**作业二**

**任务：**

**1.用Mapper和Reducer汇总迪拜机场免税店销售数据每天的销售金额**

**2.用Mapper和Reducer汇总迪拜机场免税店三个航站楼每天的销售金额**

**3.用过滤出购买两次的顾客数据**

**MAP阶段代码**

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class MultiTaskMapper extends Mapper<LongWritable, Text, Text, Text> {

private Text outKey = new Text();

private Text outValue = new Text();

@Override

protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

// 跳过标题行

if (key.get() == 0 && value.toString().contains("city\_of\_airport")) {

return;

}

String line = value.toString();

String[] fields = line.split(",", -1); // 使用-1保持所有字段

// 确保字段数量足够

if (fields.length >= 41) {

String trx\_datime\_c1 = fields[6].trim();

String trx\_datime\_c2 = fields[7].trim();

String has\_c2 = fields[15].trim();

String exp\_c1 = fields[16].trim();

String exp\_c2 = fields[17].trim();

String concourse\_c1 = fields[20].trim();

String concourse\_c2 = fields[21].trim();

// 计算总花费

double expense1 = parseDouble(exp\_c1);

double expense2 = parseDouble(exp\_c2);

double totalExpense = expense1 + expense2;

// 提取日期并写入上下文

writeDateExpenses(trx\_datime\_c1, totalExpense, "Task1");

writeDateExpenses(trx\_datime\_c2, totalExpense, "Task1");

// 提取航站楼信息并写入上下文

writeConcourseExpenses(concourse\_c1, trx\_datime\_c1, totalExpense);

writeConcourseExpenses(concourse\_c2, trx\_datime\_c2, totalExpense);

// 处理 has\_c2 为 TRUE 的情况

if ("TRUE".equalsIgnoreCase(has\_c2)) {

outKey.set("Task3");

outValue.set(line);

context.write(outKey, outValue);

}

}

}

// 提取并写入日期花费

private void writeDateExpenses(String trxDateTime, double totalExpense, String taskPrefix) throws IOException, InterruptedException {

String date = trxDateTime.split(" ")[0]; // 获取日期部分

if (!date.isEmpty()) {

outKey.set(taskPrefix + "\_" + date);

outValue.set(String.valueOf(totalExpense));

context.write(outKey, outValue);

}

}

// 提取并写入航站楼花费

private void writeConcourseExpenses(String concourse, String trxDateTime, double totalExpense) throws IOException, InterruptedException {

if (!concourse.isEmpty()) {

String date = trxDateTime.split(" ")[0]; // 获取日期部分

if (!date.isEmpty()) {

outKey.set("Task2\_" + concourse + "\_" + date);

outValue.set(String.valueOf(totalExpense));

context.write(outKey, outValue);

}

}

}

// 安全解析字符串为双精度数

private double parseDouble(String str) {

try {

return Double.parseDouble(str);

} catch (NumberFormatException e) {

return 0.0; // 返回0以避免异常

}

}

}

**REDUCER阶段代码**

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.output.MultipleOutputs;

import java.io.IOException;

public class MultiTaskReducer extends Reducer<Text, Text, NullWritable, Text> {

private Text result = new Text();

private MultipleOutputs<NullWritable, Text> multipleOutputs;

@Override

protected void setup(Context context) {

multipleOutputs = new MultipleOutputs<>(context); // 初始化MultipleOutputs

}

@Override

protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

String keyStr = key.toString();

double sumExpense = 0.0;

// 处理任务一：汇总每天的花费总和

if (keyStr.startsWith("Task1\_")) {

for (Text val : values) {

sumExpense += parseDouble(val.toString());

}

String date = keyStr.substring(6); // 获取日期

String output = "日期: " + date + ", 总花费: " + sumExpense;

result.set(output);

multipleOutputs.write(NullWritable.get(), result, "Task1/output");

// 处理任务二：汇总航站楼每天的销售金额

} else if (keyStr.startsWith("Task2\_")) {

for (Text val : values) {

sumExpense += parseDouble(val.toString());

}

String[] parts = keyStr.substring(6).split("\_", 2); // 获取航站楼和日期

String concourse = parts[0];

String date = parts[1];

String output = concourse + "\t" + date + "\t" + (long) sumExpense;

result.set(output);

multipleOutputs.write(NullWritable.get(), result, "Task2/output");

// 处理任务三：输出 has\_c2 为 TRUE 的数据

} else if ("Task3".equals(keyStr)) {

for (Text val : values) {

result.set(val.toString());

multipleOutputs.write(NullWritable.get(), result, "Task3/output");

}

}

}

@Override

protected void cleanup(Context context) throws IOException, InterruptedException {

multipleOutputs.close(); // 关闭MultipleOutputs

}

// 安全解析字符串为双精度数

private double parseDouble(String str) {

try {

return Double.parseDouble(str);

} catch (NumberFormatException e) {

return 0.0; // 返回0以避免异常

}

}

}

上述代码将三次任务的代码通过MultiTaskMapper类和MultiTaskReducer共同执行了，因此这里只写了合在一起的代码。

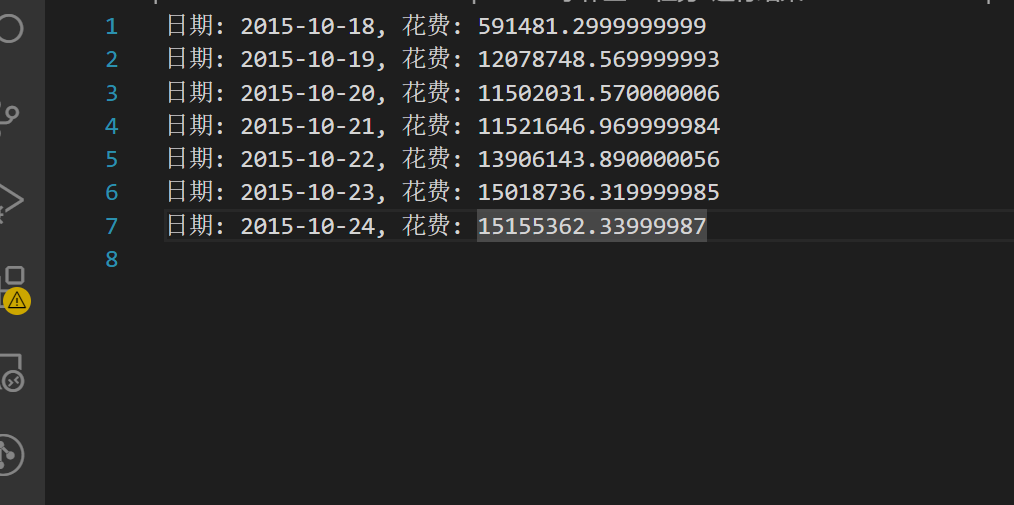


图1 任务2-1输出结果

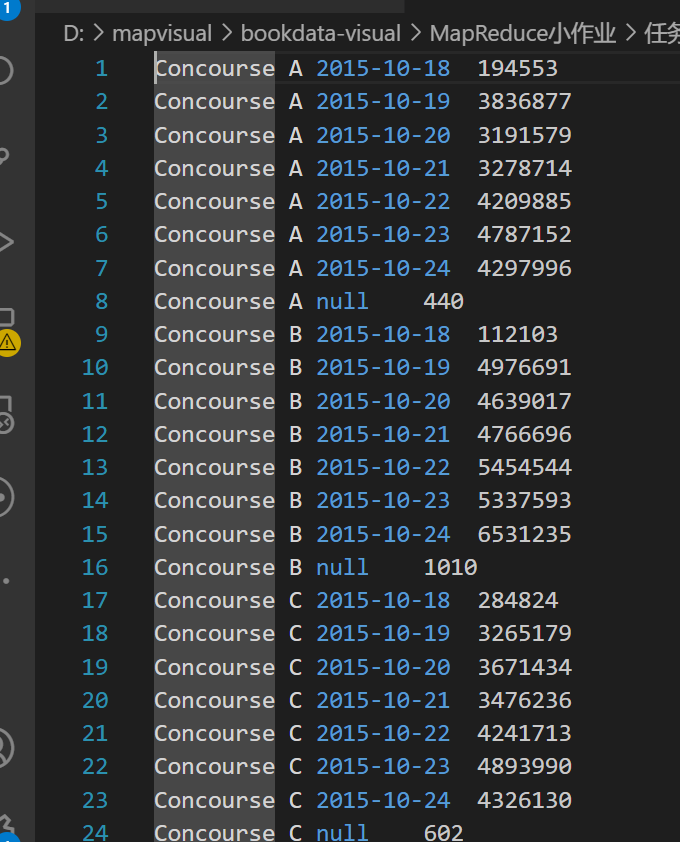


图2 任务2-2输出结果



图3 任务2-3输出结果

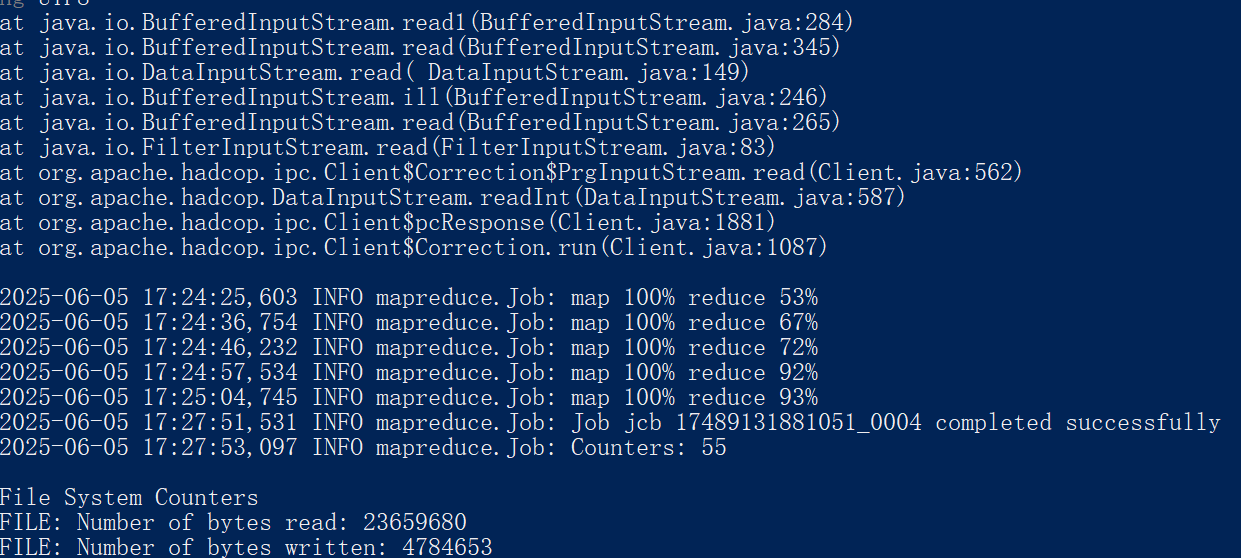


图4 MapReduce任务运行界面

总结：在本次实验中，实现了一个基于MapReduce框架的数据处理程序，分析机场交易数据。通过Map阶段，过滤标题行、提取重要字段，并生成不同的键值对以进行汇总。Reduce阶段则对这些数据进行聚合，使用MultipleOutputs将结果写入不同文件。实验提升了我的Java编程能力和数据处理技巧，同时让我学会了如何解决数据质量问题，如缺失值和格式不一致。总体而言，这次实验加深了我对Hadoop编程模型的理解。