Bielefeld University
Faculty of Technology

Genome Data Science
Dr. Daniel Dörr

Programming

Summer 2020

Exercises

Number 08, Submission Deadline: June 28, 2020

1. Database queries in MongoDB (4P)

Answer the following questions by querying the `books` collection with MongoDB / `pymongo`

1. How many books have been published in 2012? (1P)
2. Inform yourself on how to use aggregators in `pymongo` (https://api.mongodb.com/python/current/examples/aggregation.html). Then answer the following questions:
3. Which book has the most co-authors? (1P)
4. Who (co-) authored the most books? (1P)
5. How many books have been published in each category? (1P)

2. Database queries with Spark SQL API (6P)

The course material contains two connected data sets of companies and their employees. Query the data sets using Spark's SQL API to answer the following questions:

1. Select the first and the last name of all employees earn more than 6000 USD a month. How many are there? (1P)
2. What is the most common last name among employees? (1P)
3. Inform yourself about join operators (e.g. https://jaceklaskowski.gitbooks.io/mastering-spark-sql/spark-sql-joins.html) and use joing queries to answer the following questions:
   (a) What ist the minimal, mean and maximal salary among the employees for every company? Sort in descending order by of their mean salary. (2P)
   (b) Select all the names of the employees who are born after their company was founded. Sort in descending order of their age. (2P)

[1]:
```
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local[*]").getOrCreate()
```

[2]:
```
companies = spark.read.csv("course_material_08/companies.csv", header=True,␣
 ↪inferSchema=True)
companies.printSchema()
companies.limit(3).show()
companies.createOrReplaceTempView("companies")
```

```
root
 |-- id: integer (nullable = true)
 |-- name: string (nullable = true)
 |-- founded_at: timestamp (nullable = true)
```

```
+---+--------------------+-------------------+
| id|                name|         founded_at|
+---+--------------------+-------------------+
|  1|Larson, Berge and…|1990-10-09 00:00:00|
|  2|            Wiza PLC|1973-08-04 00:00:00|
|  3|    Watsica and Sons|1989-01-25 00:00:00|
+---+--------------------+-------------------+
```

[3]:
```
employees = spark.read.csv("data/employees.csv",header=True,inferSchema=True)
employees.printSchema()
employees.limit(3).show()
employees.createOrReplaceTempView("employees")
```

```
root
 |-- id: integer (nullable = true)
 |-- first_name: string (nullable = true)
 |-- last_name: string (nullable = true)
 |-- birthday: timestamp (nullable = true)
 |-- salary: integer (nullable = true)
 |-- company_id: integer (nullable = true)

+---+----------+---------+-------------------+------+----------+
| id|first_name|last_name|           birthday|salary|company_id|
+---+----------+---------+-------------------+------+----------+
|  1|   Charity|  Dickens|1975-11-23 00:00:00|  8559|         1|
|  2|    Maggie|  Langosh|1980-06-24 00:00:00|  8196|         2|
|  3|   Gilbert|  Hermann|1976-09-06 00:00:00|  4381|         3|
+---+----------+---------+-------------------+------+----------+
```

3. Spark Advanced Analytics: Prices of retail cars (8P)

Exercise Goal: Predict the prices of retail cars

1. Load dataset into spark
2. Transform String variable vendor (VM,BMW,…) into Index (0,1,…) (1P)
3. Use OneHotEncoding for index variable varibale (1P)
4. Transform feature into single feature vector (1P)
5. Fit a LinearRegressionModel to predict the price of the car. (1P)
6. How good does the model fit? Visualize and analyze your results. (2P)
7. Add the following new features and fit a new model and compare the results (2P):
   (a) age^2
   (b) consumption^2

Name / Measurement Unit / Description

vendor / – / vendor of the car
consumption / liter per 100 km / Fuel consumption
age / year / age of the car
price / € times 1000 / price of the car

[16]:
```
df = spark.read.csv('course_material_08/car.csv', header=True, inferSchema=True)
df.printSchema()
df.drop('vendor').describe().show()
```

```
root
 |-- vendor: string (nullable = true)
 |-- consumption: double (nullable = true)
 |-- age: integer (nullable = true)
 |-- price: double (nullable = true)

+-------+------------------+-----------------+------------------+
|summary|       consumption|              age|             price|
+-------+------------------+-----------------+------------------+
|  count|             10000|            10000|             10000|
|   mean| 7.037033000000019|           9.7118|25.940625999999952|
| stddev|1.9796448586268283|9.663874989256684|13.599931970531477|
|    min|             -0.02|                0|              1.01|
|    max|             14.27|               87|             92.33|
+-------+------------------+-----------------+------------------+
```