Import Libraries

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

import sys
import os

pd.options.mode.chained_assignment = None

pd.set_option("display.max_rows", None)
pd.set_option("display.max_columns", None)
pd.set_option('display.max_colwidth', None)

import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
```

In [2]:
```
np.random.seed(311)
```

Load Data

In [3]:
```
current_file_path = os.path.dirname(os.path.realpath('__file__'))
current_folder_name = 'code'

project_base_path = current_file_path.removesuffix(current_folder_name)
temperature_filepath = project_base_path + '/data/input/temperature data.xls
```

In [4]:
```
temperature_df = pd.read_excel(
    temperature_filepath,
    header = 1
)

temperature_df.head()
```

Out[4]:

	Date	Time	Date/time	Room A	GSW+FM	AFLN+MU	AFLN+FB	AFLH+FM	AFLH+MU
0	2023-02-01	22:00:00	2023-02-01 22:00:00	20.5	15.2	13.3	13.7	15.3	15.6
1	2023-02-01	23:00:00	2023-02-01 23:00:00	20.8	15.2	13.4	13.8	15.4	15.7
2	2023-02-02	00:00:00	2023-02-02 00:00:00	21.0	15.2	13.5	14.0	15.5	15.8
3	2023-02-02	01:00:00	2023-02-02 01:00:00	21.1	15.3	13.5	14.1	15.6	15.8
4	2023-02-02	02:00:00	2023-02-02 02:00:00	21.3	15.3	13.7	14.2	15.7	15.9

In [5]:
```
temperature_df.shape
```

Out[5]:
```
(2154, 36)
```

In [6]:
```
print('Columns:\n')
print(*temperature_df.columns, sep = ' ')
```

```
Columns:

Date Time Date/time Room A GSW+FM AFLN+MU AFLN+FB AFLH+FM AFLH+MU AFLH GSW+
FB AFLN+FM GSW+MU AFLH+FB Room B GSW+MU.1 AFLH.1 AFLH+FB.1 GSW+FM.1 AFLH+M
U.1 AFLN+FM.1 GSW+FB.1 AFLN+FB.1 AFLH+FM.1 AFLN+MU.1 Room C AFLH+FM.2 GSW+F
B.2 GSW+MU.2 AFLN+FM.2 GSW+FM.2 AFLH.2 AFLN+MU.2 AFLN+FB.2 AFLH+FB.2 AFLH+M
U.2
```

Transform Dataframe for organizing blocks(bottles) better

Split dataset into subsets per each block(bottle)

In [7]:
```
temperature_df_subset_A = temperature_df[
    [
        'Date', 'Time', 'Date/time', 'Room A',
        'GSW+FM', 'AFLN+MU', 'AFLN+FB',
       'AFLH+FM', 'AFLH+MU', 'AFLH',
        'GSW+FB', 'AFLN+FM', 'GSW+MU', 'AFLH+FB'
    ]
]

temperature_df_subset_B = temperature_df[
    [
        'Date', 'Time', 'Date/time', 'Room B',
        'GSW+MU.1', 'AFLH.1', 'AFLH+FB.1',
        'GSW+FM.1', 'AFLH+MU.1', 'AFLN+FM.1',
        'GSW+FB.1', 'AFLN+FB.1', 'AFLH+FM.1', 'AFLN+MU.1'
    ]
]
```

```
temperature_df_subset_C = temperature_df[
    [
        'Date', 'Time', 'Date/time', 'Room C',
        'AFLH+FM.2', 'GSW+FB.2', 'GSW+MU.2',
        'AFLN+FM.2', 'GSW+FM.2', 'AFLH.2',
        'AFLN+MU.2', 'AFLN+FB.2', 'AFLH+FB.2', 'AFLH+MU.2'
    ]
]
```

In [8]:
```
temperature_df_subset_A.head()
```

Out[8]:

	Date	Time	Date/time	Room A	GSW+FM	AFLN+MU	AFLN+FB	AFLH+FM	AFLH+MU
0	2023-02-01	22:00:00	2023-02-01 22:00:00	20.5	15.2	13.3	13.7	15.3	15.6
1	2023-02-01	23:00:00	2023-02-01 23:00:00	20.8	15.2	13.4	13.8	15.4	15.7
2	2023-02-02	00:00:00	2023-02-02 00:00:00	21.0	15.2	13.5	14.0	15.5	15.8
3	2023-02-02	01:00:00	2023-02-02 01:00:00	21.1	15.3	13.5	14.1	15.6	15.8
4	2023-02-02	02:00:00	2023-02-02 02:00:00	21.3	15.3	13.7	14.2	15.7	15.9

Transform columns into a common style:

Add a column indicating a block

In [9]:
```
temperature_df_subset_A['Block'] = 'A'
temperature_df_subset_B['Block'] = 'B'
temperature_df_subset_C['Block'] = 'C'
```

Remove block indicator

In [10]:
```
temperature_df_subset_A = temperature_df_subset_A.rename(columns = {'Room A
temperature_df_subset_B = temperature_df_subset_B.rename(columns = {'Room B
temperature_df_subset_C = temperature_df_subset_C.rename(columns = {'Room C
```

Remove digits from column names for treatments

In [11]:
```
def remove_digits_from_column_names(df):
    df.columns = df.columns.str.replace('[\.\d]', '', regex = True)

remove_digits_from_column_names(temperature_df_subset_B)
remove_digits_from_column_names(temperature_df_subset_C)
```

Combine datasets with each block into one

In [12]:
```
temperature_df_transformed = pd.concat([
    temperature_df_subset_A,
```

```
    temperature_df_subset_B,
    temperature_df_subset_C
],
    ignore_index = True
)
```

In [13]:
```
temperature_df_transformed.head(10)
```

Out[13]:

	Date	Time	Date/time	Room	GSW+FM	AFLN+MU	AFLN+FB	AFLH+FM	AFLH+MU
0	2023-02-01	22:00:00	2023-02-01 22:00:00	20.5	15.2	13.3	13.7	15.3	15.6
1	2023-02-01	23:00:00	2023-02-01 23:00:00	20.8	15.2	13.4	13.8	15.4	15.7
2	2023-02-02	00:00:00	2023-02-02 00:00:00	21.0	15.2	13.5	14.0	15.5	15.8
3	2023-02-02	01:00:00	2023-02-02 01:00:00	21.1	15.3	13.5	14.1	15.6	15.8
4	2023-02-02	02:00:00	2023-02-02 02:00:00	21.3	15.3	13.7	14.2	15.7	15.9
5	2023-02-02	03:00:00	2023-02-02 03:00:00	21.4	15.4	13.7	14.3	15.8	16.0
6	2023-02-02	04:00:00	2023-02-02 04:00:00	21.5	15.4	13.8	14.5	15.9	16.1
7	2023-02-02	05:00:00	2023-02-02 05:00:00	21.6	15.5	14.0	14.6	16.0	16.2
8	2023-02-02	06:00:00	2023-02-02 06:00:00	21.7	15.5	14.1	14.7	16.1	16.2
9	2023-02-02	07:00:00	2023-02-02 07:00:00	21.9	15.6	14.2	14.9	16.2	16.3

Data Cleaning

Convert Date to a pandas datetime object

In [14]:
```
temperature_df_transformed['Date'] = pd.to_datetime(
    temperature_df_transformed['Date']
)

temperature_df_transformed['Date/time'] = pd.to_datetime(
    temperature_df_transformed['Date/time']
)
```

Some records have unrealistic values for Date/time

In [15]:
```
temperature_df_transformed['Date/time'].min()
```

Out[15]:
```
Timestamp('1970-01-01 00:00:00.000000045')
```

In [16]:
```
temperature_df_transformed[
    temperature_df_transformed['Date/time'] == temperature_df_transformed['[
].head()
```

Out[16]:

	Date	Time	Date/time	Room	GSW+FM	AFLN+MU	AFLN+FB	AFLH+FM	A
2150	NaT	NaN	1970-01-01 00:00:00.000000045	0.0	68.0	209.0	410.0	435.0	
4304	NaT	NaN	1970-01-01 00:00:00.000000045	0.0	0.0	317.0	529.0	497.0	
6458	NaT	NaN	1970-01-01 00:00:00.000000045	0.0	0.0	210.0	617.0	465.0	

Remove the records with unrealistic or missing values for Date and time

In [17]:
```
n_records_before_cleaning_dates = temperature_df_transformed.shape[0]
```

In [18]:
```
temperature_df_transformed.drop(
    temperature_df_transformed[
        temperature_df_transformed['Date/time'] <= np.datetime64('2000-01-01
    ].index,
    inplace = True
)
```

In [19]:
```
temperature_df_transformed['Date/time'].min()
```

Out[19]:
```
Timestamp('2023-02-01 22:00:00')
```

In [20]:
```
n_records_after_cleaning_dates = temperature_df_transformed.shape[0]
```

In [21]:
```
difference = n_records_before_cleaning_dates - n_records_after_cleaning_date

print(
    f'{difference} rows have been removed'
)
```

```
12 rows have been removed
```

In [22]:
```
print('Columns:\n')
print(*temperature_df_transformed.columns.tolist(), sep = ' ')
```

```
Columns:

Date Time Date/time Room GSW+FM AFLN+MU AFLN+FB AFLH+FM AFLH+MU AFLH GSW+FB
AFLN+FM GSW+MU AFLH+FB Block
```

Column 'Time' is redundant, we can drop it

In [23]:
```
temperature_df_transformed.drop('Time', axis = 1, inplace = True)
```

We want to have a column counting number of hours as a part of a day from the start of the experiment for visualization

In [24]:
```
starting_time = temperature_df_transformed['Date/time'].min()

temperature_df_transformed['Relative_time'] = temperature_df_transformed[
    'Date/time'
] - starting_time

temperature_df_transformed['Hours'] = (
    (
        temperature_df_transformed[
            'Relative_time'
        ].dt.days
    ) * 24 + (
        temperature_df_transformed[
            'Relative_time'
        ].dt.components['hours']
    ) + 1
) / 24
```

And we want to have a column counting number of days from the start of the experiment for taking an average per day

In [25]:
```
starting_day = temperature_df_transformed['Date'].min()

temperature_df_transformed['Day'] = (
    temperature_df_transformed['Date'] - starting_day
).dt.days.astype(int)

temperature_df_transformed.head()
```

Out[25]:

	Date	Date/time	Room	GSW+FM	AFLN+MU	AFLN+FB	AFLH+FM	AFLH+MU	AFLH	GS
0	2023-02-01	2023-02-01 22:00:00	20.5	15.2	13.3	13.7	15.3	15.6	17.7	
1	2023-02-01	2023-02-01 23:00:00	20.8	15.2	13.4	13.8	15.4	15.7	17.8	
2	2023-02-02	2023-02-02 00:00:00	21.0	15.2	13.5	14.0	15.5	15.8	17.9	
3	2023-02-02	2023-02-02 01:00:00	21.1	15.3	13.5	14.1	15.6	15.8	18.0	
4	2023-02-02	2023-02-02 02:00:00	21.3	15.3	13.7	14.2	15.7	15.9	18.0	

We want to get averages per block for each experiment

In [26]:
```
columns = temperature_df_transformed.columns.tolist()
columns = [c for c in columns if c not in ['Block', 'Date']]

temperature_df_transformed = temperature_df_transformed[
```

```
    columns
].groupby(['Date/time']).mean()
```

And here we get an average per day

In [27]:
```
temperature_df_transformed = temperature_df_transformed.groupby(['Day']).mea

temperature_df_transformed.head(48)
```

Out[27]:

Day	Room	GSW+FM	AFLN+MU	AFLN+FB	AFLH+FM	AFLH+MU	AFLH	G
0.0	20.633333	15.050000	13.016667	13.583333	16.200000	15.200000	17.350000	14.
1.0	21.856944	15.818056	14.388889	15.144444	17.372222	16.240278	18.441667	15.
2.0	21.573611	17.466667	16.966667	19.266667	19.572222	18.286111	20.523611	17
3.0	20.890278	18.333333	21.704167	29.616667	25.413889	24.113889	28.254167	18.
4.0	22.237500	18.868056	28.381944	39.888889	35.730556	31.930556	36.127778	18.
5.0	20.786111	19.994444	32.550000	49.048611	42.369444	36.683333	41.041667	19.
6.0	19.129167	20.993056	34.204167	55.452778	45.705556	39.013889	43.286111	19.
7.0	18.943056	22.746528	34.425694	57.953472	47.738194	40.188889	44.011806	19.
8.0	19.161111	25.415278	33.933333	57.829167	49.338889	40.626389	43.538889	19
9.0	18.826389	28.722222	33.093056	57.052778	50.993056	40.662500	42.847222	20.
10.0	18.908333	31.872222	32.070833	56.180556	52.209722	40.552778	42.100000	21.
11.0	19.327778	34.338889	31.380556	55.630556	52.851389	40.568056	41.530556	23.
12.0	19.740278	36.340278	31.140278	55.518056	52.970833	40.811111	41.352778	26.
13.0	20.055556	38.181944	31.036111	55.463889	52.719444	41.112500	41.301389	28.
14.0	20.684722	40.238889	31.150000	55.331944	52.305556	41.465278	41.484722	30.
15.0	19.837500	41.933333	30.950000	54.540278	51.658333	41.505556	41.291667	33.
16.0	19.513889	42.950000	30.558333	53.726389	50.972222	41.265278	40.702778	35.
17.0	19.466667	43.098611	30.176389	53.225000	50.431944	40.901389	39.797222	38.
18.0	19.352778	42.391667	30.002778	53.016667	50.029167	40.436111	38.690278	42.
19.0	19.268056	40.662500	29.648611	52.513889	49.441667	39.759722	37.255556	45.
20.0	19.554167	38.265278	29.504167	51.797222	48.962500	39.126389	35.791667	46.
21.0	19.391667	35.541667	29.512500	50.663889	48.481944	38.555556	34.643056	46.
22.0	19.555556	32.750000	29.483333	48.772222	47.806944	37.883333	33.405556	46.

Day	Room	GSW+FM	AFLN+MU	AFLN+FB	AFLH+FM	AFLH+MU	AFLH	G
23.0	19.736111	30.334722	29.562500	46.428472	47.033333	37.269444	31.996528	45.
24.0	19.583333	28.247222	29.637500	43.452778	46.029167	36.640278	30.872222	43.
25.0	19.676389	26.583333	29.709722	40.380556	44.850000	35.988889	29.818056	40.
26.0	19.987500	25.388889	29.880556	37.493056	43.622222	35.412500	29.050000	36.
27.0	20.213889	24.625000	30.197222	34.990278	42.368056	34.955556	28.550000	33.
28.0	20.126389	24.076389	30.437500	32.784722	41.006944	34.494444	28.123611	30.
29.0	19.077778	21.125490	29.237255	28.896078	39.374510	29.945098	27.639216	27.
30.0	19.716667	19.059649	36.620635	28.563158	34.949206	27.346032	25.342105	24.
31.0	19.604167	21.755556	48.365278	32.926389	33.051389	30.370833	24.781944	24.
32.0	19.536111	23.954167	53.250000	37.709722	34.015278	32.484722	25.415278	25.
33.0	19.458333	25.322222	55.157639	42.428472	34.277083	33.833333	25.690972	26.
34.0	19.323611	25.795833	55.471528	45.246528	34.168056	34.597222	25.822917	25.
35.0	19.291667	25.663194	54.611806	45.255556	33.779167	34.954167	25.870139	25.
36.0	19.072917	25.299306	53.150000	43.871528	33.183333	35.057639	25.815278	24.
37.0	18.875694	24.763194	51.347222	42.332639	32.292361	35.002778	25.608333	23.
38.0	19.395139	24.509722	49.489583	41.443750	31.481944	35.025000	25.603472	22.
39.0	19.386111	24.606944	47.943750	40.118056	30.906250	35.255556	25.858333	22.
40.0	19.439583	24.854861	46.464583	38.180556	30.348611	35.388889	26.137500	22.
41.0	19.597917	25.242361	44.731944	36.332639	29.777083	35.400000	26.395833	22.
42.0	19.703472	25.840278	43.031250	34.884722	29.271528	35.359028	26.676389	22.
43.0	19.740278	26.594444	41.473611	33.745833	28.834722	35.219444	26.963194	22.
44.0	20.084028	27.445139	39.754167	32.629167	28.391667	34.857639	27.297917	22.
45.0	20.047917	28.553472	38.179861	31.824306	28.113194	34.529861	27.786806	22.

Day	Room	GSW+FM	AFLN+MU	AFLN+FB	AFLH+FM	AFLH+MU	AFLH	G
46.0	19.833333	29.692361	36.638889	31.029167	27.865278	34.136111	27.900000	22.5

Change of requirements

We don't need the treatments below anymore due to change of requirements. Therefore we can drop them

In [28]:
```
not_needed_treatments = ['AFLH+FM', 'AFLH+FB', 'AFLH+MU', 'AFLH']
```

In [29]:
```
temperature_df_transformed.drop(
    columns = not_needed_treatments,
    inplace = True
)
```

In [30]:
```
temperature_df_transformed.columns
```

Out[30]:
```
Index(['Room', 'GSW+FM', 'AFLN+MU', 'AFLN+FB', 'GSW+FB', 'AFLN+FM', 'GSW+MU',
       'Relative_time', 'Hours'],
      dtype='object')
```

We need to change the treatments that begin with “AFLN” to “AF” to correspond to the text in the article

In [31]:
```
temperature_df_transformed = temperature_df_transformed.rename(
    columns = {
        'AFLN+MU': 'AF+MU',
        'AFLN+FB': 'AF+FB',
        'AFLN+FM': 'AF+FM',
              }
)
```

We also need to change the treatments that begin with “GSW” to “GS”

In [32]:
```
temperature_df_transformed = temperature_df_transformed.rename(
    columns = {
        'GSW+FM': 'GS+FM',
        'GSW+FB': 'GS+FB',
        'GSW+MU': 'GS+MU',
              }
)
```

Visualize Temperature

Select treatments to visualize

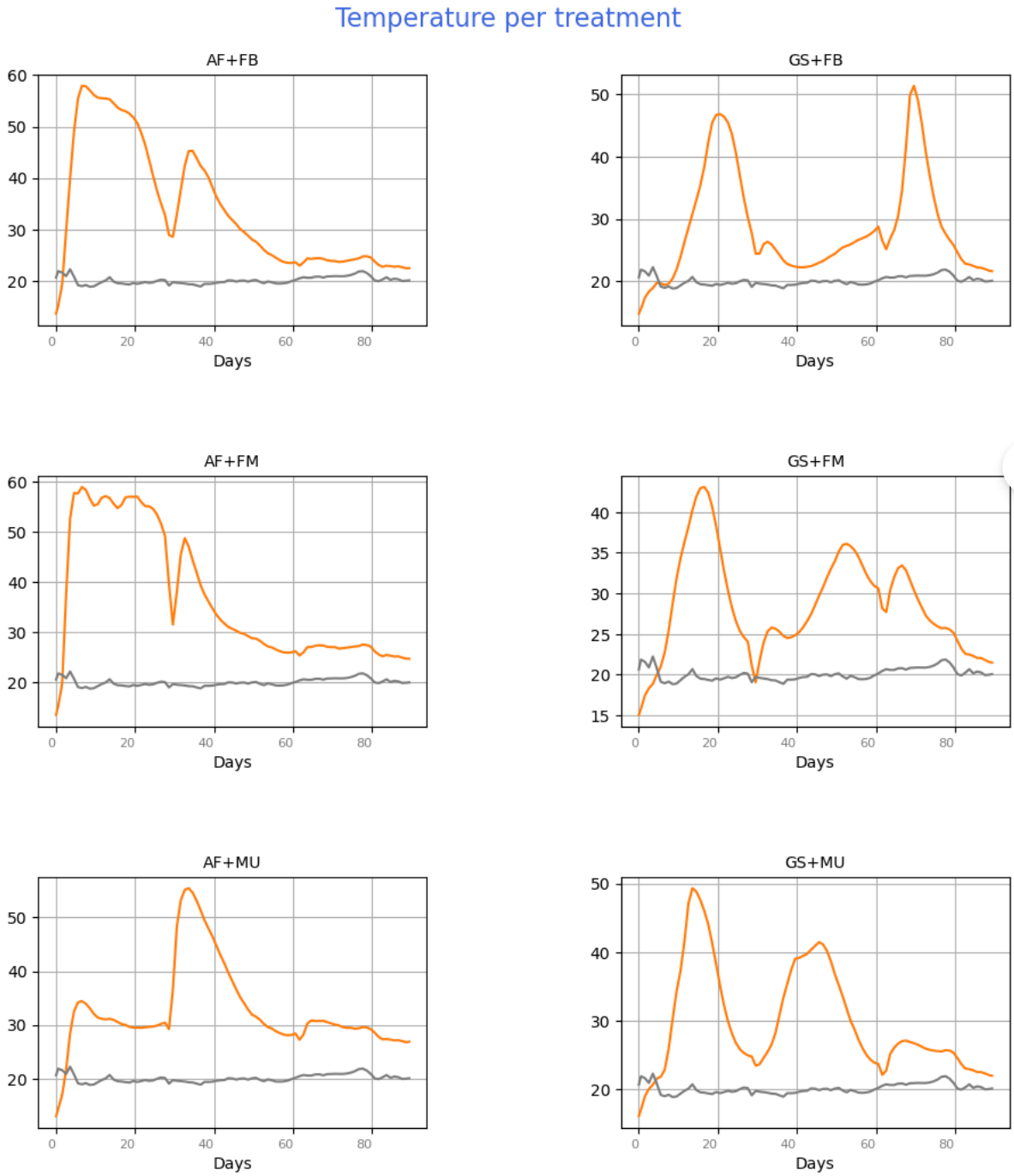
In [33]:
```
temperature_columns = [
    'AF+FB', 'GS+FB',
    'AF+FM', 'GS+FM',
```

```
    'AF+MU', 'GS+MU'
]
```

In [34]:
```
def plot_temperature(df, date_column, column, ax, plot_room_temperature):

    is_room_temperature_plotted = False

    if plot_room_temperature:
        columns = [column, date_column, 'Room']
    else:
        columns = [column, date_column]

    ax.plot(date_column, column, data = df, color = 'tab:orange')

    if plot_room_temperature and not is_room_temperature_plotted:

        ax.plot(date_column, 'Room', data = df, color = 'tab:gray')

    ax.set_title(column, loc = 'center', fontsize = 'medium')
    ax.set_ylabel('')
    ax.set_xlabel('Days')
    ax.grid(True)

    for label in ax.get_xticklabels():
        label.set_ha('right')
        label.set_color('tab:gray')
        label.set_size(8)
```

In [35]:
```
def plot_temperature_for_multiple_columns(
    df,
    date_column,
    title = 'Temperature per treatment',
    plot_room_temperature = True,
    columns = temperature_columns
):
    n = math.ceil(len(temperature_columns) / 2)

    fig = plt.figure(figsize = (11, 4 * n))
    fig.subplots_adjust(hspace = 0.6, wspace = 0.5)

    fig.suptitle(
        title,
        color = 'royalblue',
        fontsize = 16,
        y = 0.93
    )

    for i, col in enumerate(temperature_columns):

        ax = fig.add_subplot(n, 2, i + 1)
        plot_temperature(df, date_column, col, ax, plot_room_temperature)
```

In [36]:
```
plot_temperature_for_multiple_columns(temperature_df_transformed, 'Hours')
```



Parameters for Accumulated Temperature

We transform dataset according to the method described in this research paper:
Thermal Load and Application

Set parameters

In [37]:
```
ROOM_TEMPERATURE = 20
THERMOPHILIC_TEMPERATURE = 45
TEMPERATURE_TO_STERILIZE = 55
```

In [38]:
```
temperature_columns
```

Out[38]:
```
['AF+FB', 'GS+FB', 'AF+FM', 'GS+FM', 'AF+MU', 'GS+MU']
```

Methods for accumulating temperature

1. Calculate the difference between treatment temperature and base (reference) temperature
2. If the treatment temperature is lower than the base (reference) temperature use 0

In [39]:
```
def get_temperature_relative_to_base_temperature(data, column, base_temperat

    return np.where(
        ((data[column] - base_temperature) <= 0),
        0,
        data[column] - base_temperature
    )
```

Calculate accumulated(integral) treatment temperature when it has values above the base temperature

In [40]:
```
def get_accumulated_temperature(data, column, base_temperature):

    data['temp_column'] = data[column].cumsum()
    data[column] = data['temp_column']
    data.drop('temp_column', axis = 1, inplace = True)

    return data
```

Calculate accumulated temperature per each block(bottle) and combine subsets together into one dataset

In [41]:
```
def get_accumulated_temperature_for_all_columns(
    data,
    base_temperature,
    columns = temperature_columns
):

    for temp_column in columns:
        data = get_accumulated_temperature(
            data, temp_column, base_temperature
        )

    return data
```

In [42]:
```
def get_dataset_with_accumulated_temperature(
    data,
    base_temperature,
    columns = temperature_columns
):
    for col in columns:
        data[col] = get_temperature_relative_to_base_temperature(
            data,
            col,
            base_temperature
        )

    selected_columns = ['Hours']
    selected_columns.extend(columns)

    data = data[selected_columns]
```

```
    data = get_accumulated_temperature_for_all_columns(data, base_temperatu
    return data
```

Accumulated Temperature Based on Room

In [43]:
```
base_temperature = ROOM_TEMPERATURE
```

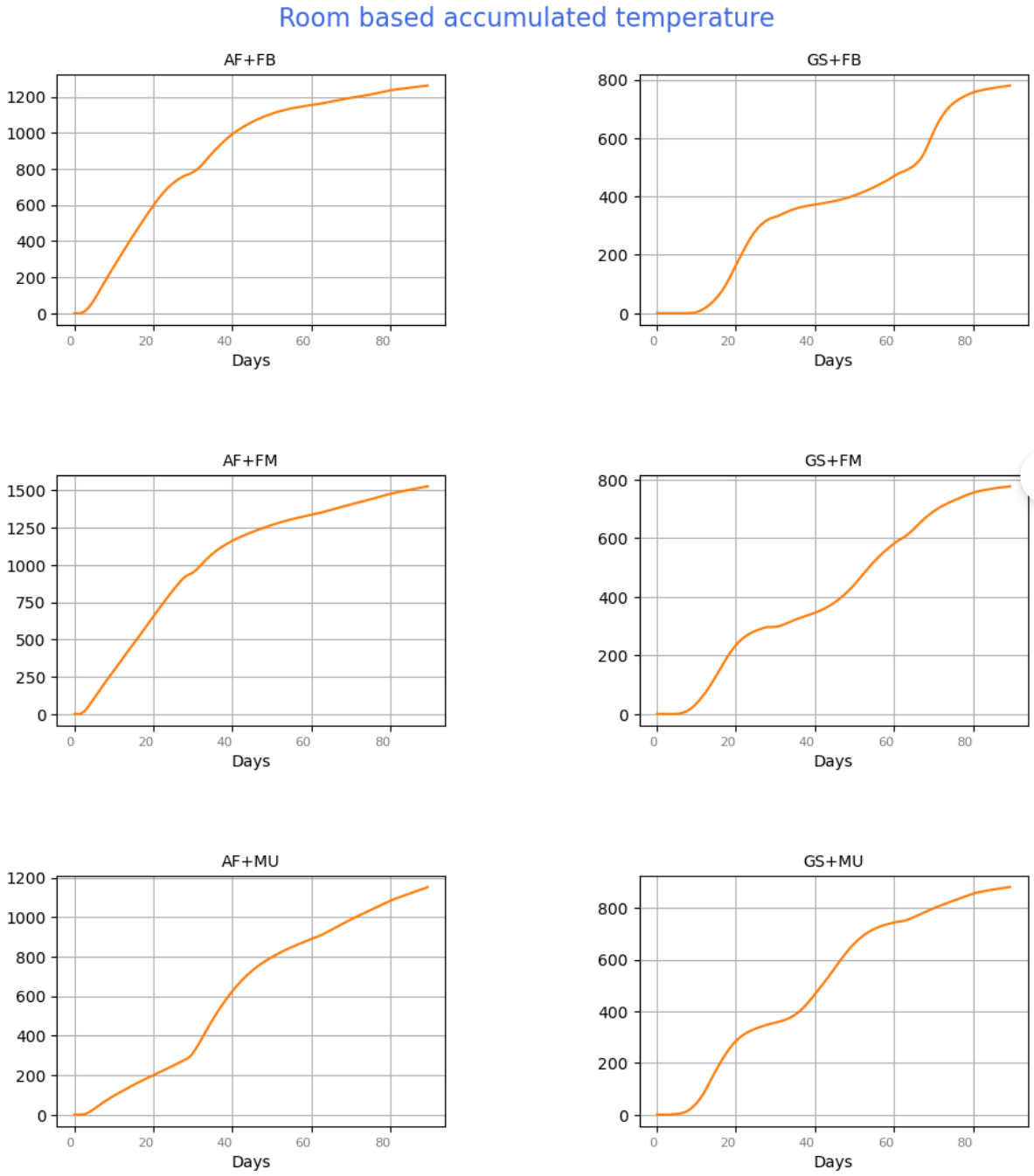
In [44]:
```
title = 'accumulated temperature'
```

In [45]:
```
room_cumulative_temperature_df = temperature_df_transformed.copy()
```

In [46]:
```
room_cumulative_temperature_df = get_dataset_with_accumulated_temperature(
    room_cumulative_temperature_df,
    base_temperature,
    columns = temperature_columns
)
```

Visualize the accumulated temperature per treatment

In [47]:
```
plot_temperature_for_multiple_columns(
    room_cumulative_temperature_df,
    'Hours',
    title = 'Room based ' + title,
    plot_room_temperature = False
)
```



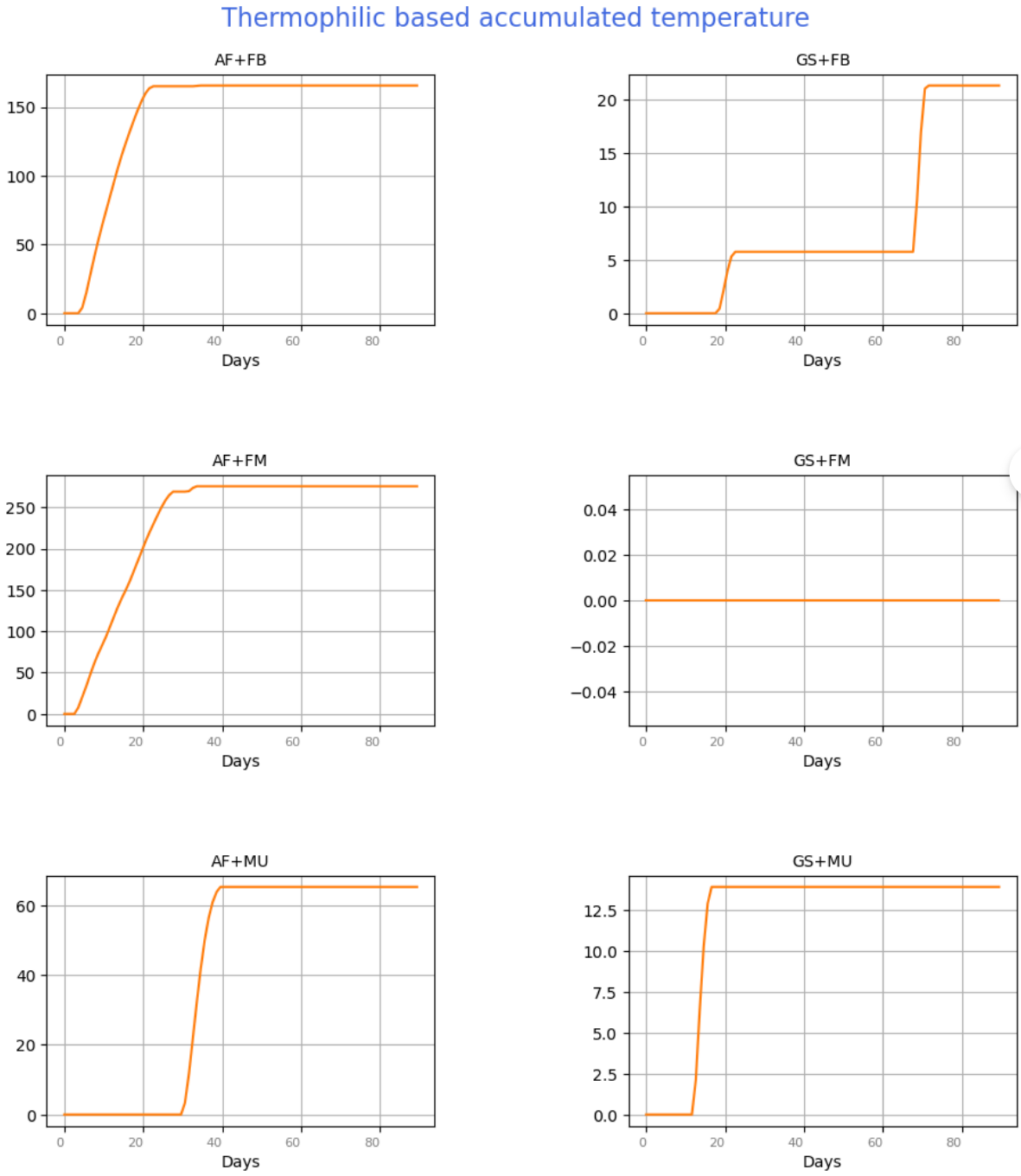
Accumulated Temperature Based on Thermophilic

In [48]:
```
base_temperature = THERMOPHILIC_TEMPERATURE
```

In [49]:
```
thermophilic_cumulative_temperature_df = temperature_df_transformed.copy()
```

In [50]:
```
thermophilic_cumulative_temperature_df = get_dataset_with_accumulated_tempe
    thermophilic_cumulative_temperature_df,
    base_temperature,
    columns = temperature_columns
)
```

In [51]:
```
plot_temperature_for_multiple_columns(
    thermophilic_cumulative_temperature_df,
    'Hours',
    title = 'Thermophilic based ' + title,
    plot_room_temperature = False
)
```

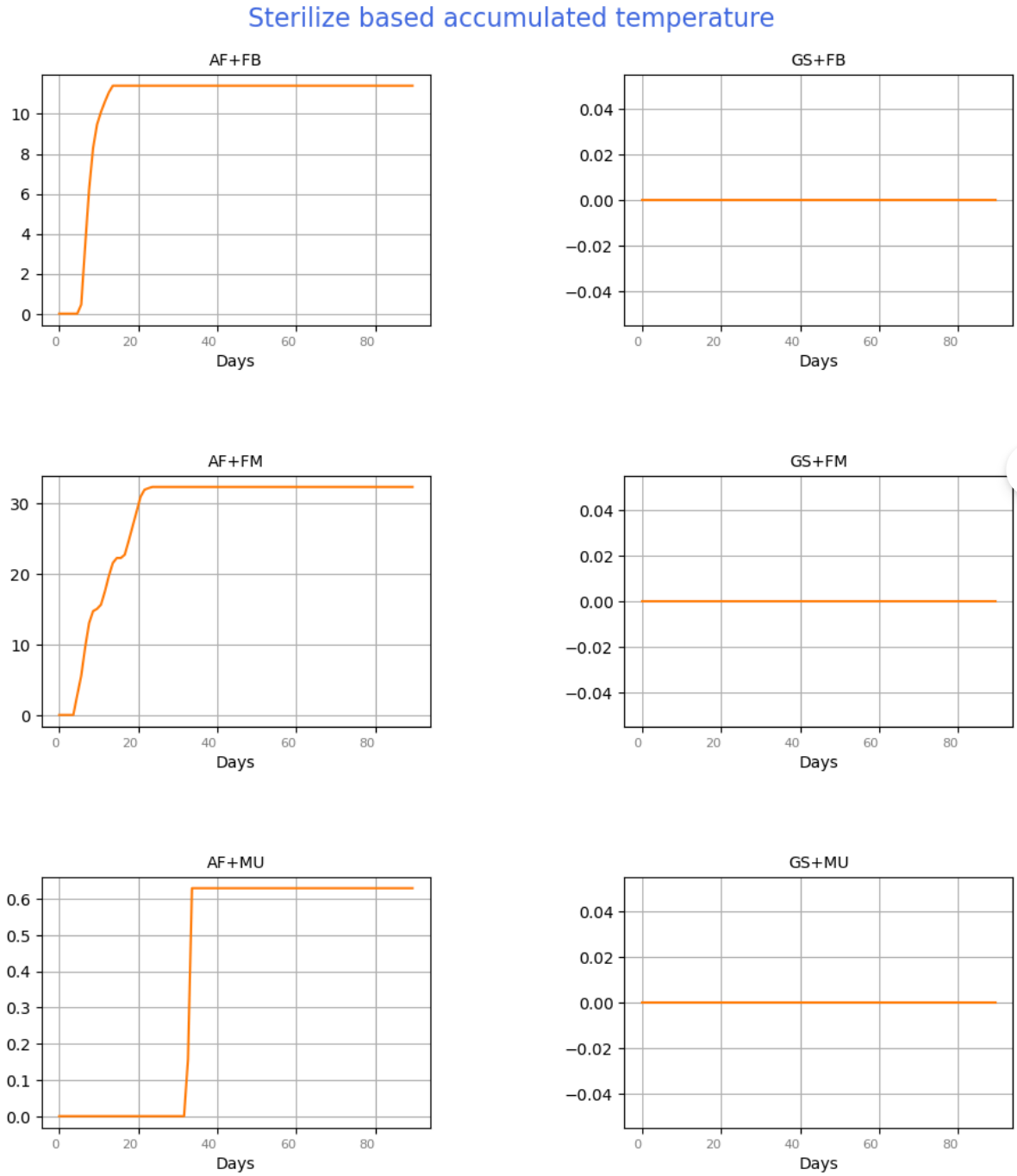


Accumulated Temperature Based on Sterilize

In [52]:
```
base_temperature = TEMPERATURE_TO_STERILIZE
```

In [53]:
```
sterilize_cumulative_temperature_df = temperature_df_transformed.copy()

sterilize_cumulative_temperature_df = get_dataset_with_accumulated_temperatu
    sterilize_cumulative_temperature_df,
    base_temperature,
    columns = temperature_columns
)
```

In [54]:
```
plot_temperature_for_multiple_columns(
    sterilize_cumulative_temperature_df,
    'Hours',
    title = 'Sterilize based ' + title,
    plot_room_temperature = False
)
```



Save data

In [55]:
```
cumulative_temperature_dataframes = {
    'room based cumulative temperature average.xlsx': room_cumulative_tempe
    'thermophilic cumulative temperature average.xlsx': thermophilic_cumula
    'sterilize cumulative temperature average.xlsx': sterilize_cumulative_te
}
```

In [56]:
```
output_data_filepath = project_base_path + '/data/output/'
task_filepath = 'Average Accumulated Temperature Generation in relation to B
```

In [57]:
```
for local_file_path, dataframe in cumulative_temperature_dataframes.items():

    filepath = output_data_filepath + task_filepath + local_file_path
    dataframe.to_excel(filepath, index = False)
```