SPARK TP2 Soufiane MOUTEI

October 19, 2018

1 Let's start with initializing Spark:

```
In [1]: from pyspark import SparkContext
        from pyspark.sql import SQLContext
        from pyspark.sql.types import *
        from pyspark.sql.functions import udf, first

        sc = SparkContext()
        sqlContext = SQLContext(sc)
```

```
In [2]: sc
```

```
Out[2]: <SparkContext master=local[*] appName=pyspark-shell>
```

2 Average monthly income of the shop in France:

```
In [3]: # A function to get: store, month, income; it will be used for mapping
        def f(x):
            key = x[0]
            line = x[1].split()
            return key, line[0], int(line[1])
```

```
In [4]: # Let's start with preparing the files
        whole_file = (
            # read the text files of the directory "input1"
            sc.wholeTextFiles("input1/")

            # Only the key is changed here: we split on "/" to get the textfile name
            # and then delete the file extension
            .map(lambda x: (x[0].split("/")[-1][:-4], x[1]))

            # Split the values to get each line on the file linked to the city name
            .flatMapValues(lambda x: x.split("\r\n"))

            # Map the function f to get: city, month, income.
            .map(f)
        )

        # The schema that will be used for 'whole_file'
        mySchema = StructType([
            StructField("Store", StringType(), True),
            StructField("Month", StringType(), True),
            StructField("Income", IntegerType(), True)
        ])

        # Create a DataFrame using 'whole_file' and the schema 'mySchema'
        df = sqlContext.createDataFrame(whole_file, schema=mySchema)

        # Add the city name as a column to 'df'
        getCity = udf(lambda t: t.split("_")[0])
        df = df.withColumn("City", getCity(df["Store"]))
```

```
In [5]: # Answer to the question
        monthly_income_france = (
            df

            # To select as columns: Month, Income
            .select("Month", "Income")

            # Group by 'Month' and take the average of 'Income' for each month
            .groupBy("Month")
            .avg()

            # Rename the column representing the average of 'Income' for each month
            .withColumnRenamed('avg(Income)', 'Average monthly income')
        )
        monthly_income_france.show()
```

```
+-----+----------------------+
|Month|Average monthly income|
+-----+----------------------+
|  APR|     20.23076923076923|
|  OCT|     26.53846153846154|
|  NOV|     24.53846153846154|
|  FEB|    19.153846153846153|
|  SEP|     25.53846153846154|
|  JAN|     20.76923076923077|
|  AUG|    23.076923076923077|
|  MAR|     17.53846153846154|
|  DEC|                  29.0|
|  JUN|    27.846153846153847|
|  JUL|    21.692307692307693|
|  MAY|     22.46153846153846|
+-----+----------------------+
```

3 Average monthly income of the shop in each city:

```
In [6]: monthly_income_per_city = (
            df

            # To select as columns: City, Month, Income
            .select("City", "Month", "Income")

            # Group by 'City' and 'Month' and take the average of 'Income' for each group
            .groupBy("City", "Month")
            .avg()

            # Rename the column representing the average of 'Income'
            .withColumnRenamed('avg(Income)', 'Average monthly income')

            # Sort by the city to get the values of each city together
            .orderBy('City', ascending=True)
        )
        monthly_income_per_city.show(200)
```

```
+----------+-----+----------------------+
|      City|Month|Average monthly income|
+----------+-----+----------------------+
|     anger|  NOV|                  14.0|
|     anger|  AUG|                  15.0|
|     anger|  JUL|                  19.0|
|     anger|  JUN|                  15.0|
|     anger|  JAN|                  13.0|
|     anger|  DEC|                  16.0|
|     anger|  OCT|                   8.0|
|     anger|  FEB|                  12.0|
|     anger|  SEP|                  13.0|
|     anger|  MAR|                  14.0|
|     anger|  APR|                  15.0|
|     anger|  MAY|                  12.0|
|      lyon|  MAR|                  14.0|
|      lyon|  AUG|                  25.0|
|      lyon|  APR|                  15.0|
|      lyon|  OCT|                  11.0|
|      lyon|  JUN|                  15.0|
|      lyon|  MAY|                  12.0|
|      lyon|  JUL|                  19.0|
|      lyon|  NOV|                  22.0|
|      lyon|  SEP|                  13.0|
|      lyon|  FEB|                   12.0|
|      lyon|  DEC|                   22.0|
|      lyon|  JAN|                   13.0|
|marseilles|  SEP|                   23.0|
|marseilles|  DEC|                   26.0|
|marseilles|  NOV|                   24.0|
|marseilles|  APR|                   22.0|
|marseilles|  JUN|                   25.0|
|marseilles|  OCT|                   28.0|
|marseilles|  JAN|                   16.0|
|marseilles|  AUG|                   22.0|
|marseilles|  MAY|                   18.5|
|marseilles|  JUL|                   21.0|
|marseilles|  MAR|                   16.0|
|marseilles|  FEB|                   16.0|
|    nantes|  APR|                   12.0|
|    nantes|  AUG|                   11.0|
|    nantes|  MAY|                   21.0|
|    nantes|  FEB|                   15.0|
|    nantes|  NOV|                   14.0|
|    nantes|  OCT|                   14.0|
|    nantes|  JUN|                   28.0|
|    nantes|  DEC|                   24.0|
|    nantes|  JAN|                   16.0|
|    nantes|  SEP|                   13.0|
|    nantes|  MAR|                   20.0|
|    nantes|  JUL|                   19.0|
|      nice|  JUL|                   19.0|
|      nice|  MAY|                   11.0|
|      nice|  MAR|                   20.0|
|      nice|  JAN|                   16.0|
|      nice|  FEB|                   15.0|
|      nice|  OCT|                   18.0|
|      nice|  JUN|                   18.0|
|      nice|  AUG|                   11.0|
|      nice|  DEC|                   29.0|
|      nice|  NOV|                   14.0|
|      nice|  APR|                    9.0|
|      nice|  SEP|                   23.0|
|    orlean|  OCT|                    8.0|
|    orlean|  MAR|                   14.0|
|    orlean|  APR|                   15.0|
|    orlean|  NOV|                   24.0|
|    orlean|  FEB|                   12.0|
|    orlean|  SEP|                   13.0|
|    orlean|  JUL|                   19.0|
|    orlean|  JUN|                   15.0|
|    orlean|  AUG|                   25.0|
|    orlean|  MAY|              12.0|
|    orlean|  JAN|              13.0|
|    orlean|  DEC|              26.0|
|     paris|  JUL| 33.666666666666664|
|     paris|  OCT| 56.666666666666664|
|     paris|  JAN| 38.333333333333336|
|     paris|  FEB|              33.0|
|     paris|  SEP|              48.0|
|     paris|  MAR| 26.333333333333332|
|     paris|  MAY|              50.0|
|     paris|  NOV| 48.666666666666664|
|     paris|  DEC| 52.666666666666664|
|     paris|  JUN|              55.0|
|     paris|  AUG| 41.666666666666664|
|     paris|  APR| 38.666666666666664|
|    rennes|  AUG|              11.0|
|    rennes|  APR|               9.0|
|    rennes|  JUN|              13.0|
|    rennes|  FEB|              18.0|
|    rennes|  SEP|              23.0|
|    rennes|  DEC|              20.0|
|    rennes|  JUL|              14.0|
|    rennes|  OCT|              18.0|
|    rennes|  MAR|              10.0|
|    rennes|  JAN|              19.0|
|    rennes|  NOV|              14.0|
|    rennes|  MAY|              11.0|
|  toulouse|  APR|              11.0|
|  toulouse|  SEP|              23.0|
|  toulouse|  DEC|              19.0|
|  toulouse|  JAN|              12.0|
|  toulouse|  MAY|              11.0|
|  toulouse|  NOV|              12.0|
|  toulouse|  MAR|              14.0|
|  toulouse|  AUG|              11.0|
|  toulouse|  FEB|              13.0|
|  toulouse|  JUN|              18.0|
|  toulouse|  OCT|              14.0|
|  toulouse|  JUL|              19.0|
|    troyes|  JUN|              25.0|
|    troyes|  JUL|              11.0|
|    troyes|  DEC|              11.0|
|    troyes|  AUG|              22.0|
|    troyes|  FEB|              21.0|
|    troyes|  APR|              17.0|
|    troyes|  SEP|              21.0|
|    troyes|  MAR|              11.0|
|    troyes|  NOV|              11.0|
|    troyes|  OCT|                    28.0|
|    troyes|  JAN|                    21.0|
|    troyes|  MAY|                    15.0|
+----------+-----+----------------------+
```

4 Total revenue per city per year:

```
In [7]: yearly_income_city = (
            df

            # To select as columns: City, Income
            .select("City", "Income")

            # Group by 'City' and take the sum of 'Income' for each city
            .groupBy("City")
            .sum()

            # Rename the column representing the sum of 'Income'
            .withColumnRenamed('sum(Income)', 'Total revenue per year')
        )
        yearly_income_city.show(20)
```

```
+----------+----------------------+
|      City|Total revenue per year|
+----------+----------------------+
|    nantes|                   207|
|    troyes|                   214|
|     paris|                  1568|
|      lyon|                   193|
|     anger|                   166|
|marseilles|                   515|
|      nice|                   203|
|    orlean|                   196|
|    rennes|                   180|
|  toulouse|                   177|
+----------+----------------------+
```

5 Total revenue per store per year:

```
In [8]: yearly_income_store = (
            df

            # To select as columns: Store, Income
            .select("Store", "Income")

            # Group by 'Store' and take the sum of 'Income' for each store
            .groupBy("Store")
            .sum()

            # Rename the column representing the sum of 'Income'
            .withColumnRenamed('sum(Income)', 'Total revenue per year')
        )
        yearly_income_store.show(20)
```

```
+------------+----------------------+
|       Store|Total revenue per year|
+------------+----------------------+
|      nantes|                   207|
|      troyes|                   214|
|        lyon|                   193|
|marseilles_1|                   284|
|     paris_2|                   642|
|       anger|                   166|
|     paris_3|                   330|
|marseilles_2|                   231|
|        nice|                   203|
|      orlean|                   196|
|      rennes|                   180|
|     paris_1|                   596|
|    toulouse|                   177|
+------------+----------------------+
```

6 The store that achieves the best performance in each month:

```
In [9]: # Method 1: Using sorting
        best_performance_store = (

            # To select as columns: Month, Store, Income
            df.select("Month", "Store", "Income")

            # Sort by (month, -income) to get the values sorted by the month,
            # and in case we have the same month, we sort in a descending order the income
            .orderBy('Month', - df['Income'], ascending=True)

            # group by the month to get the best store for each month (that is the first store s
            .groupby("Month")
            .agg(first("Store").alias("Best Store"))
        )
        best_performance_store.show(20)
```

```
+-----+----------+
|Month|Best Store|
+-----+----------+
|  APR|   paris_1|
|  OCT|   paris_1|
|  NOV|   paris_2|
|  FEB|   paris_2|
|  SEP|   paris_2|
|  JAN|   paris_1|
|  AUG|   paris_2|
|  MAR|   paris_2|
|  DEC|   paris_1|
|  JUN|   paris_2|
|  JUL|   paris_1|
|  MAY|   paris_2|
+-----+----------+
```

```
In [10]: # Method 2: Without sorting, using the maximum function
         best_performance_store_2 = (
             df

             # To select as columns: Month, Income
             .select("Month", "Income")

             # group by the month to get the maximum income for each month
             .groupby("Month")
             .max("Income")

             # Join the initial data (selecting: Month, Store, Income) on the month
             .join(df.select("Month", "Store", "Income"), "Month", how='outer')
         )

         best_performance_store_2 = (
             best_performance_store_2

             # Select only where the income is equal to the maximum, which is the best value
             .filter(best_performance_store_2["Income"] == best_performance_store_2["max(Income)

             # Drop the maximum column and the income column because we need only the month and
             .drop("max(Income)", "Income")
         )
         best_performance_store_2.show(20)
```

```
+-----+-------+
|Month|  Store|
+-----+-------+
|  APR|paris_1|
|  MAY|paris_2|
|  SEP|paris_2|
|  JUN|paris_2|
|  JAN|paris_1|
|  OCT|paris_1|
|  NOV|paris_2|
|  FEB|paris_2|
|  MAR|paris_2|
|  AUG|paris_2|
|  JUL|paris_1|
|  DEC|paris_1|
+-----+-------+
```