**Question 2**

sampling\_rate\_1 = 120 # This is above the Nyquist rate

sampling\_rate\_2 = 40 # This is below the Nyquist rate

frequency = 45

Defining the sampling rates and signal frequency: The next few lines define the sampling rates and the frequency of the signal. In this case, we’re using a signal with a frequency of 45 Hz, and we’re sampling it at two different rates: 120 Hz and 40 Hz.

t1 = np.arange(0, 1, 1/sampling\_rate\_1)

t2 = np.arange(0, 1, 1/sampling\_rate\_2)

Defining the time axis: The np.arange() function is used to generate the time axis for the signal. It creates an array of evenly spaced values over a specified range. The range is defined by the start value, the stop value, and the step size. In this case, the start value is 0, the stop value is 1, and the step size is the reciprocal of the sampling rate (i.e., the time interval between samples).

signal1 = np.sin(2 \* np.pi \* frequency \* t1)

signal2 = np.sin(2 \* np.pi \* frequency \* t2)

Generating the signal: The np.sin() function is used to generate the signal. This function generates a sine wave with a frequency specified by the frequency variable. The 2 \* np.pi \* frequency \* t gives the angle in radians for each point in time t.

fft1 = np.fft.fft(signal1)

fft2 = np.fft.fft(signal2)

Computing the FFT: The np.fft.fft() function is used to compute the Fast Fourier Transform (FFT) of the signal. The FFT is a method for computing the DFT that is much faster than direct computation of the DFT formula. The DFT converts a sequence of values (in this case, the signal) into a list of coefficients of a finite combination of complex sinusoids, ordered by their frequencies.

freq1 = np.fft.fftfreq(len(fft1), 1/sampling\_rate\_1)

freq2 = np.fft.fftfreq(len(fft2), 1/sampling\_rate\_2)

Computing the frequency axis for the FFT: The np.fft.fftfreq() function is used to compute the frequency axis for the FFT. It returns an array giving the frequencies of corresponding elements in the output. The frequency is expressed in cycles per time unit and is always in the range [-n/2, n/2), where n is the length of the input signal.

plt.figure(figsize=(12, 8))

plt.subplot(2, 2, 1)

plt.plot(t1, signal1)

plt.title('Signal sampled at 120 Hz')

plt.xlabel('Time (s)')

plt.ylabel('Amplitude')

plt.subplot(2, 2, 2)

plt.plot(t2, signal2)

plt.title('Signal sampled at 40 Hz')

plt.xlabel('Time (s)')

plt.ylabel('Amplitude')

plt.subplot(2, 2, 3)

plt.plot(freq1, np.abs(fft1))

plt.title('Magnitude Spectrum (120 Hz sampling rate)')

plt.xlabel('Frequency (Hz)')

plt.ylabel('Magnitude')

plt.subplot(2, 2, 4)

plt.plot(freq2, np.abs(fft2))

plt.title('Magnitude Spectrum (40 Hz sampling rate)')

plt.xlabel('Frequency (Hz)')

plt.ylabel('Magnitude')

plt.tight\_layout()

plt.show()

Plotting the signals and their magnitude spectra: The plt.plot() function is used to plot the signals and their magnitude spectra. The np.abs() function is used to compute the absolute values of the FFT outputs, which gives the magnitude spectrum of the signal.