BA305_project

April 28, 2025

1 Imports

[ ]:
```
# import google drive files
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
!pip install -q condacolab
import condacolab
condacolab.install()
!conda install -c conda-forge ipopt
```

```
   Everything looks OK!
Channels:
 - conda-forge
Platform: linux-64
Collecting package metadata (repodata.json): - \ | / - \ | /
done
Solving environment: \ | / - \ done


==> WARNING: A newer version of conda exists. <==
    current version: 24.11.3
    latest version: 25.3.1

Please update conda by running

    $ conda update -n base -c conda-forge conda



# All requested packages already installed.
```

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
```

[ ]:
```
!pip install openpyxl
!pip install pyomo
import pyomo.environ as pyo
from pyomo.environ import *
from pyomo.opt import SolverFactory

# Install optimization engine (solution algorithms)
# Linear program solver https://www.gnu.org/software/glpk/
# !apt-get install -y -qq glpk-utils
# !which glpsol
# !apt-get update
# !apt-get install -y coinor-libipopt1v5 coinor-ipopt
```

```
Requirement already satisfied: openpyxl in /usr/local/lib/python3.11/site-
packages (3.1.5)
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.11/site-
packages (from openpyxl) (2.0.0)
Requirement already satisfied: pyomo in /usr/local/lib/python3.11/site-packages
(6.9.2)
Requirement already satisfied: ply in /usr/local/lib/python3.11/site-packages
(from pyomo) (3.11)
```

[ ]:
```
# # INFORMS dataset contextual data
# logbook = pd.read_excel('/content/drive/MyDrive/Senior Year/BA305/BA305␣
 ↪Project/INFORMS Datasets 2025/Contextual quality data/Logbook Long Run Days.
 ↪xlsx')
# rm_content_uniformity = pd.read_excel('/content/drive/MyDrive/Senior Year/
 ↪BA305/BA305 Project/INFORMS Datasets 2025/Contextual quality data/RM Content␣
 ↪Uniformity.xlsx')
# rm_material_properties = pd.read_excel('/content/drive/MyDrive/Senior Year/
 ↪BA305/BA305 Project/INFORMS Datasets 2025/Contextual quality data/RM␣
 ↪Material Properties.xlsx')
# rm_tablet_properties_and_drum_change = pd.read_excel('/content/drive/MyDrive/
 ↪Senior Year/BA305/BA305 Project/INFORMS Datasets 2025/Contextual quality␣
 ↪data/RM Tablet Properties and Drum Change.xlsx')
```

2 Data Cleaning

[ ]:
```
# machine data
# Replace with where you have data located
blenders = pd.read_csv('/content/drive/MyDrive/Senior Year/BA305/BA305 Project/
 ↪INFORMS Datasets 2025/Machine data/Blenders.csv')
humidity = pd.read_csv('/content/drive/MyDrive/Senior Year/BA305/BA305 Project/
 ↪INFORMS Datasets 2025/Machine data/Humidity.csv')
liw_feeders1 = pd.read_csv('/content/drive/MyDrive/Senior Year/BA305/BA305␣
 ↪Project/INFORMS Datasets 2025/Machine data/LiW Feeders 1.csv')
liw_feeders2 = pd.read_csv('/content/drive/MyDrive/Senior Year/BA305/BA305␣
 ↪Project/INFORMS Datasets 2025/Machine data/LiW Feeders 2.csv')
tablet_press = pd.read_csv('/content/drive/MyDrive/Senior Year/BA305/BA305␣
 ↪Project/INFORMS Datasets 2025/Machine data/Tablet Press.csv')
temperature = pd.read_csv('/content/drive/MyDrive/Senior Year/BA305/BA305␣
 ↪Project/INFORMS Datasets 2025/Machine data/Temperature.csv')
```

[ ]:
```
# basic EDA of the data
#display(blenders)
# Convert timestamp to datetime, convert massflow blender 1 and 2 to numerical␣
 ↪to represent kg / hr, blend potency 1 and 2 to represent
# relative percentage, and OOS Concentration to represent percentage, removing␣
 ↪any values that aren't numeric (columns are all objects right now)
blenders['TimeStamp'] = pd.to_datetime(blenders['TimeStamp'], format = 'mixed',␣
 ↪dayfirst = True, errors = 'coerce')
# Rename and convert columns to numeric using dictionary and inplace=True
blenders.rename(columns={
    'Massflow Blender 1': 'Massflow_Blender_1',
    'Massflow Blender 2': 'Massflow_Blender_2',
    'Blend Potency Blender 1': 'Blend_Potency_Blender_1',
    'Blend Potency Blender 2': 'Blend_Potency_Blender_2',
    'OOS Concentration at Blender 1 inlet':␣
 ↪'OOS_Concentration_at_Blender_1_inlet',
    'OOS Concentration at Blender 2 inlet':␣
 ↪'OOS_Concentration_at_Blender_2_inlet'
}, inplace=True)

# Convert to numeric
for col in ['Massflow_Blender_1', 'Massflow_Blender_2',␣
 ↪'Blend_Potency_Blender_1', 'Blend_Potency_Blender_2',␣
 ↪'OOS_Concentration_at_Blender_1_inlet',␣
 ↪'OOS_Concentration_at_Blender_2_inlet']:
    blenders[col] = pd.to_numeric(blenders[col], errors='coerce')
# Drop na and drop if column is binary or if entire column is 0
blenders.dropna(inplace=True)
blenders = blenders.loc[:, (blenders != 0).any(axis=0)]
```

```
blenders.info()
blenders.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 439844 entries, 0 to 439845
Data columns (total 7 columns):
 #   Column                                Non-Null Count   Dtype
---  ------                                --------------   -----
 0   TimeStamp                             439844 non-null  datetime64[ns]
 1   Massflow_Blender_1                    439844 non-null  float64
 2   Massflow_Blender_2                    439844 non-null  float64
 3   Blend_Potency_Blender_1               439844 non-null  float64
 4   Blend_Potency_Blender_2               439844 non-null  float64
 5   OOS_Concentration_at_Blender_1_inlet  439844 non-null  float64
 6   OOS_Concentration_at_Blender_2_inlet  439844 non-null  float64
dtypes: datetime64[ns](1), float64(6)
memory usage: 26.8 MB
```

[ ]:

	TimeStamp	Massflow_Blender_1	Massflow_Blender_2	\
0	2018-12-01 08:43:00	0.0	0.0	
1	2018-12-01 08:43:00	0.0	0.0	
2	2018-12-01 08:43:00	0.0	0.0	
3	2018-12-01 08:43:00	0.0	0.0	
4	2018-12-01 08:43:00	0.0	0.0	

	Blend_Potency_Blender_1	Blend_Potency_Blender_2	\
0	100.0	100.0	
1	100.0	100.0	
2	100.0	100.0	
3	100.0	100.0	
4	100.0	100.0	

	OOS_Concentration_at_Blender_1_inlet	OOS_Concentration_at_Blender_2_inlet
0	0.0	0.0
1	0.0	0.0
2	0.0	0.0
3	0.0	0.0
4	0.0	0.0

[ ]:
```
humidity = humidity.rename(columns={
    'Feeder': 'Feeder_TimeStamp',
    'Unnamed: 2': 'Feeder_Relative_Humidity_Pct',
    'Tablet Press': 'Tablet_Press_TimeStamp',
    'Unnamed: 5': 'Tablet_Press_Relative_Humidity_Pct'})
humidity.drop(columns=['Unnamed: 1', 'Unnamed: 4'], inplace = True)
#display(humidity.info())
for col in ['Feeder_Relative_Humidity_Pct',␣
 ↪'Tablet_Press_Relative_Humidity_Pct']:
```

```
    humidity[col] = pd.to_numeric(humidity[col], errors='coerce')
for col in ['Feeder_TimeStamp', 'Tablet_Press_TimeStamp']:
    humidity[col] = pd.to_datetime(humidity[col], format = 'mixed', dayfirst =␣
 ↪True, errors = 'coerce')

humidity.dropna(inplace=True)
humidity = humidity.loc[:, (humidity != 0).any(axis=0)]
humidity.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 492 entries, 1 to 492
Data columns (total 4 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Feeder_TimeStamp                  492 non-null    datetime64[ns]
 1   Feeder_Relative_Humidity_Pct      492 non-null    float64
 2   Tablet_Press_TimeStamp            492 non-null    datetime64[ns]
 3   Tablet_Press_Relative_Humidity_Pct  492 non-null    float64
dtypes: datetime64[ns](2), float64(2)
memory usage: 19.2 KB
```

[ ]:
```
print(humidity)
```

```
       Feeder_TimeStamp  Feeder_Relative_Humidity_Pct Tablet_Press_TimeStamp  \
1   2018-01-12 08:13:00                        44.242    2018-01-12 08:15:00
2   2018-01-12 08:28:00                        44.242    2018-01-12 08:30:00
3   2018-01-12 08:43:00                        43.605    2018-01-12 08:45:00
4   2018-01-12 08:58:00                        43.605    2018-01-12 09:00:00
5   2018-01-12 09:13:00                        43.605    2018-01-12 09:15:00
..                  …                           …                    …
488 2018-01-17 09:58:00                        29.869    2018-01-17 10:00:00
489 2018-01-17 10:13:00                        29.869    2018-01-17 10:15:00
490 2018-01-17 10:28:00                        29.869    2018-01-17 10:30:00
491 2018-01-17 10:43:00                        29.869    2018-01-17 10:45:00
492 2018-01-17 10:58:00                        31.208    2018-01-17 11:00:00

     Tablet_Press_Relative_Humidity_Pct
1                                34.140
2                                33.485
3                                33.485
4                                31.511
5                                32.829
..                                  …
488                               8.896
489                               8.896
490                              11.020
491                               9.606
492                              10.314
```

[492 rows x 4 columns]

[ ]:
```
display(liw_feeders1)
```

```
              TimeStamp Feed Factor PD1 Feed Factor PD2 Feed Factor PD3  \
0       01/12/2018 08:43          1.3334          1.9152          1.3027
1       01/12/2018 08:43          1.3334          1.9152          1.3027
2       01/12/2018 08:43          1.3334          1.9152          1.3027
3       01/12/2018 08:43          1.3334          1.9152          1.3027
4       01/12/2018 08:43          1.3334          1.9152          1.3027
…                    …               …               …               …
439841  17/01/2018 11:01           1.273          1.7314          1.0819
439842  17/01/2018 11:01          1.2785          1.7333          1.0896
439843  17/01/2018 11:01           1.259          1.7341          1.0873
439844  17/01/2018 11:01          1.2684          1.7312          1.0874
439845  17/01/2018 11:01          1.2727          1.7326          1.0887

       Feed Factor PD4 Feed Factor PD5 Feed Factor PD7 Screw RPM PD1  \
0               1.3089          1.1295         0.83728           0.0
1               1.3089          1.1295         0.83728           0.0
2               1.3089          1.1295         0.83728           0.0
3               1.3089          1.1295         0.83728           0.0
4               1.3089          1.1295         0.83728           0.0
…                  …               …               …               …
439841          1.3218          1.1042         0.80946      203.6408
439842          1.3273          1.1048         0.80929      203.3338
439843          1.3227          1.1057         0.81131      207.0689
439844          1.3188          1.1067         0.81018      205.4316
439845          1.3193          1.1083         0.81219      183.7372

       Screw RPM PD2 Screw RPM PD3  … Unnamed: 45 Unnamed: 46 Unnamed: 47  \
0                0.0           0.0  …         NaN         NaN         NaN
1                0.0           0.0  …         NaN         NaN         NaN
2                0.0           0.0  …         NaN         NaN         NaN
3                0.0           0.0  …         NaN         NaN         NaN
4                0.0           0.0  …         NaN         NaN         NaN
…                …             …    …           …           …           …
439841      152.4236       83.4518  …         NaN         NaN         NaN
439842      153.1399        83.196  …         NaN         NaN         NaN
439843      153.2423       84.9356  …         NaN         NaN         NaN
439844      152.6283       84.3728  …         NaN         NaN         NaN
439845       137.739       74.4466  …         NaN         NaN         NaN

       Unnamed: 48 Unnamed: 49 Unnamed: 50 Unnamed: 51 Unnamed: 52  \
0              NaN         NaN         NaN         NaN         NaN
1              NaN         NaN         NaN         NaN         NaN
2              NaN         NaN         NaN         NaN         NaN
```

```
3              NaN          NaN          NaN          NaN          NaN
4              NaN          NaN          NaN          NaN          NaN
…              …            …            …            …            …
439841         NaN          NaN          NaN          NaN          NaN
439842         NaN          NaN          NaN          NaN          NaN
439843         NaN          NaN          NaN          NaN          NaN
439844         NaN          NaN          NaN          NaN          NaN
439845         NaN          NaN          NaN          NaN          NaN

       Unnamed: 53 Unnamed: 54
0              NaN          NaN
1              NaN          NaN
2              NaN          NaN
3              NaN          NaN
4              NaN          NaN
…              …            …
439841         NaN          NaN
439842         NaN          NaN
439843         NaN          NaN
439844         NaN          NaN
439845         NaN          NaN

[439846 rows x 55 columns]
```

[ ]:
```
#display(liw_feeders1)
#display(liw_feeders1.dtypes)
liw_feeders1.info()
# Convert TimeStamp to datetime format
liw_feeders1['TimeStamp'] = pd.to_datetime(liw_feeders1['TimeStamp'],␣
 ↪format='mixed', dayfirst=True, errors='coerce')

# Rename columns and drop unnecessary columns
liw_feeders1.columns = liw_feeders1.columns.str.replace('  ', ' ').str.
 ↪replace(' ', '_')
cols_to_drop = [f'Unnamed:_{i}' for i in range(28, 55)] + [f'VolMode_PD{i}' for␣
 ↪i in range(1, 8)]
liw_feeders1 = liw_feeders1.drop(columns=cols_to_drop, errors='ignore')

#print(liw_feeders1.columns)
# Convert specified columns to numeric
numeric_columns = [
    'Feed_Factor_PD1', 'Feed_Factor_PD2', 'Feed_Factor_PD3', 'Feed_Factor_PD4',␣
 ↪'Feed_Factor_PD5', 'Feed_Factor_PD7',
    'Screw_RPM_PD1', 'Screw_RPM_PD2', 'Screw_RPM_PD3', 'Screw_RPM_PD4',␣
 ↪'Screw_RPM_PD5', 'Screw_RPM_PD7',
    'Massflow_PD_1', 'Massflow_PD_2', 'Massflow_PD_3', 'Massflow_PD_4',␣
 ↪'Massflow_PD_5', 'Massflow_PD_7',
```

```
    '%_PD1', '%_PD2', '%_PD3'
]

liw_feeders1[numeric_columns] = liw_feeders1[numeric_columns].apply(pd.
 ↪to_numeric, errors='coerce')

# Drop rows with NaN values
liw_feeders1.dropna(inplace=True)
liw_feeders1 = liw_feeders1.loc[:, (liw_feeders1 != 0).any(axis=0)]
liw_feeders1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 439846 entries, 0 to 439845
Data columns (total 55 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   TimeStamp        439846 non-null  object
 1   Feed Factor PD1  439846 non-null  object
 2   Feed Factor PD2  439846 non-null  object
 3   Feed Factor PD3  439846 non-null  object
 4   Feed Factor PD4  439846 non-null  object
 5   Feed Factor PD5  439846 non-null  object
 6   Feed Factor PD7  439846 non-null  object
 7   Screw RPM PD1    439846 non-null  object
 8   Screw RPM PD2    439846 non-null  object
 9   Screw RPM PD3    439846 non-null  object
 10  Screw RPM PD4    439846 non-null  object
 11  Screw RPM PD5    439846 non-null  object
 12  Screw RPM PD7    439846 non-null  object
 13  VolMode PD1      439846 non-null  object
 14  VolMode PD2      439846 non-null  object
 15  VolMode PD3      439846 non-null  object
 16  VolMode PD4      439846 non-null  object
 17  VolMode PD5      439846 non-null  object
 18  VolMode PD7      439846 non-null  object
 19  Massflow PD 1    439846 non-null  object
 20  Massflow PD 2    439846 non-null  object
 21  Massflow PD 3    439846 non-null  object
 22  Massflow PD 4    439846 non-null  object
 23  Massflow PD 5    439846 non-null  object
 24  Massflow PD 7    439846 non-null  object
 25  %  PD1           439846 non-null  object
 26  %  PD2           439846 non-null  object
 27  %  PD3           439846 non-null  object
 28  Unnamed: 28      0 non-null       float64
 29  Unnamed: 29      0 non-null       float64
 30  Unnamed: 30      0 non-null       float64
 31  Unnamed: 31      0 non-null       float64
```

```
 32  Unnamed: 32      0 non-null       float64
 33  Unnamed: 33      0 non-null       float64
 34  Unnamed: 34      0 non-null       float64
 35  Unnamed: 35      0 non-null       float64
 36  Unnamed: 36      0 non-null       float64
 37  Unnamed: 37      0 non-null       float64
 38  Unnamed: 38      0 non-null       float64
 39  Unnamed: 39      0 non-null       float64
 40  Unnamed: 40      0 non-null       float64
 41  Unnamed: 41      0 non-null       float64
 42  Unnamed: 42      0 non-null       float64
 43  Unnamed: 43      0 non-null       float64
 44  Unnamed: 44      0 non-null       float64
 45  Unnamed: 45      0 non-null       float64
 46  Unnamed: 46      0 non-null       float64
 47  Unnamed: 47      0 non-null       float64
 48  Unnamed: 48      0 non-null       float64
 49  Unnamed: 49      0 non-null       float64
 50  Unnamed: 50      0 non-null       float64
 51  Unnamed: 51      0 non-null       float64
 52  Unnamed: 52      0 non-null       float64
 53  Unnamed: 53      0 non-null       float64
 54  Unnamed: 54      0 non-null       float64
dtypes: float64(27), object(28)
memory usage: 184.6+ MB
<class 'pandas.core.frame.DataFrame'>
Index: 439844 entries, 0 to 439845
Data columns (total 22 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   TimeStamp        439844 non-null  datetime64[ns]
 1   Feed_Factor_PD1  439844 non-null  float64
 2   Feed_Factor_PD2  439844 non-null  float64
 3   Feed_Factor_PD3  439844 non-null  float64
 4   Feed_Factor_PD4  439844 non-null  float64
 5   Feed_Factor_PD5  439844 non-null  float64
 6   Feed_Factor_PD7  439844 non-null  float64
 7   Screw_RPM_PD1    439844 non-null  float64
 8   Screw_RPM_PD2    439844 non-null  float64
 9   Screw_RPM_PD3    439844 non-null  float64
 10  Screw_RPM_PD4    439844 non-null  float64
 11  Screw_RPM_PD5    439844 non-null  float64
 12  Screw_RPM_PD7    439844 non-null  float64
 13  Massflow_PD_1    439844 non-null  float64
 14  Massflow_PD_2    439844 non-null  float64
 15  Massflow_PD_3    439844 non-null  float64
 16  Massflow_PD_4    439844 non-null  float64
 17  Massflow_PD_5    439844 non-null  float64
```

```
 18  Massflow_PD_7    439844 non-null  float64
 19  %_PD1            439844 non-null  float64
 20  %_PD2            439844 non-null  float64
 21  %_PD3            439844 non-null  float64
dtypes: datetime64[ns](1), float64(21)
memory usage: 77.2 MB
```

[ ]:
```
display(liw_feeders2, liw_feeders2.columns)
```

```
         %  PD4  %  PD5  %  PD7 Estimated weight IBC PS1  \
0           0.0     0.0     0.0                  697.2808
1           0.0     0.0     0.0                  697.2808
2           0.0     0.0     0.0                  697.2808
3           0.0     0.0     0.0                  697.2808
4           0.0     0.0     0.0                  697.2808
…          …      …      …                      …
439841   28.411  2.9679  1.0281                2041.4215
439842  28.4275  2.9726  1.0272                2041.4215
439843  28.3322  2.9815  1.0264                2041.4215
439844  28.5676  2.9908  1.0258                2041.4215
439845  28.5676  2.9955  1.0255                2041.4215

       Estimated weight IBC PS2 Estimated weight IBC PS3  \
0                     997.8973                  22.2826
1                     997.8973                  22.2826
2                     997.8973                  22.2826
3                     997.8973                  22.2826
4                     997.8973                  22.2826
…                        …                      …
439841               9280.8457                  89.7484
439842               9280.8457                  89.7484
439843               9280.8457                  89.7484
439844               9280.8457                  89.7484
439845               9280.8457                  89.7484

       Estimated weight IBC PS4 Estimated weight IBC PS5  \
0                     755.6357                 240.9432
1                     755.6357                 240.9432
2                     755.6357                 240.9432
3                     755.6357                 240.9432
4                     755.6357                 240.9432
…                        …                      …
439841               4143.6841                 178.8231
439842               4143.6841                 178.8231
439843               4143.6841                 178.8231
439844               4143.6841                 178.8231
439845               4143.6841                 178.8231
```

```
       Estimated weight IBC PS7 RefAct PD1  … Net Weight PD3 Net Weight PD4  \
0                      141.9876          0  …        0.42855         0.5205
1                      141.9876          0  …        0.42855        0.52045
2                      141.9876          0  …        0.42855         0.5205
3                      141.9876          0  …        0.42855         0.5205
4                      141.9876          0  …        0.42855         0.5205
…                         …        …  …             …              …
439841                  81.5899          0  …        0.42515         0.4136
439842                  81.5899          0  …         0.4235        0.40975
439843                  81.5899          0  …         0.4223         0.4057
439844                  81.5899          0  …         0.4203        0.40055
439845                  81.5899          0  …        0.41895        0.39655

       Net Weight PD5 Net Weight PD7 Totalizer PD1 Totalizer PD2  \
0              0.8823        0.40355        2.5048        2.5454
1             0.88225        0.40355        2.5048        2.5454
2              0.8823        0.40355        2.5048        2.5454
3              0.8823        0.40355        2.5048        2.5454
4             0.88225        0.40355        2.5048        2.5454
…               …             …              …             …
439841         0.6818        0.38745       754.1812      759.0052
439842         0.6813        0.38775       754.1844      759.0084
439843        0.68095        0.38745       754.1912      759.0153
439844         0.6803          0.387       754.1981      759.0221
439845        0.67995        0.38705       754.2053      759.0293

       Totalizer PD3 Totalizer PD4 Totalizer PD5 Totalizer PD7
0            0.87711        2.5031       0.26459      0.072622
1            0.87711        2.5031       0.26459      0.072622
2            0.87711        2.5031       0.26459      0.072622
3            0.87711        2.5031       0.26459      0.072622
4            0.87711        2.5031       0.26459      0.072622
…              …             …              …            …
439841       266.002      758.0178       131.072        65.536
439842       266.002      758.0245       131.072        65.536
439843       266.002      758.0314       131.072        65.536
439844       266.002      758.0385       131.072        65.536
439845       266.002      758.0449       131.072        65.536

[439846 rows x 27 columns]
```

```
Index(['%  PD4', '%  PD5', '%  PD7', 'Estimated weight IBC PS1',
       'Estimated weight IBC PS2', 'Estimated weight IBC PS3',
       'Estimated weight IBC PS4', 'Estimated weight IBC PS5',
       'Estimated weight IBC PS7', 'RefAct PD1', 'RefAct PD2', 'RefAct PD3',
       'RefAct PD4', 'RefAct PD5', 'RefAct PD7', 'Net Weight PD1',
       'Net Weight PD2', 'Net Weight PD3', 'Net Weight PD4', 'Net Weight PD5',
       'Net Weight PD7', 'Totalizer PD1', 'Totalizer PD2', 'Totalizer PD3',
```

```
       'Totalizer PD4', 'Totalizer PD5', 'Totalizer PD7'],
      dtype='object')
```

[ ]:
```
#
#display(liw_feeders2.dtypes)
display(liw_feeders2.columns, liw_feeders2.info())

# Drop unnecessary columns
liw_feeders2.columns = liw_feeders2.columns.str.replace('  ', ' ').str.
 ↪replace(' ', '_')
cols_to_drop = [f'RefAct_PD{i}' for i in range(1, 8)]
liw_feeders2 = liw_feeders2.drop(columns=cols_to_drop, errors='ignore')

# Convert specified columns to numeric
numeric_columns = [
    '%_PD4', '%_PD5', '%_PD7',
    'Estimated_weight_IBC_PS1', 'Estimated_weight_IBC_PS2',␣
 ↪'Estimated_weight_IBC_PS3', 'Estimated_weight_IBC_PS4',␣
 ↪'Estimated_weight_IBC_PS5', 'Estimated_weight_IBC_PS7',
    'Net_Weight_PD1', 'Net_Weight_PD2', 'Net_Weight_PD3', 'Net_Weight_PD4',␣
 ↪'Net_Weight_PD5', 'Net_Weight_PD7',
    'Totalizer_PD1', 'Totalizer_PD2', 'Totalizer_PD3', 'Totalizer_PD4',␣
 ↪'Totalizer_PD5', 'Totalizer_PD7',
]

liw_feeders2[numeric_columns] = liw_feeders2[numeric_columns].apply(pd.
 ↪to_numeric, errors='coerce')

# Drop rows with NaN values
liw_feeders2.dropna(inplace=True)
liw_feeders2 = liw_feeders2.loc[:, (liw_feeders2 != 0).any(axis=0)]
liw_feeders2.info()
```

[ ]:
```
display(tablet_press.columns, tablet_press.info())
# import pandas as pd

# Convert TimeStamp to datetime format
tablet_press['TimeStamp'] = pd.to_datetime(tablet_press['TimeStamp'],␣
 ↪format='mixed', dayfirst=True, errors='coerce')

# Convert specified columns to numeric
numeric_columns = [
    'Pre-compression height bottom', 'Pre-compresion top dwell time',␣
 ↪'Pre-compression force',
    'Pre-compression displacement top sigma', 'Main compression height bottom',␣
 ↪'Main compression top dwell time',
```

```
    'Main compression force', 'Compression cycle fill depth', 'Filling Shoe␣
 ↪M20M13 speed',
    'Filling Shoe M20M23 speed', 'Material inlet: Hopper level detection',␣
 ↪'Ejection force tablet'
]
# Replace spaces with underscores
tablet_press.columns = tablet_press.columns.str.replace(' ', '_')
# Add underscores to numeric_columns
numeric_columns = [col.replace(' ', '_') for col in numeric_columns]

tablet_press[numeric_columns] = tablet_press[numeric_columns].apply(pd.
 ↪to_numeric, errors='coerce')

# Drop rows with NaN values
tablet_press.dropna(inplace=True)
tablet_press = tablet_press.loc[:, (tablet_press != 0).any(axis=0)]
tablet_press.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 439846 entries, 0 to 439845
Data columns (total 13 columns):
 #   Column                                 Non-Null Count   Dtype
---  ------                                 --------------   -----
 0   TimeStamp                              439846 non-null  object
 1   Pre-compression height bottom          439846 non-null  object
 2   Pre-compresion top dwell time          439846 non-null  object
 3   Pre-compression force                  439846 non-null  object
 4   Pre-compression displacement top sigma 439846 non-null  object
 5   Main compression height bottom         439846 non-null  object
 6   Main compression top dwell time        439846 non-null  object
 7   Main compression force                 439846 non-null  object
 8   Compression cycle fill depth           439846 non-null  object
 9   Filling Shoe M20M13 speed              439846 non-null  object
 10  Filling Shoe M20M23 speed              439846 non-null  object
 11  Material inlet: Hopper level detection 439846 non-null  object
 12  Ejection force tablet                  439846 non-null  object
dtypes: object(13)
memory usage: 43.6+ MB
```

```
Index(['TimeStamp', 'Pre-compression height bottom',
       'Pre-compresion top dwell time', 'Pre-compression force',
       'Pre-compression displacement top sigma',
       'Main compression height bottom', 'Main compression top dwell time',
       'Main compression force', 'Compression cycle fill depth',
       'Filling Shoe M20M13 speed', 'Filling Shoe M20M23 speed',
       'Material inlet: Hopper level detection', 'Ejection force tablet'],
      dtype='object')
```

```
None
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 439844 entries, 0 to 439845
Data columns (total 13 columns):
 #   Column                                  Non-Null Count   Dtype
---  ------                                  --------------   -----
 0   TimeStamp                               439844 non-null  datetime64[ns]
 1   Pre-compression_height_bottom           439844 non-null  float64
 2   Pre-compresion_top_dwell_time           439844 non-null  float64
 3   Pre-compression_force                   439844 non-null  float64
 4   Pre-compression_displacement_top_sigma  439844 non-null  float64
 5   Main_compression_height_bottom          439844 non-null  float64
 6   Main_compression_top_dwell_time         439844 non-null  float64
 7   Main_compression_force                  439844 non-null  float64
 8   Compression_cycle_fill_depth            439844 non-null  float64
 9   Filling_Shoe_M20M13_speed               439844 non-null  float64
 10  Filling_Shoe_M20M23_speed               439844 non-null  float64
 11  Material_inlet:_Hopper_level_detection  439844 non-null  float64
 12  Ejection_force_tablet                   439844 non-null  float64
dtypes: datetime64[ns](1), float64(12)
memory usage: 47.0 MB
```

[ ]:
```
# Rename columns for clarity
temperature = temperature.rename(columns={
    'Feeder': 'Feeder_TimeStamp',
    'Unnamed: 2': 'Feeder Temperature_in_C',
    'Tablet Press': 'Tablet_Press_TimeStamp',
    'Unnamed: 5': 'Tablet_Press_Temperature in C'
})

# Drop unnecessary columns, convert timestamps to datetime format, and convert␣
 ↪numeric columns
temperature.drop(columns=['Unnamed: 1', 'Unnamed: 4'], inplace=True,␣
 ↪errors='ignore')
for col in ['Feeder Temperature_in_C', 'Tablet_Press_Temperature in C']:
    temperature[col] = pd.to_numeric(temperature[col], errors='coerce')
for col in ['Feeder_TimeStamp', 'Tablet_Press_TimeStamp']:
    temperature[col] = pd.to_datetime(temperature[col], format = 'mixed',␣
 ↪dayfirst = True, errors = 'coerce')

# Drop rows with NaN values
temperature.dropna(inplace=True)
temperature.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 492 entries, 1 to 492
Data columns (total 4 columns):
```

```
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Feeder_TimeStamp                492 non-null    datetime64[ns]
 1   Feeder Temperature_in_C         492 non-null    float64
 2   Tablet_Press_TimeStamp          492 non-null    datetime64[ns]
 3   Tablet_Press_Temperature in C   492 non-null    float64
dtypes: datetime64[ns](2), float64(2)
memory usage: 19.2 KB
```

3 Separating Materials

[ ]:
```
rm_tablet_properties_and_drum_change = pd.read_excel('/content/drive/MyDrive/
 ↪Senior Year/BA305/BA305 Project/INFORMS Datasets 2025/Contextual quality␣
 ↪data/RM Tablet Properties and Drum Change.xlsx', sheet_name='Raw Material␣
 ↪Drum change')
#display(rm_tablet_properties_and_drum_change)
# The table is formatted like so: "Material, Amount (kg), Lot, Est Refill time"␣
 ↪but the last 3 are Unnamed because the excel sheet merged the material cell␣
 ↪and put them under.
# for example the columns are ["API", "Unnamed: 1", "Unnamed: 2", "Unnamed: 3",␣
 ↪"Magnesium Stearate", ...] and the first row is ["Date/Time", "Amount (kg)",␣
 ↪"Lot", "Est Refill Time", "Date/Time"...]
# Drop the 2nd row and make a separate table for each material. The materials␣
 ↪are API, Magnesium Stearate, Lactose (Lot = 85170761), Avicel 102 (PD 1) -␣
 ↪(Lot = 71734C), and Avicel 102 (PD 4) - (Lot = 71734C)
# and make the first row into the columns
rm_tablet_properties_and_drum_change = rm_tablet_properties_and_drum_change.
 ↪drop(1)
API = rm_tablet_properties_and_drum_change.iloc[:, :4]
API.columns = API.iloc[0]
API = API.drop(0)
API = API.reset_index(drop=True)

Magnesium_Stearate = rm_tablet_properties_and_drum_change.iloc[:, 4:8]
Magnesium_Stearate.columns = Magnesium_Stearate.iloc[0]
Magnesium_Stearate = Magnesium_Stearate.drop(0)
Magnesium_Stearate = Magnesium_Stearate.reset_index(drop=True).dropna(axis = 0,␣
 ↪how = 'any')
Magnesium_Stearate['Lot'] = '31400041 ' + Magnesium_Stearate['Lot']

Lactose = rm_tablet_properties_and_drum_change.iloc[:, 8:12]
Lactose.columns = Lactose.iloc[0]
Lactose = Lactose.drop(0)
Lactose = Lactose.reset_index(drop=True).dropna(axis = 0, how = 'any')

Avicel_102_PD_1 = rm_tablet_properties_and_drum_change.iloc[:, 16:20]
```

```
Avicel_102_PD_1.columns = Avicel_102_PD_1.iloc[0]
Avicel_102_PD_1 = Avicel_102_PD_1.drop(0)
Avicel_102_PD_1 = Avicel_102_PD_1.reset_index(drop=True).dropna(axis = 0, how =␣
 ↪'any')

Avicel_102_PD_4 = rm_tablet_properties_and_drum_change.iloc[:, 20:24]
Avicel_102_PD_4.columns = Avicel_102_PD_4.iloc[0]
Avicel_102_PD_4 = Avicel_102_PD_4.drop(0)
Avicel_102_PD_4 = Avicel_102_PD_4.reset_index(drop=True).dropna(axis = 0, how =␣
 ↪'any')

print(API.columns)
print(Magnesium_Stearate.columns)
print(Lactose.columns)
print(Avicel_102_PD_1.columns)
print(Avicel_102_PD_4.columns)

display(API, Magnesium_Stearate, Lactose, Avicel_102_PD_1, Avicel_102_PD_4)
```

```
Index(['Date/Time', 'Amount (kg)', 'Lot', 'Est Refill time'], dtype='object',
name=0)
Index(['Date/Time', 'Amount (kg)', 'Lot', 'Est Refill time'], dtype='object',
name=0)
Index(['Date/Time', 'Amount (kg)', 'Drum #', 'Est Refill time'], dtype='object',
name=0)
Index(['Date/Time', 'Amount (kg)', 'Drum #', 'Est Refill time'], dtype='object',
name=0)
Index(['Date/Time', 'Amount (kg)', 'Drum #', 'Est Refill time'], dtype='object',
name=0)
```

0	Date/Time	Amount (kg)	Lot	Est Refill time
0	2018-01-12 07:25:00	8.78	00077	2018-01-12 09:30:00
1	2018-01-12 09:38:00	8.9	00077	2018-01-12 11:10:00
2	2018-01-12 11:20:00	7.42	00077	2018-01-12 13:50:00
3	2018-01-12 12:48:00	9.92	00080	2018-01-12 15:40:00
4	2018-01-12 14:55:00	11.34	00080	2018-01-12 17:00:00
..	…	…	…	…
64	2018-01-17 03:05:00	10.78	00078	2018-01-17 05:10:00
65	2018-01-17 05:00:00	11.28	00079	2018-01-17 07:05:00
66	2018-01-17 07:00:00	8.02	00079	2018-01-17 08:30:00
67	2018-01-17 08:28:00	8.02	00079	2018-01-17 10:00:00
68	2018-01-17 09:57:00	4.523	00079	END

[69 rows x 4 columns]

0	Date/Time	Amount (kg)	Lot	Est Refill time
0	2018-01-12 08:16:00	2.693	31400041 bag 1	2018-01-12 11:30:00
1	2018-01-12 11:00:00	1.768	31400041 bag 1	2018-01-12 13:30:00

2	2018-01-12 13:42:00	1.555	31400041	bag 1	2018-01-12 16:30:00
3	2018-01-12 16:30:00	0.75	31400041	bag 1	2018-01-12 19:30:00
4	2018-01-12 21:17:00	1.388	31400041	bag 1	2018-01-12 22:50:00
5	2018-01-12 23:23:00	1	31400041	bag 1	2018-01-13 01:00:00
6	2018-01-13 01:54:00	1.12	31400041	bag 1	2018-01-13 06:00:00
7	2018-01-13 05:45:00	0.9	31400041	bag 1	2018-01-13 10:00:00
8	2018-01-13 10:34:00	1	31400041	bag 1	2018-01-13 12:30:00
9	2018-01-13 00:29:00	1.08	31400041	bag 1	2018-01-13 14:30:00
10	2018-01-13 01:20:00	1.02	31400041	bag 1	2018-01-13 16:30:00
11	2018-01-13 04:00:00	0.98	31400041	bag 1	2018-01-13 18:00:00
12	2018-01-13 17:21:00	0.846	31400041	bag 1	2018-01-13 19:20:00
13	2018-01-13 18:41:00	0.8	31400041	bag 2	2018-01-13 21:00:00
14	2018-01-13 20:05:00	1.02	31400041	bag 1	2018-01-13 22:05:00
15	2018-01-13 21:46:00	1.2	31400041	bag 1	2018-01-13 22:05:00
16	2018-01-13 23:55:00	1.02	31400041	bag 2	2018-01-14 03:00:00
17	2018-01-14 02:30:00	1.3	31400041	bag 2	2018-01-14 05:00:00
18	2018-01-14 05:16:00	1.18	31400041	bag 2	2018-01-14 07:00:00
19	2018-01-14 06:46:00	1.16	31400041	bag 2	2018-01-14 09:00:00
20	2018-01-14 09:00:00	1.14	31400041	bag 2	2018-01-14 11:00:00
21	2018-01-14 11:07:00	1.02	31400041	bag 2	2018-01-14 14:00:00
22	2018-01-14 14:00:00	1.02	31400041	bag 2	2018-01-14 16:00:00
23	2018-01-14 15:26:00	0.94	31400041	bag 2	2018-01-14 17:30:00
24	2018-01-14 17:15:00	1	31400041	bag 2	2018-01-14 19:15:00
25	2018-01-14 18:30:00	1.08	31400041	bag 2	2018-01-14 20:30:00
26	2018-01-14 20:14:00	1.12	31400041	bag 2	2018-01-14 22:14:00
27	2018-01-14 22:30:00	0.86	31400041	bag 2	2018-01-15 00:00:00
28	2018-01-15 00:38:00	1	31400041	bag 2	2018-01-15 02:30:00
29	2018-01-15 02:38:00	1	31400041	bag 2	2018-01-15 04:30:00
30	2018-01-15 04:39:00	1.26	31400041	bag 2	2018-01-15 08:00:00
31	2018-01-15 08:11:00	1.14	31400041	bag 2	2018-01-15 10:10:00
32	2018-01-15 11:10:00	1.04	31400041	bag 2	2018-01-15 13:00:00
33	2018-01-15 13:34:00	0.98	31400041	bag 2	2018-01-15 17:00:00
34	2018-01-15 15:47:00	1	31400041	bag 2	2018-01-15 17:47:00
35	2018-01-15 17:20:00	0.92	31400041	bag 2	2018-01-15 19:20:00
36	2018-01-15 18:50:00	1.04	31400041	bag 2	2018-01-15 20:50:00
37	2018-01-15 21:20:00	1.04	31400041	bag 2	2018-01-15 23:20:00
38	2018-01-15 23:55:00	1.02	31400041	bag 3	2018-01-16 02:00:00
39	2018-01-16 02:10:00	1.1	31400041	bag 3	2018-01-16 04:00:00
40	2018-01-16 03:58:00	0.98	31400041	bag 3	2018-01-16 06:00:00
41	2018-01-16 06:00:00	1.08	31400041	bag 3	2018-01-16 08:00:00
42	2018-01-16 08:00:00	1	31400041	bag 3	2018-01-16 10:00:00
43	2018-01-16 10:00:00	1.24	31400041	bag 3	2018-01-16 12:00:00
44	2018-01-16 12:00:00	1.08	31400041	bag 3	2018-01-16 14:00:00
45	2018-01-16 14:00:00	1.08	31400041	bag 3	2018-01-16 16:00:00
46	2018-01-16 16:20:00	1.1	31400041	bag 3	2018-01-16 18:20:00
47	2018-01-16 18:33:00	1.08	31400041	bag 3	2018-01-16 20:33:00
48	2018-01-16 20:23:00	1.24	31400041	bag 3	2018-01-16 22:23:00
49	2018-01-16 22:09:00	1.12	31400041	bag 3	2018-01-17 00:09:00

50	2018-01-17 00:15:00	1.13	31400041 bag 4	2018-01-17 00:09:00	
51	2018-01-17 02:10:00	1.09	31400041 bag 4	2018-01-17 05:00:00	
52	2018-01-17 05:00:00	1.24	31400041 bag 4	2018-01-17 07:30:00	
53	2018-01-17 07:30:00	0.9769	31400041 bag 4	2018-01-17 09:30:00	
54	2018-01-17 09:30:00	1.0332	31400041 bag 4		END

0	Date/Time	Amount (kg)	Drum #	Est Refill time
0	2018-01-12 07:29:00	75	162	2018-01-12 13:15:00
1	2018-01-12 13:02:00	75	179	2018-01-12 13:15:00
2	2018-01-12 17:54:00	75	192	2018-01-13 01:15:00
3	2018-01-13 02:03:00	75	180	2018-01-13 07:15:00
4	2018-01-13 08:08:00	75	177	2018-01-13 14:15:00
5	2018-01-13 14:26:00	75	198	2018-01-13 19:45:00
6	2018-01-13 19:26:00	75	199	2018-01-13 23:30:00
7	2018-01-14 01:21:00	75	82	2018-01-14 06:30:00
8	2018-01-14 06:40:00	75	197	2018-01-14 11:50:00
9	2018-01-14 11:55:00	75	176	2018-01-14 16:30:00
10	2018-01-14 15:45:00	75	200	2018-01-14 19:45:00
11	2018-01-14 20:39:00	75	165	2018-01-15 00:39:00
12	2018-01-14 20:39:00	75	165	2018-01-15 00:39:00
13	2018-01-15 02:05:00	75	164	2018-01-15 07:15:00
14	2018-01-15 07:10:00	75	161	2018-01-15 12:00:00
15	2018-01-15 12:10:00	75	196	2018-01-15 19:10:00
16	2018-01-15 17:30:00	75	163	2018-01-15 21:10:00
17	2018-01-15 23:21:00	75	90	2018-01-16 04:30:00
18	2018-01-16 04:38:00	75	89	2018-01-16 09:50:00
19	2018-01-16 09:36:00	75	86	2018-01-16 02:36:00
20	2018-01-16 14:36:00	75	87	2018-01-16 19:36:00
21	2018-01-16 18:43:00	75	191	2018-01-16 22:43:00
22	2018-01-17 00:11:00	75	88	2018-01-17 05:20:00
23	2018-01-17 05:02:00	75	194	2018-01-17 10:00:00
24	2018-01-17 10:00:00	16.044	195	END

0	Date/Time	Amount (kg)	Drum #	Est Refill time
0	2018-01-12 07:35:00	50	20826	2018-01-12 11:30:00
1	2018-01-12 11:24:00	50	20869	2018-01-12 15:00:00
2	2018-01-12 14:50:00	50	20633	2018-01-12 18:50:00
3	2018-01-12 18:17:00	50	20697	2018-01-12 22:17:00
4	2018-01-13 00:19:00	50	20632	2018-01-13 04:19:00
5	2018-01-13 05:40:00	50	20593	2018-01-13 07:40:00
6	2018-01-13 08:10:00	50	20648	2018-01-13 12:40:00
7	2018-01-13 12:50:00	50	20594	2018-01-13 16:20:00
8	2018-01-13 16:15:00	50	20824	2018-01-13 19:45:00
9	2018-01-13 19:49:00	50	20827	2018-01-13 23:15:00
10	2018-01-13 23:56:00	50	20679	2018-01-14 03:30:00
11	2018-01-14 03:31:00	50	20629	2018-01-14 07:05:00
12	2018-01-14 07:03:00	50	20758	2018-01-14 10:30:00
13	2018-01-14 10:45:00	50	20143	2018-01-14 13:43:00
14	2018-01-14 13:43:00	50	20644	2018-01-14 17:00:00

15	2018-01-14 16:31:00	50	20578	2018-01-14 20:00:00
16	2018-01-14 19:58:00	50	20763	2018-01-14 23:30:00
17	2018-01-14 23:00:00	50	20828	2018-01-15 02:30:00
18	2018-01-15 02:56:00	50	20806	2018-01-15 05:40:00
19	2018-01-15 06:23:00	50	20539	2018-01-15 10:05:00
20	2018-01-15 09:42:00	50	20626	2018-01-15 13:15:00
21	2018-01-15 13:15:00	50	20623	2018-01-15 16:45:00
22	2018-01-15 16:50:00	50	20415	2018-01-15 20:20:00
23	2018-01-15 20:17:00	50	20627	2018-01-15 23:47:00
24	2018-01-16 00:36:00	50	20769	2018-01-16 04:05:00
25	2018-01-16 00:36:00	50	20769	2018-01-16 04:05:00
26	2018-01-16 04:09:00	50	20821	2018-01-16 07:45:00
27	2018-01-16 07:22:00	50	20816	2018-01-16 10:40:00
28	2018-01-16 11:00:00	50	20807	2018-01-16 14:20:00
29	2018-01-16 14:20:00	50	20606	2018-01-16 17:50:00
30	2018-01-16 17:00:00	50	20603	2018-01-16 20:30:00
31	2018-01-16 20:19:00	50	20604	2018-01-16 23:50:00
32	2018-01-17 00:00:00	50	20534	2018-01-17 03:35:00
33	2018-01-17 03:16:00	50	20809	2018-01-17 07:00:00
34	2018-01-17 06:30:00	50	20416	2018-01-17 10:00:00
35	2018-01-17 09:35:00	0	20418	END

0	Date/Time	Amount (kg)		Drum #	Est Refill time
0	2018-01-12 07:30:00	50		20649	2018-01-12 11:30:00
1	2018-01-12 11:35:00	50		20636	2018-01-12 15:00:00
2	2018-01-12 14:58:00	50		20634	2018-01-12 18:28:00
3	2018-01-12 18:13:00	50		20631	2018-01-12 21:43:00
4	2018-01-13 00:27:00	50		20591	2018-01-13 04:00:00
5	2018-01-13 05:42:00	50		20592	2018-01-13 09:15:00
6	2018-01-13 09:33:00	50		20642	2018-01-13 13:05:00
7	2018-01-13 13:15:00	50		20775	2018-01-13 16:45:00
8	2018-01-13 16:22:00	50		20825	2018-01-13 19:46:00
9	2018-01-13 19:54:00	50		20823	2018-01-13 23:00:00
10	2018-01-14 00:00:00	50		20621	2018-01-14 03:35:00
11	2018-01-14 03:53:00	50		20709	2018-01-14 07:40:00
12	2018-01-14 07:10:00	50		20673	2018-01-14 10:35:00
13	2018-01-14 11:00:00	50		20646	2018-01-14 14:00:00
14	2018-01-14 14:15:00	50		20420	2018-01-14 17:15:00
15	2018-01-14 17:00:00	50		20625	2018-01-14 20:00:00
16	2018-01-14 20:06:00	50		20655	2018-01-14 23:36:00
17	2018-01-15 00:00:00	50		20517	2018-01-15 03:40:00
18	2018-01-15 03:26:00	50		20590	2018-01-15 07:01:00
19	2018-01-15 06:42:00	50		20635	2018-01-15 10:25:00
20	2018-01-15 10:42:00	20	(LOT 71732C)	04577	2018-01-15 12:00:00
21	2018-01-15 12:00:00	20	(LOT 71732C)	04562	2018-01-15 13:30:00
22	2018-01-15 13:30:00	20	(LOT 71732C)	04570	2018-01-15 15:00:00
23	2018-01-15 14:45:00	20	(LOT 71732C)	04561	2018-01-15 16:15:00
24	2018-01-15 16:20:00	20	(LOT 71732C)	04569	2018-01-15 17:50:00

```
25  2018-01-15 18:02:00          50                20628  2018-01-15 21:32:00
26  2018-01-15 20:50:00          50                20771  2018-01-16 00:20:00
27  2018-01-16 01:00:00          50                20773  2018-01-16 04:40:00
28  2018-01-16 04:41:00          50                20774  2018-01-16 08:15:00
29  2018-01-16 08:15:00          50                20601  2018-01-16 11:45:00
30  2018-01-16 11:30:00          50                20602  2018-01-16 15:00:00
31  2018-01-16 14:40:00          50                20595  2018-01-16 18:00:00
32  2018-01-16 17:35:00          50                20605  2018-01-16 21:00:00
33  2018-01-16 20:35:00          50                20805  2018-01-16 23:55:00
34  2018-01-16 12:17:00          50                20810  2018-01-16 15:43:00
36  2018-01-17 07:00:00          50                20477  2018-01-16 22:00:00
37  2018-01-17 10:00:00      30.966                20418                  END
```

[ ]:
```
print(API.columns)
print(Magnesium_Stearate.columns)
print(Lactose.columns)
print(Avicel_102_PD_1.columns)
print(Avicel_102_PD_4.columns)
```

```
Index(['Date/Time', 'Amount (kg)', 'Lot', 'Est Refill time'], dtype='object',
name=0)
Index(['Date/Time', 'Amount (kg)', 'Lot', 'Est Refill time'], dtype='object',
name=0)
Index(['Date/Time', 'Amount (kg)', 'Drum #', 'Est Refill time'], dtype='object',
name=0)
Index(['Date/Time', 'Amount (kg)', 'Drum #', 'Est Refill time'], dtype='object',
name=0)
Index(['Date/Time', 'Amount (kg)', 'Drum #', 'Est Refill time'], dtype='object',
name=0)
```

[ ]:
```
# Remove spaces from columns and convert them to numeric and datetime
for df in [API, Magnesium_Stearate, Lactose, Avicel_102_PD_1, Avicel_102_PD_4]:
    df.columns = df.columns.str.replace(' ', '_')
    for col in df.columns:
        if col != 'Date/Time' and col != 'Est_Refill_time':
            if col == 'Lot':
                continue
            else:
                df[col] = pd.to_numeric(df[col], errors='coerce')


    df['Date/Time'] = pd.to_datetime(df['Date/Time'], format='mixed',␣
 ↪dayfirst=True, errors='coerce')
    df['Est_Refill_time'] = pd.to_datetime(df['Est_Refill_time'],␣
 ↪format='mixed', dayfirst=True, errors='coerce')
# Add lot numbers for Lactose and Avicel 102
Lactose['Lot'] = '85170761'
Avicel_102_PD_1['Lot'] = '71734C'
```

```
Avicel_102_PD_4['Lot'] = '71734C'
# Find rows where drum numbers contain "(LOT" and extract lot numbers
for df in [Avicel_102_PD_1, Avicel_102_PD_4]:
    if 'Drum #' in df.columns:
        # Convert Drum # column to string type if not already
        df['Drum #'] = df['Drum #'].fillna('').astype(str)

        # Extract lot numbers and clean up Drum # column
        for idx, drum in enumerate(df['Drum #']):
            if '(LOT' in drum:
                # Extract lot number
                lot_start = drum.find('(LOT') + 4
                lot_end = drum.find(')')
                lot_number = drum[lot_start:lot_end].strip()

                # Update Lot column
                df.loc[idx, 'Lot'] = lot_number

                # Clean up Drum # by removing the LOT text
                clean_drum = drum[:drum.find('(LOT')].strip() + ' ' + drum[drum.
 ↪find(')')+1:].strip()
                df.loc[idx, 'Drum #'] = clean_drum.strip()

# Use LiW feeders and link the timestamps to make a designated dataset for each␣
 ↪material
API_liw_feeders = pd.merge(API, liw_feeders1, left_on='Date/Time',␣
 ↪right_on='TimeStamp', how='inner')
Magnesium_Stearate_liw_feeders = pd.merge(Magnesium_Stearate, liw_feeders1,␣
 ↪left_on='Date/Time', right_on='TimeStamp', how='inner')
Lactose_liw_feeders = pd.merge(Lactose, liw_feeders1, left_on='Date/Time',␣
 ↪right_on='TimeStamp', how='inner')
Avicel_102_PD_1_liw_feeders = pd.merge(Avicel_102_PD_1, liw_feeders1,␣
 ↪left_on='Date/Time', right_on='TimeStamp', how='inner')
# for Avicel_102_PD_4_liw_feeders, the %_PD4 column is in the liw_feeders2␣
 ↪dataframe, so we need to take that as well as the data from liw_feeders1
liw_feeders1['%_PD4'] = liw_feeders2['%_PD4']
Avicel_102_PD_4_liw_feeders = pd.merge(Avicel_102_PD_4, liw_feeders1,␣
 ↪left_on='Date/Time', right_on='TimeStamp', how='inner')
```

[ ]:
```
display(API_liw_feeders, Magnesium_Stearate_liw_feeders, Lactose_liw_feeders,␣
 ↪Avicel_102_PD_1_liw_feeders, Avicel_102_PD_4_liw_feeders)
```

```
             Date/Time  Amount_(kg)    Lot     Est_Refill_time  \
0  2018-01-13 00:50:00       12.020  00071 2018-01-13 03:10:00
1  2018-01-13 00:50:00       12.020  00071 2018-01-13 03:10:00
2  2018-01-13 00:50:00       12.020  00071 2018-01-13 03:10:00
3  2018-01-13 00:50:00       12.020  00071 2018-01-13 03:10:00
```

```
4    2018-01-13 00:50:00       12.020  00071 2018-01-13 03:10:00
…                    …          …    …                    …
3595 2018-01-17 09:57:00        4.523  00079                NaT
3596 2018-01-17 09:57:00        4.523  00079                NaT
3597 2018-01-17 09:57:00        4.523  00079                NaT
3598 2018-01-17 09:57:00        4.523  00079                NaT
3599 2018-01-17 09:57:00        4.523  00079                NaT

                TimeStamp  Feed_Factor_PD1  Feed_Factor_PD2  Feed_Factor_PD3  \
0    2018-01-13 00:50:00           1.2967           1.8658           1.0562
1    2018-01-13 00:50:00           1.2938           1.8676           1.0682
2    2018-01-13 00:50:00           1.2870           1.8651           1.0834
3    2018-01-13 00:50:00           1.2923           1.8668           1.0847
4    2018-01-13 00:50:00           1.2984           1.8660           1.0669
…                    …                …                …                …
3595 2018-01-17 09:57:00           1.2502           1.7378           1.1504
3596 2018-01-17 09:57:00           1.2460           1.7391           1.1400
3597 2018-01-17 09:57:00           1.2402           1.7345           1.1359
3598 2018-01-17 09:57:00           1.2359           1.7376           1.1332
3599 2018-01-17 09:57:00           1.2340           1.7370           1.1323

      Feed_Factor_PD4  Feed_Factor_PD5  …  Screw_RPM_PD7  Massflow_PD_1  \
0              1.3070           1.1158  …        10.1124        14.4241
1              1.3068           1.1159  …        10.1266        14.2649
2              1.2989           1.1168  …        10.1266        14.1464
3              1.3013           1.1176  …        10.1337        14.2608
4              1.3010           1.1180  …        10.1621        14.4481
…                 …                …  …              …              …
3595           1.3245           1.1312  …        11.3756        15.5661
3596           1.3308           1.1311  …        11.3543        15.5661
3597           1.3215           1.1312  …        11.3543        15.5851
3598           1.3259           1.1311  …        11.3472        15.6875
3599           1.3282           1.1309  …        11.3756        15.7052

      Massflow_PD_2  Massflow_PD_3  Massflow_PD_4  Massflow_PD_5  \
0           14.4986         5.1281        14.2451         1.5034
1           14.5288         5.0449        14.3058         1.5041
2           14.4837         5.1318        14.1420         1.5052
3           14.5196         5.0889        14.2424         1.5067
4           14.5111         5.0146        14.2679         1.5086
…              …               …              …              …
3595        15.9512         5.4234        15.7580         1.6452
3596        15.9512         5.4234        15.5499         1.6438
3597        15.8976         5.3324        15.7777         1.6437
3598        15.9126         5.3388        15.4729         1.6446
3599        16.0331         5.3470        15.5886         1.6446

      Massflow_PD_7    %_PD1    %_PD2    %_PD3
```

```
0           0.49850  28.6942  29.1040  10.2423
1           0.49804  28.6942  29.1040  10.2423
2           0.49784  28.5231  28.9749  10.0989
3           0.49746  28.6330  28.9012  10.2683
4           0.49713  28.0506  29.0676  10.0616
…             …      …      …       …
3595        0.54884  28.3402  29.0425  10.0913
3596        0.54876  28.4671  29.0934   9.8456
3597        0.54877  28.4984  28.8994   9.6901
3598        0.54877  28.4633  28.8823   9.7139
3599        0.54876  28.5296  29.0168   9.7236

[3600 rows x 26 columns]
```

```
                Date/Time  Amount_(kg)             Lot     Est_Refill_time  \
0     2018-01-13 01:54:00       1.1200  31400041 bag 1 2018-01-13 06:00:00
1     2018-01-13 01:54:00       1.1200  31400041 bag 1 2018-01-13 06:00:00
2     2018-01-13 01:54:00       1.1200  31400041 bag 1 2018-01-13 06:00:00
3     2018-01-13 01:54:00       1.1200  31400041 bag 1 2018-01-13 06:00:00
4     2018-01-13 01:54:00       1.1200  31400041 bag 1 2018-01-13 06:00:00
…                     …           …             …                   …
2935 2018-01-17 09:30:00       1.0332  31400041 bag 4                 NaT
2936 2018-01-17 09:30:00       1.0332  31400041 bag 4                 NaT
2937 2018-01-17 09:30:00       1.0332  31400041 bag 4                 NaT
2938 2018-01-17 09:30:00       1.0332  31400041 bag 4                 NaT
2939 2018-01-17 09:30:00       1.0332  31400041 bag 4                 NaT

                TimeStamp  Feed_Factor_PD1  Feed_Factor_PD2  Feed_Factor_PD3  \
0     2018-01-13 01:54:00           1.2773           1.8557           1.1931
1     2018-01-13 01:54:00           1.2793           1.8551           1.1939
2     2018-01-13 01:54:00           1.2784           1.8574           1.2050
3     2018-01-13 01:54:00           1.2780           1.8573           1.2108
4     2018-01-13 01:54:00           1.2776           1.8582           1.2095
…                     …                …                …                …
2935 2018-01-17 09:30:00           1.2861           1.7289           1.2296
2936 2018-01-17 09:30:00           1.2788           1.7276           1.2252
2937 2018-01-17 09:30:00           1.2808           1.7329           1.2246
2938 2018-01-17 09:30:00           1.2811           1.7306           1.2243
2939 2018-01-17 09:30:00           1.2876           1.7357           1.2235

      Feed_Factor_PD4  Feed_Factor_PD5  …  Screw_RPM_PD7  Massflow_PD_1  \
0              1.3251           1.1155  …        10.0344        14.1572
1              1.3253           1.1157  …        10.0344        14.2150
2              1.3254           1.1159  …        10.0344        14.2174
3              1.3238           1.1158  …        10.0415        14.2206
4              1.3242           1.1158  …        10.0273        14.2410
…                 …                …  …              …              …
2935           1.2930           1.1194  …        11.2550        15.7932
```

```
2936           1.2865           1.1195  …        11.2266        15.5427
2937           1.2807           1.1197  …        11.2621        15.6628
2938           1.2859           1.1198  …        11.2266        15.5425
2939           1.2837           1.1200  …        11.2195        15.7694

      Massflow_PD_2  Massflow_PD_3  Massflow_PD_4  Massflow_PD_5  \
0           14.4736         4.9156        14.1591         1.4987
1           14.4649         4.9256        14.1721         1.4995
2           14.5121         5.0560        14.1864         1.5005
3           14.5009         5.1249        14.1361         1.5012
4           14.5272         5.1114        14.1698         1.5014
…             …             …             …              …
2935        16.0145         5.4868        15.7154         1.6488
2936        15.9001         5.4380        15.7322         1.6491
2937        15.9013         5.4403        15.5284         1.6498
2938        15.9429         5.4717        15.5317         1.6509
2939        16.0344         5.5066        15.8437         1.6511

      Massflow_PD_7    %_PD1    %_PD2    %_PD3
0           0.51195  28.4437  29.0456   9.8587
1           0.51176  28.5622  28.9479   9.8394
2           0.51149  28.4243  29.1129  10.1119
3           0.51133  28.3809  28.9715  10.2544
4           0.51128  28.4132  28.9297  10.1964
…             …      …      …       …
2935        0.54866  28.6622  29.0586  10.0109
2936        0.54844  28.3938  29.0605   9.8847
2937        0.54806  28.4747  28.9415   9.8877
2938        0.54741  28.4322  28.9710   9.9674
2939        0.54636  28.4537  29.0465  10.0191

[2940 rows x 26 columns]

               Date/Time  Amount_(kg)  Drum_#     Est_Refill_time       Lot  \
0    2018-01-13 02:03:00       75.000     180 2018-01-13 07:15:00  85170761
1    2018-01-13 02:03:00       75.000     180 2018-01-13 07:15:00  85170761
2    2018-01-13 02:03:00       75.000     180 2018-01-13 07:15:00  85170761
3    2018-01-13 02:03:00       75.000     180 2018-01-13 07:15:00  85170761
4    2018-01-13 02:03:00       75.000     180 2018-01-13 07:15:00  85170761
…                  …          …     …                   …       …
1314 2018-01-17 10:00:00       16.044     195                 NaT  85170761
1315 2018-01-17 10:00:00       16.044     195                 NaT  85170761
1316 2018-01-17 10:00:00       16.044     195                 NaT  85170761
1317 2018-01-17 10:00:00       16.044     195                 NaT  85170761
1318 2018-01-17 10:00:00       16.044     195                 NaT  85170761

               TimeStamp  Feed_Factor_PD1  Feed_Factor_PD2  Feed_Factor_PD3  \
0    2018-01-13 02:03:00           1.3098           1.8554           1.1216
```

```
1    2018-01-13 02:03:00           1.3141           1.8530           1.1209
2    2018-01-13 02:03:00           1.3118           1.8557           1.1261
3    2018-01-13 02:03:00           1.3151           1.8563           1.1625
4    2018-01-13 02:03:00           1.3117           1.8553           1.2009
…                    …              …                …                …
1314 2018-01-17 10:00:00           1.3202           1.7403           1.1817
1315 2018-01-17 10:00:00           1.3165           1.7377           1.2002
1316 2018-01-17 10:00:00           1.3157           1.7369           1.2335
1317 2018-01-17 10:00:00           1.3163           1.7371           1.2398
1318 2018-01-17 10:00:00           1.3164           1.7377           1.2355

      Feed_Factor_PD4  …  Screw_RPM_PD7  Massflow_PD_1  Massflow_PD_2  \
0              1.3447  …         9.7931        14.1601        14.4989
1              1.3403  …         9.8144        14.2645        14.4545
2              1.3352  …         9.8215        14.2692        14.4495
3              1.3428  …         9.8286        14.2487        14.5316
4              1.3462  …         9.8215        14.1667        14.5092
…                 …  …              …              …              …
1314           1.2986  …        11.4821        15.7703        16.0054
1315           1.2967  …        11.4466        15.6666        15.9621
1316           1.3027  …        11.4608        15.6461        15.8924
1317           1.2945  …        11.4466        15.6768        15.9306
1318           1.2947  …        11.4679        15.6621        15.9469

      Massflow_PD_3  Massflow_PD_4  Massflow_PD_5  Massflow_PD_7    %_PD1  \
0            4.6829        14.3371         1.5082        0.49670  28.5399
1            4.6540        14.2621         1.5078        0.49674  28.4745
2            4.6549        14.1699         1.5069        0.49678  28.5774
3            4.8437        14.2608         1.5028        0.49695  28.4507
4            5.1084        14.3239         1.4996        0.49706  28.4732
…               …              …              …              …       …
1314         5.7128        15.5357         1.6555        0.54442  28.4320
1315         5.7669        15.5294         1.6557        0.54434  28.5567
1316         5.9634        15.7891         1.6549        0.54432  28.6206
1317         6.0354        15.5792         1.6533        0.54437  28.6206
1318         5.9872        15.6109         1.6525        0.54446  28.3905

        %_PD2    %_PD3
0     28.9488   9.4969
1     28.9653   9.3096
2     29.0503   9.3160
3     28.9845   9.5900
4     28.9841  10.1178
…       …      …
1314  29.0069  10.3996
1315  29.0701  10.4875
1316  28.9559  10.8576
1317  28.9559  10.8576
```

```
1318  28.9286  10.9682

[1319 rows x 27 columns]
```

	Date/Time	Amount_(kg)	Drum_#	Est_Refill_time	Lot	\
0	2018-01-13 00:19:00	50	20632	2018-01-13 04:19:00	71734C	
1	2018-01-13 00:19:00	50	20632	2018-01-13 04:19:00	71734C	
2	2018-01-13 00:19:00	50	20632	2018-01-13 04:19:00	71734C	
3	2018-01-13 00:19:00	50	20632	2018-01-13 04:19:00	71734C	
4	2018-01-13 00:19:00	50	20632	2018-01-13 04:19:00	71734C	
…	…	…	…	…	…	
1913	2018-01-17 09:35:00	0	20418	NaT	71734C	
1914	2018-01-17 09:35:00	0	20418	NaT	71734C	
1915	2018-01-17 09:35:00	0	20418	NaT	71734C	
1916	2018-01-17 09:35:00	0	20418	NaT	71734C	
1917	2018-01-17 09:35:00	0	20418	NaT	71734C	

	TimeStamp	Feed_Factor_PD1	Feed_Factor_PD2	Feed_Factor_PD3	\
0	2018-01-13 00:19:00	1.2911	1.8634	1.1438	
1	2018-01-13 00:19:00	1.2911	1.8634	1.1438	
2	2018-01-13 00:19:00	1.2911	1.8634	1.1438	
3	2018-01-13 00:19:00	1.2911	1.8634	1.1438	
4	2018-01-13 00:19:00	1.2911	1.8634	1.1438	
…	…	…	…	…	
1913	2018-01-17 09:35:00	1.2679	1.7321	1.2835	
1914	2018-01-17 09:35:00	1.2693	1.7322	1.2806	
1915	2018-01-17 09:35:00	1.2704	1.7326	1.2836	
1916	2018-01-17 09:35:00	1.2667	1.7284	1.2842	
1917	2018-01-17 09:35:00	1.2643	1.7326	1.2823	

	Feed_Factor_PD4	…	Screw_RPM_PD7	Massflow_PD_1	Massflow_PD_2	\
0	1.2803	…	0.0000	0.0000	0.0000	
1	1.2803	…	0.0000	0.0000	0.0000	
2	1.2803	…	0.0000	0.0000	0.0000	
3	1.2803	…	0.0000	0.0000	0.0000	
4	1.2803	…	0.0000	0.0000	0.0000	
…	…	…	…	…	…	
1913	1.3159	…	11.4963	15.5881	15.9613	
1914	1.3292	…	11.4608	15.6096	16.0007	
1915	1.3393	…	11.5034	15.6391	15.9602	
1916	1.3465	…	11.4963	15.5755	15.8581	
1917	1.3361	…	11.4466	15.5844	15.9663	

	Massflow_PD_3	Massflow_PD_4	Massflow_PD_5	Massflow_PD_7	%_PD1	\
0	0.0000	0.0000	0.0000	0.00000	0.0000	
1	0.0000	0.0000	0.0000	0.00000	0.0000	
2	0.0000	0.0000	0.0000	0.00000	0.0000	
3	0.0000	0.0000	0.0000	0.00000	0.0000	

```
4            0.0000         0.0000         0.0000        0.00000   0.0000
…             …             …              …             …         …
1913         5.6570        16.2316         1.6386        0.54611  28.3614
1914         5.6382        16.1761         1.6378        0.54634  28.3614
1915         5.6562        15.9710         1.6372        0.54657  28.6602
1916         5.6364        15.9813         1.6370        0.54700  28.5402
1917         5.6037        15.6952         1.6370        0.54747  28.2981

        %_PD2    %_PD3
0      0.0000   0.0000
1      0.0000   0.0000
2      0.0000   0.0000
3      0.0000   0.0000
4      0.0000   0.0000
…       …       …
1913  29.0029  10.2985
1914  29.0029  10.2985
1915  28.9507  10.2649
1916  29.0157  10.2816
1917  29.0213  10.1906

[1918 rows x 27 columns]

                Date/Time  Amount_(kg)   Drum_#     Est_Refill_time     Lot  \
0    2018-01-13 00:27:00       50.000  20591.0 2018-01-13 04:00:00  71734C
1    2018-01-13 00:27:00       50.000  20591.0 2018-01-13 04:00:00  71734C
2    2018-01-13 00:27:00       50.000  20591.0 2018-01-13 04:00:00  71734C
3    2018-01-13 00:27:00       50.000  20591.0 2018-01-13 04:00:00  71734C
4    2018-01-13 00:27:00       50.000  20591.0 2018-01-13 04:00:00  71734C
…                    …          …       …                  …      …
1975 2018-01-17 10:00:00       30.966  20418.0                NaT  71734C
1976 2018-01-17 10:00:00       30.966  20418.0                NaT  71734C
1977 2018-01-17 10:00:00       30.966  20418.0                NaT  71734C
1978 2018-01-17 10:00:00       30.966  20418.0                NaT  71734C
1979 2018-01-17 10:00:00       30.966  20418.0                NaT  71734C

                TimeStamp  Feed_Factor_PD1  Feed_Factor_PD2  Feed_Factor_PD3  \
0    2018-01-13 00:27:00           1.2977           1.8655           1.1343
1    2018-01-13 00:27:00           1.2926           1.8669           1.1336
2    2018-01-13 00:27:00           1.2940           1.8655           1.1382
3    2018-01-13 00:27:00           1.2958           1.8675           1.1387
4    2018-01-13 00:27:00           1.2947           1.8653           1.1424
…                    …              …                …                …
1975 2018-01-17 10:00:00           1.3202           1.7403           1.1817
1976 2018-01-17 10:00:00           1.3165           1.7377           1.2002
1977 2018-01-17 10:00:00           1.3157           1.7369           1.2335
1978 2018-01-17 10:00:00           1.3163           1.7371           1.2398
1979 2018-01-17 10:00:00           1.3164           1.7377           1.2355
```

