**Univ.-Prof. Dr.-Ing. Matthias Boehm**
Graz University of Technology
Computer Science and Biomedical Engineering
Institute of Interactive Systems and Data Science
BMK endowed chair for Data Management

# 4. Data Management SS21: Exercise 04 – Large-Scale Data Analysis

**Published: May 29, 2021** (updates: N/A)
**Deadline: June 22, 2021, 11.59pm**

This exercise on large-scale data analysis aims to provide practical experience with distributed data management and large-scale data analysis on top of Apache Spark. The expected result is a zip archive named `DBExercise04_<student_ID>.zip`, submitted in TeachCenter. The entire exercise is *extra credit* for the course data management.

## 4.1. Apache Spark Setup (3/25 points)

As a preparation step, setup Apache Spark and necessary Hadoop client APIs inside an IDE (integrated development environment) of your language choice. This exercise can be done with the Spark language bindings Java, Scala, or Python. For example in Java, you include the maven dependencies `spark-core` and `spark-sql`. On Windows, please download `winutils.exe` from `https://github.com/steveloughran/winutils/tree/master/hadoop-2.7.1/bin`[1], put it into a directory `<some-path>/hadoop/bin`, and create an environment variable `HADOOP_HOME=<some-path>/hadoop`. The input data for this exercise is available at `https://mboehm7.github.io/teaching/ss21_dbs/All_data.zip` (from Ex 3, based on the schema from Ex 2).

**Partial Results:** N/A (every submission receives these points).

## 4.2. Query Processing via Spark RDDs (10/25 points)

Apache Spark's basic abstraction for distributed collections are so-called Resilient Distributed Datasets (RDDs). In this task, you should implement the query **Q07** (see Appendix A) from Task 2.3 via RDD operations, collect the results in the driver and print the result list to std-out. Please implement this query as a self-contained function/method `executeQ07RDD()` that internally creates a `SparkContext` sc, reads the files via `sc.textFile()`, and uses only RDD[2] operations to compute the query results.

**Partial Results:** Source file `QueryRDD.*`.

[1] The latest versions of precompiled `winutils.exe` can be found at `https://github.com/cdarlint/winutils`.
[2] `https://spark.apache.org/docs/latest/rdd-programming-guide.html`

### 4.3. Query Processing via Spark SQL (5/25 points)

Spark also provides the high-level APIs `Dataframe` and `Dataset` for SQL processing. In this task, you should implement queries **Q07** (see Appendix A) from Task 2.3 via Dataset operations, and write the outputs to JSON files `out07.json`. Please implement this query as a self-contained function/method `executeQ07Dataset()` that internally creates a `SparkSession` sc, reads the inputs files via `sc.read().format("csv")`, and uses only SQL or Dataset operations to compute and write the query results. You might either (1) register the individual input Datasets as temporary views and compute the results directly via SQL, or (2) alternatively use the functional API of Datasets. Both specifications share a common query optimization and processing pipeline.

**Partial Results:** Source file `QueryDataset.*`.

### 4.4. Medal Prediction (7 points)

Given the full Summer Olympics dataset from Exercises 2 and 3 (`https://mboehm7.github.io/teaching/ss21_dbs/All_data.zip`), create a regression model[3] for predicting the number of medals won by individual athletes at a specific instance of the games. The model should be trained with the data from 1896 through 2012, and tested with the data from 2016 (i.e., for all athletes participating in Rio 2016). Please write the data preparation and model training pipeline using existing or custom feature transformations and ML algorithms in Apache Spark. Finally, evaluate your trained model by computing the average residuals ($\frac{1}{N}\sum(\mathbf{y}-\hat{\mathbf{y}})$), sum of squared residuals ($\sum(\mathbf{y}-\hat{\mathbf{y}})^2$), and $R^2$ (coefficient of determination).

**Partial Results:** Source file `MLPipeline.*`.

## A. Query Q07 from Exercise 2

```
SELECT A.AKey, A.Name, A.DoB, count(*)
  FROM Athletes A, Results R
  WHERE A.AKey = R.AKey
    AND A.AKey NOT IN( -- not in medal winners
      SELECT DISTINCT A2.AKey FROM Athletes A2, Results R2
      WHERE A2.AKey = R2.AKey AND R2.Medal IS NOT NULL)
    AND R.Year BETWEEN 1948 AND 2016
  GROUP BY A.AKey
  ORDER BY count(*) DESC, A.Name ASC
  LIMIT 10
```

[3] This task aims to show the relationship of the course *Data Management* with the course *Introduction to Data Science and Artificial Intelligence* (in the module *Data Management and Data Science*).