درس : Big Data
استاد : خانم دکتر صفری
دانشجو : رقیه یزدانی
تمرین : Spark DataFrame

Spark DataFrame

ایجاد بستر لازم برای امکان کار با Spark

مشابه تمرین قبلی spark، cloudera vm را start کرده و سرویس‌های لازم را بالا می‌آوریم. ابتدا یک ترمینال باز کرده و دستور spark-shell را اجرا می‌کنیم، می‌توانیم version مربوط به spark و همینطور scala را ببینیم:

Spark 1.6.0
Scala 2.10.5

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.6.0
      /_/

Using Scala version 2.10.5 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_67)
Type in expressions to have them evaluated.
Type :help for more information.
Spark context available as sc (master = yarn-client, app id = application_1594357675283_0001).
20/07/09 22:18:54 WARN metastore.ObjectStore: Version information not found in metastore. hive.metastore.schema.verification
the schema version 1.1.0-cdh5.13.0
20/07/09 22:18:54 WARN metastore.ObjectStore: Failed to get database default, returning NoSuchObjectException
SQL context available as sqlContext.

scala>
```

حال در سایت https://spark-packages.org نسخه پکیج spark-csv را که متناسب با نسخه scala بدست آمده در بالا باشد، انتخاب می‌کنیم. برای کار با csv لازم است که هنگام اجرای pyspark، پکیج مربوط به csv را هم import کنیم. برای اینکار دستورات زیر را اجرا می‌کنیم. در دستور دوم، پس از مشخص کردن نام package، عدد اول (2.10) ورژن scala و عدد دوم (1.5.0) ورژن spark-csv می‌باشد.

```
$ PYSPARK_DRIVER_PYTHON=ipython
$ pyspark –packages com.databricks:spark-csv_2.10:1.5.0
```

پس از اجرای دستورات بالا خطایی به شکل زیر تولید شد :

```
:::: ERRORS
        Server access error at url https://repo1.maven.org/maven2/com/databricks/spark-csv_2.10/1.2.0/spark-csv_2.10-1.2.0.pom (javax.net.ssl.SSLExceptio
n: Received fatal alert: protocol_version)

        Server access error at url https://repo1.maven.org/maven2/com/databricks/spark-csv_2.10/1.2.0/spark-csv_2.10-1.2.0.jar (javax.net.ssl.SSLExceptio
n: Received fatal alert: protocol_version)


:: USE VERBOSE OR DEBUG MESSAGE LEVEL FOR MORE DETAILS
Exception in thread "main" java.lang.RuntimeException: [unresolved dependency: com.databricks#spark-csv_2.10;1.2.0: not found]
        at org.apache.spark.deploy.SparkSubmitUtils$.resolveMavenCoordinates(SparkSubmit.scala:1067)
        at org.apache.spark.deploy.SparkSubmit$.prepareSubmitEnvironment(SparkSubmit.scala:287)
        at org.apache.spark.deploy.SparkSubmit$.submit(SparkSubmit.scala:154)
        at org.apache.spark.deploy.SparkSubmit$.main(SparkSubmit.scala:121)
        at org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)
```

برای رفع این مشکل که به نظر می‌رسد مربوط به نسخه TLS در پروتکل SSL می باشد، پس از جستجو در نت، بصورت زیر عمل شد. ابتدا متغیر زیر به فایل bashrc. اضافه شد و پس از لود آن، مجددا دستور بالا اجرا شد که این بار مشکلی نداشت.

export JAVA_TOOL_OPTIONS="-Dhttps.protocols=TLSv1.2"

s

```
[cloudera@quickstart ~]$ sudo nano .bashrc
[cloudera@quickstart ~]$ cat .bashrc
# .bashrc

# User specific aliases and functions

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

export JAVA_TOOL_OPTIONS="-Dhttps.protocols=TLSv1.2"

[cloudera@quickstart ~]$ source .bashrc
[cloudera@quickstart ~]$ pyspark --packages com.databricks:spark-csv_2.10:1.5.0
Picked up JAVA_TOOL_OPTIONS: -Dhttps.protocols=TLSv1.2
Python 2.6.6 (r266:84292, Jul 23 2015, 15:22:56)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-11)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Picked up JAVA_TOOL_OPTIONS: -Dhttps.protocols=TLSv1.2
Picked up JAVA_TOOL_OPTIONS: -Dhttps.protocols=TLSv1.2
Ivy Default Cache set to: /home/cloudera/.ivy2/cache
The jars for the packages stored in: /home/cloudera/.ivy2/jars
:: loading settings :: url = jar:file:/usr/lib/spark/lib/spark-assembly-1.6.0-cdh5.13.0-hadoop2.6.0-cdh5.13.0.jar!/
gs.xml
com.databricks#spark-csv_2.10 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent;1.0
        confs: [default]
        found com.databricks#spark-csv_2.10;1.5.0 in central
        found org.apache.commons#commons-csv;1.1 in central
        found com.univocity#univocity-parsers;1.5.1 in central
downloading https://repo1.maven.org/maven2/com/databricks/spark-csv_2.10/1.5.0/spark-csv_2.10-1.5.0.jar ...
        [SUCCESSFUL ] com.databricks#spark-csv_2.10;1.5.0!spark-csv_2.10.jar (393ms)
downloading https://repo1.maven.org/maven2/org/apache/commons/commons-csv/1.1/commons-csv-1.1.jar ...
        [SUCCESSFUL ] org.apache.commons#commons-csv;1.1!commons-csv.jar (153ms)
downloading https://repo1.maven.org/maven2/com/univocity/univocity-parsers/1.5.1/univocity-parsers-1.5.1.jar ...
        [SUCCESSFUL ] com.univocity#univocity-parsers;1.5.1!univocity-parsers.jar (350ms)
:: resolution report :: resolve 5306ms :: artifacts dl 902ms
        :: modules in use:
        com.databricks#spark-csv_2.10;1.5.0 from central in [default]
        com.univocity#univocity-parsers;1.5.1 from central in [default]
        org.apache.commons#commons-csv;1.1 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
```

حال می‌توانیم فایل csv را لود کنیم، پس از لود کردن فایل، با استفاده از فراخوانی تابع printSchema ساختار داده اطلاعات لود شده، نمایش داده می شود که شامل نلم ستون؛ نوع داده آن و nullable بودن یا نبودن می‌باشد.

```
>>> yelp_df = sqlCtx.load(source='com.databricks.spark.csv', header='true', inferSchema='true', path ='file:///usr/lib/hue/apps/search/examples/collectio
ns/solr_configs_yelp_demo/index_data.csv')
>>>
>>> yelp_df.printSchema()
root
 |-- business_id: string (nullable = true)
 |-- cool: integer (nullable = true)
 |-- date: string (nullable = true)
 |-- full_address: string (nullable = true)
 |-- funny: integer (nullable = true)
 |-- id: string (nullable = true)
 |-- latitude: double (nullable = true)
 |-- longitude: double (nullable = true)
 |-- name: string (nullable = true)
 |-- neighborhoods: string (nullable = true)
 |-- open: boolean (nullable = true)
 |-- review_count: integer (nullable = true)
 |-- stars: integer (nullable = true)
 |-- state: string (nullable = true)
 |-- text: string (nullable = true)
 |-- type: string (nullable = true)
 |-- useful: integer (nullable = true)
 |-- user_id: string (nullable = true)

>>>
```

سوال 1 : برای یافتن میانگین ستون cool بصورت زیر عمل می‌کنیم. با توجه به نتیجه؛ گزینه 3 صحیح است.

```
>>> yelp_df.select("cool").agg({"cool":"mean"}).collect()
[Row(avg(cool)=0.998)]
>>>
```

سوال 2 : برای این سوال، ابتدا با تابع filter، سطرهای دارای شرط review_count>=10 را جدا کرده و سپس روی فیلد stars گروه بندی کرده و برای هر گروه میانگین cool را محاسبه می‌کنیم. با توجه به نتیجه؛ گزینه 3 صحیح است.

```
>>> yelp_df.filter("review_count>=10").groupBy("stars").mean("cool").collect()
[Row(stars=2, avg(cool)=0.52173913043478259), Row(stars=3, avg(cool)=1.0817610062893082), Ro
w(stars=4, avg(cool)=1.0675944333996024), Row(stars=5, avg(cool)=2.2222222222222223)]
>>>
```

سوال 3 : برای این سوال، ابتدا با تابع filter، سطرهای دارای شرط review_count>=10 و open=True را جدا کرده و سپس روی فیلد stars گروه بندی کرده و برای گروه cool، میانگین را محاسبه می‌کنیم. با توجه به نتیجه؛ گزینه 4 صحیح است.

```
>>> yelp_df.filter("review_count>=10 and open=True").groupBy("stars").mean("cool").collect()
[Row(stars=2, avg(cool)=0.5714285714285714), Row(stars=3, avg(cool)=1.0456140350877192), Row
(stars=4, avg(cool)=1.0774193548387097), Row(stars=5, avg(cool)=2.25)]
>>>
>>>
```

سوال 4 : برای این سوال، ابتدا با تابع filter، سطرهای دارای شرط review_count>=10 و open=True را جدا کرده و سپس روی فیلد state گروه بندی کرده و برای هر گروه count، را محاسبه می‌کنیم. سپس نتیجه را بصورت نزولی

بر اساس count مرتب می کنیم. با توجه به نتیجه؛ گزینه 2 یعنی CO با 43 تا review جایگاه سوم را دارد و صحیح است.

```
>>> yelp_df.filter("review_count>=10 and open=True").groupBy("state").count().orderBy('count', ascending=False).collect()
[Row(state=u'AZ', count=538), Row(state=u'LA', count=53), Row(state=u'CO', count=43), Row(state=u'GA', count=33), Row(state=u'NY', cou
nt=27), Row(state=u'TX', count=20), Row(state=u'CA', count=20), Row(state=u'OR', count=19), Row(state=u'MN', count=19), Row(state=u'WA
', count=4), Row(state=u'ID', count=3)]
>>>
```

سوال 5 : برای این سوال، ابتدا با روی ستون business_id گروه‌بندی کرده و به ازای هر گروه؛ تعداد سطرها را بدست آورده و سپس در مرحله آخر، ماکزیمم تعداد سطرها را بدست می‌آوریم. با توجه به نتیجه؛ گزینه 4 صحیح است.

```
>>> yelp_df.groupBy("business_id").count().agg({"count":"max"}).collect()
[Row(max(count)=6)]
>>>
```