In [1]:
```
#        ASSIGNMENT
# MINI-PROJECT MILESTONE- DATA CLEANING

###
NAME:  OLUWASEGUN AJAYI
CLASS: DATA ANALYTICS COHORT 12
###
```

In [ ]:
```
# QUESTION ONE:
# 1. import the data into your jupyter notebook and analyze it in detail.
```

In [7]:
```
import pandas as pd
fileName = "cancerdata.csv"

filePath ="C:/Users/OLUWASEGUN/Documents/Myworkspace/"
data = pd.read_csv(filePath + fileName)
```

In [8]:
```
#     INTERPRETATION OF RESULT
# This will the return the top first-Seven rows/sample
# This tells me the nature and structure of my data and
# I have a mixture of numerics and non-numerics meaning that I have some coding to do i
# The data contains some missing values  too (as depicted with NaN).

data.head(7)
```

Out[8]:

	Unnamed: 0	avgAnnCount	avgDeathsPerYear	TARGET_deathRate	incidenceRate	medIncome	popEst2
0	0	1397.0	469.0	164.9	489.8	61898.0	2601
1	1	173.0	70.0	161.3	411.6	48127.0	
2	2	NaN	50.0	174.7	349.7	49348.0	210
3	3	427.0	NaN	194.8	430.4	44243.0	758
4	4	57.0	NaN	144.4	350.1	49955.0	103
5	5	428.0	152.0	176.0	505.4	NaN	
6	6	250.0	97.0	175.9	461.8	37782.0	415

7 rows × 35 columns

In [9]:
```
#     INTERPRETATION OF RESULT
# This will the return the seven last or bottom rows/ samples
# This shows that I have about 3047 samples
```

```
data.tail(7)
```

Out[9]:

	Unnamed: 0	avgAnnCount	avgDeathsPerYear	TARGET_deathRate	incidenceRate	medIncome	pop
3040	3040	1962.667684	NaN	215.9	453.549422	NaN	
3041	3041	1962.667684	48.0	NaN	453.549422	NaN	
3042	3042	1962.667684	15.0	NaN	453.549422	46961.0	
3043	3043	1962.667684	43.0	150.1	453.549422	48609.0	
3044	3044	NaN	46.0	153.9	453.549422	NaN	
3045	3045	1962.667684	52.0	175.0	453.549422	50745.0	
3046	3046	1962.667684	48.0	213.6	453.549422	41193.0	

7 rows × 35 columns

In [ ]:

In [ ]:
```
# QUESTION TWO
# 2. Identify and report on the structure and any problems you find in the data
```

In [10]:
```
#         INTERPRETATION OF RESULT
# TO UNDERSTAND THE DISTRIBUTION OF THE CANCER DATA

# This reveals the data type (float values, integers and object), also the memory usage
# Tells me the numbers of features/columns (35) with 35 labels too and total data enter
# Moreso, it also reveals the total missing values in each feature/column(s) which tell
# helps to identify the structure of the cancer data.


data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3047 entries, 0 to 3046
Data columns (total 35 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Unnamed: 0              3047 non-null   int64
 1   avgAnnCount             2637 non-null   float64
 2   avgDeathsPerYear        2638 non-null   float64
 3   TARGET_deathRate        2629 non-null   float64
 4   incidenceRate           2640 non-null   float64
 5   medIncome               2633 non-null   float64
 6   popEst2015              2630 non-null   float64
```

```
 7   povertyPercent           2633 non-null   float64
 8   studyPerCap              2644 non-null   float64
 9   binnedInc                2632 non-null   object
 10  MedianAge                2634 non-null   float64
 11  MedianAgeMale            2636 non-null   float64
 12  MedianAgeFemale          2631 non-null   float64
 13  Geography                2641 non-null   object
 14  AvgHouseholdSize         2638 non-null   float64
 15  PercentMarried           2639 non-null   float64
 16  PctNoHS18_24             2640 non-null   float64
 17  PctHS18_24               2634 non-null   float64
 18  PctSomeCol18_24          653 non-null    float64
 19  PctBachDeg18_24          2634 non-null   float64
 20  PctHS25_Over             2631 non-null   float64
 21  PctBachDeg25_Over        2638 non-null   float64
 22  PctEmployed16_Over       2497 non-null   float64
 23  PctUnemployed16_Over     2641 non-null   float64
 24  PctPrivateCoverage       2639 non-null   float64
 25  PctPrivateCoverageAlone  2118 non-null   float64
 26  PctEmpPrivCoverage       2636 non-null   float64
 27  PctPublicCoverage        2639 non-null   float64
 28  PctPublicCoverageAlone   2646 non-null   float64
 29  PctWhite                 2637 non-null   float64
 30  PctBlack                 2632 non-null   float64
 31  PctAsian                 2634 non-null   float64
 32  PctOtherRace             2630 non-null   float64
 33  PctMarriedHouseholds     2637 non-null   float64
 34  BirthRate                2634 non-null   float64
dtypes: float64(32), int64(1), object(2)
memory usage: 833.3+ KB
```

In [11]:

```
#         INTERPRETATION OF RESULT
# This specifically reveals the total samples and features of the cancer data.
# 3047 samples and 35 features respectively


data.shape
```

Out[11]: (3047, 35)

In [12]:

```
#         INTERPRETATION OF RESULT
### THIS GIVE DATA DISTRIBUTION AND SKEWNESS

# This will not return features that are not numeric but only numerics.
# This gives me the summary statistics of the cancer data
# The cancer data is poorly behaved after careful inspection of the statistical feature
# Hence, the cancer data requires a RENORMALISATION of the data before usage, having co
# STANDARD DEVIATION, MINIMUM and MAXIMUM values.


data.describe()
```

Out[12]:

	Unnamed: 0	avgAnnCount	avgDeathsPerYear	TARGET_deathRate	incidenceRate	medIncome
count	3047.000000	2637.000000	2638.000000	2629.000000	2640.000000	2633.000000
mean	1523.000000	616.813493	190.055724	178.748840	447.960586	47024.274592

	Unnamed: 0	avgAnnCount	avgDeathsPerYear	TARGET_deathRate	incidenceRate	medIncome
std	879.737461	1467.543751	527.267114	27.803006	55.406582	11921.226519
min	0.000000	7.000000	3.000000	59.700000	201.300000	22640.000000
25%	761.500000	77.000000	27.000000	161.300000	419.500000	38989.000000
50%	1523.000000	173.000000	60.000000	178.100000	453.549422	45235.000000
75%	2284.500000	526.000000	148.000000	195.200000	480.900000	52513.000000
max	3046.000000	38150.000000	14010.000000	362.800000	1206.900000	125635.000000

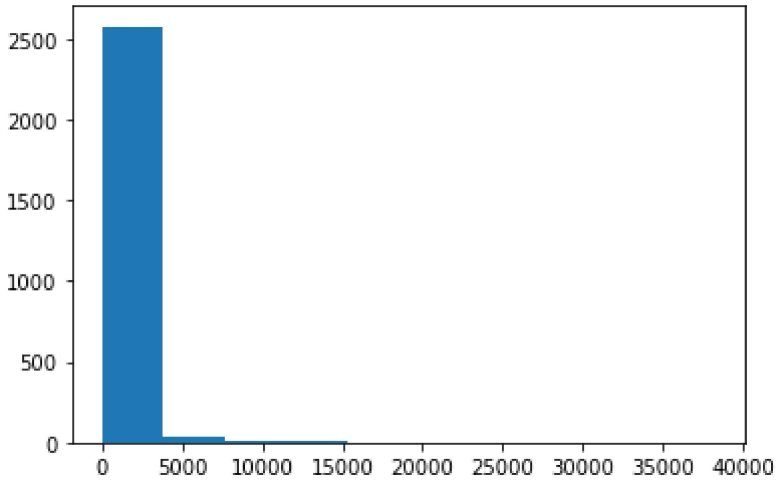
8 rows × 33 columns

In [13]:
```
#         INTERPRETATION OF RESULT
# This reveals the hidden labels in the features of cancer data


data.columns.tolist()
```

Out[13]:
```
['Unnamed: 0',
 'avgAnnCount',
 'avgDeathsPerYear',
 'TARGET_deathRate',
 'incidenceRate',
 'medIncome',
 'popEst2015',
 'povertyPercent',
 'studyPerCap',
 'binnedInc',
 'MedianAge',
 'MedianAgeMale',
 'MedianAgeFemale',
 'Geography',
 'AvgHouseholdSize',
 'PercentMarried',
 'PctNoHS18_24',
 'PctHS18_24',
 'PctSomeCol18_24',
 'PctBachDeg18_24',
 'PctHS25_Over',
 'PctBachDeg25_Over',
 'PctEmployed16_Over',
 'PctUnemployed16_Over',
 'PctPrivateCoverage',
 'PctPrivateCoverageAlone',
 'PctEmpPrivCoverage',
 'PctPublicCoverage',
 'PctPublicCoverageAlone',
 'PctWhite',
 'PctBlack',
 'PctAsian',
 'PctOtherRace',
 'PctMarriedHouseholds',
 'BirthRate']
```

In [15]:
```
import matplotlib.pyplot as plt
plt.hist(data.avgAnnCount.values)


#         INTERPRETATION OF RESULT
# This is not normal distribution, then, it is positively skewed. Exponentially Skewed.
```

Out[15]:
```
(array([2.577e+03, 4.100e+01, 1.200e+01, 4.000e+00, 1.000e+00, 0.000e+00,
        1.000e+00, 0.000e+00, 0.000e+00, 1.000e+00]),
 array([7.00000e+00, 3.82130e+03, 7.63560e+03, 1.14499e+04, 1.52642e+04,
        1.90785e+04, 2.28928e+04, 2.67071e+04, 3.05214e+04, 3.43357e+04,
        3.81500e+04]),
 <BarContainer object of 10 artists>)
```



In [ ]:
```

```

In [ ]:
```
# QUESTION THREE
# 3. Identify and report missing values and duplicate samples in the data
```

In [17]:
```
#         INTERPRETATION OF RESULT
# Identifying missing values

# This reveals the sum total number of  missing values per features/column(s) from 3047
# Hence, the result below shows that the caancer data is UNCLEAN.


data.isna().sum()
```

Out[17]:
```
Unnamed: 0                  0
avgAnnCount               410
avgDeathsPerYear          409
TARGET_deathRate          418
incidenceRate             407
medIncome                 414
popEst2015                417
povertyPercent            414
studyPerCap               403
binnedInc                 415
MedianAge                 413
MedianAgeMale             411
```

```
MedianAgeFemale             416
Geography                   406
AvgHouseholdSize            409
PercentMarried              408
PctNoHS18_24                407
PctHS18_24                  413
PctSomeCol18_24            2394
PctBachDeg18_24             413
PctHS25_Over                416
PctBachDeg25_Over           409
PctEmployed16_Over          550
PctUnemployed16_Over        406
PctPrivateCoverage          408
PctPrivateCoverageAlone     929
PctEmpPrivCoverage          411
PctPublicCoverage           408
PctPublicCoverageAlone      401
PctWhite                    410
PctBlack                    415
PctAsian                    413
PctOtherRace                417
PctMarriedHouseholds        410
BirthRate                   413
dtype: int64
```

In [19]:
```
# This shows the total number of missing values from the entire dataset


data.isna().sum().sum()
```

Out[19]:
```
16613
```

In [20]:
```
#         INTERPRETATION OF RESULT
# Identifying duplicates


data.duplicated()
```

Out[20]:
```
0       False
1       False
2       False
3       False
4       False
        ...
3042    False
3043    False
3044    False
3045    False
3046    False
Length: 3047, dtype: bool
```

In [ ]:
```

```

In [ ]:
```
# QUESTION FOUR
# 4. Clean the data of missing values and also of duplicate values.
# (Explain your choices and give justifications for any method you use to clean the dat
```

In [ ]:
```
#         INTERPRETATION OF RESULT

# To treat the missing values discovered from the cancer data (from avgAnnCount through
# a decision must be taken either to delete, or insert values there

# REASONS BEHIND CHOICE OF METHOD
# Considering the (hugeness of the) STANDARD DEVIATION the missing values are best repl
# PADDING METHOD to fill the remaining missing values in each features.

# dropna or deletion will not be suitable for cancerdata case study because if used the
# the total missing values in each feature is above 10% of the total samples.
```

In [48]:
```
        ### MEDIAN METHOD IS BEING USED TO CLEAN THE DATA

data.fillna(data.median())
```

```
C:\Users\OLUWAS~1\AppData\Local\Temp/ipykernel_13876/3404659454.py:1: FutureWarning: Dro
pping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecat
ed; in a future version this will raise TypeError.  Select only valid columns before cal
ling the reduction.
  data.fillna(data.median())
```

Out[48]:

	Unnamed: 0	avgAnnCount	avgDeathsPerYear	TARGET_deathRate	incidenceRate	medIncome	pop
0	0	1397.000000	469.0	164.9	489.800000	61898.0	2
1	1	173.000000	70.0	161.3	411.600000	48127.0	
2	2	173.000000	50.0	174.7	349.700000	49348.0	
3	3	427.000000	60.0	194.8	430.400000	44243.0	
4	4	57.000000	60.0	144.4	350.100000	49955.0	
...	...	...	...	...	...	...	
3042	3042	1962.667684	15.0	178.1	453.549422	46961.0	
3043	3043	1962.667684	43.0	150.1	453.549422	48609.0	
3044	3044	173.000000	46.0	153.9	453.549422	45235.0	
3045	3045	1962.667684	52.0	175.0	453.549422	50745.0	
3046	3046	1962.667684	48.0	213.6	453.549422	41193.0	

3047 rows × 35 columns

In [ ]:

In [49]:
```
#   CHECKING TO SEE IF THE MISSING VALUES IN SAMPLES AND FEATURES HAS BEEN SORTED USING
#   RESULT BELOW SHOWS THAT THE DATA IS NOT CLEANED YET AND THERE ARE STILL SOME MISSIN
# ACTION TO BE TAKEN WILL BE TO USE THE SECOND METHOD WHICH THE "PADDING METHOD" TO CLE



newData = data.fillna(data.median())

newData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3047 entries, 0 to 3046
Data columns (total 35 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Unnamed: 0               3047 non-null   int64  
 1   avgAnnCount              3047 non-null   float64
 2   avgDeathsPerYear         3047 non-null   float64
 3   TARGET_deathRate         3047 non-null   float64
 4   incidenceRate            3047 non-null   float64
 5   medIncome                3047 non-null   float64
 6   popEst2015               3047 non-null   float64
 7   povertyPercent           3047 non-null   float64
 8   studyPerCap              3047 non-null   float64
 9   binnedInc                2632 non-null   object 
 10  MedianAge                3047 non-null   float64
 11  MedianAgeMale            3047 non-null   float64
 12  MedianAgeFemale          3047 non-null   float64
 13  Geography                2641 non-null   object 
 14  AvgHouseholdSize         3047 non-null   float64
 15  PercentMarried           3047 non-null   float64
 16  PctNoHS18_24             3047 non-null   float64
 17  PctHS18_24               3047 non-null   float64
 18  PctSomeCol18_24          3047 non-null   float64
 19  PctBachDeg18_24          3047 non-null   float64
 20  PctHS25_Over             3047 non-null   float64
 21  PctBachDeg25_Over        3047 non-null   float64
 22  PctEmployed16_Over       3047 non-null   float64
 23  PctUnemployed16_Over     3047 non-null   float64
 24  PctPrivateCoverage       3047 non-null   float64
 25  PctPrivateCoverageAlone  3047 non-null   float64
 26  PctEmpPrivCoverage       3047 non-null   float64
 27  PctPublicCoverage        3047 non-null   float64
 28  PctPublicCoverageAlone   3047 non-null   float64
 29  PctWhite                 3047 non-null   float64
 30  PctBlack                 3047 non-null   float64
 31  PctAsian                 3047 non-null   float64
 32  PctOtherRace             3047 non-null   float64
 33  PctMarriedHouseholds     3047 non-null   float64
 34  BirthRate                3047 non-null   float64
dtypes: float64(32), int64(1), object(2)
memory usage: 833.3+ KB
```

```
C:\Users\OLUWAS~1\AppData\Local\Temp/ipykernel_13876/2349457108.py:1: FutureWarning: Dro
pping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecat
ed; in a future version this will raise TypeError.  Select only valid columns before cal
```

```
ling the reduction.
  newData = data.fillna(data.median())
```

In [42]:

```
# This reveals that the newData still has some missing values at features labelled (bin
# The newData is still unclean, hence, the padding method will then be used to complete
# SEE BELOW: (NOT TOTALLY CLEAN YET)


newData.isna().sum()
```

Out[42]:

```
Unnamed: 0                  0
avgAnnCount                 0
avgDeathsPerYear            0
TARGET_deathRate            0
incidenceRate               0
medIncome                   0
popEst2015                  0
povertyPercent              0
studyPerCap                 0
binnedInc                 415
MedianAge                   0
MedianAgeMale               0
MedianAgeFemale             0
Geography                 406
AvgHouseholdSize            0
PercentMarried              0
PctNoHS18_24                0
PctHS18_24                  0
PctSomeCol18_24             0
PctBachDeg18_24             0
PctHS25_Over                0
PctBachDeg25_Over           0
PctEmployed16_Over          0
PctUnemployed16_Over        0
PctPrivateCoverage          0
PctPrivateCoverageAlone     0
PctEmpPrivCoverage          0
PctPublicCoverage           0
PctPublicCoverageAlone      0
PctWhite                    0
PctBlack                    0
PctAsian                    0
PctOtherRace                0
PctMarriedHouseholds        0
BirthRate                   0
dtype: int64
```

In [43]:

```
###      PADDING METHOD (SECOND METHOD)
# Hence, the second method which is "Padding method" will be used to complete cleaning
# SEE BELOW


newData.fillna(method = "pad")
```

Out[43]:

	Unnamed: 0	avgAnnCount	avgDeathsPerYear	TARGET_deathRate	incidenceRate	medIncome	pop

	Unnamed: 0	avgAnnCount	avgDeathsPerYear	TARGET_deathRate	incidenceRate	medIncome	pop
0	0	1397.000000	469.0	164.9	489.800000	61898.0	2
1	1	173.000000	70.0	161.3	411.600000	48127.0	
2	2	173.000000	50.0	174.7	349.700000	49348.0	
3	3	427.000000	60.0	194.8	430.400000	44243.0	
4	4	57.000000	60.0	144.4	350.100000	49955.0	
...	...	...	...	...	...	...	...
3042	3042	1962.667684	15.0	178.1	453.549422	46961.0	
3043	3043	1962.667684	43.0	150.1	453.549422	48609.0	
3044	3044	173.000000	46.0	153.9	453.549422	45235.0	
3045	3045	1962.667684	52.0	175.0	453.549422	50745.0	
3046	3046	1962.667684	48.0	213.6	453.549422	41193.0	

3047 rows × 35 columns

In [ ]:
```
###
    CHECKING THE RESULTS TO CONFIRM IF THE DATA HAS BEEN CLEANED OF MISSING VALUES
###
```

In [59]:
```
# cleanData = newData.fillna(method = "pad")

# cleanData.info()shows that all the missing values in each feature(s) has been filled,
# Hence, the data is cleaned from missing values.


cleanData = newData.fillna(method = "pad")

cleanData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3047 entries, 0 to 3046
Data columns (total 35 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Unnamed: 0               3047 non-null   int64
 1   avgAnnCount              3047 non-null   float64
```

```
 2   avgDeathsPerYear         3047 non-null   float64
 3   TARGET_deathRate         3047 non-null   float64
 4   incidenceRate            3047 non-null   float64
 5   medIncome                3047 non-null   float64
 6   popEst2015               3047 non-null   float64
 7   povertyPercent           3047 non-null   float64
 8   studyPerCap              3047 non-null   float64
 9   binnedInc                3047 non-null   object
 10  MedianAge                3047 non-null   float64
 11  MedianAgeMale            3047 non-null   float64
 12  MedianAgeFemale          3047 non-null   float64
 13  Geography                3047 non-null   object
 14  AvgHouseholdSize         3047 non-null   float64
 15  PercentMarried           3047 non-null   float64
 16  PctNoHS18_24             3047 non-null   float64
 17  PctHS18_24               3047 non-null   float64
 18  PctSomeCol18_24          3047 non-null   float64
 19  PctBachDeg18_24          3047 non-null   float64
 20  PctHS25_Over             3047 non-null   float64
 21  PctBachDeg25_Over        3047 non-null   float64
 22  PctEmployed16_Over       3047 non-null   float64
 23  PctUnemployed16_Over     3047 non-null   float64
 24  PctPrivateCoverage       3047 non-null   float64
 25  PctPrivateCoverageAlone  3047 non-null   float64
 26  PctEmpPrivCoverage       3047 non-null   float64
 27  PctPublicCoverage        3047 non-null   float64
 28  PctPublicCoverageAlone   3047 non-null   float64
 29  PctWhite                 3047 non-null   float64
 30  PctBlack                 3047 non-null   float64
 31  PctAsian                 3047 non-null   float64
 32  PctOtherRace             3047 non-null   float64
 33  PctMarriedHouseholds     3047 non-null   float64
 34  BirthRate                3047 non-null   float64
dtypes: float64(32), int64(1), object(2)
memory usage: 833.3+ KB
```

In [60]:

```
# CHECKING TO SEE IF THE DATA IS TOTALLY CLEANED

cleanData2.isna().sum()
```

Out[60]:

```
Unnamed: 0               0
avgAnnCount              0
avgDeathsPerYear         0
TARGET_deathRate         0
incidenceRate            0
medIncome                0
popEst2015               0
povertyPercent           0
studyPerCap              0
binnedInc                0
MedianAge                0
MedianAgeMale            0
MedianAgeFemale          0
Geography                0
AvgHouseholdSize         0
PercentMarried           0
PctNoHS18_24             0
PctHS18_24               0
PctSomeCol18_24          0
```

```
PctBachDeg18_24           0
PctHS25_Over              0
PctBachDeg25_Over         0
PctEmployed16_Over        0
PctUnemployed16_Over      0
PctPrivateCoverage        0
PctPrivateCoverageAlone   0
PctEmpPrivCoverage        0
PctPublicCoverage         0
PctPublicCoverageAlone    0
PctWhite                  0
PctBlack                  0
PctAsian                  0
PctOtherRace              0
PctMarriedHouseholds      0
BirthRate                 0
dtype: int64
```

In [73]:
```
# This is to check the total number of missing values from the entire cleaned dataset


cleanData.isna().sum().sum()
```

Out[73]:
```
0
```

In [ ]:
```
###
    CHECKING THE RESULTS TO CONFIRM IF THE DATA HAS BEEN CLEANED OF DUPLICATE VALUES
###
```

In [61]:
```
cleanData.duplicated()
```

Out[61]:
```
0       False
1       False
2       False
3       False
4       False
        ...
3042    False
3043    False
3044    False
3045    False
3046    False
Length: 3047, dtype: bool
```

In [63]:
```
# This is to check if the cancer data is void of Duplicate values.


data.duplicated().sum()
```

Out[63]:
```
0
```

In [62]:
```
# This is to check and confirm the total samples and features after cleaning
# 3047 samples and 35 features respectively
```

```
cleanData.shape
```

Out[62]:
```
(3047, 35)
```

In [ ]:
```

```

In [77]:
```
# QUESTION FIVE
# 5. Report the structure of the final data after cleaning (number of samples and numbe

            # REPORT
# cleanData.info()shows that all the missing values in each feature(s) has been filled,
# Hence, the data is cleaned from missing values.



cleanData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3047 entries, 0 to 3046
Data columns (total 35 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Unnamed: 0               3047 non-null   int64
 1   avgAnnCount              3047 non-null   float64
 2   avgDeathsPerYear         3047 non-null   float64
 3   TARGET_deathRate         3047 non-null   float64
 4   incidenceRate            3047 non-null   float64
 5   medIncome                3047 non-null   float64
 6   popEst2015               3047 non-null   float64
 7   povertyPercent           3047 non-null   float64
 8   studyPerCap              3047 non-null   float64
 9   binnedInc                3047 non-null   object
 10  MedianAge                3047 non-null   float64
 11  MedianAgeMale            3047 non-null   float64
 12  MedianAgeFemale          3047 non-null   float64
 13  Geography                3047 non-null   object
 14  AvgHouseholdSize         3047 non-null   float64
 15  PercentMarried           3047 non-null   float64
 16  PctNoHS18_24             3047 non-null   float64
 17  PctHS18_24               3047 non-null   float64
 18  PctSomeCol18_24          3047 non-null   float64
 19  PctBachDeg18_24          3047 non-null   float64
 20  PctHS25_Over             3047 non-null   float64
 21  PctBachDeg25_Over        3047 non-null   float64
 22  PctEmployed16_Over       3047 non-null   float64
 23  PctUnemployed16_Over     3047 non-null   float64
 24  PctPrivateCoverage       3047 non-null   float64
 25  PctPrivateCoverageAlone  3047 non-null   float64
 26  PctEmpPrivCoverage       3047 non-null   float64
 27  PctPublicCoverage        3047 non-null   float64
 28  PctPublicCoverageAlone   3047 non-null   float64
 29  PctWhite                 3047 non-null   float64
 30  PctBlack                 3047 non-null   float64
 31  PctAsian                 3047 non-null   float64
 32  PctOtherRace             3047 non-null   float64
 33  PctMarriedHouseholds     3047 non-null   float64
```

```
 34  BirthRate                3047 non-null   float64
dtypes: float64(32), int64(1), object(2)
memory usage: 833.3+ KB
```

In [76]:
```
# This reveals that the cancer data has been cleaned of all missing values in each feat


cleanData.isna().sum()
```

Out[76]:
```
Unnamed: 0                0
avgAnnCount               0
avgDeathsPerYear          0
TARGET_deathRate          0
incidenceRate             0
medIncome                 0
popEst2015                0
povertyPercent            0
studyPerCap               0
binnedInc                 0
MedianAge                 0
MedianAgeMale             0
MedianAgeFemale           0
Geography                 0
AvgHouseholdSize          0
PercentMarried            0
PctNoHS18_24              0
PctHS18_24                0
PctSomeCol18_24           0
PctBachDeg18_24           0
PctHS25_Over              0
PctBachDeg25_Over         0
PctEmployed16_Over        0
PctUnemployed16_Over      0
PctPrivateCoverage        0
PctPrivateCoverageAlone   0
PctEmpPrivCoverage        0
PctPublicCoverage         0
PctPublicCoverageAlone    0
PctWhite                  0
PctBlack                  0
PctAsian                  0
PctOtherRace              0
PctMarriedHouseholds      0
BirthRate                 0
dtype: int64
```

In [74]:
```
# This confirms that there are no missing values in the entire cleaned dataset


cleanData.isna().sum().sum()
```

Out[74]:
```
0
```

In [53]:
```
# This confirms the total samples and features after cleaning

cleanData.shape
```

Out[53]: (3047, 35)

In [54]:
```
# This reveals that the cancer data has been cleaned of all Duplicate values.


cleanData.duplicated()
```

Out[54]:
```
0       False
1       False
2       False
3       False
4       False
        ...
3042    False
3043    False
3044    False
3045    False
3046    False
Length: 3047, dtype: bool
```

In [55]:
```
# This confirms that the cancer data is void of Duplicate values.


data.duplicated().sum()
```

Out[55]: 0

In [75]:
```
cleanData.describe()
```

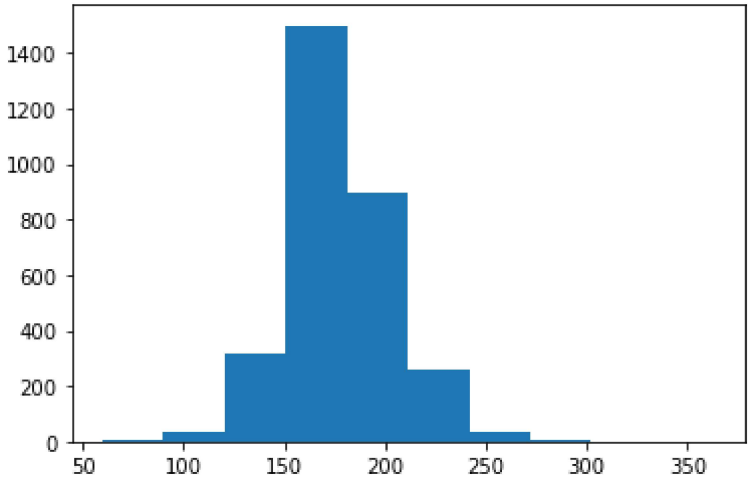
Out[75]:

	Unnamed: 0	avgAnnCount	avgDeathsPerYear	TARGET_deathRate	incidenceRate	medIncome
count	3047.000000	3047.000000	3047.000000	3047.000000	3047.000000	3047.000000
mean	1523.000000	557.094578	172.598293	178.659829	448.707109	46781.163439
std	879.737461	1373.585821	492.592334	25.825915	51.607300	11098.462572
min	0.000000	7.000000	3.000000	59.700000	201.300000	22640.000000
25%	761.500000	89.000000	31.000000	163.800000	425.400000	40123.000000
50%	1523.000000	173.000000	60.000000	178.100000	453.549422	45235.000000
75%	2284.500000	415.500000	122.500000	192.050000	475.300000	51317.500000
max	3046.000000	38150.000000	14010.000000	362.800000	1206.900000	125635.000000

8 rows × 33 columns

In [64]:
```
# THE HISTOGRAM DISTRIBUTION TO CONFIRM THE SKEWNESS


plt.hist(cleanData.TARGET_deathRate     .values)
```

Out[64]:
```
(array([3.000e+00, 3.600e+01, 3.140e+02, 1.498e+03, 8.940e+02, 2.560e+02,
        3.800e+01, 7.000e+00, 0.000e+00, 1.000e+00]),
 array([ 59.7 ,  90.01, 120.32, 150.63, 180.94, 211.25, 241.56, 271.87,
        302.18, 332.49, 362.8 ]),
 <BarContainer object of 10 artists>)
```



In [ ]:
```
# REASONS BEHIND CHOICE OF METHODS

      ### NOTE:
# dropna or deletion would not have been a suitable method for cancerdata case study be
# one will lose data representation and the total missing values in each feature is abo

# Having considered the hugeness of the STANDARD DEVIATION, the missing values are best
# PADDING METHOD to fill the remaining missing values in each features.

# Hence,the cancerdata has cleaned of all missing values and duplicates, it is now  wel
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: