**Explanation (Matplotlib):**

1. **Import Libraries:** Import streamlit, numpy, scipy.stats, and matplotlib.pyplot.
2. **Sample Data:** Create or load your data. The example uses np.random.normal to generate normally distributed data. **Replace this with your actual dataset.**
3. **Calculate Quantiles:**
   * ppoints = np.linspace(0.01, 0.99, len(data)) generates a sequence of probabilities between 0.01 and 0.99.
   * quantiles\_sample = np.quantile(data, ppoints) calculates the quantiles of your sample data for these probabilities.
   * quantiles\_theoretical = stats.norm.ppf(ppoints) calculates the theoretical quantiles of a standard normal distribution for the same probabilities using the percent point function (ppf, which is the inverse of the CDF). If you want to compare against a different distribution, replace stats.norm with the appropriate distribution from scipy.stats (e.g., stats.expon, stats.gamma).
4. **Create Plot:**
   * fig, ax = plt.subplots() creates a Matplotlib figure and axes.
   * ax.scatter(quantiles\_theoretical, quantiles\_sample) plots the theoretical quantiles against the sample quantiles. If the data is normally distributed, the points should roughly fall along a straight line.
   * ax.plot([-4, 4], [-4, 4], color='r', linestyle='--') adds a reference line (y=x) to visually assess linearity. Adjust the range [-4, 4] based on the expected range of your quantiles.
   * Labels and title are set for clarity.
   * ax.grid(True) adds a grid to the plot.
5. **Display in Streamlit:** st.pyplot(fig) displays the Matplotlib figure in your Streamlit app.
6. **Interactive Options (Optional):** The added section demonstrates how to make the data source and parameters interactive using Streamlit widgets like st.radio, st.file\_uploader, and st.slider.