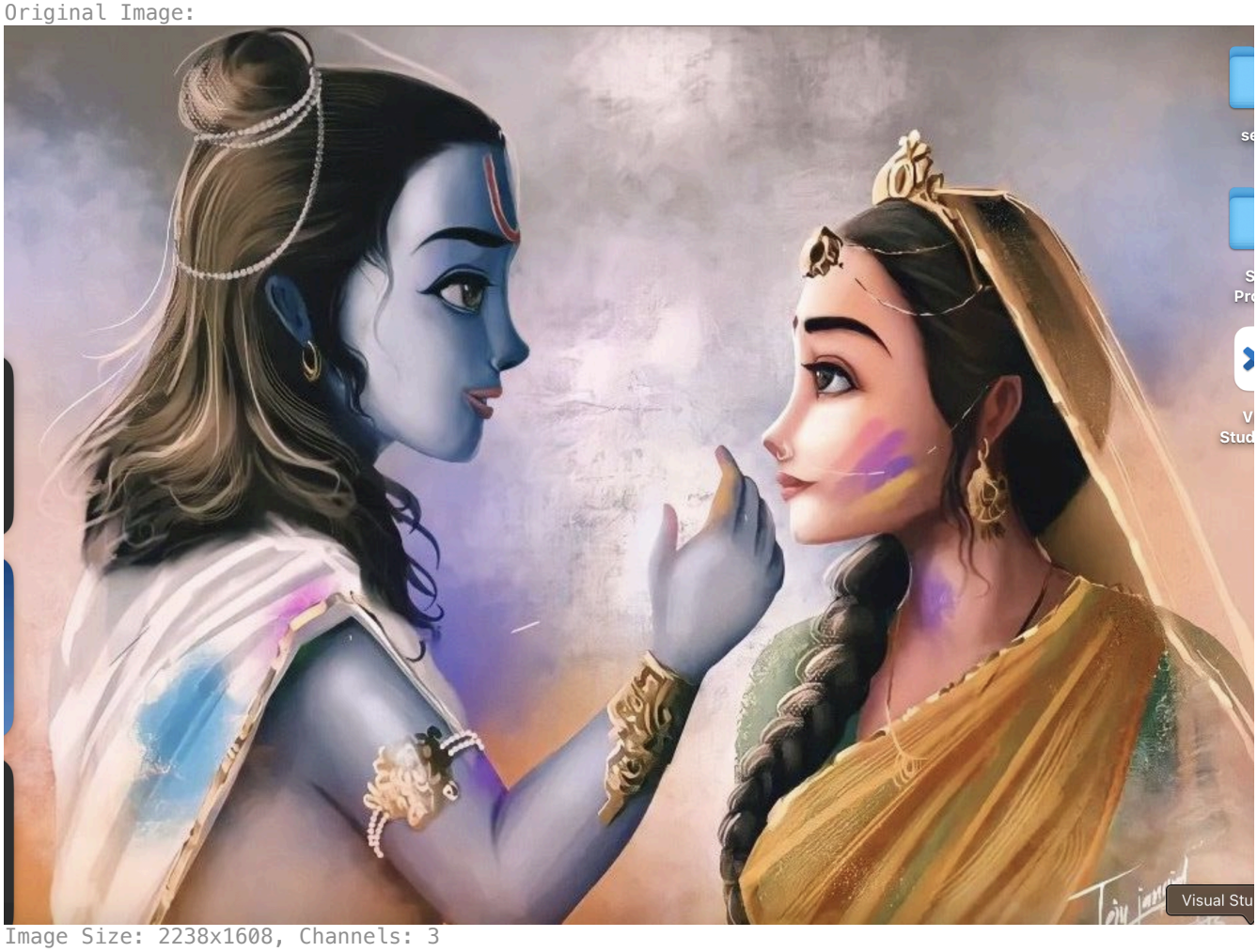
```
#!/usr/bin/env python

import cv2
import numpy as np
from matplotlib import pyplot as plt
from google.colab.patches import cv2_imshow

# Task-1
# 1. Read and display the image
print('Original Image:')
image = cv2.imread('/content/Screenshot 2025-02-13 at 8.38.02 AM.png')
cv2_imshow(image)

# 2. Extract image size
height, width, channels = image.shape
print(f'Image Size: {width}x{height}, Channels: {channels}')

# 3. Calculate image pixels
print(f'Total Pixels: {image.size}')

# 4. Convert BGR to RGB
print('BGR to RGB Conversion:')
rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
cv2_imshow(rgb_image)

# 5. Convert RGB to Grayscale
print('RGB to Grayscale Conversion:')
gray_image = cv2.cvtColor(rgb_image, cv2.COLOR_RGB2GRAY)
cv2_imshow(gray_image)

# 6. Grayscale to Binary using threshold
print('Grayscale to Binary Conversion:')
_, binary_image = cv2.threshold(gray_image, 90, 255, cv2.THRESH_BINARY)
cv2_imshow(binary_image)

# black pixels
black_area = np.sum(binary_image == 0)
print(f'Black Pixel Area: {black_area}, Image Size: {binary_image.shape}')
```

```
Original Image:
```



```
Image Size: 2238x1608, Channels: 3
Total Pixels: 10796112
BGR to RGB Conversion:
```

```
RGB to Grayscale Conversion:
```

