In [2]:

```
#importing all necessary libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:

```
#question 1
A = [1,2,3,4,5,6]
B = [13, 21, 34]
print(A)
A_B = A.extend(B)
print(A)
print(A_B)
```

```
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6, 13, 21, 34]
None
```

In [98]:

```
np.identity(3)
```

Out[98]:

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

In [4]:

```
url="https://bit.ly/HDSC-StageOneDataset"
#read the dataset
fuel_data=pd.read_csv(url, error_bad_lines=False)

#checking for missing values


fuel_data.describe(include="all")

fuel_data["fuel_unit"].describe(include="all")
#Question 7
#From the output, mode and categorical imputation method will be the best since the missin
g data
#is not a number but a unit. so i replaced with the most common value, ie mode, which is m
cf
```

Out[4]:

```
count     29343
unique        9
top         mcf
freq      11354
Name: fuel_unit, dtype: object
```

In [5]:

```
fuel_data.isnull().sum()
#Question 6

missingvaluesInFuelunit = fuel_data["fuel_unit"].isnull().sum()
valuesInFuelunit= fuel_data["fuel_unit"].count()
percentage = (missingvaluesInFuelunit/valuesInFuelunit)*100
print("No of values in fuelunit is %d ; the number of missing values is %d and the percent
age of missing values is %.3f "%(valuesInFuelunit,missingvaluesInFuelunit,percentage))
#Feature: fuel_unit, Total: 180, Percent: 0.613)
```

```
No of values in fuelunit is 29343 ; the number of missing values is 180 and t
he percentage of missing values is 0.613
```

In [6]:

```
#filling empty data in fuel units with mcf as earlier stated
fuel_data[["fuel_unit"]]=fuel_data[["fuel_unit"]].fillna("mcf")
#Question 3
fuel_data.groupby("fuel_type_code_pudl")["fuel_cost_per_unit_delivered"].describe()
```

Out[6]:

fuel_type_code_pudl	count	mean	std	min	25%	50%	75%
coal	8547.0	116.951141	4043.732781	-40.725	21.7400	31.407	43.38050
gas	11486.0	12.095172	157.032849	0.000	2.8300	4.295	6.49250
nuclear	818.0	28616.915039	411287.325331	0.000	0.0000	0.000	28.91900
oil	8064.0	313.907691	9015.713229	-874.937	32.1545	58.240	95.74075
other	167.0	16.871485	27.659197	-118.340	2.4830	8.500	19.85300
waste	441.0	18.413052	27.966921	-174.670	7.2400	21.560	31.33000

In [7]:

```
#the analysis above shows that gas has the least average cost per unit delivered

#Question 4
fuel_data["fuel_mmbtu_per_unit"].describe()
#Answer: The standard deviation is 10.60 and 75th percentile is 17.01
```

Out[7]:

```
count    29523.000000
mean         8.492111
std         10.600220
min          0.000001
25%          1.024000
50%          5.762694
75%         17.006000
max        341.260000
Name: fuel_mmbtu_per_unit, dtype: float64
```

In [7]:

```
#Question 5
import scipy.stats as stat
skew = stat.skew(fuel_data["fuel_qty_burned"])
kurtosis = stat.kurtosis(fuel_data["fuel_qty_burned"])
print("The skewness is %.2f and the kurtosis is %.2f"%(skew, kurtosis))
# The skewness is 15.85 and the kurtosis is 651.26
```

```
The skewness is 15.85 and the kurtosis is 651.26
```

In [8]:

```
#Question 8
t=fuel_data["fuel_cost_per_unit_burned"].corr(fuel_data["fuel_qty_burned"])
r=fuel_data["fuel_cost_per_unit_burned"].corr(fuel_data["fuel_cost_per_unit_delivered"])
e=fuel_data["fuel_cost_per_unit_burned"].corr(fuel_data["report_year"])
w=fuel_data["fuel_cost_per_unit_burned"].corr(fuel_data["fuel_mmbtu_per_unit"])
q=fuel_data["fuel_cost_per_unit_burned"].corr(fuel_data["utility_id_ferc1"])
corrArray=[t,r,w,e,q]
print(corrArray)
corrArray.sort(reverse=True) #to sort in descending Order

print(corrArray)


# the Output shows that t is second to last and e is third to the last
# the Output shows that correlation of fuel cost per unit burned with fuel qty burned and
 report year are
# second to the last and third to the last respectively
```

```
[-0.018535416794825232, 0.011007403299165127, -0.010033808428853076, 0.013599
13443813416, -0.037862707599972574]
[0.01359913443813416, 0.011007403299165127, -0.010033808428853076, -0.0185354
16794825232, -0.037862707599972574]
```

In [30]:

```
#Question 9
is_1998=fuel_data["report_year"]==1998
is_1994=fuel_data["report_year"]==1994
year_1998=pd.DataFrame(fuel_data[is_1998])

year_1994=pd.DataFrame(fuel_data[is_1994])

sum98=year_1998.groupby("fuel_type_code_pudl")["fuel_cost_per_unit_burned"].sum()
sum94=year_1994.groupby("fuel_type_code_pudl")["fuel_cost_per_unit_burned"].sum()
print("summary for year 1998 is")
print(sum98)
print("summary for year 1994 is")
print(sum94)
print("fuel cost per unit of coal burned  in 1998 is %.5f for 1994 it is %.5f"%(sum98["coa
l"],sum94["coal"]))
change_percent= (sum98["coal"]-sum94["coal"])*100/sum98["coal"]
print(" the percentage change from 1998 to 1994 is %f"%change_percent)
# THe output gives a negative value, so we can say the usage of coal actually reduced in t
he year 1998 from 1994 and  by 25.89%
```

```
summary for year 1998 is
fuel_type_code_pudl
coal        11902.597
gas          7475.596
nuclear    117788.269
oil          2608.657
other         225.428
waste         238.109
Name: fuel_cost_per_unit_burned, dtype: float64
summary for year 1994 is
fuel_type_code_pudl
coal        14984.572
gas         10792.496
nuclear    227983.354
oil          9362.194
other         116.737
waste          52.762
Name: fuel_cost_per_unit_burned, dtype: float64
fuel cost per unit of coal burned  in 1998 is 11902.59700 for 1994 it is 1498
4.57200
 the percentage change from 1998 to 1994 is -25.893299
```

In [44]:

```
#Question 10

years=pd.DataFrame(fuel_data.groupby("report_year")
["fuel_cost_per_unit_delivered"].mean())

print(years)

# the results shows that 1997 has the highest mean for Cost Per Unit delivered
```

```
             fuel_cost_per_unit_delivered
report_year
1994                            63.636060
1995                            32.735269
1996                          9196.705948
1997                         11140.197239
1998                           287.154420
1999                            25.551627
2000                           985.362877
2001                            60.050396
2002                            47.594361
2003                            55.663493
2004                           139.524275
2005                            41.438184
2006                            38.657484
2007                            43.325023
2008                            58.588197
2009                           652.694163
2010                            91.862105
2011                            59.774667
2012                            60.994502
2013                           172.307591
2014                           192.737183
2015                           326.535511
2016                           103.901761
2017                            46.196861
2018                           499.269966
```

In [ ]: