Name: Vivek Vitthal Avhad (3059)

In [1]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
image1 = cv2.imread(r'Ex2.png')
```

In [3]:
```
gray_image = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)
```

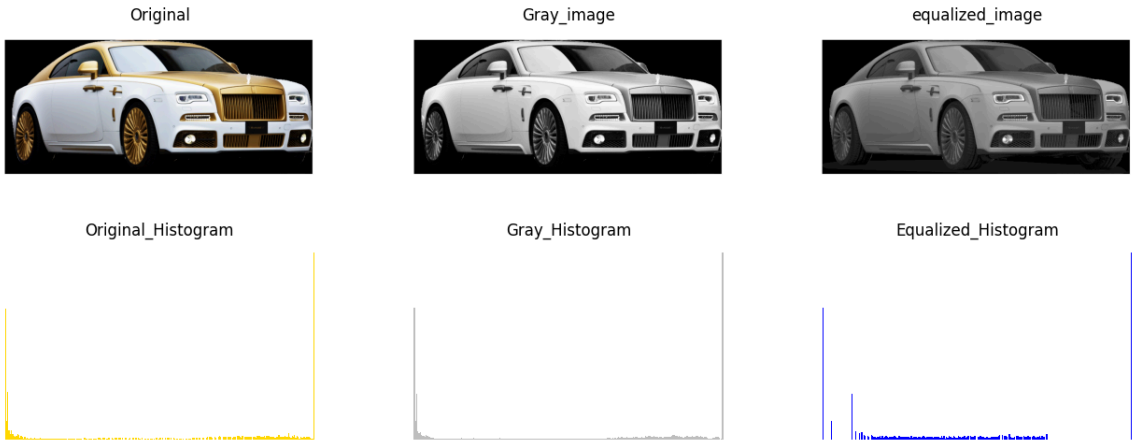
In [4]:
```
equalized_image=cv2.equalizeHist(gray_image)
plt.figure(figsize=(12,6))
```

Out[4]:
```
<Figure size 1200x600 with 0 Axes>
```

```
<Figure size 1200x600 with 0 Axes>
```

In [5]:
```
plt.figure(figsize=(16,6))
plt.subplot(2,3,1)
plt.title("Original")
plt.imshow(cv2.cvtColor(image1, cv2.COLOR_BGR2RGB))
plt.axis('off')

plt.subplot(2,3,2)
plt.title("Gray_image")
plt.imshow(cv2.cvtColor(gray_image, cv2.COLOR_BGR2RGB))
plt.axis('off')

plt.subplot(2,3,3)
plt.title("equalized_image")
plt.imshow(cv2.cvtColor(equalized_image, cv2.COLOR_BGR2RGB))
plt.axis('off')

plt.subplot(2,3,4)
plt.title("Original_Histogram")
plt.hist(image1.ravel(),bins=256,range=(0,256),color='gold')
plt.axis('off')

plt.subplot(2,3,5)
plt.title("Gray_Histogram")
plt.hist(gray_image.ravel(),bins=256,range=(0,256),color='silver')
plt.axis('off')

plt.subplot(2,3,6)
plt.title("Equalized_Histogram")
plt.hist(equalized_image.ravel(),bins=256,range=(0,256),color='blue')
plt.axis('off')
```

Out[5]:
```
(-12.8, 268.8, 0.0, 84281.4)
```



1. Apply Histogram Equalization on a colored image by converting it to YCrCb color space.

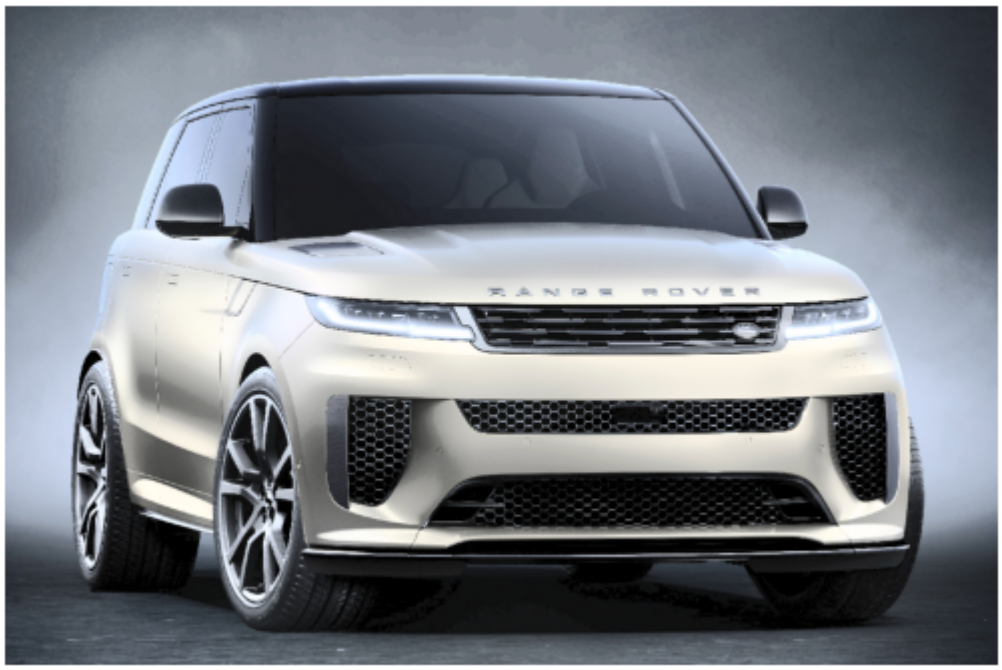
In [6]:
```
image2 = cv2.imread(r'Que.jpg')
```

In [7]:
```
# Convert image to YCrCb color space and apply Histogram Equalization on the Y c
def global_histogram_equalization(image):
    ycrcb_img = cv2.cvtColor(image, cv2.COLOR_BGR2YCrCb)
    ycrcb_img[:, :, 0] = cv2.equalizeHist(ycrcb_img[:, :, 0])
    equalized_img = cv2.cvtColor(ycrcb_img, cv2.COLOR_YCrCb2BGR)
    return equalized_img
```

In [8]:
```
# Apply Global Histogram Equalization
global_eq_img = global_histogram_equalization(image2)
```

In [9]:
```
plt.imshow(cv2.cvtColor(global_eq_img, cv2.COLOR_BGR2RGB))
plt.axis('off')
```

Out[9]: (-0.5, 1199.5, 794.5, -0.5)



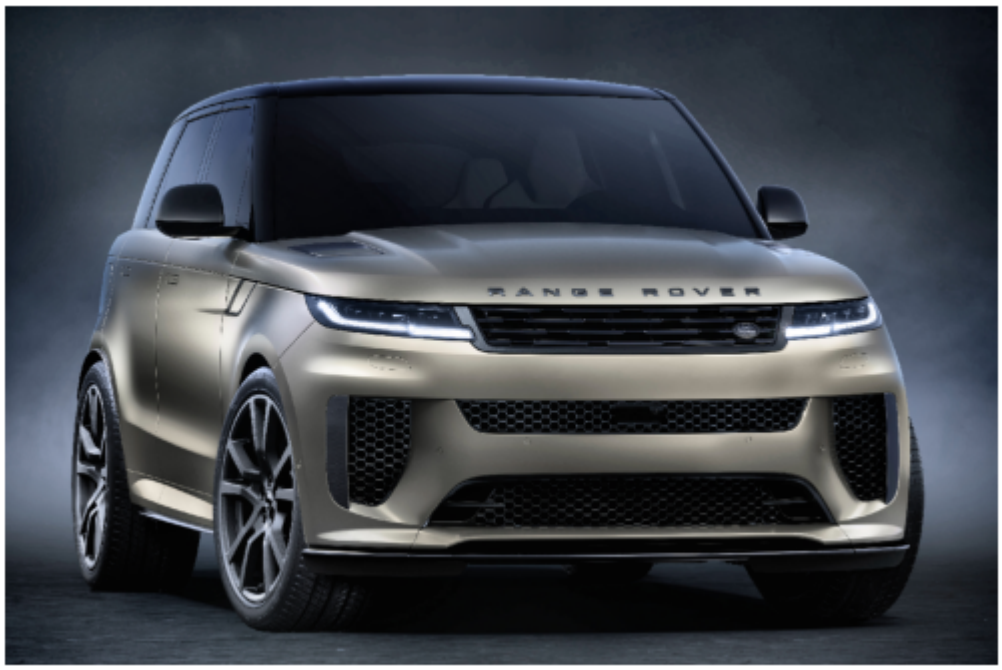
2. Implement Adaptive Histogram Equalization using OpenCV's cv2.createCLAHE() function.

In [10]:
```
# Adaptive Histogram Equalization using OpenCV's cv2.createCLAHE() function
def adaptive_histogram_equalization(image):
    ycrcb_img = cv2.cvtColor(image, cv2.COLOR_BGR2YCrCb)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    ycrcb_img[:, :, 0] = clahe.apply(ycrcb_img[:, :, 0])
    equalized_img = cv2.cvtColor(ycrcb_img, cv2.COLOR_YCrCb2BGR)
    return equalized_img
```

In [11]:
```
# Apply Adaptive Histogram Equalization
adaptive_eq_img = adaptive_histogram_equalization(image2)
```

In [12]:
```
plt.imshow(cv2.cvtColor(adaptive_eq_img, cv2.COLOR_BGR2RGB))
plt.axis('off')
```

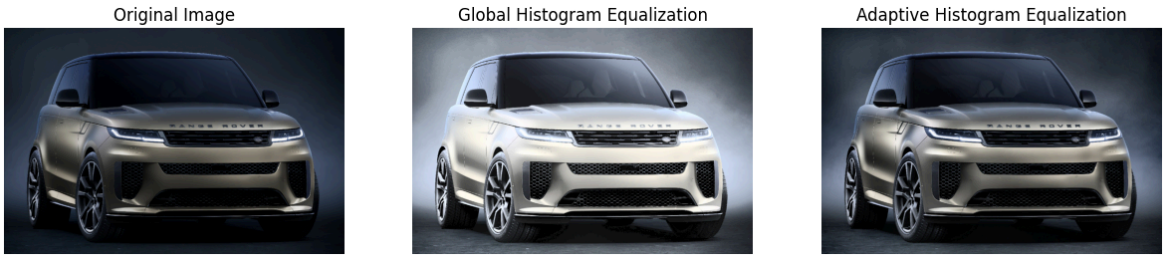
Out[12]: (-0.5, 1199.5, 794.5, -0.5)



3. Compare the results of Global and Adaptive Histogram Equalization.

In [13]:
```
# Plot the original image, Global Histogram Equalization, and Adaptive Histogram
plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
plt.imshow(cv2.cvtColor(image2, cv2.COLOR_BGR2RGB))
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(cv2.cvtColor(global_eq_img, cv2.COLOR_BGR2RGB))
plt.title('Global Histogram Equalization')
plt.axis('off')

plt.subplot(1, 3, 3)
```

```
plt.imshow(cv2.cvtColor(adaptive_eq_img, cv2.COLOR_BGR2RGB))
plt.title('Adaptive Histogram Equalization')
plt.axis('off')

plt.show()
```



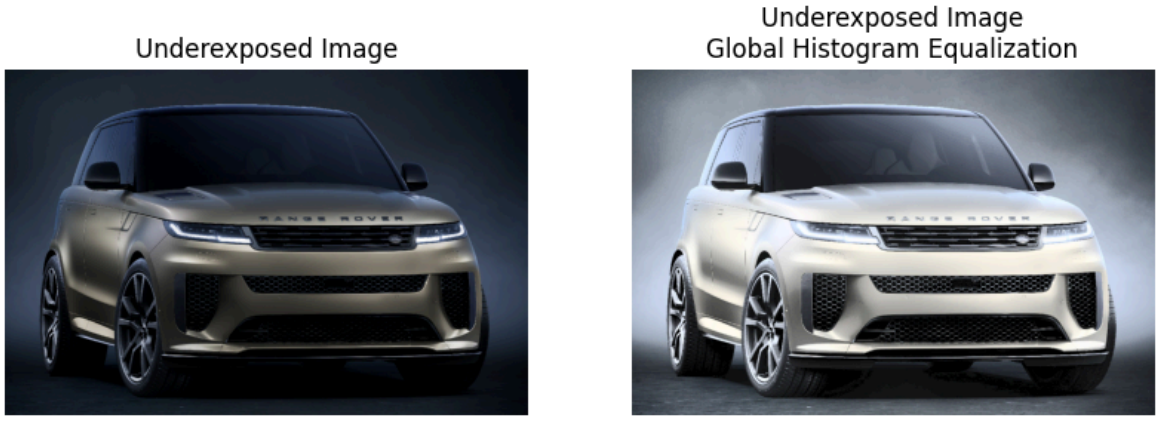
4. Perform Histogram Equalization on an underexposed image and observe the changes.

In [14]:
```
underexposed_image = cv2.imread(r'Que.jpg')
```

In [15]:
```
def global_histogram_equalization(image):
    ycrcb_img = cv2.cvtColor(image, cv2.COLOR_BGR2YCrCb)
    ycrcb_img[:, :, 0] = cv2.equalizeHist(ycrcb_img[:, :, 0])
    equalized_img = cv2.cvtColor(ycrcb_img, cv2.COLOR_YCrCb2BGR)
    return equalized_img
```

In [16]:
```
underexposed_global_eq_img = global_histogram_equalization(underexposed_image)
```

In [17]:
```
# Plot the underexposed image and the result of Global Histogram Equalization
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(underexposed_image, cv2.COLOR_BGR2RGB))
plt.title('Underexposed Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(underexposed_global_eq_img, cv2.COLOR_BGR2RGB))
plt.title('Underexposed Image\nGlobal Histogram Equalization')
plt.axis('off')

plt.show()
```



5. Write a function to implement Histogram Equalization manually without using OpenCV's built- in function.

In [18]:
```
image3 = cv2.imread('Que.jpg', cv2.IMREAD_GRAYSCALE)
```

In [19]:
```
# Manually implement Histogram Equalization
def manual_histogram_equalization(image):
    # Compute the histogram of the input image
    hist, bins = np.histogram(image.flatten(), 256, [0, 256])

    # Compute the cumulative distribution function (CDF)
    cdf = hist.cumsum()
    cdf_normalized = 255 * (cdf - cdf.min()) / (cdf.max() - cdf.min())

    # Use linear interpolation of the CDF to find new pixel values
    image_equalized = np.interp(image.flatten(), bins[:-1], cdf_normalized)

    # Reshape the flattened image back to the original image shape
    image_equalized = image_equalized.reshape(image.shape)

    return image_equalized
```

In [20]:
```
# Apply the manual Histogram Equalization
manual_eq_img = manual_histogram_equalization(image3)
```

In [21]:
```
# Save and display the result
cv2.imwrite('manual_histogram_equalized_image.jpg', manual_eq_img)
plt.figure(figsize=(10, 5))
plt.title('Manual Histogram Equalization')
plt.imshow(manual_eq_img, cmap='gray')
plt.axis('off')
plt.show()
```

Manual Histogram Equalization