AXIS IN PANDAS AND THE USAGE OF COUNT(),VALUE_COUNTS(),SUM() FUNCTIONS

I specifically wanted to add this concept in my document as this was tricky as well as interesting. Denoting the Axis in Pandas refers to X Axis(Rows) or Y Axis(Columns), Conceptually Axis = 1 denotes Columns and Axis =0 denotes Rows.

This is emphasized only on all operations like deleting ,concatenating ,counting etc.For E.g:
df = pd.DataFrame(randn(5,6),index =('A','B','C','D','E'),columns = 'P Q R S T U'.split())

//Creation of a dataframe

OUTPUT:

index	P	Q	R	S	T	U
A	-0.7826158	0.27695868	-1.4548526	0.92452716	-0.8868142	-1.2989734
B	0.71729634	-1.0518799	-1.6159051	49930232	-0.0433428	0876886
C	-1.0286334	-0.9196623	0.51524553	1.11606145	-1.0592868	0.12453460
D	-1.5435754	0.12455745	-0.9373690	-0.8458458	0.03753753	-1.3978555
E	0.19842579	-0.9161830	1.55475070	0.56533389	0.0958269	-0.2210476

df.drop(columns="S",axis=1)

OUTPUT:

index	P	Q	R	T	U
A	-0.782615858	0.2769586812	-1.454852617	-0.886814235	-1.298973488
B	0.7172963477	-1.051879959	-1.615905169	-0.043342854	0.2399194800
C	-1.028633435	-0.919662399	0.5152455375	-1.059286887	0.1245346070
D	-1.543575428	0.1245574589	-0.937369070	0.0375375397	-1.397855508
E	0.1984257979	-0.916183072	1.554750708	0.0958269248	-0.221047310

If we want to perform calculations instead of operations like finding the mean,median,mode,sum of columns then we have to mention Axis =0 .Let's check it out with examples:

df.mean(axis=1)

OUTPUT:

A -0.536962

B -0.270390

C -0.208624

D -0.760425

E 0.212851

dtype: float64

This calculates the mean for each row and updates us.

Thus to finalise axis=1 is for columns for operations and rows for calculations.

COUNT() FUNCTION:

This will give me the overall null values of a column of dataframe or the entire dataframe itself.

Df[“yes_or_no”].count() or df.count()

will give me the entire number of non-null values.This cannot check conditions as it is defaultly set to the count of columns.

VALUE_COUNTS() FUNCTION:

This will give me the number of unique null values .

Df[“yes_or_no”].value_counts()

Will give me the number of ‘yes’ values and the number of ‘no’ values.This cannot check for conditions.

BOTH COUNT() AND VALUE_COUNTS() DOES NOT SUPPORT FILTERING.

Therefore making use of eq() maybe one of the option.

No_count= df[‘yes_or_no’].dropna().eq(‘no’).sum()

NULL VALUES IN COUNT(),VALUE_COUNTS() AND EQ():

By default only the number of null values are counted in count().

For value_counts the syntax the consideration of null values is mentioned in the syntax.

Series.value_counts(normalize=False, sort=True, ascending=False, bins=None, dropna=True)

ONE CLASSIC EXAMPLE OF VALUE_COUNTS :

```
import pandas as pd

s = pd.Series([1, 2, 2, 3, 3, 3, 4, 4, 4, 4])

value_counts_series = s.value_counts()

print(value_counts_series)
```

OUTPUT:

```
4    4
```

3 3

2 2

1 1

dtype: int64

The first parameter is the unique value itself. The second parameter is the number of occurrences. Since the no of unique values are always sorted in descending order the first parameter also appears to be in descending order but that is not the case. The corresponding unique value is printed.

value_counts_series = s.value_counts().sort_index()

print(value_counts_series)

This prints in ascending order of the unique values.