RDDs and DataFrames

- **Creating RDDs and DataFrames using SparkContext**
- **Interoperability between RDDs and DataFrames**
- **Multiple rows and multiple column specifications for DataFrames**
- **Creating DataFrames using SQLContext**
- **Selecting, editing and renaming columns in dataframes**
- **Interoperability between Pandas and Spark dataframes**

In [1]:

```
sc
```

Out[1]:

SparkContext

Spark UI

Version

`v2.3.0`

Master

`local[*]`

AppName

`PySparkShell`

In [65]:

```
from pyspark.sql.types import Row
from datetime import datetime
```

Creating RDDs using sc.parallelize()

In [28]:

```
simple_data = sc.parallelize([1, "Alice", 50])
simple_data
```

Out[28]:

```
ParallelCollectionRDD[30] at parallelize at PythonRDD.scala:175
```

In [29]:

```
simple_data.count()
```

Out[29]:

```
3
```

In [30]:

```
simple_data.first()
```

Out[30]:

```
1
```

In [31]:

```
simple_data.take(2)
```

Out[31]:

```
[1, 'Alice']
```

In [32]:

```
simple_data.collect()
```

Out[32]:

```
[1, 'Alice', 50]
```

This is an ERROR!

- **This RDD does not have "columns", it cannot be represented as a tabular data frame**
- **DataFrames are structured datasets**

In [ ]:

```
df = simple_data.toDF()
```

RDDs with records using sc.parallelize()

In [42]:

```
records = sc.parallelize([[1, "Alice", 50], [2, "Bob", 80]])
records
```

Out[42]:

```
ParallelCollectionRDD[52] at parallelize at PythonRDD.scala:175
```

In [43]:

```
records.collect()
```

Out[43]:

```
[[1, 'Alice', 50], [2, 'Bob', 80]]
```

In [44]:

```
records.count()
```

Out[44]:

```
2
```

In [45]:

```
records.first()
```

Out[45]:

```
[1, 'Alice', 50]
```

In [46]:

```
records.take(2)
```

Out[46]:

```
[[1, 'Alice', 50], [2, 'Bob', 80]]
```

In [47]:

```
records.collect()
```

Out[47]:

```
[[1, 'Alice', 50], [2, 'Bob', 80]]
```

This is an NOT an error!

This is an NOT an error!

- **This RDD does have "columns", it can be represented as a tabular data frame**

In [48]:

```
df = records.toDF()
```

In [49]:

```
df
```

Out[49]:

```
DataFrame[_1: bigint, _2: string, _3: bigint]
```

In [50]:

```
df.show()
```

```
+---+-----+---+
| _1|   _2| _3|
+---+-----+---+
|  1|Alice| 50|
|  2|  Bob| 80|
+---+-----+---+
```

Creating dataframes using sc.parallelize() and Row() functions

- **Row functions allow specifying column names for dataframes**

In [51]:

```
data = sc.parallelize([Row(id=1,
                           name="Alice",
                           score=50)])
data
```

Out[51]:

```
ParallelCollectionRDD[68] at parallelize at PythonRDD.scala:175
```

In [53]:

```
data.count()
```

Out[53]:

```
1
```

In [52]:

```
data.collect()
```

Out[52]:

```
[Row(id=1, name='Alice', score=50)]
```

In [54]:

```
df = data.toDF()
df.show()
```

```
+---+-----+-----+
| id| name|score|
+---+-----+-----+
|  1|Alice|   50|
+---+-----+-----+
```

Working with multiple rows

In [66]:

```
data = sc.parallelize([Row(
                            id=1,
                            name="Alice",
                            score=50
                        ),
                        Row(
                             id=2,
                             name="Bob",
                             score=80
                        ),
                        Row(
                             id=3,
                             name="Charlee",
                             score=75
                        )])
```

In [67]:

```
df = data.toDF()
df.show()
```

```
+---+-------+-----+
| id|   name|score|
+---+-------+-----+
|  1|  Alice|   50|
|  2|    Bob|   80|
|  3|Charlee|   75|
+---+-------+-----+
```

Multiple columns with complex data types

In [71]:

```
complex_data = sc.parallelize([Row(
                                    col_float=1.44,
                                    col_integer=10,
                                    col_string="John")
                               ])
```

In [72]:

```
complex_data_df = complex_data.toDF()
complex_data_df.show()
```

```
+---------+-----------+----------+
|col_float|col_integer|col_string|
+---------+-----------+----------+
|     1.44|         10|      John|
+---------+-----------+----------+
```

In [73]:

```
complex_data = sc.parallelize([Row(
                                    col_float=1.44,
                                    col_integer=10,
                                    col_string="John",
                                    col_boolean=True,
                                    col_list=[1, 2, 3])
                               ])
```

In [74]:

```
complex_data_df = complex_data.toDF()
```

```
complex_data_df.show()
```

```
+-----------+---------+-----------+---------+----------+
|col_boolean|col_float|col_integer| col_list|col_string|
+-----------+---------+-----------+---------+----------+
|       true|     1.44|         10|[1, 2, 3]|      John|
+-----------+---------+-----------+---------+----------+
```

In [79]:

```
complex_data = sc.parallelize([Row(
                                col_list = [1, 2, 3],
                                col_dict = {"k1": 0, "k2": 1, "k3": 2},
                                col_row = Row(columnA = 10, columnB = 20, columnC = 30)
,
                                col_time = datetime(2014, 8, 1, 14, 1, 5)
                            )])
```

In [80]:

```
complex_data_df = complex_data.toDF()
complex_data_df.show()
```

```
+--------------------+---------+------------+--------------------+
|            col_dict| col_list|     col_row|            col_time|
+--------------------+---------+------------+--------------------+
|[k3 -> 2, k1 -> 0...|[1, 2, 3]|[10, 20, 30]|2014-08-01 14:01:05|
+--------------------+---------+------------+--------------------+
```

Multiple rows with complex data types

In [89]:

```
complex_data = sc.parallelize([Row(
                                col_list = [1, 2, 3],
                                col_dict = {"k1": 0},
                                col_row = Row(a=10, b=20, c=30),
                                col_time = datetime(2014, 8, 1, 14, 1, 5)
                            ),
                            Row(
                                col_list = [1, 2, 3, 4, 5],
                                col_dict = {"k1": 0,"k2": 1 },
                                col_row = Row(a=40, b=50, c=60),
                                col_time = datetime(2014, 8, 2, 14, 1, 6)
                            ),
                            Row(
                                col_list = [1, 2, 3, 4, 5, 6, 7],
                                col_dict = {"k1": 0, "k2": 1, "k3": 2 },
                                col_row = Row(a=70, b=80, c=90),
                                col_time = datetime(2014, 8, 3, 14, 1, 7)
                            )])
```

In [90]:

```
complex_data_df = complex_data.toDF()
complex_data_df.show()
```

```
+--------------------+--------------------+------------+--------------------+
|            col_dict|            col_list|     col_row|            col_time|
+--------------------+--------------------+------------+--------------------+
|           [k1 -> 0]|           [1, 2, 3]|[10, 20, 30]|2014-08-01 14:01:05|
|  [k1 -> 0, k2 -> 1]|     [1, 2, 3, 4, 5]|[40, 50, 60]|2014-08-02 14:01:06|
|[k3 -> 2, k1 -> 0...|[1, 2, 3, 4, 5, 6...|[70, 80, 90]|2014-08-03 14:01:07|
+--------------------+--------------------+------------+--------------------+
```

Creating DataFrames using SQLContext

- **SQLContext can create dataframes directly from raw data**

In [92]:

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

In [93]:

```
sqlContext
```

Out[93]:

```
<pyspark.sql.context.SQLContext at 0x10d4f7438>
```

In [99]:

```
df = sqlContext.range(5)
df
```

Out[99]:

```
DataFrame[id: bigint]
```

In [100]:

```
df.show()
```

```
+---+
| id|
+---+
|  0|
|  1|
|  2|
|  3|
|  4|
+---+
```

In [101]:

```
df.count()
```

Out[101]:

```
5
```

Rows specified in tuples

In [104]:

```
data = [('Alice', 50),
        ('Bob', 80),
        ('Charlee', 75)]
```

In [105]:

```
sqlContext.createDataFrame(data).show()
```

```
+-------+---+
|     _1| _2|
+-------+---+
|  Alice| 50|
|    Bob| 80|
|Charlee| 75|
+-------+---+
```

In [106]:

```
sqlContext.createDataFrame(data, ['Name', 'Score']).show()
```

```
+-------+-----+
|   Name|Score|
+-------+-----+
|  Alice|   50|
|    Bob|   80|
|Charlee|   75|
+-------+-----+
```

In [200]:

```
complex_data = [
                  (1.0,
                   10,
                   "Alice",
                   True,
                   [1, 2, 3],
                   {"k1": 0},
                   Row(a=1, b=2, c=3),
                   datetime(2014, 8, 1, 14, 1, 5)),

                  (2.0,
                   20,
                   "Bob",
                   True,
                   [1, 2, 3, 4, 5],
                   {"k1": 0,"k2": 1 },
                   Row(a=1, b=2, c=3),
                   datetime(2014, 8, 1, 14, 1, 5)),

                   (3.0,
                    30,
                    "Charlee",
                    False,
                    [1, 2, 3, 4, 5, 6],
                    {"k1": 0, "k2": 1, "k3": 2 },
                    Row(a=1, b=2, c=3),
                    datetime(2014, 8, 1, 14, 1, 5))
                ]
```

In [201]:

```
sqlContext.createDataFrame(complex_data).show()
```

```
+---+---+-------+-----+-----------------+--------------------+---------+--------------
-----+
| _1| _2|     _3|   _4|               _5|                  _6|       _7|
_8|
+---+---+-------+-----+-----------------+--------------------+---------+--------------
-----+
|1.0| 10|  Alice| true|          [1, 2, 3]|           [k1 -> 0]||[1, 2, 3]||2014-08-01 14:01
:05|
|2.0| 20|    Bob| true|    [1, 2, 3, 4, 5]|  [k1 -> 0, k2 -> 1]||[1, 2, 3]||2014-08-01 14:01
:05|
|3.0| 30|Charlee|false|[1, 2, 3, 4, 5, 6]||[k3 -> 2, k1 -> 0...|[1, 2, 3]||2014-08-01 14:01
:05|
+---+---+-------+-----+-----------------+--------------------+---------+--------------
-----+
```

In [202]:

```
complex_data_df = sqlContext.createDataFrame(complex_data, [
        'col_integer',
        'col_float',
        'col_string',
        'col_boolean',
        'col_list',
        'col_dictionary',
        'col_row',
        'col_date_time']
```

```
    )
complex_data_df.show()
```

```
+-----------+---------+----------+-----------+-----------------+--------------------+-
--------+--------------------+
|col_integer|col_float|col_string|col_boolean|         col_list|      col_dictionary|  c
ol_row|      col_date_time|
+-----------+---------+----------+-----------+-----------------+--------------------+-
--------+--------------------+
|        1.0|       10|     Alice|       true|        [1, 2, 3]|           [k1 -> 0]|[1
, 2, 3]|2014-08-01 14:01:05|
|        2.0|       20|       Bob|       true|   [1, 2, 3, 4, 5]|  [k1 -> 0, k2 -> 1]|[1,
2, 3]|2014-08-01 14:01:05|
|        3.0|       30|   Charlee|      false|[1, 2, 3, 4, 5, 6]|[k3 -> 2, k1 -> 0...|[1,
2, 3]|2014-08-01 14:01:05|
+-----------+---------+----------+-----------+-----------------+--------------------+-
--------+--------------------+
```

Creating dataframes using SQL Context and the Row function

- **Row functions can be used without specifying column names**

In [203]:

```
data = sc.parallelize([
    Row(1, "Alice", 50),
    Row(2, "Bob", 80),
    Row(3, "Charlee", 75)
])
```

In [204]:

```
column_names = Row('id', 'name', 'score')
students = data.map(lambda r: column_names(*r))
```

In [205]:

```
students
```

Out[205]:

```
PythonRDD[570] at RDD at PythonRDD.scala:48
```

In [206]:

```
students.collect()
```

Out[206]:

```
[Row(id=1, name='Alice', score=50),
 Row(id=2, name='Bob', score=80),
 Row(id=3, name='Charlee', score=75)]
```

