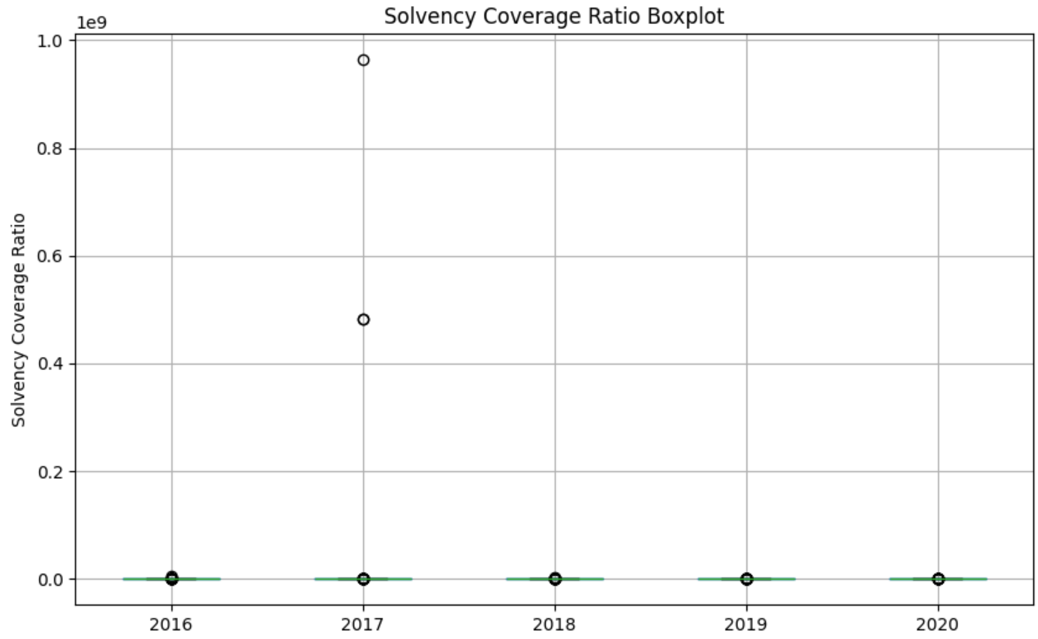


- Using describe method, there is extereme data value in the dataset in year 2017, with a solvency coverage ratio 963,584,000.00. Upon sorting by year 2017, the firm identified is Firm 216. The solvency coverage ratio are followed closely by Firm 1, Firm 131, and Firm 127 with 481,792,000%, 481,792,000% and 182,412% each respectively.
-

	2016	2017	2018	2019	2020
count	3.250000e+02	3.250000e+02	3.250000e+02	325.000000	325.000000
mean	1.286947e+04	5.930313e+06	1.246741e+04	533.017116	514.215109
std	2.319518e+05	6.529400e+07	2.141325e+05	9539.236980	9229.786796
min	-1.974450e+00	-1.973652e+00	-5.515428e-01	-0.669013	-1.066521
25%	1.276845e+00	1.302423e+00	1.268210e+00	1.177754	0.000000
50%	1.662747e+00	1.755881e+00	1.693416e+00	1.710544	1.565516
75%	2.780175e+00	3.203697e+00	2.795907e+00	2.691263	2.367988
max	4.181573e+06	9.635840e+08	3.856018e+06	171974.690816	166394.575872

-

Firms_ID	2016	2017	2018	2019	2020
Firm_216	1.49	963,584,000.00	0.00	0.00	0.00
Firm_1	1.98	481,792,000.00	0.00	0.00	0.00
Firm_131	1.75	481,792,000.00	0.00	0.00	0.00
Firm_127	0.00	182,412.23	194,753.82	171,974.69	166,394.58
Firm_291	146.89	128.53	150.81	261.57	0.00
Firm_319	57.84	67.23	27.86	39.24	0.00
Firm_16	12.25	41.83	26.13	0.00	0.00
Firm_125	2.92	39.83	4.33	14.87	3.65
Firm_103	26.18	34.88	37.91	37.56	6.35
Firm_177	0.00	23.29	31.01	53.35	23.44

- The data for Firm 127 the following years from 2018 to 2020 shows a close average ranging from 166,394% to 194,753%.

Annex

The following section details out the Python script for this report:

In [ ]:
```
# script made by Muhammad Syarmine Bin Mohd Shah
# Feels free to contact me at syarmineshah@yahoo.com for feedback
# import required libraries and packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import openpyxl as xl # import the dataset using pandas pd.read_excel with openp
import plotly_express as px
```

Getting and structuring the data

In [ ]:
```
# get data from github
url = "https://github.com/Syarmine/Portfolio/raw/main/BoE%20Assignment/Data%20fc
dict_df = pd.read_excel(url,
                header=0, #add indexing without including the dataset
                sheet_name=[0,1]) # get sheet 1 and sheet 2

df = dict_df.get(0)
df_1 = dict_df.get(1)
```

In [ ]:
```
# check the data shape
print(df.shape)
print(df_1.shape)

# check the Dataset 1 head
df.head()
```

```
(326, 41)
(326, 46)
```

Out[ ]:

	Unnamed: 0	NWP (£m)	NWP (£m) .1	NWP (£m) .2	NWP (£m) .3	NWP (£m) .4
0	NaN	2016YE	2017YE	2018YE	2019YE	2020YE
1	Firm 1	-13779.815629	0	0	0	0
2	Firm 2	28.178059	26.865049	25.064438	23.226445	21.718558
3	Firm 3	0	75.609681	70.578732	78.432782	85.73583
4	Firm 4	22344.199923	23963.910709	25760.390158	25512.748836	24996.021042

5 rows × 41 columns

In [ ]:
```
# check the Dataset 2 head
```

```
df_1.head()
```

Out[ ]:

	Unnamed: 0	Gross claims incurred (£m)	Gross claims incurred (£m).1	Gross claims incurred (£m).2	Gross claims incurred (£m).3	Gross claims incurred (£m).4	Gross BEL (inc. TPs as whole, pre-TMTP) (£m)	Gros (inc as w TI (£
0	NaN	2016YE	2017YE	2018YE	2019YE	2020YE	2016YE	20
1	Firm 1	0	0.046674	0	0	0	0	7.67
2	Firm 2	39.241067	35.948249	29.002244	0	0	163.597907	195.52
3	Firm 3	0	0	0	0	0	0	
4	Firm 4	17.125976	75.535951	119.426995	35.884204	6.567988	-18.297099	22.64

5 rows × 46 columns

In [ ]:

```
# check for missing values for Dataset 1
df.isnull().sum()
```

Out[ ]:

```
Unnamed: 0                                              1
NWP (£m)                                                0
NWP (£m) .1                                             0
NWP (£m) .2                                             0
NWP (£m) .3                                             0
NWP (£m) .4                                             0
SCR (£m)                                                0
SCR (£m).1                                              0
SCR (£m).2                                              0
SCR (£m).3                                              0
SCR (£m).4                                              0
EoF for SCR (£m)                                        0
EoF for SCR (£m).1                                      0
EoF for SCR (£m).2                                      0
EoF for SCR (£m).3                                      0
EoF for SCR (£m).4                                      0
SCR coverage ratio                                      0
SCR coverage ratio.1                                    0
SCR coverage ratio.2                                    0
SCR coverage ratio.3                                    0
SCR coverage ratio.4                                    0
GWP (£m)                                                0
GWP (£m).1                                              0
GWP (£m).2                                              0
GWP (£m).3                                              0
GWP (£m).4                                              0
Total assets (£m)                                       0
Total assets (£m).1                                     0
Total assets (£m).2                                     0
Total assets (£m).3                                     0
Total assets (£m).4                                     0
Total liabilities (£m)                                  0
Total liabilities (£m).1                                0
Total liabilities (£m).2                                0
Total liabilities (£m).3                                0
Total liabilities (£m).4                                0
Excess of assets over liabilities (£m) [= equity]       0
Excess of assets over liabilities (£m) [= equity].1     0
Excess of assets over liabilities (£m) [= equity].2     0
Excess of assets over liabilities (£m) [= equity].3     0
Excess of assets over liabilities (£m) [= equity].4     0
dtype: int64
```

In [ ]:

```
# check for missing values for Dataset 2
df_1.isnull().sum()
```

Out[ ]:
```
Unnamed: 0                                          1
Gross claims incurred (£m)                          0
Gross claims incurred (£m).1                        0
Gross claims incurred (£m).2                        0
Gross claims incurred (£m).3                        0
Gross claims incurred (£m).4                        0
Gross BEL (inc. TPs as whole, pre-TMTP) (£m)        0
Gross BEL (inc. TPs as whole, pre-TMTP) (£m).1      0
Gross BEL (inc. TPs as whole, pre-TMTP) (£m).2      0
Gross BEL (inc. TPs as whole, pre-TMTP) (£m).3      0
Gross BEL (inc. TPs as whole, pre-TMTP) (£m).4      0
Net BEL (inc. TPs as a whole, pre-TMTP) (£m)        0
Net BEL (inc. TPs as a whole, pre-TMTP) (£m).1      0
Net BEL (inc. TPs as a whole, pre-TMTP) (£m).2      0
Net BEL (inc. TPs as a whole, pre-TMTP) (£m).3      0
Net BEL (inc. TPs as a whole, pre-TMTP) (£m).4      0
Pure net claims ratio                               0
Pure net claims ratio.1                             0
Pure net claims ratio.2                             0
Pure net claims ratio.3                             0
Pure net claims ratio.4                             0
Net expense ratio                                   0
Net expense ratio.1                                 0
Net expense ratio.2                                 0
Net expense ratio.3                                 0
Net expense ratio.4                                 0
Net combined ratio                                  0
Net combined ratio.1                                0
Net combined ratio.2                                0
Net combined ratio.3                                0
Net combined ratio.4                                0
Pure gross claims ratio                             0
Pure gross claims ratio.1                           0
Pure gross claims ratio.2                           0
Pure gross claims ratio.3                           0
Pure gross claims ratio.4                           0
Gross expense ratio                                 0
Gross expense ratio.1                               0
Gross expense ratio.2                               0
Gross expense ratio.3                               0
Gross expense ratio.4                               0
Gross combined ratio                                0
Gross combined ratio.1                              0
Gross combined ratio.2                              0
Gross combined ratio.3                              0
Gross combined ratio.4                              0
dtype: int64
```

In [ ]:
```
# check the data type for Dataset 1
df.dtypes
```

Out[ ]:
```
Unnamed: 0                                              object
NWP (£m)                                                object
NWP (£m) .1                                             object
NWP (£m) .2                                             object
NWP (£m) .3                                             object
NWP (£m) .4                                             object
SCR (£m)                                                object
SCR (£m).1                                              object
SCR (£m).2                                              object
SCR (£m).3                                              object
SCR (£m).4                                              object
EoF for SCR (£m)                                        object
EoF for SCR (£m).1                                      object
EoF for SCR (£m).2                                      object
EoF for SCR (£m).3                                      object
EoF for SCR (£m).4                                      object
SCR coverage ratio                                      object
SCR coverage ratio.1                                    object
SCR coverage ratio.2                                    object
SCR coverage ratio.3                                    object
SCR coverage ratio.4                                    object
GWP (£m)                                                object
GWP (£m).1                                              object
GWP (£m).2                                              object
GWP (£m).3                                              object
GWP (£m).4                                              object
Total assets (£m)                                       object
Total assets (£m).1                                     object
Total assets (£m).2                                     object
Total assets (£m).3                                     object
Total assets (£m).4                                     object
Total liabilities (£m)                                  object
Total liabilities (£m).1                                object
Total liabilities (£m).2                                object
Total liabilities (£m).3                                object
Total liabilities (£m).4                                object
Excess of assets over liabilities (£m) [= equity]       object
Excess of assets over liabilities (£m) [= equity].1     object
Excess of assets over liabilities (£m) [= equity].2     object
Excess of assets over liabilities (£m) [= equity].3     object
Excess of assets over liabilities (£m) [= equity].4     object
dtype: object
```

In [ ]:
```
# check the data type for Dataset 2
df_1.dtypes
```

```
Out[ ]: Unnamed: 0                                              object
        Gross claims incurred (£m)                              object
        Gross claims incurred (£m).1                            object
        Gross claims incurred (£m).2                            object
        Gross claims incurred (£m).3                            object
        Gross claims incurred (£m).4                            object
        Gross BEL (inc. TPs as whole, pre-TMTP) (£m)            object
        Gross BEL (inc. TPs as whole, pre-TMTP) (£m).1          object
        Gross BEL (inc. TPs as whole, pre-TMTP) (£m).2          object
        Gross BEL (inc. TPs as whole, pre-TMTP) (£m).3          object
        Gross BEL (inc. TPs as whole, pre-TMTP) (£m).4          object
        Net BEL (inc. TPs as a whole, pre-TMTP) (£m)            object
        Net BEL (inc. TPs as a whole, pre-TMTP) (£m).1          object
        Net BEL (inc. TPs as a whole, pre-TMTP) (£m).2          object
        Net BEL (inc. TPs as a whole, pre-TMTP) (£m).3          object
        Net BEL (inc. TPs as a whole, pre-TMTP) (£m).4          object
        Pure net claims ratio                                   object
        Pure net claims ratio.1                                 object
        Pure net claims ratio.2                                 object
        Pure net claims ratio.3                                 object
        Pure net claims ratio.4                                 object
        Net expense ratio                                       object
        Net expense ratio.1                                     object
        Net expense ratio.2                                     object
        Net expense ratio.3                                     object
        Net expense ratio.4                                     object
        Net combined ratio                                      object
        Net combined ratio.1                                    object
        Net combined ratio.2                                    object
        Net combined ratio.3                                    object
        Net combined ratio.4                                    object
        Pure gross claims ratio                                 object
        Pure gross claims ratio.1                               object
        Pure gross claims ratio.2                               object
        Pure gross claims ratio.3                               object
        Pure gross claims ratio.4                               object
        Gross expense ratio                                     object
        Gross expense ratio.1                                   object
        Gross expense ratio.2                                   object
        Gross expense ratio.3                                   object
        Gross expense ratio.4                                   object
        Gross combined ratio                                    object
        Gross combined ratio.1                                  object
        Gross combined ratio.2                                  object
        Gross combined ratio.3                                  object
        Gross combined ratio.4                                  object
        dtype: object
```

```
In [ ]: # There are 326 rows and 6 columns in Dataset 1 and 326 rows and 5 columns in Da
        # The first two rows of Dataset 1 and Dataset 2 being header and the third row b
        # The first column of Dataset 1 and Dataset 2 being the index column with 'Firm_

        # As there are 2 headers rows for Dataset 1, create a new header from the first
        # Create a new header from the first two rows
        new_header = [f"{df.iloc[0, i]}_{df.columns[i]}" if not pd.isna(df.iloc[0, i]) e

        # Remove any generated trailing numbers (e.g., '.1', '.2', etc.) from the header
        import re
        new_header = [re.sub(r'\.\d+$', '', header) for header in new_header]
```

```python
# Assign the new header to the DataFrame and drop the first two rows
df.columns = new_header
df = df.drop(df.index[0])

# Reset the index
df.reset_index(drop=True, inplace=True)

# Set to display all columns
pd.set_option('display.max_columns', None)

# reinspect the dataframe
df.head(5)
```

Out[ ]:

	Unnamed: 0	2016YE_NWP (£m)	2017YE_NWP (£m)	2018YE_NWP (£m)	2019YE_NWP (£m)	2020YE_NWP (£m)
0	Firm 1	-13779.815629	0	0	0	0
1	Firm 2	28.178059	26.865049	25.064438	23.226445	21.718558
2	Firm 3	0	75.609681	70.578732	78.432782	85.73583
3	Firm 4	22344.199923	23963.910709	25760.390158	25512.748836	24996.021042
4	Firm 5	68.200993	51.663132	44.010833	42.008556	81.273653

In [ ]:

```python
# Similarly, there are 2 headers rows for Dataset 2, create a new header from th
# Create a new header from the first two rows for clarity
new_header2 = [f"{df_1.iloc[0, i]}_{df_1.columns[i]}" if not pd.isna(df_1.iloc[0

# Remove any trailing numbers (e.g., '.1', '.2', etc.) from the header
import re
new_header2 = [re.sub(r'\.\d+$', '', header) for header in new_header2]

# Assign the new header to the DataFrame and drop the first two rows
df_1.columns = new_header2
df_1 = df_1.drop(df.index[0])

# Reset the index
df_1.reset_index(drop=True, inplace=True)

# Set to display all columns
pd.set_option('display.max_columns', None)

# reinspect the dataframe
df_1.head(5)
```

Out[ ]:

	Unnamed: 0	2016YE_Gross claims incurred (£m)	2017YE_Gross claims incurred (£m)	2018YE_Gross claims incurred (£m)	2019YE_Gross claims incurred (£m)	2020YE_Gross claims incurred (£m)
0	Firm 1	0	0.046674	0	0	0
1	Firm 2	39.241067	35.948249	29.002244	0	0
2	Firm 3	0	0	0	0	0
3	Firm 4	17.125976	75.535951	119.426995	35.884204	6.567988
4	Firm 5	30.485185	247.872969	449.87474	348.536337	373.786832

In [ ]:

```
# for the first dataset, rename the first column "Unnamed: 0" to "Firms_ID".
df = df.rename(columns={'Unnamed: 0':'Firms_ID'})

# for the second dataset, rename the first column "Unnamed: 0" to "Firms_ID".
df_1 = df_1.rename(columns={'Unnamed: 0':'Firms_ID'})

print(df.columns)
print(df_1.columns)
```

```
Index(['Firms_ID', '2016YE_NWP (£m) ', '2017YE_NWP (£m) ', '2018YE_NWP (£m) ',
       '2019YE_NWP (£m) ', '2020YE_NWP (£m) ', '2016YE_SCR (£m)',
       '2017YE_SCR (£m)', '2018YE_SCR (£m)', '2019YE_SCR (£m)',
       '2020YE_SCR (£m)', '2016YE_EoF for SCR (£m)', '2017YE_EoF for SCR (£m)',
       '2018YE_EoF for SCR (£m)', '2019YE_EoF for SCR (£m)',
       '2020YE_EoF for SCR (£m)', '2016YE_SCR coverage ratio',
       '2017YE_SCR coverage ratio', '2018YE_SCR coverage ratio',
       '2019YE_SCR coverage ratio', '2020YE_SCR coverage ratio',
       '2016YE_GWP (£m)', '2017YE_GWP (£m)', '2018YE_GWP (£m)',
       '2019YE_GWP (£m)', '2020YE_GWP (£m)', '2016YE_Total assets (£m)',
       '2017YE_Total assets (£m)', '2018YE_Total assets (£m)',
       '2019YE_Total assets (£m)', '2020YE_Total assets (£m)',
       '2016YE_Total liabilities (£m)', '2017YE_Total liabilities (£m)',
       '2018YE_Total liabilities (£m)', '2019YE_Total liabilities (£m)',
       '2020YE_Total liabilities (£m)',
       '2016YE_Excess of assets over liabilities (£m) [= equity]',
       '2017YE_Excess of assets over liabilities (£m) [= equity]',
       '2018YE_Excess of assets over liabilities (£m) [= equity]',
       '2019YE_Excess of assets over liabilities (£m) [= equity]',
       '2020YE_Excess of assets over liabilities (£m) [= equity]'],
      dtype='object')
Index(['Firms_ID', '2016YE_Gross claims incurred (£m)',
       '2017YE_Gross claims incurred (£m)',
       '2018YE_Gross claims incurred (£m)',
       '2019YE_Gross claims incurred (£m)',
       '2020YE_Gross claims incurred (£m)',
       '2016YE_Gross BEL (inc. TPs as whole, pre-TMTP) (£m)',
       '2017YE_Gross BEL (inc. TPs as whole, pre-TMTP) (£m)',
       '2018YE_Gross BEL (inc. TPs as whole, pre-TMTP) (£m)',
       '2019YE_Gross BEL (inc. TPs as whole, pre-TMTP) (£m)',
       '2020YE_Gross BEL (inc. TPs as whole, pre-TMTP) (£m)',
       '2016YE_Net BEL (inc. TPs as a whole, pre-TMTP) (£m)',
       '2017YE_Net BEL (inc. TPs as a whole, pre-TMTP) (£m)',
       '2018YE_Net BEL (inc. TPs as a whole, pre-TMTP) (£m)',
       '2019YE_Net BEL (inc. TPs as a whole, pre-TMTP) (£m)',
       '2020YE_Net BEL (inc. TPs as a whole, pre-TMTP) (£m)',
       '2016YE_Pure net claims ratio', '2017YE_Pure net claims ratio',
       '2018YE_Pure net claims ratio', '2019YE_Pure net claims ratio',
       '2020YE_Pure net claims ratio', '2016YE_Net expense ratio',
       '2017YE_Net expense ratio', '2018YE_Net expense ratio',
       '2019YE_Net expense ratio', '2020YE_Net expense ratio',
       '2016YE_Net combined ratio', '2017YE_Net combined ratio',
       '2018YE_Net combined ratio', '2019YE_Net combined ratio',
       '2020YE_Net combined ratio', '2016YE_Pure gross claims ratio',
       '2017YE_Pure gross claims ratio', '2018YE_Pure gross claims ratio',
       '2019YE_Pure gross claims ratio', '2020YE_Pure gross claims ratio',
       '2016YE_Gross expense ratio', '2017YE_Gross expense ratio',
       '2018YE_Gross expense ratio', '2019YE_Gross expense ratio',
       '2020YE_Gross expense ratio', '2016YE_Gross combined ratio',
       '2017YE_Gross combined ratio', '2018YE_Gross combined ratio',
       '2019YE_Gross combined ratio', '2020YE_Gross combined ratio'],
      dtype='object')
```

In [ ]:

```
# as Dataset 1 and Dataset 2 have the same column index which is the 'Firm_ID',
df_c = pd.merge(df, df_1, on = "Firms_ID")
df_c.head(5)
```

Out[ ]:

	Firms_ID	2016YE_NWP (£m)	2017YE_NWP (£m)	2018YE_NWP (£m)	2019YE_NWP (£m)	2020YE_NWP (£m)	2
0	Firm 1	-13779.815629	0	0	0	0	1
1	Firm 2	28.178059	26.865049	25.064438	23.226445	21.718558	
2	Firm 3	0	75.609681	70.578732	78.432782	85.73583	
3	Firm 4	22344.199923	23963.910709	25760.390158	25512.748836	24996.021042	
4	Firm 5	68.200993	51.663132	44.010833	42.008556	81.273653	

In [ ]:

```python
# converting the data type for data columns (except 'Firm_ID') in the merged dat
cols = df_c.columns.drop('Firms_ID')
df_c[cols] = df_c[cols].astype(float)
```

In [ ]:

```python
df_c[cols]
```

Out[ ]:

	2016YE_NWP (£m)	2017YE_NWP (£m)	2018YE_NWP (£m)	2019YE_NWP (£m)	2020YE_NWP (£m)	2016YE_(
0	-13779.815629	0.000000	0.000000	0.000000	0.000000	1085.360
1	28.178059	26.865049	25.064438	23.226445	21.718558	10.190
2	0.000000	75.609681	70.578732	78.432782	85.735830	322.955
3	22344.199923	23963.910709	25760.390158	25512.748836	24996.021042	16573.644
4	68.200993	51.663132	44.010833	42.008556	81.273653	52.824
...	...	...	...	...	...	
320	0.000000	0.000000	-1.011367	-6.599067	24.632234	0.000
321	2092.156137	2084.124818	2022.212247	2103.048716	2029.697013	1711.220
322	0.000000	0.000000	0.000000	0.000000	0.000000	30.438
323	23.415380	22.650321	24.268465	25.811984	26.546638	32.096
324	240.999886	252.698937	332.521848	294.886332	0.000000	209.181

325 rows × 85 columns

In [ ]:

```python
# set 'Firms_ID' as index - this is to accuracy and easier to plot the graph
df = df.set_index('Firms_ID')
df_1 = df_1.set_index('Firms_ID')
```

```
df_c = df_c.set_index('Firms_ID')

# rewrite 'Firms_ID' remove space, and replace with underscore
df.index = df.index.str.replace(' ', '_')
df_1.index = df_1.index.str.replace(' ', '_')
df_c.index = df_c.index.str.replace(' ', '_')

# rewrite so that include firms_id as index
```

Firms Size Analysis

```
In [ ]: # combine total asset columns to a single dataframe
ta = df_c[['2016YE_Total assets (£m)','2017YE_Total assets (£m)', '2018YE_Total

# combine gross written premium columns to a single dataframe
tgwp = df_c[['2016YE_GWP (£m)','2017YE_GWP (£m)', '2018YE_GWP (£m)', '2019YE_GWP

# combine net written premium columns to a single dataframe
tnwp = df_c[['2016YE_NWP (£m) ','2017YE_NWP (£m) ', '2018YE_NWP (£m) ', '2019YE_
```

```
In [ ]: # Calculate total sum of each metrics to 2 decimal places and show general numbe
print(ta.sum().map('{:,.2f}'.format))
print(ta.sum().pct_change().map('{:.2%}'.format))
print(tgwp.sum().map('{:,.2f}'.format))
print(tgwp.sum().pct_change().map('{:.2%}'.format))
print(tnwp.sum().map('{:,.2f}'.format))
print(tnwp.sum().pct_change().map('{:.2%}'.format))
```

```
2016YE_Total assets (£m)    2,302,481.25
2017YE_Total assets (£m)    2,355,996.35
2018YE_Total assets (£m)    2,252,268.47
2019YE_Total assets (£m)    2,473,086.61
2020YE_Total assets (£m)    2,458,373.15
dtype: object
2016YE_Total assets (£m)      nan%
2017YE_Total assets (£m)     2.32%
2018YE_Total assets (£m)    -4.40%
2019YE_Total assets (£m)     9.80%
2020YE_Total assets (£m)    -0.59%
dtype: object
2016YE_GWP (£m)    273,691.18
2017YE_GWP (£m)    295,676.66
2018YE_GWP (£m)    340,196.14
2019YE_GWP (£m)    316,059.10
2020YE_GWP (£m)    269,890.43
dtype: object
2016YE_GWP (£m)       nan%
2017YE_GWP (£m)      8.03%
2018YE_GWP (£m)     15.06%
2019YE_GWP (£m)     -7.10%
2020YE_GWP (£m)    -14.61%
dtype: object
2016YE_NWP (£m)     207,222.22
2017YE_NWP (£m)     259,873.53
2018YE_NWP (£m)     274,310.07
2019YE_NWP (£m)     254,271.27
2020YE_NWP (£m)     213,963.82
dtype: object
2016YE_NWP (£m)        nan%
2017YE_NWP (£m)      25.41%
2018YE_NWP (£m)       5.56%
2019YE_NWP (£m)      -7.31%
2020YE_NWP (£m)     -15.85%
dtype: object
```

In [ ]:

```
# check for descriptive statistics for each
print(ta.describe())
print(tgwp.describe())
print(tnwp.describe())
```

```
       2016YE_Total assets (£m)  2017YE_Total assets (£m)  \
count                325.000000                325.000000
mean                7084.557700               7249.219525
std                26542.111520              30607.376781
min                    0.000000                -54.526842
25%                   26.418659                 19.791889
50%                  157.456339                137.837056
75%                 1199.598522               1091.167855
max               284329.904525             298172.843043

       2018YE_Total assets (£m)  2019YE_Total assets (£m)  \
count                325.000000                325.000000
mean                6930.056846               7609.497261
std                28996.531235              32341.403195
min                 -161.954321               -217.429787
25%                   17.780399                  8.467582
50%                  120.712705                104.938152
75%                 1008.409821                959.515597
max               282829.545364             311228.369410

       2020YE_Total assets (£m)
count                325.000000
mean                7564.225088
std                33396.056313
min                  -83.997561
25%                    0.000000
50%                   57.509809
75%                  678.317542
max               332875.903638
       2016YE_GWP (£m)  2017YE_GWP (£m)  2018YE_GWP (£m)  2019YE_GWP (£m)  \
count       325.000000       325.000000       325.000000       325.000000
mean        842.126715       909.774328      1046.757353       972.489525
std        3980.798811      3545.771115      4163.304896      4008.175641
min         -13.873442        -4.948002        -7.917129        -0.151805
25%           0.000000         0.000000         0.000000         0.000000
50%          10.940987        19.812373        19.623853        13.355539
75%         200.808629       263.818789       264.539561       219.977537
max       45309.819760     38199.311256     48117.993733     44638.769640

       2020YE_GWP (£m)
count       325.000000
mean        830.432091
std        3600.528585
min         -95.424434
25%           0.000000
50%           6.205218
75%         166.926417
max       40135.692258
       2016YE_NWP (£m)   2017YE_NWP (£m)   2018YE_NWP (£m)   2019YE_NWP (£m)   \
count        325.000000        325.000000        325.000000        325.000000
mean         637.606837        799.610849        844.030970        782.373137
std         3858.919674       3287.474920       3687.130996       3453.361810
min       -13779.815629      -2305.854316       -193.083319       -181.612136
25%           0.000000          0.000000          0.000000          0.000000
50%           4.390134         10.742034         10.160408          6.581324
75%         104.181634        190.786032        180.646731        106.600218
max       45309.838702      38199.311256      48117.993733      44638.769640

       2020YE_NWP (£m)
count        325.000000
```

```
mean        658.350221
std        3036.308650
min       -1336.553317
25%           0.000000
50%           2.742672
75%          64.369695
max       40135.692258
```

In [ ]:
```
# create a table for total asset, sorted by 2020YE_Total assets (£m) in descendi
ta = ta.sort_values(by = '2020YE_Total assets (£m)', ascending = False)
ta.head(10).style.set_sticky(axis="index").background_gradient(cmap='Blues').for
```

Out[ ]:

Firms_ID	2016YE_Total assets (£m)	2017YE_Total assets (£m)	2018YE_Total assets (£m)	2019YE_Total assets (£m)	2020YE_Total assets (£m)
Firm_210	284,329.90	298,172.84	282,829.55	311,228.37	332,875.90
Firm_311	139,095.39	295,913.90	280,664.42	302,099.26	321,563.60
Firm_105	176,860.53	189,105.53	177,363.07	190,804.88	190,431.21
Firm_34	123,131.66	133,659.76	137,198.17	177,652.79	185,108.33
Firm_10	155,205.16	155,343.13	136,914.32	138,456.98	142,144.99
Firm_7	81,043.29	88,006.34	87,435.58	101,517.06	110,371.66
Firm_4	67,404.35	73,034.63	81,184.83	84,801.52	94,065.08
Firm_73	36,769.52	35,482.37	46,223.79	70,743.36	89,412.17
Firm_199	47,564.38	57,192.48	60,263.02	74,546.83	83,298.94
Firm_151	63,229.58	56,191.57	66,160.04	77,101.97	82,397.81

In [ ]:
```
# create a table for gross written premium, sorted by 2020YE_GWP (£m) in descend
tgwp = tgwp.sort_values(by = '2020YE_GWP (£m)', ascending = False)
tgwp.head(10).style.background_gradient(cmap='Blues').format("{:,.2f}") #format
```

Out[ ]:

Firms_ID	2016YE_GWP (£m)	2017YE_GWP (£m)	2018YE_GWP (£m)	2019YE_GWP (£m)	2020YE_GWP (£m)
Firm_210	27,889.34	38,199.31	48,117.99	44,638.77	40,135.69
Firm_4	29,424.57	32,935.40	35,867.64	36,135.46	34,922.70
Firm_34	8,772.50	12,550.06	17,214.59	20,729.17	20,510.75
Firm_311	2,210.97	11,493.62	16,553.86	18,988.45	19,180.02
Firm_26	45,309.82	7,239.36	7,616.76	10,450.18	10,489.25
Firm_247	13,589.81	24,202.65	22,694.94	10,624.48	9,961.52
Firm_199	28.30	8,808.03	10,211.52	9,756.35	9,149.58
Firm_7	6,567.62	9,542.90	9,662.75	11,259.29	8,652.95
Firm_151	7,753.57	8,125.44	8,076.39	9,188.80	8,341.64
Firm_10	1,317.98	10,559.90	10,122.41	8,960.73	7,923.37

In [ ]:

```
# create a table for net written premium, sorted by 2020YE_NWP (£m) in descendin
tnwp = tnwp.sort_values(by = '2020YE_NWP (£m) ', ascending = False)
tnwp.head(10).style.background_gradient(cmap='Blues').format("{:,.2f}") #format
```

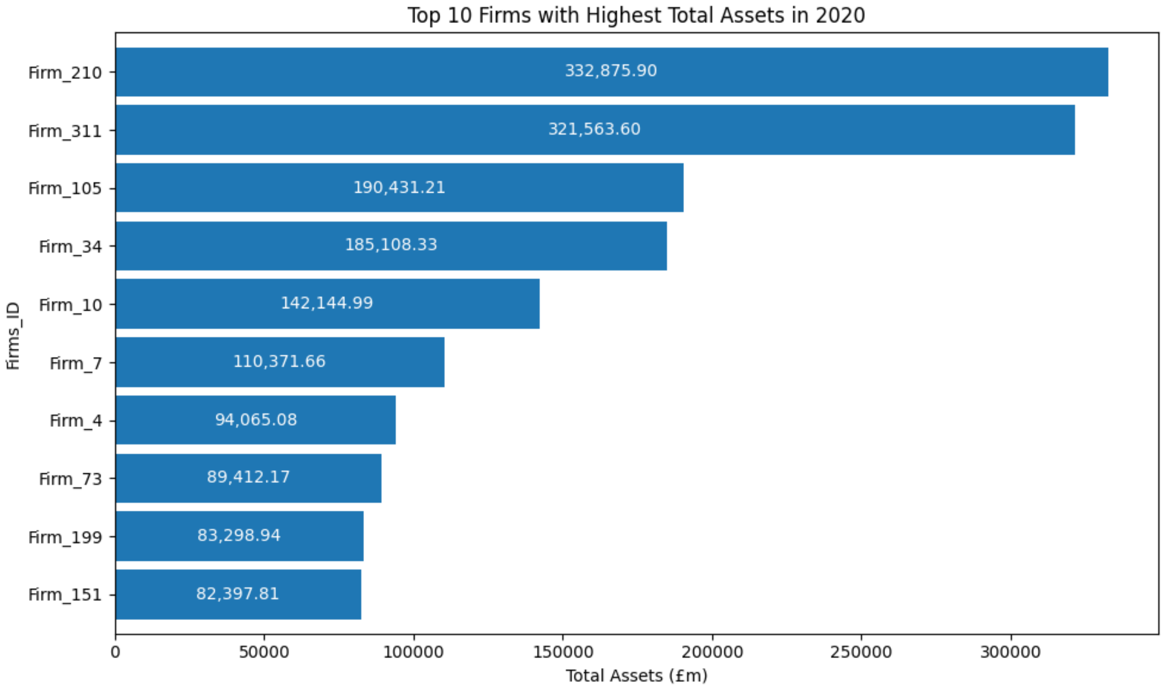
Out[ ]:

Firms_ID	2016YE_NWP (£m)	2017YE_NWP (£m)	2018YE_NWP (£m)	2019YE_NWP (£m)	2020YE_NWP (£m)
Firm_210	27,889.34	38,199.31	48,117.99	44,638.77	40,135.69
Firm_4	22,344.20	23,963.91	25,760.39	25,512.75	24,996.02
Firm_311	-1,862.24	9,777.53	12,009.16	12,719.40	10,830.97
Firm_26	45,309.84	7,239.36	7,616.76	10,450.18	10,489.25
Firm_247	13,377.53	24,031.38	22,475.77	10,624.48	9,961.52
Firm_199	6.77	8,787.82	10,191.58	9,739.14	9,134.28
Firm_7	5,855.17	11,688.57	9,414.98	10,975.19	8,359.91
Firm_151	-1,750.95	7,626.42	7,867.16	9,028.41	8,180.39
Firm_34	6,817.40	5,780.78	4,497.98	-181.61	8,145.62
Firm_10	1,273.95	10,516.21	10,087.57	8,921.88	7,893.06

In [ ]:

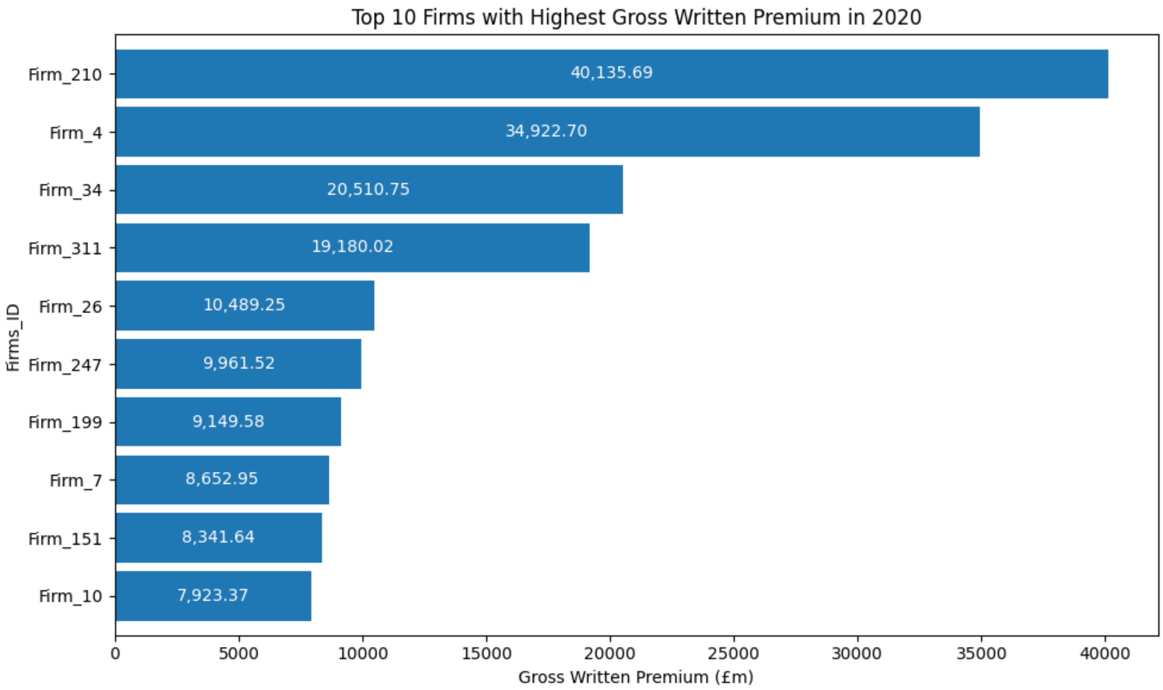
```
# vizualize the total asset data using horizontal bar chart, display top 10 firm
ta_2020 = ta['2020YE_Total assets (£m)'].sort_values(ascending=False).head(10)

# highest horizontal bar chart at the top
ax = ta_2020.plot(kind='barh', figsize=(10, 6), zorder=2, width=0.85)
plt.xlabel('Total Assets (£m)')
plt.ylabel('Firms_ID')
plt.title('Top 10 Firms with Highest Total Assets in 2020')
plt.tight_layout()
```

```
ax.invert_yaxis()
ax.bar_label(ax.containers[0], label_type='center', color='white', fontsize=10,
plt.show()
```



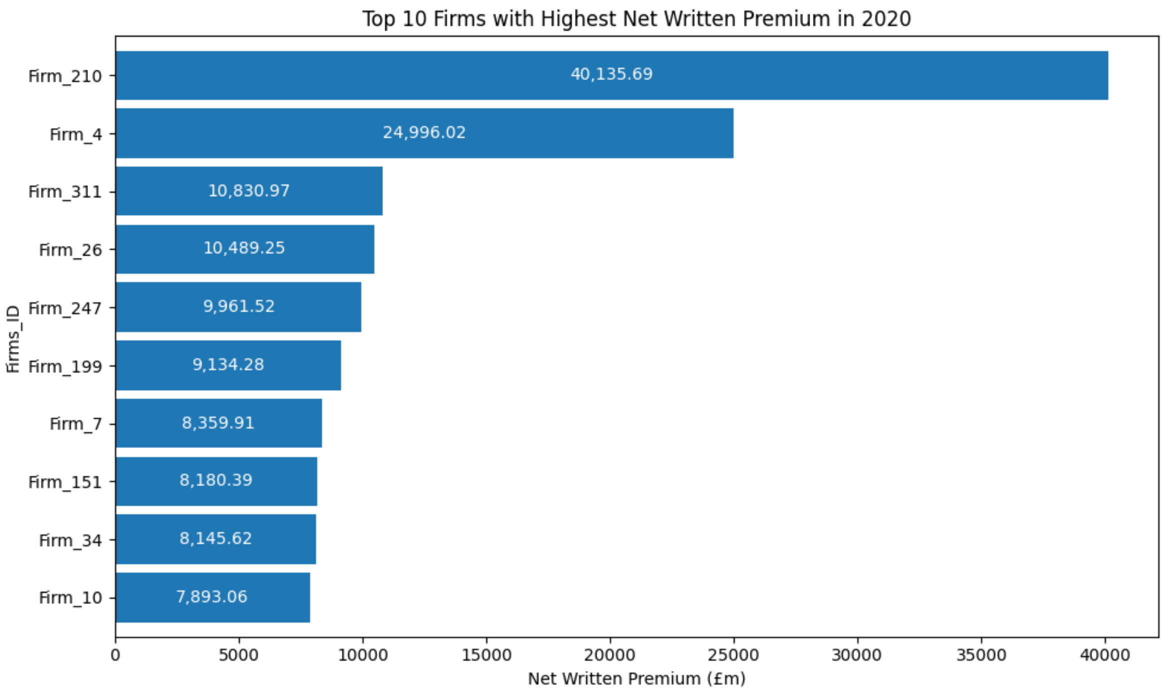
In [ ]:

```
# vizualize the gross written premium data using horizontal bar chart, display t
tgwp_2020 = tgwp['2020YE_GWP (£m)'].sort_values(ascending=False).head(10)

# highest horizontal bar chart at the top
ax = tgwp_2020.plot(kind='barh', figsize=(10, 6), zorder=2, width=0.85)
plt.xlabel('Gross Written Premium (£m)')
plt.ylabel('Firms_ID')
plt.title('Top 10 Firms with Highest Gross Written Premium in 2020')
plt.tight_layout()
ax.invert_yaxis()
ax.bar_label(ax.containers[0], label_type='center', color='white', fontsize=10,
plt.show()
```



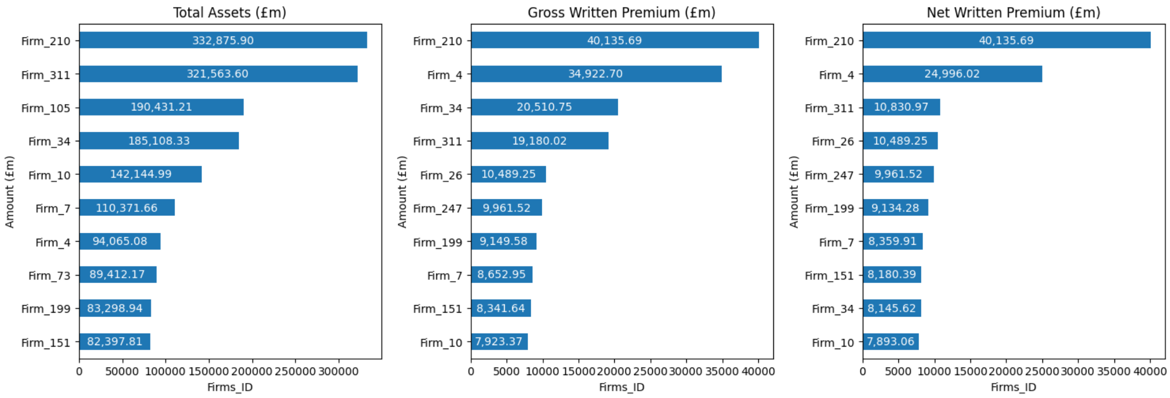
In [ ]:

```
# vizualize the net written premium data using horizontal bar chart, display top
tnwp_2020 = tnwp['2020YE_NWP (£m) '].sort_values(ascending=False).head(10)

# highest horizontal bar chart at the top
ax = tnwp_2020.plot(kind='barh', figsize=(10, 6), zorder=2, width=0.85)
plt.xlabel('Net Written Premium (£m)')
plt.ylabel('Firms_ID')
plt.title('Top 10 Firms with Highest Net Written Premium in 2020')
plt.tight_layout()
ax.invert_yaxis()
ax.bar_label(ax.containers[0], label_type='center', color='white', fontsize=10,
plt.show()
```



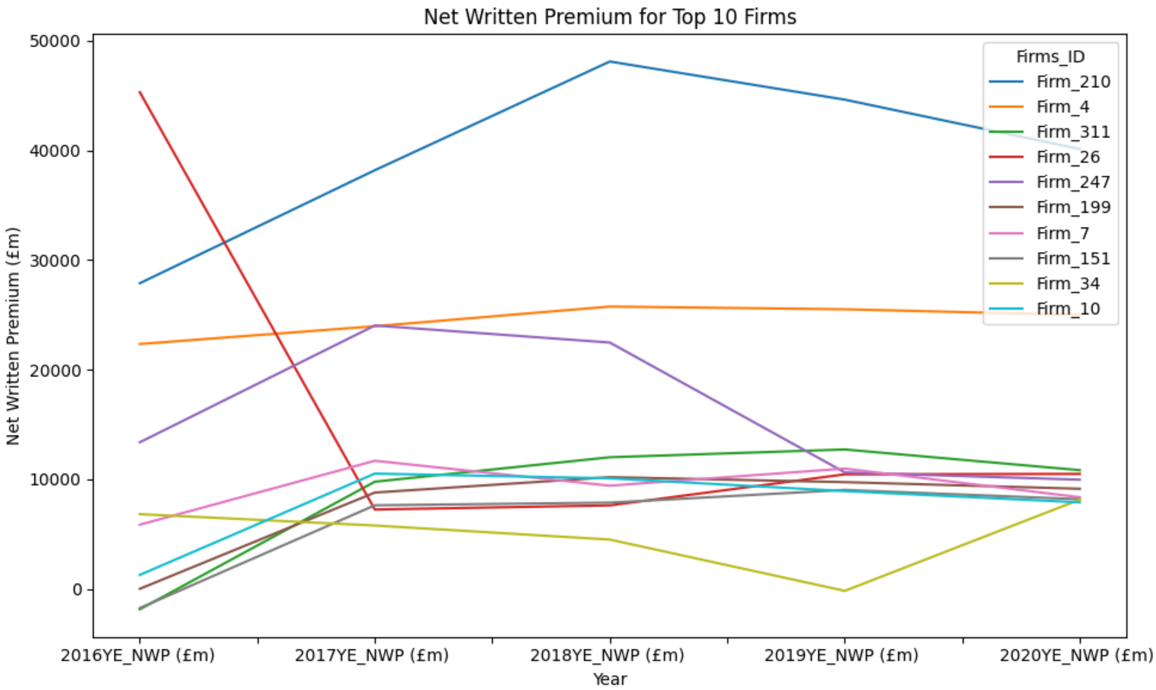
In [ ]:

```
# put all ta, tgwp and tnwp horizontal bar into one plot, inver
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15, 5))
ta_2020.plot(kind='barh', ax=axes[0], title='Total Assets (£m)')
tgwp_2020.plot(kind='barh', ax=axes[1], title='Gross Written Premium (£m)')
tnwp_2020.plot(kind='barh', ax=axes[2], title='Net Written Premium (£m)')
plt.tight_layout()
#invert all y axis for all plots
for ax in axes:
    ax.invert_yaxis()
    ax.set_xlabel('Firms_ID')
    ax.set_ylabel('Amount (£m)')
# add label to all bar, with thousand separator and 2 decimal places, inside the
for ax in axes:
    ax.bar_label(ax.containers[0], label_type='center', color='white', fontsize=
plt.show()
```



In [ ]:

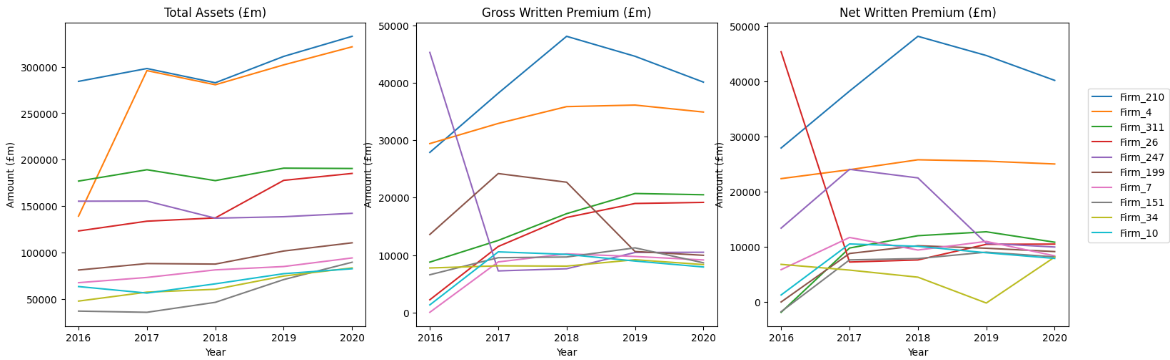
```python
# create a line chart for tnwp(10)
tnwp_10 = tnwp.head(10)
tnwp_10 = tnwp_10.T
tnwp_10.plot(kind='line', figsize=(10, 6))
plt.xlabel('Year')
plt.ylabel('Net Written Premium (£m)')
plt.title('Net Written Premium for Top 10 Firms')
plt.tight_layout()
plt.show()
```



In [ ]:

```python
#modify each column name to remove the 'YE' and 'Total assets (£m)', 'GWP (£m)',
#this is to make the graph more readable
#rename the column name without chaning the original dataframe
ta = ta.rename(columns={'2016YE_Total assets (£m)':'2016', '2017YE_Total assets
tgwp = tgwp.rename(columns={'2016YE_GWP (£m)':'2016', '2017YE_GWP (£m)':'2017',
tnwp = tnwp.rename(columns={'2016YE_NWP (£m) ':'2016', '2017YE_NWP (£m) ':'2017'

# create a line chart for tnwp(10), ta(10) and tgwp(10) in one plot
tnwp_10 = tnwp.head(10)
ta_10 = ta.head(10)
tgwp_10 = tgwp.head(10)
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15, 5))
tnwp_10 = tnwp_10.T
ta_10 = ta_10.T
tgwp_10 = tgwp_10.T
```

```
ta_10.plot(kind='line', ax=axes[0], title='Total Assets (£m)', legend=False)
tgwp_10.plot(kind='line', ax=axes[1], title='Gross Written Premium (£m)', legend
tnwp_10.plot(kind='line', ax=axes[2], title='Net Written Premium (£m)', legend=F
# add legend to only one plot, with padding and not overlapping with the plot
plt.tight_layout()
#invert all y axis for all plots
for ax in axes:
    ax.set_xlabel('Year')
    ax.set_ylabel('Amount (£m)')
    axes[2].legend(loc='upper center', bbox_to_anchor=(1.2, 0.8), ncol=1)
plt.show()
```



In [ ]:

```
# calculate firms concentration by total asset, for every period
for col in ta.columns:
    total = ta[col].sum()
    ta[col + '(%)'] = round((ta[col] / total) * 100, 2)

ta_sort = ta.iloc[:,5:10].sort_values(by=['2020(%)'], ascending=False)
ta_sort.head(10).style.background_gradient(cmap='Blues').format("{:,.2f}")
```

Out[ ]:

	2016(%)	2017(%)	2018(%)	2019(%)	2020(%)
Firms_ID					
Firm_210	12.35	12.66	12.56	12.58	13.54
Firm_311	6.04	12.56	12.46	12.22	13.08
Firm_105	7.68	8.03	7.87	7.72	7.75
Firm_34	5.35	5.67	6.09	7.18	7.53
Firm_10	6.74	6.59	6.08	5.60	5.78
Firm_7	3.52	3.74	3.88	4.10	4.49
Firm_4	2.93	3.10	3.60	3.43	3.83
Firm_73	1.60	1.51	2.05	2.86	3.64
Firm_199	2.07	2.43	2.68	3.01	3.39
Firm_151	2.75	2.39	2.94	3.12	3.35

In [ ]:

```
# calculate firms concentration by gross written premium, for every period
for col in tgwp.columns:
    total = tgwp[col].sum()
    tgwp[col + '(%)'] = round((tgwp[col] / total) * 100, 2)
```

```
tgwp_sort = tgwp.iloc[:,5:10].sort_values(by=['2020(%)'], ascending=False)
tgwp_sort.head(10).style.background_gradient(cmap='Blues').format("{:,.2f}")
```

Out[ ]:

Firms_ID	2016(%)	2017(%)	2018(%)	2019(%)	2020(%)
Firm_210	10.19	12.92	14.14	14.12	14.87
Firm_4	10.75	11.14	10.54	11.43	12.94
Firm_34	3.21	4.24	5.06	6.56	7.60
Firm_311	0.81	3.89	4.87	6.01	7.11
Firm_26	16.56	2.45	2.24	3.31	3.89
Firm_247	4.97	8.19	6.67	3.36	3.69
Firm_199	0.01	2.98	3.00	3.09	3.39
Firm_7	2.40	3.23	2.84	3.56	3.21
Firm_151	2.83	2.75	2.37	2.91	3.09
Firm_10	0.48	3.57	2.98	2.84	2.94

In [ ]:

```
# calculate firms concentration by net written premium, for every period
for col in tnwp.columns:
    total = tnwp[col].sum()
    tnwp[col + '(%)'] = round((tnwp[col] / total) * 100, 2)


tnwp_sort = tnwp.iloc[:,5:10].sort_values(by=['2020(%)'], ascending=False)
tnwp_sort.head(10).style.background_gradient(cmap='Blues').format("{:,.2f}")
```

Out[ ]:

Firms_ID	2016(%)	2017(%)	2018(%)	2019(%)	2020(%)
Firm_210	13.46	14.70	17.54	17.56	18.76
Firm_4	10.78	9.22	9.39	10.03	11.68
Firm_311	-0.90	3.76	4.38	5.00	5.06
Firm_26	21.87	2.79	2.78	4.11	4.90
Firm_247	6.46	9.25	8.19	4.18	4.66
Firm_199	0.00	3.38	3.72	3.83	4.27
Firm_7	2.83	4.50	3.43	4.32	3.91
Firm_151	-0.84	2.93	2.87	3.55	3.82
Firm_34	3.29	2.22	1.64	-0.07	3.81
Firm_10	0.61	4.05	3.68	3.51	3.69

In [ ]:

```
tnwp_tgwp_ratio = (tnwp.iloc[:,0:5] / tgwp.iloc[:,0:5]).sort_values(by=['2020'],
tnwp_tgwp_ratio.head(10).style.background_gradient(cmap='Blues').format("{:,.2f}
```

```
c:\Users\Syarmine\AppData\Local\Programs\Python\Python312\Lib\site-packages\panda
s\io\formats\style.py:3823: RuntimeWarning: invalid value encountered in scalar m
ultiply
  norm = _matplotlib.colors.Normalize(smin - (rng * low), smax + (rng * high))
```

Out[ ]:

Firms_ID	2016	2017	2018	2019	2020
Firm_21	nan	nan	nan	nan	inf
Firm_303	-0.77	-26.24	1.08	6.22	6.82
Firm_152	0.78	0.46	0.68	0.50	2.63
Firm_76	0.87	1.26	1.26	1.24	1.35
Firm_61	0.74	1.06	0.99	-2.67	1.12
Firm_28	0.97	1.02	1.41	1.00	1.00
Firm_226	1.00	1.00	1.00	1.00	1.00
Firm_265	1.00	1.00	1.00	1.00	1.00
Firm_26	1.00	1.00	1.00	1.00	1.00
Firm_25	1.00	1.00	1.00	1.00	1.00

Changing Business Profile

In [ ]:

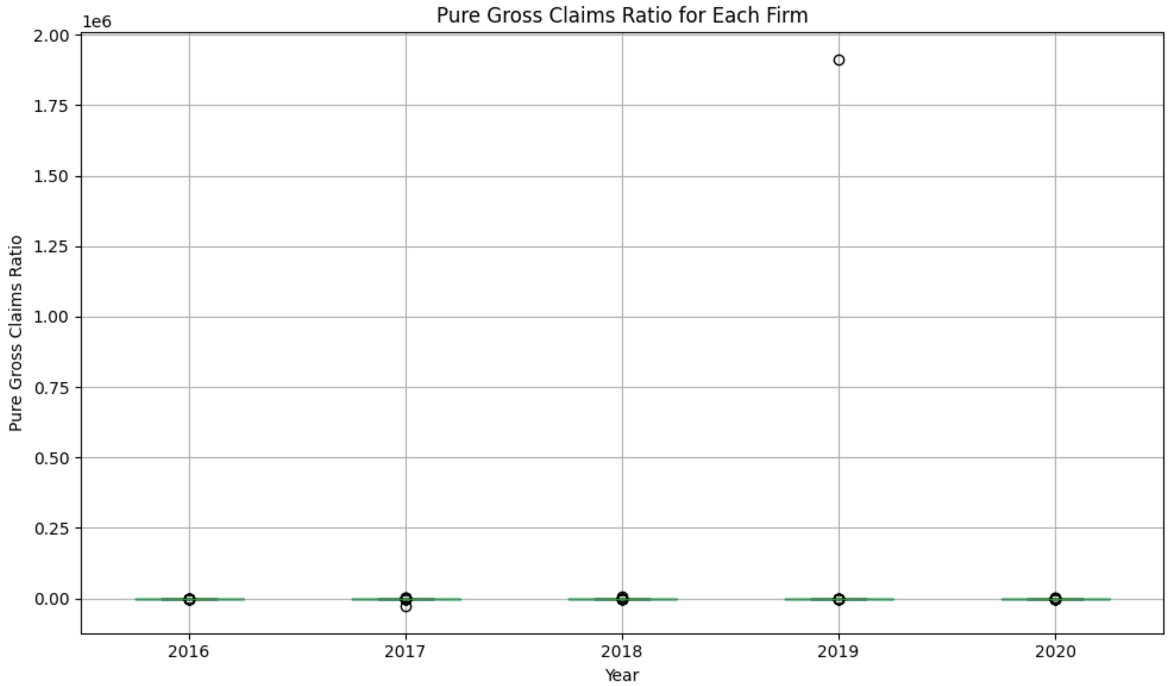
```
# selecting eligible own funds
teof = df_c[['2016YE_EoF for SCR (£m)','2017YE_EoF for SCR (£m)', '2018YE_EoF fo

# selecting SCR
tscr = df_c[['2016YE_SCR (£m)','2017YE_SCR (£m)', '2018YE_SCR (£m)', '2019YE_SCR

# selecting SCR coverage ratio
tscrcr = df_c[['2016YE_SCR coverage ratio','2017YE_SCR coverage ratio', '2018YE_

# selecting gross claims incurred
tgci = df_c[['2016YE_Gross claims incurred (£m)','2017YE_Gross claims incurred (

# selecting gross BEL
tgbel = df_c[['2016YE_Gross BEL (inc. TPs as whole, pre-TMTP) (£m)','2017YE_Gros

# selecting net BEL
tnbel = df_c[['2016YE_Net BEL (inc. TPs as a whole, pre-TMTP) (£m)','2017YE_Net

# selecting pure net claims ratio
tpncr = df_c[['2016YE_Pure net claims ratio','2017YE_Pure net claims ratio', '20

df_c['2016YE_Net expense ratio']
# selecting net expense ratio
tnexr = df_c[['2016YE_Net expense ratio','2017YE_Net expense ratio', '2018YE_Net

# selecting net combined ratio
tncri = df_c[['2016YE_Net combined ratio','2017YE_Net combined ratio', '2018YE_N

# selecting pure gross claims ratio
tpgcr = df_c[['2016YE_Pure gross claims ratio','2017YE_Pure gross claims ratio',
```

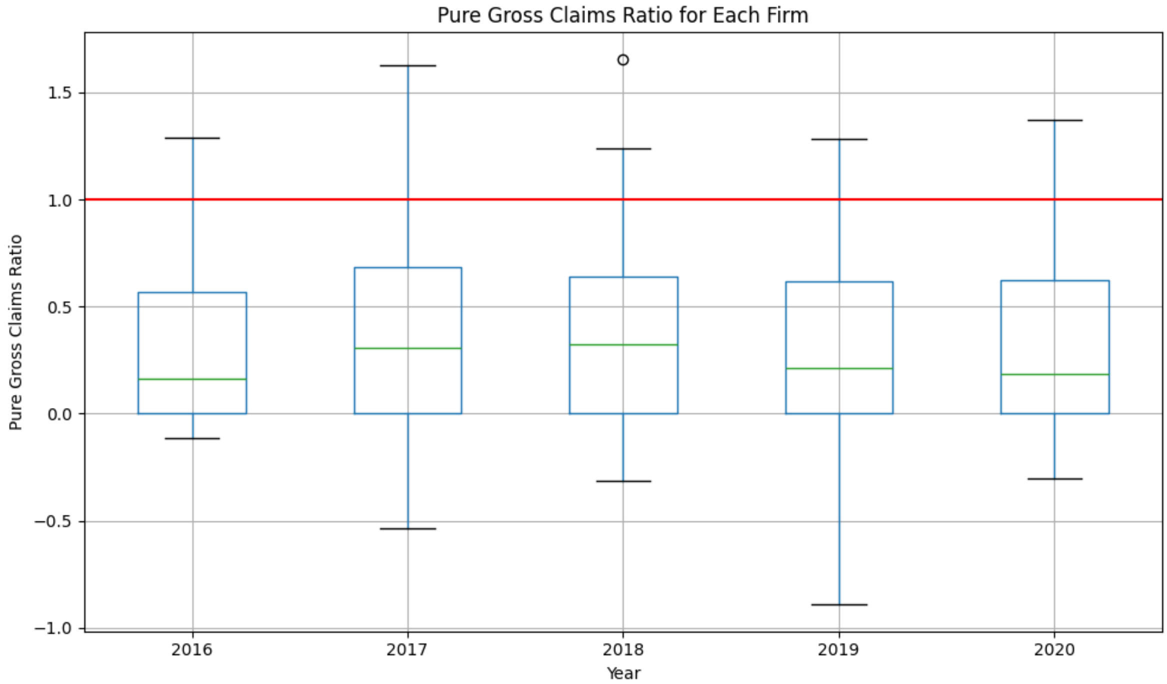
```
# selecting gross expense ratio
tgexr = df_c[['2016YE_Gross expense ratio','2017YE_Gross expense ratio', '2018YE

# selecting gross combined ratio
tgcri = df_c[['2016YE_Gross combined ratio','2017YE_Gross combined ratio', '2018

#rename each column
teof = teof.rename(columns={'2016YE_EoF for SCR (£m)':'2016', '2017YE_EoF for SC
tscr = tscr.rename(columns={'2016YE_SCR (£m)':'2016', '2017YE_SCR (£m)':'2017',
tscrcr = tscrcr.rename(columns={'2016YE_SCR coverage ratio':'2016', '2017YE_SCR
tgci = tgci.rename(columns={'2016YE_Gross claims incurred (£m)':'2016', '2017YE_
tgbel = tgbel.rename(columns={'2016YE_Gross BEL (inc. TPs as whole, pre-TMTP) (£
tnbel = tnbel.rename(columns={'2016YE_Net BEL (inc. TPs as a whole, pre-TMTP) (£
tpncr = tpncr.rename(columns={'2016YE_Pure net claims ratio':'2016', '2017YE_Pur
tnexr = tnexr.rename(columns={'2016YE_Net expense ratio':'2016', '2017YE_Net exp
tncri = tncri.rename(columns={'2016YE_Net combined ratio':'2016', '2017YE_Net co
tpgcr = tpgcr.rename(columns={'2016YE_Pure gross claims ratio':'2016', '2017YE_P
tgexr = tgexr.rename(columns={'2016YE_Gross expense ratio':'2016', '2017YE_Gross
tgcri = tgcri.rename(columns={'2016YE_Gross combined ratio':'2016', '2017YE_Gros
```

In [ ]:

```
# calculate pure gross claims ratio for every period for each firm using boxplot
print(tpgcr.describe())
tpgcr.boxplot(figsize=(10,6))
plt.xlabel('Year')
plt.ylabel('Pure Gross Claims Ratio')
plt.title('Pure Gross Claims Ratio for Each Firm')
plt.tight_layout()
plt.show()

# create a variable outlier dataframe to allow return of the outlier in a datafr
q1 = tpgcr.quantile(0.25)
q3 = tpgcr.quantile(0.75)
iqr = q3 - q1
tpgcr_outlier = ((tpgcr < (q1 - 1.5 * iqr)) | (tpgcr > (q3 + 1.5 * iqr))).any(ax

# new variable without outliers to improve readability of the boxplot
tpgcr_mod = tpgcr[~((tpgcr < (q1 - 1.5 * iqr)) | (tpgcr > (q3 + 1.5 * iqr))).any

# create a box plot of tpgcr
# add a reference line of 100% or at 1.0, this ratio should be less than this, i
# if more than 100%, there is sign of risk  mispricing and an incentive to inves
tpgcr_mod.boxplot(figsize=(10,6))
plt.axhline(y=1.0, color='r', linestyle='-')
plt.xlabel('Year')
plt.ylabel('Pure Gross Claims Ratio')
plt.title('Pure Gross Claims Ratio for Each Firm')
plt.tight_layout()
plt.show()
```

```
             2016          2017         2018          2019         2020
count  325.000000    325.000000   325.000000  3.250000e+02   325.000000
mean    -0.110588    -72.376536    26.204866  5.872168e+03     9.801827
std     31.446878   1438.129350   461.725941  1.060098e+05    94.435270
min   -387.557255 -25876.266020  -160.128397 -2.893406e+03   -21.021266
25%      0.000000      0.000000     0.000000  0.000000e+00     0.000000
50%      0.162931      0.310568     0.323024  1.962327e-01     0.000000
75%      0.578166      0.698855     0.663285  6.300366e-01     0.627148
max    321.359543    899.702268  8321.272846  1.911108e+06  1165.884916
```





In [ ]:
```
# sort the pure gross claims (before cleaning) ratio by 2020 in descending order
tpgcr_sort = tpgcr.sort_values(by=['2020'], ascending=False)
tpgcr_sort.head(10).style.background_gradient(cmap='Blues').format("{:,.2f}")
```

Out[ ]:

Firms_ID	2016	2017	2018	2019	2020
Firm_270	-387.56	-109.74	-160.13	-2,893.41	1,165.88
Firm_99	0.00	-25,876.27	111.68	1.42	1,132.42
Firm_284	0.00	0.00	0.00	183.25	423.54
Firm_72	116.15	899.70	113.70	245.58	301.33
Firm_166	22.00	-0.04	40.85	-214.80	90.21
Firm_146	0.86	-0.93	1.42	-3.85	4.76
Firm_205	0.00	0.58	0.44	0.94	2.68
Firm_214	0.00	0.81	0.51	0.55	2.19
Firm_163	0.56	0.71	0.66	0.63	1.80
Firm_21	0.60	0.58	0.43	0.16	1.77

In [ ]:

```
# sort the pure gross claims (after cleaning) ratio by 2020 in descending order
print(tpgcr_mod.describe())
tpgcr_mod_sort = tpgcr_mod.sort_values(by=['2020'], ascending=False)
tpgcr_mod_sort.head(10).style.background_gradient(cmap='Blues').format("{:,.2f}"

#to calculate use shape
#tpgcr_mod_sort.shape = 285, 5
```

```
             2016        2017        2018        2019        2020
count  285.000000  285.000000  285.000000  285.000000  285.000000
mean     0.295359    0.368160    0.342712    0.309643    0.323780
std      0.326850    0.391340    0.349080    0.341326    0.357707
min     -0.113431   -0.536276   -0.315361   -0.889447   -0.301601
25%      0.000000    0.000000    0.000000    0.000000    0.000000
50%      0.162931    0.305848    0.321162    0.213447    0.183942
75%      0.567755    0.685976    0.641764    0.619649    0.622522
max      1.286180    1.626692    1.652928    1.282594    1.373235
```

Out[ ]:

Firms_ID	2016	2017	2018	2019	2020
Firm_144	0.66	1.04	0.63	0.78	1.37
Firm_29	0.71	1.63	0.81	0.00	1.34
Firm_91	0.00	0.65	0.94	0.78	1.30
Firm_319	0.35	1.21	0.25	0.04	1.13
Firm_194	0.53	0.64	0.34	0.60	1.09
Firm_206	0.31	0.85	0.71	0.46	1.09
Firm_239	1.29	0.64	0.68	0.58	1.06
Firm_52	0.53	0.85	0.74	0.62	1.05
Firm_160	0.60	0.72	0.60	0.70	1.04
Firm_294	0.78	0.53	0.72	1.28	1.04

In [ ]:

```python
# calculate YoY of pure gross claims ratio of tpgcr_mod_sort all years
tpgcr_mod_yoy = tpgcr_mod_sort.pct_change(axis='columns').mul(100).round(2)
tpgcr_mod_yoy.head(10).style.background_gradient(cmap='Blues').format("{:,.2f}")
```

```
c:\Users\Syarmine\AppData\Local\Programs\Python\Python312\Lib\site-packages\panda
s\io\formats\style.py:3819: RuntimeWarning: All-NaN slice encountered
  smin = np.nanmin(gmap) if vmin is None else vmin
c:\Users\Syarmine\AppData\Local\Programs\Python\Python312\Lib\site-packages\panda
s\io\formats\style.py:3820: RuntimeWarning: All-NaN slice encountered
  smax = np.nanmax(gmap) if vmax is None else vmax
c:\Users\Syarmine\AppData\Local\Programs\Python\Python312\Lib\site-packages\panda
s\io\formats\style.py:3823: RuntimeWarning: invalid value encountered in scalar m
ultiply
  norm = _matplotlib.colors.Normalize(smin - (rng * low), smax + (rng * high))
```

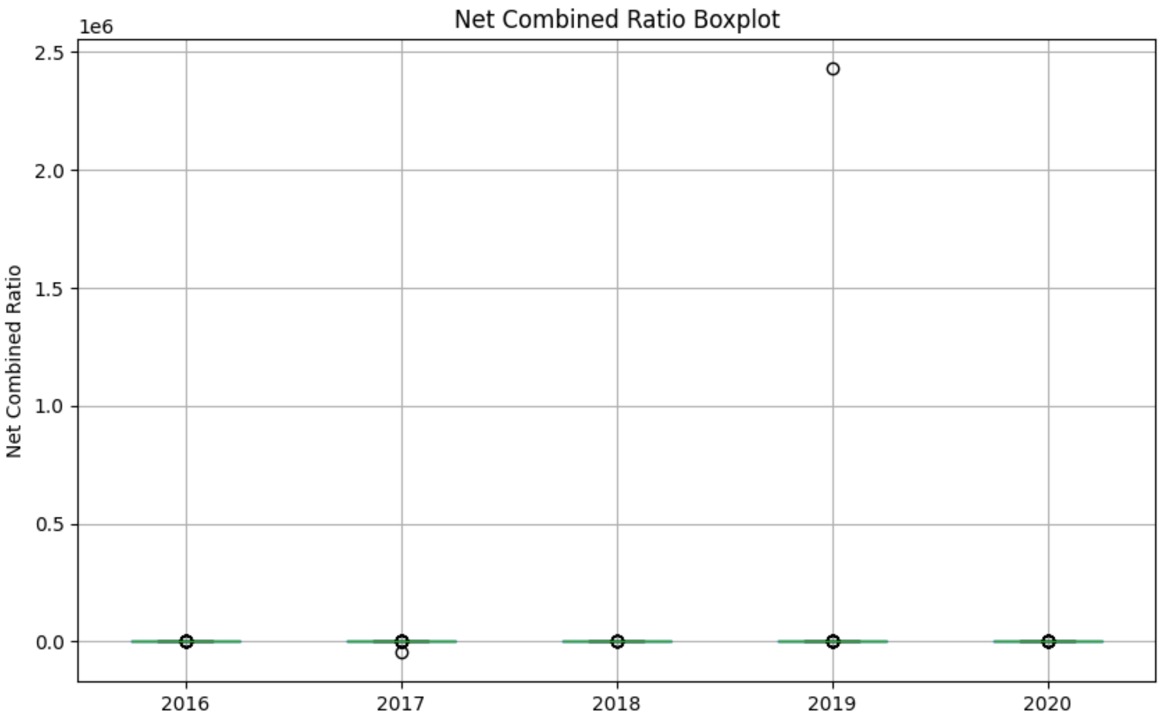
Out[ ]:

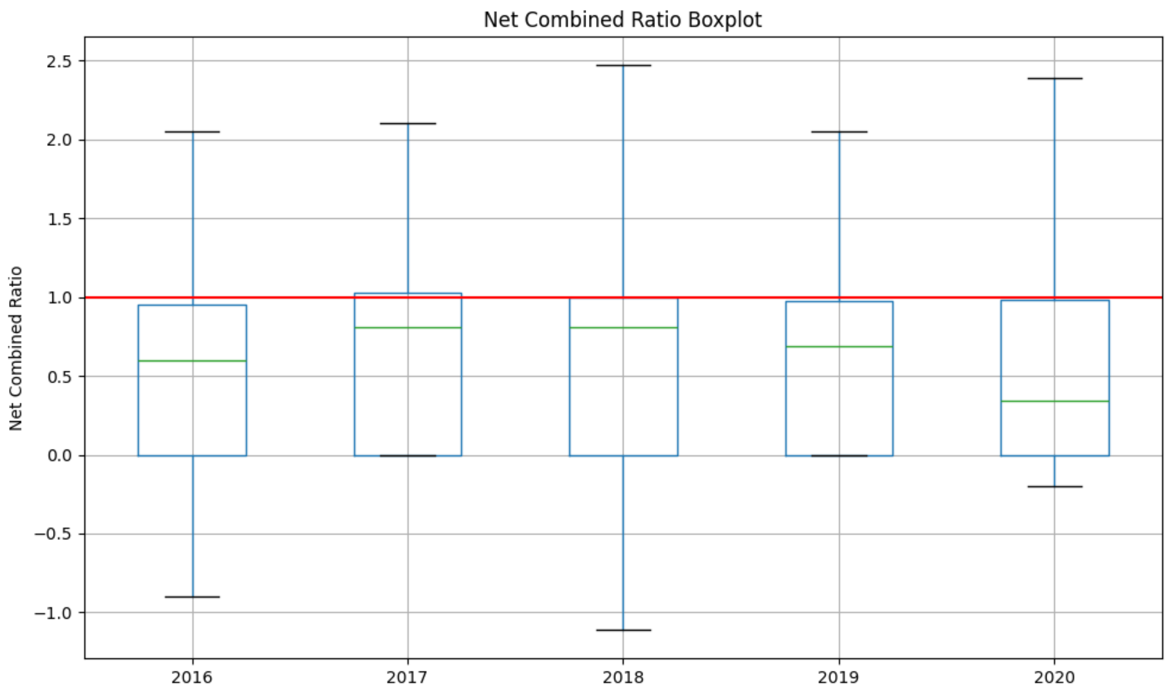
Firms_ID	2016	2017	2018	2019	2020
Firm_144	nan	56.67	-39.43	24.02	75.82
Firm_29	nan	130.40	-50.09	-99.96	401,486.63
Firm_91	nan	inf	44.25	-16.54	65.41
Firm_319	nan	249.94	-79.22	-84.35	2,771.05
Firm_194	nan	19.70	-46.64	77.06	81.18
Firm_206	nan	176.35	-16.93	-34.47	134.12
Firm_239	nan	-50.03	5.63	-14.18	81.11
Firm_52	nan	61.46	-13.22	-16.16	70.45
Firm_160	nan	18.79	-16.56	16.34	50.04
Firm_294	nan	-32.43	36.04	78.26	-18.85

In [ ]:

```
# create a box plot of net combined ratio for each firm
print(tncri.describe())
tncri.boxplot(figsize=(10, 6))
plt.ylabel('Net Combined Ratio')
plt.title('Net Combined Ratio Boxplot')
plt.show()

# create a variable outlier dataframe to allow return of the outlier in a datafr
q1 = tncri.quantile(0.25)
q3 = tncri.quantile(0.75)
iqr = q3 - q1
tncri_outlier = ((tncri < (q1 - 1.5 * iqr)) | (tncri > (q3 + 1.5 * iqr))).any(ax

# new variable without outliers to improve readability of the boxplot
tncri_mod = tncri[~((tncri < (q1 - 1.5 * iqr)) | (tncri > (q3 + 1.5 * iqr))).any

# create a box plot of tncri
# add a reference line of 100% or at 1.0
tncri_mod.boxplot(figsize=(10, 6))
plt.axhline(y=1.0, color='r', linestyle='-')
plt.ylabel('Net Combined Ratio')
plt.title('Net Combined Ratio Boxplot')
plt.tight_layout()
plt.show()
```

```
             2016          2017         2018          2019         2020
count  325.000000    325.000000   325.000000  3.250000e+02   325.000000
mean     1.349348   -133.048986     5.986255  7.477469e+03     9.853370
std     15.126252   2560.806827    87.892041  1.347934e+05    95.090048
min   -124.288370 -46116.696842    -2.013885 -2.516977e+02    -9.490844
25%      0.000000      0.000000     0.000000  0.000000e+00     0.000000
50%      0.327623      0.732771     0.793606  6.178107e-01     0.227627
75%      0.959809      1.041880     1.005938  9.751001e-01     0.988144
max    198.007110   1738.295847  1578.853894  2.430023e+06  1076.158703
```





In [ ]:

```
# sort the net combined ratio (before cleaning - without outlier) ratio by 2020
tncri_sort = tncri.sort_values(by=['2020'], ascending=False)
tncri_sort.head(10).style.background_gradient(cmap='Blues').format("{:,.2f}")
```

Out[ ]:

Firms_ID	2016	2017	2018	2019	2020
Firm_228	0.00	0.00	0.00	0.00	1,076.16
Firm_166	101.72	-2.44	2.41	-248.63	989.16
Firm_284	0.00	0.00	0.00	435.58	906.31
Firm_72	67.33	-20.54	144.00	48.63	49.51
Firm_178	0.00	0.00	0.00	0.00	21.00
Firm_39	0.00	0.00	0.00	33.18	5.77
Firm_146	1.06	-1.49	1.12	-32.86	5.45
Firm_88	0.00	3.35	13.94	6.72	4.06
Firm_203	0.94	1.10	1.19	0.85	3.12
Firm_97	0.99	1.06	1.01	1.00	2.39

In [ ]:

```
# sort the net combined ratio (after cleaning - without outlier) ratio by 2020 i
print(tncri_mod.describe())
tncri_mod_sort = tncri_mod.sort_values(by=['2020'], ascending=False)
tncri_mod_sort.head(10).style.background_gradient(cmap='Blues').format("{:,.2f}"
```

```
          2016        2017        2018        2019        2020
count  288.000000  288.000000  288.000000  288.000000  288.000000
mean     0.508397    0.585547    0.567773    0.521803    0.516621
std      0.539851    0.546949    0.545102    0.515412    0.552255
min     -0.896605    0.000000   -1.113032    0.000000   -0.200397
25%      0.000000    0.000000    0.000000    0.000000    0.000000
50%      0.597382    0.811088    0.809756    0.691984    0.345514
75%      0.949548    1.029891    0.997362    0.971385    0.979243
max      2.053673    2.103911    2.470816    2.052330    2.387685
```

Out[ ]:

Firms_ID	2016	2017	2018	2019	2020
Firm_97	0.99	1.06	1.01	1.00	2.39
Firm_21	2.01	1.07	0.85	0.68	2.30
Firm_29	0.10	1.52	1.33	0.76	2.15
Firm_214	0.00	0.93	0.93	0.97	1.98
Firm_316	1.57	1.36	2.47	1.63	1.75
Firm_239	2.05	0.99	1.13	1.11	1.74
Firm_137	0.82	1.31	1.59	1.13	1.53
Firm_144	1.03	1.45	1.05	1.14	1.51
Firm_300	0.92	1.00	1.05	1.32	1.45
Firm_160	0.93	1.05	0.94	1.06	1.45

In [ ]:

```
# calculate YoY of net combined ratio of tncri_mod_sort all years
tncri_mod_yoy = tncri_mod_sort.pct_change(axis='columns').mul(100).round(2)
tncri_mod_yoy.head(10).style.background_gradient(cmap='Blues').format("{:,.2f}")
```

```
c:\Users\Syarmine\AppData\Local\Programs\Python\Python312\Lib\site-packages\panda
s\io\formats\style.py:3819: RuntimeWarning: All-NaN slice encountered
  smin = np.nanmin(gmap) if vmin is None else vmin
c:\Users\Syarmine\AppData\Local\Programs\Python\Python312\Lib\site-packages\panda
s\io\formats\style.py:3820: RuntimeWarning: All-NaN slice encountered
  smax = np.nanmax(gmap) if vmax is None else vmax
c:\Users\Syarmine\AppData\Local\Programs\Python\Python312\Lib\site-packages\panda
s\io\formats\style.py:3823: RuntimeWarning: invalid value encountered in scalar m
ultiply
  norm = _matplotlib.colors.Normalize(smin - (rng * low), smax + (rng * high))
```

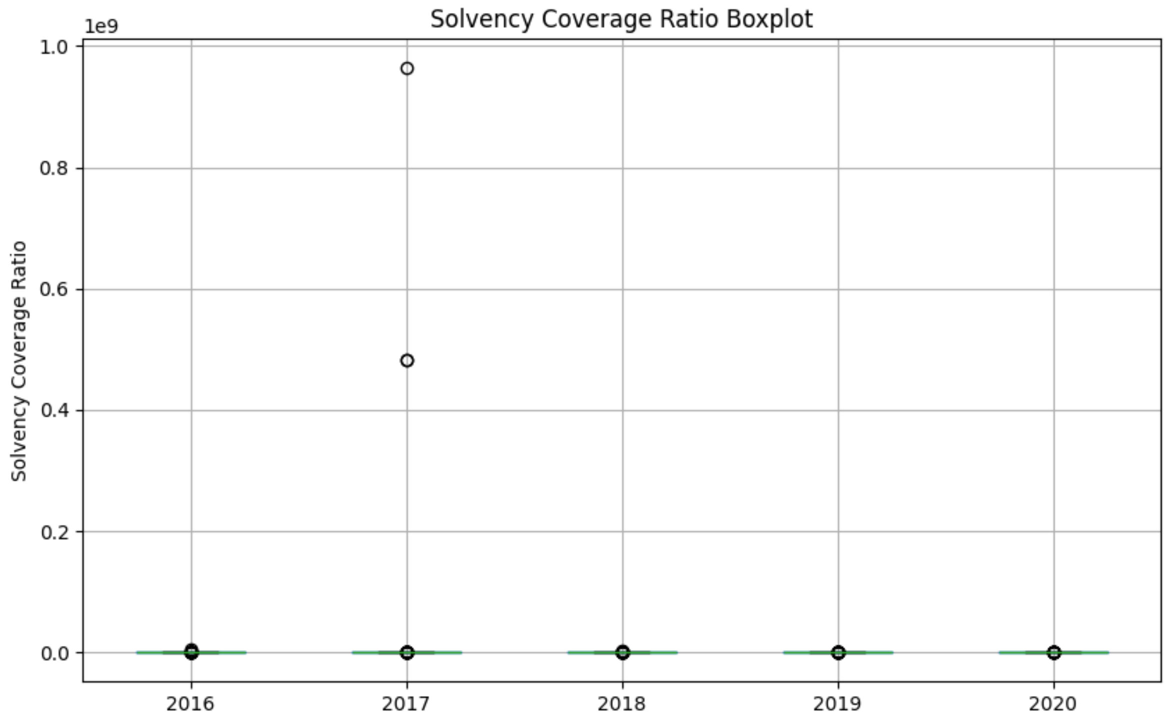
Out[ ]:

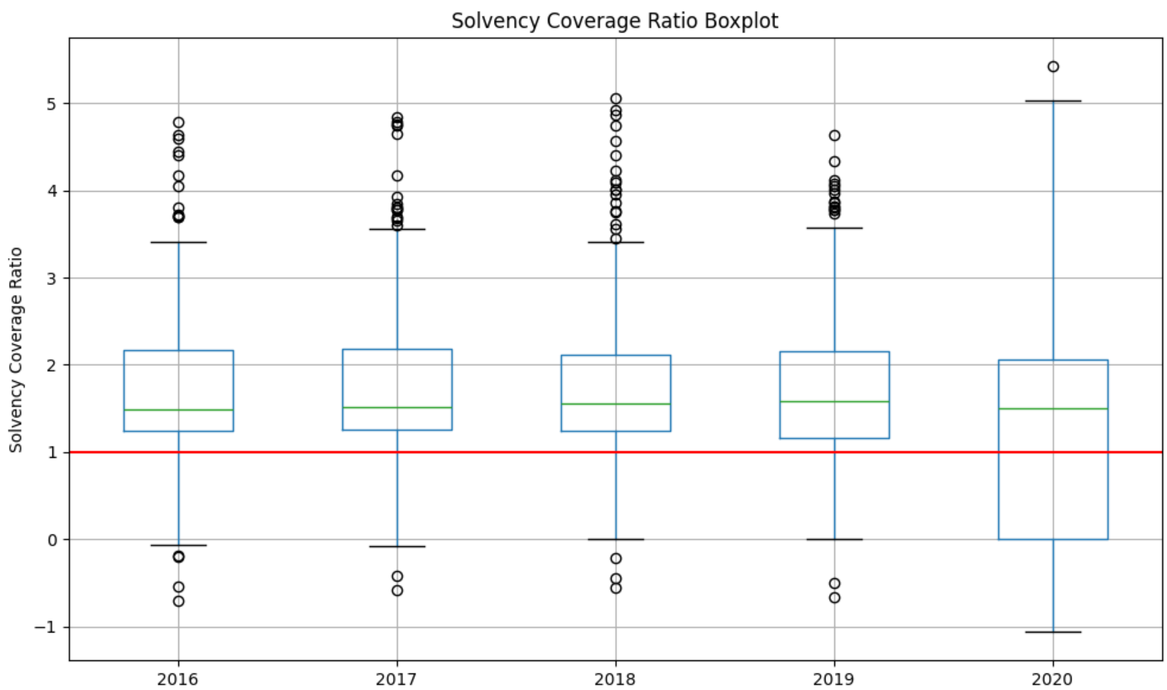
Firms_ID	2016	2017	2018	2019	2020
Firm_97	nan	6.53	-4.79	-0.68	138.98
Firm_21	nan	-46.75	-20.91	-19.63	238.17
Firm_29	nan	1,376.04	-12.44	-43.07	183.62
Firm_214	nan	inf	-0.39	4.36	104.56
Firm_316	nan	-13.44	81.39	-33.89	7.27
Firm_239	nan	-51.55	13.96	-1.82	56.39
Firm_137	nan	59.40	21.07	-28.90	35.28
Firm_144	nan	40.65	-28.01	8.92	32.51
Firm_300	nan	7.88	5.80	24.87	10.59
Firm_160	nan	13.13	-10.74	13.61	36.62

In [ ]:

```
# create a box plot for solvency coverage ratio for each firm
print(tscrcr.describe())
tscrcr.boxplot(figsize=(10, 6))
plt.ylabel('Solvency Coverage Ratio')
plt.title('Solvency Coverage Ratio Boxplot')
plt.show()

# create a variable outlier dataframe to allow return of the outlier in a datafr
q1 = tscrcr.quantile(0.25)
q3 = tscrcr.quantile(0.75)
iqr = q3 - q1
tscrcr_outlier = ((tscrcr < (q1 - 1.5 * iqr)) | (tscrcr > (q3 + 1.5 * iqr))).any

# new variable without outliers to improve readability of the boxplot
tscrcr_mod = tscrcr[~((tscrcr < (q1 - 1.5 * iqr)) | (tscrcr > (q3 + 1.5 * iqr)))

# create a box plot of tscrcr
# add a reference line of 100% or at 1.0
tscrcr_mod.boxplot(figsize=(10, 6))
plt.axhline(y=1.0, color='r', linestyle='-')
plt.ylabel('Solvency Coverage Ratio')
plt.title('Solvency Coverage Ratio Boxplot')
plt.tight_layout()
plt.show()
```

```
               2016           2017          2018           2019           2020
count  3.250000e+02  3.250000e+02  3.250000e+02     325.000000     325.000000
mean   1.286947e+04  5.930313e+06  1.246741e+04     533.017116     514.215109
std    2.319518e+05  6.529400e+07  2.141325e+05    9539.236980    9229.786796
min   -1.974450e+00 -1.973652e+00 -5.515428e-01      -0.669013      -1.066521
25%    1.276845e+00  1.302423e+00  1.268210e+00       1.177754       0.000000
50%    1.662747e+00  1.755881e+00  1.693416e+00       1.710544       1.565516
75%    2.780175e+00  3.203697e+00  2.795907e+00       2.691263       2.367988
max    4.181573e+06  9.635840e+08  3.856018e+06  171974.690816  166394.575872
```





In [ ]:
```
# sort the solvency coverage ratio (before cleaning - without outlier) ratio by
tscrcr_sort = tscrcr.sort_values(by=['2020'], ascending=False)
tscrcr_sort.head(10).style.background_gradient(cmap='Blues').format("{:,.2f}")
```

Out[ ]:

Firms_ID	2016	2017	2018	2019	2020
Firm_127	0.00	182,412.23	194,753.82	171,974.69	166,394.58
Firm_177	0.00	23.29	31.01	53.35	23.44
Firm_312	3.56	14.43	22.40	22.49	21.86
Firm_232	13.52	15.69	17.83	18.08	18.57
Firm_190	5.94	7.43	14.14	16.77	16.03
Firm_133	1.99	1.90	2.80	2.56	15.41
Firm_282	2.18	5.21	5.43	13.67	15.23
Firm_300	11.46	11.17	11.15	10.45	14.35
Firm_278	2.79	12.36	14.48	14.40	13.65
Firm_94	6.09	12.77	13.03	13.58	13.62

In [ ]:

```
# sort the solvency coverage ratio (after cleaning - without outlier) ratio by 2
print(tscrcr_mod.describe())
tscrcr_mod_sort = tscrcr_mod.sort_values(by=['2020'], ascending=False)
tscrcr_mod_sort.head(10).style.background_gradient(cmap='Blues').format("{:,.2f}
```

```
             2016        2017        2018        2019        2020
count  257.000000  257.000000  257.000000  257.000000  257.000000
mean     1.670129    1.705842    1.668066    1.594209    1.417733
std      0.993736    1.041048    1.090796    1.096293    1.123326
min     -0.699561   -0.579237   -0.551543   -0.669013   -1.066521
25%      1.237247    1.254904    1.246388    1.154880    0.000000
50%      1.495556    1.516419    1.553863    1.579328    1.497979
75%      2.167277    2.180775    2.117042    2.152630    2.057611
max      4.788990    4.839561    5.060825    4.631008    5.426021
```

Out[ ]:

Firms_ID	2016	2017	2018	2019	2020
Firm_20	2.32	4.84	5.06	3.82	5.43
Firm_62	0.00	2.19	4.10	3.36	5.03
Firm_13	3.26	3.56	3.77	3.96	4.59
Firm_196	2.48	3.93	4.01	4.02	4.08
Firm_147	2.22	3.51	2.57	4.08	4.05
Firm_75	2.84	2.79	3.12	4.05	3.68
Firm_250	4.05	3.70	3.76	3.04	3.67
Firm_180	2.24	4.17	3.96	3.47	3.64
Firm_138	3.30	3.49	2.75	0.00	3.53
Firm_136	2.96	1.70	1.76	1.89	3.52

In [ ]:
```python
# calculate YoY of solvency coverage ratio of tscrcr_mod_sort all years
tscrcr_mod_yoy = tscrcr_mod_sort.pct_change(axis='columns').mul(100).round(2)
tscrcr_mod_yoy.head(10).style.background_gradient(cmap='Blues').format("{:,.2f}"
```

```
c:\Users\Syarmine\AppData\Local\Programs\Python\Python312\Lib\site-packages\panda
s\io\formats\style.py:3819: RuntimeWarning: All-NaN slice encountered
  smin = np.nanmin(gmap) if vmin is None else vmin
c:\Users\Syarmine\AppData\Local\Programs\Python\Python312\Lib\site-packages\panda
s\io\formats\style.py:3820: RuntimeWarning: All-NaN slice encountered
  smax = np.nanmax(gmap) if vmax is None else vmax
c:\Users\Syarmine\AppData\Local\Programs\Python\Python312\Lib\site-packages\panda
s\io\formats\style.py:3823: RuntimeWarning: invalid value encountered in scalar m
ultiply
  norm = _matplotlib.colors.Normalize(smin - (rng * low), smax + (rng * high))
```

Out[ ]:

	2016	2017	2018	2019	2020
Firms_ID					
Firm_20	nan	108.16	4.57	-24.61	42.22
Firm_62	nan	inf	86.87	-17.94	49.74
Firm_13	nan	9.00	5.96	5.22	15.83
Firm_196	nan	58.60	1.91	0.19	1.53
Firm_147	nan	57.83	-26.79	58.87	-0.84
Firm_75	nan	-1.94	11.92	29.91	-9.10
Firm_250	nan	-8.63	1.63	-19.07	20.64
Firm_180	nan	86.10	-5.08	-12.40	4.89
Firm_138	nan	5.57	-21.13	-100.00	inf
Firm_136	nan	-42.68	3.90	7.37	85.68

In [ ]: