Distributed Computing

DIANE WOODBRIDGE, PH.D

Distributed Computing

Course Objectives

Understand needs and concepts of distributed computing.

Understand the Spark and its stack.

Being competent to work with Spark on a distributed computing environment.

- Programming with RDDs.
- Work with key/value pairs.
- Work with DataFrame.
- Work with SparkSQL, MLlib, ML, Spark Streaming, (and GarphX).
- Work on Amazon AWS.

Spark Interview Questions

What is Apache Spark?

Explain the key features of Spark.

~~**What is RDD?**~~

~~**How to create RDD.**~~

~~**What is "partitions"?**~~

~~**Types or RDD operations?**~~

~~**What is "transformation"?**~~

~~**What is "action"?**~~

~~**Functions of "spark core"?**~~

~~**What is "spark context"?**~~

~~**What is an "RDD lineage"?**~~

Which file systems does Spark support?

~~**List the various types of "Cluster Managers" in Spark.**~~

~~**What is "YARN"?**~~

~~**What is "Mesos"?**~~

~~**What is a "worker node"?**~~

~~**What is an "accumulator"?**~~

What is "Spark SQL" (Shark)?

What is "SparkStreaming"?

What is "GraphX"?

What is "MLlib"?

https://www.edureka.co/blog/interview-questions/top-apache-spark-interview-questions-2016/

Spark Interview Questions

~~What are the advantages of using Apache Spark over Hadoop MapReduce for big data processing?~~

~~What are the languages supported by Apache Spark for developing big data applications?~~

Can you use Spark to access and analyze data stored in Cassandra databases?

~~Is it possible to run Apache Spark on Apache Mesos?~~

~~How can you minimize data transfers when working with Spark?~~

~~Why is there a need for broadcast variables?~~

~~Name a few companies that use Apache Spark in production.~~

What are the various data sources available in SparkSQL?

What is the advantage of a Parquet file?

~~What do you understand by Pair RDD?~~

Is Apache Spark a good fit for Reinforcement learning?

https://www.dezyre.com/article/top-50-spark-interview-questions-and-answers-for-2016/208

Loading and Saving Data

Filesystems

- Local Filesystem
- Amazon S3
- HDFS

Loading and Saving Data

Filesystems

- Local Filesystem
  - Specify sc.textFile(file://path)
  - The filesystem should be available at the same path on all nodes in the cluster for both the master and executors. – If the file is not on all nodes in the cluster, load it locally and then call parallelize to distribute the contents to workers.

```
rdd = sc.textFile("file:///Users/dwoodbridge/Class/MSAN694/Example_Data/README.md")
```

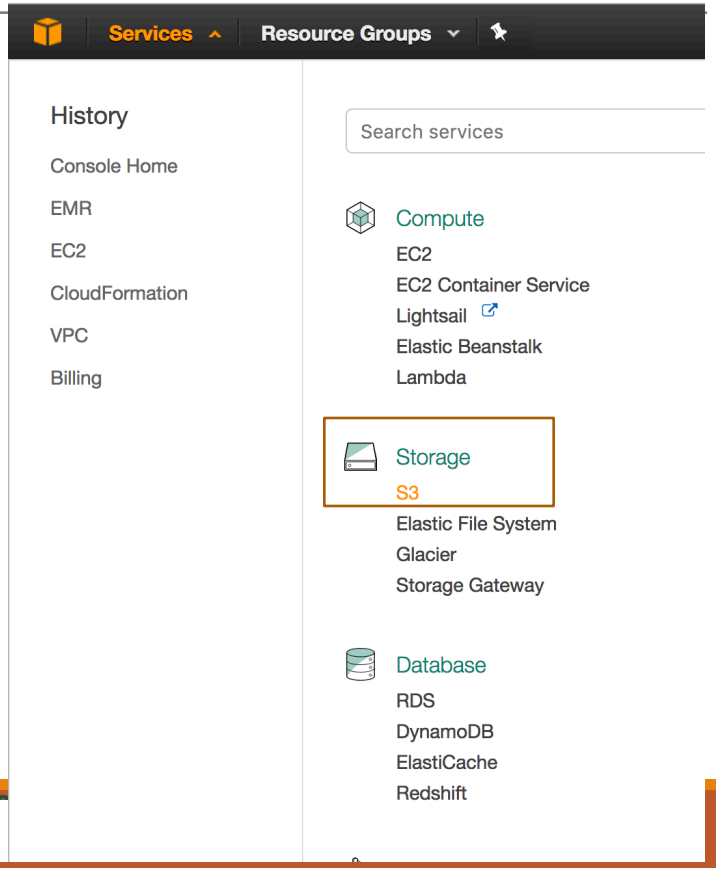
```
rdd
```

```
file:///Users/dwoodbridge/Class/MSAN694/Example_Data/README.md MapPartitionsRDD[1] at textFile at NativeMethodAccessorImpl.java:-2
```

Loading and Saving Data

Filesystems

- Amazon S3
  - S3 (Simple Storage Service) – Web Storage Service
    - Place data on S3.
      - Log in to the console on https://aws.amazon.com/.
      - Choose S3 Service.
      - Create a bucket and upload ”README.md”.

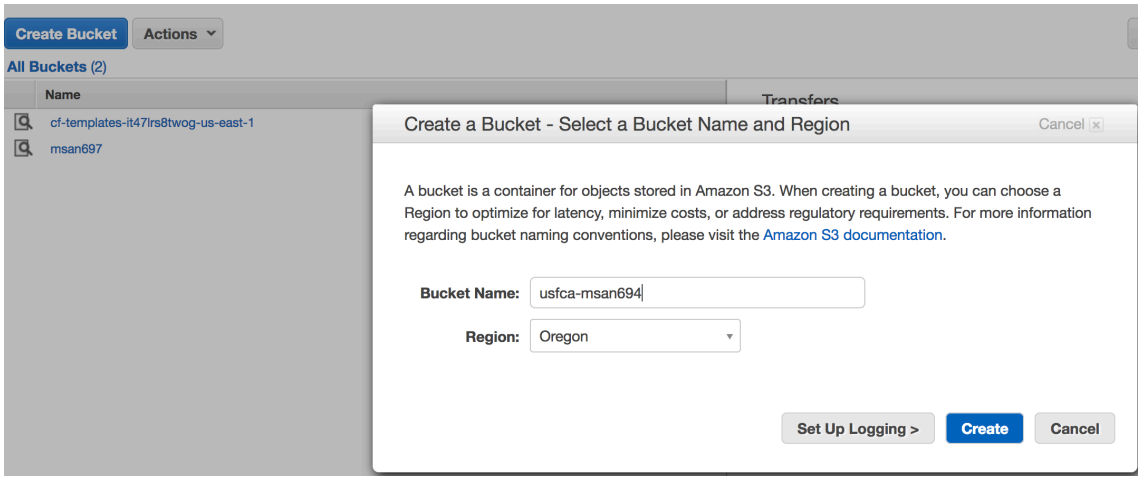


https://aws.amazon.com/s3/

Loading and Saving Data

Filesystems

- Amazon S3
  - S3 (Simple Storage Service) – Web Storage Service
    - Place data on S3.
      - Log in to the console on https://aws.amazon.com/.
      - Choose S3 Service.
      - Create a bucket and upload ”README.md”.

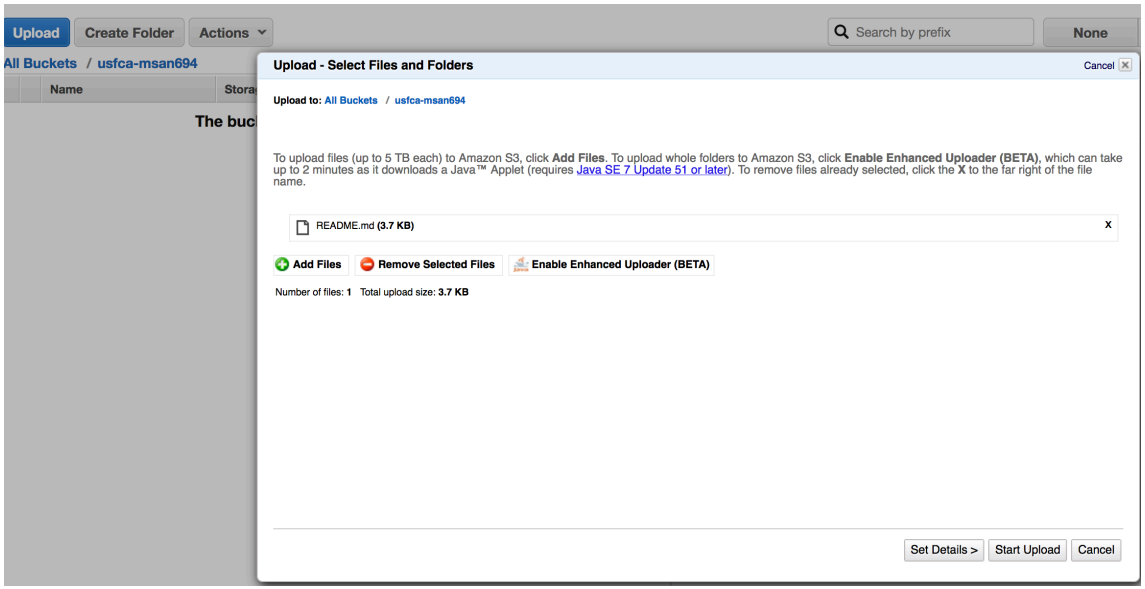


https://aws.amazon.com/s3/

Loading and Saving Data

Filesystems

- Amazon S3
  - S3 (Simple Storage Service) – Web Storage Service
    - Place data on S3.
      - Log in to the console on https://aws.amazon.com/.
      - Choose S3 Service.
      - Create a bucket and upload ”README.md”.

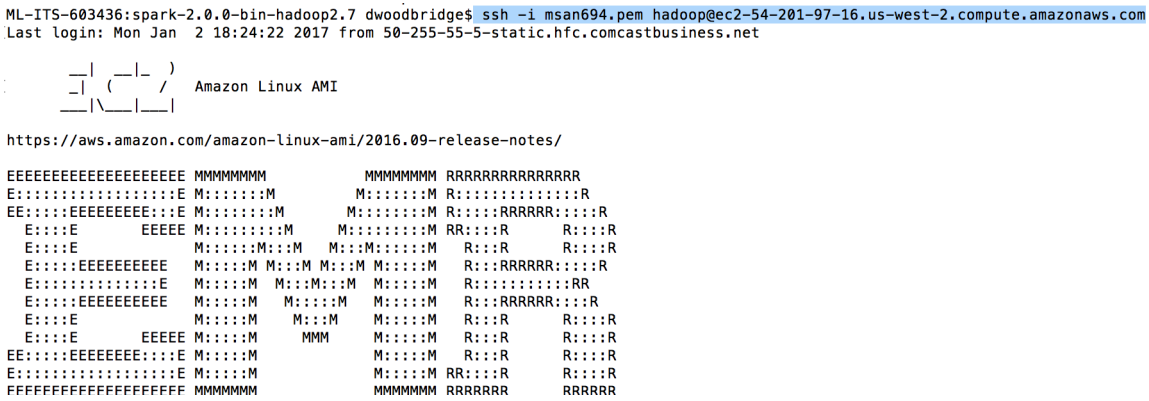


https://aws.amazon.com/s3/

Loading and Saving Data

Filesystems

- Amazon S3
  - S3 (Simple Storage Service) – Web Storage Service
    - Place data on S3.
      - Log in to the console on https://aws.amazon.com/.
      - Choose S3 Service.
      - Create a bucket and upload ”README.md”.
    - Load from S3.
      - Let’s try this on your EMR.

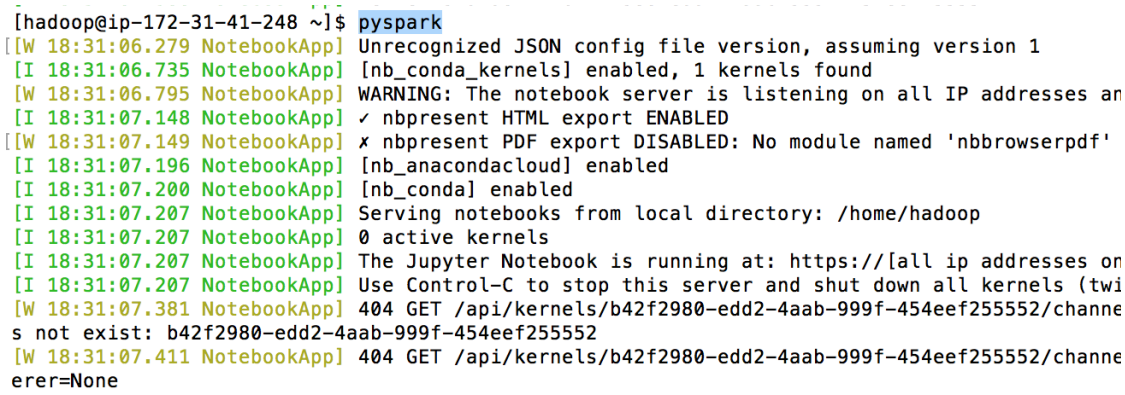


https://aws.amazon.com/s3/

Loading and Saving Data

Filesystems

- Amazon S3
  - S3 (Simple Storage Service) – Web Storage Service
    - Place data on S3.
      - Log in to the console on https://aws.amazon.com/.
      - Choose S3 Service.
      - Create a bucket and upload ”README.md”.
    - Load from S3.
      - Let’s try this on your EMR.
      - Pass a path starting with “s3n://”



https://aws.amazon.com/s3/

Example 1

Filesystems

- Amazon S3
  - Load from S3.
    - Use `sc.textfile("s3n://bucket_name/filename")`

Example 1

Filesystems

- Amazon S3
  - Load from S3.
    - Use `sc.textfile("s3n://bucket_name/filename")`

```
rdd = sc.textFile("s3n://usfca-msan694/README.md")
```

```
rdd
```

```
s3n://usfca-msan694/README.md MapPartitionsRDD[11] at textFile at NativeMethodAccessorImpl.java:-2
```

```
rdd.collect()
```

```
['# Apache Spark',
 '',
 'Spark is a fast and general cluster computing system for Big Data. It provides',
 'high-level APIs in Scala, Java, Python, and R, and an optimized engine that',
 'supports general computation graphs for data analysis. It also supports a',
 'rich set of higher-level tools including Spark SQL for SQL and DataFrames,',
 'MLlib for machine learning, GraphX for graph processing,',
 'and Spark Streaming for stream processing.',
```

Loading and Saving Data

Filesystems

- HDFS (Hadoop Distributed File system)
  - Designed to work on commodity hardware and be resilient to node failure, while providing high data throughput.
  - Specify “hdfs://master:port/path” for input and output data.

Loading and Saving Data

Databases

- Spark can access several popular databases using either Hadoop connectors or custom Spark connectors.
  - Postgres and others (MySQL) using JDBC
  - Cassandra
  - Elasticsearch
  - HBase

Loading and Saving Data

Databases

- Postgres and others (MySQL) using JDBC
  - Spark can load data from any relational databases that supports Java Database Connectivity (JDBC) including MySQL, Postgres, and other systems.
    - Download a Postgres JDBC jar from https://jdbc.postgresql.org/download.html
    - export SPARK_CLASSPATH=the_location_of_JDBC.jar:$SPARK_CLASSPATH

```
ML-ITS-603436:~ dwoodbridge$ export SPARK_CLASSPATH=~/spark-2.0.0-bin-hadoop2.7/postgresql-9.4.1212.jre6.jar:$SPARK_CLASSPATH
ML-ITS-603436:~ dwoodbridge$ echo $SPARK_CLASSPATH
/Users/dwoodbridge/spark-2.0.0-bin-hadoop2.7/postgresql-9.4.1212.jre6.jar::/usr/local/Cellar/hadoop/2.7.3/libexec/etc/aws-java-
adoop/2.7.3/libexec/etc/hadoop-aws-2.6.0.jar
ML-ITS-603436:~ dwoodbridge$ pyspark
[W 13:23:59.854 NotebookApp] Unrecognized JSON config file version, assuming version 1
[I 13:24:00.481 NotebookApp] [nb_conda_kernels] enabled, 1 kernels found
[I 13:24:00.536 NotebookApp] The port 8888 is already in use, trying another port.
```

Loading and Saving Data

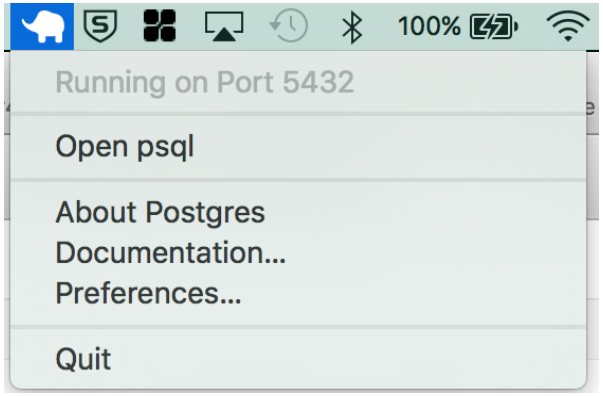
Databases

- Postgres and others (MySQL) using JDBC
  - Spark can load data from any relational databases that supports Java Database Connectivity (JDBC) including MySQL, Postgres, and other systems.
    - Download a Postgres JDBC jar from https://jdbc.postgresql.org/download.html
    - `export SPARK_CLASSPATH=the_location_of_JDBC.jar:$SPARK_CLASSPATH`
    - Using

```
from pyspark.sql import DataFrameReader
df = DataFrameReader(sqlContext).jdbc(url='jdbc:<URL>’,
table=’<table_name>', properties=<property_list>), where
<property_list> includes user, password.
```

Example 2

Connect to ‘ecommerce’ (if you still have it) and load data from items table in your Postgres.



