REPORT

In this project, I implemented a C++ program to visualize cryptocurrency data using candlestick charts and volume graphs. The focus was on three core tasks: computing candlestick data, creating a text-based plot of the candlestick data, and plotting a text graph of trading data.

TASK 1: Compute Candlestick Data

Methodology used:

1- **Data Processing:** In the **calculateCandlesticks** function, the initial data processing step is crucial. It begins by interpreting user input to determine the product type - either 'ask' or 'bid'. This distinction is fundamental as it guides the subsequent processing of the order book data. The order book contains a list of buy and sell orders ('bids' and 'asks') at various price points, and identifying the correct product type ensures that the function processes the relevant subset of this data. This step is critical for accurate candlestick computation, as the characteristics of 'ask' and 'bid' data can significantly differ, impacting the resulting candlestick metrics.

2- **Candlestick Calculation:** The candlestick calculation process within the **calculateCandlesticks** function is iterative and detailed. For each candlestick in the order book entries, the function calculates several key metrics:

- **Opening Price:** Determined from the first entry in the time frame, representing the starting price.
- **Closing Price:** Obtained from the last entry, indicating the final trading price in the interval.
- **Highest and Lowest Prices:** The function identifies the maximum and minimum prices within the time frame, critical for understanding market volatility.
- **Volume Accumulation:** It also calculates the total trading volume during the period, aggregating the quantity of traded assets.

This multi-faceted approach to data analysis ensures that each candlestick provides a comprehensive view of market behavior during its respective time frame.

3- **Vector Handling:** In the **calculateCandlesticks** function, vector handling plays a crucial role. Each candlestick, after being calculated, is encapsulated as an object. These objects contain all the relevant metrics like open, high, low, close prices, and volume. These candlestick objects are then sequentially stored in a vector. This approach ensures that the candlesticks are organized in a time-sequential manner, mirroring the chronological order of the market data. Such vector handling is essential for maintaining data integrity and facilitates further processing or analysis of the candlestick series.

TASK 2: Create a Text-based Plot of Candlestick Data

Methodology used:

1- **User Input and Data Preparation:** The initial step involves processing user input and preparing the candlestick data for visualization. This step is crucial as it determines the specific subset of candlestick data to be visualized. The function first interprets user commands to select the relevant candlestick data, which may involve filtering based on time frames or specific market conditions. Then, it organizes this data to ensure that it is in a suitable format for the graphical representation process that follows. This preparation is vital for creating accurate and effective visualizations.

2- **Graphing Mechanics:** The graphing mechanics in the **drawCandlesticks** function involve the use of the **candleDataGraphing** class to translate the calculated candlestick metrics into a textual format. This class plays a critical role in determining how each candlestick is visually represented in a text-based chart. It involves sophisticated calculations to scale the relative sizes of the candle wicks and bodies accurately. These elements are then meticulously mapped onto a grid-like structure made of characters, ensuring that the visual representation is both accurate and comprehensible. This careful mapping allows for a clear and detailed depiction of market trends and patterns in a textual format.

3- **Visualization and Output:** In the **drawCandlesticks** function, visualization and output are designed with a focus on readability. The output graph features clear demarcations for different components of the candlestick, such as the body and wicks. This differentiation aids in quickly understanding the displayed data. Additionally, color coding is utilized to distinguish between bullish and bearish trends, further enhancing the interpretability of the chart. This thoughtful design ensures that users can easily decipher market movements and patterns from the text-based visualization.

Task 3: Plot a text graph of some other trading data (VOLUME GRAPH)

Methodology used:

1- **Volume Data Extraction:** In the **drawVolumeGraph** function, the first step is to extract volume data from the candlestick objects. This step is crucial for plotting the trading volumes in a graphical format. The function processes the candlestick data to retrieve volume metrics, which represent the total amount of trading activity within each candlestick's time frame. This extracted volume data forms the basis for constructing the volume graph, allowing for a visual representation of trading activity over time, complementing the candlestick chart.

2- **Graph Construction:** In the **drawVolumeGraph** function, the construction of the graph involves creating a bar graph to represent trading volumes. This process includes scaling the volume data in relation to the highest volume recorded in the dataset to ensure proportional representation. Character symbols are used to create a graphical depiction of this data. This approach allows for an intuitive understanding of the volume trends in relation to the price movements indicated by the candlestick chart. The use of character-based graphing ensures compatibility with the text-based visualization approach of the overall program.

3- **Rendering and Fine-tuning:** In the rendering and fine-tuning phase of the **drawVolumeGraph** function, the primary focus is on making the volume graph intuitively understandable and ensuring that it aligns seamlessly with the candlestick chart. This involves carefully adjusting the scaling factors of the volume data to ensure that the graphical representation accurately reflects market dynamics. Additionally, the layout is optimized for display in a console environment, which requires specific attention to the formatting and spacing of the character symbols used in the graph. This fine-tuning is crucial for achieving a clear and coherent visual presentation.

The project successfully combines detailed programming techniques with financial data visualization, showcasing a robust application of C++ in creating text-based representations of complex market data. Through meticulous development and careful design, the program effectively translates quantitative trading data into accessible visual formats, enhancing the understanding of market dynamics.