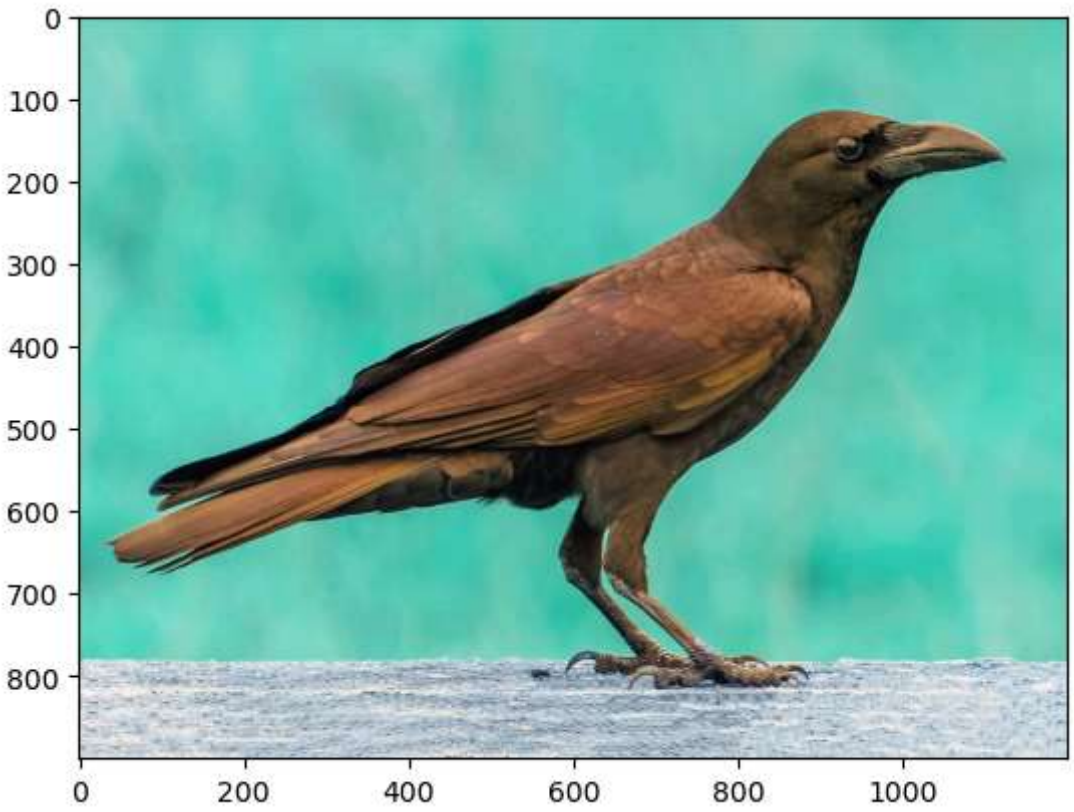
In [1]:
```
pip install opencv-python
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: opencv-python in c:\users\gaurav\appdata\roami
ng\python\python39\site-packages (4.9.0.80)
Requirement already satisfied: numpy>=1.17.0 in c:\users\gaurav\appdata\roami
ng\python\python39\site-packages (from opencv-python) (1.26.4)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
import cv2
import matplotlib.pyplot as plt
import numpy as np
```

In [3]:
```
image_path = 'crow.jpg'

image = cv2.imread(image_path)
plt.imshow(image)
```

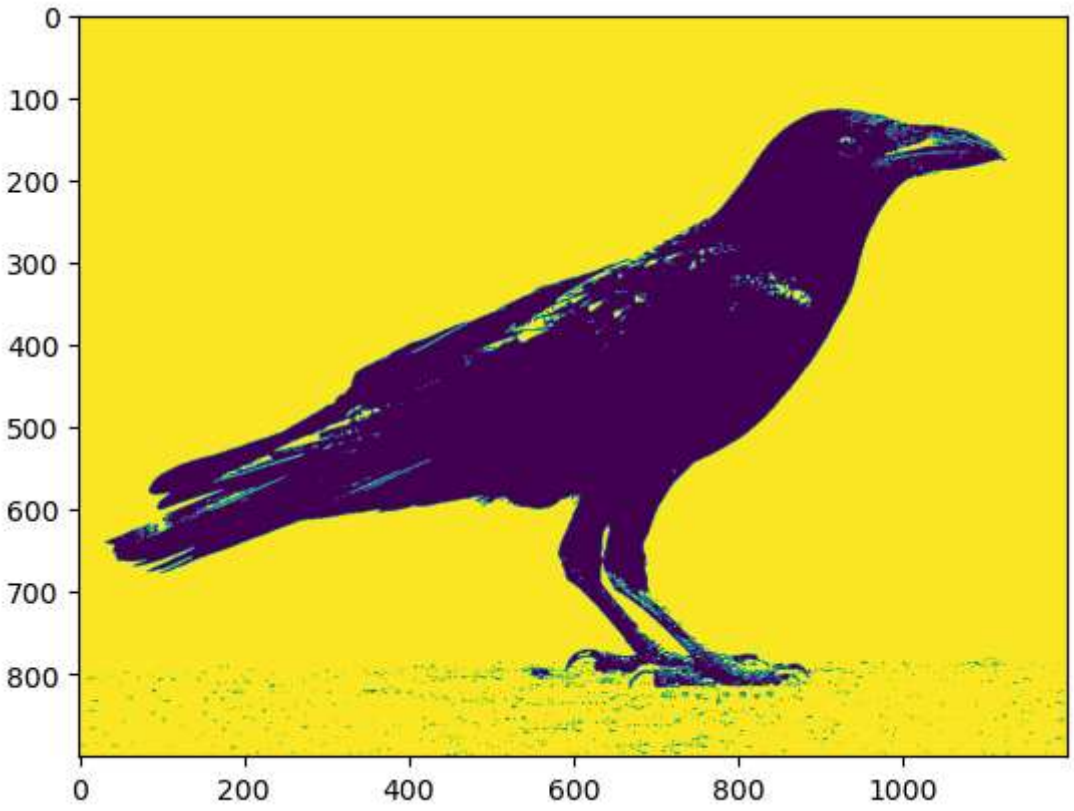
Out[3]:
```
<matplotlib.image.AxesImage at 0x14fe03a1160>
```



In [4]:
```
# Convert the image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Set the threshold value
threshold_value = 120

# Threshold the image to create a binary image
_, binary_image = cv2.threshold(gray,threshold_value,255, cv2.THRESH_BINARY)
```

In [5]:
```
plt.imshow(binary_image)
```

Out[5]:
```
<matplotlib.image.AxesImage at 0x14fe043e370>
```

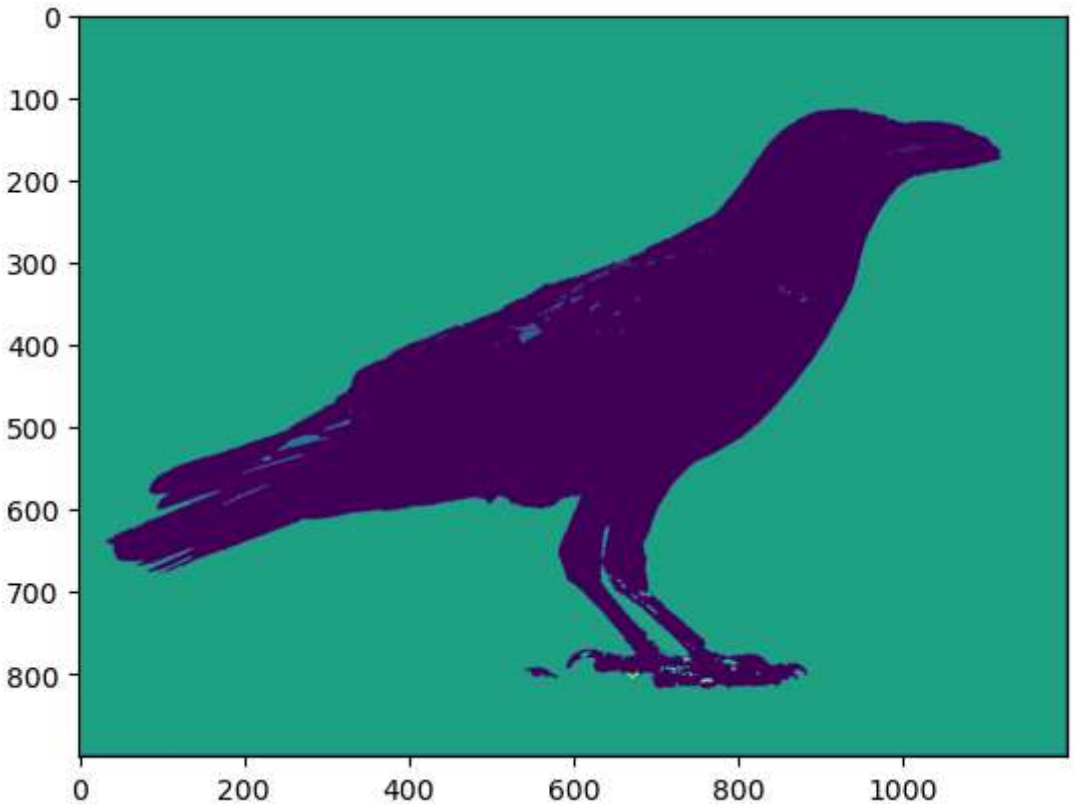


In [6]:

```
# Load the image (assuming 'image' is your input image)
image = cv2.imread('crow.jpg')

# Convert the image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Blur the image to reduce noise
blurred = cv2.GaussianBlur(gray, (5, 5), 0)

# Threshold the image to create a binary image
_, binary_image = cv2.threshold(blurred, 0, 255, cv2.THRESH_BINARY | cv2.THRES

# Create a kernel for the morphological operation
kernel = np.ones((3, 3), np.uint8)

# Perform morphological opening to remove small objects
opening = cv2.morphologyEx(binary_image, cv2.MORPH_OPEN, kernel, iterations=2)

# Create a mask for the background
background_mask = cv2.dilate(opening, kernel, iterations=3)

# Create a mask for the foreground
foreground_mask = cv2.subtract(binary_image, opening)

# Find the markers for the watershed transformation
_, markers = cv2.connectedComponents(foreground_mask)

# Add one to all labels so that the background is not 0, but 1
markers += 1

# Set the background to 0
markers[background_mask == 255] = 0

# Perform the watershed transformation
segmentation = cv2.watershed(image, markers)

# Optionally, you can apply colors to the segmentation result for visualizatio
segmentation_result = np.zeros_like(image)
segmentation_result[segmentation == -1] = [255, 255, 255]  # Mark boundaries i

# Display the segmentation result
cv2.imshow('Segmentation Result', segmentation_result)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [7]:
```
plt.imshow(segmentation)
```

Out[7]:
```
<matplotlib.image.AxesImage at 0x14fe06350a0>
```



In [ ]:
```

```