Pandas Cheat Sheet

What is the Pandas library?

In this class, we've heavily relied on the **datascience** module to read in, wrangle, and work with our data. It's a great package for educational purposes, but not super popular in industry or academic settings. **Pandas** is a fast, powerful, flexible, and easy-to-use open-source data analysis and manipulation tool, built on top of the Python programming language.

Common libraries used with Pandas:

- **NumPy**—Pandas is built on top of it. This library facilitates efficient numerical operations on large quantities of data.
- **Matplotlib** is used for data plotting and can produce a large variety of plots, charts, maps, and other visuals.
- **Seaborn** - provides a high-level, dataset-oriented interface for creating attractive statistical graphs.

Tables vs. Dataframes

Tables (datascience library)	**Dataframes** (pandas library)
Tables serve as a method for organizing and storing data. Modifications to tables may occur, but each change will result in the creation of a new table, rather than altering the original. Tables utilize NumPy for numerical manipulation and Matplotlib for simple visualizations.	Dataframes are a way to organize and store data. They can be modified directly, allowing for alterations to the original dataframe. Pandas integrates well with various libraries, such as Numpy for numerical manipulation, Seaborn for statistical graphics, SciPy for statistical analysis, and many others.

Structural Differences

	datascience library	pandas library
Indexing	Rows are accessed by position.	label-based and integer indexing.
Mutability	Immutable - modifications to columns return a new table.	Mutable - can directly modify columns and rows.
Columns	Stores columns as a list of values.	Stores columns as NumPy arrays.

Functionality Differences

	datascience library	pandas library
Data storage	Table().with_columns (“column_name”, [array_name)], “column_name”, [array_name]) OR Table.read_table(“file name”)	pd.dataframe OR pd.read_csv(“file_name”)
Column Access	table.select(“column name”	df.iloc[position #]
Row Access	table.row(n)	df.loc[row[#]]
Adding Rows	table.with_row([”name”])	pd.concat([df, new_row], ignore_index = True)
Sorting	table.sort(“column_name”)	df.sort_values(“column_name”)
Filtering	table.where(“column_name”, are.above(2))	df[df[”column_name”] > 30]
Grouping	table.group(“column name”, collect = np.mean)	df.groupby(“column name”).mean
Ploting	table.scatter( ) table.hist() table.boxplot() table.plot() table.barh()	df.plot(x = “column”, y = “column” , kind = “bar”)
Describing	table.stats()	df.describe()

Pandas Syntax

Describing Dataset

df.head(5) vs. tbl.show(5)	Displays first 5 rows.
df.tail(5) vs. tbl.tail(5)	Displays last 5 rows.
df.shape vs. tbl.shape	Gets the number of rows and columns in the dataframe.
df.describe vs. tbl.stats()	Summary statistics for numerical columns.
df.dtypes vs. type()	Check data types of each column.
df.column.value_counts() vs. tbl.group('column')	Frequency of the unique values in a specified column.
df['column_name'].sum() vs. tbl.column('name').sum()	Returns the sum of values within a column.
df['column_name'].min() vs. tbl.column('name').min()	Returns the minimum value of a column.
df['column_name'].max() vs. tbl.column('name').max()	Returns the maximum value of a column.
df['column_name'].mean() vs. tbl.column('name').mean()	Returns the mean value of a column.
df['column_name'].median() vs. tbl.column('name').median()	Returns the median value of a column.
df['column_name'].std() vs. tbl.column('name').std()	Return the standard deviation of a column.
df.rename(columns = {"old name": "new_name"}, inplace = True) vs. tbl.relabel("column", "new name")	Renames a column.

Additional Resources

https://pandas.pydata.org/community/ecosystem.html
https://pandas.pydata.org/docs/reference/general_functions.html