```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
```

```
!wget -q https://archive.apache.org/dist/spark/spark-3.0.0/spark-3.0.0-bin-hadoop3.2.tgz
```

```
!tar xf spark-3.0.0-bin-hadoop3.2.tgz
```

```
# set your spark folder to your system path environment.
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.0.0-bin-hadoop3.2"
```

```
# install findspark using pip
!pip install -q findspark
```

```
!pip install pyspark
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.7/dist-packages (3.2.0)
Requirement already satisfied: py4j==0.10.9.2 in /usr/local/lib/python3.7/dist-packages
```

```
import findspark
findspark.init()
```

```
findspark.find()
```

```
'/content/spark-3.0.0-bin-hadoop3.2'
```

```
from pyspark.sql import SparkSession

spark = SparkSession.builder\
        .master("local")\
        .appName("Colab")\
        .config('spark.ui.port', '4050')\
        .getOrCreate()
```

```
spark
```

SparkSession - in-memory

SparkContext

Spark UI

Version
    v3.0.0
Master
    local
AppName
    Colab

```
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
import sqlite3
import seaborn as sns
import sklearn
```

New Section

```
import pyspark
from pyspark import SparkContext
from pyspark.sql import SQLContext
```

```
import pandas as pd
from pyspark.sql import SparkSession
from pyspark.context import SparkContext
import time
```

```
from pyspark.sql import SparkSession
from pyspark.sql.types import *
from pyspark.sql.functions import *
from pyspark.sql.types import Row
from datetime import datetime
```

```
from pyspark.sql import functions as F
```

```
from pyspark.ml.feature import VectorAssembler
```

```
from pyspark.ml.regression import DecisionTreeRegressor
```

```
from pyspark.ml.classification import DecisionTreeClassifier
```

```
from sklearn.linear_model import LinearRegression
```

Double-click (or enter) to edit

```
  from pyspark.ml import Pipeline
  from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler
  from pyspark.sql.functions import col
```

```
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.tree import RandomForest
from pyspark.mllib.linalg import SparseVector
from pyspark.ml.classification import RandomForestClassifier
```

```
Airpollution=pd.read_csv("drive/MyDrive/my_data/Airpollution2.csv")
Lifeexpectancy=pd.read_csv("drive/MyDrive/my_data/Life expectancy_Li2.csv")
Human_rights_score=pd.read_csv("drive/MyDrive/my_data/Human rights score2.csv")
```

```
Airpollution
```

	Entity	Code	Year	water	indoor	Alcohol	BMI	Smoking	Outdoor	
0	Afghanistan	AFG	1990	7554.05	22388.50	356.53	7701.58	6393.67	4383.83	113
1	Afghanistan	AFG	1991	7359.68	22128.76	320.60	7747.77	6429.25	4426.36	114
2	Afghanistan	AFG	1992	7650.44	22873.77	293.26	7991.02	6561.05	4568.91	118
3	Afghanistan	AFG	1993	10270.73	25599.76	278.13	8281.56	6731.97	5080.29	123
4	Afghanistan	AFG	1994	11409.18	28013.17	250.69	8472.30	6889.33	5499.23	126
...	...	...	...	...	...	...	...	...	...	
6463	Zimbabwe	ZWE	2013	4254.28	7613.56	7377.88	5229.30	9099.55	2053.58	26
6464	Zimbabwe	ZWE	2014	4098.77	7429.45	7202.13	5291.30	8902.22	2030.92	26
6465	Zimbabwe	ZWE	2015	3921.29	7267.03	7174.62	5371.63	8818.57	1994.91	26
6466	Zimbabwe	ZWE	2016	3802.26	7134.60	7174.60	5524.17	8758.49	2030.88	26
6467	Zimbabwe	ZWE	2017	3796.07	6982.34	7227.99	5713.00	8714.71	2112.19	26

6468 rows × 10 columns

```
pandaDF=pd.merge(Lifeexpectancy, Airpollution, on=["Entity", "Code", "Year"])
```

```
pandaDF
```

	Entity	Code	Year	LE	GDP	Pop	water	indoor	Alcohol	
0	Afghanistan	AFG	1990	50.331	963.00	12412000.0	7554.05	22388.50	356.53	77
1	Afghanistan	AFG	1991	50.999	881.17	13299000.0	7359.68	22128.76	320.60	77
2	Afghanistan	AFG	1992	51.641	843.88	14486000.0	7650.44	22873.77	293.26	79
3	Afghanistan	AFG	1993	52.256	578.40	15817000.0	10270.73	25599.76	278.13	82
4	Afghanistan	AFG	1994	52.842	428.42	17076000.0	11409.18	28013.17	250.69	84
...	...	...	...	...	...	...	...	...	...	
5554	Zimbabwe	ZWE	2013	56.897	1604.00	13350000.0	4254.28	7613.56	7377.88	52
5555	Zimbabwe	ZWE	2014	58.410	1594.00	13587000.0	4098.77	7429.45	7202.13	52
5556	Zimbabwe	ZWE	2015	59.534	1560.00	13815000.0	3921.29	7267.03	7174.62	53
5557	Zimbabwe	ZWE	2016	60.294	1534.00	14030000.0	3802.26	7134.60	7174.60	55
5558	Zimbabwe	ZWE	2017	60.812	1582.37	14237000.0	3796.07	6982.34	7227.99	57

5559 rows × 13 columns

```
pandaDF1=pd.merge(pandaDF, Human_rights_score, on=["Entity", "Code", "Year"])
```

```
pandaDF1
```

	Entity	Code	Year	LE	GDP	Pop	water	indoor	Alcohol

```
from pyspark.sql.types import DoubleType
from pyspark.sql.functions import col
```

2	Afghanistan	AFG	1992	51.641	643.66	14486000.0	7650.44	22873.77	293.26	79

```
pandaDF1.describe()
```

	Year	LE	GDP	Pop	water	indoor
count	5159.000000	5159.000000	4522.000000	5.159000e+03	5159.000000	5159.000000
mean	2003.574142	67.901436	13560.018618	3.472152e+07	8838.602016	11605.92015
std	8.061120	9.741005	15629.614473	1.307415e+08	53305.993049	58971.62323
min	1990.000000	26.172000	0.000000	4.800000e+04	0.010000	0.020000
25%	1997.000000	61.422500	2546.195000	2.041500e+06	8.945000	83.305000
50%	2004.000000	70.333000	7903.330000	7.582000e+06	180.030000	770.240000
75%	2011.000000	75.100000	18963.750000	2.291550e+07	3385.680000	5891.875000
max	2017.000000	84.290000	156299.000000	1.421022e+09	811265.910000	790215.840000

```
pandaDF1.isna().sum()
```

```
Entity        0
Code          0
Year          0
LE            0
GDP         637
Pop           0
water         0
indoor        0
Alcohol       0
BMI           0
Smoking       0
Outdoor       0
Diet          0
HRS           0
dtype: int64
```

```
pandaDF1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5159 entries, 0 to 5158
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Entity    5159 non-null   object
```

```
 1   Code      5159 non-null   object
 2   Year      5159 non-null   int64
 3   LE        5159 non-null   float64
 4   GDP       4522 non-null   float64
 5   Pop       5159 non-null   float64
 6   water     5159 non-null   float64
 7   indoor    5159 non-null   float64
 8   Alcohol   5159 non-null   float64
 9   BMI       5159 non-null   float64
 10  Smoking   5159 non-null   float64
 11  Outdoor   5159 non-null   float64
 12  Diet      5159 non-null   float64
 13  HRS       5159 non-null   float64
dtypes: float64(11), int64(1), object(2)
memory usage: 604.6+ KB
```

```
pandaDF1.dtypes
```

```
Entity       object
Code         object
Year          int64
LE          float64
GDP         float64
Pop         float64
water       float64
indoor      float64
Alcohol     float64
BMI         float64
Smoking     float64
Outdoor     float64
Diet        float64
HRS         float64
dtype: object
```

```
pandaDF1.aggregate
```

```
<bound method DataFrame.aggregate of           Entity Code  Year      LE  ...  Smoking
0     Afghanistan  AFG  1990  50.331  ...   6393.67   4383.83  11381.38 -2.74
1     Afghanistan  AFG  1991  50.999  ...   6429.25   4426.36  11487.83 -2.66
2     Afghanistan  AFG  1992  51.641  ...   6561.05   4568.91  11866.24 -2.59
3     Afghanistan  AFG  1993  52.256  ...   6731.97   5080.29  12335.96 -2.51
4     Afghanistan  AFG  1994  52.842  ...   6889.33   5499.23  12672.95 -2.48
...           ...  ...   ...     ...  ...       ...       ...       ...   ...
5154     Zimbabwe  ZWE  2013  56.897  ...   9099.55   2053.58   2687.64 -0.83
5155     Zimbabwe  ZWE  2014  58.410  ...   8902.22   2030.92   2654.38 -0.78
5156     Zimbabwe  ZWE  2015  59.534  ...   8818.57   1994.91   2635.95 -0.85
5157     Zimbabwe  ZWE  2016  60.294  ...   8758.49   2030.88   2641.38 -0.88
5158     Zimbabwe  ZWE  2017  60.812  ...   8714.71   2112.19   2664.13 -0.74

[5159 rows x 14 columns]>
```

```
pandaDF2=pandaDF1[pandaDF1["Year"] == 2000]
```

```
pandaDF2
```

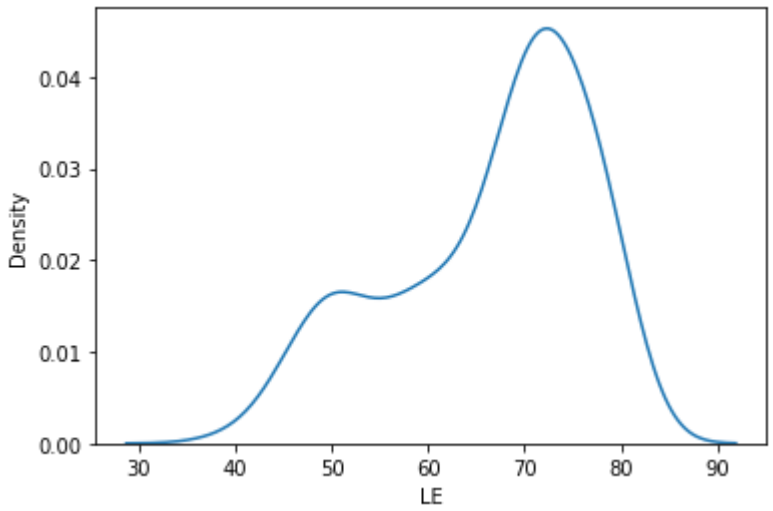
	Entity	Code	Year	LE	GDP	Pop	water	indoor	Alcohol
10	Afghanistan	AFG	2000	55.841	502.37	20780000.0	11121.65	28329.38	167.51
38	Albania	ALB	2000	73.955	4808.48	3129000.0	11.05	994.48	864.91
66	Algeria	DZA	2000	70.640	6834.55	31042000.0	788.45	336.49	673.03
91	Andorra	AND	2000	78.857	NaN	65000.0	0.01	0.35	46.45
119	Angola	AGO	2000	46.522	2013.64	16395000.0	22087.22	14498.16	3560.78
...	...	...	...	...	...	...	...	...	...
5029	Venezuela	VEN	2000	72.112	13992.61	24192000.0	1293.12	212.07	10022.18
5057	Vietnam	VNM	2000	73.025	2773.10	79910000.0	2858.19	29539.65	32313.21
5085	Yemen	YEM	2000	60.683	4212.11	17409000.0	14981.53	9066.98	118.54
5113	Zambia	ZMB	2000	44.000	1428.50	10416000.0	11561.74	7217.43	5115.96
5141	Zimbabwe	ZWE	2000	44.649	2211.20	11881000.0	3602.97	6028.13	6006.02

185 rows × 14 columns

Data Visulization

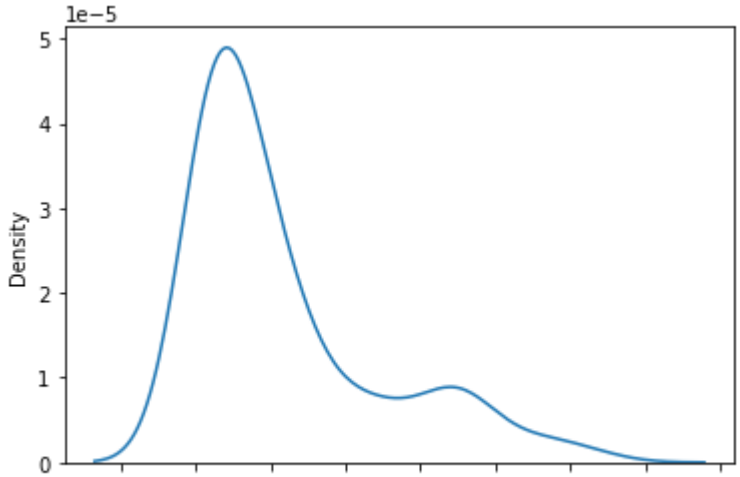
```
sns.kdeplot(x='LE', data=pandaDF2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5322deea10>
```



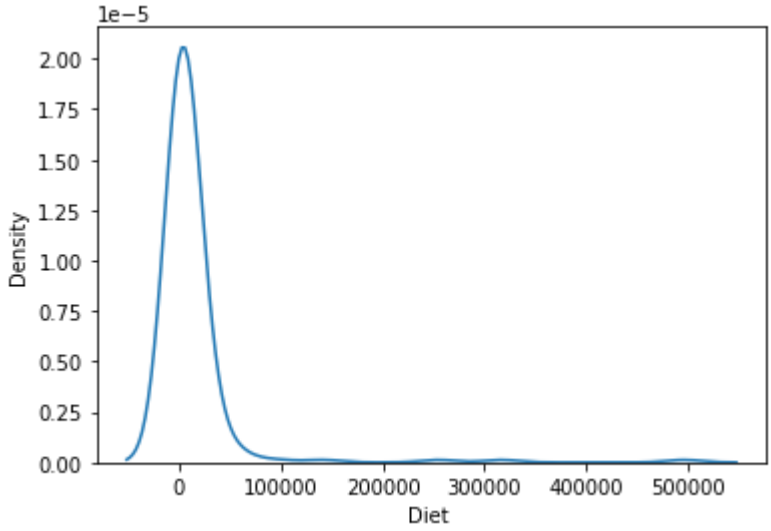

```
sns.kdeplot(x='GDP', data=pandaDF2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5322de0590>
```



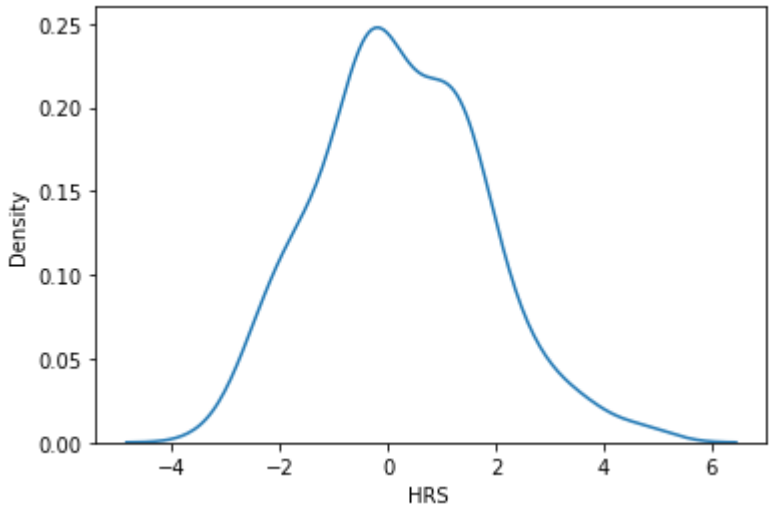

```
sns.kdeplot(x='Diet ', data=pandaDF2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5322d3e550>
```



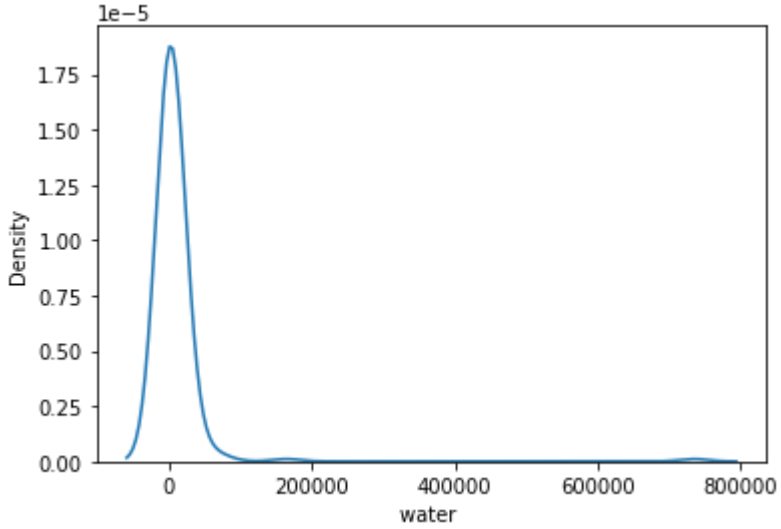

```
sns.kdeplot(x='HRS ', data=pandaDF2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5322cf75d0>
```



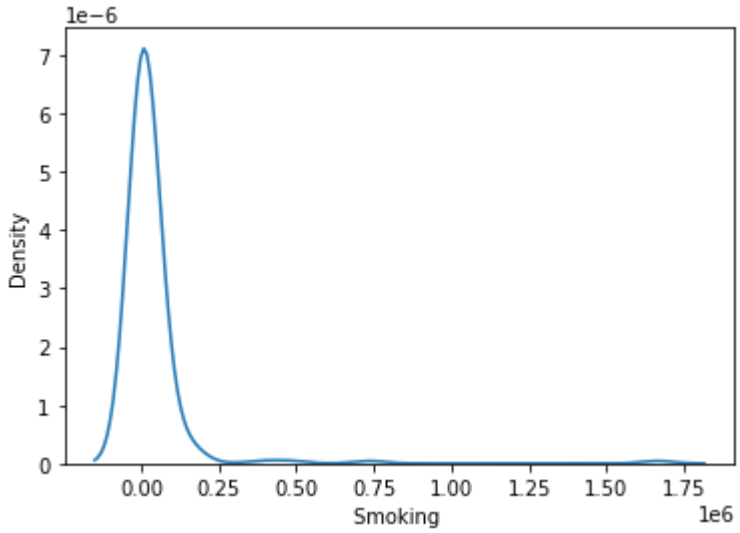
```
sns.kdeplot(x='water ', data=pandaDF2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5322c31a90>
```



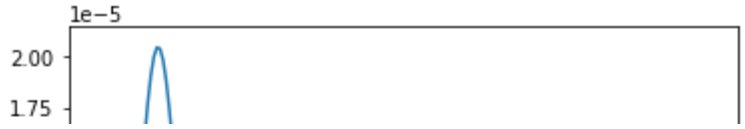

```
sns.kdeplot(x='Smoking ', data=pandaDF2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5322c12bd0>
```




```
sns.kdeplot(x='Alcohol ', data=pandaDF2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5322b84150>
```



Double-click (or enter) to edit

```
pandaDF2['logAlcohol']=np.log(pandaDF2['Alcohol '])
pandaDF2['logDiet']=np.log(pandaDF2['Diet '])
pandaDF2['logSmoking']=np.log(pandaDF2['Smoking '])
pandaDF2['logWater']=np.log(pandaDF2['water '])
pandaDF2['logBMI']=np.log(pandaDF2['BMI '])
pandaDF2['logindoor']=np.log(pandaDF2['indoor'])
pandaDF2['logoutdoor']=np.log(pandaDF2['Outdoor '])
pandaDF2['logpop']=np.log(pandaDF2['Pop '])
pandaDF2['logGDP']=np.log(pandaDF2['GDP'])
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/series.py:726: RuntimeWarning: in
  result = getattr(ufunc, method)(*inputs, **kwargs)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarni
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarni
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarni
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
  This is separate from the ipykernel package so we can avoid doing imports until
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: SettingWithCopyWarni
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
  after removing the cwd from sys.path.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarni
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
  """
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: SettingWithCopyWarni
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: SettingWithCopyWarni
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
  import sys
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: SettingWithCopyWarni
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:9: SettingWithCopyWarni
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
  if __name__ == '__main__':
```

```
pandaDF2
```

	Entity	Code	Year	LE	GDP	Pop	water	indoor	Alcohol
10	Afghanistan	AFG	2000	55.841	502.37	20780000.0	11121.65	28329.38	167.51
38	Albania	ALB	2000	73.955	4808.48	3129000.0	11.05	994.48	864.91
66	Algeria	DZA	2000	70.640	6834.55	31042000.0	788.45	336.49	673.03
91	Andorra	AND	2000	78.857	NaN	65000.0	0.01	0.35	46.45
119	Angola	AGO	2000	46.522	2013.64	16395000.0	22087.22	14498.16	3560.78
...	...	...	...	...	...	...	...	...	...
5029	Venezuela	VEN	2000	72.112	13992.61	24192000.0	1293.12	212.07	10022.18
5057	Vietnam	VNM	2000	73.025	2773.10	79910000.0	2858.19	29539.65	32313.21
5085	Yemen	YEM	2000	60.683	4212.11	17409000.0	14981.53	9066.98	118.54
5113	Zambia	ZMB	2000	44.000	1428.50	10416000.0	11561.74	7217.43	5115.96
5141	Zimbabwe	ZWE	2000	44.649	2211.20	11881000.0	3602.97	6028.13	6006.02

185 rows × 23 columns

```
pandaDF2_drop=pandaDF2.drop(columns=['Entity', 'Code', 'Year'])
```

```
pandaDF3_drop=pandaDF2_drop.drop(columns=['Alcohol ', 'Diet ', 'Smoking ', 'water ', 'BMI ',
```

```
pandaDF3_drop
```

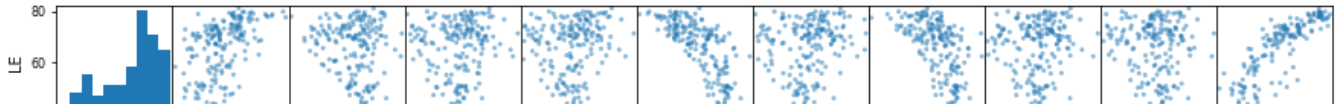
	LE	HRS	logAlcohol	logDiet	logSmoking	logWater	logBMI	logindoor	log
10	55.841	-2.61	5.121043	9.519374	8.910522	9.316649	9.067573	10.251655	8
38	73.955	-0.23	6.762625	7.350792	7.953592	2.402430	7.430820	6.902220	6
66	70.640	-1.94	6.511790	9.377673	9.584756	6.670069	9.697890	5.818568	9
91	78.857	3.35	3.838376	2.927989	4.171614	-4.605170	3.546163	-1.049822	2
119	46.522	-2.55	8.177735	8.117975	8.874335	10.002754	7.752812	9.581777	7
...	...	...	...	...	...	...	...	...	
5029	72.112	-1.03	9.212556	8.530587	9.393360	7.164813	9.442448	5.356916	8
5057	73.025	-0.15	10.383231	9.834629	10.964289	7.957944	9.118498	10.293489	9
5085	60.683	-0.86	4.775250	9.061188	9.045824	9.614573	8.773922	9.112395	8
5113	44.000	-0.28	8.540120	7.319925	8.331827	9.355457	7.613562	8.884254	7
5141	44.649	-0.92	8.700518	7.767222	9.056184	8.189514	8.453109	8.704192	7

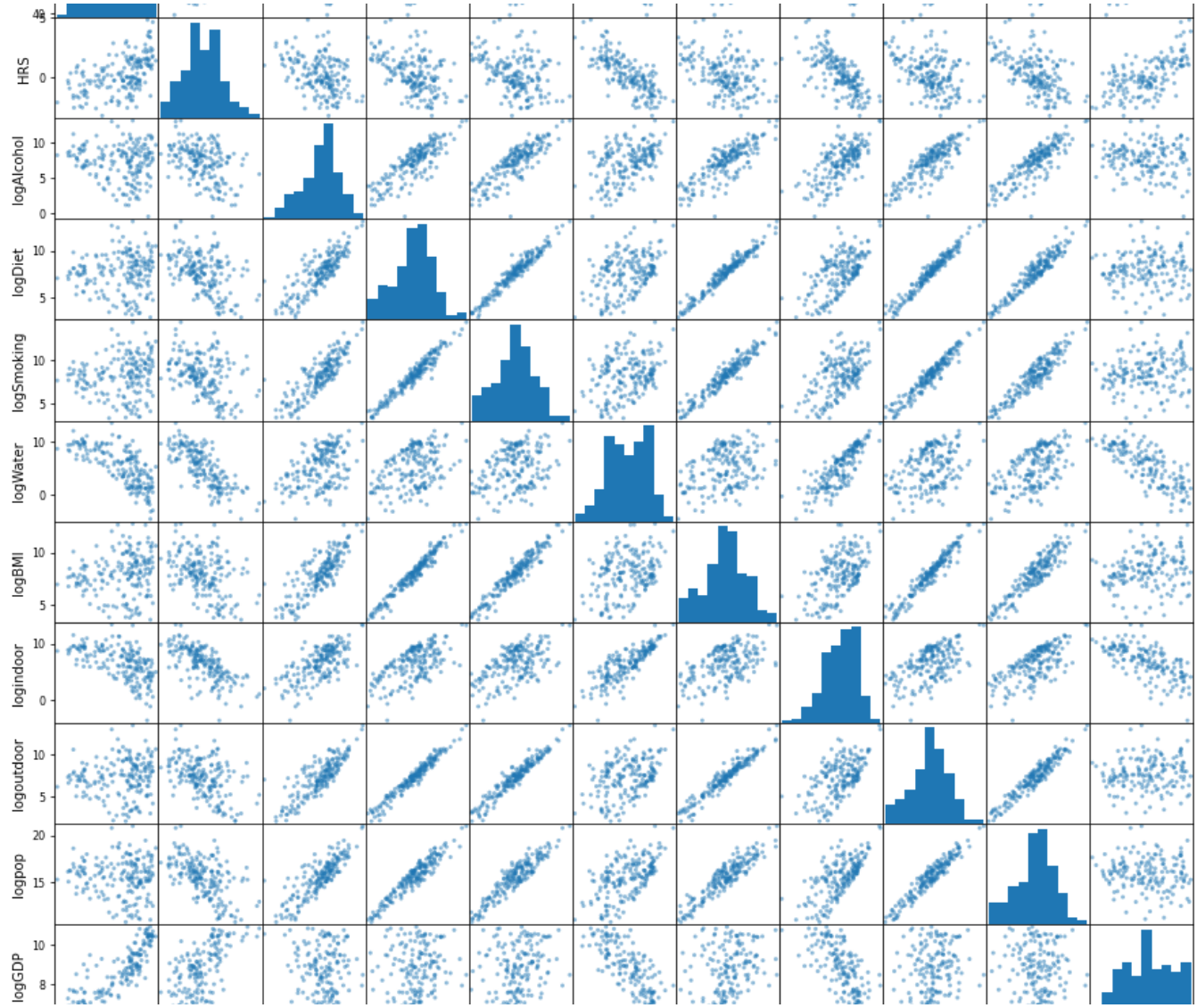
185 rows × 11 columns

```
pd.plotting.scatter_matrix(pandaDF3_drop, alpha=0.5, figsize=(15,15), ax=None, grid=False, di
```

```
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5322828e50>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f53227ea390>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f532281e890>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f53227d4d90>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f53227992d0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f532274d7d0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5322704cd0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f53226c6210>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f532267a710>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5322632c10>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x7f53225f4150>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f53225aa650>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f53225e1b50>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f532258bb90>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5322559590>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f532250fa90>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f53224c5f90>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f53224884d0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f532243e9d0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f53223f4ed0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f53223b5410>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x7f532236a910>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f53223a1e10>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5322364350>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5322319850>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f53222ced50>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5322293290>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5322249790>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f53221fec90>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f53221c11d0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f53221776d0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f532212ebd0>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x7f53220f2110>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5322125610>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f53220dcb10>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5322092fd0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5322054550>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f532200aa50>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321fbff50>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321f82490>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321f37990>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321eeee90>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321eb03d0>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x7f5321e678d0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321e9ddd0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321e5f310>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5328ada090>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5322db8090>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5322c98310>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5322b71690>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5322abc910>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321d0ccd0>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321ccc310>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321c82910>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x7f5321c39f10>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321bfb550>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321bb2b50>,
```

```
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321b75190>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321b2c790>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321b61d90>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321b243d0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321adc9d0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321a87b10>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321a54610>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321a0bc10>],
      [<matplotlib.axes._subplots.AxesSubplot object at 0x7f53219ce250>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321983850>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321939e50>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f53218fd490>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f53218b1a90>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321875110>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f532182b6d0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321861cd0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321823310>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f53217db910>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321790f10>],
      [<matplotlib.axes._subplots.AxesSubplot object at 0x7f532173fb90>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f532170e590>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f53216c3a90>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321678f90>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f532163c4d0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f53215f19d0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f53215a7ed0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f532156a410>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f532159f910>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321555e10>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321519350>],
      [<matplotlib.axes._subplots.AxesSubplot object at 0x7f53214cd850>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321484d50>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321448290>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f53213fd790>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f53213b2c90>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f53213761d0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f532132a6d0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321362bd0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321324110>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f53212da610>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321290b10>],
      [<matplotlib.axes._subplots.AxesSubplot object at 0x7f5321247fd0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321208550>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f53211c5990>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321176f50>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321137490>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f53210ec990>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321123e90>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f53210e53d0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f53210998d0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f532104fdd0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f5321015310>]],
      dtype=object)
```






```
from pyspark.sql import SparkSession
#Create PySpark SparkSession
spark = SparkSession.builder \
    .master("local[1]") \
    .appName("SparkByExamples.com") \
    .getOrCreate()
#Create PySpark DataFrame from Pandas
sparkDF=spark.createDataFrame(pandaDF2)
sparkDF.printSchema()
sparkDF.show()
```

```
root
 |-- Entity: string (nullable = true)
 |-- Code: string (nullable = true)
 |-- Year: long (nullable = true)
 |-- LE: double (nullable = true)
 |-- GDP: double (nullable = true)
 |-- Pop : double (nullable = true)
 |-- water : double (nullable = true)
 |-- indoor: double (nullable = true)
 |-- Alcohol : double (nullable = true)
 |-- BMI : double (nullable = true)
```

```
 |-- Smoking : double (nullable = true)
 |-- Outdoor : double (nullable = true)
 |-- Diet : double (nullable = true)
 |-- HRS : double (nullable = true)
 |-- logAlcohol: double (nullable = true)
 |-- logDiet: double (nullable = true)
 |-- logSmoking: double (nullable = true)
 |-- logWater: double (nullable = true)
 |-- logBMI: double (nullable = true)
 |-- logindoor: double (nullable = true)
 |-- logoutdoor: double (nullable = true)
 |-- logpop: double (nullable = true)
 |-- logGDP: double (nullable = true)
```

```
+------------------+----+----+------------------+--------+---------+--------+--------+-
|            Entity|Code|Year|                LE|     GDP|      Pop |  water |  indoor|A
+------------------+----+----+------------------+--------+---------+--------+--------+-
|       Afghanistan| AFG|2000|            55.841|  502.37|  2.078E7|11121.65|28329.38|
|           Albania| ALB|2000|            73.955| 4808.48|3129000.0|   11.05|  994.48|
|           Algeria| DZA|2000|             70.64| 6834.55| 3.1042E7|  788.45|  336.49|
|           Andorra| AND|2000|            78.857|     NaN|  65000.0|    0.01|    0.35|
|            Angola| AGO|2000|            46.522| 2013.64| 1.6395E7|22087.22|14498.16|
|Antigua and Barbuda| ATG|2000|             73.94|     NaN|  76000.0|    1.31|    2.43|
|         Argentina| ARG|2000| 73.57600000000001|14368.94| 3.6871E7|  420.72| 3717.56|
|           Armenia| ARM|2000| 71.40899999999999| 5139.83|3070000.0|    41.1| 1137.76|
|         Australia| AUS|2000|            79.592|36603.05| 1.8991E7|    9.45|   63.94|
|           Austria| AUT|2000| 78.23899999999999|34796.26|8069000.0|    0.29|  133.54|
|        Azerbaijan| AZE|2000|            66.763| 4214.89|8123000.0|   426.1| 2690.85|
|           Bahamas| BHS|2000|            71.914|     NaN| 298000.0|    3.07|    9.02|
|           Bahrain| BHR|2000|             74.44|17021.72| 665000.0|    3.91|    2.82|
|        Bangladesh| BGD|2000|            65.447| 1485.31|1.27658E8|49136.61|79623.84|
|          Barbados| BRB|2000| 77.16199999999999|12600.02| 272000.0|    3.15|    0.55|
|           Belarus| BLR|2000| 67.36399999999999| 9110.98|9872000.0|    30.6|  618.33|1
|           Belgium| BEL|2000|            77.867|33719.77| 1.0282E7|    7.26|   65.51|
|            Belize| BLZ|2000|            68.847|     NaN| 247000.0|   12.58|   38.18|
|             Benin| BEN|2000|55.391000000000005| 1908.94|6866000.0| 5467.56| 5174.32|
|            Bhutan| BTN|2000| 60.88399999999999|     NaN| 591000.0|  188.41|  287.65|
+------------------+----+----+------------------+--------+---------+--------+--------+-
only showing top 20 rows
```

```
sparkDF.show()
```

```
+------------------+----+----+------------------+--------+---------+--------+--------+-
|            Entity|Code|Year|                LE|     GDP|      Pop |  water |  indoor|A
+------------------+----+----+------------------+--------+---------+--------+--------+-
|       Afghanistan| AFG|2000|            55.841|  502.37|  2.078E7|11121.65|28329.38|
|           Albania| ALB|2000|            73.955| 4808.48|3129000.0|   11.05|  994.48|
|           Algeria| DZA|2000|             70.64| 6834.55| 3.1042E7|  788.45|  336.49|
|           Andorra| AND|2000|            78.857|     NaN|  65000.0|    0.01|    0.35|
|            Angola| AGO|2000|            46.522| 2013.64| 1.6395E7|22087.22|14498.16|
|Antigua and Barbuda| ATG|2000|             73.94|     NaN|  76000.0|    1.31|    2.43|
|         Argentina| ARG|2000| 73.57600000000001|14368.94| 3.6871E7|  420.72| 3717.56|
|           Armenia| ARM|2000| 71.40899999999999| 5139.83|3070000.0|    41.1| 1137.76|
```

```
|          Australia| AUS|2000|            79.592|36603.05| 1.8991E7|    9.45|   63.94|
|            Austria| AUT|2000| 78.23899999999999|34796.26|8069000.0|    0.29|  133.54|
|         Azerbaijan| AZE|2000|            66.763| 4214.89|8123000.0|   426.1| 2690.85|
|            Bahamas| BHS|2000|            71.914|     NaN| 298000.0|    3.07|    9.02|
|            Bahrain| BHR|2000|             74.44|17021.72| 665000.0|    3.91|    2.82|
|         Bangladesh| BGD|2000|            65.447| 1485.31|1.27658E8|49136.61|79623.84|
|           Barbados| BRB|2000| 77.16199999999999|12600.02| 272000.0|    3.15|    0.55|
|            Belarus| BLR|2000| 67.36399999999999| 9110.98|9872000.0|    30.6|  618.33|1
|            Belgium| BEL|2000|            77.867|33719.77| 1.0282E7|    7.26|   65.51|
|             Belize| BLZ|2000|            68.847|     NaN| 247000.0|   12.58|   38.18|
|              Benin| BEN|2000|55.391000000000005| 1908.94|6866000.0| 5467.56| 5174.32|
|             Bhutan| BTN|2000| 60.88399999999999|     NaN| 591000.0|  188.41|  287.65|
+-------------------+----+----+------------------+--------+---------+--------+--------+
only showing top 20 rows
```

```
sparkDFdrop=sparkDF.na.drop()
```

```
sparkDFdrop.show()
```

```
+--------------------+----+----+------------------+--------+---------+--------+--------+
|              Entity|Code|Year|                LE|     GDP|     Pop |  water |  indoor
+--------------------+----+----+------------------+--------+---------+--------+--------+
|         Afghanistan| AFG|2000|            55.841|  502.37|  2.078E7|11121.65|28329.38
|             Albania| ALB|2000|            73.955| 4808.48|3129000.0|   11.05|  994.48
|             Algeria| DZA|2000|             70.64| 6834.55| 3.1042E7|  788.45|  336.49
|              Angola| AGO|2000|            46.522| 2013.64| 1.6395E7|22087.22|14498.16
|           Argentina| ARG|2000| 73.57600000000001|14368.94| 3.6871E7|  420.72| 3717.56
|             Armenia| ARM|2000| 71.40899999999999| 5139.83|3070000.0|    41.1| 1137.76
|           Australia| AUS|2000|            79.592|36603.05| 1.8991E7|    9.45|   63.94
|             Austria| AUT|2000| 78.23899999999999|34796.26|8069000.0|    0.29|  133.54
|          Azerbaijan| AZE|2000|            66.763| 4214.89|8123000.0|   426.1| 2690.85
|             Bahrain| BHR|2000|             74.44|17021.72| 665000.0|    3.91|    2.82
|          Bangladesh| BGD|2000|            65.447| 1485.31|1.27658E8|49136.61|79623.84
|           Barbados| BRB|2000| 77.16199999999999|12600.02| 272000.0|    3.15|    0.55
|            Belarus| BLR|2000| 67.36399999999999| 9110.98|9872000.0|    30.6|  618.33
|            Belgium| BEL|2000|            77.867|33719.77| 1.0282E7|    7.26|   65.51
|              Benin| BEN|2000|55.391000000000005| 1908.94|6866000.0| 5467.56| 5174.32
|            Bolivia| BOL|2000|            62.452| 4173.55|8418000.0| 1370.32|  3007.4
|Bosnia and Herzeg...| BIH|2000|            74.403| 7082.04|3751000.0|    5.28| 1657.34
|            Botswana| BWA|2000|            50.629| 8044.14|1643000.0|  665.39|  391.98
|              Brazil| BRA|2000|            70.116| 9834.42| 1.7479E8|11101.07|25221.52
|            Bulgaria| BGR|2000|            71.501|  8410.8|7998000.0|   11.39| 2730.34
+--------------------+----+----+------------------+--------+---------+--------+--------+
only showing top 20 rows
```

```
sparkDFdrop.select(mean("GDP")).show(truncate=False)
```

```
+------------------+
|avg(GDP)          |
+------------------+
|11295.573836477997|
+------------------+
```

```
from pyspark.sql.functions import col, skewness, kurtosis
sparkDFdrop.select(skewness('GDP'),kurtosis('GDP')).show()
```

```
+------------------+------------------+
|     skewness(GDP)|     kurtosis(GDP)|
+------------------+------------------+
|1.4330526864939621|1.0271548775507586|
+------------------+------------------+
```

```
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.functions import approx_count_distinct,collect_list
from pyspark.sql.functions import collect_set,sum,avg,max,countDistinct,count
from pyspark.sql.functions import first, last, kurtosis, min, mean, skewness
from pyspark.sql.functions import stddev, stddev_samp, stddev_pop, sumDistinct
from pyspark.sql.functions import variance,var_samp,  var_pop
from pyspark.mllib.stat import Statistics
```

```
from pyspark import SparkConf, SparkContext
from pyspark.sql import SQLContext
```

Double-click (or enter) to edit

```
sparkDFdrop.select(sparkDFdrop['Code'], sparkDFdrop['GDP'] + 1).show()
```

```
+----+---------+
|Code|(GDP + 1)|
+----+---------+
| AFG|   503.37|
| ALB|  4809.48|
| DZA|  6835.55|
| AGO|  2014.64|
| ARG| 14369.94|
| ARM|  5140.83|
| AUS| 36604.05|
| AUT| 34797.26|
| AZE|  4215.89|
| BHR| 17022.72|
| BGD|  1486.31|
```

```
| BRB| 12601.02|
| BLR|  9111.98|
| BEL| 33720.77|
| BEN|  1909.94|
| BOL|  4174.55|
| BIH|  7083.04|
| BWA|  8045.14|
| BRA|  9835.42|
| BGR|   8411.8|
+----+---------+
only showing top 20 rows
```

```
sparkDFdrop.filter(sparkDFdrop['LE'] > 65).show()
```

```
+--------------------+----+----+-----------------+--------+------------+--------+------
|              Entity|Code|Year|               LE|     GDP|         Pop |  water |   indo
+--------------------+----+----+-----------------+--------+------------+--------+------
|             Albania| ALB|2000|           73.955| 4808.48|   3129000.0|   11.05|   994
|             Algeria| DZA|2000|            70.64| 6834.55|    3.1042E7|  788.45|   336
|           Argentina| ARG|2000|73.57600000000001|14368.94|    3.6871E7|  420.72|  3717
|             Armenia| ARM|2000|71.40899999999999| 5139.83|   3070000.0|    41.1|  1137
|           Australia| AUS|2000|           79.592|36603.05|    1.8991E7|    9.45|    63
|             Austria| AUT|2000|78.23899999999999|34796.26|   8069000.0|    0.29|   133
|          Azerbaijan| AZE|2000|           66.763| 4214.89|   8123000.0|   426.1|  2690
|             Bahrain| BHR|2000|            74.44|17021.72|    665000.0|    3.91|     2
|          Bangladesh| BGD|2000|           65.447| 1485.31|   1.27658E8|49136.61| 79623
|            Barbados| BRB|2000|77.16199999999999|12600.02|    272000.0|    3.15|     0
|             Belarus| BLR|2000|67.36399999999999| 9110.98|   9872000.0|    30.6|   618
|             Belgium| BEL|2000|           77.867|33719.77|    1.0282E7|    7.26|    65
|Bosnia and Herzeg...| BIH|2000|           74.403| 7082.04|   3751000.0|    5.28|  1657
|              Brazil| BRA|2000|           70.116| 9834.42|    1.7479E8|11101.07| 25221
|            Bulgaria| BGR|2000|           71.501|  8410.8|   7998000.0|   11.39|  2730
|              Canada| CAN|2000|           79.118|36942.56|    3.0588E7|   24.08|    29
|          Cape Verde| CPV|2000|           68.583| 3583.59|    428000.0|   78.76|   153
|               Chile| CHL|2000|           76.366|15211.62|    1.5342E7|    86.4|  1145
|               China| CHN|2000|71.39699999999999| 4730.35|1.29055104E9|25217.51|548103
|            Colombia| COL|2000|           72.945| 8496.79|     3.963E7| 1369.15|  4758
+--------------------+----+----+-----------------+--------+------------+--------+------
only showing top 20 rows
```

```
sparkDFdrop.groupBy("LE").count().show()
```

```
+-----------------+-----+
|               LE|count|
+-----------------+-----+
|           79.592|    1|
|           70.546|    1|
|71.40899999999999|    1|
|           76.812|    1|
|           74.899|    1|
|           69.082|    1|
```

```
|             45.09|    1|
| 77.16199999999999|    1|
|            65.082|    1|
|            58.485|    1|
|            51.941|    1|
|            68.684|    1|
|44.648999999999994|    1|
|            70.852|    1|
|            58.472|    1|
|            51.203|    1|
|            39.441|    1|
| 74.73100000000001|    1|
|            77.779|    1|
|            60.706|    1|
+------------------+-----+
only showing top 20 rows
```

```
sparkDFdrop.createOrReplaceTempView("LE")

sqlDF = spark.sql("SELECT * FROM LE")
sqlDF.show()
```

```
+--------------------+----+----+------------------+--------+---------+--------+--------+
|              Entity|Code|Year|                LE|     GDP|     Pop |  water |  indoor
+--------------------+----+----+------------------+--------+---------+--------+--------+
|         Afghanistan| AFG|2000|            55.841|  502.37|  2.078E7|11121.65|28329.38
|             Albania| ALB|2000|            73.955| 4808.48|3129000.0|   11.05|  994.48
|             Algeria| DZA|2000|             70.64| 6834.55| 3.1042E7|  788.45|  336.49
|              Angola| AGO|2000|            46.522| 2013.64| 1.6395E7|22087.22|14498.16
|           Argentina| ARG|2000| 73.57600000000001|14368.94| 3.6871E7|  420.72| 3717.56
|             Armenia| ARM|2000| 71.40899999999999| 5139.83|3070000.0|    41.1| 1137.76
|           Australia| AUS|2000|            79.592|36603.05| 1.8991E7|    9.45|   63.94
|             Austria| AUT|2000| 78.23899999999999|34796.26|8069000.0|    0.29|  133.54
|          Azerbaijan| AZE|2000|            66.763| 4214.89|8123000.0|   426.1| 2690.85
|             Bahrain| BHR|2000|             74.44|17021.72| 665000.0|    3.91|    2.82
|          Bangladesh| BGD|2000|            65.447| 1485.31|1.27658E8|49136.61|79623.84
|            Barbados| BRB|2000| 77.16199999999999|12600.02| 272000.0|    3.15|    0.55
|             Belarus| BLR|2000| 67.36399999999999| 9110.98|9872000.0|    30.6|  618.33
|             Belgium| BEL|2000|            77.867|33719.77| 1.0282E7|    7.26|   65.51
|               Benin| BEN|2000|55.391000000000005| 1908.94|6866000.0| 5467.56| 5174.32
|             Bolivia| BOL|2000|            62.452| 4173.55|8418000.0| 1370.32|  3007.4
|Bosnia and Herzeg...| BIH|2000|            74.403| 7082.04|3751000.0|    5.28| 1657.34
|            Botswana| BWA|2000|            50.629| 8044.14|1643000.0|  665.39|  391.98
|              Brazil| BRA|2000|            70.116| 9834.42| 1.7479E8|11101.07|25221.52
|            Bulgaria| BGR|2000|            71.501|  8410.8|7998000.0|   11.39| 2730.34
+--------------------+----+----+------------------+--------+---------+--------+--------+
only showing top 20 rows
```

SQL queries

```
#1. SQL query with ORDER BY
sqlDF.select(sqlDF["GDP"], sqlDF["LE"]).where(col("GDP") > 10000).orderBy(desc("GDP")).show(1
```

```
+--------+------+
|     GDP|    LE|
+--------+------+
|50063.82|77.652|
|48888.03|74.327|
|45886.47|76.812|
|45788.12|77.467|
| 43251.1|79.858|
|39021.18|76.712|
| 38806.5|76.852|
|37899.95|78.187|
|37772.76|77.969|
|36942.56|79.118|
+--------+------+
only showing top 10 rows
```

```
#2. SQL query with BETWEEN
sqlDF.select(sqlDF["GDP"].between(10000, 30000)).show(5)
```

```
+-----------------------------------+
|((GDP >= 10000) AND (GDP <= 30000))|
+-----------------------------------+
|                              false|
|                              false|
|                              false|
|                              false|
|                               true|
+-----------------------------------+
only showing top 5 rows
```

```
#3. SQL query with min, max and mean
sqlDF.agg({'GDP': 'max'}).show()
sqlDF.agg({'GDP': 'min'}).show()
sqlDF.agg({'GDP': 'mean'}).show()
```

```
+--------+
|max(GDP)|
+--------+
|50063.82|
+--------+

+--------+
|min(GDP)|
+--------+
|  414.05|
+--------+
```

```
+------------------+
|          avg(GDP)|
+------------------+
|11295.573836477997|
+------------------+
```

```
#4. SQL query with like
sqlDF.select("Code",sqlDF["Entity"].like("A%")).show(5)
```

```
+----+--------------+
|Code|Entity LIKE A%|
+----+--------------+
| AFG|          true|
| ALB|          true|
| DZA|          true|
| AGO|          true|
| ARG|          true|
+----+--------------+
only showing top 5 rows
```

```
sqlDF.select("Code", sqlDF.Entity.endswith("a")).show(n=10)
```

```
+----+-------------------+
|Code|endswith(Entity, a)|
+----+-------------------+
| AFG|              false|
| ALB|               true|
| DZA|               true|
| AGO|               true|
| ARG|               true|
| ARM|               true|
| AUS|               true|
| AUT|               true|
| AZE|              false|
| BHR|              false|
+----+-------------------+
only showing top 10 rows
```

```
sqlDF.select("Code", sqlDF.Entity.startswith("A")).show(n=10, truncate=False)
```

```
+----+---------------------+
|Code|startswith(Entity, A)|
+----+---------------------+
|AFG |true                 |
|ALB |true                 |
|DZA |true                 |
|AGO |true                 |
|ARG |true                 |
|ARM |true                 |
|AUS |true                 |
```

```
|AUT |true                |
|AZE |true                |
|BHR |false               |
+----+--------------------+
only showing top 10 rows
```

```
#5. SQL query with group by and order by
sqlDF.select("Entity").where(col("Entity").isNotNull()).groupBy("Entity").count().orderBy("co
```

```
+-----------+-----+
|Entity     |count|
+-----------+-----+
|Chad       |1    |
|Paraguay   |1    |
|Russia     |1    |
|Yemen      |1    |
|Senegal    |1    |
|Sweden     |1    |
|Philippines|1    |
|Djibouti   |1    |
|Malaysia   |1    |
|Singapore  |1    |
+-----------+-----+
only showing top 10 rows
```

Double-click (or enter) to edit

Random Forest Regression Analysis

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.inspection import permutation_importance
```

```
pandaDF3_dropnull=pandaDF3_drop.dropna()
```

```
import random
random.seed(123)
```

```
feature_names=['HRS ', 'logAlcohol', 'logDiet', 'logSmoking', 'logWater', 'logBMI', 'logindoo
X = pd.DataFrame(pandaDF3_dropnull, columns=feature_names)
y = pandaDF3_dropnull.LE
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=12)
```

```
X_train
```

	HRS	logAlcohol	logDiet	logSmoking	logWater	logBMI	logindoor	logoutdoo
1097	1.63	6.733152	6.571611	7.399618	3.869116	7.297525	5.593000	6.27144
2076	0.94	9.906702	9.301336	10.307491	1.729884	9.926525	7.883823	8.80233
567	0.15	7.370659	7.692688	8.074714	7.222800	8.074605	8.008831	7.52228
3317	3.43	6.817524	7.348813	8.460064	0.698135	7.948756	2.705380	6.75404
2020	-0.82	7.817714	8.209716	8.171732	8.491783	8.062943	8.837280	6.75434
...	...	...	...	...	...	...	...	..
119	-2.55	8.177735	8.117975	8.874335	10.002754	7.752812	9.581777	7.95673
4238	-0.70	10.927496	9.375142	10.764683	9.754409	10.634066	9.546614	9.74030
4357	-3.23	8.365111	9.546891	7.909655	9.599719	9.409492	9.818140	8.47449
5057	-0.15	10.383231	9.834629	10.964289	7.957944	9.118498	10.293489	9.59308
2412	-0.28	9.532163	9.764791	10.124997	5.635218	10.195697	8.512450	8.77135

119 rows × 10 columns

```
rf = RandomForestRegressor(n_estimators=100, oob_score=True)
rf.fit(X_train, y_train)
```

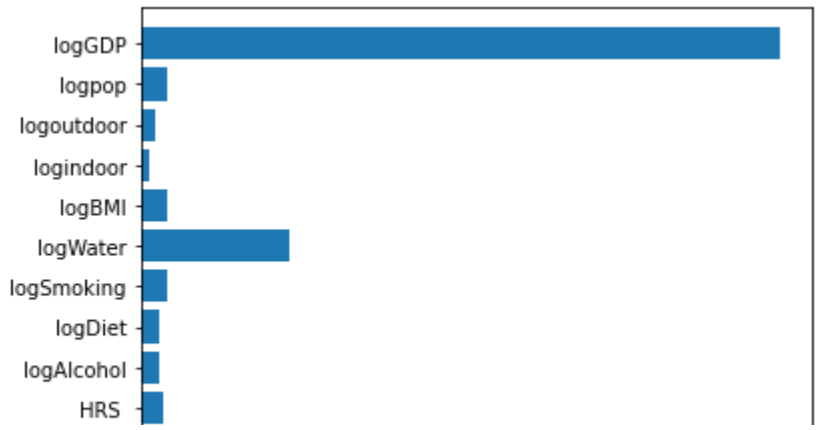
```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning: X does not have
  "X does not have valid feature names, but"
RandomForestRegressor(oob_score=True)
```

```
rf.feature_importances_
```

```
array([0.0233963 , 0.01860784, 0.01983792, 0.02829134, 0.15692406,
       0.02859029, 0.00827395, 0.01478526, 0.02862626, 0.67266678])
```

```
plt.barh(feature_names, rf.feature_importances_)
plt.xlabel("Random Forest Feature Importance")
```

```
Text(0.5, 0, 'Random Forest Feature Importance')
```




```
# Predicting the Test set results
y_pred = rf.predict(X_test)


# Evaluating the Algorithm
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 2.927326500000008
Mean Squared Error: 14.633554043975062
Root Mean Squared Error: 3.82538286240411
```

```
from sklearn.metrics import r2_score
from scipy.stats import spearmanr, pearsonr
predicted_train = rf.predict(X_train)
predicted_test = rf.predict(X_test)
test_score = r2_score(y_test, predicted_test)
spearman = spearmanr(y_test, predicted_test)
pearson = pearsonr(y_test, predicted_test)
print(f'Out-of-bag R-2 score estimate: {rf.oob_score_:>5.3}')
print(f'Test data R-2 score: {test_score:>5.3}')
print(f'Test data Spearman correlation: {spearman[0]:.3}')
print(f'Test data Pearson correlation: {pearson[0]:.3}')
```

```
Out-of-bag R-2 score estimate: 0.766
Test data R-2 score: 0.881
Test data Spearman correlation: 0.922
Test data Pearson correlation: 0.94
```

Double-click (or enter) to edit

Multiple linear regression model analysis

```
from pyspark.ml.feature import VectorAssembler


features_cols=list(sparkDFdrop.columns)
features_cols
```

```
['Entity',
 'Code',
 'Year',
 'LE',
 'GDP',
 'Pop ',
 'water ',
 'indoor',
 'Alcohol ',
 'BMI ',
 'Smoking ',
 'Outdoor ',
 'Diet ',
 'HRS ',
 'logAlcohol',
 'logDiet',
 'logSmoking',
 'logWater',
 'logBMI',
 'logindoor',
 'logoutdoor',
 'logpop',
 'logGDP']
```

```
features_cols.remove('LE')
features_cols.remove('Entity')
features_cols.remove('Code')
features_cols.remove('Year')


features_cols.remove('Pop ')
features_cols.remove('water ')
features_cols.remove('indoor')
features_cols.remove('Outdoor ')
features_cols.remove('Alcohol ')
features_cols.remove('BMI ')
features_cols.remove('Diet ')
features_cols.remove('Smoking ')
features_cols.remove('GDP')


vec=VectorAssembler(inputCols=features_cols, outputCol='features')
```

```
df=vec.transform(sparkDFdrop)
```

```
ml_ready_df=df.select(['LE', 'features'])
```

```
ml_ready_df.show(5)
```

```
+-----------------+--------------------+
|               LE|            features|
+-----------------+--------------------+
|           55.841|[-2.61,5.12104305...|
|           73.955|[-0.23,6.76262545...|
|            70.64|[-1.94,6.51178990...|
|           46.522|[-2.55,8.17773490...|
|73.57600000000001|[-0.27,8.94870414...|
+-----------------+--------------------+
only showing top 5 rows
```

```
from pyspark.sql.functions import rand
```

```
import random
random.seed(123)
```

```
splits = ml_ready_df.randomSplit([0.75, 0.25])
train_df = splits[0]
test_df = splits[1]
```

```
from pyspark.ml.regression import LinearRegression
lr = LinearRegression(featuresCol = 'features', labelCol='LE', maxIter=10, regParam=0.3, elas
lr_model = lr.fit(train_df)
print("Coefficients: " + str(lr_model.coefficients))
print("Intercept: " + str(lr_model.intercept))
```

```
Coefficients: [-0.15804686841402,-0.5158848116539819,0.35723941350459776,0.411304228120
Intercept: 25.58605148899589
```

```
trainingSummary = lr_model.summary
print("RMSE: %f" % trainingSummary.rootMeanSquaredError)
print("R^2: %f" % trainingSummary.r2)
```

```
RMSE: 5.156488
R^2: 0.765020
```

```
lr_predictions = lr_model.transform(test_df)
```

```
lr_predictions.select("prediction","LE","features").show(5)
```

```
from pyspark.ml.evaluation import RegressionEvaluator
lr_evaluator = RegressionEvaluator(predictionCol="prediction", \
                labelCol="LE",metricName="r2")
print("R Squared (R2) on test data = %g" % lr_evaluator.evaluate(lr_predictions))
```

```
+------------------+------------------+--------------------+
|        prediction|                LE|            features|
+------------------+------------------+--------------------+
| 54.25793546530106|            46.229|[-1.52,9.44798783...|
| 60.25835546236745|             47.49|[0.43,6.694735310...|
|54.506866469946374|48.946000000000005|[-0.64,8.44340455...|
| 60.54450862567883|52.123000000000005|[-1.68,7.36300053...|
| 62.57950315262083|            52.192|[-0.29,6.74497749...|
+------------------+------------------+--------------------+
only showing top 5 rows

R Squared (R2) on test data = 0.785412
```

Store data in SQL Database

```
conn=sqlite3.connect("Lifeexpectancy.db")
print(conn)
```

```
<sqlite3.Connection object at 0x7f532315cc70>
```

```
conn.execute("DROP TABLE IF EXISTS `Lifeexpectancy`")
print("Table dropped")
conn=sqlite3.connect("Lifeexpectancy.db")
try:
  conn.execute('''CREATE TABLE Lifeexpectancy
              (Entity       TEXT NOT NULL,
               Code       TEXT  NOT NULL,
               Year     INTEGER,
               Lifeexpectancy     FLOAT DEFAULT 0,
               GDPpercap     FLOAT DEFAULT 0,
               Population     FLOAT DEFAULT 0,
               WaterDeaths    FLOAT DEFAULT 0,
               IndoorDeaths   FLOAT DEFAULT 0,
               AlcoholDeaths  FLOAT DEFAULT 0,
               HighBMIDeaths  FLOAT DEFAULT 0,
               SmokingDeaths  FLOAT DEFAULT 0,
               OutdoorDeaths  FLOAT DEFAULT 0,
               DietDeaths     FLOAT DEFAULT 0,
               HRS            FLOAT DEFAULT 0,
               logAlcohol     FLOAT DEFAULT 0,
               logDiet        FLOAT DEFAULT 0,
               logSmoking     FLOAT DEFAULT 0,
```

```
                logWater       FLOAT DEFAULT 0,
                logBMI         FLOAT DEFAULT 0,
                logIndoor      FLOAT DEFAULT 0,
                logOutdoor     FLOAT DEFAULT 0,
                logPop         FLOAT DEFAULT 0,
                logGDP         FLOAT DEFAULT 0               );''')
  print("Table created successfully")
except Exception as e:
  print(str(e))
  print('Table creation failed!!!')
finally:
  conn.close()
```

```
    Table dropped
    Table created successfully
```

```
Lifeexpectancy_list=pandaDF2.values.tolist()
Lifeexpectancy_list
```

```
conn = sqlite3.connect("Lifeexpectancy.db")

cursor = conn.cursor()

try:
  cursor.executemany('''
        INSERT INTO Lifeexpectancy (Entity, Code, Year,
        Lifeexpectancy, GDPpercap, Population, WaterDeaths,
        IndoorDeaths, AlcoholDeaths, HighBMIDeaths, SmokingDeaths, OutdoorDeaths,
        DietDeaths, HRS, logAlcohol, logDiet, logSmoking, logWater, logBMI,
        logIndoor, logOutdoor, logPop, logGDP)
        VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)''', Lifeexpectancy_list)
  conn.commit()
  print("Data Insterted successfully")
except Exception as e:
  print(str(e))
  print("Insertion failed!")
finally:
  conn.close()
```

```
    Data Insterted successfully
```

```
conn = sqlite3.connect("Lifeexpectancy.db")

try:
  df = pd.read_sql_query('''
              SELECT GDPpercap,
              count(*) as N,
              avg(GDPpercap) as mean_GDPpercap,
```

```
                max(GDPpercap) as max_GDPpercap,
                min(GDPpercap) as min_GDPpercap
        FROM Lifeexpectancy ;''', conn)
except Exception as e:
  print(str(e))
finally:
  conn.close()
df
```

	GDPpercap	N	mean_GDPpercap	max_GDPpercap	min_GDPpercap
0	414.05	185	11750.487901	54039.96	414.05