Yandex

Joins

Sample data

› Financial data from http://finance.yahoo.com for NASDAQ
› Stored in the CSV format in the file 'nasdaq.csv'

```
2017-01-03,5425.620117,5452.569824,5397.990234,5429.080078,5429.080078,1886200000
2017-01-04,5440.910156,5482.350098,5440.240234,5477.000000,5477.000000,1883360000
2017-01-05,5474.390137,5495.850098,5464.359863,5487.939941,5487.939941,1792610000
2017-01-06,5499.080078,5536.520020,5482.810059,5521.060059,5521.060059,1710770000
2017-01-09,5527.580078,5541.080078,5517.140137,5531.819824,5531.819824,1885500000
2017-01-10,5536.540039,5564.250000,5528.109863,5551.819824,5551.819824,1796500000
2017-01-11,5550.720215,5564.080078,5524.029785,5563.649902,5563.649902,1954720000
2017-01-12,5542.560059,5550.669922,5496.819824,5547.490234,5547.490234,1801750000
2017-01-13,5557.569824,5584.259766,5557.200195,5574.120117,5574.120117,1605110000
2017-01-17,5555.160156,5557.049805,5527.220215,5538.729980,5538.729980,1757030000
...
```

$ pyspark

```
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                         "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>>
```

$ pyspark

```
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                         "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>>
```

```
Date: 2017-01-03, Close: 5429.080078
Date: 2017-01-04, Close: 5477.000000, Return: 0.88% = (5477.000000 / 5429.080078 – 1) *
100%
Date: 2017-01-05, Close: 5487.939941, Return: 0.20% = (5487.939941 / 5477.000000 – 1) *
100%
...
Return[i] = (Close[i] / Close[i-1] – 1) * 100%
```

$ pyspark

```
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                        "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> date_and_close_price = parsed_data.map(lambda r: (r.date, r.close))
>>> date_and_close_price.take(3)
[('2017-01-03', 5429.080078),
 ('2017-01-04', 5477.0),
 ('2017-01-05', 5487.939941)]
>>>
```

$ pyspark

```
>>> from collections import namedtuple
>>> Record = namedtuple("Record", ["date", "open", "high", "low", "close",
                        "adj_close", "volume"])
>>> def parse_record(s):
...   fields = s.split(",")
...   return Record(fields[0], *map(float, fields[1:6]), int(fields[6]))
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> date_and_close_price = parsed_data.map(lambda r: (r.date, r.close))
>>> date_and_close_price.take(3)
[('2017-01-03', 5429.080078),
 ('2017-01-04', 5477.0),
 ('2017-01-05', 5487.939941)]
>>> from datetime import datetime, timedelta
>>> def get_next_date(s):
...   fmt = "%Y-%m-%d"
...   return (datetime.strptime(s, fmt) + timedelta(days=1)).strftime(fmt)
...
>>> get_next_date("2017-01-03")
'2017-01-04'
>>>
```

$ pyspark

```
...
>>> from datetime import datetime, timedelta
>>> def get_next_date(s):
...   fmt = "%Y-%m-%d"
...   return (datetime.strptime(s, fmt) + timedelta(days=1)).strftime(fmt)
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> date_and_close_price = parsed_data.map(lambda r: (r.date, r.close))
>>> date_and_close_price.take(3)
[('2017-01-03', 5429.080078),
 ('2017-01-04', 5477.0),
 ('2017-01-05', 5487.939941)]
>>>
```

$ pyspark

```
...
>>> from datetime import datetime, timedelta
>>> def get_next_date(s):
...   fmt = "%Y-%m-%d"
...   return (datetime.strptime(s, fmt) + timedelta(days=1)).strftime(fmt)
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> date_and_close_price = parsed_data.map(lambda r: (r.date, r.close))
>>> date_and_close_price.take(3)
[('2017-01-03', 5429.080078),
 ('2017-01-04', 5477.0),
 ('2017-01-05', 5487.939941)]
>>> date_and_prev_close_price = parsed_data.map(lambda r: (get_next_date(r.date), r.close))
>>> date_and_prev_close_price.take(3)
[('2017-01-04', 5429.080078),
 ('2017-01-05', 5477.0),
 ('2017-01-06', 5487.939941)]
>>>
```

$ pyspark

```
...
>>> from datetime import datetime, timedelta
>>> def get_next_date(s):
...   fmt = "%Y-%m-%d"
...   return (datetime.strptime(s, fmt) + timedelta(days=1)).strftime(fmt)
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> date_and_close_price = parsed_data.map(lambda r: (r.date, r.close))
>>> date_and_close_price.take(3)
[('2017-01-03', 5429.080078
 ('2017-01-04', 5477.0),
 ('2017-01-05', 5487.939941)]
>>> date_and_prev_close_price = parsed_data.map(lambda r: (get_next_date(r.date), r.close))
>>> date_and_prev_close_price.take(3)
[('2017-01-04', 5429.080078),
 ('2017-01-05', 5477.0),
 ('2017-01-06', 5487.939941)]
>>>
```

Return = Close price / Previous close price

$ pyspark

```
...
>>> from datetime import datetime, timedelta
>>> def get_next_date(s):
...   fmt = "%Y-%m-%d"
...   return (datetime.strptime(s, fmt) + timedelta(days=1)).strftime(fmt)
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> date_and_close_price = parsed_data.map(lambda r: (r.date, r.close))
>>> date_and_close_price.take(3)
[('2017-01-03'
 ('2017-01-04
 ('2017-01-05'
>>> date_and                                                        e), r.close))
>>> date_and_prev_close_price.take(3)
[('2017-01-04', 5429.080078),
 ('2017-01-05', 5477.0),
 ('2017-01-06', 5487.939941)]
>>> joined = date_and_close_price.join(date_and_prev_close_price)
>>>
```

Def: X.join(Y: RDD[(K, W)]): RDD[(K, V)] ➡ RDD[(K, V, W)]
given two keyed RDDs, returns all matching items in two datasets that are triples (k, x, y) where (k, x) is in X and (k, y) is in Y

$ pyspark

```
...
>>> from datetime import datetime, timedelta
>>> def get_next_date(s):
...   fmt = "%Y-%m-%d"
...   return (datetime.strptime(s, fmt) + timedelta(days=1)).strftime(fmt)
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> date_and_close_price = parsed_data.map(lambda r: (r.date, r.close))
>>> date_and_close_price.take(3)
[('2017-01-03', 5429.080078),
 ('2017-01-04', 5477.0),
 ('2017-01-05', 5487.939941)]
>>> date_and_prev_close_price = parsed_data.map(lambda r: (get_next_date(r.date), r.close))
>>> date_and_prev_close_price.take(3)
[('2017-01-04', 5429.080078),
 ('2017-01-05', 5477.0),
 ('2017-01-06', 5487.939941)]
>>> joined = date_and_close_price.join(date_and_prev_close_price)
>>> joined.lookup("2017-01-04")
[(5477.0, 5429.080078)]
>>>
```

$ pyspark

```
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> date_and_close_price = parsed_data.map(lambda r: (r.date, r.close))
>>> date_and_prev_close_price = parsed_data.map(lambda r: (get_next_date(r.date), r.close))
>>> joined = date_and_close_price.join(date_and_prev_close_price)
>>>
```

$ pyspark

```
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> date_and_close_price = parsed_data.map(lambda r: (r.date, r.close))
>>> date_and_prev_close_price = parsed_data.map(lambda r: (get_next_date(r.date), r.close))
>>> joined = date_and_close_price.join(date_and_prev_close_price)
>>> returns = joined.mapValues(lambda p: (p[0] / p[1] - 1.0) * 100.0)
>>> returns.lookup('2017-01-04')
[0.8826527019592856]
>>> returns.sortByKey().take(3)
[('2017-01-04', 0.8826527019592856),
 ('2017-01-05', 0.19974330838048449),
 ('2017-01-06', 0.6035072970198341)]
>>>
```

$ pyspark

```
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> date_and_close_price = parsed_data.map(lambda r: (r.date, r.close))
>>> date_and_prev_close_price = parsed_data.map(lambda r: (get_next_date(r.date), r.close))
>>> joined = date_and_close_price.join(date_and_prev_close_price)
>>> returns = joined.mapValues(lambda p: (p[0] / p[1] - 1.0) * 100.0)
>>> returns.lookup('2017-01-04')
[0.8826527019592856]
>>> returns.sortByKey().take(3)
[('2017-01-04', 0.8826527019592856),
 ('2017-01-05', 0.19974330838048449),
 ('2017-01-06', 0.6035072970198341)]
>>> date_and_close_price.lookup("2017-01-03")
[5429.080078]
>>> returns.lookup("2017-01-03")
[]
>>>
```

$ pyspark

```
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> date_and_close_price = parsed_data.map(lambda r: (r.date, r.close))
>>> date_and_prev_close_price = parsed_data.map(lambda r: (get_next_date(r.date), r.close))
>>> joined = date_and_close_price.join(date_and_prev_close_price)
>>> joined_left = date_and_close_price.leftOuterJoin(date_and_prev_close_price)
>>> joined.lookup("2017-01-03")
[]
>>> joined_left.lookup("2017-01-03")
[(5429.080078, None)]
>>>
```

$ pyspark

```
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> date_and_close_price = parsed_data.map(lambda r: (r.date, r.close))
>>> date_and_prev_close_price = parsed_data.map(lambda r: (get_next_date(r.date), r.close))
>>> joined = date_and_close_price.join(date_and_prev_close_price)
>>> joined_left = date_and_close_price.leftOuterJoin(date_and_prev_close_price)
>>> joined_right = date_and_close_price.rightOuterJoin(date_and_prev_close_price)
>>> joined.lookup("2017-01-03")
[]
>>> joined_left.lookup("2017-01-03")
[(5429.080078, None)]
>>> joined_right.lookup("2017-01-03")
[]
>>> joined.lookup("2017-07-22")
[]
>>> joined_left.lookup("2017-07-22")
[]
>>> joined_right.lookup("2017-07-22")
[(None, 6387.75)]
>>>
```

$ pyspark

```
...
>>> parsed_data = sc.textFile("nasdaq.csv").map(parse_record).cache()
>>> date_and_close_price = parsed_data.map(lambda r: (r.date, r.close))
>>> date_and_prev_close_price = parsed_data.map(lambda r: (get_next_date(r.date), r.close))
>>> joined = date_and_close_price.join(date_and_prev_close_price)
>>> joined_left = date_and_close_price.leftOuterJoin(date_and_prev_close_price)
>>> joined_right = date_and_close_price.rightOuterJoin(date_and_prev_close_price)
>>> joined_full = date_and_close_price.fullOuterJoin(date_and_prev_close_price)
>>> joined_full.lookup("2017-01-03")
[(5429.080078, None)]
>>> joined_full.lookup("2017-07-22")
[(None, 6387.75)]
>>>
```

Summary

- You have learned how to:
  - compute "lagged" time series to compute daily returns
  - join datasets with Spark
  - differentiate between inner, left, right and full outer joins
  - use the 'lookup' action to explore a particular key in a dataset

BigDATAteam