**[Data Visualization with Python](https://www.coursera.org/learn/python-for-data-visualization" \o "Data Visualization with Python" \t "_blank)**

**Cheat Sheet : Data Preprocessing Tasks in Pandas**

| **Task** | **Syntax** | **Description** | **Example** |
| --- | --- | --- | --- |
| Load CSV data | pd.read\_csv('filename.csv') | Read data from a CSV file into a Pandas DataFrame | df\_can=pd.read\_csv('data.csv') |
| Handling Missing Values | df.dropna() | Drop rows with missing values | df\_can.dropna() |
|  | df.fillna(value) | Fill missing values with a specified value | df\_can.fillna(0) |
| Removing Duplicates | df.drop\_duplicates() | Remove duplicate rows | df\_can.drop\_duplicates() |
| Renaming Columns | df.rename(columns={'old\_name': 'new\_name'}) | Rename one or more columns | df\_can.rename(columns={'Age': 'Years'}) |
| Selecting Columns | df['column\_name'] or df.column\_name | Select a single column | df\_can.Age or df\_can['Age]' |
|  | df[['col1', 'col2']] | Select multiple columns | df\_can[['Name', 'Age']] |
| Filtering Rows | df[df['column'] > value] | Filter rows based on a condition | df\_can[df\_can['Age'] > 30] |
| Applying Functions to Columns | df['column'].apply(function\_name) | Apply a function to transform values in a column | df\_can['Age'].apply(lambda x: x + 1) |
| Creating New Columns | df['new\_column'] = expression | Create a new column with values derived from existing ones | df\_can['Total'] = df\_can['Quantity'] \* df\_can['Price'] |
| Grouping and Aggregating | df.groupby('column').agg({'col1': 'sum', 'col2': 'mean'}) | Group rows by a column and apply aggregate functions | df\_can.groupby('Category').agg({'Total': 'mean'}) |
| Sorting Rows | df.sort\_values('column', ascending=True/False) | Sort rows based on a column | df\_can.sort\_values('Date', ascending=True) |
| Displaying First n Rows | df.head(n) | Show the first n rows of the DataFrame | df\_can.head(3) |
| Displaying Last n Rows | df.tail(n) | Show the last n rows of the DataFrame | df\_can.tail(3) |
| Checking for Null Values | df.isnull() | Check for null values in the DataFrame | df\_can.isnull() |
| Selecting Rows by Index | df.iloc[index] | Select rows based on integer index | df\_can.iloc[3] |
|  | df.iloc[start:end] | Select rows in a specified range | df\_can.iloc[2:5] |
| Selecting Rows by Label | df.loc[label] | Select rows based on label/index name | df\_can.loc['Label'] |
|  | df.loc[start:end] | Select rows in a specified label/index range | df\_can.loc['Age':'Quantity'] |
| Summary Statistics | df.describe() | Generates descriptive statistics for numerical columns | df\_can.describe() |

**Cheat Sheet : Plot Libraries**

| **Library** | **Main Purpose** | **Key Features** | **Programming Language** | **Level of Customization** | **Dashboard Capabilities** | **Types of Plots Possible** |
| --- | --- | --- | --- | --- | --- | --- |
| **Matplotlib** | General-purpose plotting | Comprehensive plot types and variety of customization options | Python | High | Requires additional components and customization | Line plots, scatter plots, bar charts, histograms, pie charts, box plots, heatmaps, etc. |
| **Pandas** | Fundamentally used for data manipulation but also has plotting functionality | Easy to plot directly on Panda data structures | Python | Medium | Can be combined with web frameworks for creating dashboards | Line plots, scatter plots, bar charts, histograms, pie charts, box plots, etc. |
| **Seaborn** | Statistical data visualization | Stylish, specialized statistical plot types | Python | Medium | Can be combined with other libraries to display plots on dashboards | Heatmaps, violin plots, scatter plots, bar plots, count plots, etc. |
| **Plotly** | Interactive data visualization | interactive web-based visualizations | Python, R, JavaScript | High | Dash framework is dedicated for building interactive dashboards | Line plots, scatter plots, bar charts, pie charts, 3D plots, choropleth maps, etc. |
| **Folium** | Geospatial data visualization | Interactive, customizable maps | Python | Medium | For incorporating maps into dashboards, it can be integrated with other frameworks/libraries | Choropleth maps, point maps, heatmaps, etc. |
| **PyWaffle** | Plotting Waffle charts | Waffle charts | Python | Low | Can be combined with other libraries to display waffle chart on dashboards |  |