SPARK SQL

DataFrames and DataSets

Working with structured data

- Extends RDD to a "DataFrame" object
- DataFrames:
  - *Contain Row objects*
  - *Can run SQL queries*
  - *Has a schema (leading to more efficient storage)*
  - *Read and write to JSON, Hive, parquet*
  - *Communicates with JDBC/ODBC, Tableau*

Using SparkSQL in Python

- from pyspark.sql import SQLContext, Row
- hiveContext = HiveContext(sc)
- inputData = spark.read.json(dataFile)
- inputData.createOrReplaceTempView("myStructuredStuff")
- myResultDataFrame = hiveContext.sql("""SELECT foo FROM bar ORDER BY foobar""")

Other stuff you can do with dataframes

- myResultDataFrame.show()
- myResultDataFrame.select("someFieldName")
- myResultDataFrame.filter(myResultDataFrame("someFieldName" > 200)
- myResultDataFrame.groupBy(myResultDataFrame("someFieldName")).mean()
- myResultDataFrame.rdd().map(mapperFunction)

Datasets

- In Spark 2.0, a DataFrame is really a DataSet of Row objects
- DataSets can wrap known, typed data too. But this is mostly transparent to you in Python, since Python is dynamically typed.
- So – don’t sweat this too much with Python. But the Spark 2.0 way is to use DataSets instead of DataFrames when you can.

Shell access

- Spark SQL exposes a JDBC/ODBC server (if you built Spark with Hive support)
- Start it with sbin/start-thriftserver.sh
- Listens on port 10000 by default
- Connect using bin/beeline -u jdbc:hive2://localhost:10000
- Viola, you have a SQL shell to Spark SQL
- You can create new tables, or query existing ones that were cached using hiveCtx.cacheTable("tableName")

User-defined functions (UDF's)

```
from pyspark.sql.types import IntegerType
hiveCtx.registerFunction("square", lambda x: x*x, IntegerType())
df = hiveCtx.sql("SELECT square('someNumericFiled') FROM tableName)
```

YOUR CHALLENGE

Filter out movies with very few ratings

The problem

- Our examples of finding the lowest-rated movies were polluted with movies only rated by one or two people.
- Modify one or both of these scripts to only consider movies with at least ten ratings.

Hints

- RDD's have a filter() function you can use
  - *It takes a function as a parameter, which accepts the entire key/value pair*
    - So if you're calling filter() on an RDD that contains (movieID, (sumOfRatings, totalRatings)) – a lambda function that takes in "x" would refer to totalRatings as x[1][1]. x[1] gives us the "value" (sumOfRatings, totalRatings) and x[1][1] pulls out totalRatings.
  - *This function should be an expression that returns True if the row should be kept, or False if it should be discarded*
- DataFrames also have a filter() function
  - *It's easier – you just pass in a string expression for what you want to filter on.*
  - *For example: df.filter("count > 10") would only pass through rows where the "count" column is greater than 10.*

GOOD LUCK