Example 2

```
from pyspark.sql import DataFrameReader
```

```
url = 'postgresql://localhost:5432/ecommerce'
```

```
properties = {'user': 'dwoodbridge', 'password': ''}
```

```
df = DataFrameReader(sqlContext).jdbc(
    url='jdbc:%s' % url, table='items', properties=properties
)
```

```
df.collect()
```

```
[Row(itemid=47919421, name=u'Fathead San Francisco Giants Teammate Logo', shortdescription=u"This Fathead Teammate
is a high definition image made of tough high-grade vinyl that's tear and fade resistant, so you can be assured your
eye catching Fathead will stay strong. And if the thought of moving your Teammate scares you, have no fear, because
it uses a low-tack adhesive that can be easily moved and reused without any damage to your wall. Choose from
hundreds of licensed sports and entertainment images.", customerrating=None, numreviews=None),
 Row(itemid=47919896, name=u'Fathead San Francisco 49ers Teammate Logo', shortdescription=u"This Fathead Teammate is
a high definition image made of tough high-grade vinyl that's tear and fade resistant, so you can be assured your
eye catching Fathead will stay strong. And if the thought of moving your Teammate scares you, have no fear, because
it uses a low-tack adhesive that can be easily moved and reused without any damage to your wall. Choose from
```

Loading and Saving Data

Databases

- Cassandra
  - Use Spark Cassandra connector from DataStax.

    `pip install cassandra-driver`

```
ML-ITS-603436:~ dwoodbridge$ pip install cassandra-driver
Requirement already satisfied: cassandra-driver in ./anaconda/lib/python2.7/site-packages
Requirement already satisfied: futures in ./anaconda/lib/python2.7/site-packages (from cassandra-driver)
Requirement already satisfied: six>=1.6 in ./anaconda/lib/python2.7/site-packages (from cassandra-driver)
ML-ITS-603436:~ dwoodbridge$ conda install cassandra-driver
```

https://medium.com/@amirziai/running-pyspark-with-cassandra-in-jupyter-2bf5e95c319#.rg52vl8wt

Loading and Saving Data

Databases

- Cassandra
  - Use

```
from cassandra.cluster import Cluster
cluster = Cluster([<Address>])
session = cluster.connect('<keyspace>')
session.execute('<query>')
```

https://medium.com/@amirziai/running-pyspark-with-cassandra-in-jupyter-2bf5e95c319#.rg52vl8wt

Example 3

Connect to ‘ecommerce’ (if you still have it) and load data from items table in your Cassandra.

Example 3

```
from cassandra.cluster import Cluster
```

```
cluster = Cluster(['localhost'])
```

```
session = cluster.connect('ecommerce')
```

```
rows = session.execute('select * from items;')
```

```
for row in rows:
    print row
```

```
Row(itemid=49689948, customerrating=None, name=u'SkootZ Wristband, San Francisco Giants', numreviews=None, qty=48,
shortdescription=u'SkootZ Wristbands and fan gear are officially licensed. They are sweat resistant, stretchy and
extremely comfortable to wear. Rubber bracelets and silicone bracelets are no competition to Skootz wristbands.')
Row(itemid=52757295, customerrating=None, name=u'NFL San Francisco 49ers Baby Boys Football Print Bodysuit',
numreviews=None, qty=95, shortdescription=None)
Row(itemid=54065364, customerrating=None, name=u'NFL San Francisco 49ers 20 oz Ultra Tumbler', numreviews=None,
qty=64, shortdescription=None)
Row(itemid=35118615, customerrating=None, name=u'24oz NFL San Francisco 49Ers Squeeze Water Bottle with Filter
Feature', numreviews=None, qty=15, shortdescription=u'Show off your favorite NFL team with this officially licensed
squeeze water bottle. Sure to hydrate you and comes in a vibrant team color with team logo. Easy grip and squeeze
technique is ideal for working out, in the car, on the go or anywhere. Comes with a filter feature option that is
```

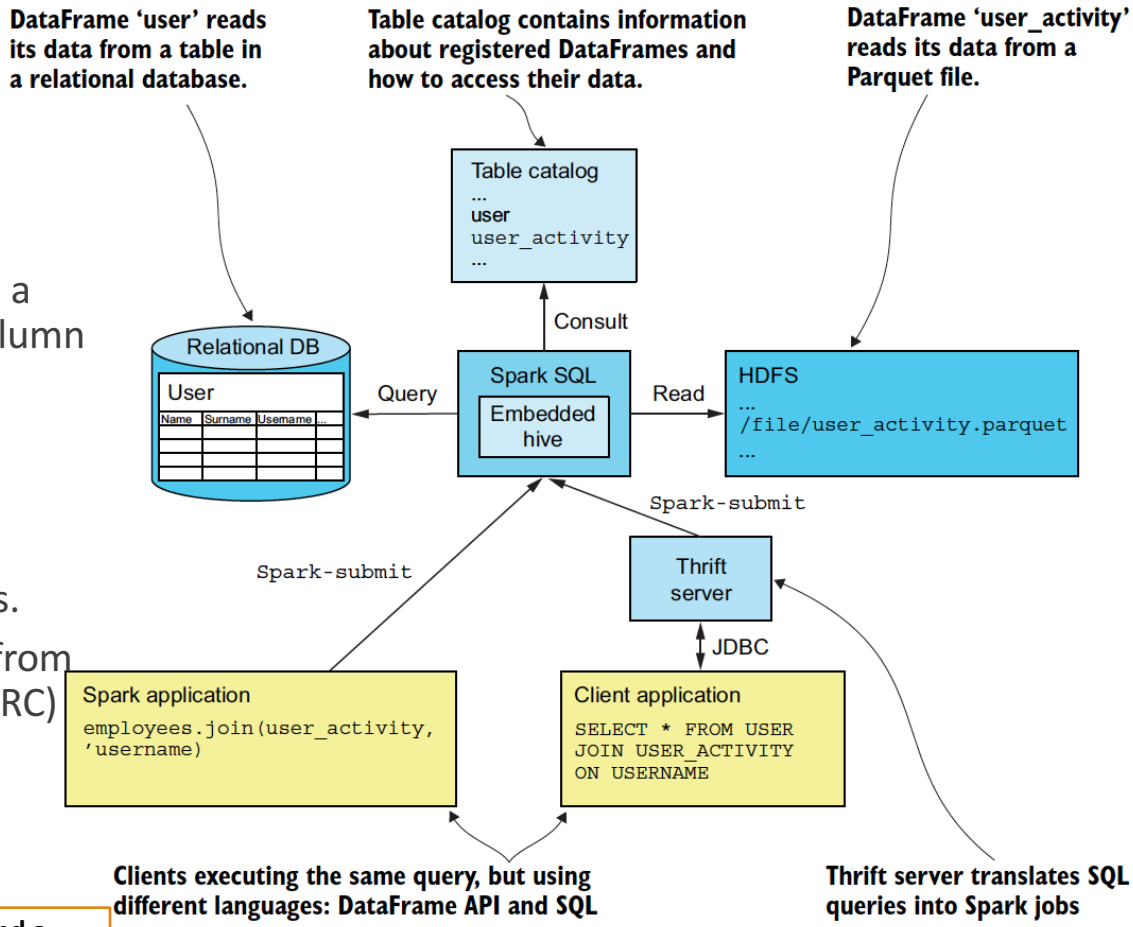
https://medium.com/@amirziai/running-pyspark-with-cassandra-in-jupyter-2bf5e95c319#.rg52vl8wt

Group Project

Spark SQL

DataFrame

- Handle structured, distributed data with a table-like representation with named column declared column types.
  - Cf. RDD : low-level and direct way of manipulating data in Spark.
- Special case of the DataSet type.
- You can join, query and save DataFrames.
- Spark SQL lets you register DataFrames from different sources (SQL, Parquet, JSON, ORC) as tables in the table catalog and query them.



An example of two clients (a Spark application using DataFrame DSL and a non-Spark client application connecting through JDBC) joining two tables: one in a relational database and another in a Parquet file in HDFS.

https://spark.apache.org/docs/2.0.1/api/java/org/apache/spark/sql/Dataset.html

Spark SQL

Creating DataFrames (3)

1. Convert existing RDDs.
2. Running SQL queries.
3. Loading external data.

Spark SQL

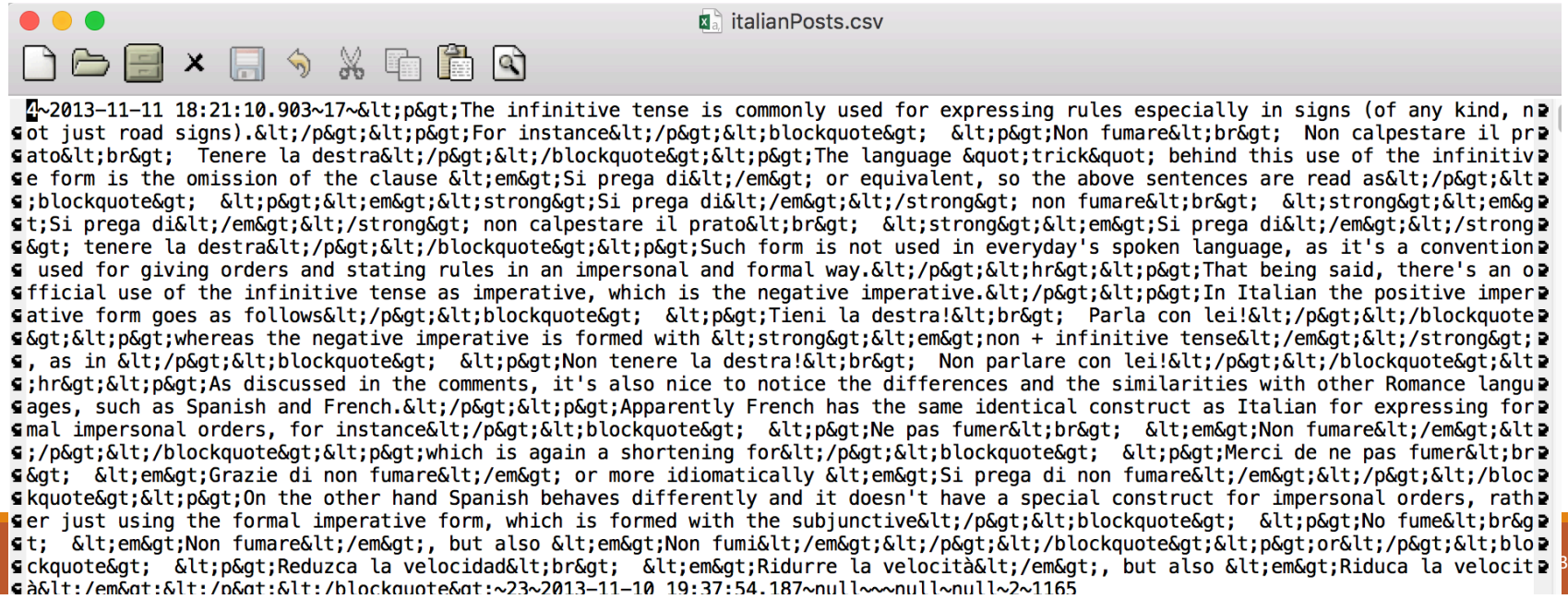
Creating DataFrames (3)

1. Convert existing RDDs.
   - In order to create DataFrames, you need to load data as text, parse the lines, and identify elements.
   - You can create DataFrames from RDD.
     1) Using RDDs containing row data as tuples.
        - Limited as it doesn't allow to specify all the schema attributes.
     2) Specifying a schema using createDataFrame.

Example Data

Italian_Stack_Exchange.

- Italian Language Stack Exchange data from stackexchange.com
- italianPosts.csv



Example Data

Located in s3n://usfca-msan694/Italian_Stack_Exchange/.

- Italian Language Stack Exchange data from stackexchange.com
- italianPosts.csv (each column is delimiated with ~)
  1. commentCount—Number of comments related to the question/answer
  2. lastActivityDate—Date and time of the last modification
  3. ownerUserId—User ID of the owner
  4. body—Textual contents of the question/answer
  5. score—Total score based on upvotes and downvotes
  6. creationDate—Date and time of creation
  7. viewCount—View count
  8. title—Title of the question
  9. tags—Set of tags the question has been marked with
  10. answerCount—Number of related answers
  11. acceptedAnswerId—If a question contains the ID of its accepted answer
  12. postTypeId—Type of the post; 1 is for questions, 2 for answers
  13. id—Post's unique ID

Example 4

Create DataFrame, using RDDs containing row data as tuples.

Step 1. Load the italianPosts.csv parsed with ”~” into an RDD.

Example 4

Step 1. Load the italianPosts.csv parsed with ”~” into an RDD.

```
#Step 1. Load the italianPosts.csv parsed with "~" into an RDD.
```

```
itPostsRows = sc.textFile("s3n://usfca-msan694/Italian_Stack_Exchange/italianPosts.csv")
```

```
itPostsSplit = itPostsRows.map(lambda x : x.split("~"))
```

Spark SQL

Creating DataFrames (3)

1. Convert existing RDDs.
   1) Using RDDs containing row data as tuples.
      - Convert the RDD's element arrays to tuples and then call toDF on the resulting RDD.
      - You can specify column names when calling the toDF method.
      - All the columns are string type.
      - All the columns are nullable.
   - ➢ You can check the schema using .printSchema().

Example 4

Step 2. Create DataFrame, using RDDs containing row data as tuples.

```
#Step 2-1. Create DataFrame, using RDDs containing row data as tuples.
```

```
itPostsRDD = itPostsSplit.map(lambda x : (x[0],x[1],x[2],x[3],x[4],x[5],x[6],x[7],x[8],x[9],x[10],x[11],x[12]))
```

```
itPostsRDD.collect()
```

```
[(u'4',
  u'2013-11-11 18:21:10.903',
  u'17',
  u"&lt;p&gt;The infinitive tense is commonly used for expressing rules especially in signs (of any kind, not just
road signs).&lt;/p&gt;&lt;p&gt;For instance&lt;/p&gt;&lt;blockquote&gt;  &lt;p&gt;Non fumare&lt;br&gt;  Non
calpestare il prato&lt;br&gt;  Tenere la destra&lt;/p&gt;&lt;/blockquote&gt;&lt;p&gt;The language &quot;trick&quot;
behind this use of the infinitive form is the omission of the clause &lt;em&gt;Si prega di&lt;/em&gt; or equivalent,
so the above sentences are read as&lt;/p&gt;&lt;blockquote&gt;  &lt;p&gt;&lt;em&gt;&lt;strong&gt;Si prega
di&lt;/em&gt;&lt;/strong&gt; non fumare&lt;br&gt;  &lt;strong&gt;&lt;em&gt;Si prega di&lt;/em&gt;&lt;/strong&gt; non
calpestare il prato&lt;br&gt;  &lt;strong&gt;&lt;em&gt;Si prega di&lt;/em&gt;&lt;/strong&gt; tenere la
destra&lt;/p&gt;&lt;/blockquote&gt;&lt;p&gt;Such form is not used in everyday's spoken language, as it's a
convention used for giving orders and stating rules in an impersonal and formal
way.&lt;/p&gt;&lt;hr&gt;&lt;p&gt;That being said, there's an official use of the infinitive tense as imperative,
which is the negative imperative.&lt;/p&gt;&lt;p&gt;In Italian the positive imperative form goes as
follows&lt;/p&gt;&lt;blockquote&gt;  &lt;p&gt;Tieni la destra!&lt;br&gt;  Parla con lei!&lt;/p&gt;&lt;
```

Example 4

Step 2. Create DataFrame, using RDDs containing row data as tuples.

```
#Step 2-1. Create DataFrame, using RDDs containing row data as tuples.
```

```
itPostsRDD = itPostsSplit.map(lambda x : (x[0],x[1],x[2],x[3],x[4],x[5],x[6],x[7],x[8],x[9],x[10],x[11],x[12]))
```

```
itPostsRDD.toDF()
```

```
DataFrame[_1: string, _2: string, _3: string, _4: string, _5: string, _6: string, _7: string, _8: string, _9: string,
 _10: string, _11: string, _12: string, _13: string]
```

```
itPostsRDD.toDF().show()
```

```
+---+--------------------+---+--------------------+---+--------------------+----+--------------------+--------------
-----+----+----+---+----+
| _1|                  _2| _3|                  _4| _5|                  _6|  _7|                  _8|
   _9| _10| _11|_12| _13|
+---+--------------------+---+--------------------+---+--------------------+----+--------------------+--------------
-----+----+----+---+----+
|  4|2013-11-11 18:21:...| 17|&lt;p&gt;The infi...| 23|2013-11-10 19:37:...|null|                    |
     |null|null|  2|1165|
|  5|2013-11-10 20:31:...| 12|&lt;p&gt;Come cre...|  1|2013-11-10 19:44:...|  61|Cosa sapreste dir...| &lt;word-choic
e&gt;|   1|null|  1|1166|
|  2|2013-11-10 20:31:...| 17|&lt;p&gt;Il verbo...|  5|2013-11-10 19:58:...|null|                    |
     |null|null|  2|1167|
|  1|2014-07-25 13:15:...|154|&lt;p&gt;As part ...| 11|2013-11-10 22:03:...| 187|Ironic constructi...|&lt;english-com
pa...|   4|1170|  1|1168|
```

Example 4

Step 2. Create DataFrame, using RDDs containing row data as tuples.

List column names of ["commentCount", "lastActivityDate", "ownerUserId", "body", "score", "creationDate", "viewCount", "title", "tags", "answerCount", "acceptedAnswerId", "postTypeId", "id"].

```
itPostsDF = itPostsRDD.toDF(["commentCount", "lastActivityDate", "ownerUserId", "body", "score", "creationDate", "viewC
```

```
itPostsDF.show()
```

```
+------------+--------------------+-----------+--------------------+-----+--------------------+---------+------------
--------+--------------------+-----------+---------------+----------+----+
|commentCount|    lastActivityDate|ownerUserId|                body|score|        creationDate|viewCount|
   title|                tags|answerCount|acceptedAnswerId|postTypeId|  id|
+------------+--------------------+-----------+--------------------+-----+--------------------+---------+------------
--------+--------------------+-----------+---------------+----------+----+
|           4|2013-11-11 18:21:...|         17|&lt;p&gt;The infi...|   23|2013-11-10 19:37:...|     null|
        |                    |       null|           null|         2|1165|
|           5|2013-11-10 20:31:...|         12|&lt;p&gt;Come cre...|    1|2013-11-10 19:44:...|       61|Cosa saprest
e dir...| &lt;word-choice&gt;|          1|           null|         1|1166|
|           2|2013-11-10 20:31:...|         17|&lt;p&gt;Il verbo...|    5|2013-11-10 19:58:...|     null|
        |                    |       null|           null|         2|1167|
```

Example 4

Step 2. Create DataFrame, using RDDs containing row data as tuples.

List column names of ["commentCount", "lastActivityDate", "ownerUserId", "body", "score", "creationDate", "viewCount", "title", "tags", "answerCount", "acceptedAnswerId", "postTypeId", "id"].

```
itPostsDF = itPostsRDD.toDF(["commentCount", "lastActivityDate", "ownerUserId", "body", "score", "creationDate", "viewC
```

```
itPostsDF.show()
```

```
+------------+--------------------+-----------+--------------------+-----+--------------------+---------+------------
--------+--------------------+-----------+---------------+----------+----+
|commentCount|    lastActivityDate|ownerUserId|                body|score|        creationDate|viewCount|
   title|                tags|answerCount|acceptedAnswerId|postTypeId|  id|
+------------+--------------------+-----------+--------------------+-----+--------------------+---------+------------
--------+--------------------+-----------+---------------+----------+----+
|           4|2013-11-11 18:21:...|         17|&lt;p&gt;The infi...|   23|2013-11-10 19:37:...|     null|
        |                    |       null|           null|         2|1165|
|           5|2013-11-10 20:31:...|         12|&lt;p&gt;Come cre...|    1|2013-11-10 19:44:...|       61|Cosa saprest
e dir...| &lt;word-choice&gt;|          1|           null|         1|1166|
|           2|2013-11-10 20:31:...|         17|&lt;p&gt;Il verbo...|    5|2013-11-10 19:58:...|     null|
        |                    |       null|           null|         2|1167|
```

Example 4

Step 2. Create DataFrame, using RDDs containing row data as tuples.

Check the schema.

- All the columns are string type.
- All the columns are nullable.

```
#Check the schema
```

```
itPostsDF.printSchema()
```

```
root
 |-- commentCount: string (nullable = true)
 |-- lastActivityDate: string (nullable = true)
 |-- ownerUserId: string (nullable = true)
 |-- body: string (nullable = true)
 |-- score: string (nullable = true)
 |-- creationDate: string (nullable = true)
 |-- viewCount: string (nullable = true)
 |-- title: string (nullable = true)
 |-- tags: string (nullable = true)
 |-- answerCount: string (nullable = true)
 |-- acceptedAnswerId: string (nullable = true)
 |-- postTypeId: string (nullable = true)
 |-- id: string (nullable = true)
```

Spark SQL

Creating DataFrames (3)

1. Convert existing RDDs.

2) Specifying a schema using createDataFrame.

- Use createDataFrame(*data*, *schema=None*, *samplingRatio=None*)
  - Creates a DataFrame from an RDD.
  - When schema is a list of column names, the type of each column will be inferred from data.
  - Parameters
    - **data** – an RDD of Row/tuple/list/dict, list, or pandas.DataFrame.
    - **schema** – a StructType or list of column names. default None.
    - **samplingRatio** – the sample ratio of rows used for inferring

https://spark.apache.org/docs/1.6.2/api/python/pyspark.sql.html

Example 5

Create DataFrame, specifying a schema using createDataFrame.

Step 1. Load the italianPosts.csv parsed with ”~” into an RDD.

```
#Step 1. Load the italianPosts.csv parsed with "~" into an RDD.
```

```
itPostsRows = sc.textFile("s3n://usfca-msan694/Italian_Stack_Exchange/italianPosts.csv")
```

```
itPostsSplit = itPostsRows.map(lambda x : x.split("~"))
```

Example 5

Step 2. Define functions to convert data set.

```
# Step 2. Define functions to convert data set.
```

```
from pyspark.sql import Row
from datetime import datetime
def toIntSafe(inval):
  try:
    return int(inval)
  except ValueError:
    return None

def toTimeSafe(inval):
  try:
    return datetime.strptime(inval, "%Y-%m-%d %H:%M:%S.%f")
  except ValueError:
    return None

def toLongSafe(inval):
  try:
    return long(inval)
  except ValueError:
    return None

def stringToPost(r):
  return Row(
    toIntSafe(r[0]),
    toTimeSafe(r[1]),
    toIntSafe(r[2]),
    r[3],
    toIntSafe(r[4]),
    toTimeSafe(r[5]),
    toIntSafe(r[6]),
    toIntSafe(r[7]),
    r[8],
    toIntSafe(r[9]),
    toLongSafe(r[10]),
    toLongSafe(r[11]),
    long(r[12]))
```

Example 5

Step 3. Define a StructType.

```
# Step 3. Define a StructType
```

```
from pyspark.sql.types import *
postSchema = StructType([
  StructField("commentCount", IntegerType(), True),
  StructField("lastActivityDate", TimestampType(), True),
  StructField("ownerUserId", LongType(), True),
  StructField("body", StringType(), True),
  StructField("score", IntegerType(), True),
  StructField("creationDate", TimestampType(), True),
  StructField("viewCount", IntegerType(), True),
  StructField("title", StringType(), True),
  StructField("tags", StringType(), True),
  StructField("answerCount", IntegerType(), True),
  StructField("acceptedAnswerId", LongType(), True),
  StructField("postTypeId", LongType(), True),
  StructField("id", LongType(), False)
  ])
```

Example 5

Step 4. Convert input data

```
# Step 4. Convert input data
```

```
rowRDD = itPostsSplit.map(lambda x: stringToPost(x))
```

Step 5. Create DataFrame with Schema

```
# Step 5. Create DataFrame with Schema
```

```
itPostsDFStruct = sqlContext.createDataFrame(rowRDD, postSchema)
```

Example 5

Output

```
itPostsDFStruct.printSchema()
```

```
root
 |-- commentCount: integer (nullable = true)
 |-- lastActivityDate: timestamp (nullable = true)
 |-- ownerUserId: long (nullable = true)
 |-- body: string (nullable = true)
 |-- score: integer (nullable = true)
 |-- creationDate: timestamp (nullable = true)
 |-- viewCount: integer (nullable = true)
 |-- title: string (nullable = true)
 |-- tags: string (nullable = true)
 |-- answerCount: integer (nullable = true)
 |-- acceptedAnswerId: long (nullable = true)
 |-- postTypeId: long (nullable = true)
 |-- id: long (nullable = false)
```

```
itPostsDFStruct.show()
```

```
+------------+-------------------+-----------+--------------------+-----+--------------------+---------+-----+------
-------------+-----------+---------------+----------+----+
|commentCount|    lastActivityDate|ownerUserId|                body|score|
creationDate|viewCount|title|               tags|answerCount|acceptedAnswerId|postTypeId|  id|
+------------+-------------------+-----------+--------------------+-----+--------------------+---------+-----+------
-------------+-----------+---------------+----------+----+
|           4|2013-11-11 18:21:...|         17|&lt;p&gt;The infi...|   23|2013-11-10 19:37:...|     null|
null|                    |       null|            null|         2|1165|
|           5|2013-11-10 20:31:...|         12|&lt;p&gt;Come cre...|    1|2013-11-10 19:44:...|       61| null|
&lt;word-choice&gt;|           1|            null|         1|1166|
|           2|2013-11-10 20:31:...|         17|&lt;p&gt;Il verbo...|    5|2013-11-10 19:58:...|     null|
null|                    |       null|            null|         2|1167|
|           1|2014-07-25 13:15:...|        154|&lt;p&gt;As part ...|   11|2013-11-10 22:03:...|      187|
```

DataFrame API Basics (Example 6)

DataFrame

- Immutable and lazy like RDDs.
  - Immutable : cannot directly change data in a DataFrame. You need to transform a DataFrame to another one.
  - Lazy : Do not return results.
- Basic APIs (Example 6)
  - select()
  - drop()
  - filter(),where()
  - withColumnRenamed(), witColumn() (Renaming and adding columns)
  - orderBy(), sort()
  - And many more : https://spark.apache.org/docs/1.6.2/api/python/pyspark.sql.html

DataFrame API Basics

DataFrame

- Basic APIs
  - select(<column_names> or dataframe[<field_name>])
    - Work with Column objects.

```
itPostsDFIdBody = itPostsDFStruct.select("id","body")
```

```
itPostsDFIdBody.show()
```

```
+----+--------------------+
|  id|                body|
+----+--------------------+
|1165|&amp;lt;p&amp;gt;The infi...|
|1166|&amp;lt;p&amp;gt;Come cre...|
|1167|&amp;lt;p&amp;gt;Il verbo...|
|1168|&amp;lt;p&amp;gt;As part ...|
|1169|&amp;lt;p&amp;gt;&amp;lt;em&amp;g...|
|1170|&amp;lt;p&amp;gt;There's ...|
|1171|&amp;lt;p&amp;gt;As other...|
|1172|&amp;lt;p&amp;gt;The expr...|
|1173|&amp;lt;p&amp;gt;When I w...|
|1174|&amp;lt;p&amp;gt;Wow, wha...|
|1175|&amp;lt;p&amp;gt;Suppose ...|
|1176|&amp;lt;p&amp;gt;Except w...|
|1177|&amp;lt;p&amp;gt;Both you...|
|1178|&amp;lt;blockquote&amp;gt...|
|1179|&amp;lt;p&amp;gt;Comparin...|
|1180|&amp;lt;p&amp;gt;Using th...|
|1181|&amp;lt;p&amp;gt;I would ...|
|1182|&amp;lt;p&amp;gt;Putting ...|
|1183|&amp;lt;p&amp;gt;Many peo...|
|1184|&amp;lt;p&amp;gt;Sono un'...|
+----+--------------------+
only showing top 20 rows
```

DataFrame API Basics

DataFrame

- Basic APIs
  - drop()
    - Select all except one column.

```
itPostsDFFilterId.drop("tags").show()
```

```
+------------+--------------------+-----------+--------------------+-----+--------------------+---------+-----+------
-----+----------------+----------+----+
|commentCount|    lastActivityDate|ownerUserId|                body|score|        creationDate|viewCount|title|answer
Count|acceptedAnswerId|postTypeId|  id|
+------------+--------------------+-----------+--------------------+-----+--------------------+---------+-----+------
-----+----------------+----------+----+
|           5|2014-06-04 21:56:...|        674|&lt;p&gt;Sardinia...|    2|2014-06-04 21:56:...|     null| null|
 null|            null|         2|2001|
|           1|2014-06-04 22:18:...|        674|&lt;p&gt;I am fro...|    3|2014-06-04 22:18:...|     null| null|
 null|            null|         2|2002|
|           3|2014-06-04 23:27:...|        193|&lt;p&gt;La rispo...|    6|2014-06-04 23:27:...|     null| null|
 null|            null|         2|2003|
|           1|2014-06-04 23:29:...|         70|&lt;p&gt;In itali...|    8|2014-06-04 23:29:...|     null| null|
 null|            null|         2|2004|
|           8|2014-06-14 07:37:...|        223|&lt;p&gt;Mi Ã¨ st...|    2|2014-06-05 12:04:...|      104| null|
    1|            2006|         1|2005|
```

DataFrame API Basics

DataFrame

- Basic APIs
  - filter(),where()
    - Apply constraints.

```
itPostsDFFilterId = itPostsDFStruct.where("id > 2000 and id < 2010")
```

```
itPostsDFFilterId.select("id","body").show()
```

```
+----+--------------------+
|  id|                body|
+----+--------------------+
|2001|&amp;lt;p&amp;gt;Sardinia...|
|2002|&amp;lt;p&amp;gt;I am fro...|
|2003|&amp;lt;p&amp;gt;La rispo...|
|2004|&amp;lt;p&amp;gt;In itali...|
|2005|&amp;lt;p&amp;gt;Mi Ã¨ st...|
|2006|&amp;lt;p&amp;gt;âCa.â...|
|2007|&amp;lt;p&amp;gt;Un chimi...|
|2008|&amp;lt;p&amp;gt;&amp;quot;sp...|
|2009|&amp;lt;p&amp;gt;Ad occhi...|
+----+--------------------+
```

DataFrame API Basics

DataFrame

- Basic APIs
  - withColumnRenamed(), withColumn()
    - Rename and add columns.

```
itPostsDFFilterId.withColumnRenamed("id","selected_id")
```

```
DataFrame[commentCount: int, lastActivityDate: timestamp, ownerUserId: bigint, body: string, score: int, creationDate
: timestamp, viewCount: int, title: string, tags: string, answerCount: int, acceptedAnswerId: bigint, postTypeId: big
int, selected id: bigint]
```

```
itPostsDFFWIthRatio = itPostsDFFilterId.withColumn("score_div_answer",itPostsDFFilterId['score']/itPostsDFFilterId['answerCount'])
```

```
itPostsDFFWIthRatio.where(itPostsDFFWIthRatio['score_div_answer'].isNotNull()).show()
```

```
|commentCount|    lastActivityDate|ownerUserId|                body|score|        creationDate|viewCount|title|
         tags|answerCount|acceptedAnswerId|postTypeId|  id|score_div_answer|
+------------+--------------------+-----------+--------------------+-----+--------------------+---------+-----+-----
--------------+-----------+----------------+----------+----+----------------+
|           8|2014-06-14 07:37:...|        223|&amp;lt;p&amp;gt;Mi Ã¨ st...|    2|2014-06-05 12:04:...|      104| null|&amp;lt;wo
rd-usage&amp;gt;|          1|            2006|         1|2005|             2.0|
```

DataFrame API Basics

DataFrame

- Basic APIs
  - orderBy(), sort()
    - Sort data.

```
itPostsDFFilterId.sort("id", ascending=0).show()
```

```
+------------+--------------------+----------+--------------------+-----+--------------------+---------+-----+------
--------------+-----------+----------------+----------+----+
|commentCount|    lastActivityDate|ownerUserId|                body|score|        creationDate|viewCount|title|
          tags|answerCount|acceptedAnswerId|postTypeId|  id|
+------------+--------------------+----------+--------------------+-----+--------------------+---------+-----+------
--------------+-----------+----------------+----------+----+
|           0|2014-06-06 13:18:...|       674|&lt;p&gt;Ad occhi...|    5|2014-06-06 13:18:...|     null| null|
             |       null|            null|         2|2009|
|           3|2014-06-07 12:11:...|       676|&lt;p&gt;&quot;sp...|    0|2014-06-06 12:29:...|     null| null|
             |       null|            null|         2|2008|
|           1|2014-07-02 19:39:...|       223|&lt;p&gt;Un chimi...|    2|2014-06-06 11:34:...|       45| null|&lt;wo
rd-usage&gt...|          1|            null|         1|2007|
|           0|2014-06-05 13:52:...|        37|&lt;p&gt;âCa.â...|    8|2014-06-05 13:52:...|     null| null|
             |       null|            null|         2|2006|
```

DataFrame API Basics

DataFrame

- Basic APIs
  - select()
  - drop()
  - filter(),where()
  - withColumnRenamed(), witColumn() (Renaming and adding columns)
  - orderBy(), sort()
  - And many more : https://spark.apache.org/docs/1.6.2/api/python/pyspark.sql.html

Example 7

Read “Italian_Stack_Exchange/italianComments.csv”.

Create a dataframe with first 3 columns in the file to have

- Id : long, not nullable
- commentDate: timestamp, nullable
- comment : string, nullable

```
root
 |-- id: long (nullable = false)
 |-- commentDate: timestamp (nullable = true)
 |-- comment: string (nullable = true)
```

Find a row that the commentDate is 2013-11-07 and include “@Daniele”.

https://spark.apache.org/docs/1.6.2/api/python/pyspark.sql.html

Example 8

Read “Italian_Stack_Exchange/italianVotes.csv”.

Create a dataframe in the file to have the following schema.

Sort the data with voteTypeId = 1 in an ascending order.

```
root
 |-- id: long (nullable = false)
 |-- postId: long (nullable = false)
 |-- voteTypeId: integer (nullable = false)
 |-- creationType: timestamp (nullable = false)
```

SQL functions with DataFrame

Use SQL functions for calculation.

- Spark SQL supports most of SQL functions.
  - Scalar functions : Return a single value for each row based on calculations on one or more columns.
  - Aggregate functions : Return a single value for a group of rows.
  - Window functions : Return several values for a group of rows.
  - User-defined functions (UDF) : Generate custom scalar or aggregate functions.

https://spark.apache.org/docs/1.6.1/api/java/org/apache/spark/sql/functions.html

SQL functions with DataFrame

Use SQL functions for calculation.

- Spark SQL supports most of SQL functions.
  - Scalar functions : Return a single value for each row based on calculations on one or more columns.
    - Math – abs, log, etc.
    - String – length, concat, trim, etc.
    - Time – year, date_add, datediff, next_day
  - Aggregate functions : Return a single value for a group of rows.
    - Can be used in combination with groupBy() .
    - Example : df.groupBy().avg(<column-name>), df.select(avg(df.column-name))

https://spark.apache.org/docs/1.6.1/api/java/org/apache/spark/sql/functions.html

Example 9

From itPostsDFStruct (DataFrame from italianPost)

1. Add a column called “duration”, which is difference between ‘lastActivityDate’ and ‘creationDate’ and sort the DataFrame by ‘duration’.
2. Calculate an average score per owner and sort data by average scores.

Example 9

```
itPostsDFStruct.withColumn('duration',datediff('lastActivityDate','creationDate')).sort('duration',ascending =0).show()
```

```
+------------+--------------------+-----------+--------------------+-----+--------------------+---------+-----+------
-------------+-----------+---------------+----------+----+--------+
|commentCount|    lastActivityDate|ownerUserId|                body|score|        creationDate|viewCount|title|
          tags|answerCount|acceptedAnswerId|postTypeId|  id|duration|
+------------+--------------------+-----------+--------------------+-----+--------------------+---------+-----+------
-------------+-----------+---------------+----------+----+--------+
|           3|2014-09-11 14:37:...|         63|&lt;p&gt;The plur...|    5|2013-11-12 13:34:...|       59| null|&lt;pl
ural&gt;&lt...|          1|           1227|         1|1221|     303|
|           4|2014-09-09 08:54:...|         63|&lt;p&gt;Some wor...|    4|2013-11-12 11:03:...|       80| null|&lt;no
uns&gt;&lt;...|          2|           1207|         1|1205|     301|
|           0|2014-09-12 10:55:...|          8|&lt;p&gt;Wikipedi...|   10|2013-11-20 16:42:...|      145| null|&lt;or
thography&g...|          5|           1336|         1|1321|     296|
|           0|2014-08-31 20:19:...|         12|&lt;p&gt;Giulia e...|    5|2013-11-12 12:04:...|      610| null|&lt;gr
ammar&gt;&l...|          4|           null|         1|1211|     292|
|           3|2014-08-24 16:01:...|        114|&lt;p&gt;Is the p...|   10|2013-11-06 22:12:...|       88| null|&lt;gr
ammar&gt;&l...|          1|             77|         1|  75|     291|
|           5|2014-09-05 21:35:...|         12|&lt;p&gt;Soprattu...|    3|2013-11-23 14:54:...|      249| null|&lt;gr
ammar&gt;&l...|          7|           null|         1|1369|     286|
|           2|2014-08-19 15:39:...|         12|&lt;p&gt;Quando s...|    4|2013-11-11 18:52:...|      149| null|&lt;id
ioms&gt;&lt...|          2|           null|         1|1187|     281|
|           0|2014-08-12 12:47:...|        124|&lt;p&gt;When do ...|    4|2013-11-07 17:43:...|      139| null| &lt;w
ord-choice&gt;|          3|             88|         1|  86|     278|
|           1|2014-08-18 18:06:...|         12|&lt;p&gt;SÃ¬, ho ...|    3|2013-11-13 19:04:...|      114| null|&lt;gr
ammar&gt;&l...|          2|           null|         1|1254|     278|
```

Example 9

```
itPostsDFStruct.groupBy('ownerUserId').avg('score').sort('avg(score)',ascending=0).show()
```

```
+-----------+-----------------+
|ownerUserId|       avg(score)|
+-----------+-----------------+
|        570|             15.0|
|          6|             15.0|
|        730|             12.0|
|        729|             11.0|
|        154|             11.0|
|        220|             10.0|
|        217|             10.0|
|        116|              9.0|
```

SQL functions with DataFrame

Use SQL functions for calculation.

- Spark SQL supports most of SQL functions.
  - Window functions : Return several values for a group of rows.
    - Unlike aggregate functions, they don't group rows into a single output per group.
    - You should define a "moving group" of rows called frames.

	SQL	DataFrame API
Ranking functions	rank	rank
	dense_rank	denseRank
	percent_rank	percentRank
	ntile	ntile
	row_number	rowNumber
Analytic functions	cume_dist	cumeDist
	first_value	firstValue
	last_value	lastValue
	lag	lag
	lead	lead

SQL functions with DataFrame

Use SQL functions for calculation. (Steps)

1. Construct a column definition with an aggregate function or window functions.
2. Build a window specification and use it as an argument to over() function.
   1. Define the partition using partitionBy() function.
   2. Specify ordering in the partition using orderBy().

Window Functions.

Function name	Description
`first (column)`	Returns the value in the first row in the frame.
`last (column)`	Returns the value in the last row in the frame.
`lag (column, offset, [default])`	Returns the value in the row that is `offset` rows behind the row in the frame. Use `default` if such a row doesn't exist.
`lead (column, offset, [default])`	Returns the value in the row that is `offset` rows before the row in the frame. Use `default` if such a row doesn't exist.
`ntile (n)`	Divides the frame into `n` parts and returns the part index of the row. If the number of rows in the frame isn't divisible by `n` and division gives a number between `x` and `x+1`, the resulting parts will contain `x` or `x+1` rows, with parts containing `x+1` rows coming first.
`cumeDist`	Calculates the fraction of rows in the frame whose value is less than or equal to the value in the row being processed.
`rank`	Returns the rank of the row in the frame (first, second, and so on). Rank is calculated by the value.
`denseRank`	Returns the rank of the row in the frame (first, second, and so on), but puts the rows with equal values in the same rank.
`percentRank`	Returns the rank of the row divided by the number of rows in the frame.
`rowNumber`	Returns the sequential number of the row in the frame.

Example 10

For each question, find the id of its owner's previous question by creation date.

```
winDf = itPostsDFStruct.filter(itPostsDFStruct.postTypeId == 1).select('ownerUserId', 'id', 'creationDate',
lag(itPostsDFStruct.id, 1).over(Window.partitionBy(itPostsDFStruct.ownerUserId).orderBy(itPostsDFStruct.creationDate))
.alias("prev")).orderBy(itPostsDFStruct.ownerUserId, itPostsDFStruct.id).show()
```

Example 10

For each question, find the id of its owner's **previous and next** questions by creation date.

SQL functions with DataFrame

Use SQL functions for calculation.

- Spark SQL supports most of SQL functions.
  - User-defined functions (UDF) : Generate custom scalar or aggregate functions.
    - udf() lets you extend the built-in functionalities.

```
udf_name = udf(lambda function definition)
```

Example 11

Create a UDF called countTags which counts the number of < and count the number of them in the tags column.

```
countTags = udf(lambda (x) : x.count("&lt;"))
```

```
itPostsDFStruct.select('id',countTags('tags')).show()
```

Example 12

Create a UDF called scoreString which discretize scores

- Lower than 10 : low
- Between 10 and 20 : med
- Higher than 20 : high

and show the score strings.

Example 12

Create a UDF called scoreString which discretize scores

- Lower than 10 : low
- Between 10 and 20 : med
- Higher than 20 : high

and show the score strings.

```
from pyspark.sql.functions import *
def score_discrete(x):
    if x < 10 :
        return 'low'
    elif (x>10 and x<20):
        return 'med'
    else:
        return 'high'
score_string = udf(lambda x: score_discrete(x))
```

```
itPostsDFStruct.select('id','score',score_string('score')).show()
```

```
+----+-----+---------------+
|  id|score|<lambda>(score)|
+----+-----+---------------+
|1165|   23|           high|
|1166|    1|            low|
|1167|    5|            low|
|1168|   11|            med|
|1169|    3|            low|
|1170|    8|            low|
|1171|    3|            low|
|1172|    1|            low|
|1173|    5|            low|
```

Grouping and Joining DataFrame

Grouping Data

- groupBy()
  - take column names or a list of column objects.
  - Return GroupedData Object.
  - Can use aggregate functions or agg(list_of_aggregate_fuctions).

Example 13

Calculate min, max, average score per ownerUserId.

```
from pyspark.sql.functions import *
itPostsDFStruct.groupBy('ownerUserId').agg(max(itPostsDFStruct['score']), min(itPostsDFStruct['score']), avg(itPostsDFStruct['score'])).show()
```

```
+-----------+----------+----------+------------------+
|ownerUserId|max(score)|min(score)|        avg(score)|
+-----------+----------+----------+------------------+
|        270|         1|         1|               1.0|
|        730|        12|        12|              12.0|
|        720|         1|         1|               1.0|
|         19|        10|        -1| 3.076923076923077|
|        348|         5|         5|               5.0|
|        415|         5|         1|2.1666666666666665|
|        656|         9|         9|               9.0|
|        736|         1|         1|               1.0|
|         22|        19|         0| 4.886363636363637|
|        198|         5|         5|               5.0|
|         77|         9|         0| 5.166666666666667|
|        677|         6|         6|               6.0|
|        202|         3|        -1|               1.0|
|        228|         2|         2|               2.0|
|        624|         4|         4|               4.0|
|        570|        15|        15|              15.0|
|        421|         9|         5|              7.25|
|        229|         8|         2|               4.0|
|        190|         4|         0|1.8333333333333333|
|         57|        13|         3|               8.3|
+-----------+----------+----------+------------------+
```

Grouping and Joining DataFrame

Joining Data

- join(dataframe, condition, join_type)
  - Join types : *inner*, *outer*, *left_outer*, *right_outer*, *leftsemi*

```
>>> df.join(df2, df.name == df2.name, 'outer').select(df.name, df2.height).collect()
[Row(name=None, height=80), Row(name=u'Alice', height=None), Row(name=u'Bob', height=85)]
```

```
>>> df.join(df2, 'name', 'outer').select('name', 'height').collect()
[Row(name=u'Tom', height=80), Row(name=u'Alice', height=None), Row(name=u'Bob', height=85)]
```

```
>>> cond = [df.name == df3.name, df.age == df3.age]
>>> df.join(df3, cond, 'outer').select(df.name, df3.age).collect()
[Row(name=u'Bob', age=5), Row(name=u'Alice', age=2)]
```

```
>>> df.join(df2, 'name').select(df.name, df2.height).collect()
[Row(name=u'Bob', height=85)]
```

Example 14

Join Posts DataFrame with Vote DataFrame with itPostsDFStruct.id == itVotesDFStruct.postId.

```
itPostsDFStruct.join(itVotesDFStruct,itPostsDFStruct.id == itVotesDFStruct.postId,'inner').
```

Registering DataFrame in the Table Catalog and Using SQL Commands

Registering DataFrame in the Table Catalog

- You can reference a DataFrame by its name by registering the DataFrame as a table.
- Spark stores the table defitition in the table catalog.
- Save as table (2)
  1. .registerTempTable(<table_name>)
     - disappear with the Spark session.
  2. .write.saveAsTable(<table_name>)
     - register a table permanently.
     - The table definitions will survive your application's restarts and are persistent.

Example 15

Save Post DataFrame as “Posts” and Vote DataFrame as “Votes”.

Example 15

Save Post DataFrame as “Posts” and Vote DataFrame as “Votes”.

```
itPostsDFStruct.write.saveAsTable("Posts")
```

```
itVotesDFStruct.write.saveAsTable("Votes")
```

Registering DataFrame in the Table Catalog and Using SQL Commands

Registering DataFrame in the Table Catalog

- You can reference a DataFrame by its name by registering the DataFrame as a table.
- Spark stores the table defitition in the table catalog.
- Save as table (2)
  1. .registerTempTable(<table_name>)
     - disappear with the Spark session.
  2. .write.saveAsTable(<table_name>)
     - register a table permanently.
     - The table deifitions will survive your application's restarts and are persistent.

Registering DataFrame in the Table Catalog and Using SQL Commands

Using SQL Commands

- Once DataFrame is registered as a table, you can query its data using SQL expressions.
  - Using sqlContext.sql(sql_query).
  - Also can use spark-sql.

Example 16

Run “select * from posts” query.

Example 16

Run “select * from posts” query.

```
resultDf = sqlContext.sql("select * from posts")
```

```
resultDf.show()
```

```
+------------+--------------------+----------+--------------------+-----+--------------------+---------+-----+------
--------------+----------+---------------+----------+----+
|commentCount|    lastActivityDate|ownerUserId|                body|score|        creationDate|viewCount|title|
         tags|answerCount|acceptedAnswerId|postTypeId|  id|
+------------+--------------------+----------+--------------------+-----+--------------------+---------+-----+------
--------------+----------+---------------+----------+----+
|           4|2013-11-11 18:21:...|        17|&amp;lt;p&amp;gt;The infi...|   23|2013-11-10 19:37:...|     null| null|
              |        null|           null|         2|1165|
|           5|2013-11-10 20:31:...|        12|&amp;lt;p&amp;gt;Come cre...|    1|2013-11-10 19:44:...|       61| null| &amp;lt;w
ord-choice&amp;gt;|           1|           null|         1|1166|
|           2|2013-11-10 20:31:...|        17|&amp;lt;p&amp;gt;Il verbo...|    5|2013-11-10 19:58:...|     null| null|
              |        null|           null|         2|1167|
|           1|2014-07-25 13:15:...|       154|&amp;lt;p&amp;gt;As part ...|   11|2013-11-10 22:03:...|      187| null|&amp;lt;en
glish-compa...|           4|           1170|         1|1168|
```

Loading and Saving Data using SparkSQL

Datatypes

- JSON
  - Spark can automatically infer a JSON schema.
- ORC
  - Optimized Row Columnar.
  - Columnar format.
- Parquet
  - Designed to be independent of any specific framework and free of unnecessary dependencies.
  - Columnar format.
- Relational Databases and JDBC

Loading Data with SparkSQL

- JSON
- Relational DB and Cassandra – covered earlier.

Loading and Saving Data using SparkSQL

Loading Data with SparkSQL

- JSON
  - Use spark.read.json(<file_name>).
- ORC
  - Use sqlContext.read.format("orc").load(<file_name>).
- Parquet
  - Use spark.read.parquet(<file_name>).
- Relational DB and Cassandra – covered earlier.

Example 17

Read world_bank_project.json as a DataFrame.

Example 17

Read s3n://usfca-msan694/world_bank_project.json as a DataFrame.

```
world_bank_prj = spark.read.json("s3n://usfca-msan694/world_bank_project.json")
```

```
world_bank_prj.show()
```

```
+--------------------+----------+--------------------+--------------------+--------------------+--------------------+
--------------------+-----------+--------------------+--------------------+--------------------+--------------------+
-+--------+-----------+-------+----------+--------------------+--------------------+----------------+---------------+
--------------------+--------------------+--------------------+--------------------+-----------+--------+-----------
--------+---------------+--------------------+--------------------+--------------------+--------------------+--------
-----------+--------------------+--------------------+--------------------+--------------------+--------------------
+--------------------+--------------------+--------------+------+------+--------------------+--------------------+---
-----------------+--------------+---------+------------+--------------------+
|                 _id|approvalfy|board_approval_month|   boardapprovaldate|            borrower|         closingdate|
    country_namecode|countrycode|         countryname|  countryshortname|               docty|envassesmentcategorycod
e|grantamt|ibrdcommamt|     id|idacommamt|           impagency|        lendinginstr|lendinginstrtype|lendprojectcost|
 majorsector_percent|   mjsector_namecode|            mjtheme|    mjtheme_namecode|mjthemecode|prodline|        prod
linetext|productlinetype|    project_abstract|        project_name|         projectdocs|projectfinancialtype|projects
tatusdisplay|          regionname|               sector|              sector1|              sector2|             sector3
|             sector4|     sector_namecode|    sectorcode|source|status|supplementprojectflg|              theme1|
   theme_namecode|     themecode| totalamt|totalcommamt|                 url|
+--------------------+----------+--------------------+--------------------+--------------------+--------------------+
--------------------+-----------+--------------------+--------------------+--------------------+--------------------+
-+--------+-----------+-------+----------+--------------------+--------------------+----------------+---------------+
--------------------+--------------------+--------------------+--------------------+-----------+--------+-----------
```

Loading and Saving Data using SparkSQL

Saving Data with SparkSQL

- SaveAsTable
  - .write.format(<format>).saveAsTable(<name>)

Example 18

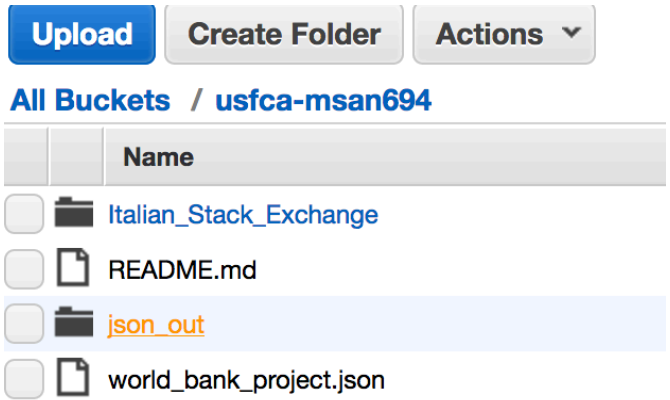
Save posts in the table catalog to a json_out folder in your S3 bucket.

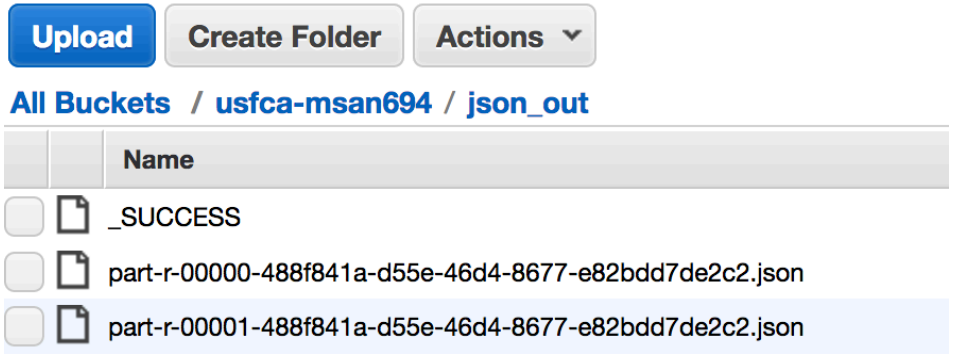
Example 18

Save posts in the table catalog to a json_out folder in your S3 bucket.

```
postsDf = sqlContext.sql("select * from posts")
```

```
postsDf.write.format("json").saveAsTable("post_json", path="s3n://usfca-msan694/json_out")
```





Reference

Distributed Computing with Spark, Reza Zadeh, http://stanford.edu/~rezab/slides/bayacm_spark.pdf.

Spark Online Documentation : http://spark.apache.org/docs/latest/

Karau, Holden, et al. *Learning spark: lightning-fast big data analysis*. O'Reilly Media, Inc., 2015.

Zecevic, Petar, et al. Spark in Action, Manning, 2016.