```
pip install pyspark
```

Show hidden output

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local").appName('Global Country Data').getOrCreate()
```

```
spark
```

SparkSession - in-memory

SparkContext

Spark UI

Version
    `v3.5.3`
Master
    `local`
AppName
    `Global Country Data`

```
df=spark.read.csv("/content/world-data-2023.csv",header=True,inferSchema=True,multiLine=True)
```

```
df.show()
```

```
+------------------+---------------+------------+--------------------+-------------+----------------+----------+------------+-------------------+-------------+------+---
|           Country|Density\n(P/Km2)|Abbreviation|Agricultural Land( %)|Land Area(Km2)|Armed Forces size|Birth Rate|Calling Code|  Capital/Major City|Co2-Emissions|   CPI|CPI
+------------------+---------------+------------+--------------------+-------------+----------------+----------+------------+-------------------+-------------+------+---
|       Afghanistan|             60|          AF|              58.10%|      652,230|         323,000|     32.49|          93|              Kabul|        8,672| 149.9|
|           Albania|            105|          AL|              43.10%|       28,748|           9,000|     11.78|         355|             Tirana|        4,536|119.05|
|           Algeria|             18|          DZ|              17.40%|    2,381,741|         317,000|     24.28|         213|            Algiers|      150,006|151.36|
|           Andorra|            164|          AD|              40.00%|          468|            NULL|       7.2|         376|   Andorra la Vella|          469|  NULL|
|            Angola|             26|          AO|              47.50%|    1,246,700|         117,000|     40.73|         244|             Luanda|       34,693|261.73|
|Antigua and Barbuda|            223|          AG|              20.50%|          443|               0|     15.33|           1|St. John's, Saint...|          557|113.81|
|         Argentina|             17|          AR|              54.30%|    2,780,400|         105,000|     17.02|          54|       Buenos Aires|      201,348|232.75|
|           Armenia|            104|          AM|              58.90%|       29,743|          49,000|     13.99|         374|            Yerevan|        5,156|129.18|
|         Australia|              3|          AU|              48.20%|    7,741,220|          58,000|      12.6|          61|           Canberra|      375,908| 119.8|
|           Austria|            109|          AT|              32.40%|       83,871|          21,000|       9.7|          43|             Vienna|       61,448|118.06|
|        Azerbaijan|            123|          AZ|              57.70%|       86,600|          82,000|      14.0|         994|               Baku|       37,620|156.32|
|       The Bahamas|             39|          BS|               1.40%|       13,880|           1,000|     13.97|           1|    Nassau, Bahamas|        1,786|116.22|
|           Bahrain|          2,239|          BH|              11.10%|          765|          19,000|     13.99|         973|             Manama|       31,694|117.59|
|        Bangladesh|          1,265|          BD|              70.60%|      148,460|         221,000|     18.18|         880|              Dhaka|       84,246|179.68|
|          Barbados|            668|          BB|              23.30%|          430|           1,000|     10.65|           1|         Bridgetown|        1,276|134.09|
|           Belarus|             47|          BY|              42.00%|      207,600|         155,000|       9.9|         375|              Minsk|       58,280|  NULL|
|           Belgium|            383|          BE|              44.60%|       30,528|          32,000|      10.3|          32|   City of Brussels|       96,889|117.11|
|            Belize|             17|          BZ|               7.00%|       22,966|           2,000|     20.79|         501|           Belmopan|          568|105.68|
|             Benin|            108|          BJ|              33.30%|      112,622|          12,000|     36.22|         229|         Porto-Novo|        6,476|110.71|
|            Bhutan|             20|          BT|              13.60%|       38,394|           6,000|     17.26|         975|            Thimphu|        1,261|167.18|
+------------------+---------------+------------+--------------------+-------------+----------------+----------+------------+-------------------+-------------+------+---
only showing top 20 rows
```

```
df.toPandas().shape
```

```
(195, 35)
```

```
# Dropping unwanted columns
df=df.drop('Abbreviation','Currency-Code')
```

```
df = df.withColumnRenamed("Density\n(P/Km2)", "Density (P/Km2)") \
           .withColumnRenamed("Longitude\r", "Longitude")
```

```
from pyspark.sql.functions import col, regexp_replace
df = df.select([regexp_replace(col(c), r'[^\x00-\x7F]+', '').alias(c) for c in df.columns])
df = df.select([regexp_replace(col(c), r'\%', '').alias(c) for c in df.columns])
```

```
df = df.select([regexp_replace(col(c), r'\$', '').alias(c) for c in df.columns])
df = df.select([regexp_replace(col(c), r'\,', '').alias(c) for c in df.columns])
df = df.select([regexp_replace(col(c), r'\r', '').alias(c) for c in df.columns])
```

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

# Strip non-numeric characters for columns containing numerical values as strings, then cast to float
columns_to_cast = ['Density (P/Km2)', 'Agricultural Land( %)', 'Land Area(Km2)', 'Armed Forces size',
                   'Co2-Emissions', 'CPI', 'CPI Change (%)', 'Forested Area (%)',
                   'Gasoline Price', 'Gross primary education enrollment (%)',
                   'Gross tertiary education enrollment (%)', 'Minimum wage',
                   'Out of pocket health expenditure', 'Population', 'Population: Labor force participation (%)',
                   'Tax revenue (%)', 'Total tax rate', 'Unemployment rate', 'Urban_population']
```

```
for col in columns_to_cast:
    df= df.withColumn(col, F.regexp_replace(F.col(col), '[^0-9.]', '').cast(FloatType()))
```

```
# Typecast relevant columns to numeric types (float or integer)
from pyspark.sql.functions import *
df = df.withColumn("Calling Code", col("Calling Code").cast(IntegerType())) \
           .withColumn("Maternal mortality ratio",col("Maternal mortality ratio").cast(IntegerType())) \
           .withColumn("Birth Rate", col("Birth Rate").cast(DoubleType())) \
           .withColumn("Fertility Rate", col("Fertility Rate").cast(DoubleType())) \
           .withColumn("Infant mortality", col("Infant mortality").cast(DoubleType())) \
           .withColumn("Life expectancy", col("Life expectancy").cast(DoubleType())) \
           .withColumn("Physicians per thousand", col("Physicians per thousand").cast(DoubleType())) \
           .withColumn("Latitude", col("Latitude").cast(DoubleType())) \
           .withColumn("Longitude", col("Longitude").cast(DoubleType()))
```

```
df.printSchema()
```

```
root
 |-- Country: string (nullable = true)
 |-- Density (P/Km2): float (nullable = true)
 |-- Agricultural Land( %): float (nullable = true)
 |-- Land Area(Km2): float (nullable = true)
 |-- Armed Forces size: float (nullable = true)
 |-- Birth Rate: double (nullable = true)
 |-- Calling Code: integer (nullable = true)
 |-- Capital/Major City: string (nullable = true)
 |-- Co2-Emissions: float (nullable = true)
 |-- CPI: float (nullable = true)
 |-- CPI Change (%): float (nullable = true)
 |-- Fertility Rate: double (nullable = true)
 |-- Forested Area (%): float (nullable = true)
 |-- Gasoline Price: float (nullable = true)
 |-- GDP: string (nullable = true)
 |-- Gross primary education enrollment (%): float (nullable = true)
 |-- Gross tertiary education enrollment (%): float (nullable = true)
 |-- Infant mortality: double (nullable = true)
 |-- Largest city: string (nullable = true)
 |-- Life expectancy: double (nullable = true)
 |-- Maternal mortality ratio: integer (nullable = true)
 |-- Minimum wage: float (nullable = true)
 |-- Official language: string (nullable = true)
 |-- Out of pocket health expenditure: float (nullable = true)
 |-- Physicians per thousand: double (nullable = true)
 |-- Population: float (nullable = true)
 |-- Population: Labor force participation (%): float (nullable = true)
 |-- Tax revenue (%): float (nullable = true)
 |-- Total tax rate: float (nullable = true)
 |-- Unemployment rate: float (nullable = true)
 |-- Urban_population: float (nullable = true)
 |-- Latitude: double (nullable = true)
 |-- Longitude: double (nullable = true)
```

```
categorical_cols=['Country','Capital/Major City','Largest city','Official language']
```

```
from pyspark.ml.feature import StringIndexer
indexers=[StringIndexer(inputCol=c,outputCol=f"{c}_indexer",handleInvalid="keep")for c in categorical_cols]
```

```
df=indexers[0].fit(df).transform(df)
df=indexers[1].fit(df).transform(df)
df=indexers[2].fit(df).transform(df)
df=indexers[3].fit(df).transform(df)
```

```
df.printSchema()
```

Show hidden output

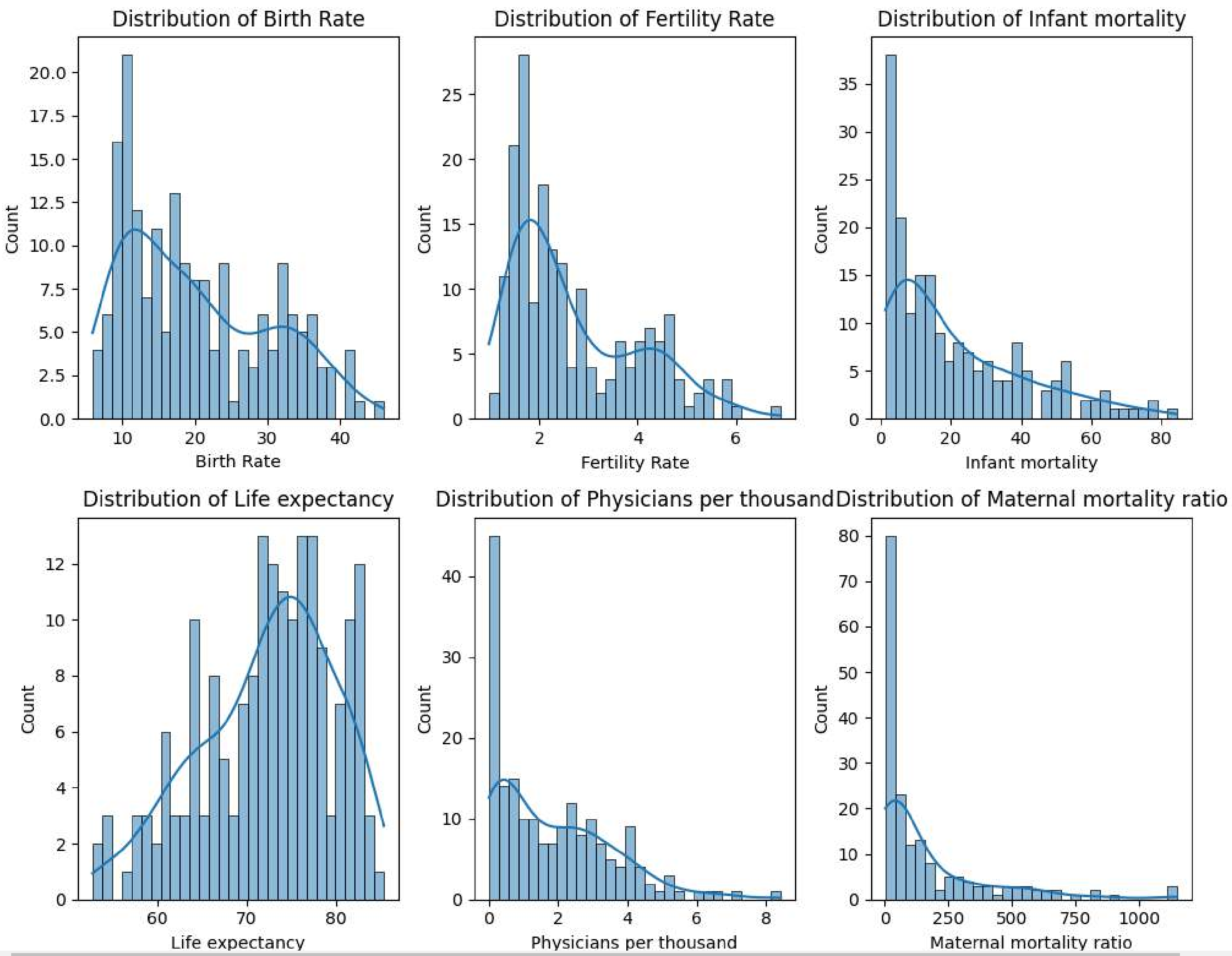
```
# Checking the shape of the dataset
df.toPandas().shape
```

```
(195, 37)
```

```
df.toPandas().isnull().sum()
```

