Missing Data

- **To show rows with missing data (NaN or None values) in a dataset using Pandas, you can use the `isna()` or `isnull()` method followed by boolean indexing. Here's how you can do it:**

```
# Assuming 'df' is your DataFrame

# Use boolean indexing to show rows with missing data
rows_with_missing_data = df[df.isna().any(axis=1)]

# Display rows with missing data
print(rows_with_missing_data)
```

- **To show some specified rows with missing data (NaN or None values)**

```
# Assuming 'df' is your DataFrame and you want to check specific columns 'Column1' and 'Column2'

# Use boolean indexing to show rows with missing data in 'Column1' or 'Column2'
rows_with_missing_data = df[df[['Column1', 'Column2']].isna().any(axis=1)]

# Display rows with missing data
print(rows_with_missing_data)
```

- **To delete rows with missing data (NaN or None values) from a DataFrame in Pandas, you can use the `dropna()` method. Here's how you can do it:**

```
# Assuming 'df' is your DataFrame

# Remove rows with missing data
df_cleaned = df.dropna()

# If you want to modify the original DataFrame in place, you can use the inplace parameter
# For example, to remove rows with missing data in place:
# df.dropna(inplace=True)
```

- **Delete rows with missing data based on some specific columns that contains the missing data**

```
# Remove rows with missing data in a specific subset of columns
df_cleaned = df.dropna(subset=['Column1', 'Column2'])

# To remove rows based on a specific subset of columns in place:
# df.dropna(subset=['Column1', 'Column2'], inplace=True)
```