|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа,**

**обработки и интерпретации больших данных.**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 10 |

**Название:**

Spark

**Дисциплина:** Языки программирования для работы с большими данными

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-22М |  |  | Д.Ю.Хотин |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В.Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

|  |
| --- |
| import org.apache.spark.sql.{SparkSession, functions}  import org.apache.spark.sql.functions.\_  import org.apache.spark.sql.types.DoubleType  import org.apache.spark.sql.types.IntegerType  val spark = SparkSession.builder().appName("Russian Demography").getOrCreate()  val russianDemography = spark.read.option("header", "true").csv("./russian\_demography.csv")  // 1 Подсчитать общее количество записей в наборе данных:  val count = russianDemography.count()  println("Total number of records: " + count)  // 2 Подсчитать количество записей для каждого региона:  val regionCounts = russianDemography.groupBy("region").agg(functions.count("\*").alias("count"))  regionCounts.show()  // 3 Найти наибольшее и наименьшее значение столбца "npg":  val dfWithNPG = russianDemography.withColumn("npg", col("npg").cast(DoubleType))  val minNPG = dfWithNPG.agg(min("npg")).as[Double].first()  val maxNPG = dfWithNPG.agg(max("npg")).as[Double].first()  println("Minimum NPG: " + minNPG)  println("Maximum NPG: " + maxNPG)  // 4 Рассчитать среднее значение столбца "birth\_rate" по регионам:  val averageBirthRateByRegion = russianDemography.groupBy("region").agg(avg("birth\_rate").alias("average\_birth\_rate"))  averageBirthRateByRegion.show()  // 5 Найти регион с наибольшим и наименьшим значением столбца "death\_rate":  val maxDeathRateRegion = russianDemography.orderBy(desc("death\_rate")).select("region").first().getString(0)  val minDeathRateRegion = russianDemography.orderBy(asc("death\_rate")).select("region").first().getString(0)  println("Region with maximum death rate: " + maxDeathRateRegion)  println("Region with minimum death rate: " + minDeathRateRegion)  // 6 Рассчитать суммарное значение столбца "gdw" для каждого года:  val gdwSumByYear = russianDemography.groupBy("year").agg(sum("gdw").alias("total\_gdw"))  gdwSumByYear.show()  // 7 Найти год с наибольшим и наименьшим значением столбца "urbanization":  val dfWithYear = russianDemography.withColumn("year", col("year").cast(IntegerType))  val maxUrbanizationYear = dfWithYear.orderBy(desc("urbanization")).select("year").first().getInt(0)  println("Year with maximum urbanization: " + maxUrbanizationYear)  val minUrbanizationYear = dfWithYear.orderBy(asc("urbanization")).select("year").first().getInt(0)  println("Year with minimum urbanization: " + minUrbanizationYear)  // 8 Рассчитать среднее значение столбца "npg" для каждого года:  val averageNPGByYear = russianDemography.groupBy("year").agg(avg("npg").alias("average\_npg"))  averageNPGByYear.show()  // 9 Найти регион с наибольшим и наименьшим значением столбца "birth\_rate":  val maxBirthRateRegion = russianDemography.orderBy(desc("birth\_rate")).select("region").first().getString(0)  val minBirthRateRegion = russianDemography.orderBy(asc("birth\_rate")).select("region").first().getString(0)  println("Region with maximum birth rate: " + maxBirthRateRegion)  println("Region with minimum birth rate: " + minBirthRateRegion)  // 10 Рассчитать среднее значение столбца "death\_rate" для каждого года:  val averageDeathRateByYear = russianDemography.groupBy("year").agg(avg("death\_rate").alias("average\_death\_rate"))  averageDeathRateByYear.show() |