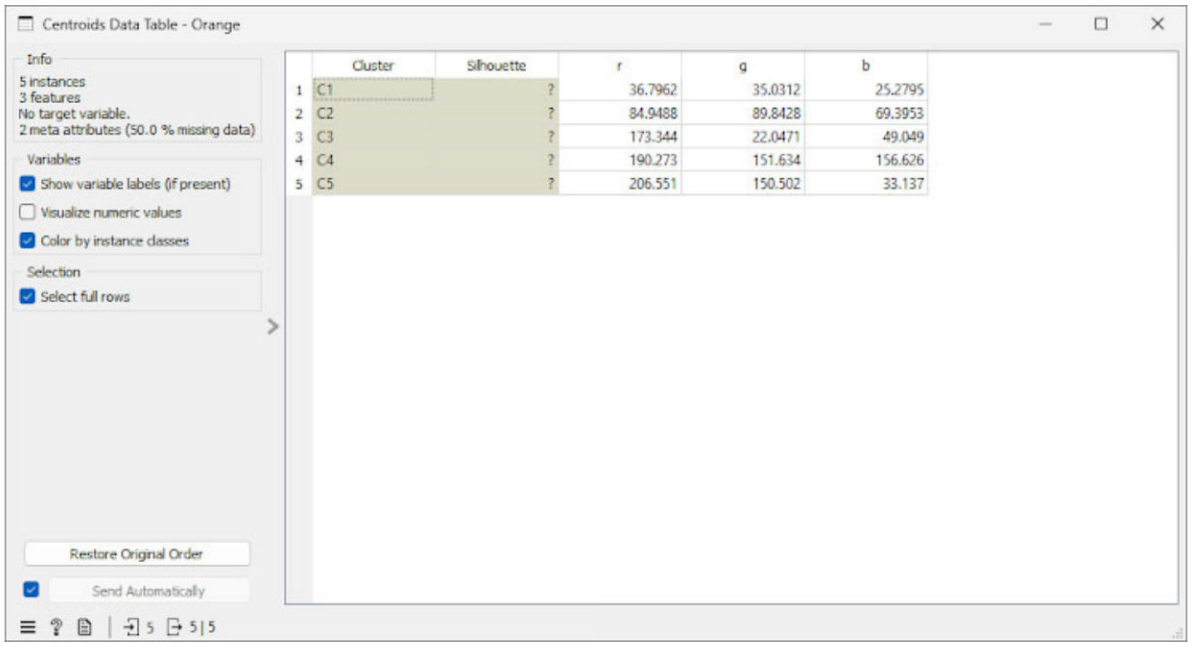
If not, try increasing the number of re-runs in "k-Means".

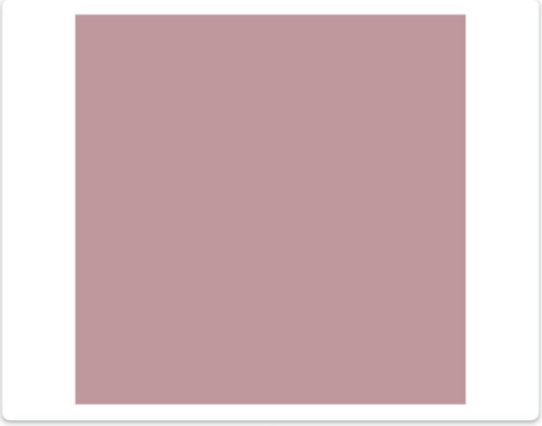


- ◯ Yes
- ◯ No

Question: Which color represents Cluster 1? *

Hint: Use an RGB calculator to find out the color.





Option 4

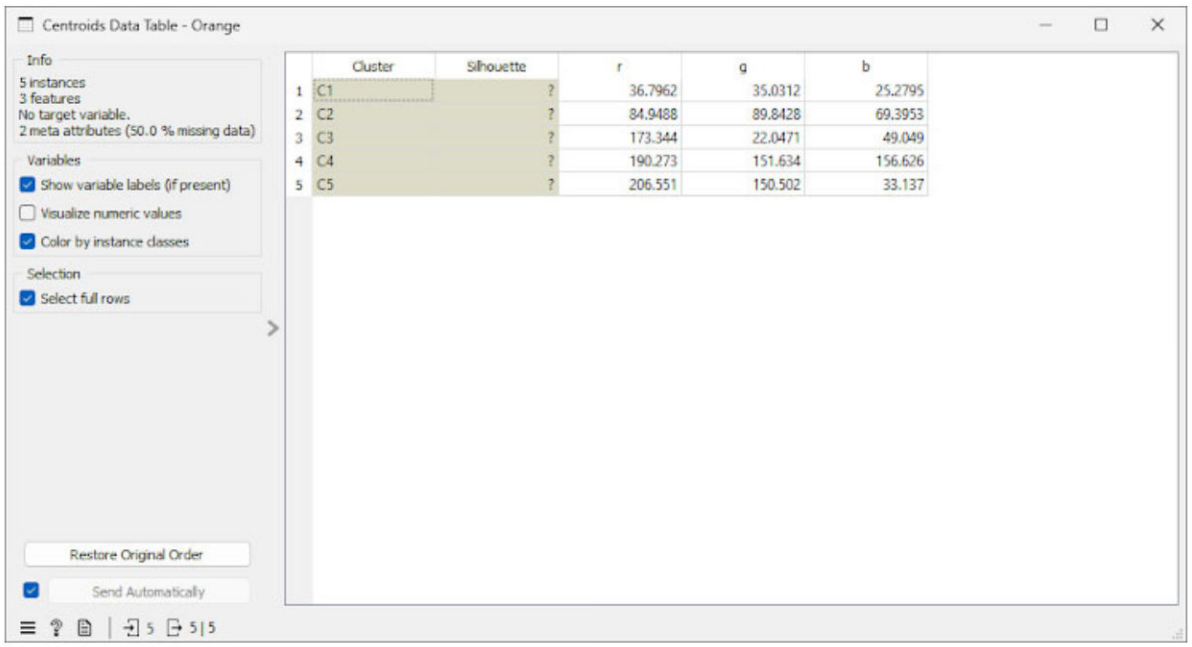
Option 2

Option 5

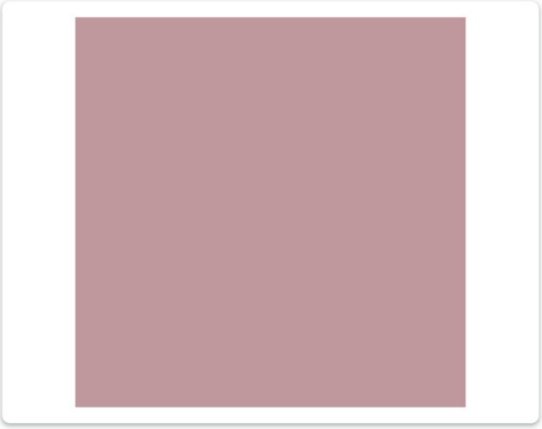
Option 1

Option 3

Question: Which color represents Cluster 2? *



	Cluster	Silhouette	r	g	b
1	C1	?	36.7962	35.0312	25.2795
2	C2	?	84.9488	89.8428	69.3953
3	C3	?	173.344	22.0471	49.049
4	C4	?	190.273	151.634	156.626
5	C5	?	206.551	150.502	33.137

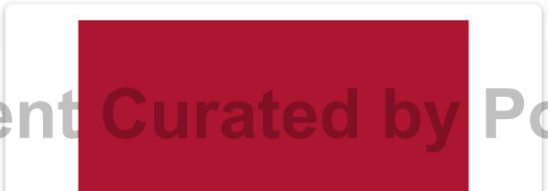


Option 4

Option 1

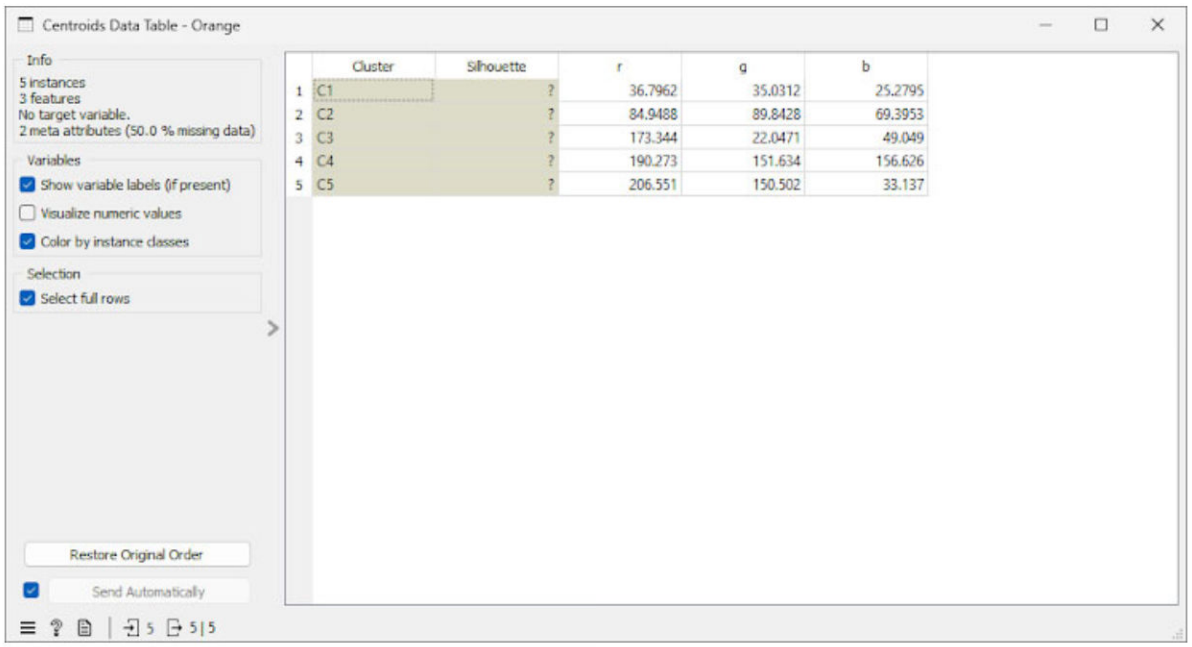
Option 5

Option 2

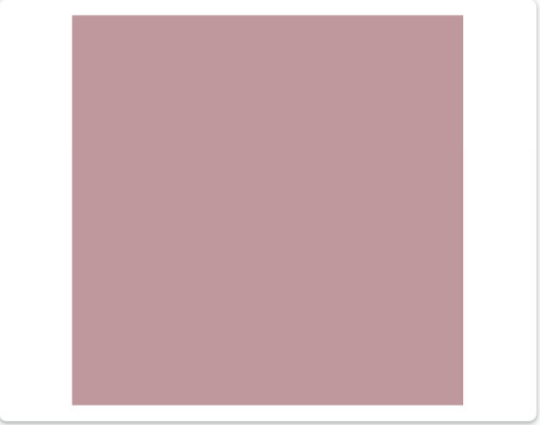


Option 3

Question: Which color represents Cluster 3? *



Option 3



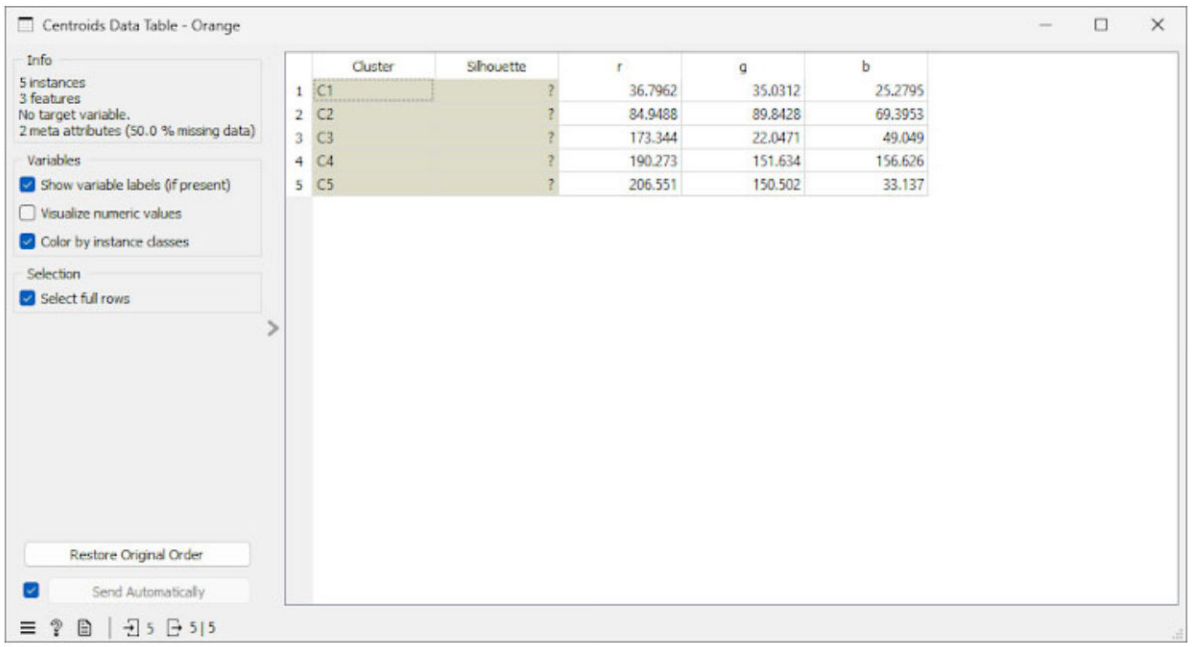
Option 4

Option 5

Option 2

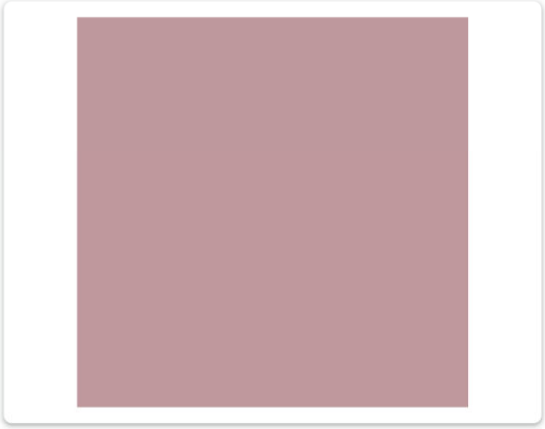
Option 1

Question: Which color represents Cluster 4? *



	Cluster	Silhouette	r	g	b
1	C1	?	36.7962	35.0312	25.2795
2	C2	?	84.9488	89.8428	69.3953
3	C3	?	173.344	22.0471	49.049
4	C4	?	190.273	151.634	156.626
5	C5	?	206.551	150.502	33.137

○ Option 5



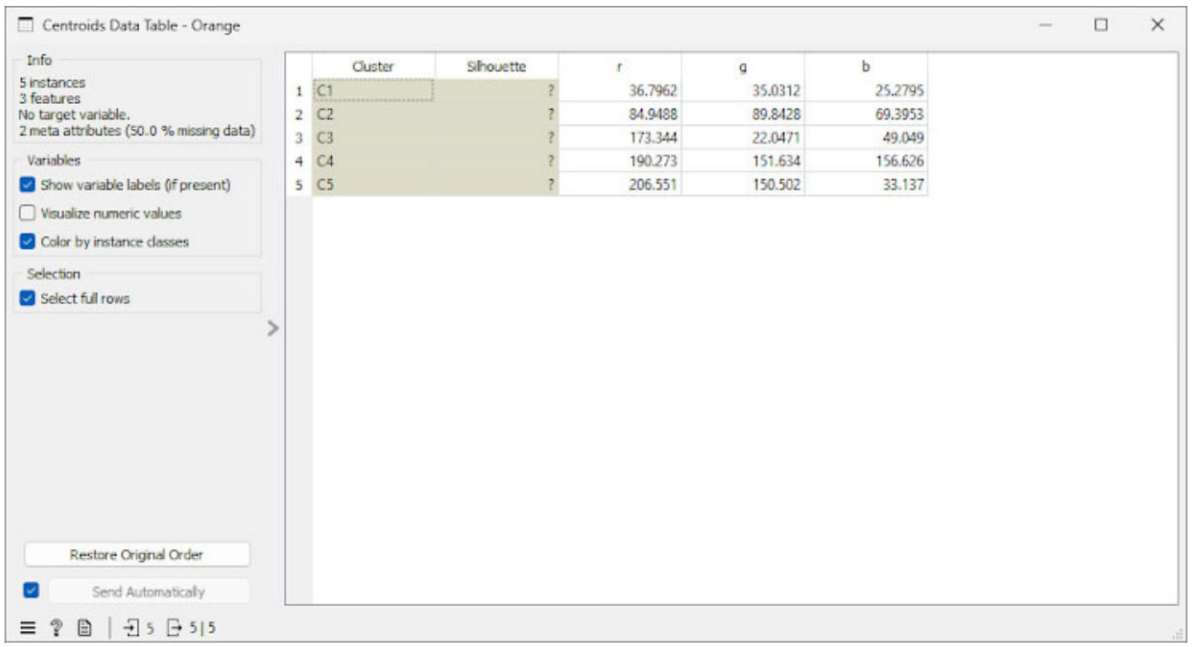
○ Option 4

○ Option 2

○ Option 3

Option 1

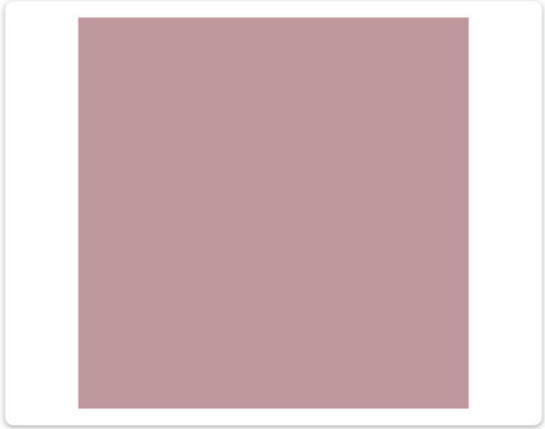
Question: Which color represents Cluster 5? *



	Cluster	Silhouette	r	g	b
1	C1	?	36.7962	35.0312	25.2795
2	C2	?	84.9488	89.8428	69.3953
3	C3	?	173.344	22.0471	49.049
4	C4	?	190.273	151.634	156.626
5	C5	?	206.551	150.502	33.137

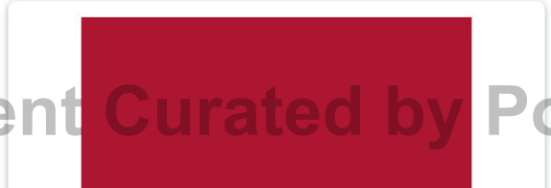
Option 2

Option 1



Option 4

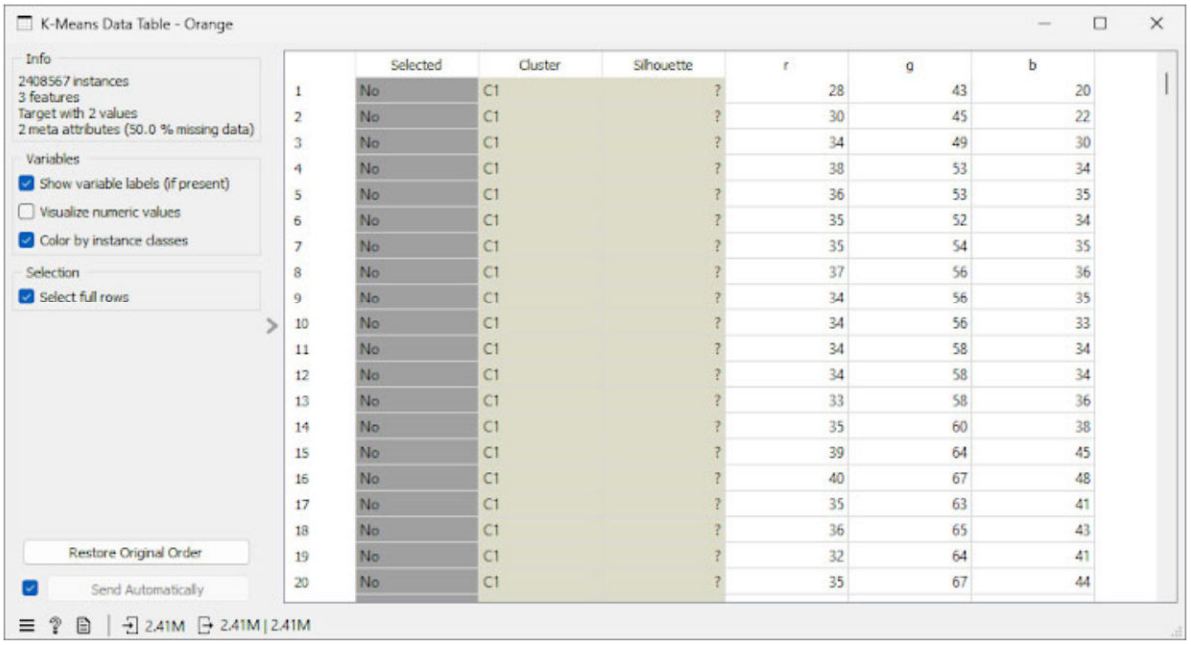
Option 5



Option 3

Check "K-Means Data Table". *

You should see something like this. If not, try clicking the "Restore Original Order" button:



	Selected	Cluster	Silhouette	r	g	b
1	No	C1	?	28	43	20
2	No	C1	?	30	45	22
3	No	C1	?	34	49	30
4	No	C1	?	38	53	34
5	No	C1	?	36	53	35
6	No	C1	?	35	52	34
7	No	C1	?	35	54	35
8	No	C1	?	37	56	36
9	No	C1	?	34	56	35
10	No	C1	?	34	56	33
11	No	C1	?	34	58	34
12	No	C1	?	34	58	34
13	No	C1	?	33	58	36
14	No	C1	?	35	60	38
15	No	C1	?	39	64	45
16	No	C1	?	40	67	48
17	No	C1	?	35	63	41
18	No	C1	?	36	65	43
19	No	C1	?	32	64	41
20	No	C1	?	35	67	44

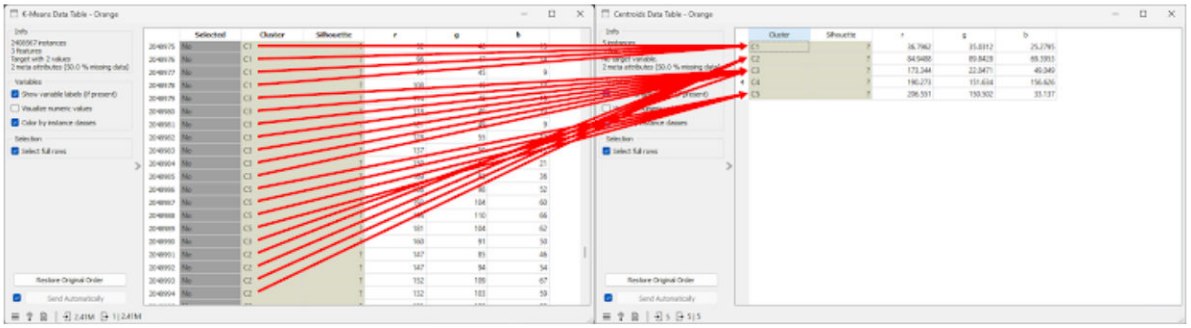
Done

In "K-Means Data Table", each pixel is assigned to a specific cluster. *

The RGB values for each cluster are stored in a separate data table ("Centroids Data Table").

To determine the new color assigned to each pixel, you need to join the two tables.

Example:

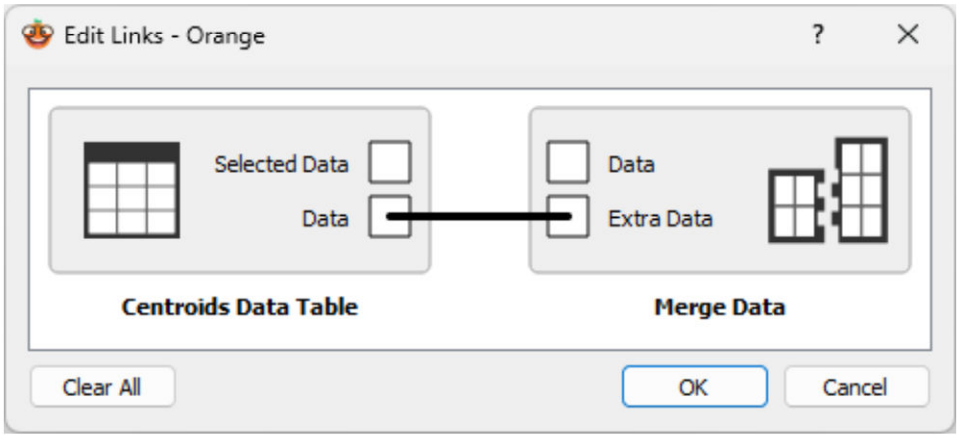


- [ ] I have read and understood.

Drag and Drop a "Merge Data" widget. *

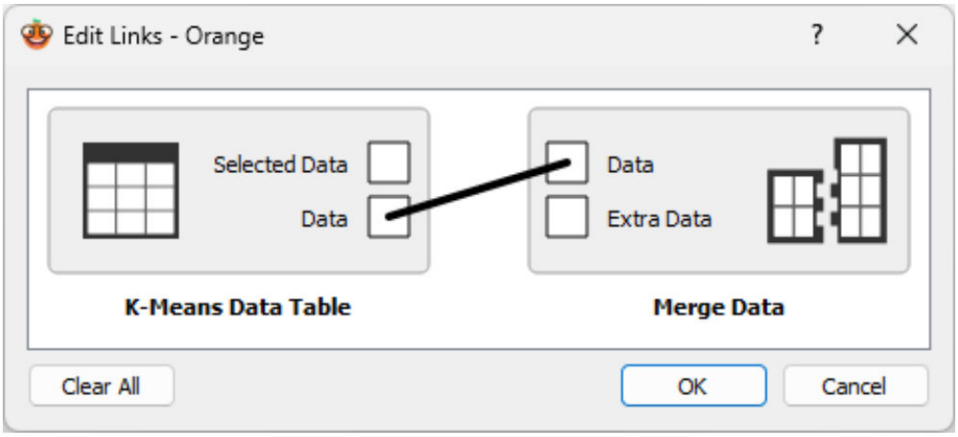
- [ ] Done

Make sure "Merge Data" is connected to "Centroids Data Table" like this: *



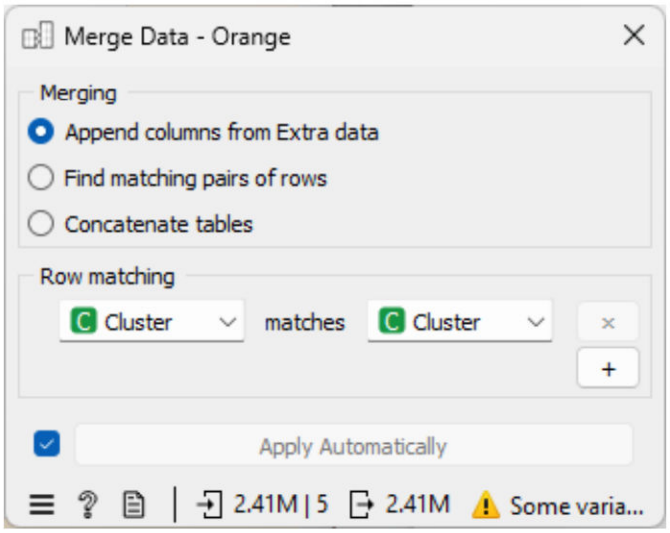
- [ ] Done

Make sure "Merge Data" is connected to "K-Means Data Table" like this: *



- [ ] Done

Set "Merge Data" to these settings: *

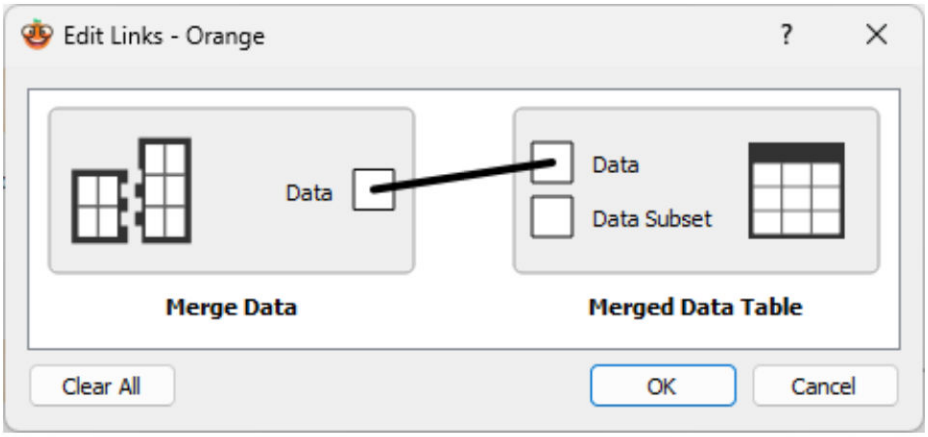


- [ ] Done

To check if the tables are merged, drag and drop a "Data Table" widget, connect it * to "Merge Data", and name it "Merged Data Table".

- [ ] Done

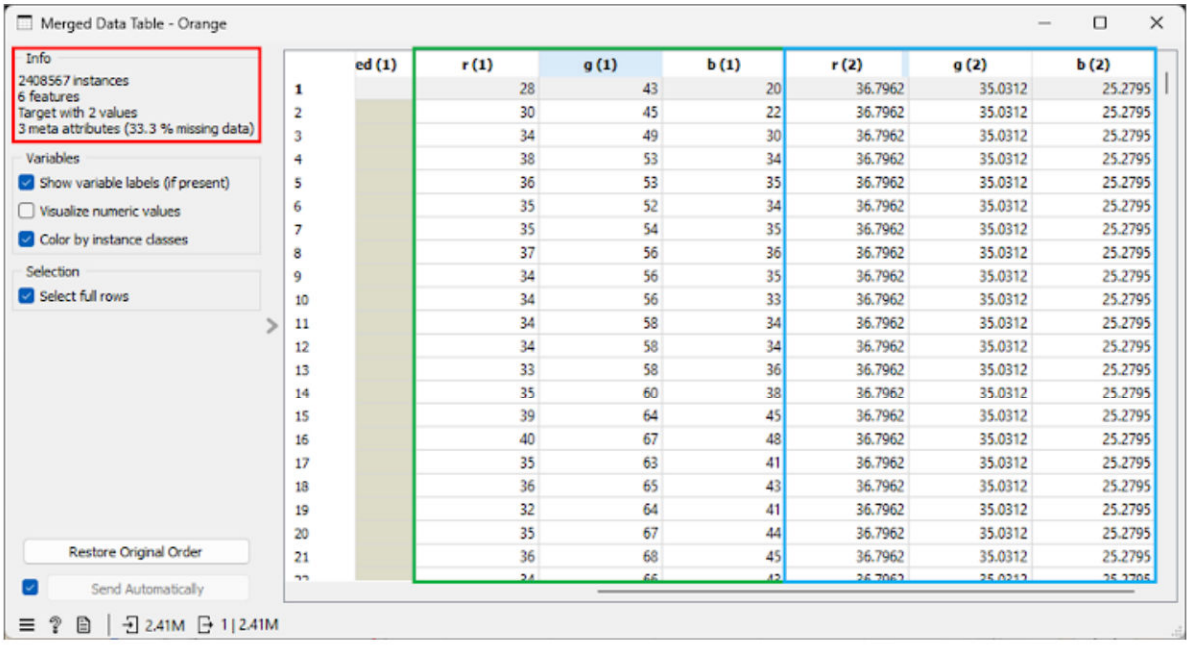
Make sure "Merged Data Table" is connected to "Merge Data" like this: *



◯ Done

Check "Merged Data Table". *

You should see something like this. The green box shows the original RGB values for each pixel, while the blue box shows the assigned RGB values based on clustering.



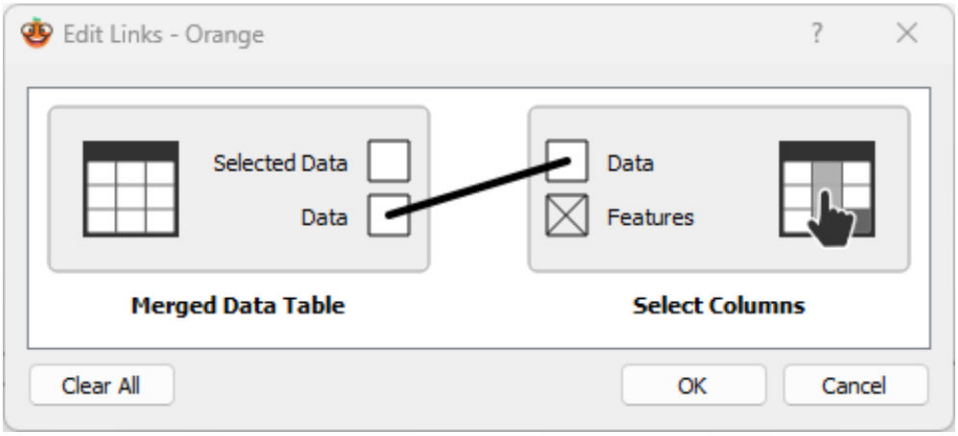
	ed (1)	r (1)	g (1)	b (1)	r (2)	g (2)	b (2)
1		28	43	20	36.7962	35.0312	25.2795
2		30	45	22	36.7962	35.0312	25.2795
3		34	49	30	36.7962	35.0312	25.2795
4		38	53	34	36.7962	35.0312	25.2795
5		36	53	35	36.7962	35.0312	25.2795
6		35	52	34	36.7962	35.0312	25.2795
7		35	54	35	36.7962	35.0312	25.2795
8		37	56	36	36.7962	35.0312	25.2795
9		34	56	35	36.7962	35.0312	25.2795
10		34	56	33	36.7962	35.0312	25.2795
11		34	58	34	36.7962	35.0312	25.2795
12		34	58	34	36.7962	35.0312	25.2795
13		33	58	36	36.7962	35.0312	25.2795
14		35	60	38	36.7962	35.0312	25.2795
15		39	64	45	36.7962	35.0312	25.2795
16		40	67	48	36.7962	35.0312	25.2795
17		35	63	41	36.7962	35.0312	25.2795
18		36	65	43	36.7962	35.0312	25.2795
19		32	64	41	36.7962	35.0312	25.2795
20		35	67	44	36.7962	35.0312	25.2795
21		36	68	45	36.7962	35.0312	25.2795

◯ Done

To keep only the updated RGB values for each pixel, drag and drop "Select Columns" and connect it to "Merged Data Table". *

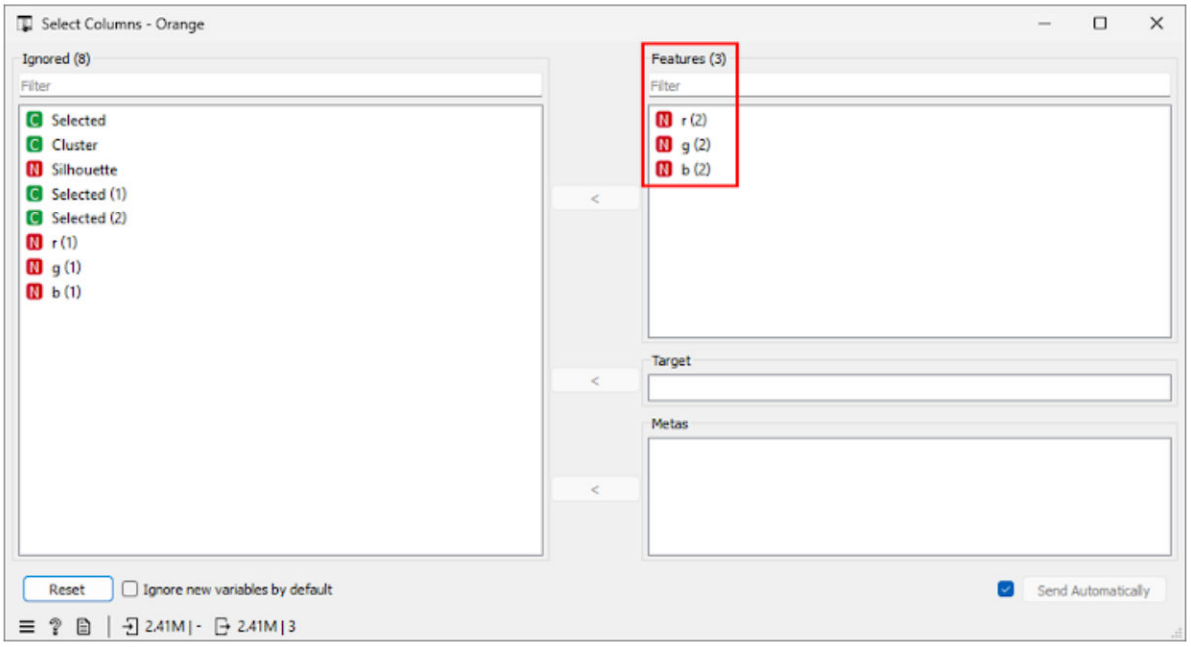
- ◯ Done

Make sure "Select Columns" is connected to "Merged Data Table" like this: *



- ◯ Done

Set "Select Columns" as shown in the image to keep only the new RGB values for each pixel. *



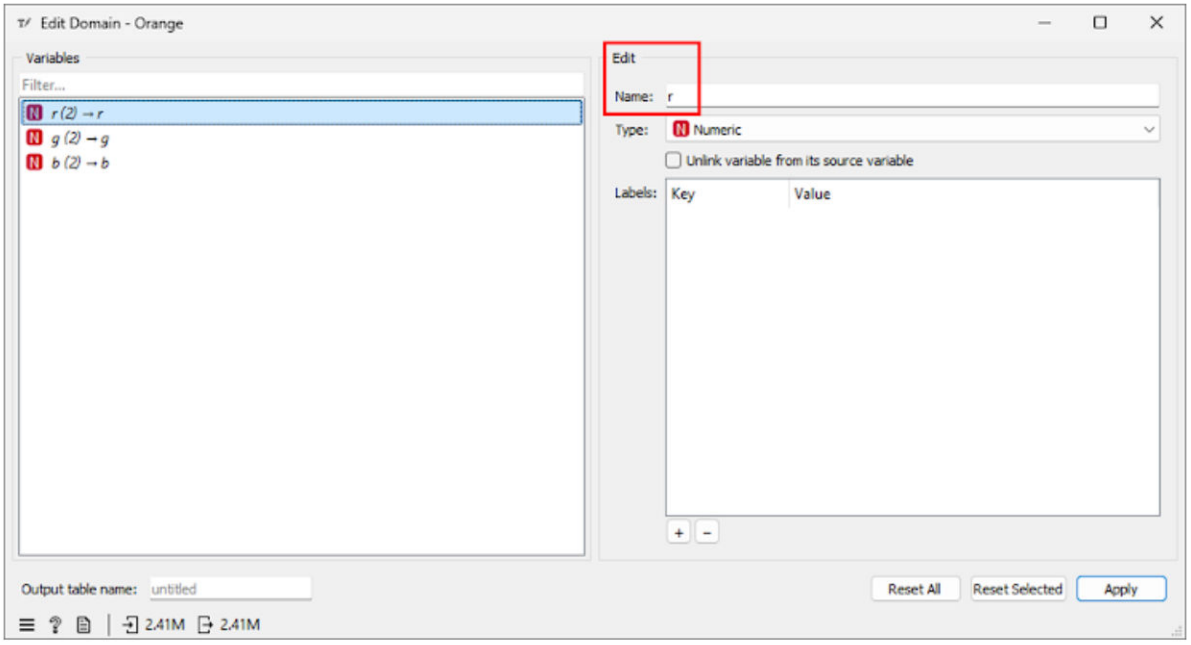
- ◯ Done

To rename columns (e.g., "r (2)" to "r"), drag and drop an "Edit Domain" widget and connect it to "Select Columns". *

- ◯ Done

In the "Edit" section of "Edit Domain," change each column name. *

Click the "Apply" button to save the changes.



- ◯ Done

Step 3: Save the Compressed Image

Pause the workflow using the button in the lower left corner of Orange. *

- ◯ Done

Download a script named "create_image_from_rgb_values.py" from Google Drive * and save it in the "Original Image" folder.

- ◯ Done

Drag and drop another "Python Script" widget. *

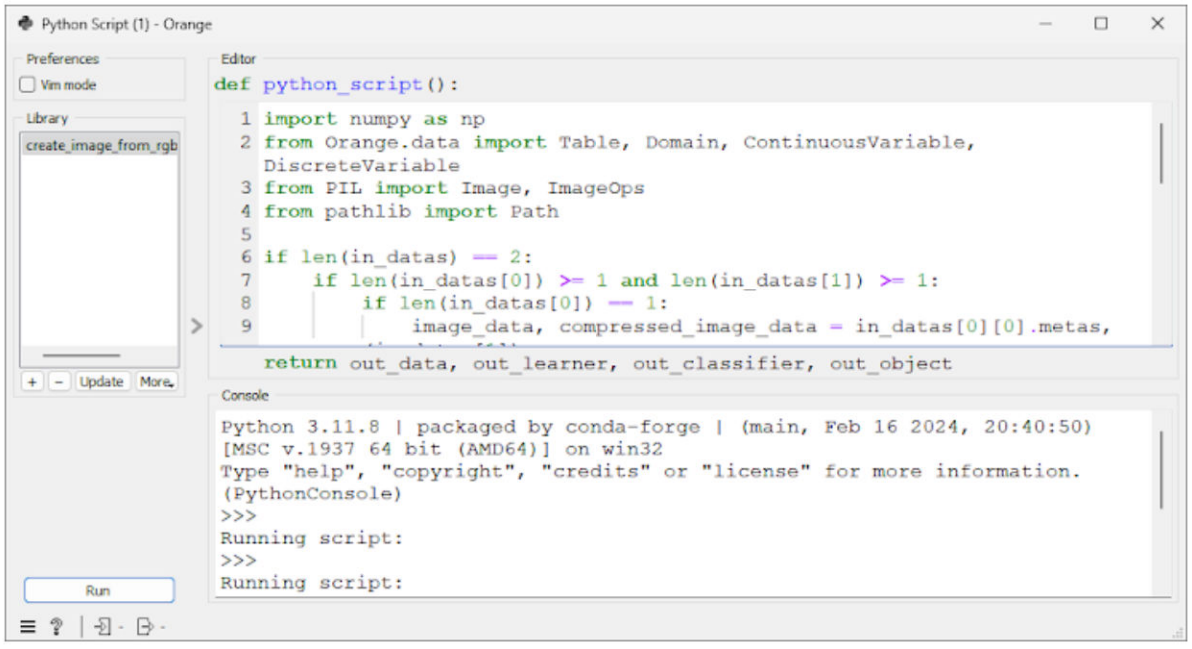
- ◯ Done

In the "Python Script (1)" window, click the "More" button under the "Library" section and choose "Import Script from File." *

- ◯ Done

Import the downloaded script. *

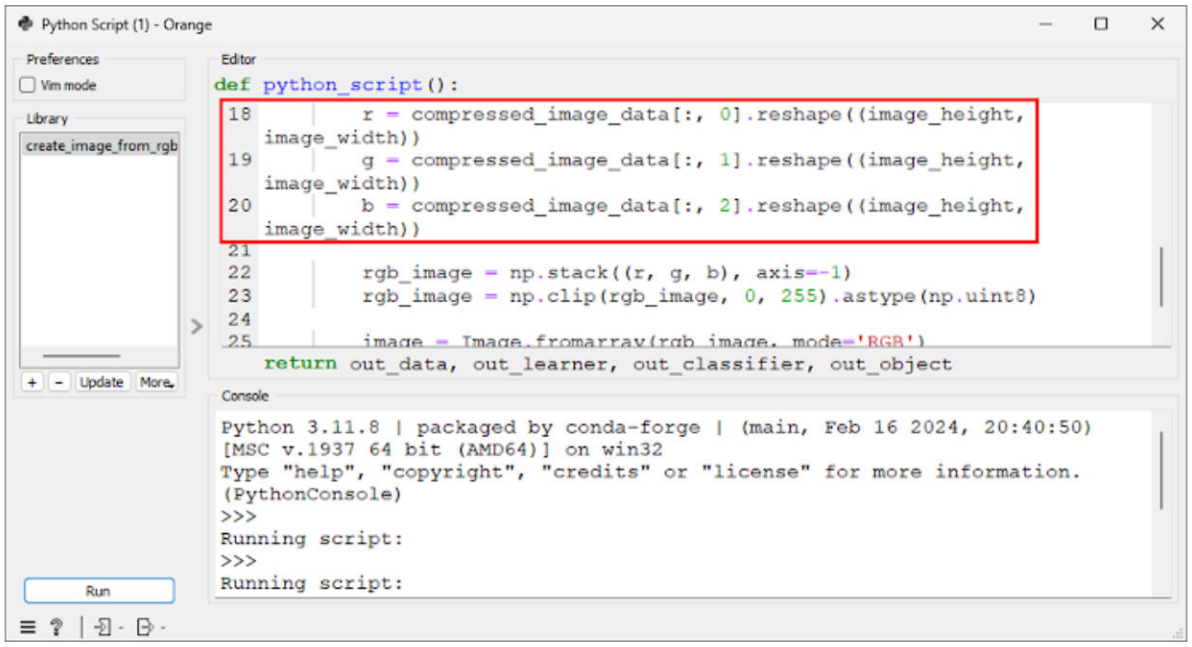
You should see something like this:



- ◯ Done

Analyze the script. *

Question: What do lines 18-20 do?



- ◯ Saves the image in the user's "Downloads" folder.
- ◯ Extracts the red, green, and blue channels from the table, reshaping each into a 2D array matching the image's height and width in pixels.
- ◯ Gets the image's width and height from the metadata.
- ◯ Generates an RGB image from the 3D array.
- ◯ Combines the red, green, and blue channels into a 3D array.

Analyze the script. *

Question: What does line 22 do?

- ◯ Combines the red, green, and blue channels into a 3D array.
- ◯ Generates an RGB image from the 3D array.
- ◯ Gets the image's width and height from the metadata.
- ◯ Saves the image in the user's "Downloads" folder.
- ◯ Extracts the red, green, and blue channels from the table, reshaping each into a 2D array matching the image's height and width in pixels.

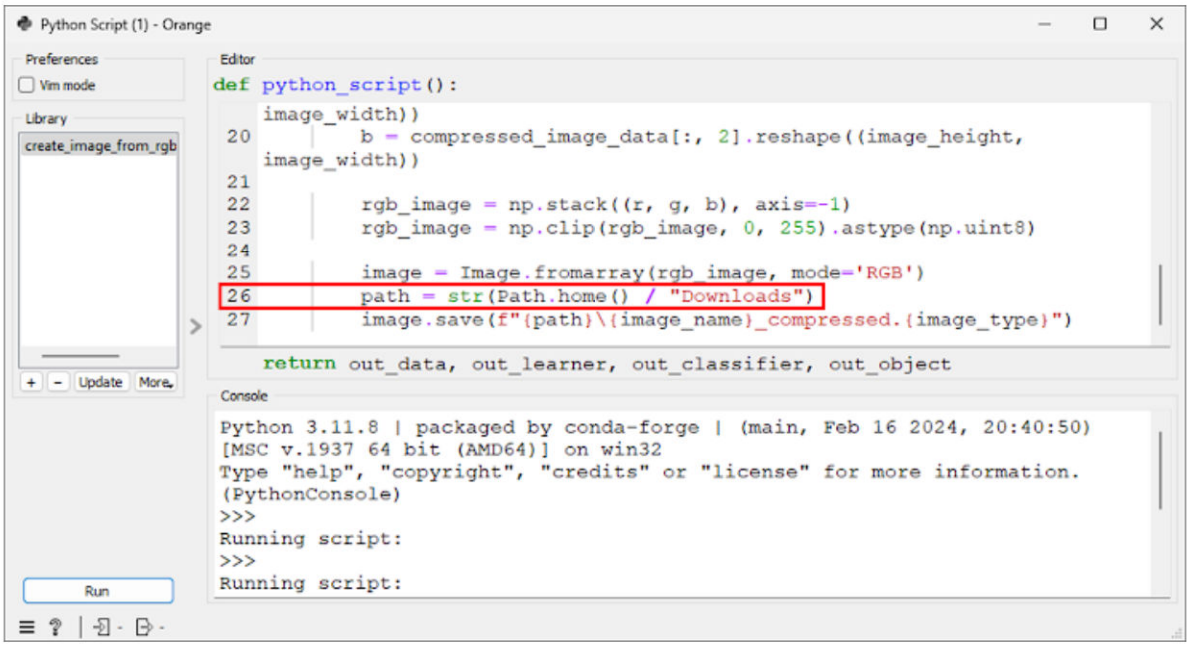
Analyze the script. *

Question: What does line 25 do?

- Extracts the red, green, and blue channels from the table, reshaping each into a 2D array matching the image's height and width in pixels.
- Generates an RGB image from the 3D array.
- Saves the image in the user's "Downloads" folder.
- Gets the image's width and height from the metadata.
- Combines the red, green, and blue channels into a 3D array.

Analyze the script. *

Question: What does line 26 do?



- Extracts the red, green, and blue channels from the table, reshaping each into a 2D array matching the image's height and width in pixels.
- Generates an RGB image from the 3D array.
- Combines the red, green, and blue channels into a 3D array.
- Gets the image's width and height from the metadata.
- Saves the image in the user's "Downloads" folder.

Connect "Python Script (1)" to "Edit Domain" and "Import Images". *

- Done

To resume the workflow, click the "Pause" button again. *

- Done

Check "Python Script (1)". *

Click the "Run" button to execute the downloaded script. This process may take some time.

- [ ] Done

Check your "Downloads" folder for an image named "Tulips_compressed.jpg". *

- [ ] Done

Rename it to "k5.jpg" to show that this image was created using five clusters. *

- [ ] Done

Create a folder named "Compressed Image" and move the compressed image into it. *

- [ ] Done

Question: What does "k5.jpg" look like? *

Option 1

Option 4

Option 2

Option 3

Generate images using 10, 20, and 30 clusters by adjusting "Number of Clusters" * in "k-Means".

Rename each image before changing the number of clusters again.

Done

Open the images with 5, 10, 20, and 30 clusters in Photo Viewer at the same time * to see both the photos and their sizes.

Done

Check the file size of the original image. *

If it is larger than the compressed images, k-Means can effectively reduce the size of this image**.**

If not, review and adjust your workflow.

◯ I have read and understood.

Check the file sizes of the four compressed images. *

If the file size decreases with fewer clusters, it indicates that less detail leads to a smaller file size for this image.

If not, review and adjust your workflow.

◯ I have read and understood.

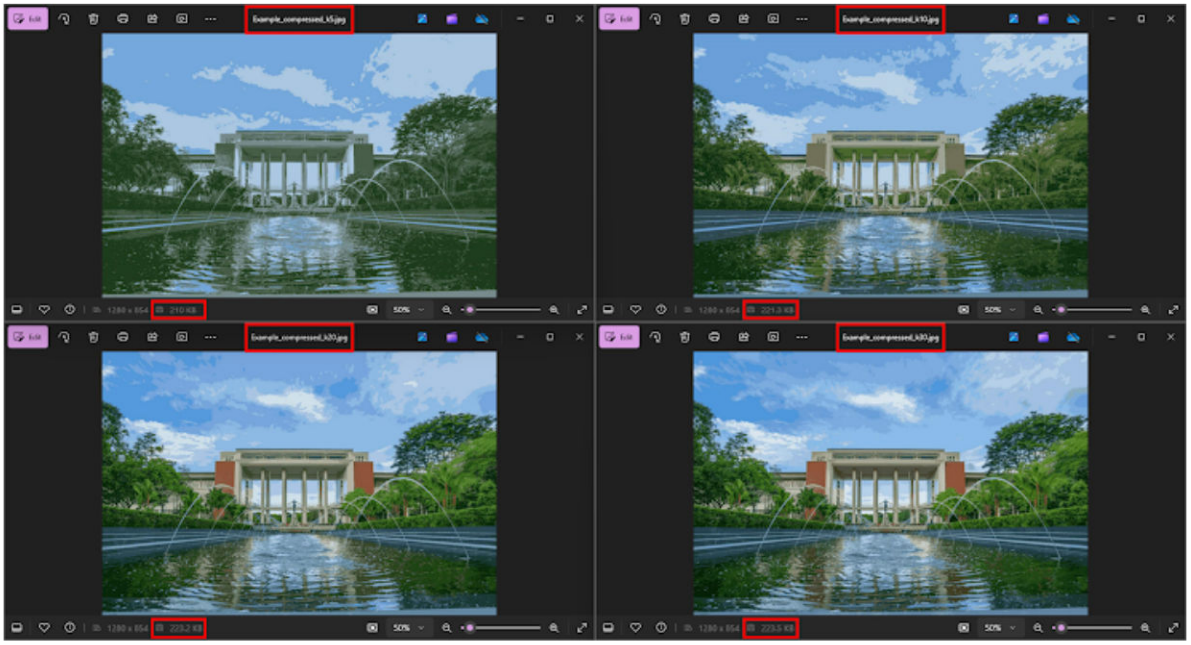
Use the 2x2 Snap Layout to view all four images at once. *

Take a full screenshot with the following arrangement:

- Top-left: 5 clusters
- Top-right: 10 clusters
- Bottom-left: 20 clusters
- Bottom-right: 30 clusters

It should display the file names along with their corresponding cluster and file sizes.

Example:

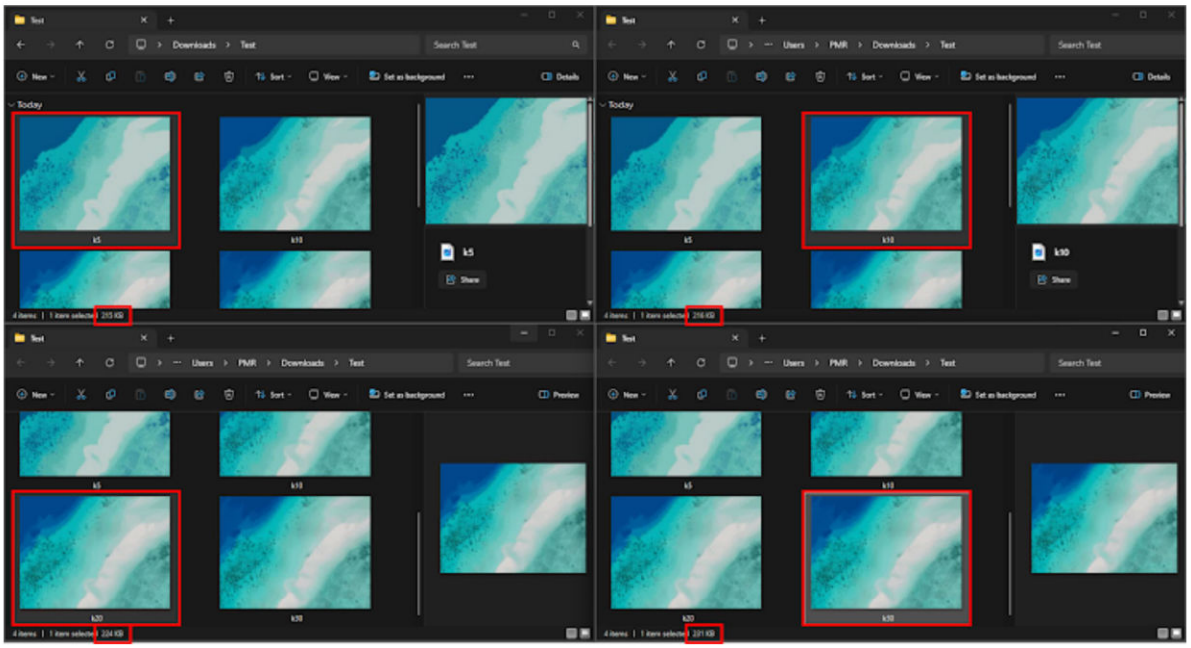


◯ Done

You can also try using File Explorer. *

It should display the file names along with their corresponding cluster and file sizes.

Example:



Done

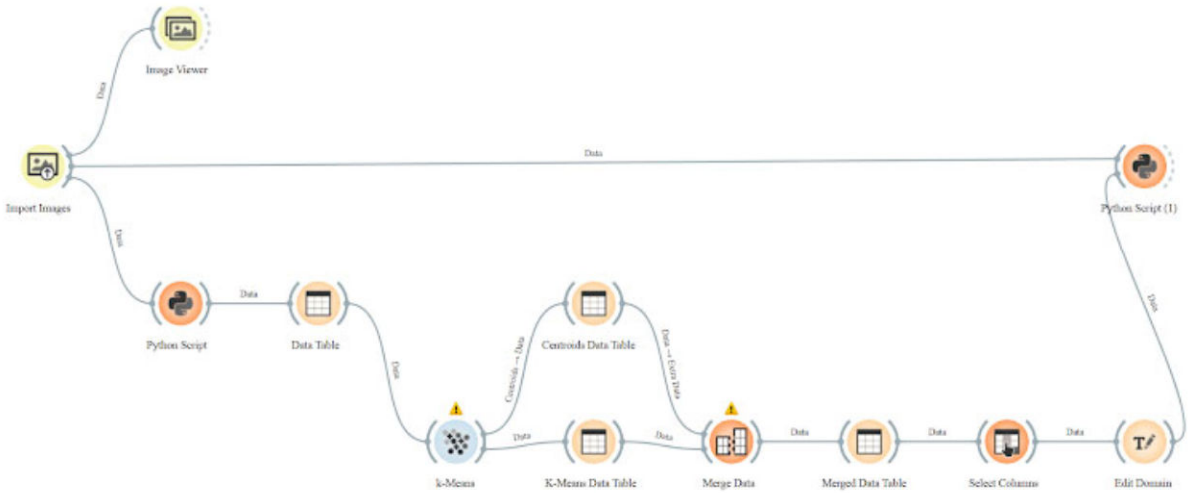
Upload the screenshot here, **showing that file sizes decrease with fewer clusters and are smaller than the original.** *

Add file

Step 4: Save your Workflow

Compare your workflow to this. *

If they differ, review and adjust your workflow.



- [ ] I have the same workflow.

Save your workflow and upload it here. *

No need to ZIP the file.

Add file

Step 5: Test it Yourself

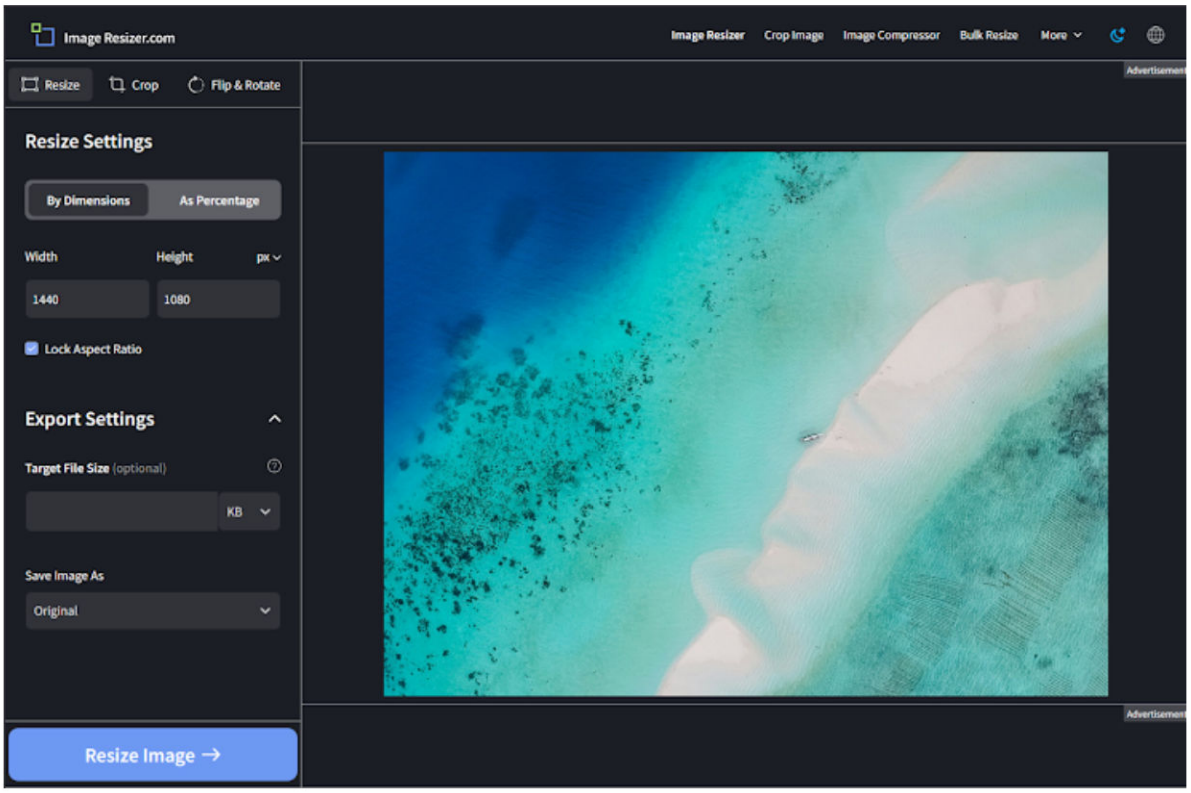
Download a photo of your choice from Unsplash. *

- [ ] Done

To prevent computer crashes and for demonstration purposes, resize the image using this website. *

Set the height to 1080px and lock the aspect ratio.

Example:



- [ ] Done

Download the resized image and save it to your "Original Image" folder. *

Keep only one image in the folder; delete any others.

- [ ] Done

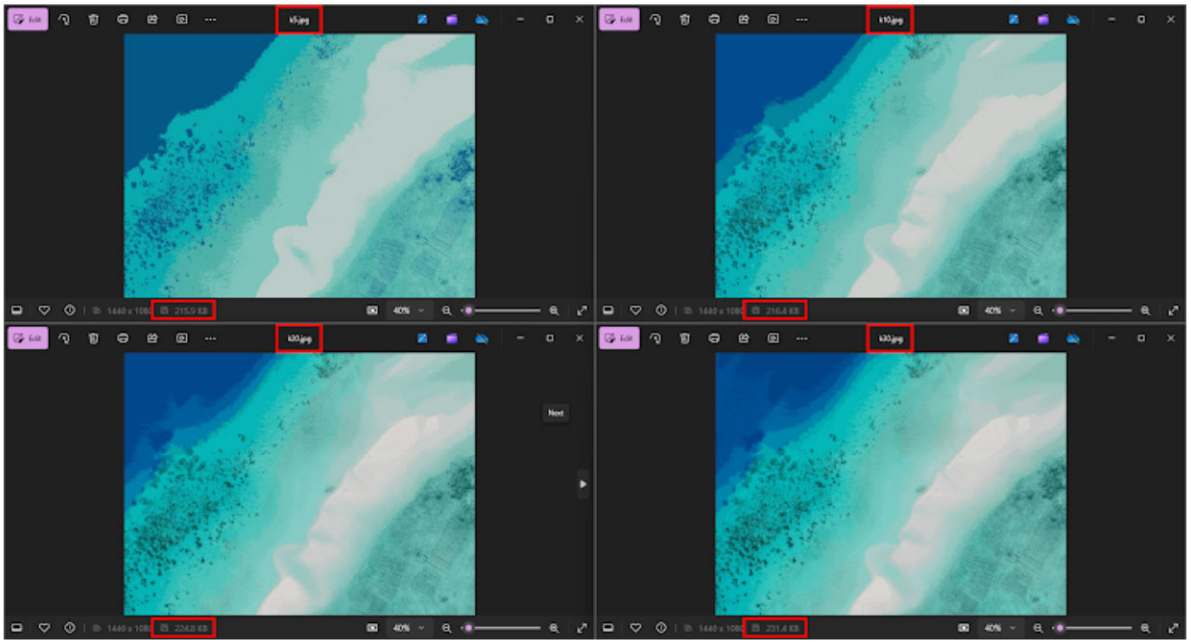
Repeat the process to generate images using 5, 10, 20, and 30 clusters. *

- [ ] Done

Use the 2x2 Snap Layout to view all four images at once. *

Take a full screenshot with the following arrangement:
• Top-left: 5 clusters
• Top-right: 10 clusters
• Bottom-left: 20 clusters
• Bottom-right: 30 clusters

Example:



◯ Done

Check if the file sizes decrease with fewer clusters and are smaller than the original. *

If not, review and adjust your workflow, or try a different photo.

◯ The file sizes decrease with fewer clusters and are smaller than the original.

Upload the screenshot here, **showing that file sizes decrease with fewer clusters and are smaller than the original.** *

Add file

Get link

Never submit passwords through Google Forms.

This form was created inside of Marinduque State College.
Does this form look suspicious? Report

Google Forms