1. Inverted images

```
inverted_image = cv2.bitwise_not(img)
cv2.imwrite("temp/inverted.jpg", inverted_image)
```

2. Binarization

```
def grayscale(image):
    return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

3. Noise removal

```
def noise_removal(image):
    import numpy as np
    kernel = np.ones((1, 1), np.uint8)
    image = cv2.dilate(image, kernel, iterations=1)
    kernel = np.ones((1, 1), np.uint8)
    image = cv2.erode(image, kernel, iterations=1)
    image = cv2.morphologyEx(image, cv2.MORPH_CLOSE, kernel)
    image = cv2.medianBlur(image, 3)
    return (image)
```

4. Dilation and Erosion

```
def thin_font(image):
    import numpy as np
    image = cv2.bitwise_not(image)
    kernel = np.ones((2,2),np.uint8)
    image = cv2.erode(image, kernel, iterations=1)
    image = cv2.bitwise_not(image)
    return (image)
```

5. Rotation/Deskewing

```
new = cv2.imread("data/page_01_rotated.JPG")
display("data/page_01_rotated.JPG")
```

6. Remove borders

```python
def remove_borders(image):
    contours, heiarchy = cv2.findContours(image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cntsSorted = sorted(contours, key=lambda x:cv2.contourArea(x))
    cnt = cntsSorted[-1]
    x, y, w, h = cv2.boundingRect(cnt)
    crop = image[y:y+h, x:x+w]
    return (crop)
```

7. Missing borders

```python
color = [255, 255, 255]
top, bottom, left, right = [150]*4
```

```python
image_with_border = cv2.copyMakeBorder(no_borders, top, bottom, left, right, cv2.BORDER_CONSTANT, value=color)
cv2.imwrite("temp/image_with_border.jpg", image_with_border)
display("temp/image_with_border.jpg")
```

8. Resize

```
public Bitmap Resize(Bitmap bmp, int newWidth, int newHeight)
    {

        Bitmap temp = (Bitmap)bmp;

        Bitmap bmap = new Bitmap(newWidth, newHeight, temp.PixelFormat);

        double nWidthFactor = (double)temp.Width / (double)newWidth;
        double nHeightFactor = (double)temp.Height / (double)newHeight;

        double fx, fy, nx, ny;
        int cx, cy, fr_x, fr_y;
        Color color1 = new Color();
        Color color2 = new Color();
        Color color3 = new Color();
        Color color4 = new Color();
        byte nRed, nGreen, nBlue;

        byte bp1, bp2;

        for (int x = 0; x < bmap.Width; ++x)
        {
          for (int y = 0; y < bmap.Height; ++y)
          {

            fr_x = (int)Math.Floor(x * nWidthFactor);
            fr_y = (int)Math.Floor(y * nHeightFactor);
            cx = fr_x + 1;
            if (cx >= temp.Width) cx = fr_x;
            cy = fr_y + 1;
            if (cy >= temp.Height) cy = fr_y;
            fx = x * nWidthFactor - fr_x;
```

```
            fy = y * nHeightFactor - fr_y;
            nx = 1.0 - fx;
            ny = 1.0 - fy;

            color1 = temp.GetPixel(fr_x, fr_y);
            color2 = temp.GetPixel(cx, fr_y);
            color3 = temp.GetPixel(fr_x, cy);
            color4 = temp.GetPixel(cx, cy);

            // Blue
            bp1 = (byte)(nx * color1.B + fx * color2.B);

            bp2 = (byte)(nx * color3.B + fx * color4.B);

            nBlue = (byte)(ny * (double)(bp1) + fy * (double)(bp2));

            // Green
            bp1 = (byte)(nx * color1.G + fx * color2.G);

            bp2 = (byte)(nx * color3.G + fx * color4.G);

            nGreen = (byte)(ny * (double)(bp1) + fy * (double)(bp2));

            // Red
            bp1 = (byte)(nx * color1.R + fx * color2.R);

            bp2 = (byte)(nx * color3.R + fx * color4.R);

            nRed = (byte)(ny * (double)(bp1) + fy * (double)(bp2));

            bmap.SetPixel(x, y, System.Drawing.Color.FromArgb
        (255, nRed, nGreen, nBlue));
          }
        }



        bmap = SetGrayscale(bmap);
        bmap = RemoveNoise(bmap);

        return bmap;

    }
```

9. SetGrayscale

```
public Bitmap SetGrayscale(Bitmap img)
        {

          Bitmap temp = (Bitmap)img;
          Bitmap bmap = (Bitmap)temp.Clone();
          Color c;
          for (int i = 0; i < bmap.Width; i++)
          {
            for (int j = 0; j < bmap.Height; j++)
            {
              c = bmap.GetPixel(i, j);
              byte gray = (byte)(.299 * c.R + .587 * c.G + .114 * c.B);




               bmap.SetPixel(i, j, Color.FromArgb(gray, gray, gray));
            }
          }
          return (Bitmap)bmap.Clone();

        }
```

10. RemoveNoise

```
public Bitmap RemoveNoise(Bitmap bmap)
        {

          for (var x = 0; x < bmap.Width; x++)
          {
            for (var y = 0; y < bmap.Height; y++)
            {
              var pixel = bmap.GetPixel(x, y);
              if (pixel.R < 162 && pixel.G < 162 && pixel.B < 162)
                bmap.SetPixel(x, y, Color.Black);
              else if (pixel.R > 162 && pixel.G > 162 && pixel.B > 162)
                bmap.SetPixel(x, y, Color.White);
            }
          }

          return bmap;
        }
```

11. GaussianBlur, bilateralFilter, medianBlur

```
cv2.threshold(cv2.GaussianBlur(img, (5, 5), 0), 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1]

cv2.threshold(cv2.bilateralFilter(img, 5, 75, 75), 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1]

cv2.threshold(cv2.medianBlur(img, 3), 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1]

cv2.adaptiveThreshold(cv2.GaussianBlur(img, (5, 5), 0), 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY, 31, 2)

cv2.adaptiveThreshold(cv2.bilateralFilter(img, 9, 75, 75), 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY, 31, 2)

cv2.adaptiveThreshold(cv2.medianBlur(img, 3), 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY, 31, 2)
```