7. Perform transformations on image using hardcoding with matrix multiplication of image and matrices of transformation

In [68]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import math as mp
img=cv2.imread("edge.png",0)
img=cv2.resize(img,(400,400))
img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
```

In [69]:
```
def translation(img,x,y):
    w,h=img.shape[:2]
    img1=np.zeros_like(img,dtype=np.uint8)
    kernel=np.array([[1,0,x],[0,1,y],[0,0,1]])
    for i in range(w):
        for j in range(h):
            mat1=np.array([[i],[j],[1]])
            result=np.dot(kernel,mat1)
            m=int(result[0][0])
            n=int(result[1][0])
            if m<w and n<h:
                img1[m][n]=img[i][j]
    return img1
trans=translation(img,100,50)
```

In [70]:
```
def shearing(img,x,y):
    w,h=img.shape[:2]
    img1=np.zeros_like(img,dtype=np.uint8)
    kernel=np.array([[1,x,0],[y,1,0],[0,0,1]])
    for i in range(w):
        for j in range(h):
            mat1=np.array([[i],[j],[1]])
            result=np.dot(kernel,mat1)
            m=int(result[0][0])
            n=int(result[1][0])
            if 0 <= m < w and 0 <= n < h:
                img1[m][n] = img[i][j]
    return img1
shear=shearing(img,0.2,0.5)
```

In [71]:
```
def scaling(img,x,y):
    w,h=img.shape[:2]
    img1=np.zeros_like(img,dtype=np.uint8)
    kernel=np.array([[x,0,1],[0,y,1],[0,0,1]])
    for i in range(w):
        for j in range(h):
            mat1=np.array([[i],[j],[1]])
            result=np.dot(kernel,mat1)
            m=int(result[0][0])
            n=int(result[1][0])
            if 0 <= m < w and 0 <= n < h:
```

```
                img1[m][n] = img[i][j]
    return img1
scale=scaling(img,2,1)
```

In [72]:
```
def Rotation(img,angle):
    w,h=img.shape[:2]
    x=mp.cos(mp.radians(angle))
    y=mp.sin(mp.radians(angle))
    img1=np.zeros_like(img,dtype=np.uint8)
    kernel=np.array([[x,y,0],[-y,x,0],[0,0,1]])
    for i in range(w):
        for j in range(h):
            mat1=np.array([[i],[j],[1]])
            result=np.dot(kernel,mat1)
            m=int(result[0][0])
            n=int(result[1][0])
            if 0 <= m < w and 0 <= n < h:
                img1[m][n] = img[i][j]
    return img1
rotate=Rotation(img,30)
```

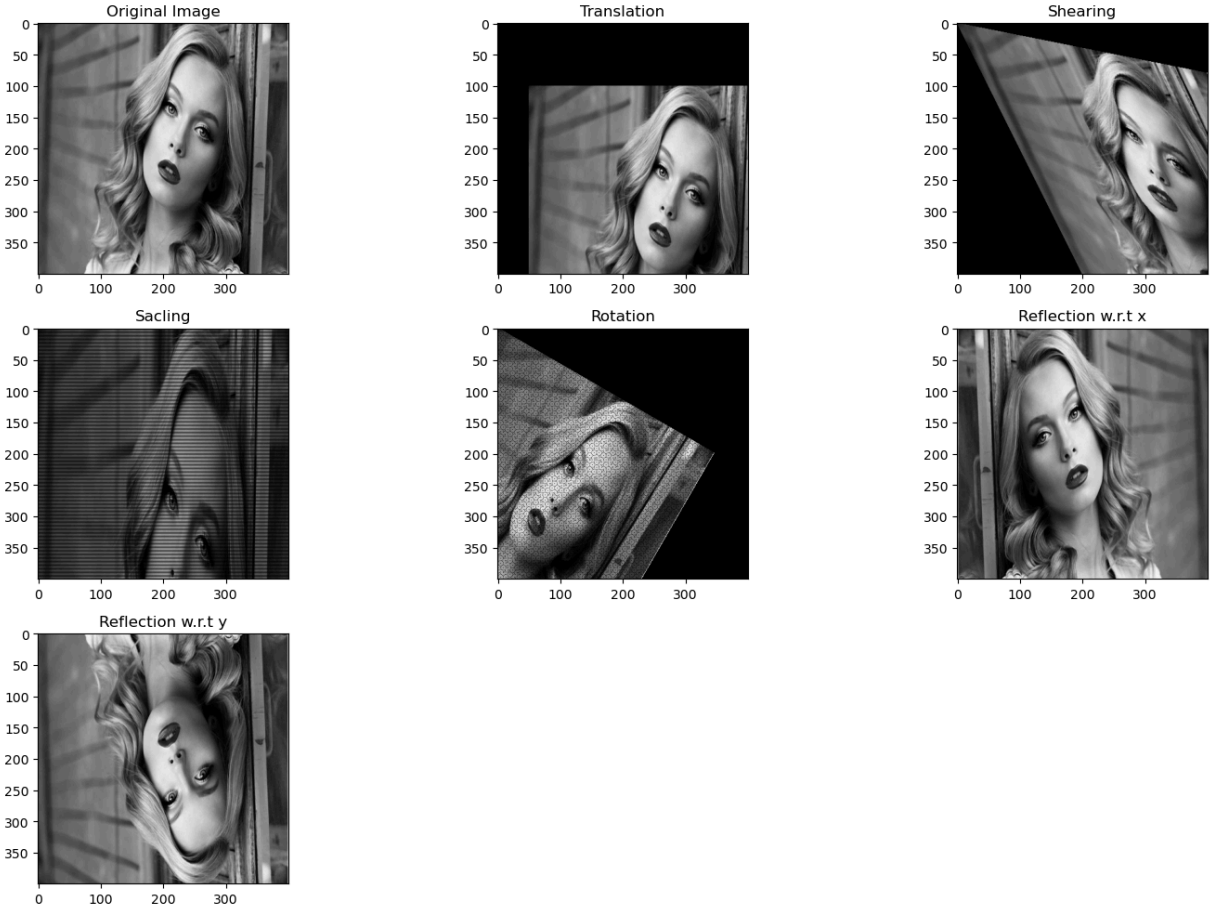
In [73]:
```
def Reflection_x(img):
    w,h=img.shape[:2]
    img1=np.zeros_like(img,dtype=np.uint8)
    kernel=np.array([[1,0,0],[0,-1,w],[0,0,1]])
    for i in range(w):
        for j in range(h):
            mat1=np.array([[i],[j],[1]])
            result=np.dot(kernel,mat1)
            m=int(result[0][0])
            n=int(result[1][0])
            if 0 <= m < w and 0 <= n < h:
                img1[m][n] = img[i][j]
    return img1
reflect_x=Reflection_x(img)
```

In [74]:
```
def Reflection_y(img):
    w,h=img.shape[:2]
    img1=np.zeros_like(img,dtype=np.uint8)
    kernel=np.array([[-1,0,h],[0,1,0],[0,0,1]])
    for i in range(w):
        for j in range(h):
            mat1=np.array([[i],[j],[1]])
            result=np.dot(kernel,mat1)
            m=int(result[0][0])
            n=int(result[1][0])
            if 0 <= m < w and 0 <= n < h:
                img1[m][n] = img[i][j]
    return img1
reflect_y=Reflection_y(img)
```

In [75]:
```
plt.figure(figsize=(15,10))
plt.subplot(3, 3, 1), plt.imshow(img, cmap='gray'), plt.title('Original Image')
plt.subplot(3, 3, 2), plt.imshow(trans, cmap='gray'), plt.title('Translation')
plt.subplot(3, 3, 3), plt.imshow(shear, cmap='gray'), plt.title('Shearing')
```

```
plt.subplot(3, 3, 4), plt.imshow(scale, cmap='gray'), plt.title('Sacling')
plt.subplot(3, 3, 5), plt.imshow(rotate, cmap='gray'), plt.title('Rotation')
plt.subplot(3, 3, 6), plt.imshow(reflect_x, cmap='gray'), plt.title('Reflection w.r
plt.subplot(3, 3, 7), plt.imshow(reflect_y, cmap='gray'), plt.title('Reflection w.r

plt.tight_layout()
plt.show()
```



In [ ]:

In [ ]: