Group by

groupby is a powerful function enables to group records according to condition. For example

df.groupby('month').count() # group according to month

df.groupby('network').count() # group according to network

df.groupby('network').describe() # provide stat summary of each group

Now for example, you want to know number of items for each network group

df.groupby('network')['item'].value\_counts()

**TASK:** find which month max number of calls were made?

Grouping can also be done on multiple attributes

df.groupby([df['month'], df['network']]).count()

These compounded group can be set against another series. Number of items (sms, call etc) for each month and network

df.groupby([df['month'], df['network']])['item'].value\_counts()

**TASK:** find out the maximum duration of each network on each month. \*\* use max()

Analysis can also be focused on particular value of a series, for example # What is the sum of durations, for calls only, to each network

df[df['item'] == 'call'].groupby('month')['network'].value\_counts()

**TASK:** plot this result in bar plot or plot of your choice

**TASK:** Which network has the highest number of sms over the 5 months period.

df[df['item'] == 'sms'].groupby('month')['network'].value\_counts()

**TASK:** Get the different items for each month df.groupby('month')['item'].value\_counts()

**TASK:** What is the sum of durations, for calls only, to each network

data[data['item'] == 'call'].groupby('network')['duration'].sum()

Creating a new column

import numpy as np

df["long\_call"] = np.where(df['duration']>=50, 'yes', 'no')

**TASK:** apart from conditional statements you can do arithmetic calculation or other functions on different columns to create a new column

df["new\_col"] = df["col\_x"] - df["col\_y"]

**TASK:** create a new column called *hour* and calculate how many hours each item was.

Reading CSV files excluding rows and footer

path = "path\\to\\folder"

df1 = pd.read\_csv(path + "\\file.csv", error\_bad\_lines=False, skiprows=# of rows to skip from the top, skip\_footer= # number of rows skip from the bottom, usecols=[#,#,#])

**TASK:** Change the # signs to numbers to digits as required read the csv file from [StatWales](https://statswales.gov.wales/Catalogue/Health-and-Social-Care/Mental-Health/Psychiatric-Census/patientsinmentalhealthhospitalsandunitsinwaleswithamentalillness)

**TASK:** Clean the dataframe so that it looks like following

df.rename(columns={'old\_name':'new\_name'}, inplace=True)

