Spark SQL

Connection to Metacentrum cluster

```
ssh username@hador.ics.muni.cz
```

PySpark launching

```
pyspark --master yarn --num-executors 2 --executor-memory 4G --packages com.databricks:spark-csv_2.10:1.5.0
```

Spark SQL in simple examples

Try out examples. They help you get familiar with Spark SQL commands.

You find more in Manual Spark SQL

```
# useful import
from pyspark.sql import functions as F

### reading DataFrame from Hive database and caching
Tep_DF = sqlContext.sql('select * from fel_bigdata.teplota').cache()
```

Getting basic information on DataFrame

```
Tep_DF.show() # prints sample of data as DataFrame
Tep_DF.take(5) # prints sample of data as RDD

Tep_DF.count()

Tep_DF.printSchema()
```

Columns and rows selecting

```
### columns
Tep_DF2 = Tep_DF.select('stat', 'mesic', 'teplota')
Tep_DF2.show()

### rows - different ways
Tep_DF2 = Tep_DF.filter(Tep_DF['stat']=='TX')
Tep_DF2.show()

Tep_DF2 = Tep_DF.filter((Tep_DF['stat']=='TX') & (Tep_DF['mesic']==11))
# brackets are necessary
Tep_DF2.show()

# the same but the condition in an alternative syntax
Tep_DF2 = Tep_DF.filter('stat="TX"')
Tep_DF2.show()

Tep_DF2 = Tep_DF.filter('stat="TX" and mesic=11')
Tep_DF2.show()

# RDD syntax
Tep_DF.filter(lambda r: r[9]=='TX' and r[1]==11).take(5)
# not working, DataFrame syntax expected
Tep_DF.rdd.filter(lambda r: r[9]=='TX' and r[1]==11).take(5)
# working after an explicit conversion to RDD

### unique values only
Tep_DF2 = Tep_DF.select('stanice', 'stat', 'nazev').distinct()
Tep_DF2.show()
```

Column transformation

```
# new column
Tep_DF2 = Tep_DF.withColumn('teplota_f', Tep_DF['teplota']*9.0/5.0 + 32)
Tep_DF2.show()
```

```
# overwriting existing column
Tep_DF2 = Tep_DF.withColumn('teplota', Tep_DF['teplota']*9.0/5.0 + 32)
Tep_DF2.show()

# function to be applied on each column member:
# --> Spark SQL column (vector) functions
# see http://spark.apache.org/docs/1.6.0/api/python/pyspark.sql.html#module-
pyspark.sql.functions
# from pyspark.sql import functions as F

Tep_DF2 = Tep_DF.withColumn('stanice2', F.lower(Tep_DF['stanice']))
Tep_DF2.show()

Tep_DF2 = Tep_DF.withColumn('nazev_delka', F.length(Tep_DF['nazev']))
Tep_DF2.show()

Tep_DF2 = Tep_DF.withColumn('nazev_slov', F.size(F.split(Tep_DF['nazev'], " ")))
Tep_DF2.show()

Tep_DF2 = Tep_DF.withColumn('nazev2', F.regexp_replace(Tep_DF['nazev'], 'A', '4'))
Tep_DF2.show()

Tep_DF2 = Tep_DF.withColumn('den2', F.when(Tep_DF['den']<=10, '1-10').otherwise('11-31'))
Tep_DF2.show()

### missing values
Tep_DF2 = Tep_DF.dropna() # drops rows containing any value null
Tep_DF2.show()

Tep_DF2 = Tep_DF.fillna(0, 'teplota')
Tep_DF2.show()

### columns renaming
# one column
Tep_DF2 = Tep_DF.withColumnRenamed('latitude', 'gps_lat')
Tep_DF2.show()

# all columns in a DataFrame
Tep_DF2 = Tep_DF.select('stat', 'latitude', 'longitude').toDF('usa_state', 'gps_lat',
'gps_long')
Tep_DF2.show()
```

Sorting, aggregation

```
### sorting
Tep_DF2 = Tep_DF.orderBy('latitude', ascending=False)
Tep_DF2.show()

Tep_DF2 = Tep_DF.orderBy(Tep_DF['latitude'].desc())
Tep_DF2.show()

### aggregation
Tep_DF2 = Tep_DF.groupBy('mesic').avg('teplota')
Tep_DF2.show()

# the same, more general aggregation syntax
Tep_DF2 = Tep_DF.groupBy('mesic').agg({'teplota': 'avg'})
Tep_DF2.show()

Tep_DF2 = Tep_DF.groupBy('stat').count()
Tep_DF2.show()

# overal aggregation
Tep_DF2 = Tep_DF.max('teplota') # not working
Tep_DF2 = Tep_DF.groupBy().max('teplota') # empty groupBy required
Tep_DF2.show()
```

Joining DataFrames

```
# another table - just to be joined
States = sqlContext.createDataFrame([('CA', 'California'), ('TX', 'Texas'), ('KY',
'Kentucky')], ('stat', 'stat_nazev'))
States.show()
```

```
# joining by fields with the same names in both DataFrames
Tep_DF2 = Tep_DF.join(States, 'stat')
Tep_DF2.show()

# joining by general condition - join field will be duplicated
Tep_DF2 = Tep_DF.join(States, Tep_DF['stat']==States['stat'])
Tep_DF2.show()
```