Explonatory data analysis

Thesis information

Thesis: Pattern extraction and profiling of historical water network demand patterns

Github: Private repositorty

Information about used dataset

Due to privacy reasons, a seperate opensource dataset will be used alongside a private dataset. This opensource dataset, called LeakDB is an artificialy created dataset which closely mimics real world sensor data. While artificial, due to its accuracy it has been chosen to use this dataset for public replications.

Within the dataset there are 1000 different scenarios of the same graph-like waternetwork structure (see image 1). Because there are no relevant or significant differences between the scenarios for this thesis, this EDA will focus on scenario 1.

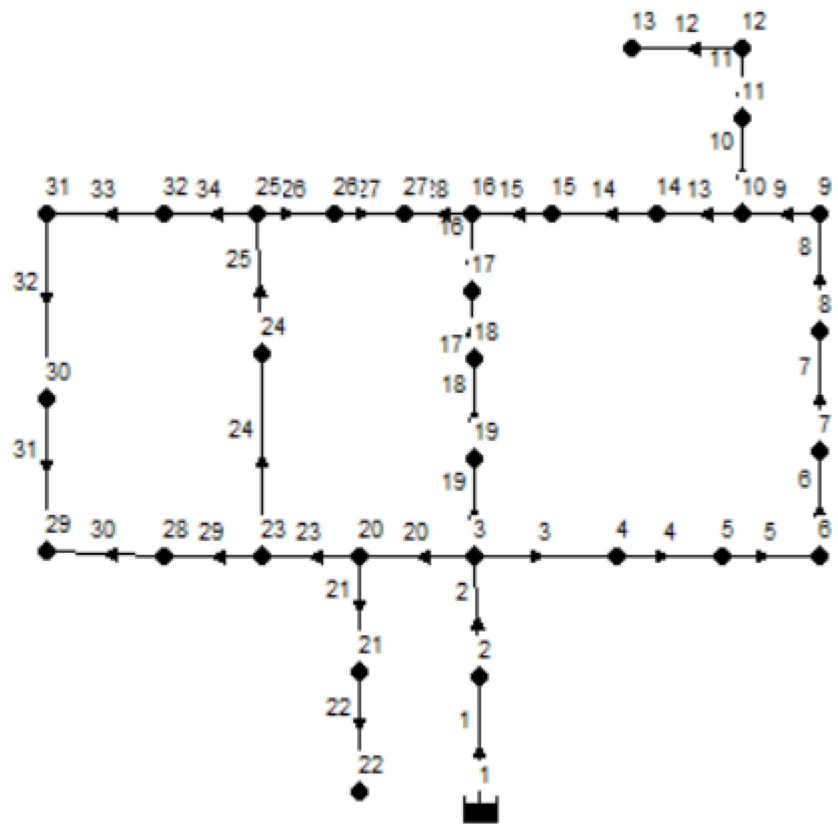


Image 1: Hanoi water network (with flow directions and element ids)

Preperations

Loading libraries and the data

In [ ]:
```
import pandas as pd
from pathlib import Path
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
with open("options.txt", 'r') as f:
    options = f.readlines()
    options = {option.split("=")[0]: option.split("=")[1].strip() for option in
```

In [ ]:
```
scenario_dir = options["hanoi_scenario_dir"]

def read_files_dataframe(scenario_dir):
    dfs = []
    for subfolder in ["Demands", "Flows", "Pressures"]:
        for file in Path(scenario_dir).glob(f"{subfolder}/*.csv"):
            dfs.append(pd.read_csv(file, index_col=0, header=0, names=["Index",
    dfs = pd.concat(dfs, axis=1)
    index = pd.read_csv(f'{scenario_dir}/Timestamps.csv', index_col=0, header=0)
    dfs.index = index.Timestamp
    return dfs

df_1 = read_files_dataframe(scenario_dir)
df_1.head()
```

Out[ ]:

	Demands_Node_1	Demands_Node_10	Demands_Node_11	Demands_Node_12	Dem
Timestamp					
2017-01-01 00:00:00	-3337.2	82.8	82.8	97.2	
2017-01-01 00:30:00	-2973.6	61.2	72.0	86.4	
2017-01-01 01:00:00	-2584.8	57.6	64.8	75.6	
2017-01-01 01:30:00	-2419.2	43.2	57.6	68.4	
2017-01-01 02:00:00	-2196.0	43.2	61.2	61.2	

5 rows × 98 columns

In [ ]:
```
df_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 17520 entries, 2017-01-01 00:00:00 to 2017-12-31 23:30:00
Data columns (total 98 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Demands_Node_1    17520 non-null  float64
 1   Demands_Node_10   17520 non-null  float64
 2   Demands_Node_11   17520 non-null  float64
 3   Demands_Node_12   17520 non-null  float64
 4   Demands_Node_13   17520 non-null  float64
 5   Demands_Node_14   17520 non-null  float64
 6   Demands_Node_15   17520 non-null  float64
 7   Demands_Node_16   17520 non-null  float64
 8   Demands_Node_17   17520 non-null  float64
 9   Demands_Node_18   17520 non-null  float64
 10  Demands_Node_19   17520 non-null  float64
 11  Demands_Node_2    17520 non-null  float64
 12  Demands_Node_20   17520 non-null  float64
 13  Demands_Node_21   17520 non-null  float64
 14  Demands_Node_22   17520 non-null  float64
 15  Demands_Node_23   17520 non-null  float64
 16  Demands_Node_24   17520 non-null  float64
 17  Demands_Node_25   17520 non-null  float64
 18  Demands_Node_26   17520 non-null  float64
 19  Demands_Node_27   17520 non-null  float64
 20  Demands_Node_28   17520 non-null  float64
 21  Demands_Node_29   17520 non-null  float64
 22  Demands_Node_3    17520 non-null  float64
 23  Demands_Node_30   17520 non-null  float64
 24  Demands_Node_31   17520 non-null  float64
 25  Demands_Node_32   17520 non-null  float64
 26  Demands_Node_4    17520 non-null  float64
 27  Demands_Node_5    17520 non-null  float64
 28  Demands_Node_6    17520 non-null  float64
 29  Demands_Node_7    17520 non-null  float64
 30  Demands_Node_8    17520 non-null  float64
 31  Demands_Node_9    17520 non-null  float64
 32  Flows_Link_1      17520 non-null  float64
 33  Flows_Link_10     17520 non-null  float64
 34  Flows_Link_11     17520 non-null  float64
 35  Flows_Link_12     17520 non-null  float64
 36  Flows_Link_13     17520 non-null  float64
 37  Flows_Link_14     17520 non-null  float64
 38  Flows_Link_15     17520 non-null  float64
 39  Flows_Link_16     17520 non-null  float64
 40  Flows_Link_17     17520 non-null  float64
 41  Flows_Link_18     17520 non-null  float64
 42  Flows_Link_19     17520 non-null  float64
 43  Flows_Link_2      17520 non-null  float64
 44  Flows_Link_20     17520 non-null  float64
 45  Flows_Link_21     17520 non-null  float64
 46  Flows_Link_22     17520 non-null  float64
 47  Flows_Link_23     17520 non-null  float64
 48  Flows_Link_24     17520 non-null  float64
 49  Flows_Link_25     17520 non-null  float64
 50  Flows_Link_26     17520 non-null  float64
 51  Flows_Link_27     17520 non-null  float64
 52  Flows_Link_28     17520 non-null  float64
 53  Flows_Link_29     17520 non-null  float64
 54  Flows_Link_3      17520 non-null  float64
```

```
 55  Flows_Link_30      17520 non-null  float64
 56  Flows_Link_31      17520 non-null  float64
 57  Flows_Link_32      17520 non-null  float64
 58  Flows_Link_33      17520 non-null  float64
 59  Flows_Link_34      17520 non-null  float64
 60  Flows_Link_4       17520 non-null  float64
 61  Flows_Link_5       17520 non-null  float64
 62  Flows_Link_6       17520 non-null  float64
 63  Flows_Link_7       17520 non-null  float64
 64  Flows_Link_8       17520 non-null  float64
 65  Flows_Link_9       17520 non-null  float64
 66  Pressures_Node_1   17520 non-null  float64
 67  Pressures_Node_10  17520 non-null  float64
 68  Pressures_Node_11  17520 non-null  float64
 69  Pressures_Node_12  17520 non-null  float64
 70  Pressures_Node_13  17520 non-null  float64
 71  Pressures_Node_14  17520 non-null  float64
 72  Pressures_Node_15  17520 non-null  float64
 73  Pressures_Node_16  17520 non-null  float64
 74  Pressures_Node_17  17520 non-null  float64
 75  Pressures_Node_18  17520 non-null  float64
 76  Pressures_Node_19  17520 non-null  float64
 77  Pressures_Node_2   17520 non-null  float64
 78  Pressures_Node_20  17520 non-null  float64
 79  Pressures_Node_21  17520 non-null  float64
 80  Pressures_Node_22  17520 non-null  float64
 81  Pressures_Node_23  17520 non-null  float64
 82  Pressures_Node_24  17520 non-null  float64
 83  Pressures_Node_25  17520 non-null  float64
 84  Pressures_Node_26  17520 non-null  float64
 85  Pressures_Node_27  17520 non-null  float64
 86  Pressures_Node_28  17520 non-null  float64
 87  Pressures_Node_29  17520 non-null  float64
 88  Pressures_Node_3   17520 non-null  float64
 89  Pressures_Node_30  17520 non-null  float64
 90  Pressures_Node_31  17520 non-null  float64
 91  Pressures_Node_32  17520 non-null  float64
 92  Pressures_Node_4   17520 non-null  float64
 93  Pressures_Node_5   17520 non-null  float64
 94  Pressures_Node_6   17520 non-null  float64
 95  Pressures_Node_7   17520 non-null  float64
 96  Pressures_Node_8   17520 non-null  float64
 97  Pressures_Node_9   17520 non-null  float64
dtypes: float64(98)
memory usage: 13.2+ MB
```

The data is gathered from 32 different nodes and 34 different pipes between the nodes, as can be seen in image 1. The nodes measure the demand and pressure while the pipes measure flow rates. This leads to the dataset having 98 different columns, each with 17520 rows of halfhourly timeseries data. This equates to 1 year of artificial data.

As can be seen, each of the columns contain floats with no missing values.

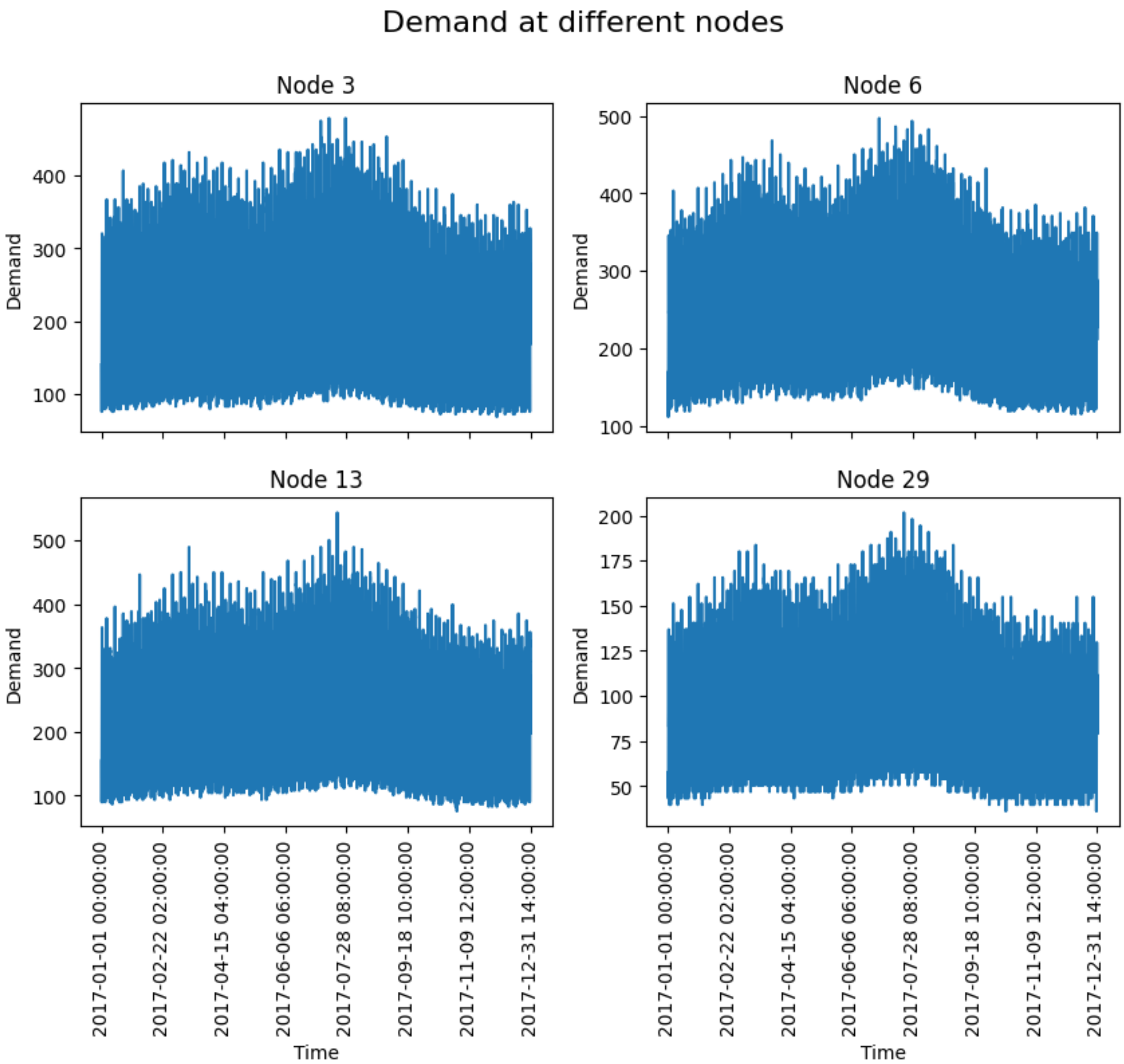
Overview of multiple nodes

To get a better understanding of how the data looks, we can plot the timeseries as a graph. Due to the different behaviours of the nodes due to their location in the graph, multiple graphs will be shown.

Nodes of different scenarios

In [ ]:
```
fig, axs = plt.subplots(2, 2, sharex=True)
df_1.Demands_Node_3.plot(ax=axs[0][0], xlabel="Time", ylabel="Demand", title="No
df_1.Demands_Node_6.plot(ax=axs[0][1], xlabel="Time", ylabel="Demand", title="No
df_1.Demands_Node_13.plot(ax=axs[1][0], xlabel="Time", ylabel="Demand", title="N
df_1.Demands_Node_29.plot(ax=axs[1][1], xlabel="Time", ylabel="Demand", title="N
fig.suptitle('Demand at different nodes', fontsize=16)
```

Out[ ]:
```
Text(0.5, 0.98, 'Demand at different nodes')
```

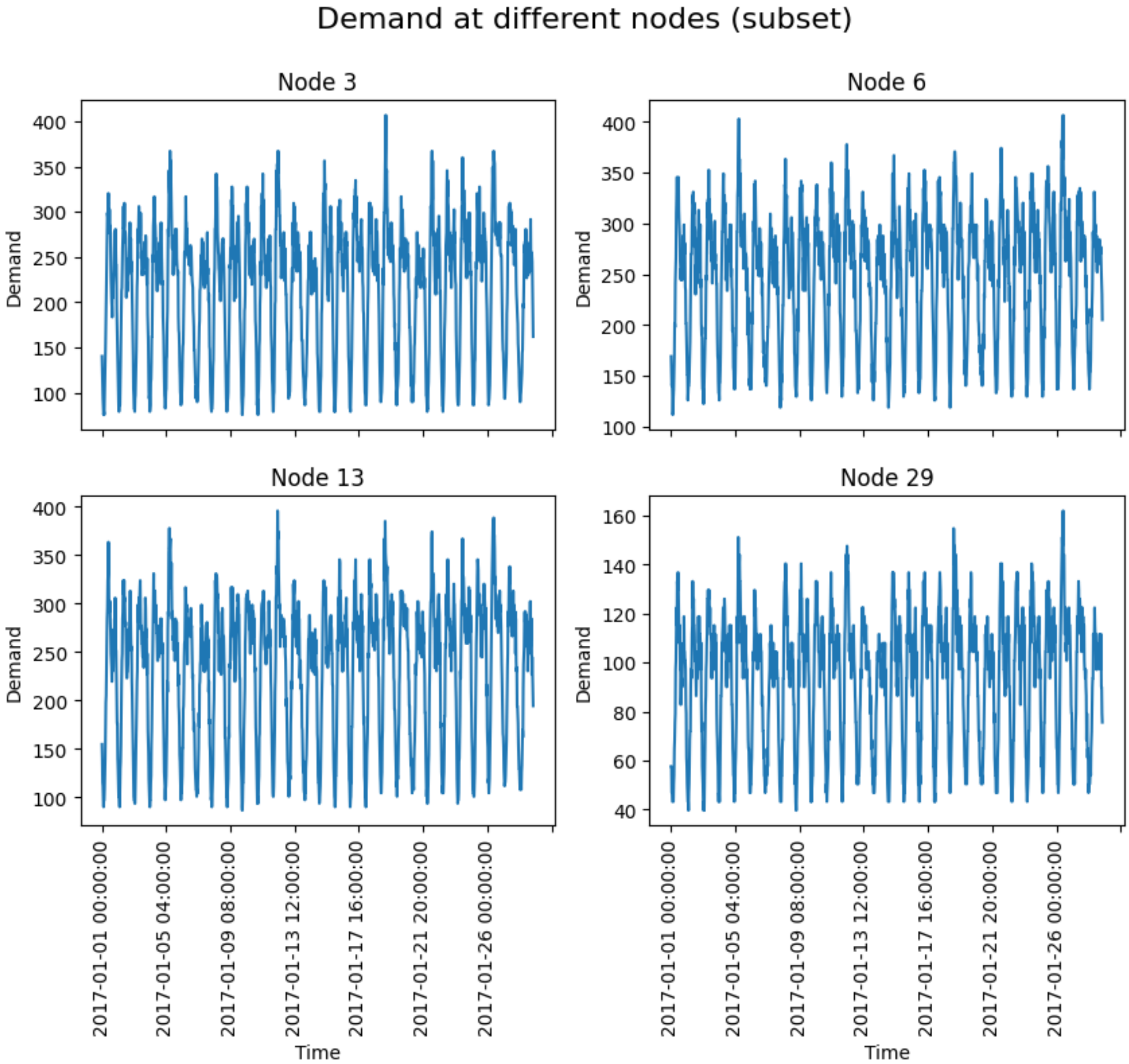


Subset of timeseries data

Due to the amount of data, small details are lost so lets plot a month worth of data.

In [ ]:
```
fig, axs = plt.subplots(2,2, sharex=True)
df_1.Demands_Node_3[:1344].plot(ax=axs[0][0], xlabel="Time", ylabel="Demand", ti
df_1.Demands_Node_6[:1344].plot(ax=axs[0][1], xlabel="Time", ylabel="Demand", ti
df_1.Demands_Node_13[:1344].plot(ax=axs[1][0], xlabel="Time", ylabel="Demand", t
df_1.Demands_Node_29[:1344].plot(ax=axs[1][1], xlabel="Time", ylabel="Demand", t
fig.suptitle('Demand at different nodes (subset)', fontsize=16)
```

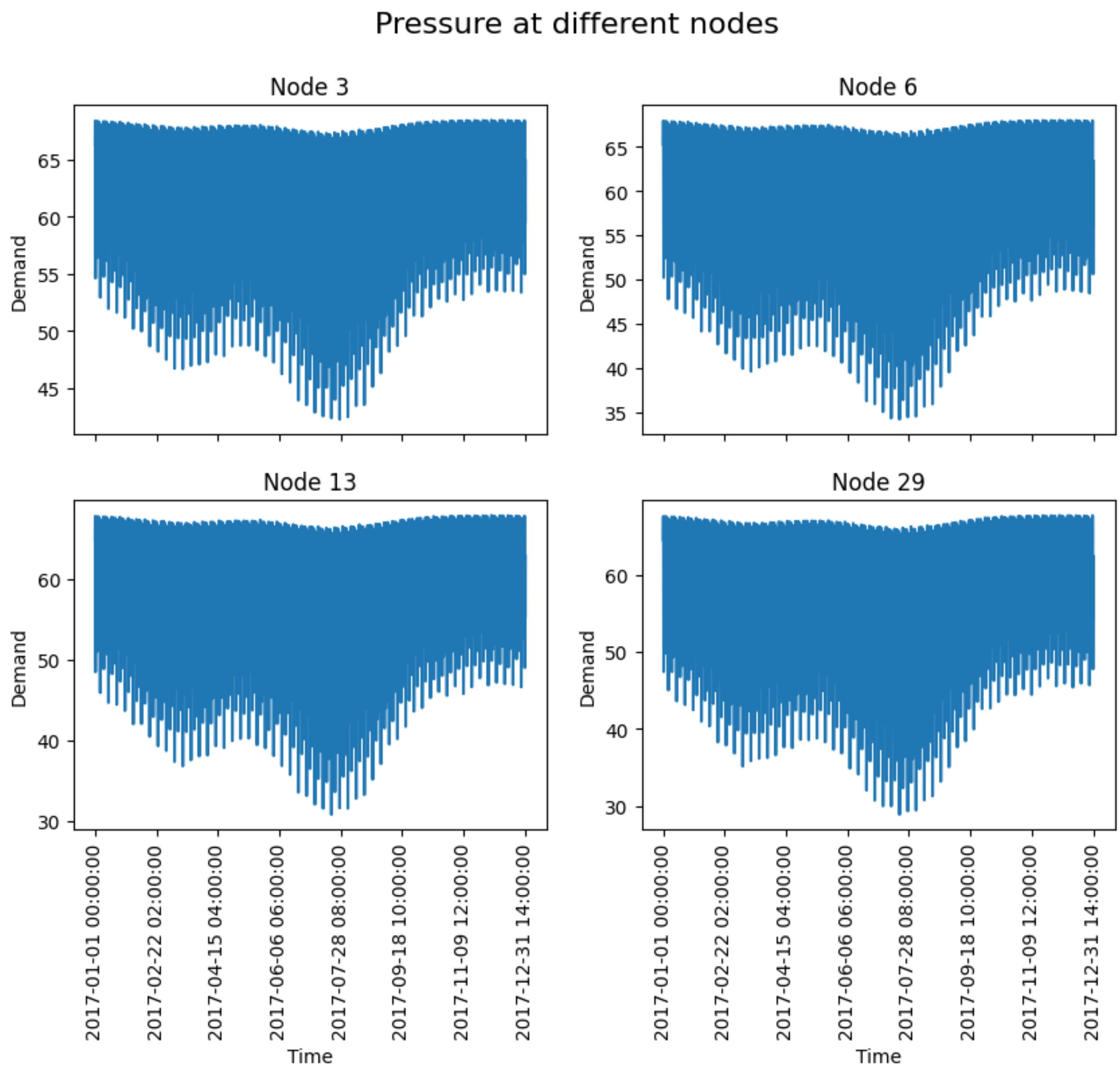
Out[ ]: Text(0.5, 0.98, 'Demand at different nodes (subset)')



The same can be done for pressure at these nodes.

In [ ]:
```
fig, axs = plt.subplots(2, 2, sharex=True)
df_1.Pressures_Node_3.plot(ax=axs[0][0], xlabel="Time", ylabel="Demand", title='
df_1.Pressures_Node_6.plot(ax=axs[0][1], xlabel="Time", ylabel="Demand", title='
df_1.Pressures_Node_13.plot(ax=axs[1][0], xlabel="Time", ylabel="Demand", title=
df_1.Pressures_Node_29.plot(ax=axs[1][1], xlabel="Time", ylabel="Demand", title=
fig.suptitle('Pressure at different nodes', fontsize=16)
```

Out[ ]: Text(0.5, 0.98, 'Pressure at different nodes')

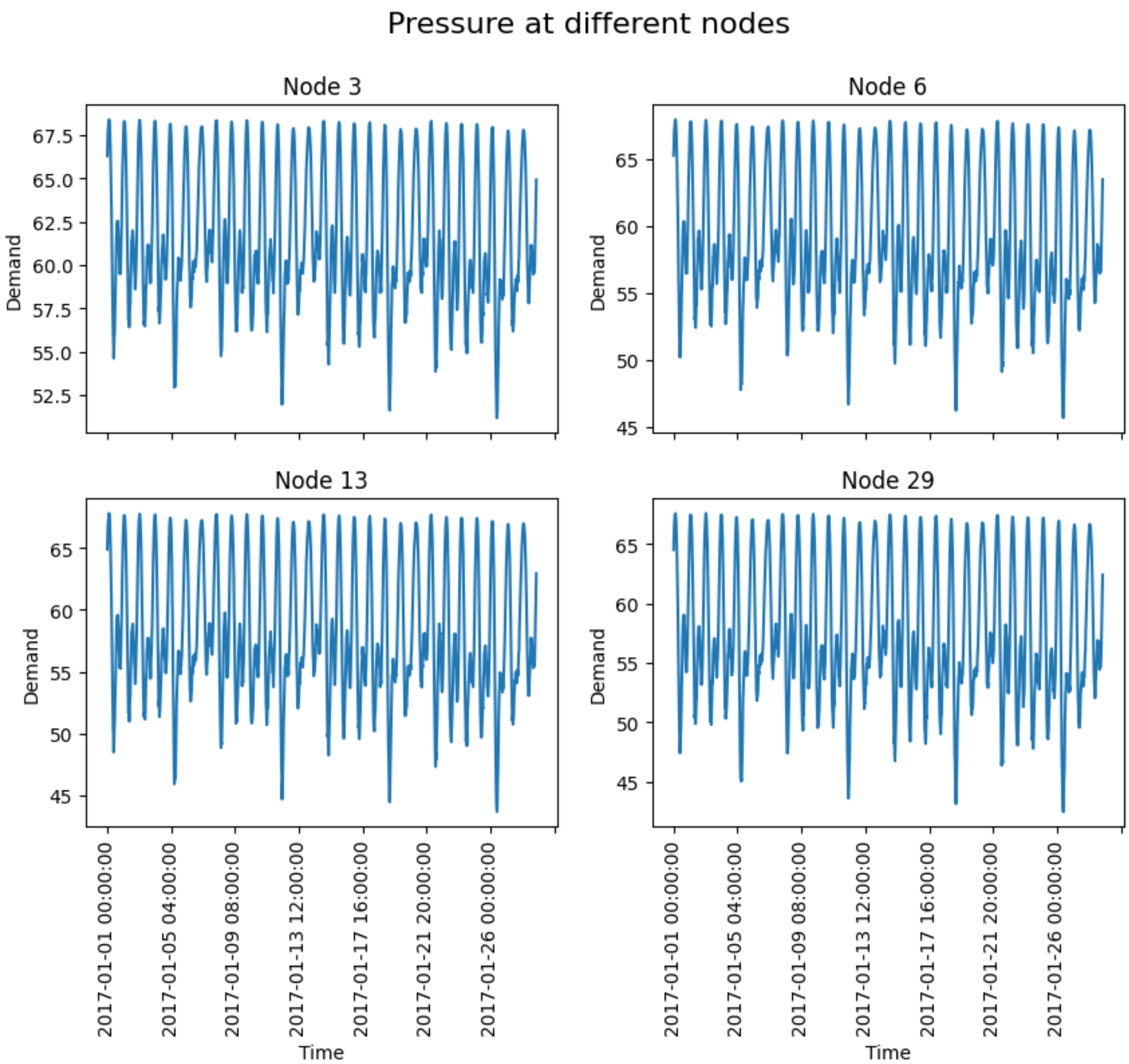


In [ ]:

```
fig, axs = plt.subplots(2, 2, sharex=True)
df_1.Pressures_Node_3[:1344].plot(ax=axs[0][0], xlabel="Time", ylabel="Demand",
df_1.Pressures_Node_6[:1344].plot(ax=axs[0][1], xlabel="Time", ylabel="Demand",
df_1.Pressures_Node_13[:1344].plot(ax=axs[1][0], xlabel="Time", ylabel="Demand",
df_1.Pressures_Node_29[:1344].plot(ax=axs[1][1], xlabel="Time", ylabel="Demand",
fig.suptitle('Pressure at different nodes', fontsize=16)
```

Out[ ]:

```
Text(0.5, 0.98, 'Pressure at different nodes')
```



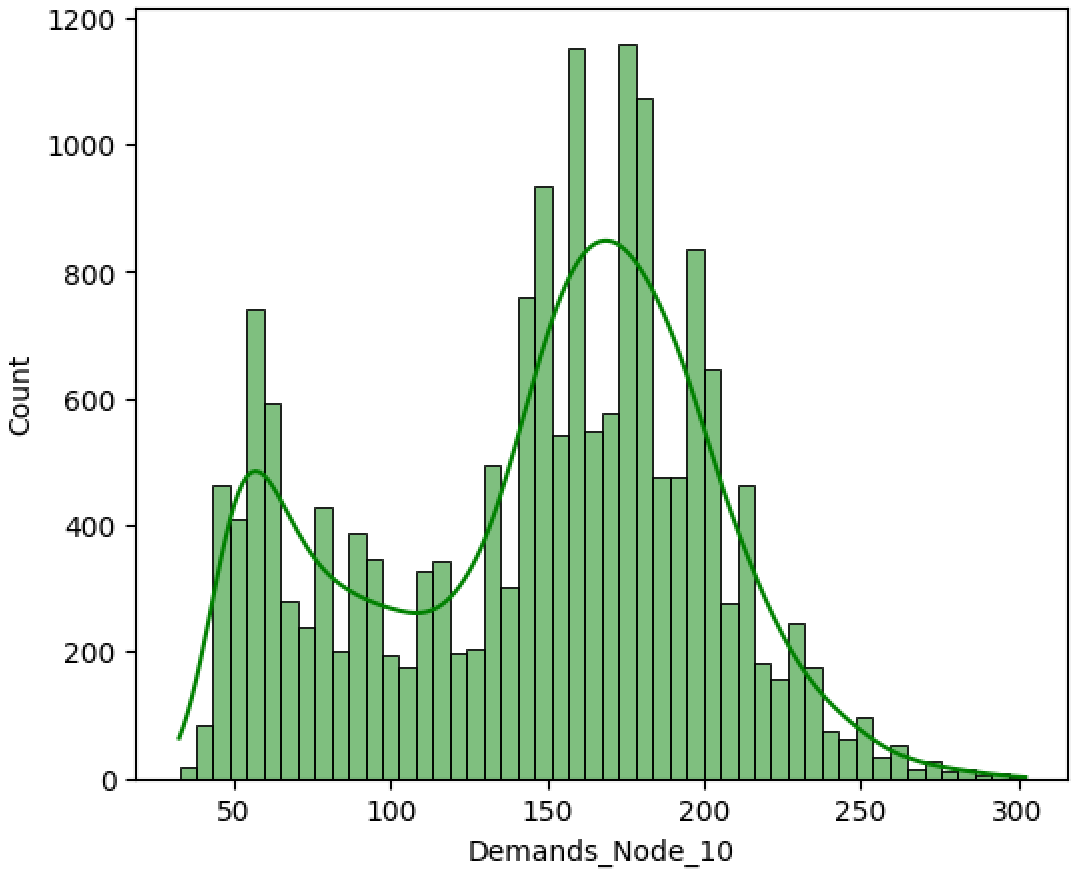
From these, it is clear that while they mostly look the same, there are some subtle differences. From these plots it also is clear that there seems to be some correlation as well as multiple repeating patterns.

Demand distribution for nodes 10 and 29

To get a better understanding of the data, we can look at the distribution of values. For this example, nodes 10 and 29 were chosen. Node 29 was chosen since it is a point where every incmoing pipe flows towards, while node 10 is on the other side while also being not being an end-point.

In [ ]:
```
# Node 10
display(df_1.Demands_Node_10.describe())
plt.figure(figsize=(6, 5))
sns.histplot(df_1.Demands_Node_10, color='g', kde=True, bins=50)
plt.show()
```

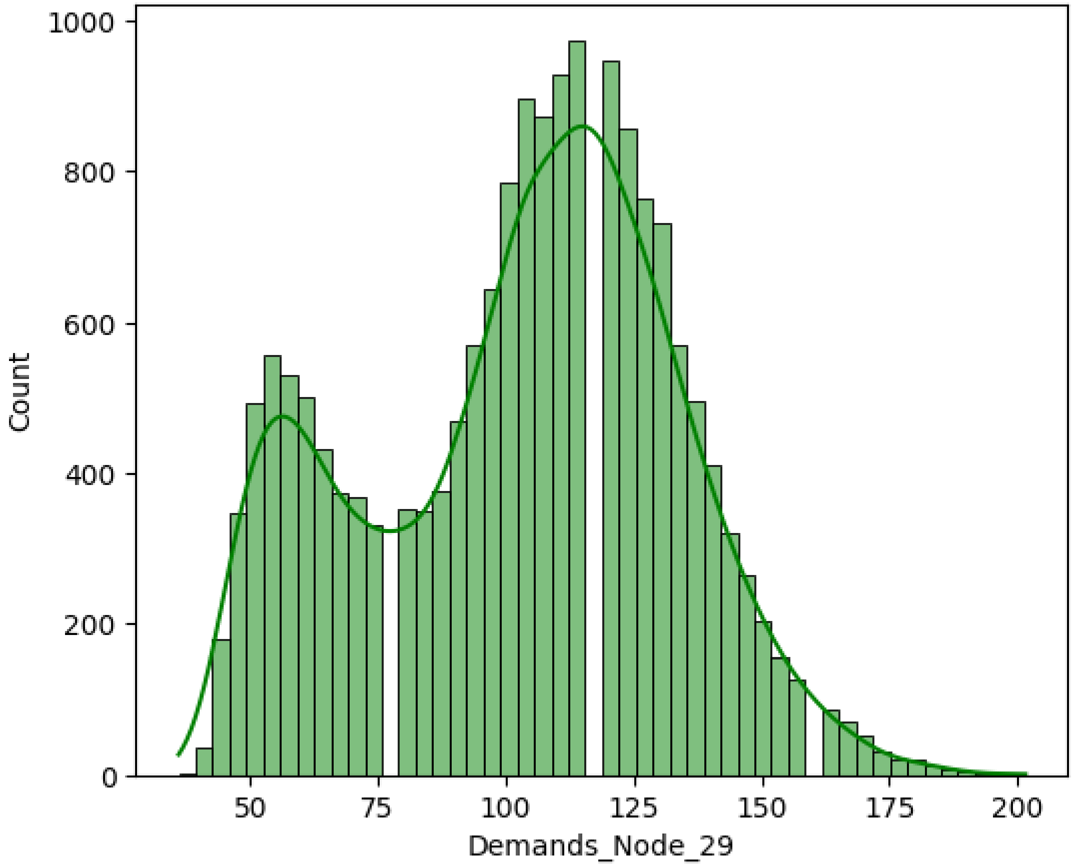
```
count    17520.000000
mean       146.056233
std         54.321542
min         32.400000
25%        100.800000
50%        158.400000
75%        183.600000
max        302.400000
Name: Demands_Node_10, dtype: float64
```



In [ ]:

```
# Node 29
display(df_1.Demands_Node_29.describe())
plt.figure(figsize=(6, 5))
sns.histplot(df_1.Demands_Node_29, color='g', kde=True, bins=50)
plt.show()
```

```
count    17520.000000
mean       102.653836
std         30.151159
min         36.000000
25%         79.200000
50%        108.000000
75%        122.400000
max        201.600000
Name: Demands_Node_29, dtype: float64
```



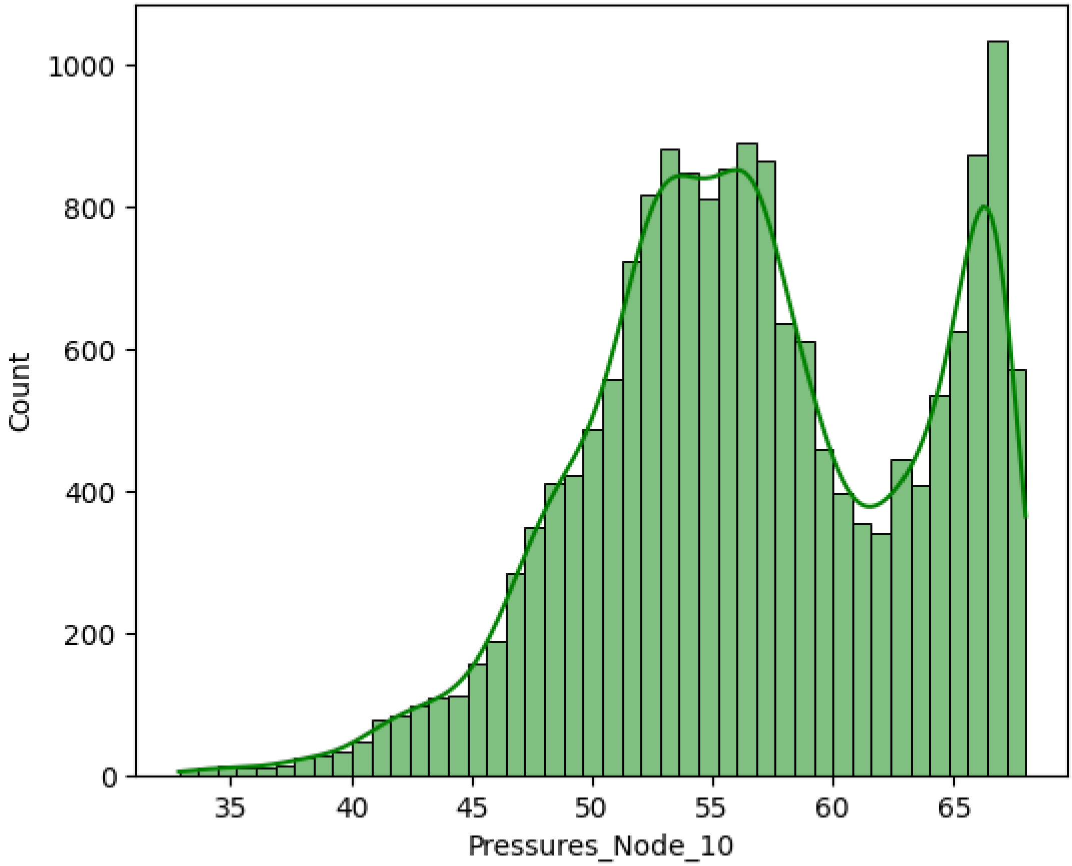
From this it can be seen that there are a few differences with regards to the distribution of values itself as well as the minumum and max values. This most probably is due to the difference in position and function of the nodes, as described earlier.

Pressure distribution for node 10

The same can be done for pressure.

In [ ]:
```
display(df_1.Pressures_Node_10.describe())
plt.figure(figsize=(6, 5))
sns.histplot(df_1.Pressures_Node_10, color='g', kde=True)
plt.show()
```

```
count    17520.000000
mean        56.706471
std          6.738747
min         32.850000
25%         52.157500
50%         56.275000
75%         62.592500
max         67.984000
Name: Pressures_Node_10, dtype: float64
```

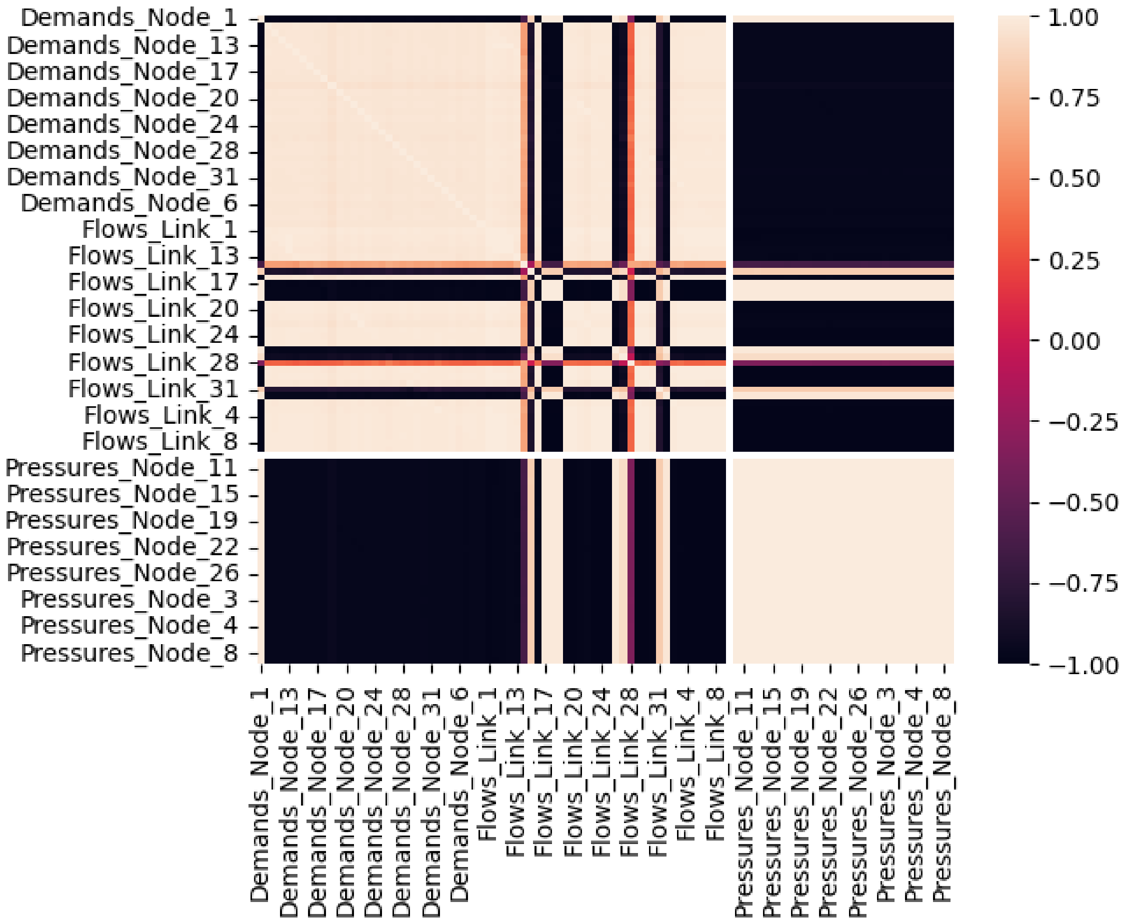


When compared to the demands graphs, the pressure seems to be a bit more stable and centered around a specific value. This seems to be the case for most nodes.

Correlation

As mentioned earlier, there seems to be some correlation between the different columns. To check if this indeed the case, we can run seaborn's `corr()` function.

In [ ]:
```
df_1_corr = df_1.corr()
sns.heatmap(df_1_corr, annot=False)
plt.show()
```



It seems like there are quite a few strongly correlated items, both negativly and positivly correlated. Almost all items seem to have at least some correlation to almost every other time, expect a few, most of which can be found in the middle of the plot. This probably is due to graph-like structure of the data making it so that a change in the network disperses through the graph.