In [207]:

```
students_df = sqlContext.createDataFrame(students)
students_df
```

Out[207]:

```
DataFrame[id: bigint, name: string, score: bigint]
```

In [208]:

```
students_df.show()
```

```
+---+-------+-----+
| id|   name|score|
+---+-------+-----+
|  1|  Alice|   50|
```

```
|  2|    Bob|   80|
|  3|Charlee|   75|
+---+-------+-----+
```

Extracting specific rows from dataframes

In [209]:

```
complex_data_df.first()
```

Out[209]:

```
Row(col_integer=1.0, col_float=10, col_string='Alice', col_boolean=True, col_list=[1, 2,
3], col_dictionary={'k1': 0}, col_row=Row(a=1, b=2, c=3), col_date_time=datetime.datetime
(2014, 8, 1, 14, 1, 5))
```

In [210]:

```
complex_data_df.take(2)
```

Out[210]:

```
[Row(col_integer=1.0, col_float=10, col_string='Alice', col_boolean=True, col_list=[1, 2,
3], col_dictionary={'k1': 0}, col_row=Row(a=1, b=2, c=3), col_date_time=datetime.datetime
(2014, 8, 1, 14, 1, 5)),
 Row(col_integer=2.0, col_float=20, col_string='Bob', col_boolean=True, col_list=[1, 2, 3
, 4, 5], col_dictionary={'k1': 0, 'k2': 1}, col_row=Row(a=1, b=2, c=3), col_date_time=dat
etime.datetime(2014, 8, 1, 14, 1, 5))]
```

Extracting specific cells from dataframes

In [211]:

```
cell_string = complex_data_df.collect()[0][2]
cell_string
```

Out[211]:

```
'Alice'
```

In [212]:

```
cell_list = complex_data_df.collect()[0][4]
cell_list
```

Out[212]:

```
[1, 2, 3]
```

In [213]:

```
cell_list.append(100)
cell_list
```

Out[213]:

```
[1, 2, 3, 100]
```

In [214]:

```
complex_data_df.show()
```

```
+-----------+---------+----------+-----------+------------------+--------------------+-
--------+-------------------+
|col_integer|col_float|col_string|col_boolean|          col_list|      col_dictionary|  c
ol_row|      col_date_time|
+-----------+---------+----------+-----------+------------------+--------------------+-
--------+-------------------+
|        1.0|       10|     Alice|       true|         [1, 2, 3]|           [k1 -> 0]|[1
, 2, 3]|2014-08-01 14:01:05|
|        2.0|       20|       Bob|       true|[1, 2, 3, 4, 5]|[k1 -> 0, k2 -> 1]|[1
```

```
|        2.0|       20|        Bob|       true|   [1, 2, 3, 4, 5]|   [k1 -> 0, k2 -> 1]|[1,
2, 3]|2014-08-01 14:01:05|
|        3.0|       30|    Charlee|      false|[1, 2, 3, 4, 5, 6]|[k3 -> 2, k1 -> 0...|[1,
2, 3]|2014-08-01 14:01:05|
+-----------+---------+-----------+-----------+------------------+--------------------+-
--------+-------------------+
```

Selecting specific columns

In [215]:

```
complex_data_df.rdd\
    .map(lambda x: (x.col_string, x.col_dictionary))\
    .collect()
```

Out[215]:

```
[('Alice', {'k1': 0}),
 ('Bob', {'k1': 0, 'k2': 1}),
 ('Charlee', {'k1': 0, 'k2': 1, 'k3': 2})]
```

In [216]:

```
complex_data_df.select(
    'col_string',
    'col_list',
    'col_date_time'
).show()
```

```
+----------+------------------+--------------------+
|col_string|          col_list|      col_date_time|
+----------+------------------+--------------------+
|     Alice|         [1, 2, 3]|2014-08-01 14:01:05|
|       Bob|   [1, 2, 3, 4, 5]|2014-08-01 14:01:05|
|   Charlee|[1, 2, 3, 4, 5, 6]|2014-08-01 14:01:05|
+----------+------------------+--------------------+
```

Editing columns

In [217]:

```
complex_data_df.rdd\
           .map(lambda x: (x.col_string + " Boo"))\
           .collect()
```

Out[217]:

```
['Alice Boo', 'Bob Boo', 'Charlee Boo']
```

Adding a column

In [218]:

```
complex_data_df.select(
                    'col_integer',
                    'col_float'
            )\
           .withColumn(
                    "col_sum",
                     complex_data_df.col_integer + complex_data_df.col_float
           )\
           .show()
```

```
+-----------+---------+-------+
|col_integer|col_float|col_sum|
+-----------+---------+-------+
|        1.0|       10|   11.0|
|        2.0|       20|   22.0|
```

```
|         3.0|        30|    33.0|
+------------+----------+--------+
```

In [220]:

```
complex_data_df.select('col_boolean')\
               .withColumn(
                   "col_opposite",
                   complex_data_df.col_boolean == False )\
               .show()
```

```
+-----------+------------+
|col_boolean|col_opposite|
+-----------+------------+
|       true|       false|
|       true|       false|
|      false|        true|
+-----------+------------+
```

Editing a column name

In [225]:

```
complex_data_df.withColumnRenamed("col_dictionary","col_map").show()
```

```
+-----------+---------+----------+-----------+------------------+--------------------+-
--------+--------------------+
|col_integer|col_float|col_string|col_boolean|          col_list|             col_map|
col_row|      col_date_time|
+-----------+---------+----------+-----------+------------------+--------------------+-
--------+--------------------+
|        1.0|       10|     Alice|       true|         [1, 2, 3]|           [k1 -> 0]|[1
, 2, 3]|2014-08-01 14:01:05|
|        2.0|       20|       Bob|       true|   [1, 2, 3, 4, 5]|  [k1 -> 0, k2 -> 1]|[1,
2, 3]|2014-08-01 14:01:05|
|        3.0|       30|   Charlee|      false|[1, 2, 3, 4, 5, 6]|[k3 -> 2, k1 -> 0...|[1,
2, 3]|2014-08-01 14:01:05|
+-----------+---------+----------+-----------+------------------+--------------------+-
--------+--------------------+
```

In [226]:

```
complex_data_df.select(complex_data_df.col_string.alias("Name")).show()
```

```
+-------+
|   Name|
+-------+
|  Alice|
|    Bob|
|Charlee|
+-------+
```

Interoperablity between Pandas dataframe and Spark dataframe

In [232]:

```
import pandas
```

In [234]:

```
df_pandas = complex_data_df.toPandas()
df_pandas
```

Out[234]:

	col_integer	col_float	col_string	col_boolean	col_list	col_dictionary	col_row	col_date_time
0	1.0	10	Alice	True	[1, 2, 3]	{'k1': 0}	(1, 2, 3)	2014-08-01 14:01:05
1	2.0	20	Bob	True	[1, 2, 3, 4, 5]	{'k1': 0, 'k2': 1}	(1, 2, 3)	2014-08-01 14:01:05
2	3.0	30	Charlee	False	[1, 2, 3, 4, 5, 6]	{'k3': 2, 'k1': 0, 'k2': 1}	(1, 2, 3)	2014-08-01 14:01:05

In [235]:

```
df_spark = sqlContext.createDataFrame(df_pandas).show()
df_spark
```

```
+-----------+---------+----------+-----------+-----------------+--------------------+-
--------+-------------------+
|col_integer|col_float|col_string|col_boolean|         col_list|      col_dictionary|  c
ol_row|      col_date_time|
+-----------+---------+----------+-----------+-----------------+--------------------+-
--------+-------------------+
|        1.0|       10|     Alice|       true|          [1, 2, 3]|           [k1 -> 0]||[1
, 2, 3]|2014-08-01 14:01:05|
|        2.0|       20|       Bob|       true|   [1, 2, 3, 4, 5]|  [k1 -> 0, k2 -> 1]||[1,
2, 3]|2014-08-01 14:01:05|
|        3.0|       30|   Charlee|      false|[1, 2, 3, 4, 5, 6]||[k3 -> 2, k1 -> 0...|[1,
2, 3]|2014-08-01 14:01:05|
+-----------+---------+----------+-----------+-----------------+--------------------+-
--------+-------------------+
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: