EXPT 3	**Histogram Equalization**

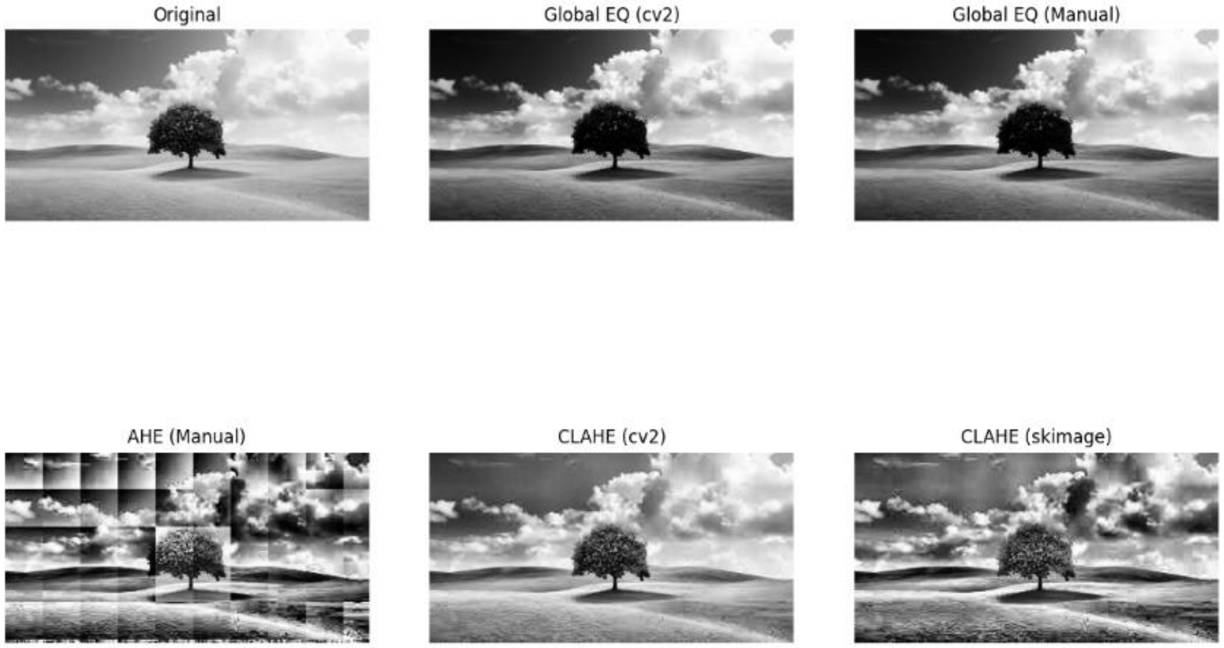
AIM:-

To implement histogram equalization techniques including global histogram equalization, adaptive histogram equalization (AHE), and contrast limited adaptive histogram equalization (CLAHE) to improve image contrast and visibility.

CODE:-

```python
# Necessary imports
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
from skimage import io, exposure, color

# Load image as grayscale - upload 'input.jpg' file first
img_path = 'input.jpg'
img_gray = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)

if img_gray is None:
    print("Image not found. Please upload 'input.jpg' to Colab.")
else:
    # Ensure all operations use the single grayscale image loaded with cv2
    # A. Global Histogram Equalization

    # USING WITH CV2 PACKAGE
    equalized_img_cv2 = cv2.equalizeHist(img_gray)

    # USING WITHOUT CV2 PACKAGE (using numpy on the cv2 loaded image)
    img_np_from_cv2 = np.array(img_gray) # Convert cv2 image to numpy array for manual processing
    # Calculate histogram
    hist, bins = np.histogram(img_np_from_cv2.flatten(), 256, [0,256])

    # Calculate normalized cumulative distribution function (CDF)
    cdf = hist.cumsum()
    # Avoid division by zero if the image is uniform
    cdf_normalized = cdf * 255 / (cdf[-1] if cdf[-1] > 0 else 1)


    # Use linear interpolation of cdf to find new pixel values
    img_flat = img_np_from_cv2.flatten()
    img_equalized_flat_manual = np.interp(img_flat, bins[:-1], cdf_normalized)
    equalized_img_manual =
img_equalized_flat_manual.reshape(img_np_from_cv2.shape).astype(np.uint8)


    # B. Adaptive Histogram Equalization (Manual Implementation)

    def adaptive_hist_eq(img, tile_size=32):
        h, w = img.shape
        out_img = np.zeros_like(img)
        n_tiles_x = w // tile_size + (1 if w % tile_size != 0 else 0) # handle edge cases
        n_tiles_y = h // tile_size + (1 if h % tile_size != 0 else 0) # handle edge cases

        for i in range(n_tiles_y):
            for j in range(n_tiles_x):
                x_start = j * tile_size
                y_start = i * tile_size
                x_end = min(x_start + tile_size, w)
                y_end = min(y_start + tile_size, h)

                tile = img[y_start:y_end, x_start:x_end]
                hist, bins = np.histogram(tile.flatten(), 256, [0,256])
                cdf = hist.cumsum()
                # Avoid division by zero if a tile is uniform
                cdf_normalized = cdf * 255 / (cdf[-1] if cdf[-1] > 0 else 1)
                tile_flat = tile.flatten()
                tile_eq_flat = np.interp(tile_flat, bins[:-1], cdf_normalized)
                tile_eq = tile_eq_flat.reshape(tile.shape).astype(np.uint8)

                out_img[y_start:y_end, x_start:x_end] = tile_eq

        return out_img

    ahe_manual = adaptive_hist_eq(img_gray, tile_size=32) # Use the cv2 loaded image


    # C. Contrast Limited Adaptive Histogram Equalization (CLAHE)

    # USING WITH CV2 PACKAGE
    clahe_cv2_obj = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
    clahe_img_cv2 = clahe_cv2_obj.apply(img_gray) # Use the cv2 loaded image

    # USING WITHOUT CV2 PACKAGE (using skimage)
    # Skimage's equalize_adapthist expects float or uint8.
    # The cv2 image is already uint8 grayscale.
    clahe_img_skimage = exposure.equalize_adapthist(img_gray, clip_limit=0.03)

    # Convert back to 8-bit for display as skimage's output is float by default
    clahe_img_skimage_8bit = (clahe_img_skimage * 255).astype(np.uint8)

    # Display images in a 2x5 grid
    fig, axes = plt.subplots(2, 5, figsize=(20, 8)) # Create a 2x5 grid of subplots
    axes = axes.ravel() # Flatten the axes array for easy iteration

    images = [img_gray, equalized_img_cv2, equalized_img_manual, ahe_manual, clahe_img_cv2, clahe_img_skimage_8bit]
    titles = ['Original', 'Global EQ (cv2)', 'Global EQ (Manual)', 'AHE (Manual)', 'CLAHE (cv2)', 'CLAHE (skimage)']

    # Plot the first 6 images in a 2x3 arrangement within the 2x5 grid
    for i in range(len(images)):
        row = i // 3  # Determine the row (0 or 1)
        col = i % 3  # Determine the column (0, 1, or 2)
        axes[row * 5 + col].imshow(images[i], cmap='gray') # Use the correct index in the flattened array
        axes[row * 5 + col].set_title(titles[i])
        axes[row * 5 + col].axis('off') # Turn off axes

    # Turn off the remaining unused subplots
    for i in range(len(images), len(axes)):
        axes[i].axis('off')

    plt.tight_layout() # Adjust layout to prevent titles overlapping
    plt.show() # Display the figure
```

OUTPUT:-



RESULT:-

Successfully enhanced image contrast using different histogram equalization methods. CLAHE proved most effective in preserving local details while avoiding over-amplification of noise compared to global and standard adaptive methods.