```
      Feed_Factor_PD4  …  Massflow_PD_1  Massflow_PD_2  Massflow_PD_3  \
0              1.2955  …        14.3161        14.4716         4.8798
1              1.2943  …        14.1705        14.5140         4.9660
2              1.2948  …        14.2176        14.4807         5.0196
3              1.2936  …        14.2705        14.5193         5.0906
4              1.2948  …        14.2798        14.4807         5.1168
…                 …  …              …              …              …
1975           1.2986  …        15.7703        16.0054         5.7128
1976           1.2967  …        15.6666        15.9621         5.7669
1977           1.3027  …        15.6461        15.8924         5.9634
1978           1.2945  …        15.6768        15.9306         6.0354
1979           1.2947  …        15.6621        15.9469         5.9872

      Massflow_PD_4  Massflow_PD_5  Massflow_PD_7    %_PD1    %_PD2    %_PD3  \
0           14.2090         1.4965        0.51186  28.3386  28.9699   9.3940
1           14.1629         1.4956        0.51204  28.5135  28.9123   9.8094
2           14.1767         1.4955        0.51242  28.5338  29.1115   9.9493
3           14.1830         1.4956        0.51284  28.4699  29.0417  10.1186
4           14.2340         1.4962        0.51330  28.4663  28.8589  10.2077
…              …              …              …       …        …        …
1975        15.5357         1.6555        0.54442  28.4320  29.0069  10.3996
1976        15.5294         1.6557        0.54434  28.5567  29.0701  10.4875
1977        15.7891         1.6549        0.54432  28.6206  28.9559  10.8576
1978        15.5792         1.6533        0.54437  28.6206  28.9559  10.8576
1979        15.6109         1.6525        0.54446  28.3905  28.9286  10.9682

       %_PD4
0     28.3195
1     28.6944
2     28.3228
3     28.3068
4     28.6471
…       …
1975  28.5373
1976  28.1870
1977  28.7247
1978  28.7247
1979  28.4576

[1980 rows x 28 columns]
```

[ ]:
```
materials = ['API', 'Magnesium_Stearate', 'Lactose', 'Avicel_102_PD_1',␣
 ↪'Avicel_102_PD_4']
for df in [API_liw_feeders, Magnesium_Stearate_liw_feeders,␣
 ↪Lactose_liw_feeders, Avicel_102_PD_1_liw_feeders,␣
 ↪Avicel_102_PD_4_liw_feeders]:
```

```
    material_name = materials.pop(0)
    df.rename(columns={'Lot': 'Lot_' + material_name}, inplace=True)

API_liw_feeders['prev_Lot_API'] = API_liw_feeders['Lot_API'].shift(1)
API_liw_feeders['inter_lot_change'] = API_liw_feeders['Lot_API'] !=␣
 ↪API_liw_feeders['Lot_API']
print(API_liw_feeders[['TimeStamp', 'Lot_API', 'prev_Lot_API',␣
 ↪'inter_lot_change']].head())

Magnesium_Stearate_liw_feeders['prev_Lot_Magnesium_Stearate'] =␣
 ↪Magnesium_Stearate_liw_feeders['Lot_Magnesium_Stearate'].shift(1)
Magnesium_Stearate_liw_feeders['inter_lot_change'] =␣
 ↪Magnesium_Stearate_liw_feeders['Lot_Magnesium_Stearate'] !=␣
 ↪Magnesium_Stearate_liw_feeders['Lot_Magnesium_Stearate']
print(Magnesium_Stearate_liw_feeders[['TimeStamp', 'Lot_Magnesium_Stearate',␣
 ↪'prev_Lot_Magnesium_Stearate', 'inter_lot_change']].head())

Lactose_liw_feeders['prev_Lot_Lactose'] = Lactose_liw_feeders['Lot_Lactose'].
 ↪shift(1)
Lactose_liw_feeders['inter_lot_change'] = Lactose_liw_feeders['Lot_Lactose'] !=␣
 ↪Lactose_liw_feeders['Lot_Lactose']
print(Lactose_liw_feeders[['TimeStamp', 'Lot_Lactose', 'prev_Lot_Lactose',␣
 ↪'inter_lot_change']].head())

Avicel_102_PD_1_liw_feeders['prev_Lot_Avicel_102_PD_1'] =␣
 ↪Avicel_102_PD_1_liw_feeders['Lot_Avicel_102_PD_1'].shift(1)
Avicel_102_PD_1_liw_feeders['inter_lot_change'] =␣
 ↪Avicel_102_PD_1_liw_feeders['Lot_Avicel_102_PD_1'] !=␣
 ↪Avicel_102_PD_1_liw_feeders['Lot_Avicel_102_PD_1']
print(Avicel_102_PD_1_liw_feeders[['TimeStamp', 'Lot_Avicel_102_PD_1',␣
 ↪'prev_Lot_Avicel_102_PD_1', 'inter_lot_change']].head())

Avicel_102_PD_4_liw_feeders['prev_Lot_Avicel_102_PD_4'] =␣
 ↪Avicel_102_PD_4_liw_feeders['Lot_Avicel_102_PD_4'].shift(1)
Avicel_102_PD_4_liw_feeders['inter_lot_change'] =␣
 ↪Avicel_102_PD_4_liw_feeders['Lot_Avicel_102_PD_4'] !=␣
 ↪Avicel_102_PD_4_liw_feeders['Lot_Avicel_102_PD_4']
print(Avicel_102_PD_4_liw_feeders[['TimeStamp', 'Lot_Avicel_102_PD_4',␣
 ↪'prev_Lot_Avicel_102_PD_4', 'inter_lot_change']].head())
```

```
            TimeStamp Lot_API prev_Lot_API  inter_lot_change
0 2018-01-13 00:50:00   00071         None             False
1 2018-01-13 00:50:00   00071        00071             False
2 2018-01-13 00:50:00   00071        00071             False
3 2018-01-13 00:50:00   00071        00071             False
4 2018-01-13 00:50:00   00071        00071             False
            TimeStamp Lot_Magnesium_Stearate prev_Lot_Magnesium_Stearate  \
```

```
0 2018-01-13 01:54:00         31400041 bag 1                        None
1 2018-01-13 01:54:00         31400041 bag 1              31400041 bag 1
2 2018-01-13 01:54:00         31400041 bag 1              31400041 bag 1
3 2018-01-13 01:54:00         31400041 bag 1              31400041 bag 1
4 2018-01-13 01:54:00         31400041 bag 1              31400041 bag 1

   inter_lot_change
0             False
1             False
2             False
3             False
4             False
            TimeStamp Lot_Lactose prev_Lot_Lactose  inter_lot_change
0 2018-01-13 02:03:00    85170761             None             False
1 2018-01-13 02:03:00    85170761         85170761             False
2 2018-01-13 02:03:00    85170761         85170761             False
3 2018-01-13 02:03:00    85170761         85170761             False
4 2018-01-13 02:03:00    85170761         85170761             False
            TimeStamp Lot_Avicel_102_PD_1 prev_Lot_Avicel_102_PD_1  \
0 2018-01-13 00:19:00              71734C                     None
1 2018-01-13 00:19:00              71734C                   71734C
2 2018-01-13 00:19:00              71734C                   71734C
3 2018-01-13 00:19:00              71734C                   71734C
4 2018-01-13 00:19:00              71734C                   71734C

   inter_lot_change
0             False
1             False
2             False
3             False
4             False
            TimeStamp Lot_Avicel_102_PD_4 prev_Lot_Avicel_102_PD_4  \
0 2018-01-13 00:27:00              71734C                     None
1 2018-01-13 00:27:00              71734C                   71734C
2 2018-01-13 00:27:00              71734C                   71734C
3 2018-01-13 00:27:00              71734C                   71734C
4 2018-01-13 00:27:00              71734C                   71734C

   inter_lot_change
0             False
1             False
2             False
3             False
4             False
```

#Data Transformation and Flagging Disturbances

[ ]:
```
# Define target compositions and allowed relative deviation limits (in percent)
feeder_targets = {
    "Feed_Factor_PD1": 28.50,
    "Feed_Factor_PD2": 29.00,
    "Feed_Factor_PD3": 10.00,
    "Feed_Factor_PD4": 28.50,
    "Feed_Factor_PD5": 3.00,
    "Feed_Factor_PD7": 1.00
}
# Allowed relative deviation, expressed as a percent of the target value.
allowed_limits = {
    "Feed_Factor_PD1": 10.00,
    "Feed_Factor_PD2": 10.00,
    "Feed_Factor_PD3": 15.00,
    "Feed_Factor_PD4": 10.00,
    "Feed_Factor_PD5": 5.00,
    "Feed_Factor_PD7": 5.00
}

# For the primary LiW feeder DataFrame (liw_feeders1), compute deviation and␣
 ↪create flags.
for key in feeder_targets:
    target = feeder_targets[key]
    allowed = allowed_limits[key]
    # Calculate relative deviation (% difference from target)
    liw_feeders1[f'{key}_deviation'] = abs(liw_feeders1[key] - target) / target␣
 ↪* 100
    # Flag as True if deviation exceeds the allowed limit.
    liw_feeders1[f'{key}_flag'] = liw_feeders1[f'{key}_deviation'] > allowed

# Create an overall flag (if any one PD is out-of-spec)
flag_columns = [f'{k}_flag' for k in feeder_targets.keys()]
liw_feeders1['overall_flag'] = liw_feeders1[flag_columns].any(axis=1)

# For blender data, create an OOS alarm flag.
# (Here we assume any non-zero OOS reading signals a disturbance; adjust the␣
 ↪threshold as needed.)
blenders['OOS_flag'] = ((blenders['OOS_Concentration_at_Blender_1_inlet'] > 0) |
                        (blenders['OOS_Concentration_at_Blender_2_inlet'] > 0))
```

[ ]:
```
# Ensure both DataFrames are sorted by their timestamp
liw_feeders1_sorted = liw_feeders1.sort_values(by='TimeStamp')
blenders_sorted = blenders.sort_values(by='TimeStamp')
# Merge the feeder events with the nearest blender event within 1 minute.
merged_feeder_blender = pd.merge_asof(
    liw_feeders1_sorted,
    blenders_sorted,
```

```
    left_on='TimeStamp',
    right_on='TimeStamp',
    tolerance=pd.Timedelta('5min'),
    direction='nearest',
    suffixes=('_feeder', '_blender')
)

# Now, you can inspect how many feeder events have an associated blender␣
 ↪disturbance:
print("Total feeder events:", len(liw_feeders1))
print("Merged events (within 1 minute):", len(merged_feeder_blender))
```

```
Total feeder events: 439844
Merged events (within 1 minute): 439844
```

[ ]:
```
tablet_press = tablet_press.sort_values(by='TimeStamp')
# Merge the tablet press events with the nearest feeder event within a minute.
merged_all = pd.merge_asof(
    merged_feeder_blender,
    tablet_press,
    left_on='TimeStamp',
    right_on='TimeStamp',
    tolerance=pd.Timedelta('1min'),
    direction='nearest',
    suffixes=('', '_tablet')
)

# Now, merged_all contains feeder, blender, and tablet press data all aligned␣
 ↪on timestamp.
print("Merged dataset shape:", merged_all.shape)
```

```
Merged dataset shape: (439844, 55)
```

[ ]:
```
for netweightcol in ['Net_Weight_PD1', 'Net_Weight_PD2', 'Net_Weight_PD3',␣
 ↪'Net_Weight_PD4', 'Net_Weight_PD5', 'Net_Weight_PD7']:
    merged_all[netweightcol] = liw_feeders2[netweightcol]
```

[ ]:
```
# For each DataFrame, add a column that identifies the material.
API['Material'] = 'API'
Magnesium_Stearate['Material'] = 'Magnesium_Stearate'
Lactose['Material'] = 'Lactose'
Avicel_102_PD_1['Material'] = 'Avicel_102_PD_1'
Avicel_102_PD_4['Material'] = 'Avicel_102_PD_4'

# If you don't need the 'Date/Time' column for further merging, drop it.
for df in [API, Magnesium_Stearate, Lactose, Avicel_102_PD_1, Avicel_102_PD_4]:
    df.rename(columns={'Date/Time': 'Material_Time'}, inplace=True)
```

```
# Now, concatenate (stack) them into one long-form DataFrame.
material_long = pd.concat([API, Magnesium_Stearate, Lactose, Avicel_102_PD_1,␣
 ↪Avicel_102_PD_4],
                          ignore_index=True)

# Reorder columns so material comes first
cols = material_long.columns.tolist()
cols.insert(0, cols.pop(cols.index('Material')))
material_long = material_long[cols]

# Check the result
print("Long-form material data shape:", material_long.shape)
print(material_long.head())
```

```
Long-form material data shape: (222, 6)
0 Material       Material_Time  Amount_(kg)    Lot     Est_Refill_time  Drum_#
0      API 2018-01-12 07:25:00         8.78  00077 2018-01-12 09:30:00     NaN
1      API 2018-01-12 09:38:00         8.90  00077 2018-01-12 11:10:00     NaN
2      API 2018-01-12 11:20:00         7.42  00077 2018-01-12 13:50:00     NaN
3      API 2018-01-12 12:48:00         9.92  00080 2018-01-12 15:40:00     NaN
4      API 2018-01-12 14:55:00        11.34  00080 2018-01-12 17:00:00     NaN
```

[ ]:
```
# merge with material dfs
materials = ['API', 'Magnesium_Stearate', 'Lactose', 'Avicel_102_PD_1',␣
 ↪'Avicel_102_PD_4']
material_long.sort_values(by='Material_Time', inplace=True)
merged_all.sort_values(by='TimeStamp', inplace=True)
merged_all = pd.merge_asof(merged_all, material_long, left_on='TimeStamp',␣
 ↪right_on='Material_Time', tolerance=pd.Timedelta('1min'),␣
 ↪direction='nearest')
```

[ ]:
```
merged_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 439844 entries, 0 to 439843
Data columns (total 67 columns):
 #   Column                                  Non-Null Count   Dtype
---  ------                                  --------------   -----
 0   TimeStamp                               439844 non-null  datetime64[ns]
 1   Feed_Factor_PD1                         439844 non-null  float64
 2   Feed_Factor_PD2                         439844 non-null  float64
 3   Feed_Factor_PD3                         439844 non-null  float64
 4   Feed_Factor_PD4                         439844 non-null  float64
 5   Feed_Factor_PD5                         439844 non-null  float64
 6   Feed_Factor_PD7                         439844 non-null  float64
 7   Screw_RPM_PD1                           439844 non-null  float64
 8   Screw_RPM_PD2                           439844 non-null  float64
 9   Screw_RPM_PD3                           439844 non-null  float64
```

```
10  Screw_RPM_PD4                          439844 non-null  float64
11  Screw_RPM_PD5                          439844 non-null  float64
12  Screw_RPM_PD7                          439844 non-null  float64
13  Massflow_PD_1                          439844 non-null  float64
14  Massflow_PD_2                          439844 non-null  float64
15  Massflow_PD_3                          439844 non-null  float64
16  Massflow_PD_4                          439844 non-null  float64
17  Massflow_PD_5                          439844 non-null  float64
18  Massflow_PD_7                          439844 non-null  float64
19  %_PD1                                  439844 non-null  float64
20  %_PD2                                  439844 non-null  float64
21  %_PD3                                  439844 non-null  float64
22  %_PD4                                  439844 non-null  float64
23  Feed_Factor_PD1_deviation              439844 non-null  float64
24  Feed_Factor_PD1_flag                   439844 non-null  bool
25  Feed_Factor_PD2_deviation              439844 non-null  float64
26  Feed_Factor_PD2_flag                   439844 non-null  bool
27  Feed_Factor_PD3_deviation              439844 non-null  float64
28  Feed_Factor_PD3_flag                   439844 non-null  bool
29  Feed_Factor_PD4_deviation              439844 non-null  float64
30  Feed_Factor_PD4_flag                   439844 non-null  bool
31  Feed_Factor_PD5_deviation              439844 non-null  float64
32  Feed_Factor_PD5_flag                   439844 non-null  bool
33  Feed_Factor_PD7_deviation              439844 non-null  float64
34  Feed_Factor_PD7_flag                   439844 non-null  bool
35  overall_flag                           439844 non-null  bool
36  Massflow_Blender_1                     439844 non-null  float64
37  Massflow_Blender_2                     439844 non-null  float64
38  Blend_Potency_Blender_1                439844 non-null  float64
39  Blend_Potency_Blender_2                439844 non-null  float64
40  OOS_Concentration_at_Blender_1_inlet   439844 non-null  float64
41  OOS_Concentration_at_Blender_2_inlet   439844 non-null  float64
42  OOS_flag                               439844 non-null  bool
43  Pre-compression_height_bottom          439844 non-null  float64
44  Pre-compresion_top_dwell_time          439844 non-null  float64
45  Pre-compression_force                  439844 non-null  float64
46  Pre-compression_displacement_top_sigma 439844 non-null  float64
47  Main_compression_height_bottom         439844 non-null  float64
48  Main_compression_top_dwell_time        439844 non-null  float64
49  Main_compression_force                 439844 non-null  float64
50  Compression_cycle_fill_depth           439844 non-null  float64
51  Filling_Shoe_M20M13_speed              439844 non-null  float64
52  Filling_Shoe_M20M23_speed              439844 non-null  float64
53  Material_inlet:_Hopper_level_detection 439844 non-null  float64
54  Ejection_force_tablet                  439844 non-null  float64
55  Net_Weight_PD1                         439842 non-null  float64
56  Net_Weight_PD2                         439842 non-null  float64
57  Net_Weight_PD3                         439842 non-null  float64
```

```
 58  Net_Weight_PD4                          439842 non-null  float64
 59  Net_Weight_PD5                          439842 non-null  float64
 60  Net_Weight_PD7                          439842 non-null  float64
 61  Material                                33649 non-null   object
 62  Material_Time                           33649 non-null   datetime64[ns]
 63  Amount_(kg)                             33649 non-null   float64
 64  Lot                                     33649 non-null   object
 65  Est_Refill_time                         32930 non-null   datetime64[ns]
 66  Drum_#                                  13732 non-null   float64
dtypes: bool(8), datetime64[ns](3), float64(54), object(2)
memory usage: 201.3+ MB
```

[ ]:
```
# merge with humidity
humidity.sort_values(by='Feeder_TimeStamp', inplace=True)
merged_all.sort_values(by='TimeStamp', inplace=True)
merged_all = pd.merge_asof(merged_all, humidity, left_on='TimeStamp',␣
 ↪right_on='Feeder_TimeStamp', tolerance=pd.Timedelta('1min'),␣
 ↪direction='nearest')
merged_all.info()
```

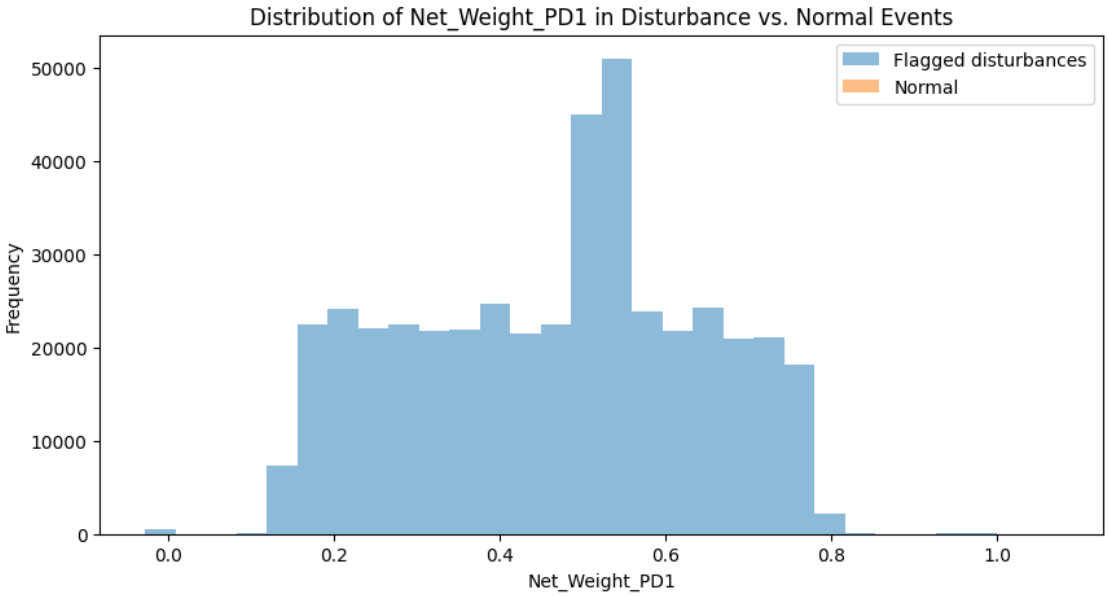
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 439844 entries, 0 to 439843
Data columns (total 71 columns):
 #   Column                                  Non-Null Count   Dtype
---  ------                                  --------------   -----
 0   TimeStamp                               439844 non-null  datetime64[ns]
 1   Feed_Factor_PD1                         439844 non-null  float64
 2   Feed_Factor_PD2                         439844 non-null  float64
 3   Feed_Factor_PD3                         439844 non-null  float64
 4   Feed_Factor_PD4                         439844 non-null  float64
 5   Feed_Factor_PD5                         439844 non-null  float64
 6   Feed_Factor_PD7                         439844 non-null  float64
 7   Screw_RPM_PD1                           439844 non-null  float64
 8   Screw_RPM_PD2                           439844 non-null  float64
 9   Screw_RPM_PD3                           439844 non-null  float64
 10  Screw_RPM_PD4                           439844 non-null  float64
 11  Screw_RPM_PD5                           439844 non-null  float64
 12  Screw_RPM_PD7                           439844 non-null  float64
 13  Massflow_PD_1                           439844 non-null  float64
 14  Massflow_PD_2                           439844 non-null  float64
 15  Massflow_PD_3                           439844 non-null  float64
 16  Massflow_PD_4                           439844 non-null  float64
 17  Massflow_PD_5                           439844 non-null  float64
 18  Massflow_PD_7                           439844 non-null  float64
 19  %_PD1                                   439844 non-null  float64
 20  %_PD2                                   439844 non-null  float64
 21  %_PD3                                   439844 non-null  float64
 22  %_PD4                                   439844 non-null  float64
```

```
23  Feed_Factor_PD1_deviation               439844 non-null  float64
24  Feed_Factor_PD1_flag                    439844 non-null  bool
25  Feed_Factor_PD2_deviation               439844 non-null  float64
26  Feed_Factor_PD2_flag                    439844 non-null  bool
27  Feed_Factor_PD3_deviation               439844 non-null  float64
28  Feed_Factor_PD3_flag                    439844 non-null  bool
29  Feed_Factor_PD4_deviation               439844 non-null  float64
30  Feed_Factor_PD4_flag                    439844 non-null  bool
31  Feed_Factor_PD5_deviation               439844 non-null  float64
32  Feed_Factor_PD5_flag                    439844 non-null  bool
33  Feed_Factor_PD7_deviation               439844 non-null  float64
34  Feed_Factor_PD7_flag                    439844 non-null  bool
35  overall_flag                            439844 non-null  bool
36  Massflow_Blender_1                      439844 non-null  float64
37  Massflow_Blender_2                      439844 non-null  float64
38  Blend_Potency_Blender_1                 439844 non-null  float64
39  Blend_Potency_Blender_2                 439844 non-null  float64
40  OOS_Concentration_at_Blender_1_inlet    439844 non-null  float64
41  OOS_Concentration_at_Blender_2_inlet    439844 non-null  float64
42  OOS_flag                                439844 non-null  bool
43  Pre-compression_height_bottom           439844 non-null  float64
44  Pre-compresion_top_dwell_time           439844 non-null  float64
45  Pre-compression_force                   439844 non-null  float64
46  Pre-compression_displacement_top_sigma  439844 non-null  float64
47  Main_compression_height_bottom          439844 non-null  float64
48  Main_compression_top_dwell_time         439844 non-null  float64
49  Main_compression_force                  439844 non-null  float64
50  Compression_cycle_fill_depth            439844 non-null  float64
51  Filling_Shoe_M20M13_speed               439844 non-null  float64
52  Filling_Shoe_M20M23_speed               439844 non-null  float64
53  Material_inlet:_Hopper_level_detection  439844 non-null  float64
54  Ejection_force_tablet                   439844 non-null  float64
55  Net_Weight_PD1                          439842 non-null  float64
56  Net_Weight_PD2                          439842 non-null  float64
57  Net_Weight_PD3                          439842 non-null  float64
58  Net_Weight_PD4                          439842 non-null  float64
59  Net_Weight_PD5                          439842 non-null  float64
60  Net_Weight_PD7                          439842 non-null  float64
61  Material                                33649 non-null   object
62  Material_Time                           33649 non-null   datetime64[ns]
63  Amount_(kg)                             33649 non-null   float64
64  Lot                                     33649 non-null   object
65  Est_Refill_time                         32930 non-null   datetime64[ns]
66  Drum_#                                  13732 non-null   float64
67  Feeder_TimeStamp                        76905 non-null   datetime64[ns]
68  Feeder_Relative_Humidity_Pct            76905 non-null   float64
69  Tablet_Press_TimeStamp                  76905 non-null   datetime64[ns]
70  Tablet_Press_Relative_Humidity_Pct      76905 non-null   float64
```

```
dtypes: bool(8), datetime64[ns](5), float64(56), object(2)
memory usage: 214.8+ MB
```

[ ]:
```python
# For illustration, suppose tablet press net weight is a relevant quality␣
 ↪metric.
# (Adjust the column name as necessary; here we assume a column like␣
 ↪'Net_Weight_PD1' exists
# either in liw_feeders2 or in the merged tablet press data.)
if 'Net_Weight_PD1' in merged_all.columns:
    quality_metric = 'Net_Weight_PD1'
else:
    # Alternatively, choose another column or compute a proxy.
    quality_metric = 'Ejection_force_tablet'  # example alternative

# Separate events: those with any feeder flag (overall_flag==True) vs.␣
 ↪non-flagged.
disturbance_events = merged_all[merged_all['overall_flag']]
normal_events = merged_all[~merged_all['overall_flag']]

print("Mean {} for flagged events: {:.2f}".format(
    quality_metric, disturbance_events[quality_metric].mean()))
print("Mean {} for normal events: {:.2f}".format(
    quality_metric, normal_events[quality_metric].mean()))

# You could also plot time series or distributions to visualize differences.

plt.figure(figsize=(10,5))
plt.hist(disturbance_events[quality_metric], bins=30, alpha=0.5, label='Flagged␣
 ↪disturbances')
plt.hist(normal_events[quality_metric], bins=30, alpha=0.5, label='Normal')
plt.xlabel(quality_metric)
plt.ylabel('Frequency')
plt.legend()
plt.title("Distribution of {} in Disturbance vs. Normal Events".
 ↪format(quality_metric))
plt.show()
```

```
Mean Net_Weight_PD1 for flagged events: 0.47
Mean Net_Weight_PD1 for normal events: nan
```



4 Optimization (The Whole Shebang!)

[ ]:
```
# ============================================================================
# 1. Data Preparation (assume your DataFrames are already loaded)
# ============================================================================
# Example names:
#   blenders: DataFrame from blenders.info() with columns:
#       TimeStamp, Massflow_Blender_1, Massflow_Blender_2,␣
 ↪Blend_Potency_Blender_1, etc.
#   liw_feeders1: DataFrame from liw_feeders1.info() with columns:
#       TimeStamp, Feed_Factor_PD1, Feed_Factor_PD2, Feed_Factor_PD3,␣
 ↪Feed_Factor_PD4,
#       Feed_Factor_PD5, Feed_Factor_PD7, Screw_RPM_PD1,..., Massflow_PD_1,...,␣
 ↪%_PD1,...
#   liw_feeders2: DataFrame from liw_feeders2.info() (not used here explicitly)

# For this example we use df_liw1 to compute average feed factors and RPM␣
 ↪bounds.
feeders = ['PD1', 'PD2', 'PD3', 'PD4', 'PD5', 'PD7']

# Compute average feed factors from liw_feeders1
avg_feed_factor = {}
for feeder in feeders:
    col = f'Feed_Factor_{feeder}'
    avg_feed_factor[feeder] = liw_feeders1[col].mean()
```

```
# Set bounds for screw speeds based on historical data from liw_feeders1
rpm_bounds = {}
for feeder in feeders:
    col = f'Screw_RPM_{feeder}'
    rpm_bounds[feeder] = (liw_feeders1[col].min(), liw_feeders1[col].max())

# Also, compute average mass flow per feeder as a reference (from liw_feeders1)
avg_mass_flow = {}
for feeder in feeders:
    # Column names: for PD1-PD5 and PD7: "Massflow_PD_1", ..., "Massflow_PD_7"
    # (Adjust column names if needed.)
    col = f'Massflow_PD_{feeder[-1]}' if feeder != 'PD7' else 'Massflow_PD_7'
    avg_mass_flow[feeder] = liw_feeders1[col].mean()

# Assume a target blend potency (percent) for Blender 1.
# For example, if a 100% potency target means the feed from PD1-PD5
# should represent 100% of the blend (PD7 may be a diluent or lubricant),
# then we define:
target_blend_potency = 100.0  # target percentage
tolerance = 10.0  # allowed deviation (±10%)

# ============================================================================
# 2. Define the Optimization Model in Pyomo
# ============================================================================

model = pyo.ConcreteModel()

# Set of feeders under control
model.feeders = pyo.Set(initialize=feeders)

# Initial guess function for RPM: midpoint of the historical bounds.
def init_rpm(model, feeder):
    lb, ub = rpm_bounds[feeder]
    # Ensure a positive guess, even if the lower bound is zero.
    return (lb + ub) / 2 if (lb + ub) > 0 else 1.0

# Decision variables: Adjusted screw speeds (RPM) for each feeder.
# These are our controllable inputs. Their bounds come from historical data.
model.rpm = pyo.Var(model.feeders, bounds=lambda model, f: rpm_bounds[f],␣
 ↪initialize=init_rpm)


# Expression: Calculate the mass flow for each feeder using the rule:
# mass_flow_i = Feed_Factor_i * RPM_i
def mass_flow_expr(model, feeder):
    return avg_feed_factor[feeder] * model.rpm[feeder]
model.mass_flow = pyo.Expression(model.feeders, rule=mass_flow_expr)
```

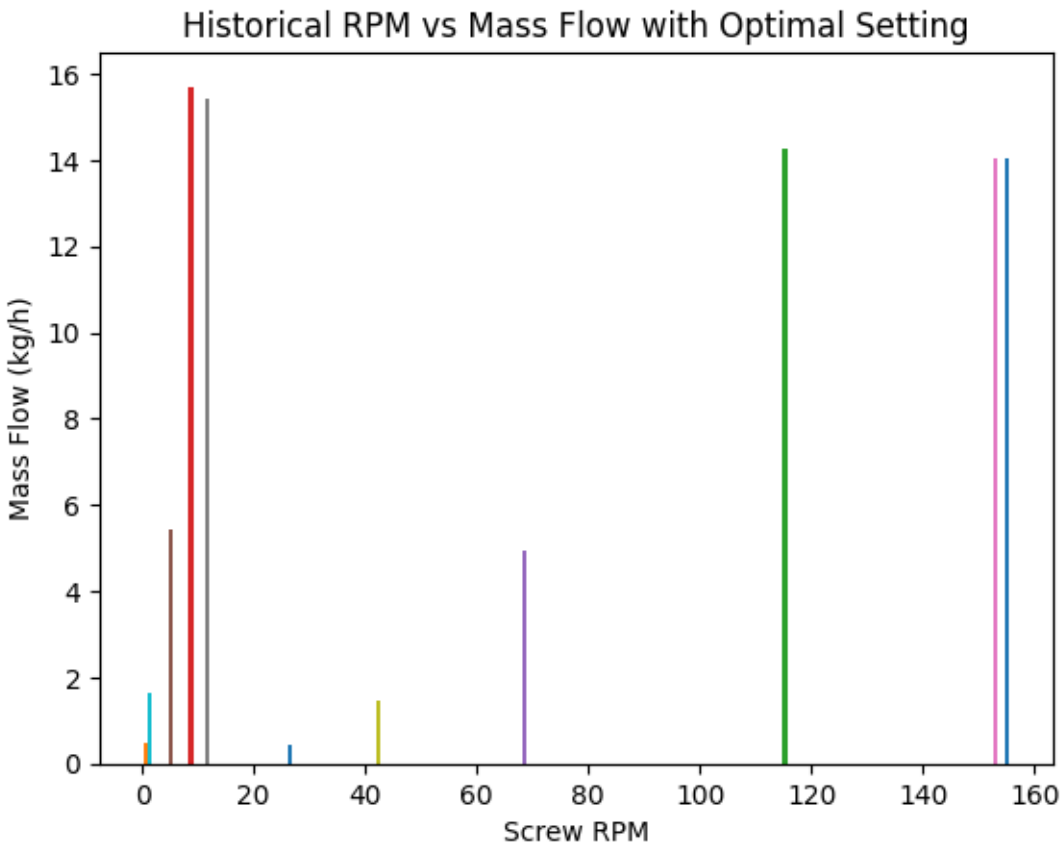
```
# Total mass flow into Blender 1 is assumed to be the sum of feeds from PD1 to␣
 ↪PD5.
model.massflow_blender1 = pyo.Expression(expr = sum(model.mass_flow[f] for f in␣
 ↪['PD1','PD2','PD3','PD4','PD5']))

# For Blender 2, we assume the total mass is that from Blender 1 plus feed PD7.
model.massflow_blender2 = pyo.Expression(expr = model.massflow_blender1 + model.
 ↪mass_flow['PD7'])

# ============================================================================
# 3. Define Objective and Constraints
# ============================================================================

# Objective: Maximize overall throughput (mass flow into Blender 2).
model.obj = pyo.Objective(expr = model.massflow_blender2, sense=pyo.maximize)

# Constraint: Blend potency
# For example, a simple model assumes that the blend potency is the fraction of␣
 ↪the
# "active" mass (from PD1-PD5) in the total mass (active + diluent PD7).
# We force the weighted potency to lie within a tolerance around the target.
def blend_potency_constraint_rule(model):
    # If PD7 acts as a diluent with zero potency,
    # then potency (in %) = (massflow_blender1 / massflow_blender2) * 100.
    potency = (model.massflow_blender1 / model.massflow_blender2)*100
    # Return a range constraint: target ± tolerance.
    return (target_blend_potency - tolerance, potency, target_blend_potency +␣
 ↪tolerance)
model.blend_potency_constraint = pyo.
 ↪Constraint(rule=blend_potency_constraint_rule)

# Additional Constraints: Ensure that individual feeder mass flows remain close␣
 ↪to their average
# (this can be interpreted as a quality/operational reliability constraint)
def mass_flow_bounds_rule(model, feeder):
    return (0.9 * avg_mass_flow[feeder], model.mass_flow[feeder], 1.1 *␣
 ↪avg_mass_flow[feeder])
model.mass_flow_constraints = pyo.Constraint(model.feeders,␣
 ↪rule=mass_flow_bounds_rule)

# ============================================================================
# 4. Solve the Model
# ============================================================================

solver = pyo.SolverFactory('ipopt')
```

```
results = solver.solve(model, tee=True)

# ============================================================================
# 5. Output the Results
# ============================================================================

print("Optimal feeder settings:")
for feeder in model.feeders:
    rpm_val = pyo.value(model.rpm[feeder])
    mass_val = pyo.value(model.mass_flow[feeder])
    print(f"  {feeder}: Optimal RPM = {rpm_val:.2f}, Mass Flow = {mass_val:.2f}␣
 ↪kg/h")

print("\nAggregate flows:")
print(f"  Total Mass Flow into Blender 1 = {pyo.value(model.massflow_blender1):.
 ↪2f} kg/h")
print(f"  Total Mass Flow into Blender 2 = {pyo.value(model.massflow_blender2):.
 ↪2f} kg/h")

# Compute and report the blend potency based on the model expression:
blend_potency = (pyo.value(model.massflow_blender1) / pyo.value(model.
 ↪massflow_blender2))*100
print(f"\nBlend Potency = {blend_potency:.2f} % (target =␣
 ↪{target_blend_potency} ± {tolerance} %)")
```

```
Ipopt 3.14.17:

******************************************************************************
This program contains Ipopt, a library for large-scale nonlinear optimization.
 Ipopt is released as open source code under the Eclipse Public License (EPL).
         For more information visit https://github.com/coin-or/Ipopt
******************************************************************************

This is Ipopt version 3.14.17, running with linear solver MUMPS 5.7.3.

Number of nonzeros in equality constraint Jacobian…:        0
Number of nonzeros in inequality constraint Jacobian.:       12
Number of nonzeros in Lagrangian Hessian…:       21

Total number of variables…:        6
                     variables with only lower bounds:        0
                variables with lower and upper bounds:        6
                     variables with only upper bounds:        0
Total number of equality constraints…:        0
Total number of inequality constraints…:        7
        inequality constraints with only lower bounds:        0
   inequality constraints with lower and upper bounds:        7
```

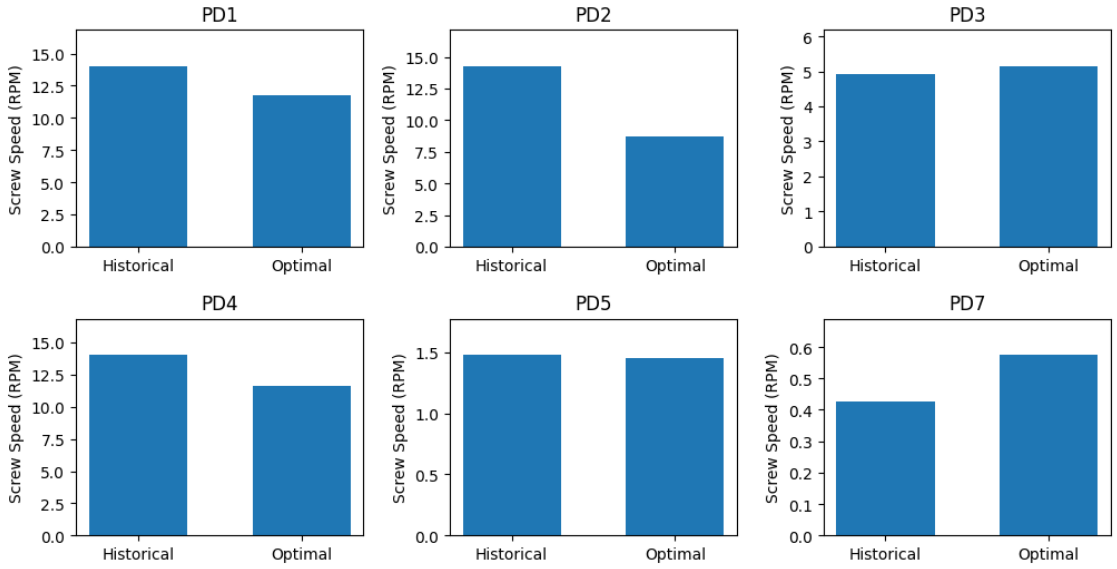
```
        inequality constraints with only upper bounds:        0

iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
   0 -8.9674174e+02 2.14e+02 7.63e-01  -1.0 0.00e+00    -  0.00e+00 0.00e+00   0
   1 -1.5295276e+02 2.51e+01 1.37e+00  -1.0 2.03e+02    -  4.97e-01 8.82e-01h  1
   2 -5.3711404e+01 0.00e+00 5.09e-01  -1.0 2.39e+01    -  5.18e-01 1.00e+00h  1
   3 -5.3638254e+01 0.00e+00 5.40e-03  -1.0 2.80e+00    -  9.94e-01 1.00e+00h  1
   4 -5.3939140e+01 0.00e+00 4.83e-04  -1.7 7.15e-02    -  1.00e+00 1.00e+00f  1
   5 -5.4067144e+01 0.00e+00 1.21e-03  -2.5 3.31e-02    -  9.53e-01 1.00e+00f  1
   6 -5.4088556e+01 0.00e+00 4.42e-08  -3.8 1.19e-02    -  1.00e+00 1.00e+00f  1
   7 -5.4089520e+01 0.00e+00 1.08e-10  -5.7 2.92e-04    -  1.00e+00 1.00e+00h  1
   8 -5.4089532e+01 0.00e+00 1.24e-14  -8.6 3.85e-06    -  1.00e+00 1.00e+00h  1

Number of Iterations…: 8

                                   (scaled)                 (unscaled)
Objective…:  -5.4089531811705406e+01   -5.4089531811705406e+01
Dual infeasibility…:   1.2405460014726712e-14    1.2405460014726712e-14
Constraint violation…:   0.0000000000000000e+00    0.0000000000000000e+00
Variable bound violation:   0.0000000000000000e+00    0.0000000000000000e+00
Complementarity…:   2.5508620491599280e-09    2.5508620491599280e-09
Overall NLP error…:   2.5508620491599280e-09    2.5508620491599280e-09


Number of objective function evaluations             = 9
Number of objective gradient evaluations             = 9
Number of equality constraint evaluations            = 0
Number of inequality constraint evaluations          = 9
Number of equality constraint Jacobian evaluations   = 0
Number of inequality constraint Jacobian evaluations = 9
Number of Lagrangian Hessian evaluations             = 8
Total seconds in IPOPT                               = 0.014

EXIT: Optimal Solution Found.
Optimal feeder settings:
  PD1: Optimal RPM = 11.79, Mass Flow = 15.43 kg/h
  PD2: Optimal RPM = 8.75, Mass Flow = 15.70 kg/h
  PD3: Optimal RPM = 5.15, Mass Flow = 5.42 kg/h
  PD4: Optimal RPM = 11.63, Mass Flow = 15.43 kg/h
  PD5: Optimal RPM = 1.45, Mass Flow = 1.62 kg/h
  PD7: Optimal RPM = 0.57, Mass Flow = 0.47 kg/h

Aggregate flows:
  Total Mass Flow into Blender 1 = 53.62 kg/h
  Total Mass Flow into Blender 2 = 54.09 kg/h

Blend Potency = 99.13 % (target = 100.0 ± 10.0 %)
```

[ ]:
```
feeders = ['PD1','PD2','PD3','PD4','PD5','PD7']

fig, ax = plt.subplots()
for feeder in feeders:
    rpm_val = pyo.value(model.rpm[feeder])
    mass_val = pyo.value(model.mass_flow[feeder])
    # historical data
    ax.bar(
        np.mean(liw_feeders1[f'Screw_RPM_{feeder}']),
        np.mean(liw_feeders1[f'Massflow_PD_{feeder[-1]}' if feeder!='PD7' else␣
 ↪'Massflow_PD_7']),
        label=f'{feeder} average'
    )
    # optimal point
    ax.bar(
        round(rpm_val, 2),
        round(mass_val, 2),
        #marker='X',
        label=f'Optimal val for {feeder}',
        #s=100
    )
ax.set_xlabel('Screw RPM')
ax.set_ylabel('Mass Flow (kg/h)')
ax.legend()
ax.set_title('Historical RPM vs Mass Flow with Optimal Setting')
plt.show()
```



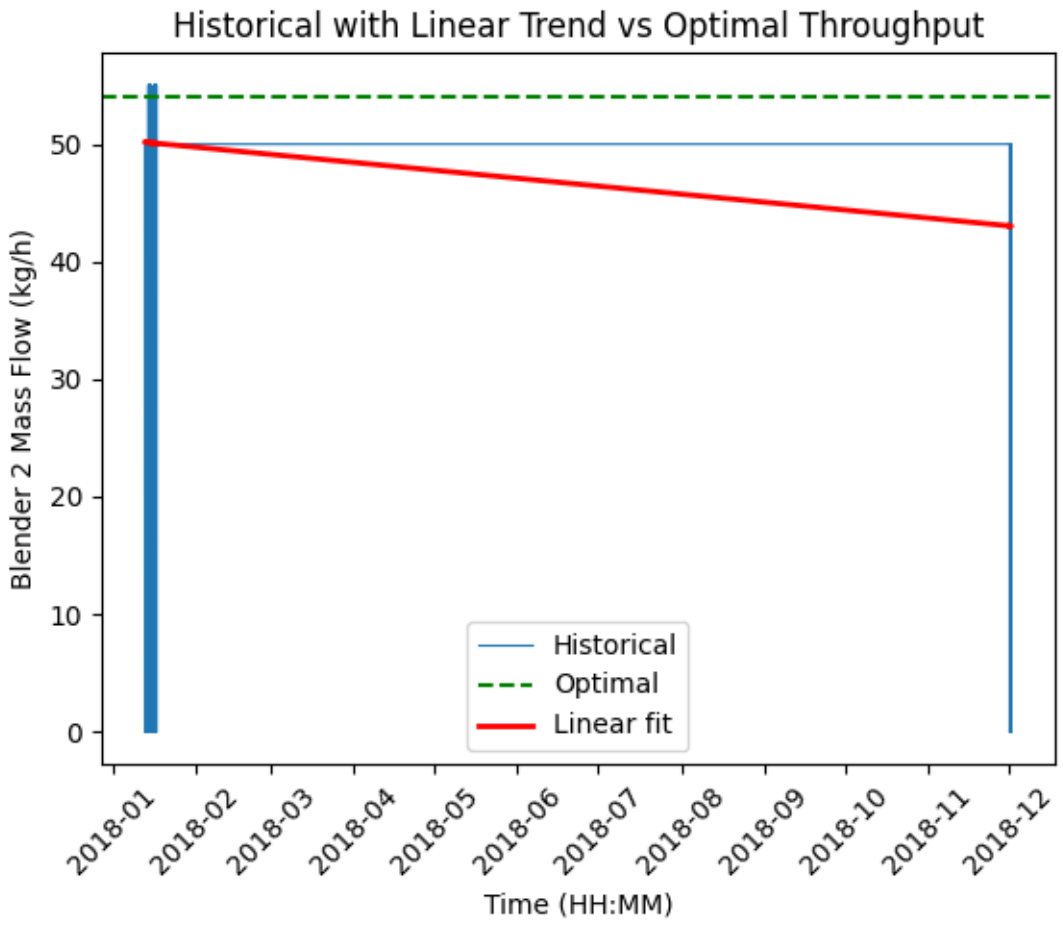
[ ]:
```
import matplotlib.gridspec as gridspec
avg_rpm = [
    liw_feeders1[f'Massflow_PD_{f[-1]}' if f!='PD7' else 'Massflow_PD_7'].mean()
    for f in feeders
]
opt_rpm = [pyo.value(model.rpm[f]) for f in feeders]

# total_hist = sum(avg_mass)
# total_opt = sum(opt_mass)

# 3) Create 2×3 feeder subplots + summary
fig = plt.figure(figsize=(12, 8))
gs = gridspec.GridSpec(3, 3, height_ratios=[1, 1, 0.5], hspace=0.4, wspace=0.3)

# Feeder plots
for idx, f in enumerate(feeders):
    ax = fig.add_subplot(gs[idx//3, idx%3])
    ax.bar(['Historical','Optimal'], [avg_rpm[idx], opt_rpm[idx]], width=0.6)
    ax.set_title(f)
```

```
    ax.set_ylabel('Screw Speed (RPM)')
    ax.set_ylim(0, max(avg_rpm[idx], opt_rpm[idx]) * 1.2)

# Summary plot across bottom row
# ax_sum = fig.add_subplot(gs[2, :])
# ax_sum.bar(['Total Hist','Total Opt'], [total_hist, total_opt], width=0.6)
# ax_sum.set_title('Aggregate Total Mass Flow')
# ax_sum.set_ylabel('Mass Flow (kg/h)')
# ax_sum.set_ylim(0, max(total_hist, total_opt) * 1.2)

plt.tight_layout()
plt.show()
```

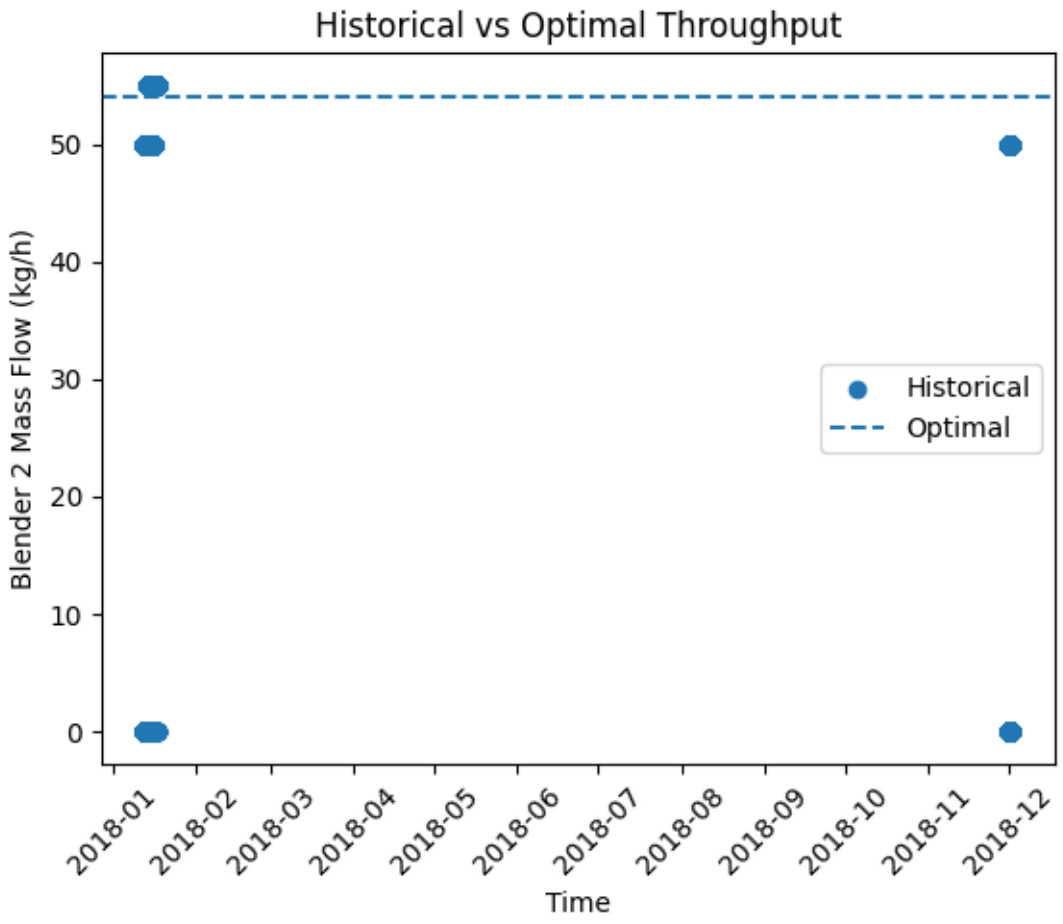


[ ]:
```
import matplotlib.dates as mdates
opt_throughput = round(pyo.value(model.massflow_blender2), 2)

# 1. Create a 24-hour time string column
# ---------------------------------------
# If your Timestamp column is a datetime64 dtype, use .dt.strftime:
blenders['Time 24h'] = blenders['TimeStamp'].dt.strftime('%H:%M')

# (Alternatively, if you really want a numeric hour+fraction, you could do:
blenders['Time 24h (num)'] = blenders['TimeStamp'].dt.hour \
                           + blenders['TimeStamp'].dt.minute/60.0
# # )

x_dt = blenders['TimeStamp']
y = blenders['Massflow_Blender_2']
```

```
# 2. Re-plot throughput vs. optimal with the new Time 24h column and linear fit
# ------------------------------------------------------------
fig, ax = plt.subplots()

x_num = mdates.date2num(x_dt)
m, b = np.polyfit(x_num, y, 1)
y_fit = m * x_num + b

ax.plot(
    blenders['TimeStamp'],
    blenders['Massflow_Blender_2'],
    label='Historical',
    linewidth=0.8,
    )

ax.axhline(
    y=opt_throughput,
    linestyle='--',
    label='Optimal',
    c='green'
)

ax.plot(
    x_dt, y_fit,
    label='Linear fit',
    linewidth=2.0,
    c='r'
)
ax.set_xlabel('Time (HH:MM)')
ax.set_ylabel('Blender 2 Mass Flow (kg/h)')
ax.legend()
ax.set_title('Historical with Linear Trend vs Optimal Throughput')
plt.xticks(rotation=45)   # rotate labels so they don't overlap
plt.show()
```

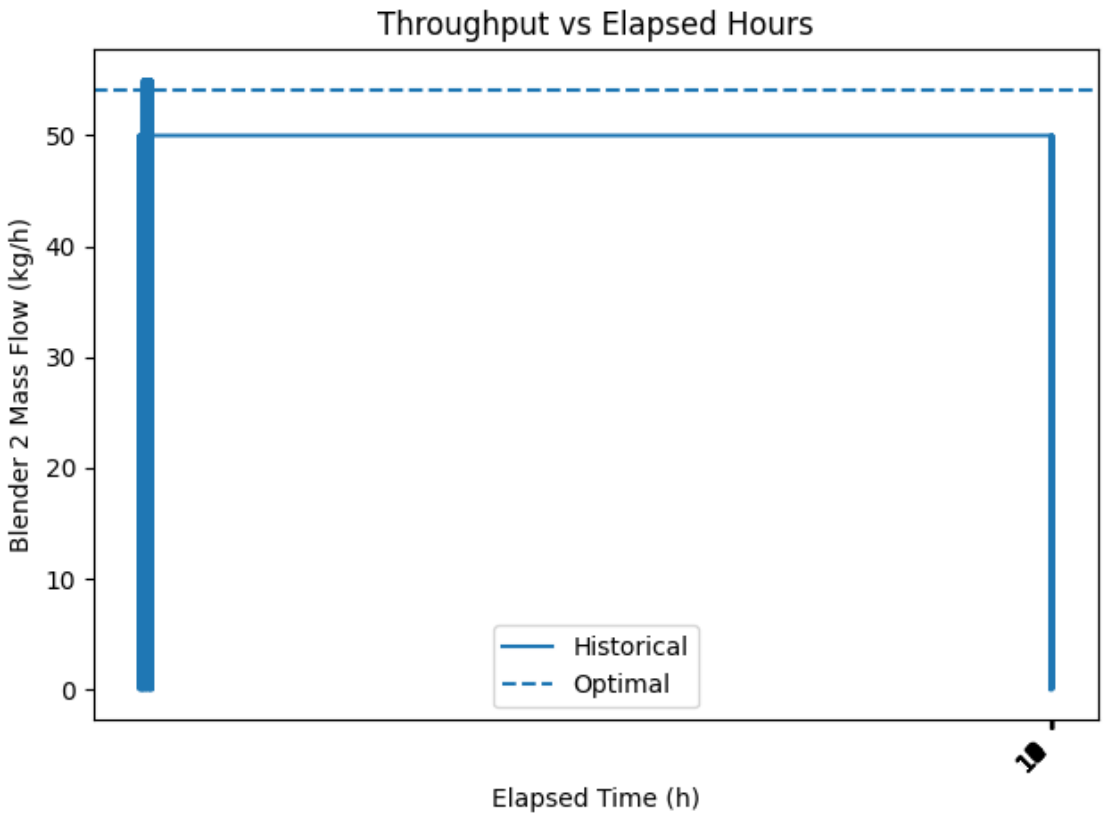


[ ]:
```
opt_throughput = round(pyo.value(model.massflow_blender2), 2)
fig, ax = plt.subplots()
ax.scatter(
    blenders['TimeStamp'],
    blenders['Massflow_Blender_2'],
    label='Historical'
)
ax.axhline(
    y=opt_throughput,
    linestyle='--',
    label='Optimal'
)
ax.set_xlabel('Time')
ax.set_ylabel('Blender 2 Mass Flow (kg/h)')
ax.legend()
ax.set_title('Historical vs Optimal Throughput')
plt.xticks(rotation=45)
plt.show()
```



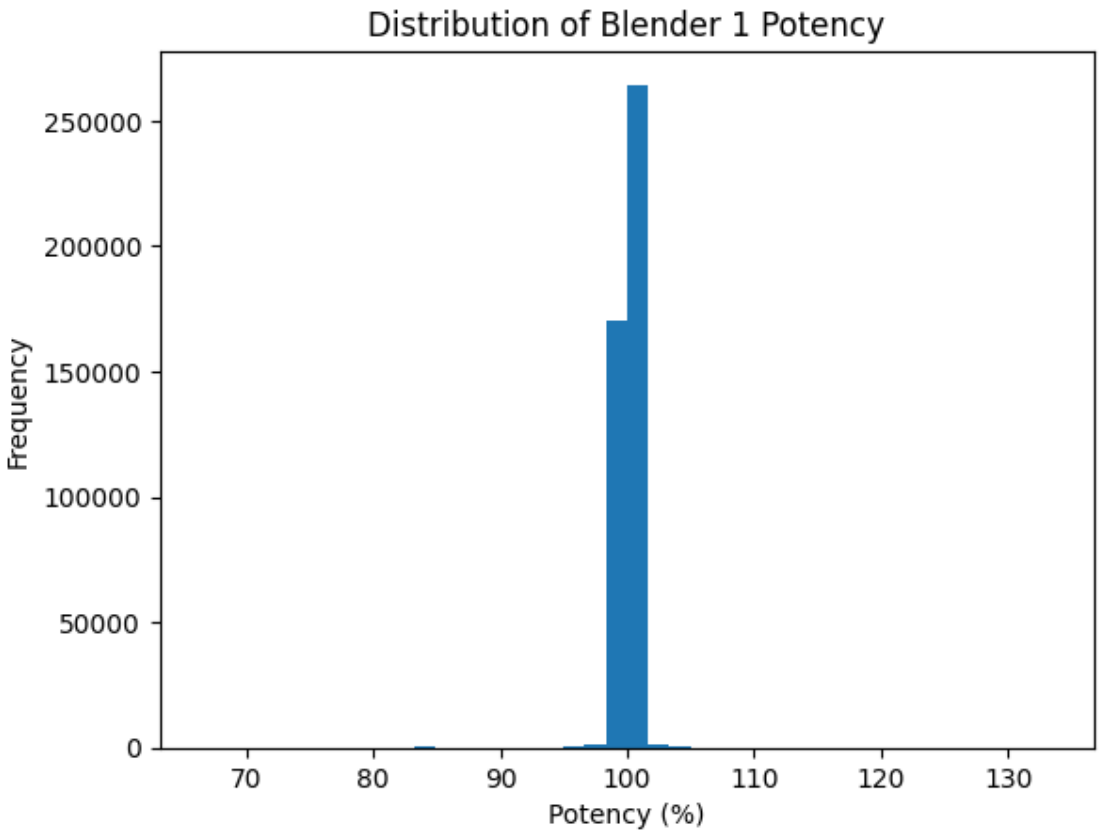
[ ]:
```
import matplotlib.dates as mdates

opt_throughput = round(pyo.value(model.massflow_blender2), 2)

# 1) Compute elapsed hours as a float
blenders['Elapsed_h'] = (
    blenders['TimeStamp'] - blenders['TimeStamp'].iloc[0]
).dt.total_seconds() / 3600

x = blenders['Elapsed_h']
y = blenders['Massflow_Blender_2']

fig, ax = plt.subplots()
ax.plot(x, y, label='Historical')
ax.axhline(y=opt_throughput, linestyle='--', label='Optimal')

# 2) Manually set major ticks at every hour:
max_h = np.ceil(x.max())
```

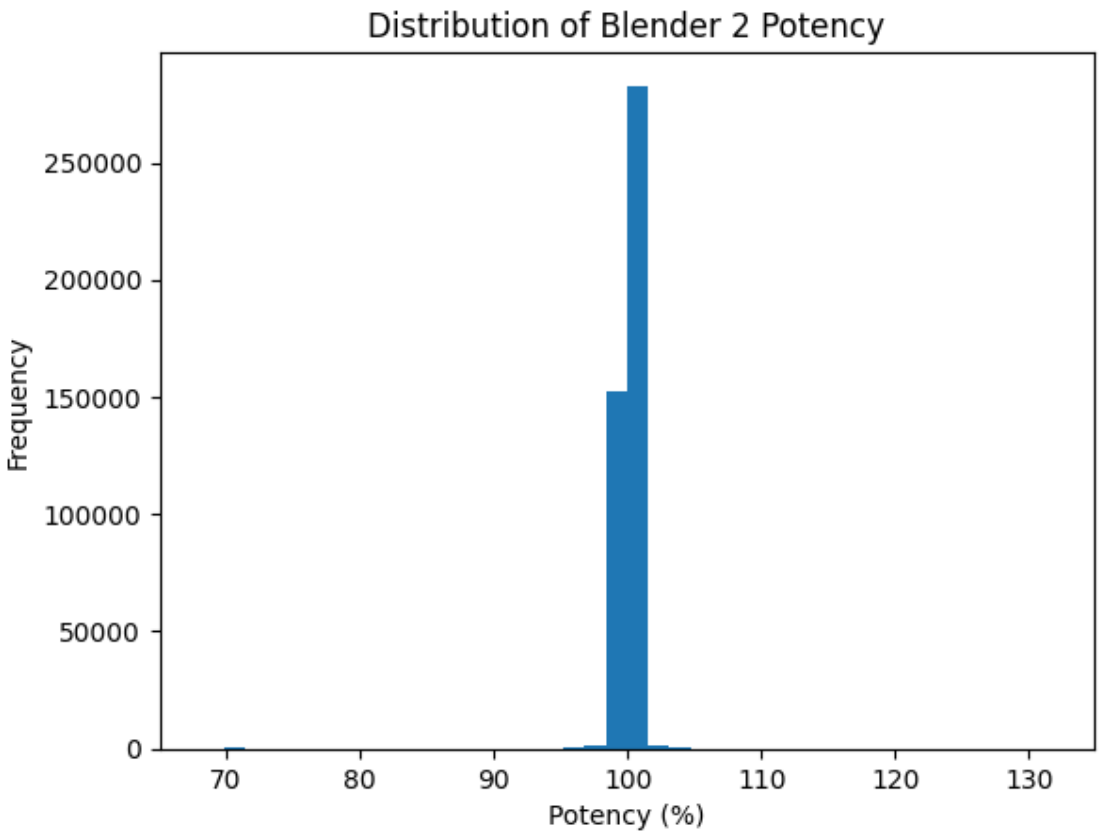
```
hour_ticks = np.arange(0, max_h+1, 1)        # every 1 h
ax.set_xticks(hour_ticks)

# 3) (optional) add minor ticks at 30 min, but don't label them
minor = np.arange(0, max_h+1, 0.5)           # every 0.5 h
ax.set_xticks(minor, minor=True)
#ax.xaxis.set_minor_formatter(plt.NullFormatter())

ax.set_xlabel('Elapsed Time (h)')
ax.set_ylabel('Blender 2 Mass Flow (kg/h)')
ax.set_title('Throughput vs Elapsed Hours')
ax.legend()

plt.setp(ax.get_xticklabels(which='major'), rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



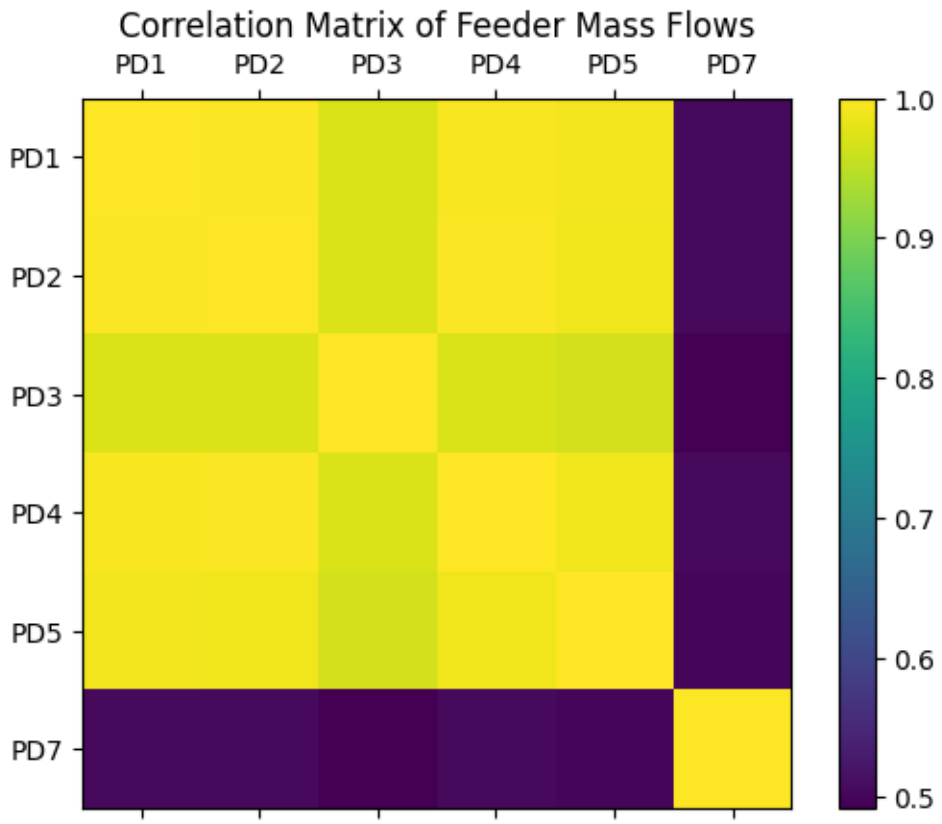
[ ]:
```
fig, ax = plt.subplots()
ax.hist(blenders['Blend_Potency_Blender_1'], bins=40)
ax.set_xlabel('Potency (%)')
```

```
ax.set_ylabel('Frequency')
ax.set_title('Distribution of Blender 1 Potency')
plt.show()

fig, ax = plt.subplots()
ax.hist(blenders['Blend_Potency_Blender_2'], bins=40)
ax.set_xlabel('Potency (%)')
ax.set_ylabel('Frequency')
ax.set_title('Distribution of Blender 2 Potency')
plt.show()
```

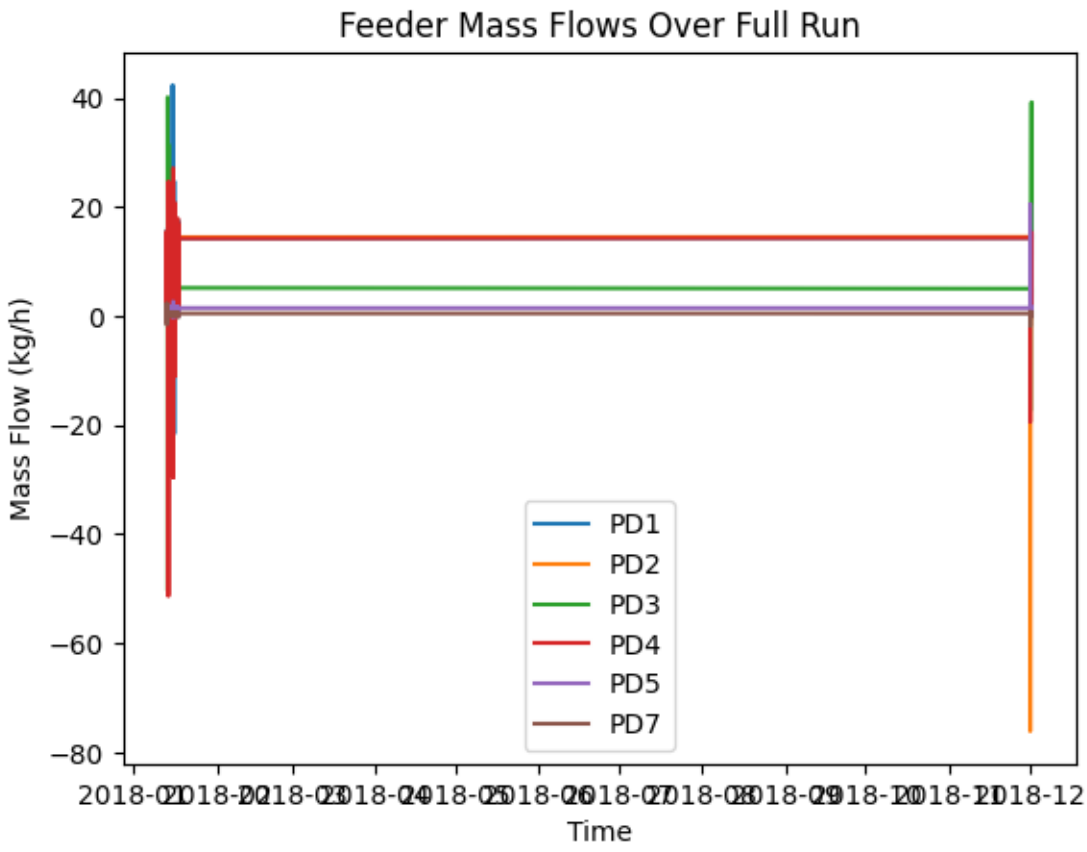




[ ]:
```
feed_cols = [f'Massflow_PD_{f[-1]}' if f!='PD7' else 'Massflow_PD_7'
             for f in feeders]
corr = liw_feeders1[feed_cols].corr()

fig, ax = plt.subplots()
cax = ax.matshow(corr)
fig.colorbar(cax)
ax.set_xticks(np.arange(len(feeders)))
ax.set_yticks(np.arange(len(feeders)))
ax.set_xticklabels(feeders)
ax.set_yticklabels(feeders)
ax.set_title('Correlation Matrix of Feeder Mass Flows')
plt.show()
```



[ ]:
```
fig, ax = plt.subplots()
for f in feeders:
    col = f'Massflow_PD_{f[-1]}' if f!='PD7' else 'Massflow_PD_7'
    ax.plot(liw_feeders1['TimeStamp'], liw_feeders1[col], label=f)
ax.set_xlabel('Time')
ax.set_ylabel('Mass Flow (kg/h)')
ax.legend()
ax.set_title('Feeder Mass Flows Over Full Run')
plt.show()
```


Feeder Mass Flows Over Full Run
Mass Flow (kg/h)
40
20
0
−20
−40
−60
−80
PD1
PD2
PD3
PD4
PD5
PD7
Time