Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)

Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e SparkSQL

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para efetuar uma análise de dados. Esta empresa possui acesso a uma base de dados com dados sobre as transações comerciais entre países nos últimos 30 anos. Sendo que, para cada transação comercial presente nesta base de dados os seguintes campos são fornecidos:

Campo	Descrição
País	País envolvido na transação comercial
Ano	Ano em que a transação foi efetuada
Código	Código da mercadoria
Mercadoria	Descrição da mercadoria
Fluxo	Fluxo, e.g. *Exportação* ou *Importação*
Valor	Valor em dólares
Peso	Peso da mercadoria
Unidade	Unidade de medida da mercadoria, e.g. *Quantidade de itens*
Quantidade	Quantidade conforme a unidade especificada da mercadoria
Categoria	Categoria da mercadoria, e.g. *Produto Animal*

No total, a base de dados possui mais de 8 milhões de transações comerciais. A base de dados foi fornecida no formato CSV, sendo que cada entrada (transação comercial), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ";". A imagem a seguir exibe as 5 primeiras transações comerciais presentes na base.

```
Afghanistan;2016;010410;Sheep, live;Export;6088;2339;Number of items;51;01_live_animals
Afghanistan;2016;010420;Goats, live;Export;3958;984;Number of items;53;01_live_animals
Afghanistan;2008;010210;Bovine animals, live pure-bred breeding;Import;1026804;272;Number of items;3769;01_live_animals
Albania;2016;010290;Bovine animals, live, except pure-bred breeding;Import;2414533;1114023;Number of items;6853;01_live_animals
Albania;2016;010392;Swine, live except pure-bred breeding > 50 kg;Import;14265937;9484953;Number of items;96040;01_live_animals
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de um conjunto de soluções em Apache Spark e Apache SparkSQL que permitam a extração das seguintes informações sobre a base:

1. País com a maior quantidade de transações comerciais efetuadas;
2. Mercadoria com a maior quantidade de transações comerciais no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
3. Quantidade de transações financeiras realizadas por ano;
4. Mercadoria com maior quantidade de transações financeiras;
5. Mercadoria com maior quantidade de transações financeiras em 2016;
6. Mercadoria com maior quantidade de transações financeiras em 2016, no Brasil (como a base de dados está em inglês utilize Brazil, com Z);
7. Mercadoria com maior total de peso, de acordo com todas transações comerciais;
8. Mercadoria com maior total de peso, de acordo com todas transações comerciais, separadas de acordo com o ano;
9. Média de peso por mercadoria, separadas de acordo com o ano;
10. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
11. Média de peso por mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), em relação ao fluxo, separadas de acordo com o ano;
12. Média de valor por peso, de acordo com a mercadoria comercializadas no Brasil (como a base de dados está em inglês utilize Brazil, com Z), separadas de acordo com o ano;
13. Valor máximo de código;
14. Mercadoria com o maior preço por unidade de peso;
15. Quantidade de transações comerciais de acordo com o fluxo, de acordo com o ano;

Com base no seu conhecimento em Apache Spark e Apache Spark SQL, para cada uma das soluções requisitadas, forneça:

1. **O resultado da execução (forneça apenas os 5 primeiros resultados)**
2. **Tempo de execução, completando a tabela abaixo**

#	Tempo Apache Spark	Tempo Apache Spark SQL
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Dicas

- *Localização dos arquivos*

Máquina	*Base parcial*	*Base completa*
NameNode	*/data/operacoes_comerciais/base_100_mil.csv*	*/data/operacoes_comerciais/base_inteira.csv*
Host	*/home/docker/Desktop/data/operacoes_comerciais/base_100_mil.csv*	*/home/docker/Desktop/data/operacoes_comerciais/base_inteira.csv*

Código

Lembre-se de atualizar os IPs e caminhos dos arquivos!

Iniciar sessão no Apache Spark SQL

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession

sc = SparkSession \
    .builder \
    .master('spark://172.18.0.2:7077') \
    .config('spark.executor.memory', '1g') \
    .getOrCreate()
```

Carregar arquivo em DataFrame

```
df = sc.read \
    .option('delimiter', ';') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/base_inteira.csv')
```

Iniciar sessão no Apache Spark

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

import pyspark
conf = pyspark.SparkConf()

conf.setMaster("spark://172.24.0.10:7077")
conf.set("spark.executor.memory", "1g")

sc = pyspark.SparkContext.getOrCreate()
sc.stop()
sc = pyspark.SparkContext(conf=conf)
```

Iniciar sessão no Apache Spark

```
arquivo = sc.textFile('hdfs://172.24.0.12:9000/base.csv')
```

Consulta Apache SQL (crie a tabela apenas uma vez)

```
df.createOrReplaceTempView('comercio')
```

```
sql = sc.sql('select * from comercio limit 5')
sql.show()
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)