Grayscale to Binary Conversion:

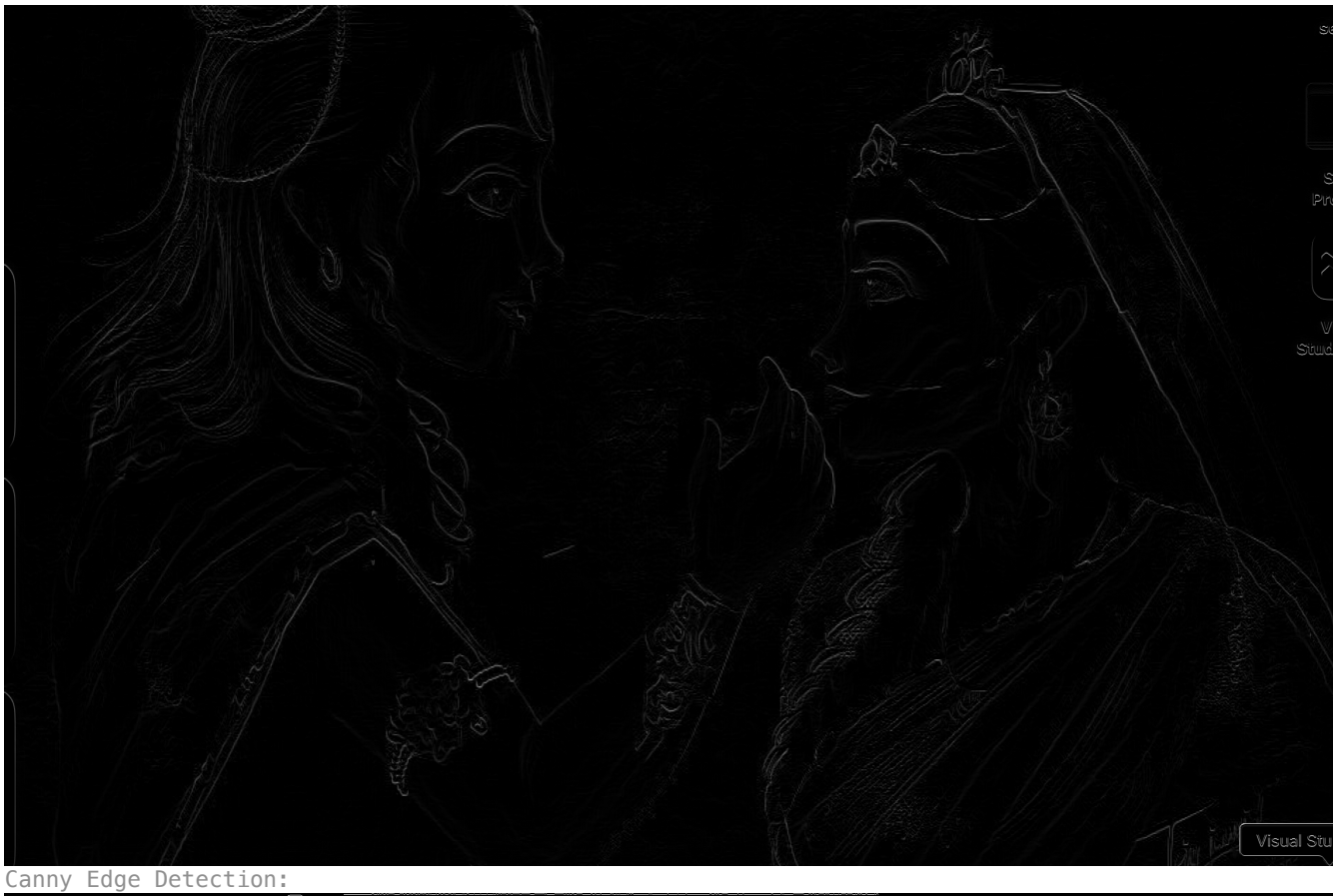
Black Pixel Area: 685290, Image Size: (1608, 2238)

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

# Load the image and convert it to grayscale

image_path = 'path_to_image.jpg'
image = cv2.imread('/content/Screenshot 2025-02-13 at 8.38.02 AM.png')
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Sobel Edge Detection:
print('Sobel Edge Detection:')
sobel_x = cv2.Sobel(gray_image, cv2.CV_64F, 1, 0, ksize=3)  # (gradient in the X direction) #Data type
sobel_y = cv2.Sobel(gray_image, cv2.CV_64F, 0, 1, ksize=3)
sobel = cv2.magnitude(sobel_x, sobel_y)  # ( G = sqrt(gx^2 + gy^2))
cv2_imshow(np.uint8(sobel))  # format(0-255)

# Prewitt Operator
print('Prewitt Edge Detection:')  # (uses a simpler kernel.)
kernelx = np.array([[1, 0, -1], [1, 0, -1], [1, 0, -1]])
kernely = np.array([[1, 1, 1], [0, 0, 0], [-1, -1, -1]])
prewitt_x = cv2.filter2D(gray_image, -1, kernelx)  # to keep the same depth(datatype) as the source ir
prewitt_y = cv2.filter2D(gray_image, -1, kernely)  # detect horizontal edges
prewitt = cv2.add(prewitt_x, prewitt_y)
cv2_imshow(prewitt)

# Roberts Cross Operator (It uses 2×2 kernels.) (grad of adjacent pixels)
print('Roberts Cross Edge Detection:')
roberts_cross_v = np.array([[1, 0], [0, -1]])
roberts_cross_h = np.array([[0, 1], [-1, 0]])
roberts_v = cv2.filter2D(gray_image, -1, roberts_cross_v)
roberts_h = cv2.filter2D(gray_image, -1, roberts_cross_h)
roberts = cv2.add(roberts_v, roberts_h)
cv2_imshow(roberts)

# Canny Edge Detector (multi-stage edge detection algorithm.)
print('Canny Edge Detection:')  # (Noise Reduction → Uses Gaussian filter, Gradient Calculation → Sobe
canny = cv2.Canny(gray_image, 100, 200)  # (Lower threshold,Upper threshold)
cv2_imshow(canny)
```

Sobel Edge Detection:

Prewitt Edge Detection:

Roberts Cross Edge Detection:



Canny Edge Detection: