https://tinyurl.com/s9jbbhf

1) Proszę wczytać dane lotnisk z całego świata korzystając z pliku dostępnego tutaj: https://www.kaggle.com/jonatancr/airports

Link bezpośredni do pliku csv: https://drive.google.com/open?id=1gISWlbZ19UOXwJioUUhLHAnGv0WqHbrm

Ponieważ w zbiorze tym nie ma nagłówka, nadaj własne nazwy kolumnom, korzystając z opis pliku na stronie https://openflights.org/data.html. Najprościej to zrobić korzystając z metody toDF("nazwa1", "nazwa2, ...), która zwraca nową ramkę ze zmienionymi nazwami kolumn.

Rozwiązanie:

```
airports = spark.read.csv("airports.csv",inferSchema=True,header=False).\
toDF("id","airport","city","country","iata","icao","latitude","longitude","altitude","timezone","dst","tz_timezone","type","data_source")
```

2) Przygotowanie/wyczyszczenie zbioru “countries of the world”

- Przycięcie stringów - usunięcie białych znaków z przodu i tyłu
- Konwersja wartości zmiennoprzecinkowych (zamiana przecinka na kropkę)
- Usunięcie kropek z nazw kolumn
- Wypełnienie pustych komórek wartościami

Potrzebne importy:

```
from pyspark.sql.functions import col, udf, trim, isnull

from pyspark.sql.types import FloatType, IntegerType
```

Usuwanie kropek z nazw kolumn:

```
countries = spark.read.csv("countries of the world.csv", inferSchema=True, header=True)

# Usuwa kropki z wszystkich nazw kolumn (powodują wiele błędów, prawdopodobnie bug w Sparku)

# (Nb. wszystkie inne metody zmiany tych nazw (np. użycie columnRenamed) zawiodły)
```

```
new_columns=[s.replace('.','') for s in countries.columns]

countries=countries.toDF(*new_columns) # tutaj '*' to operator "splat" -- robi z tablicy listę argumentów



# Utworzenie ramki 'cdf' -- konwersja wszystkich kolumn zmiennoprzecinkowych na float + trim stringów

cdf=countries.na.fill("-1").select( # wartosci puste wypelnione "-1", zeby nie usuwac wierszy

             trim(col('Country')).alias('Country'),\

             trim(col('Region')).alias('Region'),'Population',\

             col('Area (sq mi)').alias('Area (sq km)'),\

             float_udf('Pop Density (per sq mi)').alias('Pop Density (per sq km)'),\

             float_udf('Coastline (coast/area ratio)').alias('Coastline (coast/area ratio)'),\

             float_udf('Net migration').alias('Net migration'),\

             float_udf('Infant mortality (per 1000 births)').alias('Infant mortality (per 1000 births)'),\

             'GDP ($ per capita)',\

             float_udf('Literacy (%)').alias('Literacy (%)'),\

             float_udf('Phones (per 1000)').alias('Phones (per 1000)'),\

             float_udf('Arable (%)').alias('Arable (%)'),\

             float_udf('Crops (%)').alias('Crops (%)'),\

             float_udf('Other (%)').alias('Other (%)'),\

             float_udf('Birthrate').alias('Birthrate'),\

             float_udf('Deathrate').alias('Deathrate'),\
```

```
            float_udf('Agriculture').alias('Agriculture'),\
            float_udf('Industry').alias('Industry'),\
            float_udf('Service').alias('Service'))
```

3) Zapytanie 1: lotniska na północ od Krakowa

```
%%time # prints execution time
krakow_df = airports.select('latitude').\
     where(col('City').like('%Krakow%'))
# display: wymuszenie wypisania
display(krakow_df)
krk_lat = krakow_df.collect()[0][0]
display(krk_lat)
display(airports.select('City', 'Airport',
'latitude').where(col('latitude')>krk_lat).\
     sort(col("latitude").desc()).toPandas())
```

4) Zapytanie 2:

Etap 1: grupowanie i wybieranie lotnisk wysuniętych najdalej na południe

```
from pyspark.sql.functions import min
south_ports=airports.select('Country','Airport','latitude').\
     groupBy('Country').agg(min("latitude").alias('latitude'))
south_ports.toPandas()
```

Etap 2: łączenie z ramką airports, żeby dodac kolumnę Airport:

```
# Why we need to join results with the original 'airports' frame?
#
https://stackoverflow.com/questions/34409875/how-to-get-other-colu
mns-when-using-spark-dataframe-groupby
a=south_ports.alias('a')
b=airports.alias('b')
a.join(b, col('a.latitude')==col('b.latitude')).\
select('a.Country','b.Airport',col('a.latitude').alias('Latitude')
).\
sort(col("a.Country").asc()).toPandas()
```

5) Zapytanie 3: rysunek

```
arp=airports.select('latitude','longitude')
arp.toPandas().plot(x="longitude",y="latitude",kind="scatter",figs
ize=(10, 6))
```