Pandas Cheat Sheet

Key and Imports

- **`df`**: Refers to any Pandas Dataframe object.
- **`s`**: Refers to any Pandas Series object. You can use the following imports to get started:

Importing Data

- **`pd.read_csv(filename)`**: It read the data from CSV file.
- **`pd.read_table(filename)`**: It is used to read the data from delimited text file.
- **`pd.read_excel(filename)`**: It read the data from an Excel file.
- **`pd.read_sql(query,connection _object)`**: It read the data from a SQL table/database.
- **`pd.read_json(json _string)`**: It read the data from a JSON formatted string, URL or file.
- **`pd.read_html(url)`**: It parses an html URL, string or the file and extract the tables to a list of dataframes.
- **`pd.read_clipboard()`**: It takes the contents of clipboard and passes it to the read_table() function.
- **`pd.DataFrame(dict)`**: From the dict, keys for the columns names, values for the data as lists.

Exporting data

- **`df.to_csv(filename)`**: It writes to a CSV file.
- **`df.to_excel(filename)`**: It writes to an Excel file.
- **`df.to_sql(table_name, connection_object)`**: It writes to a SQL table.
- **`df.to_json(filename)`**: It write to a file in JSON format.

Create Test objects

It is useful for testing the code segments.

- **`pd.DataFrame(np.random.rand(7,18))`**: Refers to 18 columns and 7 rows of random floats.
- **`pd.Series(my_list)`**: It creates a Series from an iterable my_list.
- **`df.index= pd.date_range(‘1940/1/20’, periods=df.shape[0])`**: It adds the date index.

Viewing/Inspecting Data

- `df.head(n)`: It returns first n rows of the DataFrame.
- `df.tail(n)`: It returns last n rows of the DataFrame.
- `df.shape`: It returns number of rows and columns.
- `df.info()`: It returns index, Datatype, and memory information.
- `s.value_counts(dropna=False)`: It views unique values and counts.
- `df.apply(pd.Series.value_counts)`: It refers to the unique values and counts for all the columns.

Selection

- `df[col1]`: It returns column with the label col as Series.
- `df[[col1, col2]]`: It returns columns as a new DataFrame.
- `s.iloc[0]`: It select by the position.
- `s.loc['index_one']`: It select by the index.
- `df.iloc[0,:]`: It returns first row.
- `df.iloc[0,0]`: It returns the first element of first column.

Data cleaning

- `df.columns = ['a','b','c']`: It rename the columns.
- `pd.isnull()`: It checks for the null values and returns the Boolean array.
- `pd.notnull()`: It is opposite of pd.isnull().
- `df.dropna()`: It drops all the rows that contain the null values.
- `df.dropna(axis = 1)`: It drops all the columns that contain null values.
- `df.dropna(axis=1,thresh=n)`: It drops all the rows that have less than n non null values.
- `df.fillna(x)`: It replaces all null values with x.
- `s.fillna(s.mean())`: It replaces all the null values with the mean(the mean can be replaced with almost any function from the statistics module).
- `s.astype(float)`: It converts the datatype of series to float.
- `s.replace(1, 'one')`: It replaces all the values equal to 1 with 'one'.
- `s.replace([1,3],[ 'one', 'three'])`:It replaces all 1 with 'one' and 3 with 'three'.

- `df.rename(columns=lambda x: x+1)`:It rename mass of the columns.
- `df.rename(columns={‘old_name’: ‘new_ name’})`: It consist selective renaming.
- `df.set_index(‘column_one’)`: Used for changing the index.
- `df.rename(index=lambda x: x+1)`: It rename mass of the index.

Filter, Sort, and GroupBy

- `df[df[col] > 0.5]`: Returns the rows where column col is greater than 0.5
- `df[(df[col] > 0.5) & (df[col] < 0.7)]`: Returns the rows where 0.7 > col > 0.5
- `df.sort_values(col1)`: It sorts the values by col1 in ascending order.
- `df.sort_values(col2,ascending=False)`: It sorts the values by col2 in descending order.
- `df.sort_values([col1,col2],ascending=[True,False])`: It sort the values by col1 in ascending order and col2 in descending order.
- `df.groupby(col1)`: Returns a groupby object for the values from one column.
- `df.groupby([col1,col2])`: Returns a groupby object for values from multiple columns.
- `df.groupby(col1)[col2])`: Returns mean of the values in col2, grouped by the values in col1.
- `df.pivot_table(index=col1,values=[col2,col3],aggfunc=mean)`: It creates the pivot table that groups by col1 and calculate mean of col2 and col3.
- `df.groupby(col1).agg(np.mean)`: It calculates the average across all the columns for every unique col1 group.
- `df.apply(np.mean)`: Its task is to apply the function np.mean() across each column.
- `df.apply(np.max,axis=1)`: Its task is to apply the function np.max() across each row.

Join/Combine

- `df1.append(df2)`: Its task is to add the rows in df1 to the end of df2(columns should be identical).
- `pd.concat([df1, df2], axis=1)`: Its task is to add the columns in df1 to the end of df2(rows should be identical).
- `df1.join(df2, on=col1, how=‘inner’)`: SQL-style join the columns in df1 with the columns on df2 where the rows for col have identical values, ‘how’ can be of ‘left’, ‘right’, ‘outer’, ‘inner’.

Statistics

The statistics functions can be applied to a Series, which are as follows:

- **`df.describe()`**: It returns the summary statistics for the numerical columns.
- **`df.mean()`**: It returns the mean of all the columns.
- **`df.corr()`**: It returns the correlation between the columns in the dataframe.
- **`df.count()`**: It returns the count of all the non-null values in each dataframe column.
- **`df.max()`**: It returns the highest value from each of the columns.
- **`df.min()`**: It returns the lowest value from each of the columns.
- **`df.median()`**: It returns the median from each of the columns.
- **`df.std()`**: It returns the standard deviation from each of the columns.