SPRAWOZDANIE

Zajęcia: Eksploracja i wizualizacja danych
Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium 3
27.11.2021
Temat: "Użycie biblioteki „PySpark" dla dużych zbiorów danych.

Wariant: 5

Marek Bubiak
Informatyka II stopień,
niestacjonarne (zaoczne),
III semestr,
https://github.com/JeremyCoco/eiwd

Zadanie 1. Instalacja pakietu „pyspark”

Instalacja biblioteki w notatniku Google Colab

```
!pip install pyspark==3.0.1 py4j==0.10.9
```

Zadanie 2. Tworzenie sesji „SparkSession”

```
from pyspark.sql import SparkSession
spark = SparkSession.builder\
        .master("local[4]")\
        .appName('lab_4')\
        .getOrCreate()
```

Zadanie 3. Wczytać dane z pliku CSV i wyświetlić pierwsze 5 wierszy

```
csv_file = r"/content/IHME_GDP_1960_2050_Y2021M09D22.CSV"
data = spark.read.csv(csv_file, header=True)

data.show(5)
```

```
+-----------+-------------+----+------+----+----------------+----------------+----------------+-------------
|location_id|location_name|iso3| level|year|    gdp_ppp_mean|   gdp_ppp_lower|   gdp_ppp_upper|    gdp_usd_m
+-----------+-------------+----+------+----+----------------+----------------+----------------+-------------
|          1|       Global|   G|Global|1960|17483449774122.9|16019146112388.8|19115862416823.5|1296862531754
|          1|       Global|   G|Global|1961|18135370554950.5|16595371585758.2|19824927264221.5|1346097288345
|          1|       Global|   G|Global|1962|18953278607513.5|17390391432341.6|20614772322197.6|1406575798093
|          1|       Global|   G|Global|1963|19656620517295.9|18117057797516.5|21349934484879.7|1461831092087
|          1|       Global|   G|Global|1964|21005747228643.4|19356640986099.7|22767910934166.1|1552986205464
+-----------+-------------+----+------+----+----------------+----------------+----------------+-------------
only showing top 5 rows
```

Zadanie 4. Utworzyć schemat danych i zastosować go na zbiorze danych

Wyświetlanie schematu danych

```
data = spark.read.csv('/content/IHME_GDP_1960_2050_Y2021M09D22.CSV'
, sep=',', header=True)
data.printSchema()
```

```
root
 |-- location_id: string (nullable = true)
 |-- location_name: string (nullable = true)
 |-- iso3: string (nullable = true)
 |-- level: string (nullable = true)
 |-- year: string (nullable = true)
 |-- gdp_ppp_mean: string (nullable = true)
 |-- gdp_ppp_lower: string (nullable = true)
 |-- gdp_ppp_upper: string (nullable = true)
 |-- gdp_usd_mean: string (nullable = true)
 |-- gdp_usd_lower: string (nullable = true)
 |-- gdp_usd_upper: string (nullable = true)
```

```
from pyspark.sql.types import *
data_schema = [
    StructField('location_id', IntegerType(), True),
    StructField('location_name', StringType(), True),
    StructField('iso3', StringType(), True),
    StructField('level', StringType(), True),
    StructField('year', IntegerType(), True),
    StructField('gdp_ppp_mean', FloatType(), True),
    StructField('gdp_ppp_lower', FloatType(), True),
    StructField('gdp_ppp_upper', FloatType(), True),
    StructField('gdp_usd_mean', FloatType(), True),
    StructField('gdp_usd_lower', FloatType(), True),
    StructField('gdp_usd_upper', FloatType(), True),
    ]

final_struc = StructType(fields = data_schema)

data2 = spark.read.csv(csv_file, header=True, schema=final_struc)
data2.printSchema()
```

```
root
 |-- location_id: integer (nullable = true)
 |-- location_name: string (nullable = true)
 |-- iso3: string (nullable = true)
 |-- level: string (nullable = true)
 |-- year: integer (nullable = true)
 |-- gdp_ppp_mean: float (nullable = true)
 |-- gdp_ppp_lower: float (nullable = true)
 |-- gdp_ppp_upper: float (nullable = true)
 |-- gdp_usd_mean: float (nullable = true)
 |-- gdp_usd_lower: float (nullable = true)
 |-- gdp_usd_upper: float (nullable = true)
```

Zadanie 5. Wyświetlić typy danych w poszczególnych kolumnach

```
data.dtypes
```

```
[('location_id', 'string'),
 ('location_name', 'string'),
 ('iso3', 'string'),
 ('level', 'string'),
 ('year', 'string'),
 ('gdp_ppp_mean', 'string'),
 ('gdp_ppp_lower', 'string'),
 ('gdp_ppp_upper', 'string'),
 ('gdp_usd_mean', 'string'),
 ('gdp_usd_lower', 'string'),
 ('gdp_usd_upper', 'string')]
```

Zadanie 6. Pokazać 2 pierwsze i 2 ostatnie wiersze danych

```
data2.head(2)
```

```
[Row(location_id=1, location_name='Global', iso3='G', level='Global', year=1960, gdp
 Row(location_id=1, location_name='Global', iso3='G', level='Global', year=1961, gdp
```

```
data2.tail(2)
```

```
[Row(location_id=44578, location_name='Low income', iso3=None, level='World Bank Inc
 Row(location_id=44578, location_name='Low income', iso3=None, level='World Bank Inc
```

Zadanie 7. Dodać nową kolumnę zawierającą zera, zmienić jej nazwę a następnie usunąć

```
res = data2.withColumn('Nowa kolumna', data2.year*0 + 1000)
```

```
res = res.withColumnRenamed ('Nowa kolumna', 'col')
```

```
res = data2.drop ('col')
```

Zadanie 8. Dodać do danych kolumnę zawierającą numer wiersza

```
from pyspark.sql.functions import udf


i = -1
def incr ():
    global i
    i = i+1
    return i

newCol = udf(incr , IntegerType ())

# dodanie nowej kolumny
data3 = data2.withColumn ('id', newCol ())

data3.show (5)
```

```
---------+-------------+-------------+-------------+-------------+-------------+---+
_ppp_mean|gdp_ppp_lower|gdp_ppp_upper| gdp_usd_mean|gdp_usd_lower|gdp_usd_upper| id|
---------+-------------+-------------+-------------+-------------+-------------+---+
834498E13|1.60191459E13|1.91158634E13|1.29686254E13|1.26689036E13|1.33417654E13|  0|
353715E13| 1.6595372E13|1.98249273E13|1.34609728E13|1.31476656E13|1.38302141E13|  1|
532796E13|1.73903918E13|2.06147714E13|1.40657578E13| 1.3760596E13|1.44374581E13|  2|
566204E13|1.81170571E13|2.13499343E13|1.46183113E13| 1.4321321E13|1.49769274E13|  3|
057476E13|1.93566417E13| 2.2767911E13|1.55298625E13| 1.5234982E13|1.58799799E13|  4|
---------+-------------+-------------+-------------+-------------+-------------+---+
```

Zadanie 9. Sprawdzić liczbę wierszy danych przed i po usunięciu wierszy bez danych dwoma sposobami.

```
# poczatkowa liczba rekordow
data3.count()
```

```
19838
```

```
# usuniecie wierszy bez danych ( sposob 1.)
data4 = data3.na.drop()
```

```
data4.count()
```

```
18655
```

```
# wstawienie zera w miejsce braku danych ( sposob 2.)
data5 = data3.na.fill(data3.select(0 * data3.year).collect()[0][0])

data5.count()
```

```
19838
```

Zadanie 10. Wyświetlić wybrane kolumny danych

```
data5.select(['year', 'location_id', 'location_name']).show(5)
```

```
+----+-----------+-------------+
|year|location_id|location_name|
+----+-----------+-------------+
|1960|          1|       Global|
|1961|          1|       Global|
|1962|          1|       Global|
|1963|          1|       Global|
|1964|          1|       Global|
+----+-----------+-------------+
only showing top 5 rows
```

Zadanie 11. Odfiltrować dane zawierające dane od roku 2000

```
from pyspark.sql.functions import col

data5 .filter(( col('year') >= 2000) & (col('location_id') > 20)).select (['year',
'location_name', 'location_id']).show (5)
```

```
+----+-------------+-----------+
|year|location_name|location_id|
+----+-------------+-----------+
|2000|         Fiji|         22|
|2001|         Fiji|         22|
|2002|         Fiji|         22|
|2003|         Fiji|         22|
|2004|         Fiji|         22|
+----+-------------+-----------+
only showing top 5 rows
```

Zadanie 12. Dodać kolumnę zawierającą wynik sprawdzania warunku

```
# dodanie kolumny zawierajacej wynik sprawdzenia ,
#czy rok jest wiekszy niz 2000
from pyspark . sql import functions as f

data5.select('year', 'location_id', 'location_name'
, f.when(data5.year > 2000, '21st century').otherwise ('20 th century')
.alias ('century')).show (5)

# dodanie kolumny zawierajacej wynik sprawdzenia ,
#czy nazwa kraju zaczyna sie od litery 'A'
data5.select('year', 'location_id', 'location_name',
            data5.location_name.rlike('^F').alias('Acountry')). show (5)
```

```
+----+-----------+-------------+-------------+
|year|location_id|location_name|      century|
+----+-----------+-------------+-------------+
|1960|          1|       Global|20 th century|
|1961|          1|       Global|20 th century|
|1962|          1|       Global|20 th century|
|1963|          1|       Global|20 th century|
|1964|          1|       Global|20 th century|
+----+-----------+-------------+-------------+
only showing top 5 rows

+----+-----------+-------------+--------+
|year|location_id|location_name|Acountry|
+----+-----------+-------------+--------+
|1960|          1|       Global|   false|
|1961|          1|       Global|   false|
|1962|          1|       Global|   false|
|1963|          1|       Global|   false|
|1964|          1|       Global|   false|
+----+-----------+-------------+--------+
only showing top 5 rows
```

Zadanie 13. Pogrupować dane wg kraju i obliczyć liczbę danych, średnią wartość oraz wartości minimalne i maksymalne

```
# pogrupowanie danych wg kraju
from pyspark.sql.functions import mean, count, min, max


data5.select(['year', 'location_id', 'location_name']).groupBy('location_
      .agg(
          count('year').alias('number od countries'),
          mean('location_id').alias('number'),
          min('year').alias('min year'),
          max('year').alias('max year')
      ).show(5)
```

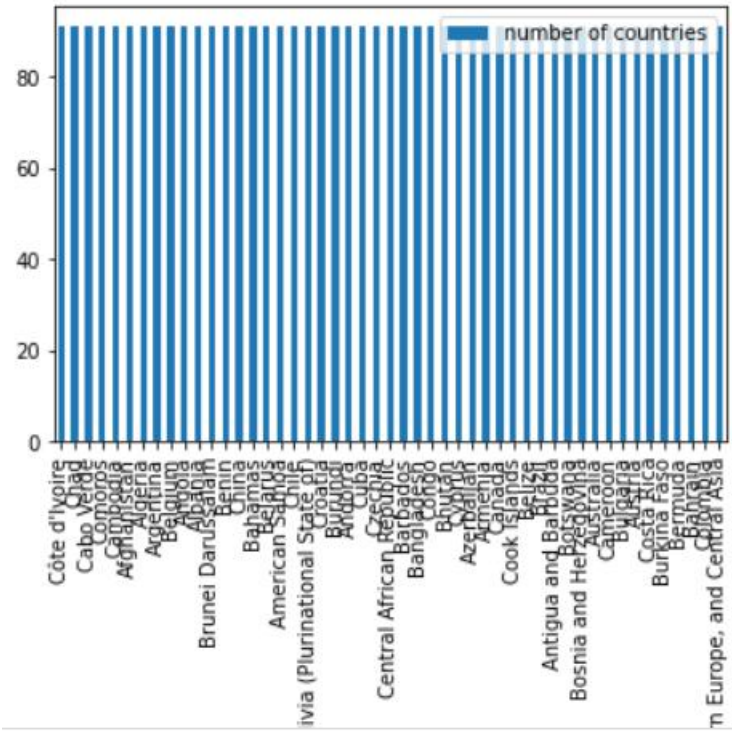
```
+--------------------+-------------------+------+--------+--------+
|       location_name|number od countries|number|min year|max year|
+--------------------+-------------------+------+--------+--------+
|          South Asia|                182| 158.5|    1960|    2050|
|       Côte d'Ivoire|                 91| 205.0|    1960|    2050|
|Micronesia (Feder...|                 91|  25.0|    1960|    2050|
|                Chad|                 91| 204.0|    1960|    2050|
|            Paraguay|                 91| 136.0|    1960|    2050|
+--------------------+-------------------+------+--------+--------+
only showing top 5 rows
```

Zadanie 14. Wygenerować wykres słupkowy na podstawie pogrupowanych danych.

```
from matplotlib import pyplot as plt

res = data5.filter(data5.location_name.rlike('^[ABC]'))\
.select(['year', 'location_id', 'location_name']).groupBy('location_name')\
      .agg(
          count('year').alias('number of countries'),
          mean('location_id').alias('number'),
          mean('year').alias('mean year'),
          max('year').alias('max year')
      ).toPandas()

res.plot(kind='bar', x='location_name', y='number of countries')
plt.show()
```


number of countries
80
60
40
20
0

Wnioski

1. Biblioteka „pyspark” jest wygodną alternatywą dla pakietu „pandas”, zwłaszcza jeśli analizowane są duże zbiory danych. Wszelkie operacje na danych wymagają utworzenia sesji „SparkSession” przy użyciu biblioteki builder.

2. Biblioteka „pyspark” pozwala wczytywać dane z różnych źródeł danych:

    * pliki csv

    * pliki json

    * pliki parquet

3. Biblioteka „pyspark” pozwala wczytywać dane z różnych źródeł danych:

```
# pliki CSV
spark.write.csv(csv_file_path, sep=',', header=True)

# pliki JSON i Parquet
data = spark.write.save(json_data_file)
data = spark.write.save(parquet_data_file)
```

4. Wczytane dane są zwykle w formacie „string”, ale można narzucić im typ przy użyciu schematu.

5. Wybieranie wierszy odbywa się analogicznie jak w bibliotece „pandas”.

6. Możliwe jest dodanie kolumny, zmiana jej nazwy oraz usunięcie.

7. Wartości w nowych kolumnach można też dodawać przy użyciu osobnej funkcji.

8. Wiersze niezawierające danych mogą być łatwo usunięte.