Show hidden output

```
import matplotlib.pyplot as plt
import seaborn as sns

# Convert a PySpark DataFrame to Pandas for plotting
numeric_cols = ['Birth Rate', 'Fertility Rate', 'Infant mortality', 'Life expectancy',
                'Physicians per thousand', 'Maternal mortality ratio']

df_pd = df.select(numeric_cols).toPandas()

# Plot histograms to visually inspect the distribution
plt.figure(figsize=(10, 8))
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(2, 3, i)
    sns.histplot(df_pd[col].dropna(), kde=True, bins=30)
    plt.title(f"Distribution of {col}")
plt.tight_layout()
plt.show()
```




```
# Handling null values
from pyspark.ml.feature import Imputer
# Impute missing numerical values with median

numerical_columns = ['Population','CPI','Armed Forces size','Gross tertiary education enrollment (%)','CPI Change (%)',
                     'Co2-Emissions', 'Tax revenue (%)', 'Life expectancy',
                     'Unemployment rate','Urban_population','Population: Labor force participation (%)',
                     'Minimum wage','Maternal mortality ratio','Gasoline Price','Forested Area (%)','Fertility Rate',
                     'Agricultural Land( %)','Total tax rate','Land Area(Km2)',
                     'Physicians per thousand','Out of pocket health expenditure',
                     'Infant mortality','Gross primary education enrollment (%)','Birth Rate']

imputer=Imputer(inputCols=numerical_columns,outputCols=numerical_columns)
imputer.setStrategy("median")
df=imputer.fit(df).transform(df)
```

```
df.show()
```

Country	Density (P/Km2)	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Calling Code	Capital/Major City	Co2-Emissions	CPI	CPI Change (%)	Fe
Afghanistan	60.0	58.1	652230.0	323000.0	32.49	93	Kabul	8672.0	149.9	2.3	
Albania	105.0	43.1	28748.0	9000.0	11.78	355	Tirana	4536.0	119.05	1.4	
Algeria	18.0	17.4	2381741.0	317000.0	24.28	213	Algiers	150006.0	151.36	2.0	
Andorra	164.0	40.0	468.0	31000.0	7.2	376	Andorra la Vella	469.0	125.08	2.5	
Angola	26.0	47.5	1246700.0	117000.0	40.73	244	Luanda	34693.0	261.73	17.1	
Antigua and Barbuda	223.0	20.5	443.0	0.0	15.33	1	St. John's Saint ...	557.0	113.81	1.2	
Argentina	17.0	54.3	2780400.0	105000.0	17.02	54	Buenos Aires	201348.0	232.75	53.5	
Armenia	104.0	58.9	29743.0	49000.0	13.99	374	Yerevan	5156.0	129.18	1.4	
Australia	3.0	48.2	7741220.0	58000.0	12.6	61	Canberra	375908.0	119.8	1.6	
Austria	109.0	32.4	83871.0	21000.0	9.7	43	Vienna	61448.0	118.06	1.5	
Azerbaijan	123.0	57.7	86600.0	82000.0	14.0	994	Baku	37620.0	156.32	2.6	
The Bahamas	39.0	1.4	13880.0	1000.0	13.97	1	Nassau Bahamas	1786.0	116.22	2.5	
Bahrain	2239.0	11.1	765.0	19000.0	13.99	973	Manama	31694.0	117.59	2.1	
Bangladesh	1265.0	70.6	148460.0	221000.0	18.18	880	Dhaka	84246.0	179.68	5.6	
Barbados	668.0	23.3	430.0	1000.0	10.65	1	Bridgetown	1276.0	134.09	4.1	
Belarus	47.0	42.0	207600.0	155000.0	9.9	375	Minsk	58280.0	125.08	5.6	
Belgium	383.0	44.6	30528.0	32000.0	10.3	32	City of Brussels	96889.0	117.11	1.4	
Belize	17.0	7.0	22966.0	2000.0	20.79	501	Belmopan	568.0	105.68	0.9	
Benin	108.0	33.3	112622.0	12000.0	36.22	229	Porto-Novo	6476.0	110.71	0.9	
Bhutan	20.0	13.6	38394.0	6000.0	17.26	975	Thimphu	1261.0	167.18	2.7	

only showing top 20 rows

```
# Removing  duplicate rows

df = df.dropDuplicates()
```

```
df.toPandas().shape
```

(195, 37)

```
df.toPandas().isnull().sum()
```

Show hidden output

```
#dropping the rows with null value

df=df.dropna(how="any")
```

```
df.toPandas().isnull().sum()
```

Show hidden output

```
df.toPandas().shape
```

```
(188, 37)
```

```
df.toPandas().describe().transpose()
```

Show hidden output

```
df.show()
```

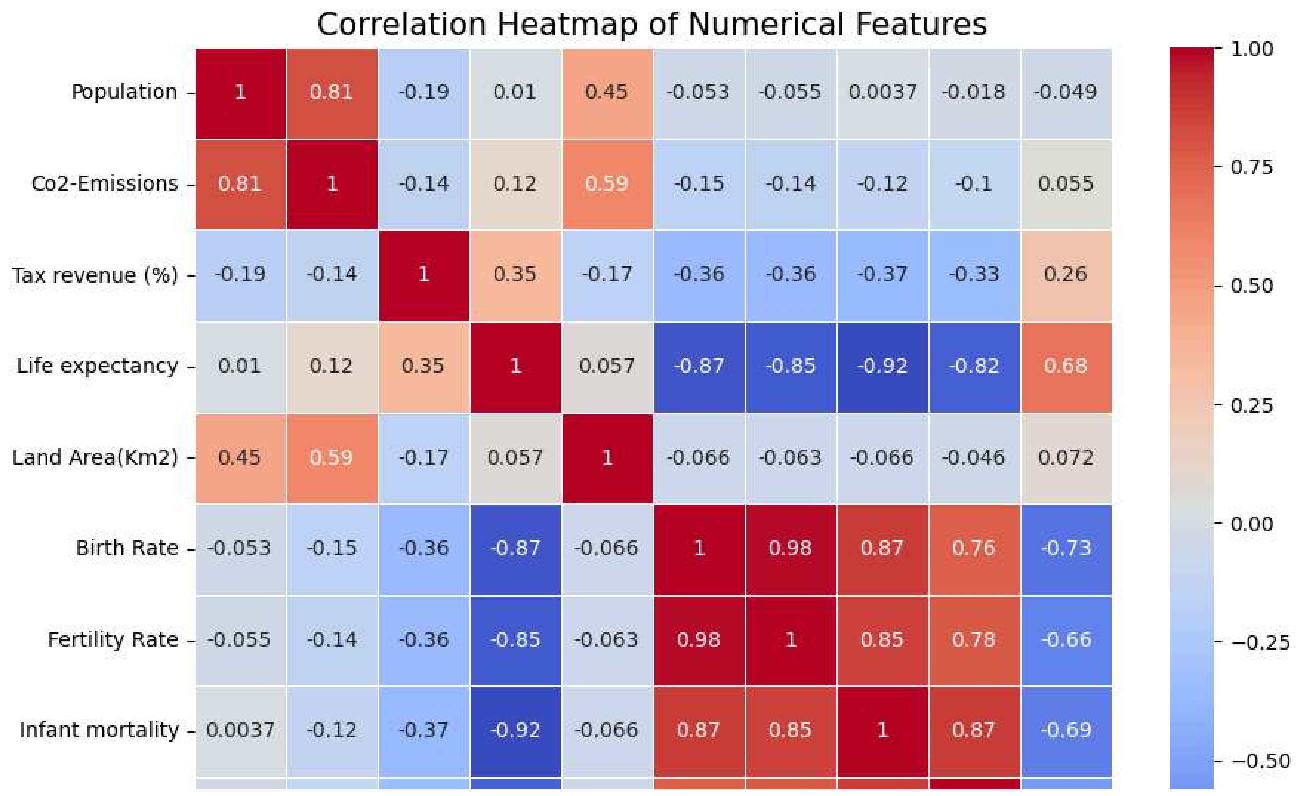
Country	Density (P/Km2)	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Calling Code	Capital/Major City	Co2-Emissions	CPI	CPI Change (%)	Fe
Nigeria	226.0	77.7	923768.0	215000.0	37.91	234	Abuja	120369.0	267.51	11.4	
Laos	32.0	10.3	236800.0	129000.0	23.55	856	Vientiane	17763.0	135.87	3.3	
Saint Vincent and...	284.0	25.6	389.0	31000.0	14.24	1	Kingstown	220.0	109.67	2.3	
Mauritania	5.0	38.5	1030700.0	21000.0	33.69	222	Nouakchott	2739.0	135.02	2.3	
Guinea-Bissau	70.0	58.0	36125.0	4000.0	35.13	245	Bissau	293.0	111.65	1.4	
The Gambia	239.0	59.8	11300.0	1000.0	38.54	220	Banjul	532.0	172.73	7.1	
Seychelles	214.0	3.4	455.0	0.0	17.1	248	Victoria Seychelles	605.0	129.96	1.8	
United Arab Emirates	118.0	5.5	83600.0	63000.0	10.33	971	Abu Dhabi	206324.0	114.52	1.9	
Guinea	53.0	59.0	245857.0	13000.0	36.36	224	Conakry	2996.0	262.95	9.5	
Nicaragua	55.0	42.1	130370.0	12000.0	20.64	505	Managua	5592.0	162.74	5.4	
Switzerland	219.0	38.4	41277.0	21000.0	10.0	41	Bern	34477.0	99.55	0.4	
Madagascar	48.0	71.2	587041.0	22000.0	32.66	261	Antananarivo	3905.0	184.33	5.6	
Palau	39.0	10.9	459.0	31000.0	14.0	680	Ngerulmud	224.0	118.17	1.3	
Netherlands	508.0	53.3	41543.0	41000.0	9.7	31	Amsterdam	170780.0	115.91	2.6	
Iceland	3.0	18.7	103000.0	0.0	12.0	354	Reykjav	2065.0	129.0	3.0	
Mexico	66.0	54.6	1964375.0	336000.0	17.6	52	Mexico City	486406.0	141.54	3.6	
Somalia	25.0	70.3	637657.0	20000.0	41.75	252	Mogadishu	645.0	125.08	2.5	
Spain	94.0	52.6	505370.0	196000.0	7.9	34	Madrid	244002.0	110.96	0.7	
Benin	108.0	33.3	112622.0	12000.0	36.22	229	Porto-Novo	6476.0	110.71	0.9	
Marshall Islands	329.0	63.9	181.0	31000.0	29.03	692	Majuro	143.0	125.08	2.5	

only showing top 20 rows

Correlation matrix

```
from pyspark.ml.stat import Correlation
from pyspark.ml.feature import VectorAssembler
import pyspark.sql.functions as F

# Select numerical columns for correlation
numeric_cols = ['Population', 'Co2-Emissions', 'Tax revenue (%)', 'Life expectancy',
                'Land Area(Km2)', 'Birth Rate', 'Fertility Rate', 'Infant mortality',
                'Maternal mortality ratio', 'Physicians per thousand']

# Assemble columns into a vector for correlation computation
assembler = VectorAssembler(inputCols=numeric_cols, outputCol="features")
vectorized_data = assembler.transform(df).select("features")

# Compute correlation matrix
correlation_matrix = Correlation.corr(vectorized_data, "features").head()

# Convert correlation matrix to a dense array
correlation_array = correlation_matrix[0].toArray()

# Convert to Pandas DataFrame for plotting
import pandas as pd

correlation_df = pd.DataFrame(correlation_array, index=numeric_cols, columns=numeric_cols)
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# Set up the figure size
plt.figure(figsize=(10, 8))

# Create a heatmap with annotations for better clarity
sns.heatmap(correlation_df, annot=True, cmap="coolwarm", linewidths=0.5)

# Set titles and labels
plt.title('Correlation Heatmap of Numerical Features', fontsize=15)
plt.show()
```


Correlation Heatmap of Numerical Features
Population
Co2-Emissions
Tax revenue (%)
Life expectancy
Land Area(Km2)
Birth Rate
Fertility Rate
Infant mortality
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50