Solvency and Financial Report Across Firms

(Analysed by: Syarmine Shah)

Background:

The Bank of England is the United Kingdom's central bank, with its pivotal mission is to deliver monetary and financial stability for the British people. On 1 April 2013 the Prudential Regulation Authority (PRA) became responsible for the prudential regulation and supervision of banks, building societies, credit unions, insurers and major investment firms. The PRA was created by the Financial Services Act (2012) and is part of the Bank of England.

The PRA regulates around 1,500 banks, building societies, credit unions, insurers and major investment firms. The list can be found here:

- List of banks
- List of building societies
- List of authorised credit unions
- List of UK authorised insurers, List of SRO authorised insurers, List of TPR authorised insurers and List of Gibraltar authorised insurers
- List of investment firms

Business Understanding:

Since the implementation of Solvency II, the Bank of England and PRA recognised the value of UK Insurance data in a timely and structured publication based on the regular submission of data by Solvency II firms across the markets. The effort and improvements by firms to provide good quality data have resulted in the PRA being able to provide aggregated market data for industry users outside of the PRA.

While the Bank of England and PRA published an quarterly and annually aggregated insurance statistics for public facing, this assignment includes granular data at firm-level. This task is to assist a Supervision Manager from the Bank to allocating scarce resources, and identify which firms their team should prioritise. Supervisory resource may be allocated according to the following characteristics:

- Firm size (i.e. the biggest firms need more attention)
- Changing business profile (are firms' data changing substantially year-on-year?)
- Outliers from the norm (when looking at a single reporting period, does a firm deviate significantly from the average?)

This assignment aims to identify and prioritise firms to supervise for resource allocation and propose a strategic approach to Supervision Manager across 325 anonymised firms under the Bank of England Prudential Regulatory Authority (PRA). All amount disclosde in this report are in British Pounds(£), unless stated otherwise.

First, we import the relevant packages to read the dataset in `Data for technical assessment.xlsx` to our Jupyter notebook

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
```

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)

df.head(5)
```

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

In [ ]:
```
# inspect data types and memory usage for Dataset 1
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 326 entries, 0 to 325
Data columns (total 41 columns):
 #   Column                                               Non-Null Count  Dtype 
---  ------                                               --------------  ----- 
 0   Unnamed: 0                                           325 non-null    object
 1   NWP (£m)                                             326 non-null    object
 2   NWP (£m) .1                                          326 non-null    object
 3   NWP (£m) .2                                          326 non-null    object
 4   NWP (£m) .3                                          326 non-null    object
 5   NWP (£m) .4                                          326 non-null    object
 6   SCR (£m)                                             326 non-null    object
 7   SCR (£m).1                                           326 non-null    object
 8   SCR (£m).2                                           326 non-null    object
 9   SCR (£m).3                                           326 non-null    object
 10  SCR (£m).4                                           326 non-null    object
 11  EoF for SCR (£m)                                     326 non-null    object
 12  EoF for SCR (£m).1                                   326 non-null    object
 13  EoF for SCR (£m).2                                   326 non-null    object
 14  EoF for SCR (£m).3                                   326 non-null    object
 15  EoF for SCR (£m).4                                   326 non-null    object
 16  SCR coverage ratio                                   326 non-null    object
 17  SCR coverage ratio.1                                 326 non-null    object
 18  SCR coverage ratio.2                                 326 non-null    object
 19  SCR coverage ratio.3                                 326 non-null    object
 20  SCR coverage ratio.4                                 326 non-null    object
 21  GWP (£m)                                             326 non-null    object
 22  GWP (£m).1                                           326 non-null    object
 23  GWP (£m).2                                           326 non-null    object
 24  GWP (£m).3                                           326 non-null    object
 25  GWP (£m).4                                           326 non-null    object
 26  Total assets (£m)                                    326 non-null    object
 27  Total assets (£m).1                                  326 non-null    object
 28  Total assets (£m).2                                  326 non-null    object
 29  Total assets (£m).3                                  326 non-null    object
 30  Total assets (£m).4                                  326 non-null    object
 31  Total liabilities (£m)                               326 non-null    object
 32  Total liabilities (£m).1                             326 non-null    object
 33  Total liabilities (£m).2                             326 non-null    object
 34  Total liabilities (£m).3                             326 non-null    object
 35  Total liabilities (£m).4                             326 non-null    object
 36  Excess of assets over liabilities (£m) [= equity]    326 non-null    object
 37  Excess of assets over liabilities (£m) [= equity].1  326 non-null    object
 38  Excess of assets over liabilities (£m) [= equity].2  326 non-null    object
 39  Excess of assets over liabilities (£m) [= equity].3  326 non-null    object
 40  Excess of assets over liabilities (£m) [= equity].4  326 non-null    object
dtypes: object(41)
memory usage: 104.6+ KB
```

In [ ]:

```
# inspect data types and memory usage for Dataset 2
df_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 326 entries, 0 to 325
Data columns (total 46 columns):
 #   Column                                       Non-Null Count  Dtype 
---  ------                                       --------------  ----- 
 0   Unnamed: 0                                   325 non-null    object
 1   Gross claims incurred (£m)                   326 non-null    object
 2   Gross claims incurred (£m).1                 326 non-null    object
 3   Gross claims incurred (£m).2                 326 non-null    object
 4   Gross claims incurred (£m).3                 326 non-null    object
 5   Gross claims incurred (£m).4                 326 non-null    object
 6   Gross BEL (inc. TPs as whole, pre-TMTP) (£m)    326 non-null    object
 7   Gross BEL (inc. TPs as whole, pre-TMTP) (£m).1  326 non-null    object
 8   Gross BEL (inc. TPs as whole, pre-TMTP) (£m).2  326 non-null    object
 9   Gross BEL (inc. TPs as whole, pre-TMTP) (£m).3  326 non-null    object
 10  Gross BEL (inc. TPs as whole, pre-TMTP) (£m).4  326 non-null    object
 11  Net BEL (inc. TPs as a whole, pre-TMTP) (£m)    326 non-null    object
 12  Net BEL (inc. TPs as a whole, pre-TMTP) (£m).1  326 non-null    object
 13  Net BEL (inc. TPs as a whole, pre-TMTP) (£m).2  326 non-null    object
 14  Net BEL (inc. TPs as a whole, pre-TMTP) (£m).3  326 non-null    object
 15  Net BEL (inc. TPs as a whole, pre-TMTP) (£m).4  326 non-null    object
 16  Pure net claims ratio                        326 non-null    object
 17  Pure net claims ratio.1                      326 non-null    object
 18  Pure net claims ratio.2                      326 non-null    object
 19  Pure net claims ratio.3                      326 non-null    object
 20  Pure net claims ratio.4                      326 non-null    object
 21  Net expense ratio                            326 non-null    object
 22  Net expense ratio.1                          326 non-null    object
 23  Net expense ratio.2                          326 non-null    object
 24  Net expense ratio.3                          326 non-null    object
 25  Net expense ratio.4                          326 non-null    object
 26  Net combined ratio                           326 non-null    object
 27  Net combined ratio.1                         326 non-null    object
 28  Net combined ratio.2                         326 non-null    object
 29  Net combined ratio.3                         326 non-null    object
 30  Net combined ratio.4                         326 non-null    object
 31  Pure gross claims ratio                      326 non-null    object
 32  Pure gross claims ratio.1                    326 non-null    object
 33  Pure gross claims ratio.2                    326 non-null    object
 34  Pure gross claims ratio.3                    326 non-null    object
 35  Pure gross claims ratio.4                    326 non-null    object
 36  Gross expense ratio                          326 non-null    object
 37  Gross expense ratio.1                        326 non-null    object
 38  Gross expense ratio.2                        326 non-null    object
 39  Gross expense ratio.3                        326 non-null    object
 40  Gross expense ratio.4                        326 non-null    object
 41  Gross combined ratio                         326 non-null    object
 42  Gross combined ratio.1                       326 non-null    object
 43  Gross combined ratio.2                       326 non-null    object
 44  Gross combined ratio.3                       326 non-null    object
 45  Gross combined ratio.4                       326 non-null    object
dtypes: object(46)
memory usage: 117.3+ KB
```

In [ ]:

```
# check column names
print(df.columns)
```

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

In [ ]:

```
# check columns names
print(df_1.columns)
```

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

In [ ]:

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

# Assign the new header to the DataFrame and drop the first two rows
df.columns = new_header
```

```python
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
```

Out[ ]:

	Unnamed: 0	2016YE_NWP (£m)	2017YE_NWP (£m)	2018YE_NWP (£m)	2019YE_NWP (£m)	2020YE_NWI (£m
0	Firm 1	-13779.815629	0	0	0	(
1	Firm 2	28.178059	26.865049	25.064438	23.226445	21.71855ł
2	Firm 3	0	75.609681	70.578732	78.432782	85.7358:
3	Firm 4	22344.199923	23963.910709	25760.390158	25512.748836	24996.02104:
4	Firm 5	68.200993	51.663132	44.010833	42.008556	81.27365:
...	...	...	...	...	...	..
320	Firm 321	0	0	-1.011367	-6.599067	24.63223
321	Firm 322	2092.156137	2084.124818	2022.212247	2103.048716	2029.69701:
322	Firm 323	0	0	0	0	(
323	Firm 324	23.41538	22.650321	24.268465	25.811984	26.54663ł
324	Firm 325	240.999886	252.698937	332.521848	294.886332	(

325 rows × 41 columns

In [ ]:

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

print(df.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')
```

In [ ]:

```
# Similarly, there are 2 headers rows for Dataset 2, create a new header from th
# Create a new header from the first two rows
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
```

Out[ ]:

	Unnamed: 0	2016YE_Gross claims incurred (£m)	2017YE_Gross claims incurred (£m)	2018YE_Gross claims incurred (£m)	2019YE_Gross claims incurred (£m)	2020YE_Gro clair incurred (£r
0	Firm 1	0	0.046674	0	0	
1	Firm 2	39.241067	35.948249	29.002244	0	
2	Firm 3	0	0	0	0	
3	Firm 4	17.125976	75.535951	119.426995	35.884204	6.5679
4	Firm 5	30.485185	247.872969	449.87474	348.536337	373.7868
...	...	...	...	...	...	...
320	Firm 321	4.928695	4.808705	4.809954	4.79802	3.4658
321	Firm 322	82.741987	96.426952	83.289347	125.995045	200.6946
322	Firm 323	0	0	0	0	
323	Firm 324	6.739499	6.837802	6.863463	7.242601	5.098
324	Firm 325	1.198052	1.37684	3.804573	2.30167	1.3381

325 rows × 46 columns

In [ ]:

```python
# rename the first column "Unnamed: 0" to "Firms_ID" for clarity
df_1 = df_1.rename(columns={'Unnamed: 0':'Firms_ID'})

print(df_1.columns)
```

```
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 [ ]:

```
# Clean column names, strip any space and replace with "_" for Dataset 1
def clean_col(col):
    col = col.strip()
    col = col.replace(" ","_")
    return col

new_columns = []
for c in df.columns:
    clean_c = clean_col(c)
    new_columns.append(clean_c)

df.columns = new_columns

print(df.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')
```

In [ ]:

```python
# Clean column names, strip any space and replace with "_" for Dataset 2
def clean_col(col):
    col = col.strip()
    col = col.replace(" ","_")
    return col

new_columns = []
for c in df_1.columns:
    clean_c = clean_col(c)
    new_columns.append(clean_c)

df_1.columns = new_columns

print(df_1.columns)
```

```
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 [ ]:
```
# melt the 1st worksheet to tall csv for dataviz
#df_tall = df.melt(id_vars=['Firms_ID'])
#df_tall.to_csv("df_tall.csv",index=False)

# melt the 2nd worksheet to tall csv for dataviz
#df_1tall = df_1.melt(id_vars=['Firms_ID'])
#df_1tall.to_csv("df_1tall.csv",index=False)
```

Data Understanding

The dataset that has been provided includes a list of 325 firms over a 5-year period, from 2016 to 2020. The key aspect of this dataset include:

- Firms identification: Each row represents a different firm.
- Annual financial: notated by 2016YE, 2017YE until 2020YE.
- Firms profile: There are several metrics included:
  - Net Written Premium(NWP)
  - Solvency Capital Ratio (SCR)
  - Eligible Own Fund Over SCR
  - SCR Coverage Ratio
  - Gross Written Premium
  - Total Assets
  - Total Liabilities

- Excess of Assets Over Liabilities (Equity):

In total, there are 325 observations (rows) across 41 features (columns), offering a comparative analysis providing valuable insights to assist strategic approach for the central bank supervisory team.

Data Cleaning

The raw dataframe looks reasonably clean, we need to check whether we need to reduce the number of columns. The importance of data cleaning to ensure that the data is accurate and reliable for any downstream analysis or use.

In [ ]:
```
# checking the number of rows and columns
print(df.shape)
print(df_1.shape)
```

```
(325, 41)
(325, 46)
```

In [ ]:
```
# Check missing value counts
df.isna().sum()
```

Out[ ]:

```
Firms_ID                                                  0
2016YE_NWP_(£m)                                           0
2017YE_NWP_(£m)                                           0
2018YE_NWP_(£m)                                           0
2019YE_NWP_(£m)                                           0
2020YE_NWP_(£m)                                           0
2016YE_SCR_(£m)                                           0
2017YE_SCR_(£m)                                           0
2018YE_SCR_(£m)                                           0
2019YE_SCR_(£m)                                           0
2020YE_SCR_(£m)                                           0
2016YE_EoF_for_SCR_(£m)                                   0
2017YE_EoF_for_SCR_(£m)                                   0
2018YE_EoF_for_SCR_(£m)                                   0
2019YE_EoF_for_SCR_(£m)                                   0
2020YE_EoF_for_SCR_(£m)                                   0
2016YE_SCR_coverage_ratio                                 0
2017YE_SCR_coverage_ratio                                 0
2018YE_SCR_coverage_ratio                                 0
2019YE_SCR_coverage_ratio                                 0
2020YE_SCR_coverage_ratio                                 0
2016YE_GWP_(£m)                                           0
2017YE_GWP_(£m)                                           0
2018YE_GWP_(£m)                                           0
2019YE_GWP_(£m)                                           0
2020YE_GWP_(£m)                                           0
2016YE_Total_assets_(£m)                                  0
2017YE_Total_assets_(£m)                                  0
2018YE_Total_assets_(£m)                                  0
2019YE_Total_assets_(£m)                                  0
2020YE_Total_assets_(£m)                                  0
2016YE_Total_liabilities_(£m)                             0
2017YE_Total_liabilities_(£m)                             0
2018YE_Total_liabilities_(£m)                             0
2019YE_Total_liabilities_(£m)                             0
2020YE_Total_liabilities_(£m)                             0
2016YE_Excess_of_assets_over_liabilities_(£m)_[=_equity]    0
2017YE_Excess_of_assets_over_liabilities_(£m)_[=_equity]    0
2018YE_Excess_of_assets_over_liabilities_(£m)_[=_equity]    0
2019YE_Excess_of_assets_over_liabilities_(£m)_[=_equity]    0
2020YE_Excess_of_assets_over_liabilities_(£m)_[=_equity]    0
dtype: int64
```

