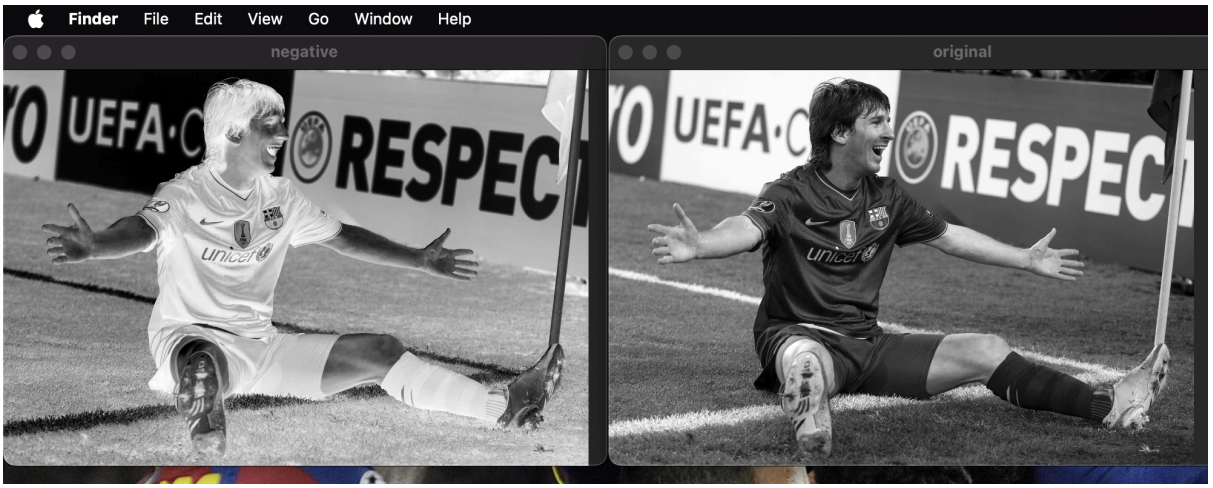
QUESTION 3 - Improve the image quality of the attached (Contrast.png image) using intensity level transformations of your choice.
(I have used my own image here)

```
In [6]: import cv2
        import numpy as np
        myimage = cv2.imread("young messi.jpeg",0)

In [7]: #NEGATIVE TRANSFORMATION

        h,w = myimage.shape
        max = -1
        for i in range(h):
            for j in range(w):
                if myimage[i][j]>max:
                    max = myimage[i][j]
        negative = max-myimage

In [8]: cv2.imshow("original",myimage)
        cv2.imshow("negative",negative)
        cv2.startWindowThread()
        print(cv2.waitKey(0))
        cv2.destroyAllWindows()
        for i in range(2):
            cv2.waitKey(1)

        13

In [ ]:
```



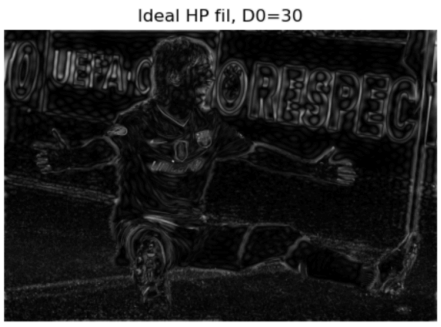
QUESTION 2 - Perform image sharpening on an image using Ideal High pass filter and Butterworth filter (Use different cut off frequency D0= 15, 30, 80).

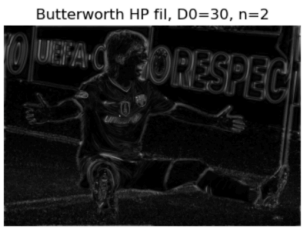
In [27]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread('young messi.jpeg', 0)  #reading image in grayscale
```

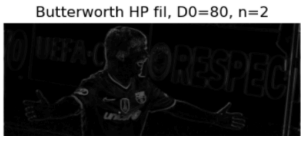
In [28]:
```
def ideal_high_pass_filter(img, D0):
    rows, cols = img.shape
    crow, ccol = rows // 2 , cols // 2

        #Performing Fourier Transform
    dft = np.fft.fft2(img)
    dft_shift = np.fft.fftshift(dft)

        #creating ideal high pass filter
    mask = np.ones((rows, cols), np.float32)
    for i in range(rows):
        for j in range(cols):
            if np.sqrt((i-crow)**2 + (j-ccol)**2) <= D0:
                mask[i, j] = 0

        #applying the mask
    filtered_dft_shift = dft_shift * mask

    #now inversing fft
    filtered_dft = np.fft.ifftshift(filtered_dft_shift)
    filtered_img = np.fft.ifft2(filtered_dft)
    filtered_img = np.abs(filtered_img)

    return filtered_img


def butterworth_high_pass_filter(img,D0,n):
    rows, cols = img.shape
    crow, ccol = rows // 2 , cols // 2

        #Performing Fourier Transform
    dft = np.fft.fft2(img)
    dft_shift = np.fft.fftshift(dft)

    #creating butterworth high pass filter
    mask = np.zeros((rows, cols), np.float32)
    for i in range(rows):
        for j in range(cols):
            distance = np.sqrt((i-crow)**2 + (j-ccol)**2)
            if distance == 0:
                mask[i, j] = 0
            else:
                mask[i, j] = 1 / (1 + (D0 / distance)**(2 * n))

    #applying mask
    filtered_dft_shift = dft_shift * mask

    #performing inverse fft
    filtered_dft = np.fft.ifftshift(filtered_dft_shift)
    filtered_img = np.fft.ifft2(filtered_dft)
    filtered_img = np.abs(filtered_img)

    return filtered_img
```

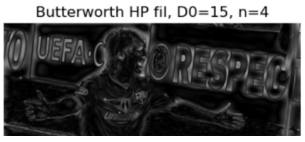
In [29]:
```
D0_values = [15, 30, 80] #Provided Cutoff frequencies

plt.figure(figsize=(12, 8))
plt.subplot(2, 2, 1), plt.imshow(img, cmap='gray')
plt.title('Original Image'), plt.axis('off')

#IDEAL HP FILTER
for i, D0 in enumerate(D0_values, 2):
    filtered_img = ideal_high_pass_filter(img, D0)
```

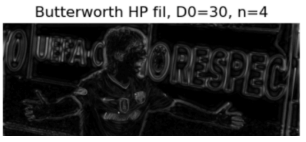
```
    plt.subplot(2, 2, i), plt.imshow(filtered_img, cmap='gray')
    plt.title(f'Ideal HP fil, D0={D0}'), plt.axis('off')

orders = [2, 4]
plt.figure(figsize=(12, 8))
plt.subplot(3, 3, 1), plt.imshow(img, cmap='gray')
plt.title('Original Image'), plt.axis('off')

#BUTTERWORTH HP FILTER
subplot_index = 2
for order in orders:
    for D0 in D0_values:
        filtered_img = butterworth_high_pass_filter(img, D0, order)
        plt.subplot(3, 3, subplot_index), plt.imshow(filtered_img, cmap='gray')
        plt.title(f'Butterworth HP fil, D0={D0}, n={order}'), plt.axis('off')
        subplot_index += 1

plt.tight_layout()
plt.show()

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











Butterworth HP fil, D0=80, n=2

Butterworth HP fil, D0=15, n=4

Butterworth HP fil, D0=30, n=4

Butterworth HP fil, D0=80, n=4

<Figure size 640x480 with 0 Axes>

In [ ]:

In [ ]: