Answer the following Questions in 1to 2 lines .

● Question 1: What is Pandas, and why is it commonly used in data cleaning tasks?

Answer : Pandas is a Python library for data manipulation and analysis. It is commonly used in data cleaning tasks due to its powerful data structures and functions that simplify tasks like handling missing data, reshaping datasets, and merging tables.

Top of Form

● Question 2: Given a DataFrame with missing values, how would you check for missing values in each column and count the total number of missing values?

Answer : You can use **df.isnull().sum()** to check for missing values in each column of a DataFrame 'df', and **df.isnull().sum().sum()** to count the total number of missing values across the entire DataFrame.

● Question 3: How can you remove duplicates from a DataFrame while retaining the first occurrence of each unique row?

Answer : You can use **df.drop\_duplicates(keep='first', inplace=True)** to remove duplicates from a DataFrame 'df' while retaining the first occurrence of each unique row.

● Question 4: If you have a DataFrame with a column containing string values, how can you convert all the values in that column to lowercase?

Answer : You can use **df['column\_name'] = df['column\_name'].str.lower()** to convert all values in a specific column ('column\_name') to lowercase in a DataFrame 'df'.

● Question 5: How do you replace missing values in a DataFrame with a specific value, like 0, for a particular column?

Answer : You can use **df['column\_name'].fillna(0, inplace=True)** to replace missing values with 0 in a specific column ('column\_name') of a DataFrame 'df'.

● Question 6: If you have a DataFrame with a datetime column, how can you extract the year, month, and day into separate columns?

Answer : You can use **df['Year'] = df['datetime\_column'].dt.year**, **df['Month'] = df['datetime\_column'].dt.month**, and **df['Day'] = df['datetime\_column'].dt.day** to extract the year, month, and day into separate columns in a DataFrame 'df' with a datetime column.

● Question 7: How can you filter rows in a DataFrame where a specific column's values meet a certain condition (e.g., all rows where 'age' is greater than 30)?

Answer : You can use **df\_filtered = df[df['age'] > 30]** to create a new DataFrame 'df\_filtered' containing only rows where the 'age' column values are greater than 30.

Top of Form

Question 8: What is the purpose of the .apply() function in Pandas, and how would you use it to create a new column based on values from existing columns?

Answer : The **.apply()** function in Pandas is used to apply a function along the axis of a DataFrame or Series. To create a new column based on values from existing columns, you can use **df['new\_column'] = df.apply(lambda row: your\_function(row['column1'], row['column2']), axis=1)**.

● Question 9: Suppose you want to merge two DataFrames, 'df1' and 'df2,' on a common column 'key.' How would you perform this merge operation in Pandas?

Answer : You can merge two DataFrames 'df1' and 'df2' on the common column 'key' using **merged\_df = pd.merge(df1, df2, on='key')**.

● Question 10: You have a DataFrame with a column containing messy text data. How can you clean and standardize the text data (e.g., remove punctuation and convert to lowercase) in that column?

Answer : You can clean and standardize text data in a column 'text\_column' using **df['text\_column'] = df['text\_column'].str.replace('[^\w\s]', '').str.lower()**.