Here is a toy example.

A company wants to connect premises to a network. They are laying cables in different areas of the network A,B and C. The cables have different diameters. This leads to 9 categories of cables They have a table that gives for each customer the cost of cables that have been used for each of the nine categories. They also want to have a more aggregated view where they look up for each customer the cost per area.

So nothing really fancy but the groupby on columns is quite convenient in this case.

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
items = ['A_10', 'A_20', 'A_30', 'B_10', 'B_20', 'B_30', 'C_10', 'C_20', 'C_30']
```

In [3]:
```
costs = np.random.default_rng().uniform(low=1, high=10_000, size = (1000, len(items)))
```

In [4]:
```
df = pd.DataFrame(costs, columns=items)
```

In [5]:
```
df
```

Out[5]:

	A_10	A_20	A_30	B_10	B_20	B_30	C_10	C_20	C_30
0	1994.076406	3965.968196	9205.470686	8680.433832	6845.318909	1966.612902	1067.135923	8171.042817	123.804489
1	7325.701017	2896.825234	6781.193354	7141.777861	6700.107961	2304.090652	6364.097045	278.088286	1265.579396
2	3621.836087	1989.481557	4601.776382	1294.118722	2033.153451	5905.357636	2789.295728	3496.527575	1357.383667
3	3410.075459	660.087819	5605.412586	6551.387493	2413.529186	7599.290733	9605.561447	7530.072459	8603.697816
4	9046.153455	487.052027	5672.480480	2434.318046	6196.671214	7482.038390	8411.602648	877.962565	9178.902306
...	...	...	...	...	...	...	...	...	...
995	139.210595	46.784774	5890.260213	7561.366969	7980.672718	8144.055595	6679.600383	7961.864501	502.704593
996	9403.330023	1653.924643	8645.698421	9429.263344	9277.563777	9938.750556	1402.551129	322.721634	8955.352582
997	2545.109157	8337.219174	3433.367894	8209.689815	6722.230640	8598.077309	1984.927166	9219.228788	130.281585
998	2906.858273	4462.270739	9977.065102	5324.066054	7569.230397	6334.269614	8784.742948	5251.798699	2315.224753
999	2112.689021	6597.503267	5256.059879	6327.150590	9986.769877	6083.325024	7901.998164	4884.753164	3713.876139

1000 rows × 9 columns

In [6]:
```
def grouper(c):
    if c.startswith('A'):
        return 'ACCESS'
    if c.startswith('B'):
        return 'BACKHAUL'
    if c.startswith('C'):
        return 'CORE'
```

In [7]:
```
aggregated_df = df.groupby(by=grouper, axis=1).sum()
aggregated_df
```

Out[7]:

	ACCESS	BACKHAUL	CORE
0	15165.515288	17492.365644	9361.983229
1	17003.719606	16145.976474	7907.764727
2	10213.094025	9232.629809	7643.206971
3	9675.575864	16564.207412	25739.331722
4	15205.685962	16113.027650	18468.467519
...	...	...	...
995	6076.255582	23686.095282	15144.169477
996	19702.953088	28645.577677	10680.625344
997	14315.696225	23529.997764	11334.437540
998	17346.194113	19227.566065	16351.766400
999	13966.252166	22397.245491	16500.627467

1000 rows × 3 columns

In [ ]:
```

```