6. The `df.describe()` method provides the 'count', 'mean', 'std', 'min', '25%', '50%', '75%', and 'max' summary statistics for each variable it analyses. Give the definitions (perhaps using help from the ChatBot if needed) of each of these summary statistics

- **Count**- the number of non missing values that are not ‘null’ in each of the columns
- **Mean**- The average value of all numbers in a column (done by sum of all values / no. of values)
- **Std**- Standard Deviation, which shows the amount of variation or dispersion of the data values across the set. It shows how spread out the data points are from the mean.
- **Min-** the minimum or smallest value in a column
- **25%-** the lower quartile or the value below which 25% of the data falls.
- **50%-** The median or the middle value of the data, with half of the values below and half above this point
- **75%-** the upper quartile, or the value below which 75% of the data falls.
- **Max-** the largest or maximum value in a column

7. Missing data can be considered "across rows" or "down columns". Consider how `df.dropna()` or `del df['col']` should be applied to most efficiently use the available non-missing data in your dataset and briefly answer the following questions in your own words

1. Provide an example of a "use case" in which using `df.dropna()` might be preferred over using `del df['col']`
    - When analysing data of a grocery store for example, with columns such as Product, date, price etc. certain rows for products may have missing values such as a null price. In order to analyse the data, such values should be avoided so using df.dropna() will remove that particular row with the null values, allowing a summary to be created using data from all other rows.

2. Provide an example of "the opposite use case" in which using `del df['col']` might be preferred over using `df.dropna()`

    - In the example for the grocery store given earlier, if a particular column such as date is unnecessary, one can use del df[‘dates’] to delete the entire column of dates. This will make it easier to to look at the rest of the columns without any redundant data

3. Discuss why applying `del df['col']` before `df.dropna()` when both are used together could be important

- If you are analysing a dataset which has one column irrelevant to the analysis but may contain many empty/null entries (for example customer feedback), using dropna() first may remove the entire row even though other data like price, or date might be available for that row. Therefore using del first to rid the column not needed for the data analysis allows us to use dropna() whilst still preserving rows that have the relevant/necessary data such as price or date.

Chat GPT Conversation Summary

Here's a summary of the conversation:

1. **Summary Statistics Definitions**: You asked for the definitions of common summary statistics (`count`, `mean`, `std`, `min`, `25%`, `50%`, `75%`, and `max`) used in `df.describe()` in pandas, and I explained what each of these statistics represents in a dataset.
2. **Use Case for `df.dropna()` vs. `del df['col']`**: We discussed a scenario where `df.dropna()` is preferred over `del df['col']`. The use case focused on cleaning a dataset by dropping rows with missing values while retaining all columns, whereas `del df['col']` is used to delete an entire column.
3. **Order of `del df['col']` and `df.dropna()`**: You asked why applying `del df['col']` before `df.dropna()` is important. I explained that deleting irrelevant columns with many missing values first ensures that `df.dropna()` focuses on the remaining relevant columns, preserving useful rows and preventing excessive row deletion.
