**71 - Today's Learning Points**

* Use .head(), .tail(), .shape and .columns to explore your DataFrame and find out the number of rows and columns as well as the column names.
* Look for NaN (not a number) values with .findna() and consider using .dropna() to clean up your DataFrame.
* You can access entire columns of a DataFrame using the square bracket notation: df['column name'] or df[['column name 1', 'column name 2', 'column name 3']]
* You can access individual cells in a DataFrame by chaining square brackets df['column name'][index] or using df['column name'].loc[index]
* The largest and smallest values, as well as their positions, can be found with methods like .max(), .min(), .idxmax() and .idxmin()
* You can sort the DataFrame with .sort\_values() and add new columns with .insert()
* To create an Excel Style Pivot Table by grouping entries that belong to a particular category use the .groupby() method

I've attached the completed notebook to this lesson as a .zip file. If you have any issues, unzip the file, upload it to google drive and open it as a Google Colab Notebook.

72 - Congratulations on completing another challenging data science project! Today we've seen how to grab some raw data and create some interesting charts using Pandas and Matplotlib. We've

* used .groupby() to explore the number of posts and entries per programming language
* converted strings to Datetime objects with to\_datetime() for easier plotting
* reshaped our DataFrame by converting categories to columns using .pivot()
* used .count() and isna().values.any() to look for NaN values in our DataFrame, which we then replaced using .fillna()
* created (multiple) line charts using .plot() with a for-loop
* styled our charts by changing the size, the labels, and the upper and lower bounds of our axis.
* added a legend to tell apart which line is which by colour
* smoothed out our time-series observations with .rolling().mean() and plotted them to better identify trends over time.

73 - In this lesson we looked at how to:

* use HTML Markdown in Notebooks, such as section headings # and how to embed images with the <img> tag.
* combine the groupby() and count() functions to aggregate data
* use the .value\_counts() function
* slice DataFrames using the square bracket notation e.g., df[:-2] or df[:10]
* use the .agg() function to run an operation on a particular column
* rename() columns of DataFrames
* create a line chart with two separate axes to visualise data that have different scales.
* create a scatter plot in Matplotlib
* work with tables in a relational database by using primary and foreign keys
* .merge() DataFrames along a particular column
* create a bar chart with Matplotlib

74 - In this lesson we looked at how to:

* How to use .describe() to quickly see some descriptive statistics at a glance.
* How to use .resample() to make a time-series data comparable to another by changing the periodicity.
* How to work with matplotlib.dates Locators to better style a timeline (e.g., an axis on a chart).
* How to find the number of NaN values with .isna().values.sum()
* How to change the resolution of a chart using the figure's dpi
* How to create dashed '--' and dotted '-.' lines using linestyles
* How to use different kinds of markers (e.g., 'o' or '^') on charts.
* Fine-tuning the styling of Matplotlib charts by using limits, labels, linewidth and colours (both in the form of named colours and HEX codes).
* Using .grid() to help visually identify seasonality in a time series.