

山东大学 计算机科学与技术 学院

大数据分析实践 课程实验报告

学号: 202300130153	姓名: 吴宇轩	班级: 数据 23																																										
实验题目: SPARK实践																																												
实验学时: 2		实验日期: 2025.12.12																																										
实验目的: 本实验旨在介绍学习者如何配置和运行Apache Spark, 以及如何使用Spark进行简单的数据处理和分析。实验将涵盖以下内容:																																												
<ol style="list-style-type: none">安装和配置Apache Spark。运行一个简单的Spark应用程序, 以理解Spark的基本概念。使用Spark进行数据处理和分析。																																												
实验环境: Apache Spark、Java																																												
实验步骤与内容: 1、安装和配置Apache Spark 2、运行一个简单的Spark应用程序 从销售数据中统计衣物商品每日的总营收																																												
代码: <pre>public static void main(String[] args) { // 1. 初始化 SparkSession SparkSession spark = SparkSession.builder() .appName("SimpleSparkApp") .master("local[*]") .getOrCreate(); // 2. 加载 CSV 数据 (开启 header + 自动推断 Schema) Dataset<Row> data = spark.read() .option("header", "true") .option("inferSchema", "true") .csv("/home/hadoop/Desktop/sales_data.csv"); // 数据文件路径 // 3. 数据处理 Dataset<Row> result = data // 过滤 .filter(col("product_category").equalTo("Clothing")) // 按 date 分组 .groupBy("date") // 求和 revenue .agg(sum("revenue").alias("sum_revenue")); // 4. 显示结果 result.show(); // 5. 关闭 SparkSession spark.stop(); }</pre>																																												
运行结果: <table border="1"><thead><tr><th>date</th><th>sum_revenue</th></tr></thead><tbody><tr><td>2016/5/12</td><td>13912</td></tr><tr><td>2015/7/29</td><td>6498</td></tr><tr><td>2015/11/24</td><td>13714</td></tr><tr><td>2013/8/2</td><td>8123</td></tr><tr><td>2013/11/10</td><td>12952</td></tr><tr><td>2015/12/23</td><td>19417</td></tr><tr><td>2016/3/22</td><td>9150</td></tr><tr><td>2014/2/28</td><td>6932</td></tr><tr><td>2013/11/3</td><td>17722</td></tr><tr><td>2015/12/19</td><td>12274</td></tr><tr><td>2014/1/4</td><td>11336</td></tr><tr><td>2016/4/25</td><td>10335</td></tr><tr><td>2014/2/24</td><td>7127</td></tr><tr><td>2016/4/17</td><td>10842</td></tr><tr><td>2015/10/8</td><td>16996</td></tr><tr><td>2015/10/5</td><td>12079</td></tr><tr><td>2016/6/20</td><td>12048</td></tr><tr><td>2014/3/29</td><td>4793</td></tr><tr><td>2016/2/4</td><td>6060</td></tr><tr><td>2014/6/17</td><td>11842</td></tr></tbody></table> <p>only showing top 20 rows</p>			date	sum_revenue	2016/5/12	13912	2015/7/29	6498	2015/11/24	13714	2013/8/2	8123	2013/11/10	12952	2015/12/23	19417	2016/3/22	9150	2014/2/28	6932	2013/11/3	17722	2015/12/19	12274	2014/1/4	11336	2016/4/25	10335	2014/2/24	7127	2016/4/17	10842	2015/10/8	16996	2015/10/5	12079	2016/6/20	12048	2014/3/29	4793	2016/2/4	6060	2014/6/17	11842
date	sum_revenue																																											
2016/5/12	13912																																											
2015/7/29	6498																																											
2015/11/24	13714																																											
2013/8/2	8123																																											
2013/11/10	12952																																											
2015/12/23	19417																																											
2016/3/22	9150																																											
2014/2/28	6932																																											
2013/11/3	17722																																											
2015/12/19	12274																																											
2014/1/4	11336																																											
2016/4/25	10335																																											
2014/2/24	7127																																											
2016/4/17	10842																																											
2015/10/8	16996																																											
2015/10/5	12079																																											
2016/6/20	12048																																											
2014/3/29	4793																																											
2016/2/4	6060																																											
2014/6/17	11842																																											

3、使用Spark进行数据处理和分析

word count进行词频统计

代码：

```
public static void main(String[] args) {
    // 1. 初始化SparkSession
    SparkSession spark = SparkSession.builder()
        .appName("JavaSparkWordCountAdvanced")
        .master("local[*]")
        .getOrCreate();
    // 2. 读取文本文件
    String inputPath = "/home/hadoop/Desktop/input.txt";
    JavaRDD<String> lines = spark.read().textFile(inputPath).javaRDD();
    // 3. 数据预处理 (清洗+分词+过滤停用词)
    JavaRDD<String> words = lines
        // 去除首尾空格
        .map(String::trim)
        // 过滤空行
        .filter(line -> !line.isEmpty())
        // 转小写
        .map(String::toLowerCase)
        // 分词
        .flatMap(line -> Arrays.asList(WORD_PATTERN.split(line)).iterator())
        // 过滤空字符串 + 过滤停用词 + 过滤单字符无意义词
        .filter(word -> !word.isEmpty()
            && !STOP_WORDS.contains(word)
            && word.length() > 1);
    // 4. 词频统计 + 降序排序
    JavaPairRDD<String, Integer> sortedWordCounts = words
        .mapToPair(word -> new Tuple2<>(word, 1))
        .reduceByKey(Integer::sum)
        .mapToPair(Tuple2::swap)
        .sortByKey(false)
        .mapToPair(Tuple2::swap);
    // 5. 输出结果
    System.out.println("过滤停用词后的词频统计 (前10) :");
    sortedWordCounts.take(10).foreach(tuple ->
        System.out.println(tuple._1() + ": " + tuple._2()));
}
```

运行结果：

```
spark: 2
hello: 2
java: 2
learn: 1
big: 1
data: 1
framework: 1
easy: 1
```

求工作日销售额最高的产品类别

代码：

```
public static void main(String[] args) {
    // 1. 初始化SparkSession
    SparkSession spark = SparkSession.builder()
        .appName("MaxWeekdayRevenueCategory")
        .master("local[*]")
        .getOrCreate();

    // 2. 读取销售数据，解析日期
    Dataset<Row> df = spark.read()
        .option("header", "true")
        .option("inferSchema", "true")
        .csv("/home/hadoop/Desktop/sales_data.csv")
        // 统一日期格式
        .withColumn("date_parsed", to_date(col("Date"), "yyyy/M/d"));

    // 3. 筛选工作日数据 (周一=2 ~ 周五=6)
    Dataset<Row> weekdayDF = df.filter(dayofweek(col("date_parsed")).between(2, 6));

    // 4. 按产品类别聚合工作日总销售额
    Dataset<Row> categoryRevenueDF = weekdayDF.groupBy("Product_Category")
        .agg(sum("Revenue").alias("total_weekday_revenue"));

    categoryRevenueDF.orderBy(desc("total_weekday_revenue")).show();

    spark.stop();
}
```

运行结果：

Product_Category	total_weekday_revenue
Bikes	43698486
Accessories	10663199
Clothing	6015432

求每个产品子类的日均销售额

代码：

```

public static void main(String[] args) {
    // 1. 初始化SparkSession
    SparkSession spark = SparkSession.builder()
        .appName("DailyAvgRevenueBySubCategory")
        .master("local[*]")
        .getOrCreate();

    // 2. 读取数据并解析日期
    Dataset<Row> df = spark.read()
        .option("header", "true")
        .option("inferSchema", "true")
        .csv("/home/hadoop/Desktop/sales_data.csv")
        .withColumn("date_parsed", to_date(col("Date"), "yyyy/M/d"));

    // 3. 按产品子类分组，统计总销售额、总营业天数
    Dataset<Row> subcategoryStatsDF = df.groupBy("Sub_Category")
        .agg(
            sum("Revenue").alias("total_revenue"),
            countDistinct("date_parsed").alias("total_days")
        );

    // 4. 计算日均销售额（保留2位小数）
    Dataset<Row> dailyAvgRevenueDF = subcategoryStatsDF.withColumn(
        "daily_avg_revenue",
        round(col("total_revenue").divide(col("total_days")), 2)
    );

    // 5. 展示结果（降序）
    dailyAvgRevenueDF.orderBy(desc("daily_avg_revenue")).show();

    spark.stop();
}

```

运行结果：

Sub_Category	total_revenue	total_days	daily_avg_revenue
Road Bikes	33363061	1814	18391.99
Mountain Bikes	21123526	1644	12848.86
Touring Bikes	7295547	726	10048.96
Helmets	5741081	792	7248.84
Tires and Tubes	4670902	792	5897.6
Jerseys	4113742	786	5233.77
Shorts	1740710	676	2575.01
Bottles and Cages	1409174	790	1783.76
Vests	949063	534	1777.27
Fenders	1245733	774	1609.47
Hydration Packs	990406	624	1587.19
Bike Racks	517800	410	1262.93
Gloves	871419	740	1177.59
Bike Stands	344075	350	983.07
Caps	548777	782	701.76
Cleaners	198821	692	287.31
Socks	147171	566	260.02

结论分析与体会：

本次实验围绕Apache Spark的环境配置与数据处理分析展开，成功完成了所有预设任务，验证了Spark在结构化数据和非结构化数据处理中的高效性，认识到Spark的核心优势：一是数据处理效率，通过RDD的惰性求值与分布式计算快速响应；二是API的灵活性，无论是结构化数据的Dataset操作，还是非结构化数据的RDD处理，均能通过简洁API实现复杂逻辑。

