**Task Description:**

**Data Quality Assessment Tool:**

Create a data quality assessment tool using ML algorithms to detect and report data quality issues such as missing values, outliers, and inconsistencies. This supports data stewardship efforts by providing real-time data quality insights and facilitating data quality improvement initiatives. Providing you an overview of whatever we discussed:

1. Data Profiling: Profile the dataset to understand its structure and quality issues. Identify common data quality problems such as missing values, duplicates, and outliers.
2. Feature Engineering: Create relevant features for data quality assessment, such as data completeness, uniqueness, and distribution statistics – you can come up with your own set of DQ rules relevant to the dataset you're finding online
3. Machine Learning Model: Build an ML model to automatically assess data quality based on the selected features.
4. Documentation: Create a short report on how you went about the design of this tool step-by-step

## FINDINGS

A moving average (MA) is a widely used technical indicator that smooths out price trends by filtering out the noise from random short-term price fluctuations.

The 200-day moving average is considered especially significant in stock trading. As long as **the 50-day moving average of a stock price remains above the 200-day moving average, the stock is generally thought to be in a bullish trend**.

'Bullish Trend' is an upward trend in the prices of an industry's stocks or the overall rise in broad market indices, characterized by high investor confidence.

Does the moving average apply to close or open?
Simple moving average is an average of prices over a specified time period and can be calculated based on closing, opening, minimum or maximum prices. However, it's preferred to use closing prices for the calculations due to their significance.

If the actual price is greater than the predicted price, it typically indicates that the market is performing better than what the model predicted. In the context of stock price prediction, this scenario would suggest a positive difference between the actual market performance and the model's prediction.

Bullish Sentiment: In the context of the stock market, a higher actual price than predicted could suggest a bullish sentiment. This means that the market is optimistic, and investors are willing to buy, which can drive up stock prices.

Positive Surprise: A situation where the actual price is consistently higher than predicted can also indicate that the market is reacting positively to unexpected news, events, or economic conditions. This positive surprise could lead to higher stock prices.

## MODEL EXPLAINATION

The model combines data retrieval, visualization, model prediction, data quality assessment, backtesting, and risk assessment in a Streamlit web application for stock price prediction and analysis. Users can input a stock symbol, and the application provides various insights and metrics related to the stock's historical data and the performance of a predictive model.

1. **Importing Libraries**:
    - Various Python libraries are imported, including NumPy, Pandas, Matplotlib, Pandas DataReader, Yahoo Finance (yfinance), Keras, Streamlit, Scikit-Learn, and Plotly.
2. **Date Range and Application Title**:
    - The script defines a date range for collecting stock data, sets up the title and subheader for the Streamlit application, and allows the user to input a stock ticker symbol (default is AAPL).
3. **Data Retrieval**:
    - The script fetches historical stock data for the specified stock ticker from Yahoo Finance using yfinance or pandas_datareader, based on user input.
4. **Data Description and Visualization**:
    - Descriptive statistics of the stock data are displayed.
    - Several Matplotlib and Plotly charts are generated to visualize the stock data, including closing price vs. time, closing price with 50-day and 200-day moving averages, candlestick chart, and technical indicators (e.g., 50-day and 200-day simple moving averages, Relative Strength Index or RSI).
5. **Candlestick Pattern Analysis**:
    - The script analyzes candlestick patterns in the stock data and displays any detected patterns, such as bullish engulfing, bearish engulfing, hammer, and inverted hammer.
6. **Data Splitting and Scaling**:
    - The stock data is split into training and testing datasets. The data is also scaled using MinMaxScaler.
7. **Model Loading and Prediction**:
    - A pre-trained Keras model (loaded from 'keras_model.h5') is used for making stock price predictions on the testing data.
8. **Visualization of Predictions**:
    - The script visualizes the predicted stock prices alongside actual prices using Matplotlib. It also provides insights into the current market state based on the comparison of predicted and actual daily returns.
9. **Data Quality Assessment**:
    - Various data quality checks are performed, including checking for missing values, duplicate rows, data types, outliers, data distribution, basic statistics, and more. It also assesses the consistency and gaps in the date range.
10. **Backtesting and Risk Assessment**:
    - The script simulates a trading strategy using predicted and actual prices to assess portfolio performance, calculating metrics like maximum drawdown and annualized volatility. It then presents the results.
11. **Model Performance Metrics**:
    - The script calculates and displays model performance metrics, including Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R2). Residuals (differences between actual and predicted prices) are visualized.
12. **Portfolio Balance Over Time and Risk Assessment Metrics**:
    - The script displays the portfolio balance over time and risk assessment metrics such as initial and final portfolio balances, maximum drawdown, and annualized volatility.
13. **Streamlit Command**:
    - The comment **streamlit run app.py** at the end indicates that you can run this script using Streamlit by running it in your terminal with the Streamlit command.