- What is Pandas, and why is it commonly used in data cleaning tasks?

---> Pandas is a Python library used for working with data sets that has functions for analyzing, cleaning, exploring and manipulating data. Pandas is used for data cleaning as it offer a wide range of functions that efficiently handle common data cleaning tasks.It provides functions and methods that make it easier to handle missing data, filtering and selection, data transformation, string operations, data type conversion, duplicated data, handling outliers and Time series data.

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

---> The info() method could be used to check the null values in all the columns at a time. We can use isnull().sum() method to count total number of missing values in each column.

- How can you remove duplicates from a DataFrame while retaining the first occurrence of each unique row?

---> To remove duplicates from DataFrame while retaining the first occurrence of each row using drop_duplicates() method.

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

--->To resolve this issue, applymap() function can be used along with lower() method.

- How do you replace missing values in a DataFrame with a specific value, like 0, for a particular column?

--->To replace a missing value in a DataFrame with a specific value, the fillna() method in pandas is used. If we want to modify the original DataFrame without creating a new DatFrame, inplace=True parameter is used inside the method.

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

---> The given problem could be solved using .dt accessory from pandas. Suppose 'datetime_column' is the column of DataFrame (df) with datetime information. Then to extract year in separate column in same DataFrame we will write code as:
df['Year'] = df['datetime_column].dt.year
To extract month in separate column in same DataFrame we will write code as:
df['Month'] = df['datetime_column].dt.month
To extract day in separate column in same DataFrame we will write code as:
df['Day'] = df['datetime_column].dt.day

- 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)?

---> Boolean indexing can be used to filter rows in a DataFrame with a specific column's values condition. If 'df' is the name of DataFrame and 'Age' is the name of column of interest then the given condition would be applied as:
New_df = df[df['Age'] > 30]
Now 'New_df' will have only the rows where age value is greater than 30 years

- 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?

---> It is a tool that allows us to transform data in custom way. The .apply() function is used to apply a custom function along either a row or column of a DataFrame. In case of series, the it can apply any custom function to each element of series. A new column could be created in exsiting DataFrame using apply() function as:
df.apply(lambda x : custom_function(x), axis=0)
Where, 'df' is the DataFrame and custom_function() is the function that we need to apply on the column.

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

---> The .merge() function could be used to accomplish this task. To merge the given DataFrames code used will be: merged_df = df1.merge(df2, on = 'key')
This code will by default perform inner join. If we want to specify the type of join (left, right, outer) as well then 'how' parameter is also added in the code.

- 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?

---> Python has an in-built library called regular expression (RegEx) to address such problems. We have to import this library as 'import re' and then we have to use findall() function along with specific keyword to get the job done. In our example, if the messy data is stored in column 'text' then to remove punctuation and get only text (stored in variable X) we will write code as:
X = re.findall('\w',text)
Now to convert this data in lowercase the following code need to be written:
df['X'] = df['X'].str.lower()