In [ ]:

```python
# Check missing value counts
print(df_1.isna().sum())
```

```
Firms_ID                                                  0
2016YE_Gross_claims_incurred_(£m)                         0
2017YE_Gross_claims_incurred_(£m)                         0
2018YE_Gross_claims_incurred_(£m)                         0
2019YE_Gross_claims_incurred_(£m)                         0
2020YE_Gross_claims_incurred_(£m)                         0
2016YE_Gross_BEL_(inc._TPs_as_whole,_pre-TMTP)_(£m)       0
2017YE_Gross_BEL_(inc._TPs_as_whole,_pre-TMTP)_(£m)       0
2018YE_Gross_BEL_(inc._TPs_as_whole,_pre-TMTP)_(£m)       0
2019YE_Gross_BEL_(inc._TPs_as_whole,_pre-TMTP)_(£m)       0
2020YE_Gross_BEL_(inc._TPs_as_whole,_pre-TMTP)_(£m)       0
2016YE_Net_BEL_(inc._TPs_as_a_whole,_pre-TMTP)_(£m)       0
2017YE_Net_BEL_(inc._TPs_as_a_whole,_pre-TMTP)_(£m)       0
2018YE_Net_BEL_(inc._TPs_as_a_whole,_pre-TMTP)_(£m)       0
2019YE_Net_BEL_(inc._TPs_as_a_whole,_pre-TMTP)_(£m)       0
2020YE_Net_BEL_(inc._TPs_as_a_whole,_pre-TMTP)_(£m)       0
2016YE_Pure_net_claims_ratio                              0
2017YE_Pure_net_claims_ratio                              0
2018YE_Pure_net_claims_ratio                              0
2019YE_Pure_net_claims_ratio                              0
2020YE_Pure_net_claims_ratio                              0
2016YE_Net_expense_ratio                                  0
2017YE_Net_expense_ratio                                  0
2018YE_Net_expense_ratio                                  0
2019YE_Net_expense_ratio                                  0
2020YE_Net_expense_ratio                                  0
2016YE_Net_combined_ratio                                 0
2017YE_Net_combined_ratio                                 0
2018YE_Net_combined_ratio                                 0
2019YE_Net_combined_ratio                                 0
2020YE_Net_combined_ratio                                 0
2016YE_Pure_gross_claims_ratio                            0
2017YE_Pure_gross_claims_ratio                            0
2018YE_Pure_gross_claims_ratio                            0
2019YE_Pure_gross_claims_ratio                            0
2020YE_Pure_gross_claims_ratio                            0
2016YE_Gross_expense_ratio                                0
2017YE_Gross_expense_ratio                                0
2018YE_Gross_expense_ratio                                0
2019YE_Gross_expense_ratio                                0
2020YE_Gross_expense_ratio                                0
2016YE_Gross_combined_ratio                               0
2017YE_Gross_combined_ratio                               0
2018YE_Gross_combined_ratio                               0
2019YE_Gross_combined_ratio                               0
2020YE_Gross_combined_ratio                               0
dtype: int64
```

In [ ]:

```
# drop empty column if any
df.dropna(axis=1, how='all', inplace=True)
df.shape
```

Out[ ]: (325, 41)

In [ ]:

```
# drop empty column if any
df_1.dropna(axis=1, how='all', inplace=True)
df_1.shape
```

Out[ ]: (325, 46)

Exploratory Data Analysis

Next, we undertake exploratory data analysis (EDA) to further understand the structure of the data, uncover patterns and relationships, and identify potential outliers and anomalies.

The dataset appears to be well-structured without any missing values, and the data types are appropriate for the analysis. Each row represents a firm, with columns for different financial metrics across various years (2016 to 2020).

The `Data for technical assessment.xlsx` dataset contains various financial metrics for different firms across multiple years. These metrics include Net Written Premium (NWP), Solvency Capital Requirement (SCR), Total Liabilities, and Excess of Assets over Liabilities, among others.

For our analysis, we will focus on the relevant key metrics:

- Firm Size: Use the Gross Written Premium (GWP) and Net Written Premium (NWP).
- Changing Business Profile: Looking at the year-on-year changes in these key metrics.
- Outliers and Deviations: Identify firms with significant deviations from the average in the reporting periods.

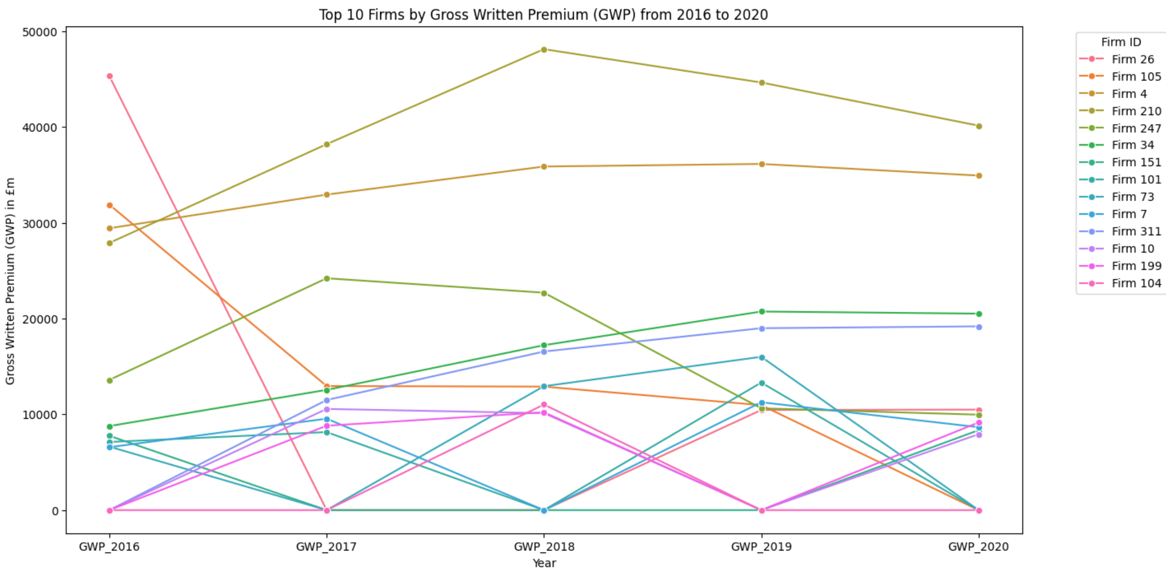
1. Firm Size Analysis

The supervisory team requires us to identify firm size (i.e. the biggest firms need more attention). The following metrics has been identified

- Gross Written Premium (GWP)
- Net Written Premium (NWP)
- Total Assets

In [ ]:
```python
# Generate a plot for top 10 firms across 5-year period by GWP
years = range(2016, 2021)
top_firms_gwp = pd.DataFrame()

for year in years:
    gwp_col = f'{year}YE_GWP_(£m)'
    if gwp_col in df.columns:
        top_10_gwp = df[['Firms_ID', gwp_col]].sort_values(by=gwp_col, ascending
        top_10_gwp.columns = ['Firms_ID', f'GWP_{year}']
        if top_firms_gwp.empty:
            top_firms_gwp = top_10_gwp
        else:
            top_firms_gwp = top_firms_gwp.merge(top_10_gwp, on='Firms_ID', how='

# Replace NaN values with 0 if a firm wasn't in the top 10 that year
top_firms_gwp.fillna(0, inplace=True)

# Reshape for plotting
top_firms_long = pd.melt(top_firms_gwp, id_vars=['Firms_ID'], var_name='Year', v
```

```
# Plot
plt.figure(figsize=(15, 8))
sns.lineplot(data=top_firms_long, x='Year', y='GWP', hue='Firms_ID', marker='o')
plt.title('Top 10 Firms by Gross Written Premium (GWP) from 2016 to 2020')
plt.xlabel('Year')
plt.ylabel('Gross Written Premium (GWP) in £m')
plt.legend(title='Firm ID', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



In [ ]:
```
# Top 10 firms by GWP across 5 year period (latest 2020)
print(top_firms_gwp.sort_values(by='GWP_2020', ascending=False).head(10))
```

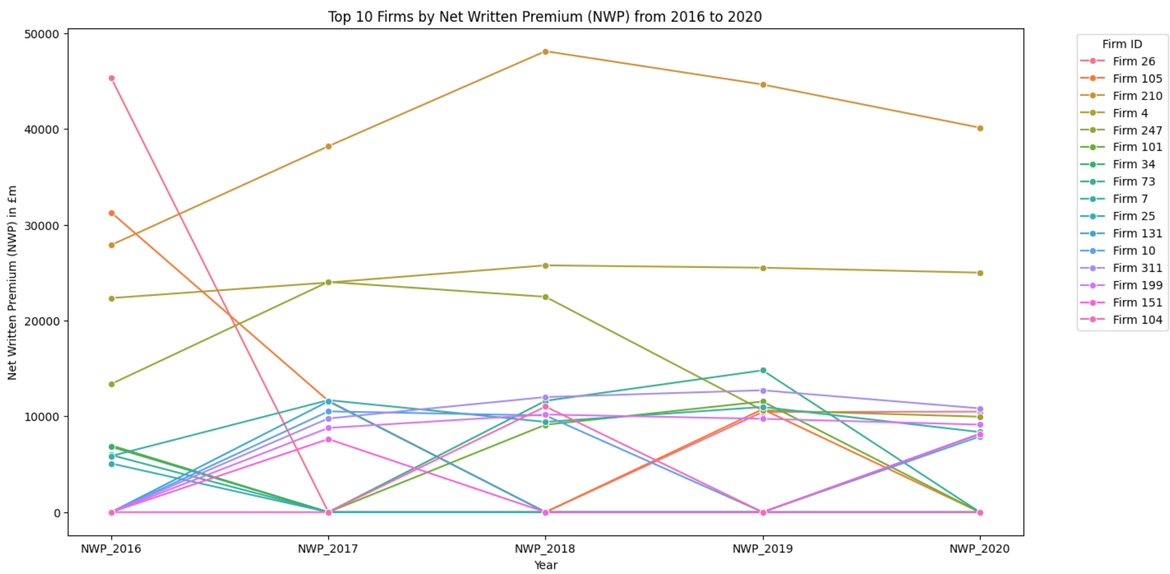
```
   Firms_ID      GWP_2016      GWP_2017      GWP_2018      GWP_2019  \
3  Firm 210  27889.340758  38199.311256  48117.993733  44638.769640
2    Firm 4  29424.574692  32935.401421  35867.637982  36135.463107
5   Firm 34   8772.495138  12550.060778  17214.590025  20729.165313
10 Firm 311      0.000000  11493.619715  16553.861172  18988.452773
0   Firm 26  45309.819760      0.000000      0.000000  10450.175547
4  Firm 247  13589.808933  24202.653436  22694.940614  10624.480076
12 Firm 199      0.000000   8808.031930  10211.520015      0.000000
9    Firm 7   6567.617425   9542.897521      0.000000  11259.286281
6  Firm 151   7753.569963      0.000000      0.000000      0.000000
11  Firm 10      0.000000  10559.901067  10122.406367      0.000000

       GWP_2020
3  40135.692258
2  34922.702554
5  20510.750552
10 19180.016479
0  10489.248083
4   9961.520679
12  9149.583691
9   8652.947413
6   8341.643250
11  7923.371752
```

The top 5 firms based on GWP (latest YE2020) are:

- Firm 210
- Firm 4
- Firm 34
- Firm 311

- Firm 26

In [ ]:
```
# Generate a plot for top 10 firms across 5-year period by NWP
years = range(2016, 2021)
top_firms_nwp = pd.DataFrame()

for year in years:
    nwp_col = f'{year}YE_NWP_(£m)'
    if nwp_col in df.columns:
        top_10_nwp = df[['Firms_ID', nwp_col]].sort_values(by=nwp_col, ascending
        top_10_nwp.columns = ['Firms_ID', f'NWP_{year}']
        if top_firms_nwp.empty:
            top_firms_nwp = top_10_nwp
        else:
            top_firms_nwp = top_firms_nwp.merge(top_10_nwp, on='Firms_ID', how='

# Replace NaN values with 0 if a firm wasn't in the top 10 that year
top_firms_nwp.fillna(0, inplace=True)

# Reshape for plotting
top_firms_long = pd.melt(top_firms_nwp, id_vars=['Firms_ID'], var_name='Year', v

# Plot
plt.figure(figsize=(15, 8))
sns.lineplot(data=top_firms_long, x='Year', y='NWP', hue='Firms_ID', marker='o')
plt.title('Top 10 Firms by Net Written Premium (NWP) from 2016 to 2020')
plt.xlabel('Year')
plt.ylabel('Net Written Premium (NWP) in £m')
plt.legend(title='Firm ID', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



In [ ]:
```
# Top 10 firms by NWP across 5 year period (latest by 2020)
print(top_firms_nwp.sort_values(by='NWP_2020', ascending=False).head(10))
```

```
    Firms_ID      NWP_2016      NWP_2017      NWP_2018      NWP_2019  \
2   Firm 210  27889.340758  38199.311256  48117.993733  44638.769640
3     Firm 4  22344.199923  23963.910709  25760.390158  25512.748836
12  Firm 311      0.000000   9777.534671  12009.157860  12719.398352
0    Firm 26  45309.838702      0.000000      0.000000  10450.175547
4   Firm 247  13377.534020  24031.377272  22475.773945  10624.480076
13  Firm 199      0.000000   8787.822318  10191.583020   9739.144474
8     Firm 7   5855.172823  11688.570412   9414.976495  10975.189662
14  Firm 151      0.000000   7626.419094      0.000000      0.000000
6    Firm 34   6817.399238      0.000000      0.000000      0.000000
11   Firm 10      0.000000  10516.210290  10087.572411      0.000000

        NWP_2020
2   40135.692258
3   24996.021042
12  10830.966262
0   10489.248083
4    9961.520679
13   9134.283485
8    8359.905292
14   8180.387573
6    8145.617320
11   7893.064120
```

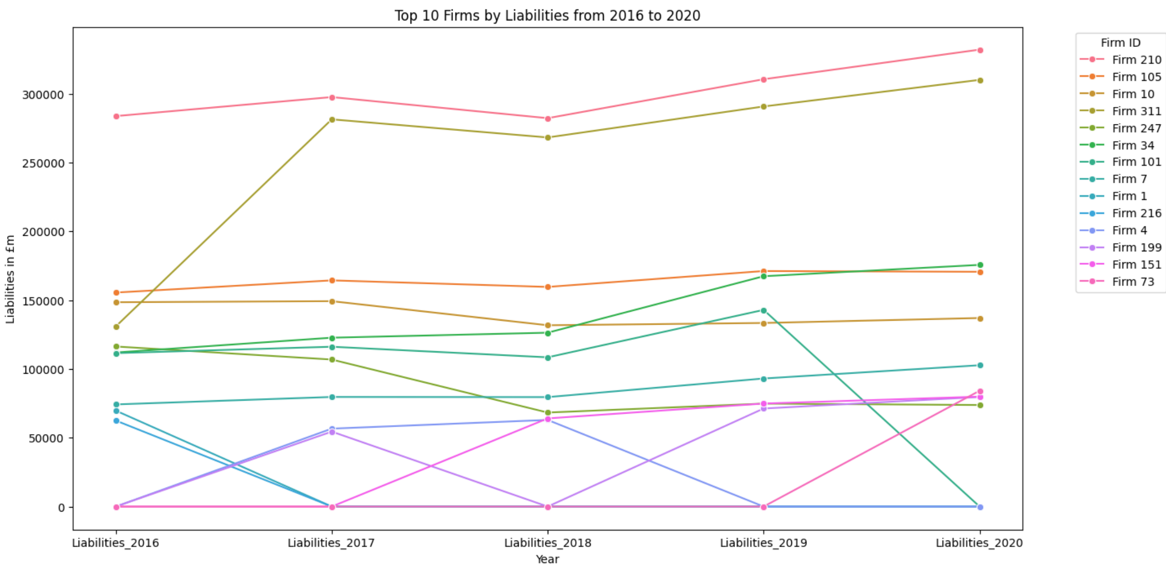
The top 5 firms based on NWP (latest YE2020) are:

- Firm 210
- Firm 4
- Firm 311
- Firm 26
- Firm 247

In [ ]:

```python
# Generate a plot for top 10 firms across 5-year period by liabilities
years = range(2016, 2021)
top_firms_liabilities = pd.DataFrame()

for year in years:
    liabilities_col = f'{year}YE_Total_liabilities_(£m)'
    if liabilities_col in df.columns:
        top_10_liabilities = df[['Firms_ID', liabilities_col]].sort_values(by=li
        top_10_liabilities.columns = ['Firms_ID', f'Liabilities_{year}']
        if top_firms_liabilities.empty:
            top_firms_liabilities = top_10_liabilities
        else:
            top_firms_liabilities = top_firms_liabilities.merge(top_10_liabiliti

# Replace NaN values with 0 if a firm wasn't in the top 10 that year
top_firms_liabilities.fillna(0, inplace=True)

# Reshape for plotting
top_firms_long = pd.melt(top_firms_liabilities, id_vars=['Firms_ID'], var_name='

# Plot
plt.figure(figsize=(15, 8))
sns.lineplot(data=top_firms_long, x='Year', y='Liabilities', hue='Firms_ID', mar
plt.title('Top 10 Firms by Liabilities from 2016 to 2020')
plt.xlabel('Year')
plt.ylabel('Liabilities in £m')
```

```
plt.legend(title='Firm ID', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



In [ ]:

```
# Top 10 firms by Total Assets across 5 year period by liabilities
print(top_firms_liabilities.sort_values(by='Liabilities_2020', ascending=False).
```

```
    Firms_ID  Liabilities_2016  Liabilities_2017  Liabilities_2018  \
0   Firm 210     283689.959282     297494.486109     282172.206629
3   Firm 311     130875.349593     281321.131413     268158.520939
5    Firm 34     111925.451583     122688.667353     126318.941718
1   Firm 105     155483.448925     164319.386755     159576.455604
2    Firm 10     148470.115416     149227.058198     131719.419536
7     Firm 7      74218.641565      79630.847612      79522.281318
13   Firm 73          0.000000          0.000000          0.000000
12  Firm 151          0.000000          0.000000      64040.529568
11  Firm 199          0.000000      54463.621320          0.000000
4   Firm 247     116290.805809     106818.700892      68307.249623

    Liabilities_2019  Liabilities_2020
0      310462.200186     331981.942022
3      290672.056115     309997.838779
5      167344.792930     175581.813381
1      171118.255519     170567.883029
2      133381.719533     136976.887186
7       93046.602201     102675.268288
13          0.000000      84131.867289
12      74838.678529      79870.395620
11      71232.628985      79602.515006
4       74811.040782      73787.703205
```

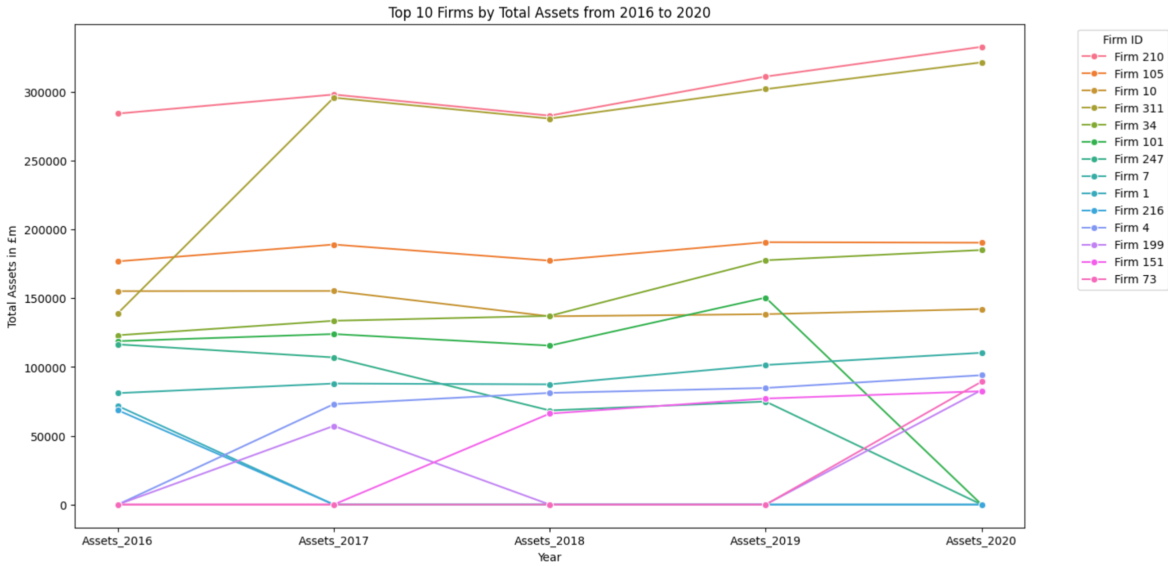
The top 5 firms based on Liabilities (latest YE2020) are:

- Firm 210
- Firm 311
- Firm 34
- Firm 105
- Firm 10

In [ ]:

```
# Generate a plot for top 10 firms across 5-year period by total assets
years = range(2016, 2021)
top_firms_assets = pd.DataFrame()
```

```python
for year in years:
    assets_col = f'{year}YE_Total_assets_(£m)'
    if assets_col in df.columns:
        top_10_assets = df[['Firms_ID', assets_col]].sort_values(by=assets_col,
        top_10_assets.columns = ['Firms_ID', f'Assets_{year}']
        if top_firms_assets.empty:
            top_firms_assets = top_10_assets
        else:
            top_firms_assets = top_firms_assets.merge(top_10_assets, on='Firms_I

# Replace NaN values with 0 if a firm wasn't in the top 10 that year
top_firms_assets.fillna(0, inplace=True)

# Reshape for plotting
top_firms_long = pd.melt(top_firms_assets, id_vars=['Firms_ID'], var_name='Year'

# Plot
plt.figure(figsize=(15, 8))
sns.lineplot(data=top_firms_long, x='Year', y='Assets', hue='Firms_ID', marker='
plt.title('Top 10 Firms by Total Assets from 2016 to 2020')
plt.xlabel('Year')
plt.ylabel('Total Assets in £m')
plt.legend(title='Firm ID', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



In [ ]:
```python
# Top 10 firms by Total Assets across 5 year period by Assets
print(top_firms_assets.sort_values(by='Assets_2020', ascending=False).head(10))
```

```
   Firms_ID    Assets_2016    Assets_2017    Assets_2018    Assets_2019  \
0   Firm 210  284329.904525  298172.843043  282829.545364  311228.369410
3   Firm 311  139095.392068  295913.903630  280664.424584  302099.261772
1   Firm 105  176860.528735  189105.525231  177363.073106  190804.876231
4    Firm 34  123131.663849  133659.761869  137198.165534  177652.791423
2    Firm 10  155205.156462  155343.125199  136914.319786  138456.983975
7     Firm 7   81043.288268   88006.339149   87435.584850  101517.063020
10    Firm 4       0.000000   73034.629260   81184.829269   84801.523977
13   Firm 73       0.000000       0.000000       0.000000       0.000000
11  Firm 199       0.000000   57192.482221       0.000000       0.000000
12  Firm 151       0.000000       0.000000   66160.041081   77101.970129

      Assets_2020
0   332875.903638
3   321563.599813
1   190431.207543
4   185108.328631
2   142144.987135
7   110371.664099
10   94065.081033
13   89412.166323
11   83298.942335
12   82397.812536
```

The top 5 firms based on Total Assets (latest YE2020) are:

- Firm 210
- Firm 311
- Firm 105
- Firm 34
- Firm 10

In summmary, the 5 firms according to categories based on 2020 observation:

Rank	GWP	NWP	Liabilities	Total Assets
1.	Firm 210	Firm 210	Firm 210	Firm 210
2.	Firm 4	Firm 4	Firm 311	Firm 311
3.	Firm 34	Firm 311	Firm 34	Firm 105
4.	Firm 311	Firm 26	Firm 105	Firm 34
5.	Firm 26	Firm 247	Firm 10	Firm 10

2. Changing Business Profile

2.1 SCR Coverage Ratio

The total SCR coverage ratio for the companies included in the dataset was 514% at year-end 2020, a decrease of 18% compared to 2019. Despite that, this shows that firms in England continue to hold a significant capital buffer in excess of solvency capital requirement.

In 2020, the majority of companies had a SCR coverage ratio between 156% and 237%, with notable firms minimum position with negative SCR ratio. The maximum SCR coverage ratio in the dataset was 16639457%, which is an extremely high value that suggests that the company had a very low SCR or a very high amount of eligible own funds or an outliers.

Below are analysis shows the SCR Coverage ratio across firms included in the dataset

In [ ]:
```
# Total of SCR Coverage Ratio By Years
years = ['2016', '2017', '2018', '2019', '2020']

for year in years:
    column_name = f'{year}YE_SCR_coverage_ratio'
    sum_value = df[column_name].sum() / 325
    print(f"The Total for SCR Coverage Ratio for Year {year} was sum of {sum_val
```

```
The Total for SCR Coverage Ratio for Year 2016 was sum of 12869.471623832738
The Total for SCR Coverage Ratio for Year 2017 was sum of 5930312.610747993
The Total for SCR Coverage Ratio for Year 2018 was sum of 12467.409446589167
The Total for SCR Coverage Ratio for Year 2019 was sum of 533.0171160017842
The Total for SCR Coverage Ratio for Year 2020 was sum of 514.2151088579244
```

In [ ]:
```
# change dtype to float for analysis
years = ['2016YE', '2017YE', '2018YE', '2019YE', '2020YE']

for year in years:
    df[f'{year}_SCR_coverage_ratio'] = df[f'{year}_SCR_coverage_ratio'].astype(f
```

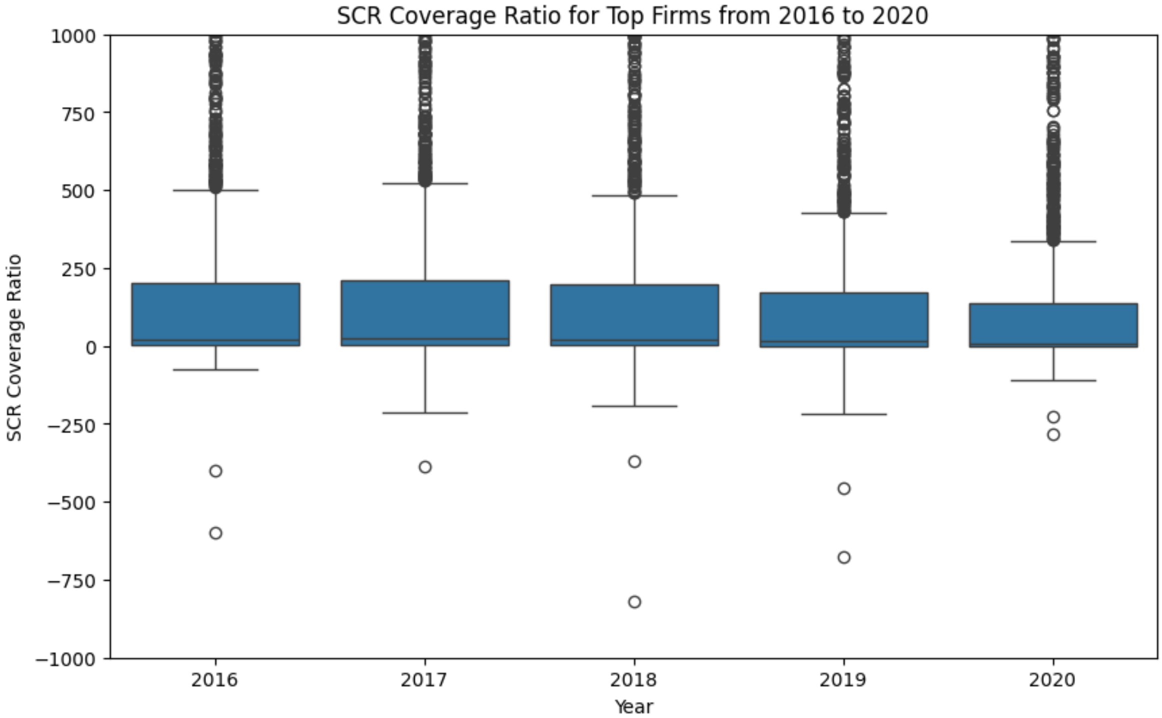
In [ ]:
```
# Descriptive statistics for SCR Coverage Ratio
years = ['2016YE', '2017YE', '2018YE', '2019YE', '2020YE']

for year in years:
    print(f"Descriptive statistics for {year} SCR coverage ratio:")
    print(df[f'{year}_SCR_coverage_ratio'].describe())
    print("\n")
```

```
Descriptive statistics for 2016YE SCR coverage ratio:
count    3.250000e+02
mean     1.286947e+04
std      2.319518e+05
min     -1.974450e+00
25%      1.276845e+00
50%      1.662747e+00
75%      2.780175e+00
max      4.181573e+06
Name: 2016YE_SCR_coverage_ratio, dtype: float64


Descriptive statistics for 2017YE SCR coverage ratio:
count    3.250000e+02
mean     5.930313e+06
std      6.529400e+07
min     -1.973652e+00
25%      1.302423e+00
50%      1.755881e+00
75%      3.203697e+00
max      9.635840e+08
Name: 2017YE_SCR_coverage_ratio, dtype: float64


Descriptive statistics for 2018YE SCR coverage ratio:
count    3.250000e+02
mean     1.246741e+04
std      2.141325e+05
min     -5.515428e-01
25%      1.268210e+00
50%      1.693416e+00
75%      2.795907e+00
max      3.856018e+06
Name: 2018YE_SCR_coverage_ratio, dtype: float64


Descriptive statistics for 2019YE SCR coverage ratio:
count       325.000000
mean        533.017116
std        9539.236980
min          -0.669013
25%           1.177754
50%           1.710544
75%           2.691263
max      171974.690816
Name: 2019YE_SCR_coverage_ratio, dtype: float64


Descriptive statistics for 2020YE SCR coverage ratio:
count       325.000000
mean        514.215109
std        9229.786796
min          -1.066521
25%           0.000000
50%           1.565516
75%           2.367988
max      166394.575872
Name: 2020YE_SCR_coverage_ratio, dtype: float64
```

In [ ]:

```
# Reshape the df DataFrame for plotting
df_long = pd.melt(df, id_vars=['Firms_ID'], var_name='Year', value_name='SCR Cov

# Now create the boxplot across 5 year period
df_long['Year'] = df_long['Year'].str.extract('(\d{4})')

# Create a boxplot for each year
plt.figure(figsize=(10, 6))
ax = sns.boxplot(data=df_long, x='Year', y='SCR Coverage Ratio')
plt.title('SCR Coverage Ratio for Top Firms from 2016 to 2020')
plt.ylabel('SCR Coverage Ratio')
plt.xlabel('Year')
ax.set_ylim(-1000, 1000)

# Show the plot
plt.show()
```

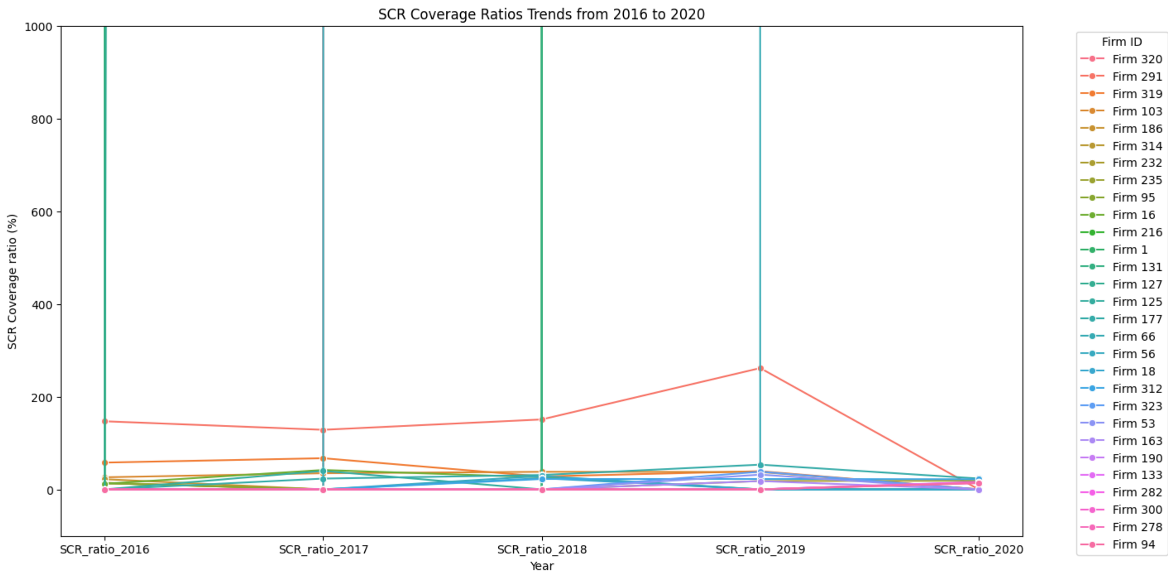
```
<>:5: SyntaxWarning: invalid escape sequence '\d'
<>:5: SyntaxWarning: invalid escape sequence '\d'
C:\Users\Syarmine\AppData\Local\Temp\ipykernel_29952\372106871.py:5: SyntaxWarnin
g: invalid escape sequence '\d'
  df_long['Year'] = df_long['Year'].str.extract('(\d{4})')
```



In [ ]:

```
# Generate a lineplot for top 10 firms across 5-year period by SCR Coverage Rati
years = range(2016, 2021)
top_firms_scratio = pd.DataFrame()

for year in years:
    scratio_col = f'{year}YE_SCR_coverage_ratio'
    if scratio_col in df.columns:
        top_10_scratio = df[['Firms_ID', scratio_col]].sort_values(by=scratio_co
        top_10_scratio.columns = ['Firms_ID', f'SCR_ratio_{year}']
        if top_firms_scratio.empty:
            top_firms_scratio = top_10_scratio
        else:
            top_firms_scratio = top_firms_scratio.merge(top_10_scratio, on='Firm
```

```
# Replace NaN values with 0 if a firm wasn't in the top 10 that year
top_firms_scratio.fillna(0, inplace=True)

# Reshape for plotting
top_firms_long = pd.melt(top_firms_scratio, id_vars=['Firms_ID'], var_name='Year

# Plot
plt.figure(figsize=(15, 8))
ax = sns.lineplot(data=top_firms_long, x='Year', y='scratio', hue='Firms_ID', ma
plt.title('SCR Coverage Ratios Trends from 2016 to 2020')
plt.xlabel('Year')
plt.ylabel('SCR Coverage ratio (%)')
plt.legend(title='Firm ID', bbox_to_anchor=(1.05, 1), loc='upper left')
ax.set_ylim(-100, 1000)
plt.show()
```



2.2 Gross Claims Incurred

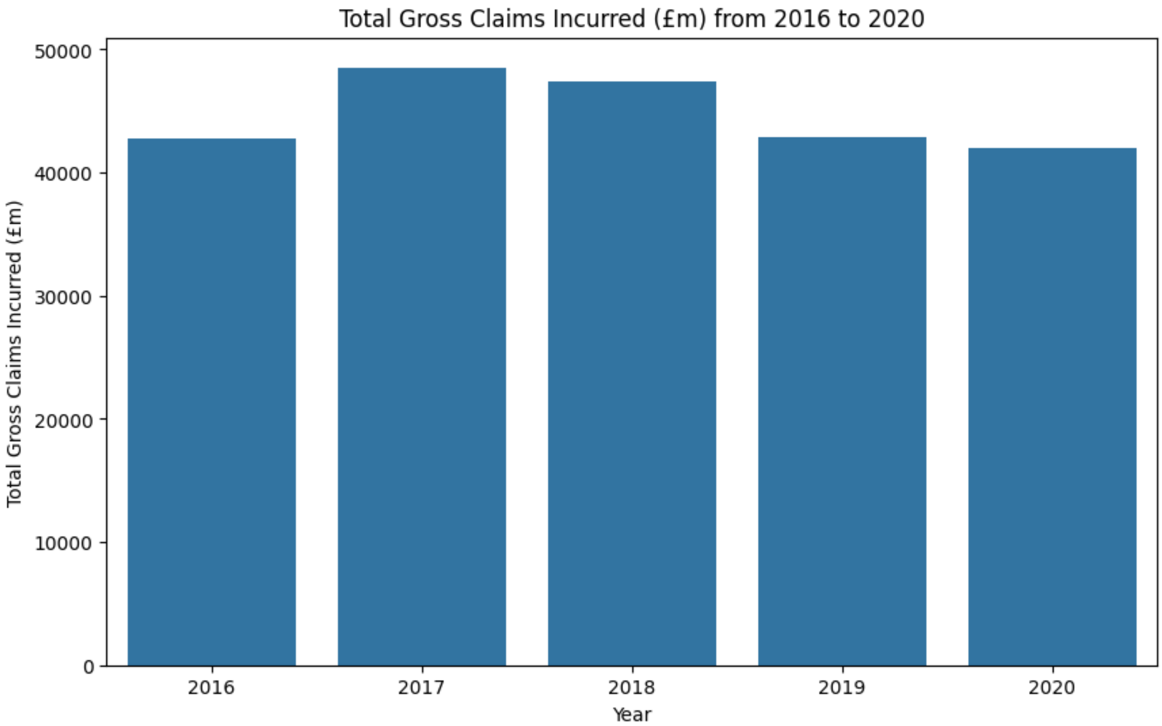
The total Gross Claims Incurred for the companies included in the dataset was 41956 (£m) at year-end 2020, a decrease of 2.1% compared to 2019.

In 2020, the majority of companies had a Gross Claims Incurred between 0.94 (£m) and 104.82 (£m), with average of 129.09 (£m) The maximum Gross Claims Incurred was 3089.46 (£m), much higher than majority of the firms.

Below are analysis shows the Gross Claims Ratio across firms included in the dataset:

In [ ]:
```
# Initialize a dictionary to store sums
sums = {}

# Loop through each year and calculate the sum
for year in range(2016, 2021):
    year_col = f'{year}YE_Gross_claims_incurred_(£m)'
    if year_col in df_1.columns:
        sums[year] = df_1[year_col].sum()

# Convert the dictionary to a DataFrame for plotting
sums_df = pd.DataFrame(list(sums.items()), columns=['Year', 'Total Gross Claims

# Creating a bar chart
```

```
plt.figure(figsize=(10, 6))
sns.barplot(data=sums_df, x='Year', y='Total Gross Claims Incurred')
plt.title('Total Gross Claims Incurred (£m) from 2016 to 2020')
plt.xlabel('Year')
plt.ylabel('Total Gross Claims Incurred (£m)')
plt.show()

# Creating the trendline
# Convert 'Year' to numeric for trendline calculation
sums_df['Year'] = pd.to_numeric(sums_df['Year'])
z = np.polyfit(sums_df['Year'], sums_df['Total Gross Claims Incurred'], 1)
p = np.poly1d(z)

# Show the plot
plt.show()
```



In [ ]:
```
years = ['2016YE', '2017YE', '2018YE', '2019YE', '2020YE']

for year in years:
    df_1[f'{year}_Gross_claims_incurred_(£m)'] = df_1[f'{year}_Gross_claims_incu
```

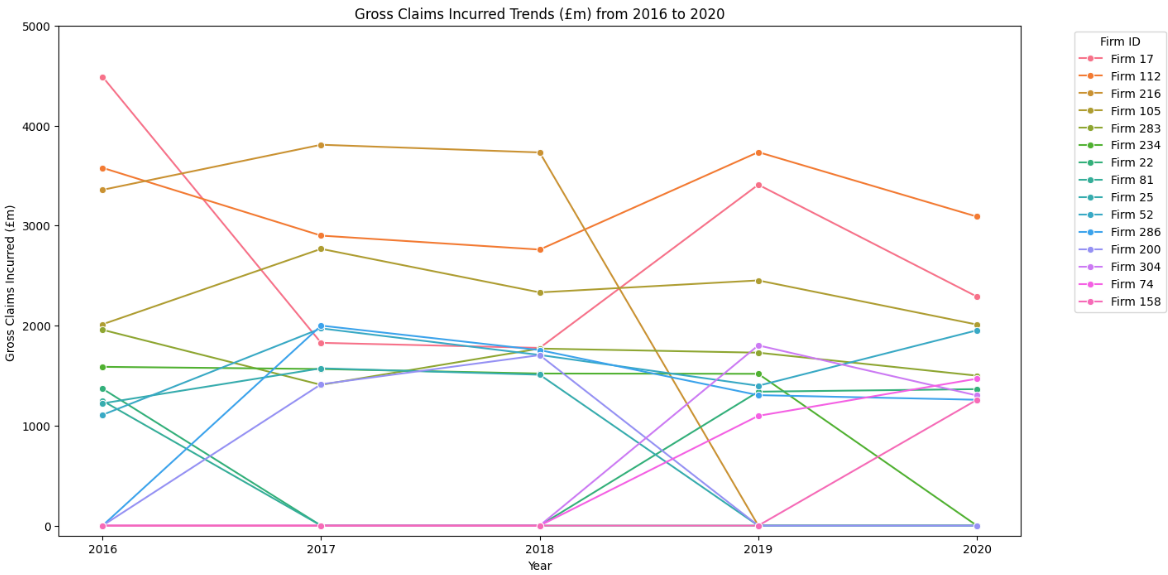
In [ ]:
```
# Descriptive statistics for Gross claims incurred
years = ['2016YE', '2017YE', '2018YE', '2019YE', '2020YE']

for year in years:
    print(f"Descriptive statistics for {year} Gross Claims Incurred:")
    print(df_1[f'{year}_Gross_claims_incurred_(£m)'].describe())
    print("\n")
```

```
Descriptive statistics for 2016YE Gross Claims Incurred:
count     325.000000
mean      131.599455
std       443.050414
min       -43.149090
25%         0.000000
50%         1.183612
75%        72.035499
max      4487.909385
Name: 2016YE_Gross_claims_incurred_(£m), dtype: float64


Descriptive statistics for 2017YE Gross Claims Incurred:
count     325.000000
mean      149.342114
std       421.151102
min       -98.000283
25%         0.000000
50%         3.233816
75%       100.381542
max      3807.986771
Name: 2017YE_Gross_claims_incurred_(£m), dtype: float64


Descriptive statistics for 2018YE Gross Claims Incurred:
count     325.000000
mean      145.787712
std       404.579776
min       -42.526820
25%         0.000000
50%         5.288307
75%        96.029417
max      3731.101331
Name: 2018YE_Gross_claims_incurred_(£m), dtype: float64


Descriptive statistics for 2019YE Gross Claims Incurred:
count     325.000000
mean      131.832772
std       392.472757
min       -18.295436
25%         0.000000
50%         1.857592
75%        87.053274
max      3734.204532
Name: 2019YE_Gross_claims_incurred_(£m), dtype: float64


Descriptive statistics for 2020YE Gross Claims Incurred:
count     325.000000
mean      129.095748
std       347.333019
min       -20.837636
25%         0.000000
50%         0.944814
75%       104.819484
max      3089.456427
Name: 2020YE_Gross_claims_incurred_(£m), dtype: float64
```

In [ ]:

```python
# Generate a lineplot for top 10 firms across 5-year period by Gross Claims Incu
years = range(2016, 2021)
top_firms_gci = pd.DataFrame()

for year in years:
    gci_col = f'{year}YE_Gross_claims_incurred_(£m)'
    if gci_col in df_1.columns:
        top_10_gci= df_1[['Firms_ID', gci_col]].sort_values(by=gci_col, ascendin
        top_10_gci.columns = ['Firms_ID', f'Gross_claims_incurred_{year}']
        if top_firms_gci.empty:
            top_firms_gci = top_10_gci
        else:
            top_firms_gci = top_firms_gci.merge(top_10_gci, on='Firms_ID', how='

# Replace NaN values with 0 if a firm wasn't in the top 10 that year
top_firms_gci.fillna(0, inplace=True)

# Reshape for plotting
top_firms_long = pd.melt(top_firms_gci, id_vars=['Firms_ID'], var_name='Year', v

# Extracting the year part for clarity
top_firms_long['Year'] = top_firms_long['Year'].str.extract('(\d{4})')

# Plot
plt.figure(figsize=(15, 8))
ax = sns.lineplot(data=top_firms_long, x='Year', y='gci', hue='Firms_ID', marker
plt.title('Gross Claims Incurred Trends (£m) from 2016 to 2020')
plt.xlabel('Year')
plt.ylabel('Gross Claims Incurred (£m) ')
plt.legend(title='Firm ID', bbox_to_anchor=(1.05, 1), loc='upper left')
ax.set_ylim(-100, 5000)
plt.show()
```

```
<>:22: SyntaxWarning: invalid escape sequence '\d'
<>:22: SyntaxWarning: invalid escape sequence '\d'
C:\Users\Syarmine\AppData\Local\Temp\ipykernel_29952\700680429.py:22: SyntaxWarni
ng: invalid escape sequence '\d'
  top_firms_long['Year'] = top_firms_long['Year'].str.extract('(\d{4})')
```



In [ ]:

```python
# Top 10 firms by Gross Claims incurred across 5 year period
print(top_firms_gci.sort_values(by='Gross_claims_incurred_2020', ascending = Fal
```

```
    Firms_ID  Gross_claims_incurred_2016  Gross_claims_incurred_2017  \
1   Firm 112                 3577.324070                 2899.569838
0    Firm 17                 4487.909385                 1827.464200
3   Firm 105                 2012.671799                 2767.075602
9    Firm 52                 1107.348203                 1972.002768
4   Firm 283                 1957.866057                 1407.502559
13   Firm 74                    0.000000                    0.000000
6    Firm 22                 1368.960775                    0.000000
12  Firm 304                    0.000000                    0.000000
10  Firm 286                    0.000000                 1999.771242
14  Firm 158                    0.000000                    0.000000

    Gross_claims_incurred_2018  Gross_claims_incurred_2019  \
1                  2759.884642                 3734.204532
0                  1776.883770                 3408.591632
3                  2331.877573                 2451.651756
9                  1706.339860                 1398.448167
4                  1770.246416                 1729.186522
13                    0.000000                 1097.479103
6                     0.000000                 1339.317443
12                    0.000000                 1802.116614
10                 1755.011077                 1304.545472
14                    0.000000                    0.000000

    Gross_claims_incurred_2020
1                  3089.456427
0                  2289.622040
3                  2010.370119
9                  1952.489045
4                  1499.450181
13                 1468.906312
6                  1364.606401
12                 1301.169879
10                 1258.303555
14                 1257.648772
```

2.3 Net Combined Ratio

The total Net Combined Ratio across firms at over 109% at year-end 2020, a decrease over 7000% compared to 2019.

In 2020, the majority of companies had a Net Combined Ratios of less than 100%, with an assumption of COVID-19 pandemic on the firms. The maximum Net Combined Ratio in the dataset was 1000%, which is an extremely high value that suggests that the company had a profitable time during the pandemic or an outliers.

Below are analysis shows the Net Combined Ratio across firms included in the dataset

In [ ]:

```
# Total of Net Combined Ratio By Years
years = ['2016', '2017', '2018', '2019', '2020']

for year in years:
    column_name = f'{year}YE_Net_combined_ratio'
    sum_value = df_1[column_name].sum() / 325
    print(f"The Total for Net Combined Ratio for Year {year} was sum of {sum_val
```

```
The Total for Net Combined Ratio for Year 2016 was sum of 1.3493484258904962
The Total for Net Combined Ratio for Year 2017 was sum of -133.04898601451444
The Total for Net Combined Ratio for Year 2018 was sum of 5.986254548991941
The Total for Net Combined Ratio for Year 2019 was sum of 7477.468647528851
The Total for Net Combined Ratio for Year 2020 was sum of 9.853370415415103
```

In [ ]:

```python
years = ['2016YE', '2017YE', '2018YE', '2019YE', '2020YE']

for year in years:
    df_1[f'{year}_Net_combined_ratio'] = df_1[f'{year}_Net_combined_ratio'].asty
```

In [ ]:

```python
# Descriptive statistics for Net Combined Ratio
years = ['2016YE', '2017YE', '2018YE', '2019YE', '2020YE']

for year in years:
    print(f"Descriptive statistics for {year} Net Combined Ratio:")
    print(df_1[f'{year}_Net_combined_ratio'].describe())
    print("\n")
```

```
Descriptive statistics for 2016YE Net Combined Ratio:
count    325.000000
mean       1.349348
std       15.126252
min     -124.288370
25%        0.000000
50%        0.327623
75%        0.959809
max      198.007110
Name: 2016YE_Net_combined_ratio, dtype: float64


Descriptive statistics for 2017YE Net Combined Ratio:
count      325.000000
mean      -133.048986
std       2560.806827
min     -46116.696842
25%          0.000000
50%          0.732771
75%          1.041880
max       1738.295847
Name: 2017YE_Net_combined_ratio, dtype: float64


Descriptive statistics for 2018YE Net Combined Ratio:
count     325.000000
mean        5.986255
std        87.892041
min        -2.013885
25%         0.000000
50%         0.793606
75%         1.005938
max      1578.853894
Name: 2018YE_Net_combined_ratio, dtype: float64


Descriptive statistics for 2019YE Net Combined Ratio:
count    3.250000e+02
mean     7.477469e+03
std      1.347934e+05
min     -2.516977e+02
25%      0.000000e+00
50%      6.178107e-01
75%      9.751001e-01
max      2.430023e+06
Name: 2019YE_Net_combined_ratio, dtype: float64


Descriptive statistics for 2020YE Net Combined Ratio:
count     325.000000
mean        9.853370
std        95.090048
min        -9.490844
25%         0.000000
50%         0.227627
75%         0.988144
max      1076.158703
Name: 2020YE_Net_combined_ratio, dtype: float64
```

In [ ]:
```
# Reshape the DataFrame for plotting
df_long2 = pd.melt(df_1, id_vars=['Firms_ID'], var_name='Year', value_name='Net

# Now create the boxplot across 5 year period
df_long2['Year'] = df_long2['Year'].str.extract('(\d{4})')

# Create a boxplot for each year
plt.figure(figsize=(10, 6))
ax = sns.boxplot(data=df_long2, x='Year', y='Net Combined Ratio')
plt.title('Net Combined Ratio for Firms from 2016 to 2020')
plt.ylabel('Net Combined Ratio')
plt.xlabel('Year')
ax.set_ylim(-5, 5)

# Show the plot
plt.show()
```

```
<>:5: SyntaxWarning: invalid escape sequence '\d'
<>:5: SyntaxWarning: invalid escape sequence '\d'
C:\Users\Syarmine\AppData\Local\Temp\ipykernel_29952\448118087.py:5: SyntaxWarnin
g: invalid escape sequence '\d'
  df_long2['Year'] = df_long2['Year'].str.extract('(\d{4})')
```