Assignment 1

January 25, 2024

Assigment 1

0.0.1 Q1 Dataset loading, cleaning & filling missing values.

[418]:
```
#!pip install missingno

# Import the necessary dependencies
import pandas as pd
import numpy as np
import missingno as mno
```

[419]:
```
#1. Load the dataset
df = pd.read_csv("/Users/xiexiaoyang/Documents/NEU learning/Spring 2024/IE6600␣
↪Visulization/Assigment/assigment 1/DSD_MCD_RY22_P06_V20_D21_BGM.csv")
```

[420]:
```
#2. Dataset inspection
#2-1 Display the head and tail of the dataset
display(df.head(5))
display(df.tail(5))

#2-2 Check the detail of the dataset
display(df.info())
display(df.describe())
```

```
            Brnd_Name                        Gnrc_Name  Tot_Mftr  \
0  8hr Arthritis Pain                     Acetaminophen         1
1  8hr Arthritis Pain                     Acetaminophen         1
2               A & D                  Vitamins A And D         1
3               A & D                  Vitamins A And D         1
4             A And D  Vits A And D/White Pet/Lanolin         1

         Mftr_Name  Tot_Spndng_2017  Tot_Dsg_Unts_2017  Tot_Clms_2017  \
0          Overall              NaN                NaN            NaN
1            Topco              NaN                NaN            NaN
2          Overall              NaN                NaN            NaN
3  Schering-Plough              NaN                NaN            NaN
4          Overall           406.33             3502.5           47.0

   Avg_Spnd_Per_Dsg_Unt_Wghtd_2017  Avg_Spnd_Per_Clm_2017  Outlier_Flag_2017  \
```

```
0                              NaN                   NaN                NaN
1                              NaN                   NaN                NaN
2                              NaN                   NaN                NaN
3                              NaN                   NaN                NaN
4                         0.116011              8.645319                1.0

   …  Avg_Spnd_Per_Clm_2020  Outlier_Flag_2020  Tot_Spndng_2021  \
0  …              9.281739                1.0           331.69
1  …              9.281739                1.0           331.69
2  …              7.866098                1.0           253.76
3  …              7.866098                1.0           253.76
4  …              7.288947                1.0           106.45

   Tot_Dsg_Unts_2021  Tot_Clms_2021  Avg_Spnd_Per_Dsg_Unt_Wghtd_2021  \
0           4642.000             54                         0.071454
1           4642.000             54                         0.071454
2           9136.000             26                         0.027776
3           9136.000             26                         0.027776
4           1032.708             14                         0.103079

   Avg_Spnd_Per_Clm_2021  Outlier_Flag_2021  Chg_Avg_Spnd_Per_Dsg_Unt_20_21  \
0               6.142407                  1                        0.469539
1               6.142407                  1                       -0.253762
2               9.760000                  1                        0.364446
3               9.760000                  1                        0.356607
4               7.603571                  1                        0.296316

   CAGR_Avg_Spnd_Per_Dsg_Unt_17_21
0                         0.047199
1                         0.047199
2                         0.594980
3                         0.594980
4                        -0.029117

[5 rows x 36 columns]

      Brnd_Name                 Gnrc_Name  Tot_Mftr        Mftr_Name  \
16141     Zyvox                 Linezolid         1  Pharmaci/Pfizer
16142     Zyvox  Linezolid In Dextrose 5%         3          Overall
16143     Zyvox  Linezolid In Dextrose 5%         1  Pharmaci/Pfizer
16144     Zyvox  Linezolid In Dextrose 5%         1  Phar-Prep/Pfize
16145     Zyvox  Linezolid In Dextrose 5%         1  Phar-Nov/Pfizer

       Tot_Spndng_2017  Tot_Dsg_Unts_2017  Tot_Clms_2017  \
16141        881336.15           52500.00          314.0
16142        516194.24         2811493.99         1308.0
16143        400349.22         2252392.99          987.0
16144         63972.27          373810.00          169.0
```

```
16145         51872.75          185291.00          152.0

       Avg_Spnd_Per_Dsg_Unt_Wghtd_2017  Avg_Spnd_Per_Clm_2017  \
16141                        90.123927            2806.803025
16142                         0.195819             394.643914
16143                         0.187088             405.622310
16144                         0.171136             378.534142
16145                         0.279953             341.268092

       Outlier_Flag_2017  …  Avg_Spnd_Per_Clm_2020  Outlier_Flag_2020  \
16141                1.0  …            1544.316444                1.0
16142                0.0  …             175.597041                0.0
16143                0.0  …             315.766394                0.0
16144                0.0  …              78.177544                0.0
16145                1.0  …              79.955617                0.0

       Tot_Spndng_2021  Tot_Dsg_Unts_2021  Tot_Clms_2021  \
16141         22141.27             2133.0             19
16142        108270.21           555519.0            613
16143         67237.88           364907.0            245
16144         17779.03           107844.0            238
16145         23253.30            82768.0            130

       Avg_Spnd_Per_Dsg_Unt_Wghtd_2021  Avg_Spnd_Per_Clm_2021  \
16141                        45.248481            1165.330000
16142                         0.202670             176.623507
16143                         0.197868             274.440327
16144                         0.164859              74.701807
16145                         0.280946             178.871538

       Outlier_Flag_2021  Chg_Avg_Spnd_Per_Dsg_Unt_20_21  \
16141                  1                        1.177785
16142                  0                        0.451434
16143                  0                        0.256644
16144                  0                        0.548867
16145                  0                        0.792195

       CAGR_Avg_Spnd_Per_Dsg_Unt_17_21
16141                        -0.158235
16142                         0.008635
16143                         0.014104
16144                        -0.009299
16145                         0.000885

[5 rows x 36 columns]

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16146 entries, 0 to 16145
```

```
Data columns (total 36 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Brnd_Name                           16146 non-null  object
 1   Gnrc_Name                           16146 non-null  object
 2   Tot_Mftr                            16146 non-null  int64
 3   Mftr_Name                           16146 non-null  object
 4   Tot_Spndng_2017                     11136 non-null  float64
 5   Tot_Dsg_Unts_2017                   11136 non-null  float64
 6   Tot_Clms_2017                       11136 non-null  float64
 7   Avg_Spnd_Per_Dsg_Unt_Wghtd_2017     11136 non-null  float64
 8   Avg_Spnd_Per_Clm_2017               11136 non-null  float64
 9   Outlier_Flag_2017                   11136 non-null  float64
 10  Tot_Spndng_2018                     12295 non-null  float64
 11  Tot_Dsg_Unts_2018                   12295 non-null  float64
 12  Tot_Clms_2018                       12295 non-null  float64
 13  Avg_Spnd_Per_Dsg_Unt_Wghtd_2018     12295 non-null  float64
 14  Avg_Spnd_Per_Clm_2018               12295 non-null  float64
 15  Outlier_Flag_2018                   12295 non-null  float64
 16  Tot_Spndng_2019                     13531 non-null  float64
 17  Tot_Dsg_Unts_2019                   13531 non-null  float64
 18  Tot_Clms_2019                       13531 non-null  float64
 19  Avg_Spnd_Per_Dsg_Unt_Wghtd_2019     13531 non-null  float64
 20  Avg_Spnd_Per_Clm_2019               13531 non-null  float64
 21  Outlier_Flag_2019                   13531 non-null  float64
 22  Tot_Spndng_2020                     14699 non-null  float64
 23  Tot_Dsg_Unts_2020                   14699 non-null  float64
 24  Tot_Clms_2020                       14699 non-null  float64
 25  Avg_Spnd_Per_Dsg_Unt_Wghtd_2020     14699 non-null  float64
 26  Avg_Spnd_Per_Clm_2020               14699 non-null  float64
 27  Outlier_Flag_2020                   14699 non-null  float64
 28  Tot_Spndng_2021                     16146 non-null  float64
 29  Tot_Dsg_Unts_2021                   16146 non-null  float64
 30  Tot_Clms_2021                       16146 non-null  int64
 31  Avg_Spnd_Per_Dsg_Unt_Wghtd_2021     16146 non-null  float64
 32  Avg_Spnd_Per_Clm_2021               16146 non-null  float64
 33  Outlier_Flag_2021                   16146 non-null  int64
 34  Chg_Avg_Spnd_Per_Dsg_Unt_20_21      14695 non-null  float64
 35  CAGR_Avg_Spnd_Per_Dsg_Unt_17_21     16121 non-null  float64
dtypes: float64(30), int64(3), object(3)
memory usage: 4.4+ MB

None

           Tot_Mftr  Tot_Spndng_2017  Tot_Dsg_Unts_2017  Tot_Clms_2017  \
count  16146.000000     1.113600e+04       1.113600e+04   1.113600e+04
mean       1.461910     1.190001e+07       7.850797e+06   1.232592e+05
std        2.289365     5.800085e+07       5.595186e+07   5.952454e+05
min        1.000000     0.000000e+00       8.400000e-02   1.100000e+01
```

```
25%        1.000000     6.250336e+04       1.370190e+04   6.060000e+02
50%        1.000000     5.597136e+05       1.537005e+05   4.382500e+03
75%        1.000000     3.503468e+06       1.586952e+06   3.574625e+04
max       44.000000     1.298196e+09       2.165630e+09   1.508415e+07

       Avg_Spnd_Per_Dsg_Unt_Wghtd_2017  Avg_Spnd_Per_Clm_2017  \
count                     11136.000000           11136.000000
mean                        135.265898            1014.855221
std                        1067.442712            4626.057958
min                           0.000000               0.000000
25%                           0.467831              19.634232
50%                           2.256165              77.634643
75%                          13.891952             306.510305
max                       33805.907721          134304.869630

       Outlier_Flag_2017  Tot_Spndng_2018  Tot_Dsg_Unts_2018  Tot_Clms_2018  \
count       11136.000000     1.229500e+04       1.229500e+04   1.229500e+04
mean            0.293552     1.075621e+07       6.619984e+06   1.104048e+05
std             0.455410     5.425716e+07       4.321680e+07   5.533821e+05
min             0.000000     0.000000e+00       1.280000e-01   1.100000e+01
25%             0.000000     4.988139e+04       1.118683e+04   4.990000e+02
50%             0.000000     4.435503e+05       1.225010e+05   3.651000e+03
75%             1.000000     2.808303e+06       1.297999e+06   2.868350e+04
max             1.000000     1.394367e+09       1.913913e+09   1.408538e+07

       …  Avg_Spnd_Per_Clm_2020  Outlier_Flag_2020  Tot_Spndng_2021  \
count  …           1.469900e+04       14699.000000     1.614600e+04
mean   …           1.477818e+03           0.333288     1.070197e+07
std    …           1.819265e+04           0.471405     6.950349e+07
min    …           0.000000e+00           0.000000     0.000000e+00
25%    …           1.767966e+01           0.000000     1.877965e+04
50%    …           7.488533e+01           0.000000     2.631212e+05
75%    …           3.585181e+02           1.000000     2.102314e+06
max    …           1.485255e+06           1.000000     2.730141e+09

       Tot_Dsg_Unts_2021  Tot_Clms_2021  Avg_Spnd_Per_Dsg_Unt_Wghtd_2021  \
count       1.614600e+04   1.614600e+04                     1.614600e+04
mean        1.503923e+07   8.505113e+04                     4.114644e+02
std         6.289203e+08   4.748528e+05                     1.469916e+04
min         4.500000e-02   1.100000e+01                     0.000000e+00
25%         4.657250e+03   1.870000e+02                     4.282419e-01
50%         5.526788e+04   1.763000e+03                     2.297355e+00
75%         7.010114e+05   1.640500e+04                     1.892245e+01
max         5.615033e+10   1.457986e+07                     1.236522e+06

       Avg_Spnd_Per_Clm_2021  Outlier_Flag_2021  \
count           1.614600e+04       16146.000000
mean            1.621154e+03           0.384306
```

```
std             1.858075e+04           0.486446
min             0.000000e+00           0.000000
25%             1.809146e+01           0.000000
50%             7.579617e+01           0.000000
75%             3.647096e+02           1.000000
max             1.511134e+06           1.000000

       Chg_Avg_Spnd_Per_Dsg_Unt_20_21  CAGR_Avg_Spnd_Per_Dsg_Unt_17_21
count                    14695.000000                     16121.000000
mean                         0.458611                         0.079144
std                          9.291121                         6.599755
min                         -1.000000                        -0.999522
25%                         -0.085346                        -0.106351
50%                          0.006944                        -0.006664
75%                          0.099577                         0.045702
max                        831.214247                       831.214247

[8 rows x 33 columns]
```

Based on the previous dataset information, we observed that the dataset consists of 36 features with three types of data: float, integer, and object. Additionally, there are occurrences of "NaN" representing missing values. The distribution of the data varies, ranging from values in the thousands to very small value.

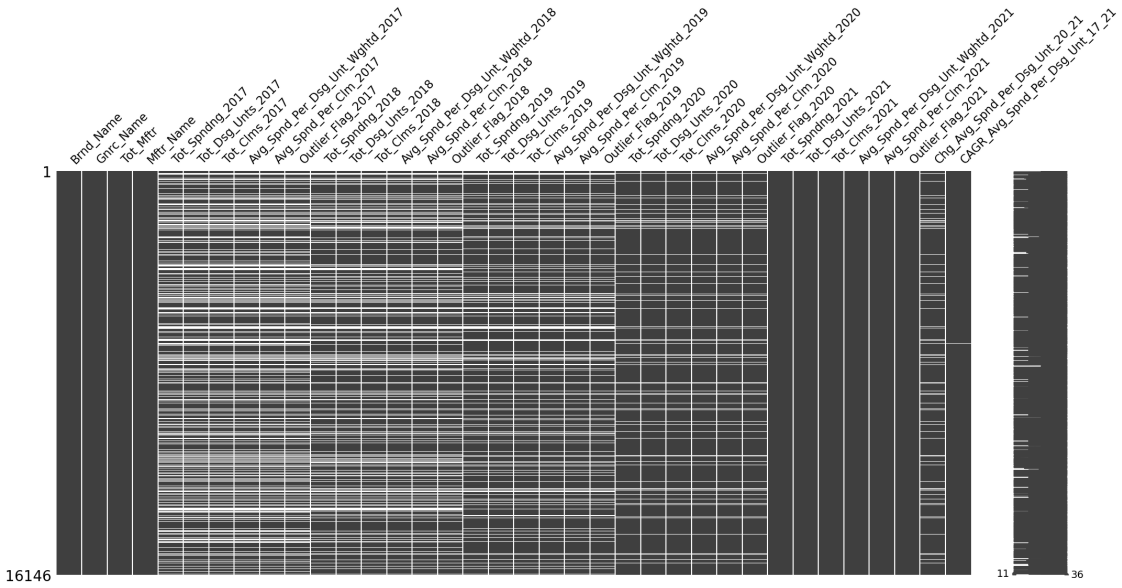
[421]:
```
#3.Data cleaning
#3-1 Check the missing value
df.isnull().sum()
```

[421]:
```
Brnd_Name                          0
Gnrc_Name                          0
Tot_Mftr                           0
Mftr_Name                          0
Tot_Spndng_2017                 5010
Tot_Dsg_Unts_2017               5010
Tot_Clms_2017                   5010
Avg_Spnd_Per_Dsg_Unt_Wghtd_2017 5010
Avg_Spnd_Per_Clm_2017           5010
Outlier_Flag_2017               5010
Tot_Spndng_2018                 3851
Tot_Dsg_Unts_2018               3851
Tot_Clms_2018                   3851
Avg_Spnd_Per_Dsg_Unt_Wghtd_2018 3851
Avg_Spnd_Per_Clm_2018           3851
Outlier_Flag_2018               3851
Tot_Spndng_2019                 2615
Tot_Dsg_Unts_2019               2615
Tot_Clms_2019                   2615
Avg_Spnd_Per_Dsg_Unt_Wghtd_2019 2615
```

```
Avg_Spnd_Per_Clm_2019                2615
Outlier_Flag_2019                    2615
Tot_Spndng_2020                      1447
Tot_Dsg_Unts_2020                    1447
Tot_Clms_2020                        1447
Avg_Spnd_Per_Dsg_Unt_Wghtd_2020      1447
Avg_Spnd_Per_Clm_2020                1447
Outlier_Flag_2020                    1447
Tot_Spndng_2021                         0
Tot_Dsg_Unts_2021                       0
Tot_Clms_2021                           0
Avg_Spnd_Per_Dsg_Unt_Wghtd_2021         0
Avg_Spnd_Per_Clm_2021                   0
Outlier_Flag_2021                       0
Chg_Avg_Spnd_Per_Dsg_Unt_20_21       1451
CAGR_Avg_Spnd_Per_Dsg_Unt_17_21        25
dtype: int64
```

[422]:
```
#Visualizing the locations of the missing data
mno.matrix(df)
```

[422]: <Axes: >



[423]:
```
#3-2 Check the duplicates
duplicates_summary_all = df[df.duplicated(keep=False)]
display(duplicates_summary_all)
```

```
Empty DataFrame
```

```
Columns: [Brnd_Name, Gnrc_Name, Tot_Mftr, Mftr_Name, Tot_Spndng_2017,␣
 ↪Tot_Dsg_Unts_2017, Tot_Clms_2017, Avg_Spnd_Per_Dsg_Unt_Wghtd_2017,␣
 ↪Avg_Spnd_Per_Clm_2017, Outlier_Flag_2017, Tot_Spndng_2018, Tot_Dsg_Unts_2018,␣
 ↪Tot_Clms_2018, Avg_Spnd_Per_Dsg_Unt_Wghtd_2018, Avg_Spnd_Per_Clm_2018,␣
 ↪Outlier_Flag_2018, Tot_Spndng_2019, Tot_Dsg_Unts_2019, Tot_Clms_2019,␣
 ↪Avg_Spnd_Per_Dsg_Unt_Wghtd_2019, Avg_Spnd_Per_Clm_2019, Outlier_Flag_2019,␣
 ↪Tot_Spndng_2020, Tot_Dsg_Unts_2020, Tot_Clms_2020,␣
 ↪Avg_Spnd_Per_Dsg_Unt_Wghtd_2020, Avg_Spnd_Per_Clm_2020, Outlier_Flag_2020,␣
 ↪Tot_Spndng_2021, Tot_Dsg_Unts_2021, Tot_Clms_2021,␣
 ↪Avg_Spnd_Per_Dsg_Unt_Wghtd_2021, Avg_Spnd_Per_Clm_2021, Outlier_Flag_2021,␣
 ↪Chg_Avg_Spnd_Per_Dsg_Unt_20_21, CAGR_Avg_Spnd_Per_Dsg_Unt_17_21]
Index: []

[0 rows x 36 columns]
```

4.Handling with missing values

Based on the summary of missing values and duplicates, it is observed that there are numerous missing values across 26 columns, with no instances of duplicates. Referring to the data dictionary, we plan to address the missing values using various techniques:

4-1. Imputing with a Constant Considering the data dictionary, we propose imputing missing values in specific columns with a constant "0." The columns to be affected include: "Tot_Spndng_20xx" "Tot_Dsg_Unts_2017" "Tot_Clms_20xx" "Avg_Spnd_Per_Dsg_Unt_Wghtd_20xx" "Avg_Spnd_Per_Clm_20xx" "Outlier_Flag_20xx" This approach signifies that there are no records, indicating an absence of spending, usage, or outliers. ##### 4-2 Handling Missing Values in "Chg_Avg_Spnd_Per_Dsg_Unt_20_21". The column "Chg_Avg_Spnd_Per_Dsg_Unt_20_21" represents the percent change in average spending per dosage unit from the prior year. To manage missing values in this column, we propose using Forward Fill, which involves propagating the last valid observation forward. ##### 4-3 Dealing with Missing Values in "CAGR_Avg_Spnd_Per_Dsg_Unt_17_21". The column "CAGR_Avg_Spnd_Per_Dsg_Unt_17_21" contains time series data with only 25 missing records. Two options are considered: (1)Exclude rows with missing values. (2)Utilize Linear Interpolation technology to impute missing values by estimating them based on adjacent non-missing values in the dataset.I used the second strategy.

[424]:
```
#4-1 Imputinng the missing value with contant "0" in the specific columns.
df.iloc[:,4:28] = df.iloc[:,4:28].fillna(0)
```

[425]:
```
#4-2 Handling Missing Values in "Chg_Avg_Spnd_Per_Dsg_Unt_20_21"with forward␣
 ↪fill
df['Chg_Avg_Spnd_Per_Dsg_Unt_20_21'] = df['Chg_Avg_Spnd_Per_Dsg_Unt_20_21'].
 ↪fillna(method='ffill')
```

[426]:
```

```

```
#4-3 Dealing with Missing Values in "CAGR_Avg_Spnd_Per_Dsg_Unt_17_21" with␣
 ↪interpolation
df["CAGR_Avg_Spnd_Per_Dsg_Unt_17_21"].
 ↪interpolate(limit_direction="both",inplace=True)
df.isnull().sum()
```

[426]:
```
Brnd_Name                          0
Gnrc_Name                          0
Tot_Mftr                           0
Mftr_Name                          0
Tot_Spndng_2017                    0
Tot_Dsg_Unts_2017                  0
Tot_Clms_2017                      0
Avg_Spnd_Per_Dsg_Unt_Wghtd_2017    0
Avg_Spnd_Per_Clm_2017              0
Outlier_Flag_2017                  0
Tot_Spndng_2018                    0
Tot_Dsg_Unts_2018                  0
Tot_Clms_2018                      0
Avg_Spnd_Per_Dsg_Unt_Wghtd_2018    0
Avg_Spnd_Per_Clm_2018              0
Outlier_Flag_2018                  0
Tot_Spndng_2019                    0
Tot_Dsg_Unts_2019                  0
Tot_Clms_2019                      0
Avg_Spnd_Per_Dsg_Unt_Wghtd_2019    0
Avg_Spnd_Per_Clm_2019              0
Outlier_Flag_2019                  0
Tot_Spndng_2020                    0
Tot_Dsg_Unts_2020                  0
Tot_Clms_2020                      0
Avg_Spnd_Per_Dsg_Unt_Wghtd_2020    0
Avg_Spnd_Per_Clm_2020              0
Outlier_Flag_2020                  0
Tot_Spndng_2021                    0
Tot_Dsg_Unts_2021                  0
Tot_Clms_2021                      0
Avg_Spnd_Per_Dsg_Unt_Wghtd_2021    0
Avg_Spnd_Per_Clm_2021              0
Outlier_Flag_2021                  0
Chg_Avg_Spnd_Per_Dsg_Unt_20_21     0
CAGR_Avg_Spnd_Per_Dsg_Unt_17_21    0
dtype: int64
```

[427]:
```
#Check for missing values in the entire DataFrame again
df.isnull().sum()
```

[427]:
```
Brnd_Name                          0
Gnrc_Name                          0
Tot_Mftr                           0
Mftr_Name                          0
Tot_Spndng_2017                    0
Tot_Dsg_Unts_2017                  0
Tot_Clms_2017                      0
Avg_Spnd_Per_Dsg_Unt_Wghtd_2017    0
Avg_Spnd_Per_Clm_2017              0
Outlier_Flag_2017                  0
Tot_Spndng_2018                    0
Tot_Dsg_Unts_2018                  0
Tot_Clms_2018                      0
Avg_Spnd_Per_Dsg_Unt_Wghtd_2018    0
Avg_Spnd_Per_Clm_2018              0
Outlier_Flag_2018                  0
Tot_Spndng_2019                    0
Tot_Dsg_Unts_2019                  0
Tot_Clms_2019                      0
Avg_Spnd_Per_Dsg_Unt_Wghtd_2019    0
Avg_Spnd_Per_Clm_2019              0
Outlier_Flag_2019                  0
Tot_Spndng_2020                    0
Tot_Dsg_Unts_2020                  0
Tot_Clms_2020                      0
Avg_Spnd_Per_Dsg_Unt_Wghtd_2020    0
Avg_Spnd_Per_Clm_2020              0
Outlier_Flag_2020                  0
Tot_Spndng_2021                    0
Tot_Dsg_Unts_2021                  0
Tot_Clms_2021                      0
Avg_Spnd_Per_Dsg_Unt_Wghtd_2021    0
Avg_Spnd_Per_Clm_2021              0
Outlier_Flag_2021                  0
Chg_Avg_Spnd_Per_Dsg_Unt_20_21     0
CAGR_Avg_Spnd_Per_Dsg_Unt_17_21    0
dtype: int64
```

0.0.2 Q2 Identify the brand with the highest total spending in 2019

[428]:
```
# 1.Group by brand and calculate the total spending for each brand in 2019
tot_spndng_2019 = df.groupby('Brnd_Name')['Tot_Spndng_2019'].sum()

# 2.Sort brands based on total spending in 2019 in descending order
sorted_tot_spndng_2019 = tot_spndng_2019.sort_values(ascending=False)

# 3.Retrieve the brand with the highest total spending in 2019
```

```
q2 = sorted_total_spending_2019.index[0]
# Display the brand with the highest total spending in 2019
print(f"The brand with the highest total spending in 2019: {q2}")
```

```
The brand with the highest total spending in 2019: Latuda
```

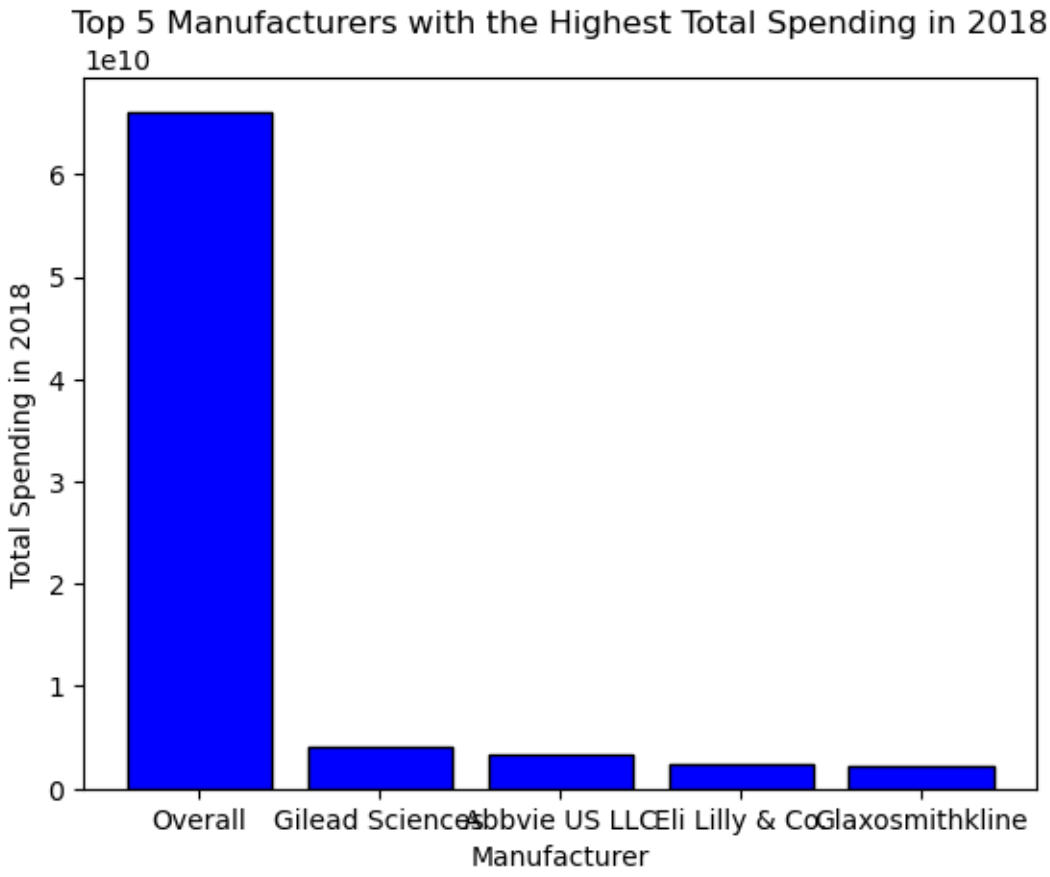
0.0.3 Q3 Generate a table showing the top 5 manufacturers with the highest total spending in 2018 and plot a histogram

[429]:
```
# Import the dependencies
import matplotlib.pyplot as plt

# 1. Generate a table showing the top 5 manufacturers with the highest total␣
 ↪spending in 2018
# Group by manufacturer and calculate total spending for each manufacturer
mftr_total_spending = df.groupby("Mftr_Name")["Tot_Spndng_2018"].sum()

# Sort manufacturers based on total spending in 2018 in descending order and␣
 ↪get the top 5 manufacturers.
top_5_mftr = mftr_total_spending.sort_values(ascending=False).head(5)

# Convert the series to DataFrame and display the table
table_q3 = top_5_mftr.to_frame()
display("The top 5 manufacturers with the highest total spending in 2018")
display(table_q3)
```

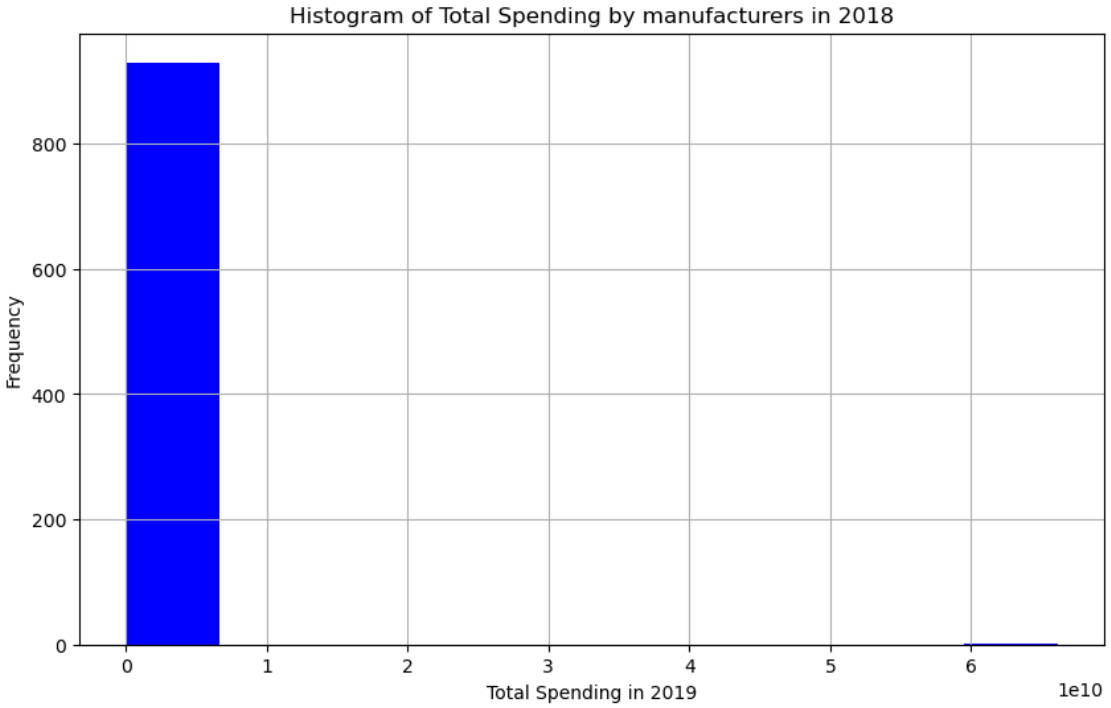
```
'The top 5 manufacturers with the highest total spending in 2018'
```

Mftr_Name	Tot_Spndng_2018
Overall	6.612381e+10
Gilead Sciences	4.041785e+09
Abbvie US LLC	3.410540e+09
Eli Lilly & Co.	2.306830e+09
Glaxosmithkline	2.269230e+09

[430]:
```
# 2-1 Since manufacturers are categorical data with only 5 records, use a bar␣
 ↪plot instead of a histogram.
plt.bar(table_q3.index, table_q3["Tot_Spndng_2018"], color='blue',␣
 ↪edgecolor='black')
plt.xlabel('Manufacturer')
plt.ylabel('Total Spending in 2018')
plt.title('Top 5 Manufacturers with the Highest Total Spending in 2018')
plt.show()
```



[431]:
```
# 2-2 Plot a histogram for the total spending in 2018 which is grouped by␣
 ↪manufacturers
plt.figure(figsize=(10, 6))
mftr_total_spending.hist(color='blue')
plt.xlabel('Total Spending in 2019')
plt.ylabel('Frequency')
plt.title('Histogram of Total Spending by manufacturers in 2018')
plt.show()
```



0.0.4 Q4 List the generic names with the highest average spending per claim in 2017.

[432]:
```
# 1.Group by generic name and calculate the total average total spending per
 ↪claim for each generic name in 2017
Avg_Spnd_Per_Clm_2017 = df.groupby('Gnrc_Name')['Avg_Spnd_Per_Clm_2017'].sum()

# 2.Sort generic names
sorted_Avg_Spnd_Per_Clm_2017 = Avg_Spnd_Per_Clm_2017.
 ↪sort_values(ascending=False)

# 3.Retrieve the top generic name with the highest average spending per claim
 ↪in 2017
q4 = sorted_Avg_Spnd_Per_Clm_2017.index[0]
print(f"Generic name with the highest average spending per claim in 2017 is:
 ↪{q4}")
```

```
Generic name with the highest average spending per claim in 2017 is: Nusinersen
Sodium/PF
```

0.0.5 Q5 Create a table displaying the brands with the highest total claims in 2018.

[433]:
```
# 1.Group by brand and calculate the total claims for each brand in 2018
total_claims_2018 = df.groupby('Brnd_Name')['Tot_Clms_2018'].sum()

# 2.Sort brands based on total claims in 2018 in a descending order
sorted_claims_2018 = total_claims_2018.sort_values(ascending=False)

# 3.Retrieve the brand with the highest total claims in 2018
brand_highest_claims_2018 = sorted_claims_2018.index[0]

# 4.Create a pivot table with the brand and total claims for the highest claims␣
 ↪brand
table_q5 = df[df['Brnd_Name'] == brand_highest_claims_2018].
 ↪pivot_table(index='Brnd_Name', values='Tot_Clms_2018', aggfunc='sum')
display(table_q5)
```

```
             Tot_Clms_2018
Brnd_Name
Amoxicillin     28170752.0
```

0.0.6 Q6 Identify the brand with the highest average spending per dosage unit in 2018

[434]:
```
# 1.Group by brand and calculate the total average spending per dosage unit for␣
 ↪each brand in 2018
Avg_Spnd_Per_Dsg_Unt_Wghtd_2018 = df.
 ↪groupby('Brnd_Name')['Avg_Spnd_Per_Dsg_Unt_Wghtd_2018'].sum()

# 2.Sort brands
sorted_Avg_Spnd_Per_Dsg_Unt_Wghtd_2018 = Avg_Spnd_Per_Dsg_Unt_Wghtd_2018.
 ↪sort_values(ascending=False)

# 3.Retrieve the brand with the highest average spending per dosage unit in 2018
q6 = sorted_Avg_Spnd_Per_Dsg_Unt_Wghtd_2018.index[0]
print(f"The brand with the highest average spending per dosage unit in 2018:␣
 ↪{q6}")
```

```
The brand with the highest average spending per dosage unit in 2018: Kymriah
```

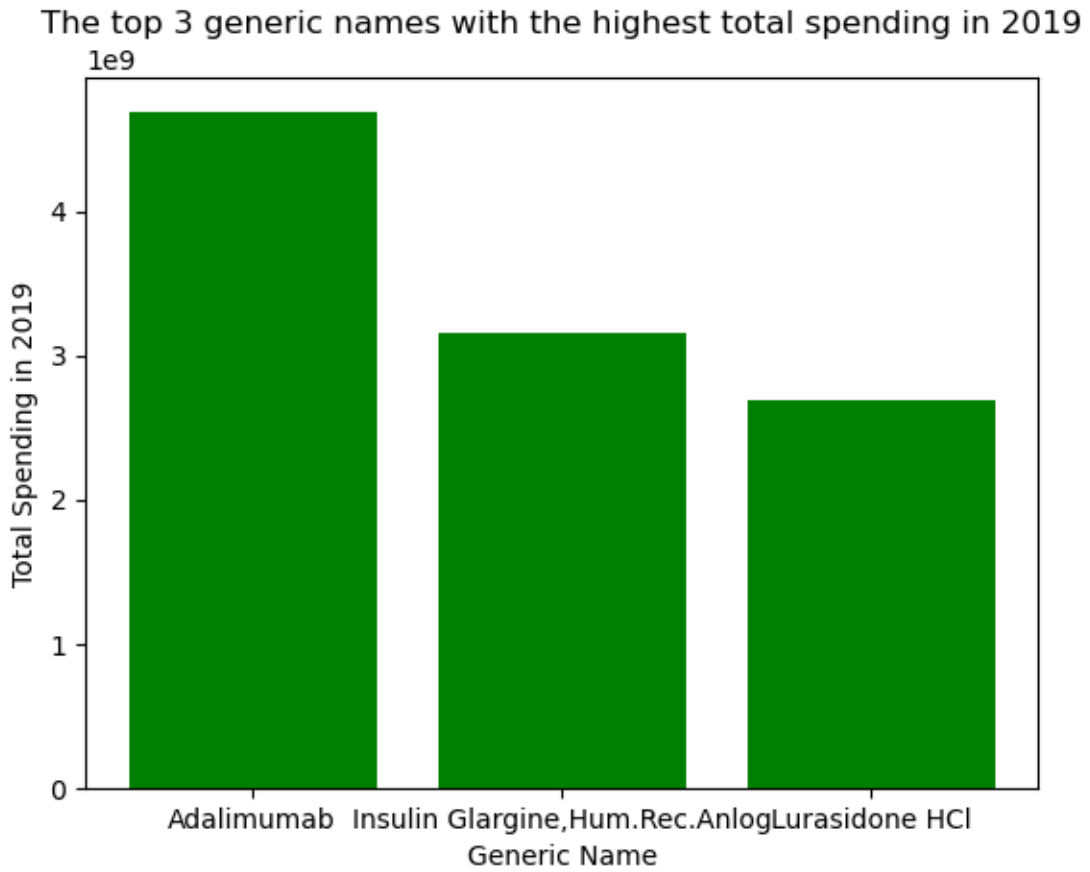
0.0.7 Q7 Create a table listing the top 3 generic names with the highest total spending in 2019? Plot a histogram using Python

[435]:
```
# 1.Group by generic name and calculate total spending for each generic name in␣
 ↪2019
generic_Tot_Spndng_2019 = df.groupby("Gnrc_Name")["Tot_Spndng_2019"].sum()

# 2.Sort generic names
```

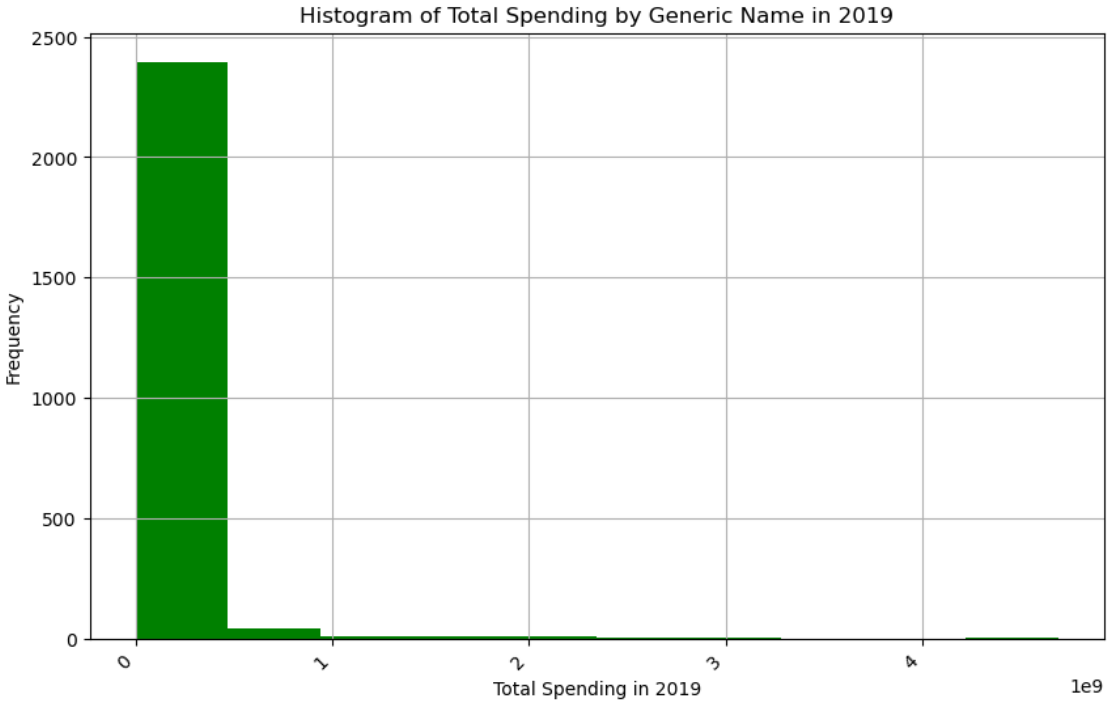
```
top_3_generic = generic_Tot_Spndng_2019.sort_values(ascending=False).head(3)

# 3.Convert the series to DataFrame and display the table
q7 = top_3_generic.to_frame()
display(q7)
```

Gnrc_Name	Tot_Spndng_2019
Adalimumab	4.690149e+09
Insulin Glargine,Hum.Rec.Anlog	3.155721e+09
Lurasidone HCl	2.692378e+09

[436]:
```
# 4.Plot a histogram by using the top3 generic name
plt.bar(q7.index, q7['Tot_Spndng_2019'], color='green')
plt.xlabel('Generic Name')
plt.ylabel('Total Spending in 2019')
plt.title('The top 3 generic names with the highest total spending in 2019')
plt.show()
```



[437]:
```
# 5.Plot a histogram for the total spending in 2019
plt.figure(figsize=(10, 6))
generic_Tot_Spndng_2019.hist(color='green')
plt.xlabel('Total Spending in 2019')
plt.ylabel('Frequency')
plt.title('Histogram of Total Spending by Generic Name in 2019')
plt.xticks(rotation=45, ha='right')
plt.show()
```



0.0.8 Q8 Build a table displaying the brands with the highest average spending per claim in 2019.

[471]:
```
# 1.Group by brand name and calculate the total average spending per claim for␣
 ↪each brand in 2019
Avg_Spnd_Per_Clm_2019 = df.groupby('Brnd_Name')['Avg_Spnd_Per_Clm_2019'].sum()

# 2.Find the highest total average spending in 2019
highest_Avg_Spnd_Per_Clm_2019 = Avg_Spnd_Per_Clm_2019.max()

# 3.Filter brands with the highest total average spending per claim in 2019
top_brands = Avg_Spnd_Per_Clm_2019[Avg_Spnd_Per_Clm_2019 ==␣
 ↪highest_Avg_Spnd_Per_Clm_2019]
```

```
# 4.Create a pivot table with the brand and total claims for the highest claims␣
 ↪brand
table_q8 = df[df['Brnd_Name'].isin(top_brands.index)].
 ↪pivot_table(index='Brnd_Name', values='Avg_Spnd_Per_Clm_2019', aggfunc='sum')
print("Brand with the highest average spending per claim in 2019")
display(table_q8)
```

```
Brand with the highest average spending per claim in 2019

           Avg_Spnd_Per_Clm_2019
Brnd_Name
Zolgensma            3435723.278
```

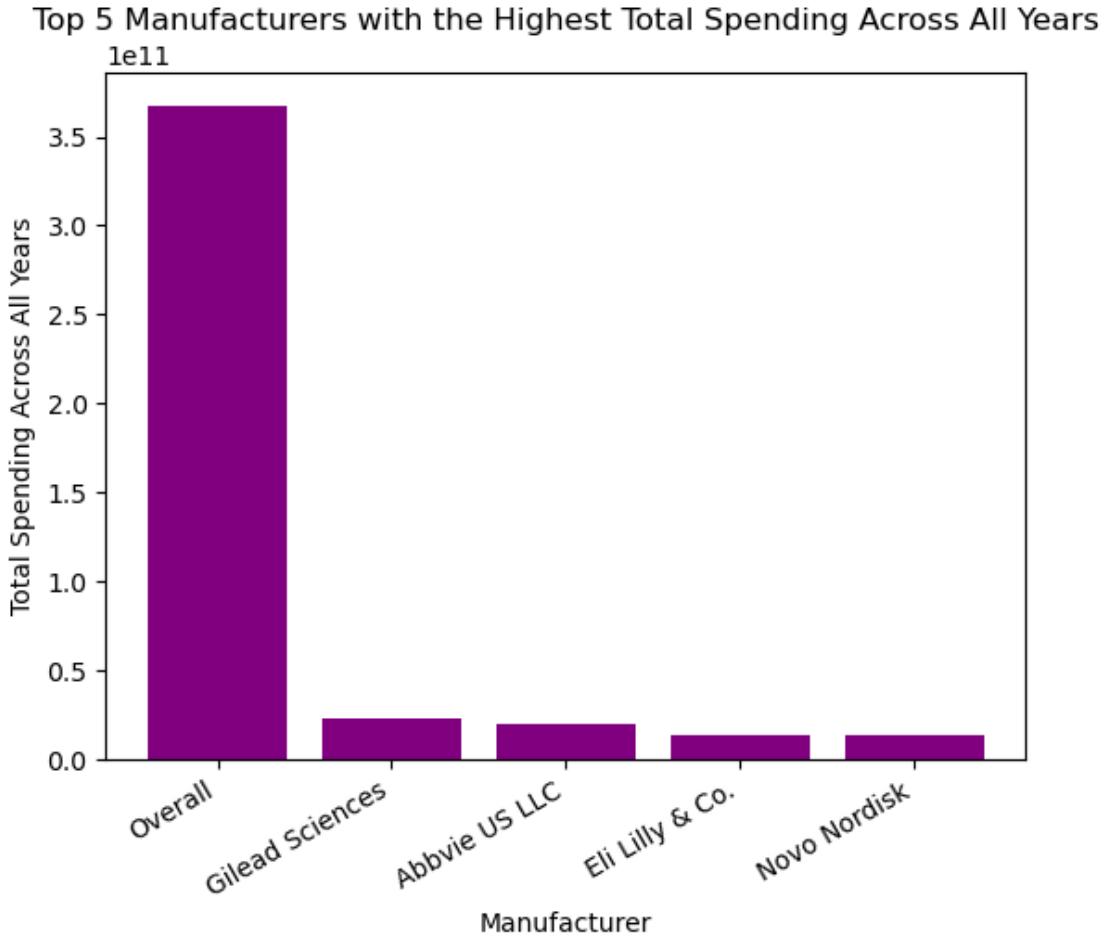
0.0.9 Q9 Generate a table showing the top 5 manufacturers with the highest total spending across all years and plot a histogram.

[439]:
```
# 1.Group by manufacturer and calculate total spending across all years.␣
 ↪Visulize the result.
mftr_total_spending_years = df.groupby("Mftr_Name")[["Tot_Spndng_2017",␣
 ↪"Tot_Spndng_2018", "Tot_Spndng_2019", "Tot_Spndng_2020", "Tot_Spndng_2021"]].
 ↪sum()
#display(mftr_total_spending_years.head())

# 2. Calculate the sum of total spending across all years. Add a new column␣
 ↪named "Total_Spending_All_Years" to store the values.
mftr_total_spending_years["Total_Spending_All_Years"] =␣
 ↪mftr_total_spending_years.sum(axis=1)
#display(mftr_total_spending_years.head(5))

# 3.Sort manufacturers by the value of total spending across all years in␣
 ↪descending order and get the top 5 rows.
top_5_mftr_all_years = mftr_total_spending_years.
 ↪sort_values(by="Total_Spending_All_Years", ascending=False).head(5)

# 4.Convert the series result to DataFrame table
table_q9 = pd.DataFrame({'Manufacturer': top_5_mftr_all_years.index,'Total␣
 ↪Spending Across All Years': top_5_mftr_all_years["Total_Spending_All_Years"].
 ↪values})
print("The top 5 manufacturers with the highest total spending across all years:
 ↪")
display(table_q9)
```

```
The top 5 manufacturers with the highest total spending across all years:

      Manufacturer  Total Spending Across All Years
0          Overall                     3.675242e+11
1  Gilead Sciences                     2.296129e+10
2    Abbvie US LLC                     1.930188e+10
```

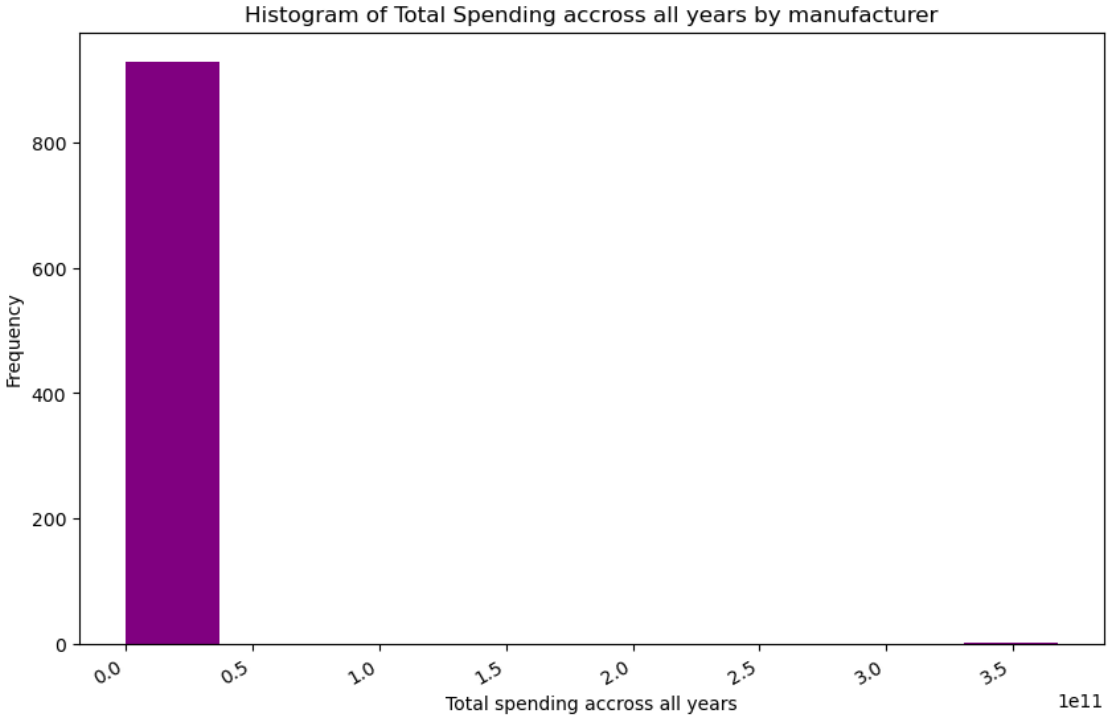
```
3  Eli Lilly & Co.                    1.320530e+10
4     Novo Nordisk                    1.287389e+10
```

[440]:
```
# 5.Plot a histogram by top 5 manufacturers with the highest total spending␣
 ↪across all years
plt.bar(table_q9['Manufacturer'], table_top_5_mftr_all_years['Total Spending␣
 ↪Across All Years'], color='purple')
plt.xlabel('Manufacturer')
plt.ylabel('Total Spending Across All Years')
plt.title('Top 5 Manufacturers with the Highest Total Spending Across All␣
 ↪Years')
plt.xticks(rotation=30, ha='right')
plt.show()
```



[441]:
```
# 6. Plot a histogram by using the all year total spending
plt.figure(figsize=(10, 6))
plt.hist(mftr_total_spending_years["Total_Spending_All_Years"],color='purple')
```

```
plt.xlabel('Total spending accross all years')
plt.ylabel('Frequency')
plt.title('Histogram of Total Spending accross all years by manufacturer')
plt.xticks(rotation=30, ha='right')
plt.show()
```



0.0.10 Q10 Identify the generic name with the highest percentage increase in total spending from 2017 to 2019.

[442]:
```
# 1.Copy the original data to a new dataframe and store the new values there.We␣
 ↪will not change the original df
# First group by generic name and calculate total spending for 2017 and 2019
total_spending_2017 = df.groupby('Gnrc_Name')['Tot_Spndng_2017'].sum()
total_spending_2019 = df.groupby('Gnrc_Name')['Tot_Spndng_2019'].sum()

 #display(total_spending_2017.head())
 #display(total_spending_2019.head())
# 2.Create a DataFrame to store the values
total_spending_df = pd.DataFrame({'Gnrc_Name': total_spending_2017.
 ↪index,'Tot_Spndng_2017': total_spending_2017.values,'Tot_Spndng_2019':␣
 ↪total_spending_2019.reindex(total_spending_2017.index, fill_value=0).values})
 #display(total_spending_df.head())
```

```
# 3.Calculate the percentage increase
# Given that there might be zero values in 2017, and will encounter division by␣
 ↪zero errors.
# We will check for non-zero value in 2017 for calculation;
# Assign nan in the zero value in 2017, which meand no records and the␣
 ↪calculation is no meaning.
total_spending_df['Percentage_Increase'] = np.where(
    total_spending_df['Tot_Spndng_2017'] != 0,
    (total_spending_df['Tot_Spndng_2019'] -␣
 ↪total_spending_df['Tot_Spndng_2017']) / total_spending_df['Tot_Spndng_2017']␣
 ↪* 100,
    np.nan)

# 4.Find the generic name with the highest percentage increase
q10 = total_spending_df.sort_values(by='Percentage_Increase',ascending=False)

print("Generic name with the highest percentage increase in total spending from␣
 ↪2017 to 2019")
display(q10.head(1))
```

```
Generic name with the highest percentage increase in total spending from 2017 to
2019
```

	Gnrc_Name	Tot_Spndng_2017	Tot_Spndng_2019	\
945	Fluticasone/Umeclidin/Vilanter	26535.36	64285801.96	

	Percentage_Increase
945	242164.668578

0.0.11 Q11 Create a table displaying the top 3 brands with the highest average spending per dosage unit weighted in 2018 and plot a histogram.

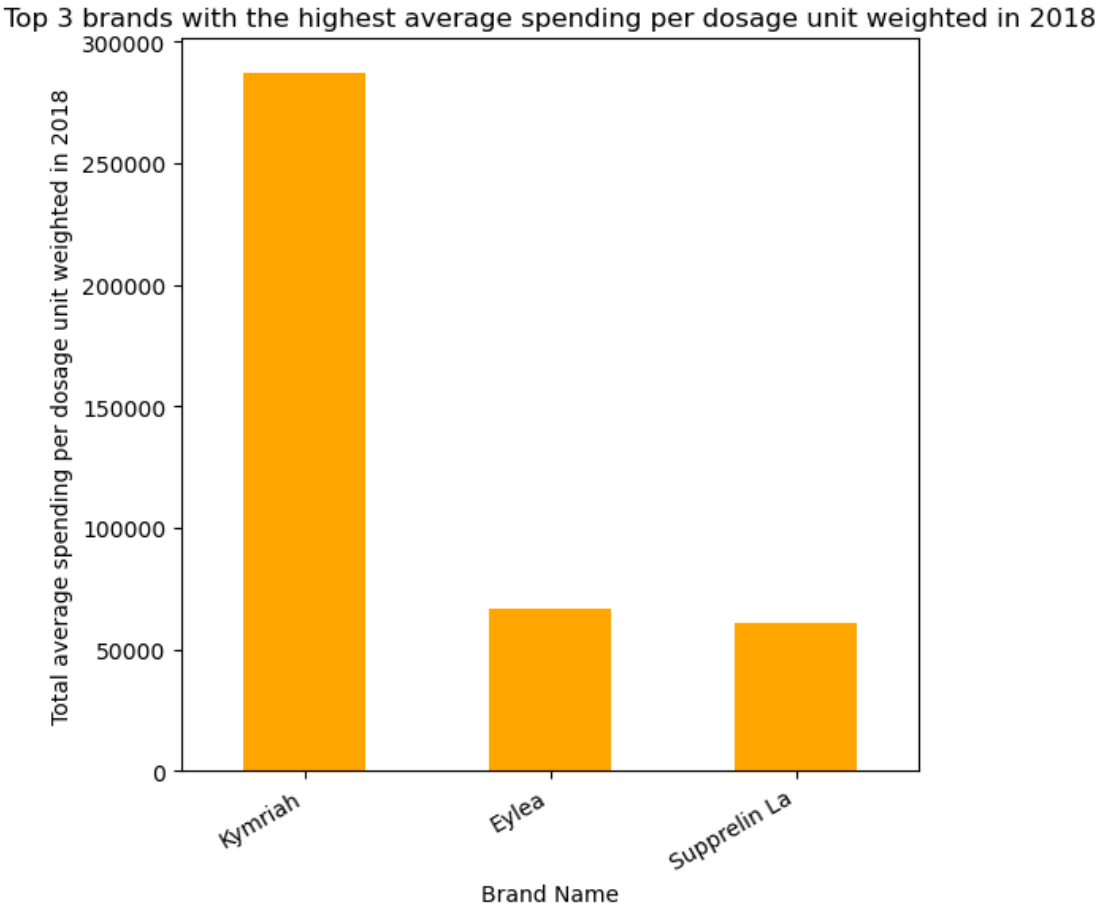
[443]:
```
# 1.Group by brand and sum the average spending per dosage unit for each brand␣
 ↪in 2018
brand_sum_ASPDUW_2018 = df.
 ↪groupby('Brnd_Name')['Avg_Spnd_Per_Dsg_Unt_Wghtd_2018'].sum()

# 2.Sort the brands by sum of the spending in a descending order
sorted_brand_sum_ASPDUW_2018 = brand_sum_ASPDUW_2018.
 ↪sort_values(ascending=False)

# 3.Get the top 3 brands with the highest average spending per dosage unit␣
 ↪weighted in 2018.
top3_brands_ASPDUW_2018 = sorted_brand_sum_ASPDUW_2018.head(3)

# 4.Convert the seriest to table
q11 = top3_brands_ASPDUW_2018.to_frame()
```

```
print("The top 3 brands with the highest average spending per dosage unit␣
 ↪weighted in 2018")
display(q11)
```

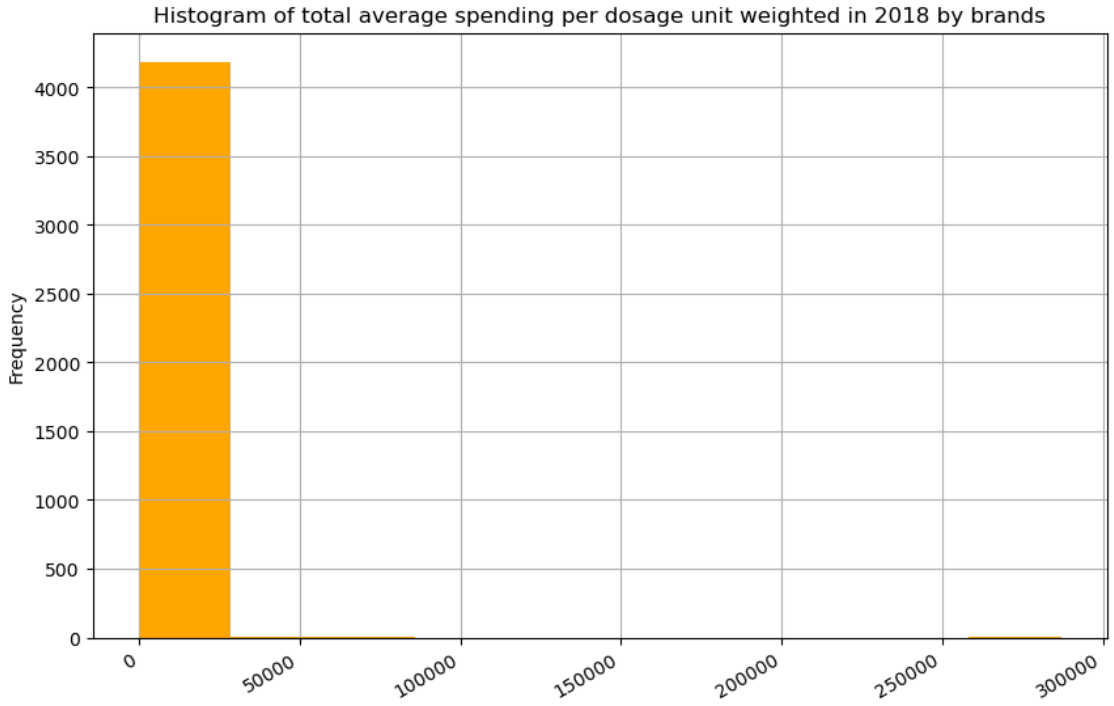
```
The top 3 brands with the highest average spending per dosage unit weighted in
2018
```

Brnd_Name	Avg_Spnd_Per_Dsg_Unt_Wghtd_2018
Kymriah	286905.171580
Eylea	66715.142344
Supprelin La	60573.203590

[444]:
```
# 5.Plot the top 3 brands
plt.figure(figsize=(6, 6))
top3_brands_ASPDUW_2018.plot(kind = "bar", x='Brand name', y='Total average␣
 ↪spending per dosage unit weighted in 2018', color='orange')
plt.xlabel('Brand Name')
plt.ylabel('Total average spending per dosage unit weighted in 2018')
plt.title('Top 3 brands with the highest average spending per dosage unit␣
 ↪weighted in 2018')
plt.xticks(rotation=30, ha='right')
plt.show()
```



[445]:
```
# 6. Plot a histogram by using the total average spending per dosage unit␣
  ↪weighted in 2018
plt.figure(figsize=(10, 6))
brand_sum_ASPDUW_2018.hist(color='orange')
plt.ylabel('Frequency')
plt.title('Histogram of total average spending per dosage unit weighted in 2018␣
  ↪by brands')
plt.xticks(rotation=30, ha='right')
plt.show()
```



0.0.12 Q12 Generate a table showing the percentage change in total claims from 2017 to 2019 for each manufacturer.

[447]:
```
# 1. Group by manufacturer and calculate total claims for each year
total_claims_2017 = df.groupby('Mftr_Name')['Tot_Clms_2017'].sum()
total_claims_2019 = df.groupby('Mftr_Name')['Tot_Clms_2019'].sum()

# 2. Create a new DataFrame with total claims for 2017 and 2019
total_claims_df = pd.DataFrame({'Mftr_Name': total_claims_2017.
 ↪index,'Tot_Clms_2017': total_claims_2017.values,
    'Tot_Clms_2019': total_claims_2019.reindex(total_claims_2017.index,␣
 ↪fill_value=0).values})

# 3.Calculate the percentage change.
# Check for non-zero value in 2017 for calculation.
# Assign nan in the zero value in 2017, which meand no records and the␣
 ↪calculation is no meaning.
total_claims_df['Percentage_Increase'] = np.where(
    total_claims_df['Tot_Clms_2017'] != 0,
    (total_claims_df['Tot_Clms_2019'] - total_claims_df['Tot_Clms_2017']) /␣
 ↪total_claims_df['Tot_Clms_2017'] * 100,np.nan)
# Reorder by descending order and convert to dataframe table.
q12 = total_claims_df.sort_values(by="Percentage_Increase", ascending=False)
```

```
print("The percentage change in total claims from 2017 to 2019 for each␣
 ↪manufacturer")
display(q12)
```

```
The percentage change in total claims from 2017 to 2019 for each manufacturer
```

	Mftr_Name	Tot_Clms_2017	Tot_Clms_2019	Percentage_Increase
15	Aci Healthcare	5728.0	2116590.0	36851.641061
246	Claris/Baxter	286.0	87247.0	30405.944056
812	Tersera Therape	38.0	8753.0	22934.210526
523	Meitheal Pharma	52.0	8412.0	16076.923077
44	Ahp*	15.0	2062.0	13646.666667
..	…	…	…	…
919	Xeris Pharmaceu	0.0	52.0	NaN
923	Y-Mabs Therapeu	0.0	0.0	NaN
925	Zealand Pharma	0.0	0.0	NaN
926	Zogenix, Inc.	0.0	0.0	NaN
930	Zylera/Cerecor	0.0	22547.0	NaN

```
[931 rows x 4 columns]
```

0.0.13 Q13 List the top 5 rows with the highest total spending in 2019 and plot a histogram.

[448]:
```
# 1.Sort the DataFrame by 'Tot_Spndng_2019' in descending order and get the top␣
 ↪5 rows.
q13 = df.sort_values(by='Tot_Spndng_2019', ascending=False).head(5)

# 2.List all columns of the top 5 rows.
print("Top 5 rows with the highest total spending in 2019")
display(q13)
```

```
Top 5 rows with the highest total spending in 2019
```

	Brnd_Name	Gnrc_Name	Tot_Mftr	\
8172	Latuda	Lurasidone HCl	1	
8173	Latuda	Lurasidone HCl	1	
1647	Biktarvy	Bictegrav/Emtricit/Tenofov Ala	1	
1648	Biktarvy	Bictegrav/Emtricit/Tenofov Ala	1	
7543	Invega Sustenna	Paliperidone Palmitate	1	

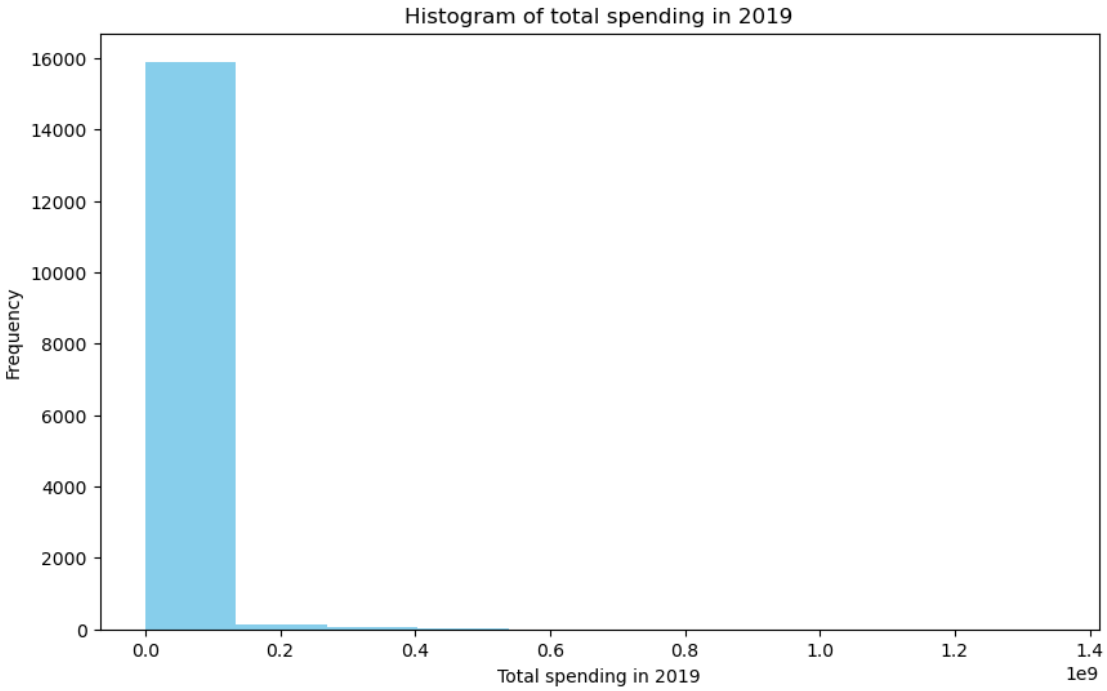
	Mftr_Name	Tot_Spndng_2017	Tot_Dsg_Unts_2017	Tot_Clms_2017	\
8172	Overall	1.105327e+09	2.961813e+07	970531.0	
8173	Sunovion Pharma	1.105327e+09	2.961813e+07	970531.0	
1647	Overall	0.000000e+00	0.000000e+00	0.0	
1648	Gilead Sciences	0.000000e+00	0.000000e+00	0.0	
7543	Janssen Pharm.	7.545156e+08	4.926159e+05	407806.0	

	Avg_Spnd_Per_Dsg_Unt_Wghtd_2017	Avg_Spnd_Per_Clm_2017	\

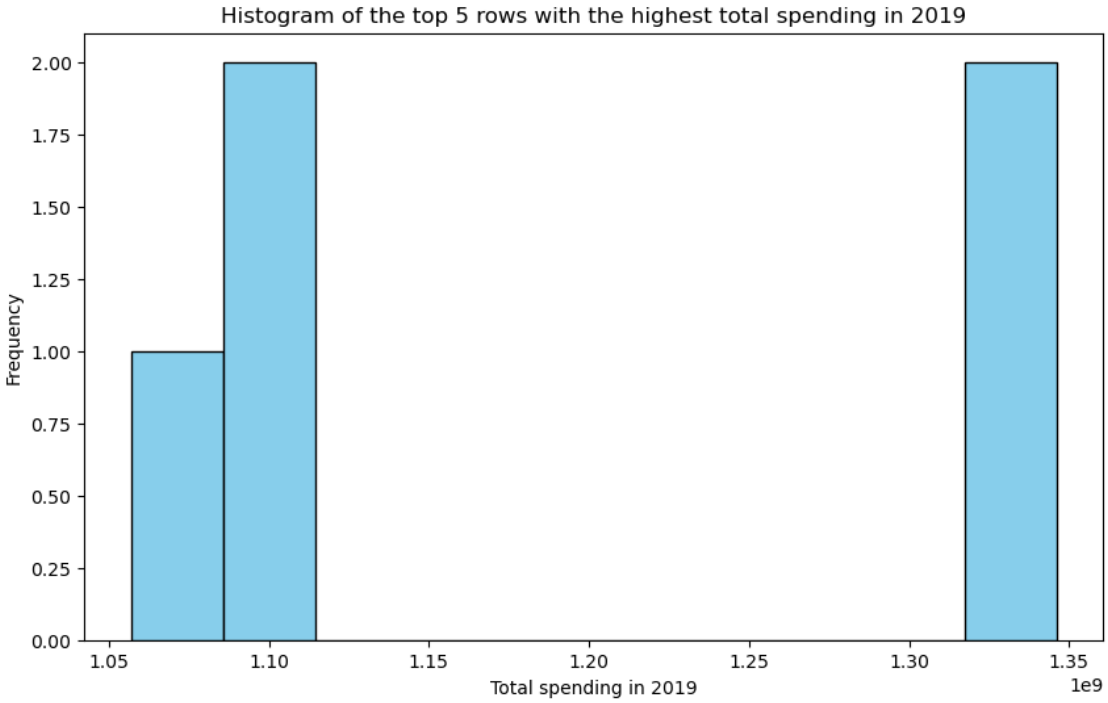
```
8172                        37.411275            1138.888885
8173                        37.411275            1138.888885
1647                         0.000000               0.000000
1648                         0.000000               0.000000
7543                      1528.399509            1850.182707

      Outlier_Flag_2017  …  Avg_Spnd_Per_Clm_2020  Outlier_Flag_2020  \
8172                0.0  …            1329.122472                0.0
8173                0.0  …            1329.122472                0.0
1647                0.0  …            3294.671118                0.0
1648                0.0  …            3294.671118                0.0
7543                0.0  …            2211.417694                0.0

      Tot_Spndng_2021  Tot_Dsg_Unts_2021  Tot_Clms_2021  \
8172    1.640908e+09        36647950.74        1177929
8173    1.640908e+09        36647950.74        1177929
1647    2.346595e+09        21467095.00         670545
1648    2.346595e+09        21467095.00         670545
7543    1.297126e+09          692739.21         559869

      Avg_Spnd_Per_Dsg_Unt_Wghtd_2021  Avg_Spnd_Per_Clm_2021  \
8172                        44.845127            1393.044650
8173                        44.845127            1393.044650
1647                       109.311236            3499.533489
1648                       109.311236            3499.533489
7543                      1869.206868            2316.837837

      Outlier_Flag_2021  Chg_Avg_Spnd_Per_Dsg_Unt_20_21  \
8172                  0                        0.041481
8173                  0                        0.041363
1647                  0                        0.042454
1648                  0                        0.042454
7543                  0                        0.043198

      CAGR_Avg_Spnd_Per_Dsg_Unt_17_21
8172                         0.046353
8173                         0.046353
1647                         0.064873
1648                         0.064873
7543                         0.051611

[5 rows x 36 columns]
```

[449]:
```
# 3.Plot a histogram using 'Tot_Spndng_2019'
plt.figure(figsize=(10, 6))
plt.hist(df['Tot_Spndng_2019'], color='skyblue')
plt.xlabel('Total spending in 2019')
```

```
plt.ylabel('Frequency')
plt.title('Histogram of total spending in 2019')
plt.show()
```



[450]:
```
# plot the top 5 rows
plt.figure(figsize=(10, 6))
plt.hist(q13['Tot_Spndng_2019'], color='skyblue',edgecolor='black')
plt.xlabel('Total spending in 2019')
plt.ylabel('Frequency')
plt.title('Histogram of the top 5 rows with the highest total spending in 2019')
plt.show()
```



0.0.14 Q14 Build a table listing the top 3 rows with the highest average spending per claim in 2018 and plot a histogram.

[451]:
```
# 1.Sort the DataFrame by 'Avg_Spnd_Per_Clm_2018' in descending order and get␣
 ↪the top 3 rows.
q14 = df.sort_values(by='Avg_Spnd_Per_Clm_2018', ascending=False).head(3)

# 2.Build a dataframe table
print("The top 3 rows with the highest average spending per claim in 2018")
display(q14)
```

The top 3 rows with the highest average spending per claim in 2018

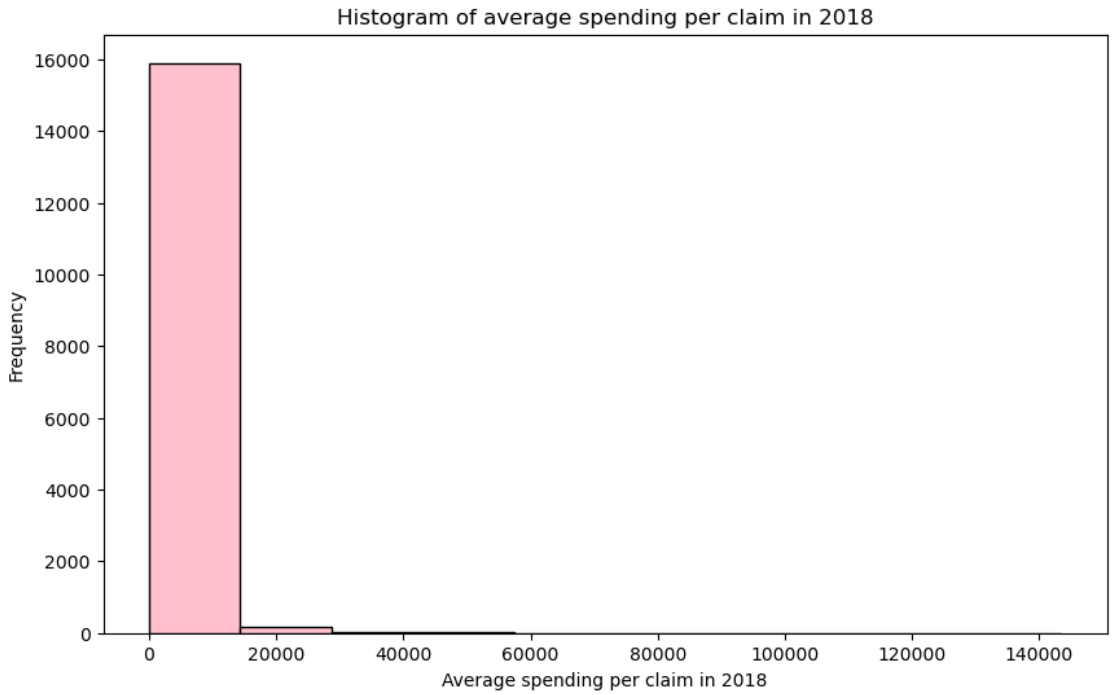
	Brnd_Name	Gnrc_Name	Tot_Mftr	Mftr_Name	Tot_Spndng_2017	\
7974	Kymriah	Tisagenlecleucel	1	Novartis	0.000000e+00	
7973	Kymriah	Tisagenlecleucel	1	Overall	0.000000e+00	
13675	Spinraza	Nusinersen Sodium/PF	1	Biogen-Idec	1.318874e+08	

	Tot_Dsg_Unts_2017	Tot_Clms_2017	Avg_Spnd_Per_Dsg_Unt_Wghtd_2017	\
7974	0.000	0.0	0.000000	
7973	0.000	0.0	0.000000	
13675	5718.499	982.0	23063.286709	

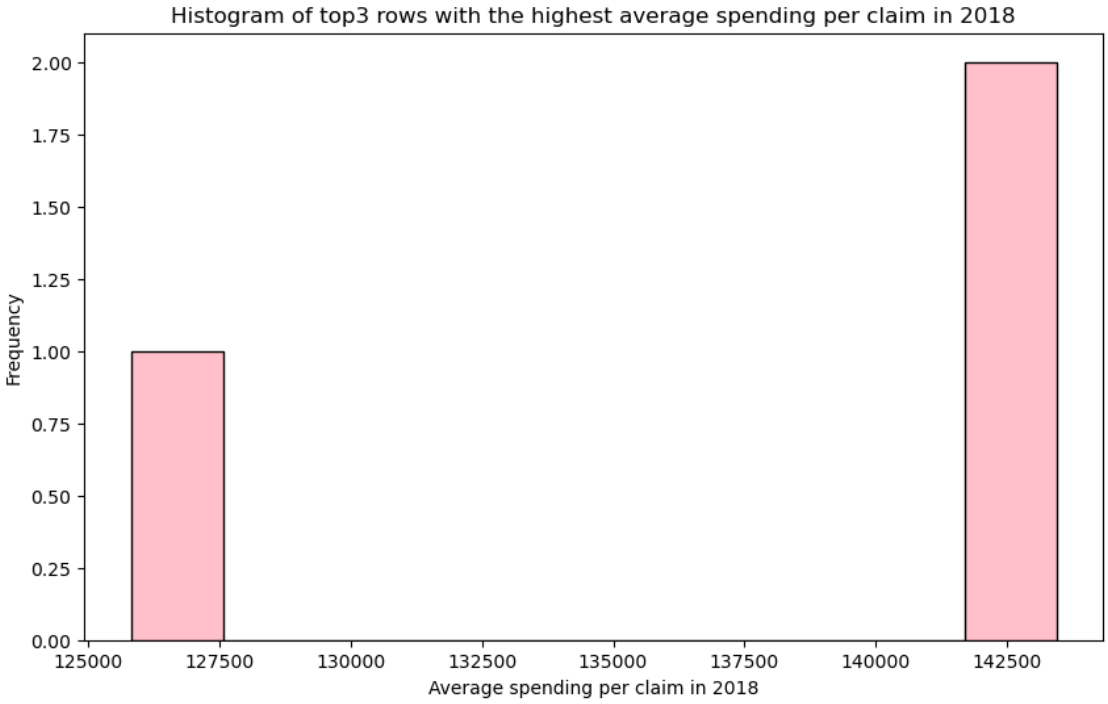
	Avg_Spnd_Per_Clm_2017	Outlier_Flag_2017	…	Avg_Spnd_Per_Clm_2020	\
7974	0.00000	0.0	…	0.00000	

```
7973                 0.00000                0.0  …                0.00000
13675           134304.86963                0.0  …           118446.05743

       Outlier_Flag_2020  Tot_Spndng_2021  Tot_Dsg_Unts_2021  Tot_Clms_2021  \
7974                 0.0     1.011257e+07             23.000             22
7973                 0.0     1.011257e+07             23.000             22
13675                1.0     1.502210e+08           7770.627           1269

       Avg_Spnd_Per_Dsg_Unt_Wghtd_2021  Avg_Spnd_Per_Clm_2021  \
7974                       439676.993040           459662.31091
7973                       439676.993040           459662.31091
13675                       19331.899672           118377.44803

       Outlier_Flag_2021  Chg_Avg_Spnd_Per_Dsg_Unt_20_21  \
7974                   1                        0.014030
7973                   1                        0.014030
13675                  1                       -0.129184

       CAGR_Avg_Spnd_Per_Dsg_Unt_17_21
7974                          0.452586
7973                          0.452586
13675                        -0.043162

[3 rows x 36 columns]
```

[452]:
```python
# 3.Plot a histogram using 'Avg_Spnd_Per_Clm_2018'
plt.figure(figsize=(10, 6))
plt.hist(df['Avg_Spnd_Per_Clm_2018'], color='pink', edgecolor='black')
plt.xlabel('Average spending per claim in 2018')
plt.ylabel('Frequency')
plt.title('Histogram of average spending per claim in 2018')
plt.show()
```



[453]:
```
# 4.plot the top 3 rows
plt.figure(figsize=(10, 6))
plt.hist(q14['Avg_Spnd_Per_Clm_2018'], color='pink',edgecolor='black')
plt.xlabel('Average spending per claim in 2018')
plt.ylabel('Frequency')
plt.title('Histogram of top3 rows with the highest average spending per claim␣
  ↪in 2018')
plt.show()
```



0.0.15 Q15 Identify the manufacturer with the highest total spending in 2017, considering only brands with a positive trend in total claims from 2017 to 2018.

[454]:
```
# 1.Calculate total claims for each brand in 2017 and 2018
tot_clms_2017 = df.groupby('Brnd_Name')['Tot_Clms_2017'].sum()
tot_clms_2018 = df.groupby('Brnd_Name')['Tot_Clms_2018'].sum()

# 2. Find out the brands with a postive trend in total claims from 2017 to 2018
positive_brands = tot_clms_2018[tot_clms_2018 > tot_clms_2017].index

# 3.Filter the brands with postive trend in total spending 2017
brands_positive_spndng = df[df['Brnd_Name'].
 ↪isin(positive_brands)][['Mftr_Name', 'Tot_Spndng_2017']]

# 4.Calculate total spending for each manufacturer with the brands with postive␣
 ↪trend.
mftr_tot_spndng_positive = brands_positive_spndng.
 ↪groupby('Mftr_Name')['Tot_Spndng_2017'].sum()

# 5.Find the manufacturer with the highest total spending in 2017
q15 = mftr_tot_spndng_positive.idxmax()
print(f"The Mmanufacturer with the highest total spending in 2017 for brands␣
 ↪with a positive trend in total claims from 2017 to 2018 is: {q15}")
```

The Mmanufacturer with the highest total spending in 2017 for brands with a positive trend in total claims from 2017 to 2018 is: Overall

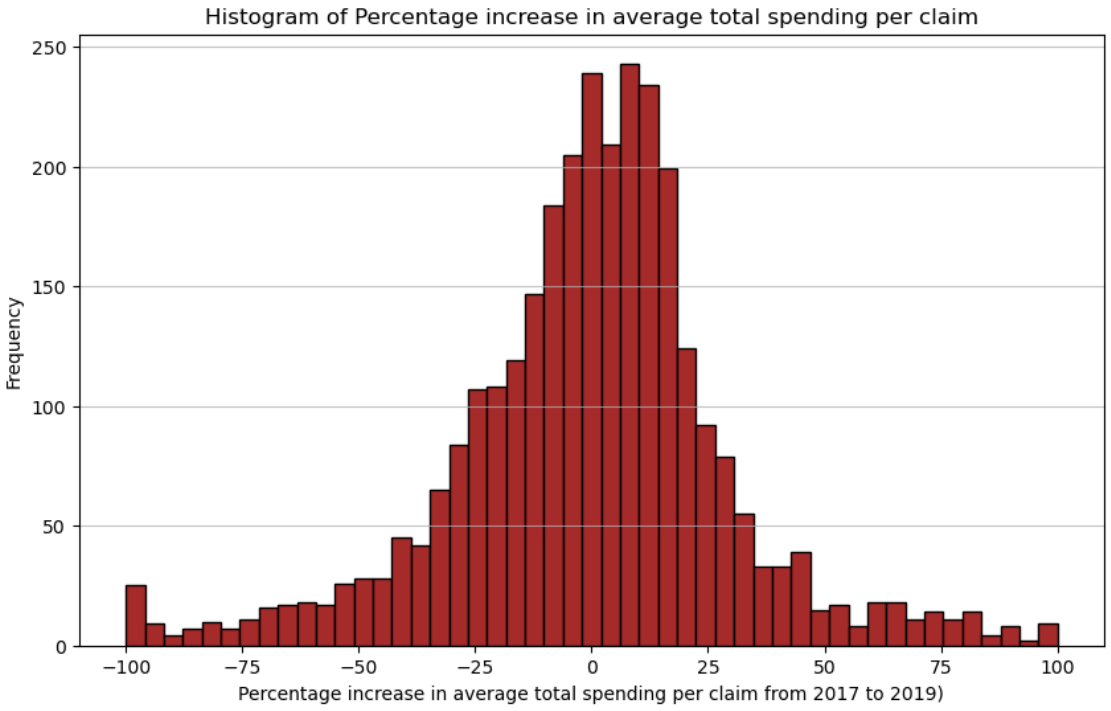
0.0.16 Q16 Create a table displaying the top 3 brands with the highest percentage increase in average spending per claim from 2017 to 2019 and plot a histogram.

[455]:
```
# 1.Calculate average total spending per claim for each brand in 2017 and 2019
avg_spnd_per_clm_2017 = df.groupby('Brnd_Name')['Avg_Spnd_Per_Clm_2017'].mean()
avg_spnd_per_clm_2019 = df.groupby('Brnd_Name')['Avg_Spnd_Per_Clm_2019'].mean()

# 2.Make a new dataframe to store the value
avg_spending_df = pd.DataFrame({
    'Brnd_Name': avg_spnd_per_clm_2017.index,
    'Avg_Spnd_Per_Clm_2017': avg_spnd_per_clm_2017.values,
    'Avg_Spnd_Per_Clm_2019': avg_spnd_per_clm_2019.
 ↪reindex(avg_spnd_per_clm_2017.index, fill_value=0).values})

# 3.Calculate the percentage increase and store the values in a new dataframe
# Check for non-zero value in 2017 for calculation.
# Assign nan in the zero value in 2017, which meand no records and the␣
 ↪calculation is no meaning.
avg_spending_df['Percentage_Increase'] = np.where(
    avg_spending_df['Avg_Spnd_Per_Clm_2017'] != 0,
    (avg_spending_df['Avg_Spnd_Per_Clm_2019'] -␣
 ↪avg_spending_df['Avg_Spnd_Per_Clm_2017']) /␣
 ↪avg_spending_df['Avg_Spnd_Per_Clm_2017'] * 100,np.nan)

# 4.Identify the top 3 brands with the highest percentage increase
q16 = avg_spending_df.sort_values(by='Percentage_Increase', ascending=False).
 ↪head(3)
print("The top 3 brands with the highest percentage increase in average␣
 ↪spending per claim from 2017 to 2019")
display(q16)
```

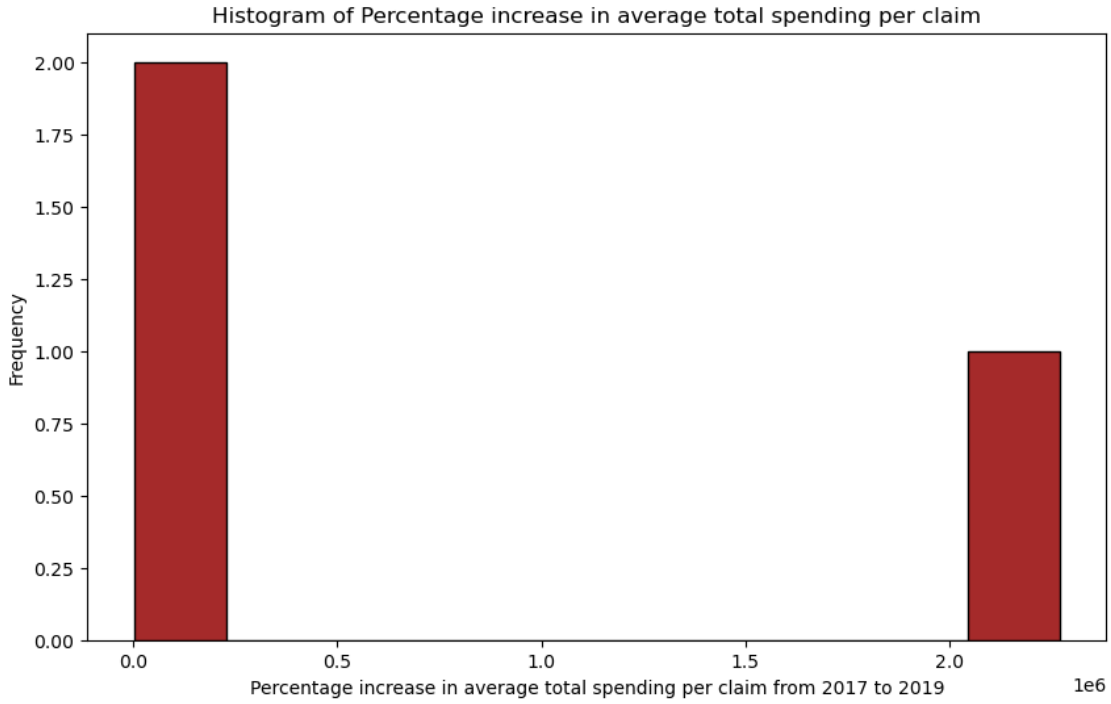
The top 3 brands with the highest percentage increase in average spending per claim from 2017 to 2019

	Brnd_Name	Avg_Spnd_Per_Clm_2017	Avg_Spnd_Per_Clm_2019	Percentage_Increase
3484	Succinylcholine Chloride	0.004961	112.583815	2.269168e+06
1028	Dextrose 5%-Potassium Chloride	9.614500	267.896667	2.686382e+03
2912	Phytonadione*	21.717346	591.808110	2.625048e+03

[456]:
```
# 5.Plot a histogram showing the percentage increase from 2017 to 2019
plt.figure(figsize=(10, 6))
plt.hist(avg_spending_df['Percentage_Increase'], color='brown',␣
 ↪edgecolor='black',bins = "auto",range=[-100, 100])
plt.xlabel('Percentage increase in average total spending per claim from 2017␣
 ↪to 2019)')
plt.ylabel('Frequency')
plt.title('Histogram of Percentage increase in average total spending per␣
 ↪claim')
plt.grid(axis='y', alpha=0.75)
plt.show()
```



[457]:
```
# 6.plot the top 3 rows
plt.figure(figsize=(10, 6))
plt.hist(q16['Percentage_Increase'], color='brown',edgecolor='black')
plt.xlabel('Percentage increase in average total spending per claim from 2017␣
 ↪to 2019')
plt.ylabel('Frequency')
plt.title('Histogram of Percentage increase in average total spending per␣
 ↪claim')
plt.show()
```



0.0.17 Q17 Identify the brand with the highest average spending per claim in 2019, considering only brands with a decreasing trend in total spending from 2018 to 2019.

[458]:
```
# 1.Calculate total spending for each brand in 2018 and 2019
tot_snpng_2018 = df.groupby('Brnd_Name')['Tot_Spndng_2018'].sum()
tot_snpng_2019 = df.groupby('Brnd_Name')['Tot_Spndng_2019'].sum()

# 2. Find out the brands with a decreasing trend, which the difference beween␣
 ↪2019 and 2018 is negative.
negative_brands = tot_snpng_2019[tot_snpng_2019 - tot_snpng_2018 < 0].index

# 3.Filter the brands with decreasing trend in average total spending per claim␣
 ↪in 2019
brands_dec_spndng_per_clms = df[df['Brnd_Name'].
 ↪isin(negative_brands)][['Brnd_Name', 'Avg_Spnd_Per_Clm_2019']]

# 4.Calculate total average total spending per claim for each brands in the␣
 ↪brands with decreasing trend.
brand_avg_tot_spndng_per_clms_dec = brands_dec_spndng_per_clms.
 ↪groupby('Brnd_Name')['Avg_Spnd_Per_Clm_2019'].sum()

# 5.Find the brand with the highest average total spending per claim in 2019
q17 = brand_avg_tot_spndng_per_clms_dec.idxmax()
```

```
print(f"The brand with the highest average total spending per claim in 2019␣
 ↪with a decreasing trend in total spending from 2018 to 2019 is: {q17}")
```

```
The brand with the highest average total spending per claim in 2019 with a
decreasing trend in total spending from 2018 to 2019 is: Spinraza
```

0.0.18 Q18 Create a table displaying the top 5 manufacturers with the highest average spending per dosage unit in 2018, but exclude manufacturers with an outlier flag and plot a histogram.

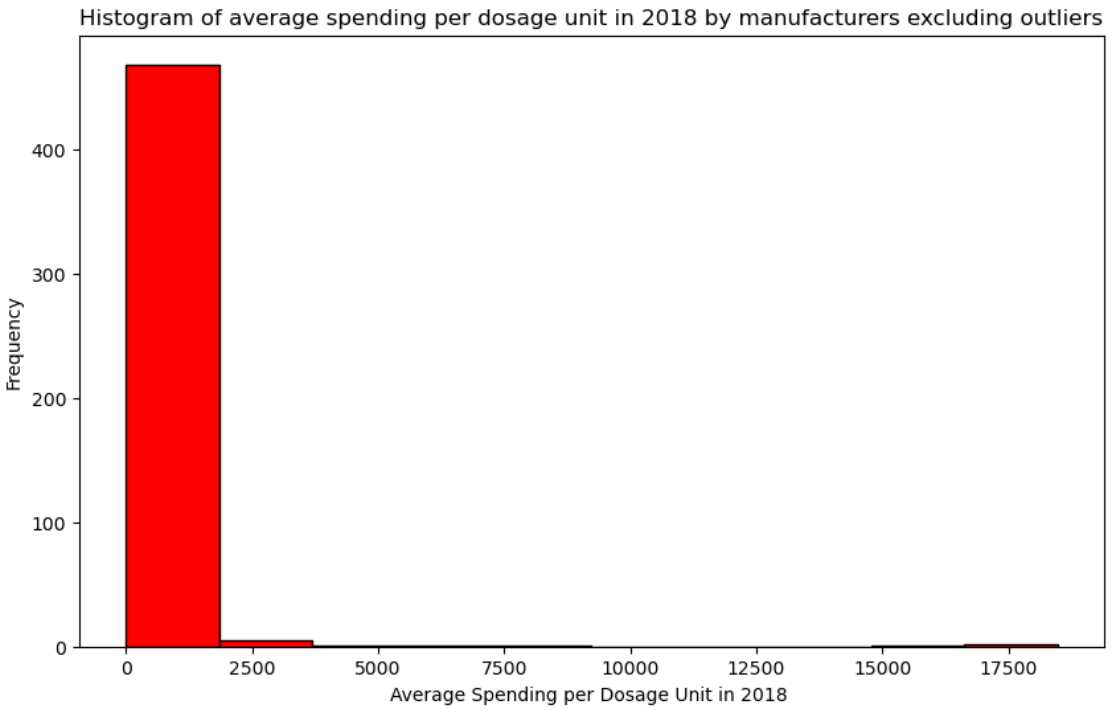
[459]:
```
# 1.Filter out manufacturers with an outlier flag, which the value is equal to␣
 ↪1.
# There might be records where mftr has a corresponding 'Outlier_Flag_2018'␣
 ↪value of 0 in some rows while value of 1 in other rows.
# I will filters out rows where any row for a specific manufacturer has␣
 ↪'Outlier_Flag_2018' equal to 1.
mfrt_no_outlier = df[df.groupby('Mftr_Name')['Outlier_Flag_2018'].
 ↪transform('max') != 1]

# 2.Group by manufacturer and calculate 2018 average total spending per dosage␣
 ↪unit.
mftr_Avg_Spnd_Per_Dsg_Unt_Wghtd_2018 = mfrt_no_outlier.
 ↪groupby('Mftr_Name')['Avg_Spnd_Per_Dsg_Unt_Wghtd_2018'].sum()

# 3.Sort by average total spending in descending order and get the top 5 rows
q18 = mftr_Avg_Spnd_Per_Dsg_Unt_Wghtd_2018.to_frame().
 ↪sort_values(by='Avg_Spnd_Per_Dsg_Unt_Wghtd_2018', ascending=False).head(5)

# 4.Create a dataframe table
print("The top 5 manufacturers with the highest average spending per dosage␣
 ↪unit in 2018 excluding outliers:")
display(q18)
```

```
The top 5 manufacturers with the highest average spending per dosage unit in
2018 excluding outliers:
```

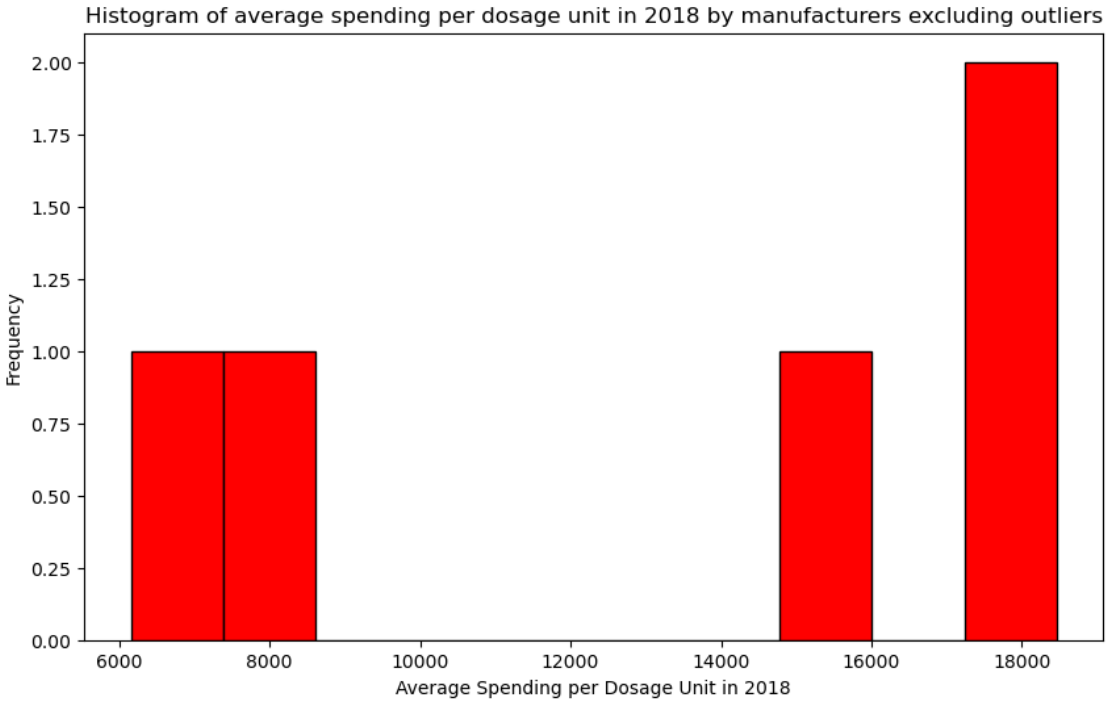
Mftr_Name	Avg_Spnd_Per_Dsg_Unt_Wghtd_2018
Janssen Biotech	18465.064612
Janssen Biotech*	17638.657610
Ipsen Biopharma	15380.250319
Alimera Science	7457.071075
Santarus/Pharmi	6152.826116

[460]:
```
# 5.Plot a histogram
plt.figure(figsize=(10, 6))
plt.hist(mftr_Avg_Spnd_Per_Dsg_Unt_Wghtd_2018, color='red', edgecolor='black')
plt.xlabel('Average Spending per Dosage Unit in 2018 ')
```

```
plt.ylabel('Frequency')
plt.title('Histogram of average spending per dosage unit in 2018 by␣
 ↪manufacturers excluding outliers')
plt.show()
```



[461]:
```
# Plot the top 5 manusfacturer
plt.figure(figsize=(10, 6))
plt.hist(q18['Avg_Spnd_Per_Dsg_Unt_Wghtd_2018'], color='red',edgecolor='black')
plt.xlabel('Average Spending per Dosage Unit in 2018')
plt.ylabel('Frequency')
plt.title('Histogram of average spending per dosage unit in 2018 by␣
 ↪manufacturers excluding outliers')
plt.show()
```



0.0.19 Q19 Find manufacturers that have names with an odd number of characters. Calculate the total spending for brands associated with these manufacturers in 2019 and list the top 3 manufacturers.

[462]:
```
# 1.Filter manufacturers with names having an odd number of characters.
# Define a function to find the odd number of characters.
def odd_character(mftr):
    return len(mftr) % 2 != 0

# Find the manusfacturers have names with an odd number of characters.
odd_number_mftr = df[df['Mftr_Name'].apply(odd_character)]
#display(odd_number_mftr)

# 2.Group by manufacturer and calculate the total spending for each brand in␣
 ↪2019
q19_1 = odd_length_manufacturers.groupby(['Mftr_Name',␣
 ↪'Brnd_Name'])['Tot_Spndng_2019'].sum()

# Convert it to a dataframe table
q19_1_table = q19_1.to_frame()
print("The total spending for brands associated with these manufacturers:")
display(q19_1_table)

# 3.Find the top 3 manufacturers based on total spending in 2019
```

```
# Group by the manufacturer and calculate 2019 total spending for each␣
 ↪manufacturer.
mftr_tot_spndng_2019 = q19_1_table.groupby('Mftr_Name')['Tot_Spndng_2019'].sum()

# Sort the value by 2019 total spending and get the tope 3 manufacturer
q19_2 = mftr_tot_spndng_2019.to_frame().sort_values(by="Tot_Spndng_2019",␣
 ↪ascending=False).head(3)

# Display the result
print("The top 3 manufacturer names with odd number of characters with the␣
 ↪highest total spending in 2019:")
display(q19_2)
```

The total spending for brands associated with these manufacturers:

		Tot_Spndng_2019
Mftr_Name	Brnd_Name	
Abbott Hosp	Morphine Sulfate*	0.000000e+00
Abbott Nutritio	Pedialyte	5.958500e+02
Abbvie US LLC	Androgel	1.238481e+07
	Creon	2.740195e+08
	Depakote	6.238449e+06
…		…
Zydus Pharmaceu	Venlafaxine HCl	1.309840e+06
	Venlafaxine HCl ER	1.036006e+07
	Verapamil HCl*	0.000000e+00
	Warfarin Sodium	3.108480e+03
	Zolmitriptan ODT	1.330990e+05

[12992 rows x 1 columns]

The top 3 manufacturer names with odd number of characters with the highest total spending in 2019:

Mftr_Name	Tot_Spndng_2019
Overall	7.207820e+10
Gilead Sciences	4.401205e+09
Abbvie US LLC	3.900690e+09

0.0.20 Q20 Identify the manufacturer with the highest total spending across all brands in 2018. List the top 3 brands associated with this manufacturer

[463]:
```
# 1.Group by manufacturer and calculate the total spending for each␣
 ↪manufacturer in 2018
tot_spndng_by_mftr_2018 = df.groupby('Mftr_Name')['Tot_Spndng_2018'].sum()

# 2.Find the manufacturer with the highest total spending in 2018
mftr_highest_tot_2018 = total_spending_by_manufacturer_2018.idxmax()
```

```
print(f"Manufacturer with the highest total spending in 2018 is:␣
 ↪{mftr_highest_tot_2018}")
print("-------------------------------------------------------------------------------------")

# 3.Filter data for the top 1 manufacturer in 2018
df_mftr_highest_tot_2018 = df[df['Mftr_Name'] == top_manufacturer_2018]
#display(df_mftr_highest_tot_2018)

# 4.Calculate the 2018 total spending for each brands asscicated with this␣
 ↪manufacturer.

brand_tot_2018_top_mftr = df_mftr_highest_tot_2018.
 ↪groupby('Brnd_Name')['Tot_Spndng_2018'].sum()

# 5.List the top 3 brands

q20 = brand_tot_2018_top_mftr.to_frame().
 ↪sort_values(by='Tot_Spndng_2018',ascending=False).head(3)
print("Top 3 brands associated with this manufacturer are:")
display(q20)
```

```
Manufacturer with the highest total spending in 2018 is: Overall
-------------------------------------------------------------------------------
----
Top 3 brands associated with this manufacturer are:
```

Brnd_Name	Tot_Spndng_2018
Humira Pen	1.394367e+09
Latuda	1.250847e+09
Mavyret	1.025846e+09

0.0.21 Q21 For each of the top 3 brands identified in Q20, calculate the percentage increase in average spending per claim from 2018 to 2019.

[464]:
```
# 1.Filter the data for the top 3 brands in 2018
top_3_brand_df = df[df['Brnd_Name'].isin(q20.index)]
#display(top_3_brand_df )

# 2.Calculate the percentage increase and assign nan in the zero value in 2018.
# Create a new dataframe to store the columns data will be used.
total_to3_brand = top_3_brand_df.groupby(['Brnd_Name']).agg({
    'Avg_Spnd_Per_Clm_2018': 'sum',
    'Avg_Spnd_Per_Clm_2019': 'sum'})
# Calculation and store the new value to a new column
total_to3_brand['Percentage_Increase'] = np.where(
    total_to3_brand['Avg_Spnd_Per_Clm_2018'] != 0,
```

```
    (total_to3_brand['Avg_Spnd_Per_Clm_2019'] -␣
 ↪total_to3_brand['Avg_Spnd_Per_Clm_2018']) /␣
 ↪total_to3_brand['Avg_Spnd_Per_Clm_2018'] * 100,np.nan)

print("The percentage increase in average spending per claim from 2018 to 2019␣
 ↪for each of the top 3 brands:")
display(total_to3_brand.reset_index()[['Brnd_Name', 'Percentage_Increase']])
```

The percentage increase in average spending per claim from 2018 to 2019 for each of the top 3 brands:

	Brnd_Name	Percentage_Increase
0	Humira Pen	12.890820
1	Latuda	1.114984
2	Mavyret	9.727944

0.0.22 22 Among the brands from Q21 that have a positive percentage increase in average spending per claim, find the brand with the highest total spending in 2019.

[465]:
```
# 1.Filter brands with positive percentage increase
positive_brands = total_to3_brand[total_to3_brand['Percentage_Increase'] > 0]
#display(positive_brands)

# 2.Filter the data of these brands from the original dataframe.
positive_brands_df = df.loc[df['Brnd_Name'].
 ↪isin(total_to3_brand[total_to3_brand['Percentage_Increase'] > 0].index)]
#display(positive_brands_df)

# 3.Calculate the 2019 total spending for each brand
tot_positive_brands = positive_brands_df.
 ↪groupby("Brnd_Name")["Tot_Spndng_2019"].sum()
# Convert to dataframe
tot_positive_brands_df = tot_positive_brands.to_frame()

# 3.Find the brand with the highest total spending in 2019
q22 = tot_positive_brands_df.sort_values(by="Tot_Spndng_2019", ascending=False).
 ↪head(1)
print("Brand with the highest total spending in 2019 among brands with positive␣
 ↪percentage increase in average spending per claim from 2018 to 2019:")
display(q22)
```

Brand with the highest total spending in 2019 among brands with positive percentage increase in average spending per claim from 2018 to 2019:

Brnd_Name	Tot_Spndng_2019
Latuda	2.692378e+09

0.0.23 Q23 Top 3 manufacturers with names starting with 'A' or 'a', based on total spending for Brands in 2018

[466]:
```
# 1.Filter the manufacturer with names start with A or a
filtered_mftr = df[df['Mftr_Name'].str.startswith(('A', 'a'))]

# 2.Calcute the 2018 total spending for each brands.
# Since each brands belong to many manufacturer, we should group by the mftr␣
 ↪too.
brands_tot_2018 = filtered_mftr.groupby(['Mftr_Name',␣
 ↪'Brnd_Name'])['Tot_Spndng_2018'].sum().reset_index()

# 3.Sort the value of 2018 total spending
sorted_brands_tot_2018 = brands_tot_2018.sort_values(by='Tot_Spndng_2018',␣
 ↪ascending=False)

# 4.Select the top 3 manufacturers based on total spending
print("Top 3 manufacturers with names starting with 'A' or 'a' based on total␣
 ↪spending for Brands in 2018:")
display(sorted_brands_tot_2018.head(3))
```

Top 3 manufacturers with names starting with 'A' or 'a' based on total spending for Brands in 2018:

	Mftr_Name	Brnd_Name	Tot_Spndng_2018
10	Abbvie US LLC	Humira Pen	1.394367e+09
25	Abbvie US LLC	Mavyret	1.025846e+09
2009	Astrazeneca	Symbicort	5.629596e+08

0.0.24 Q24 Identify the top 5 generic drugs based on the total number of dosage units in 2019 and plot a histogram.

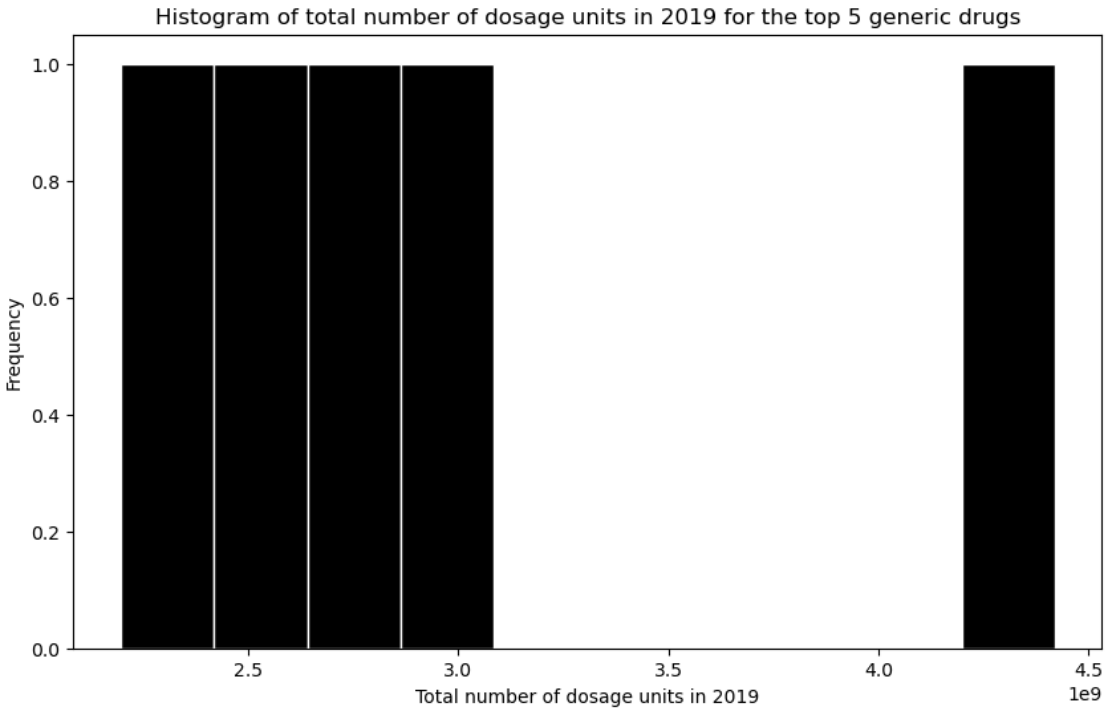
[467]:
```
# 1.Group by generic drug and calculate the 2019 total number of dosage units
generic_tot_unts_2019 = df.groupby('Gnrc_Name')['Tot_Dsg_Unts_2019'].sum()

# 2.Sort in descending order
sorted_generic_tot_unts_2019 = generic_tot_unts_2019.to_frame().
 ↪sort_values(by='Tot_Dsg_Unts_2019', ascending=False)

# 3. Get the top 5
q24 = sorted_generic_tot_unts_2019.head(5)

print("The top 5 generic drugs based on the total number of dosage units in␣
 ↪2019 are:")
display(q24)
```

The top 5 generic drugs based on the total number of dosage units in 2019 are:

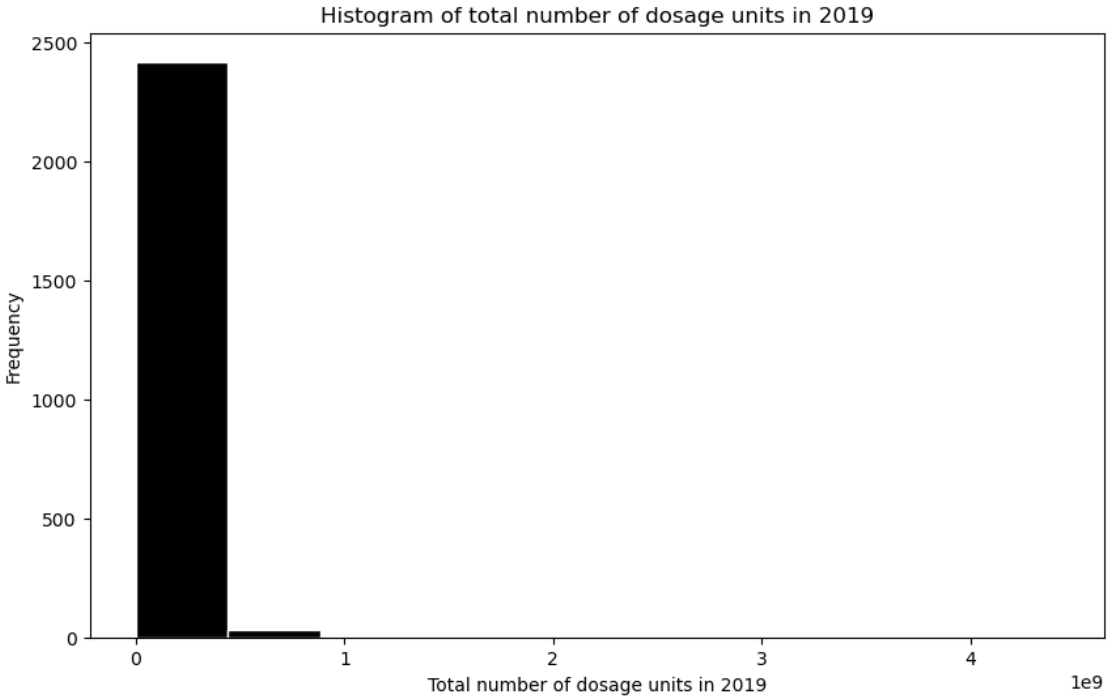
	Tot_Dsg_Unts_2019

```
Gnrc_Name
0.9 % Sodium Chloride                4.420203e+09
Amoxicillin                          2.889964e+09
PEG3350/Sod Sulf,Bicarb,Cl/KCl       2.815912e+09
Gabapentin                           2.562289e+09
Ibuprofen                            2.195557e+09
```

[468]:
```
# 4.Plot a histogram for the top 5 generic drugs
plt.figure(figsize=(10, 6))
plt.hist(q24['Tot_Dsg_Unts_2019'], color='black',edgecolor='white')
plt.xlabel('Total number of dosage units in 2019')
plt.ylabel('Frequency')
plt.title('Histogram of total number of dosage units in 2019 for the top 5␣
 ↪generic drugs')
plt.show()
```



[469]:
```
# plot a histogram for all total number of dosage units in 2019
plt.figure(figsize=(10, 6))
plt.hist(generic_tot_unts_2019, color='black', edgecolor='white')
plt.xlabel('Total number of dosage units in 2019')
plt.ylabel('Frequency')
plt.title('Histogram of total number of dosage units in 2019')
plt.show()
```


Histogram of total number of dosage units in 2019
Frequency
2500
2000
1500
1000
500
0
0
1
2
3
4
Total number of dosage units in 2019
1e9