Report with Error Handling and Deduplication in Talend Studio - Solution

Course: T101 - Tele Demo Course

Created: July 03, 2025

Total Marks: 100

Problem Statement

Scenario: You are given two different files: customers.csv – Contains customer information, potentially with duplicates. orders.csv – Contains order data including customer IDs and order values. Your task is to: Build a Talend job to: Read and parse both customers.csv and orders.csv. Join the two datasets based on the customer_id field. Deduplicate customer records based on unique customer identifiers. Calculate a loyalty score for each customer using total order value (e.g., tiered score based on spending). Handle and log errors, such as missing or invalid customer IDs in the order file. Export the final enriched and cleaned dataset to an Excel file. Bonus: Flag customers with no orders as "INACTIVE" in the output. Generate warnings if an order refers to a customer ID not present in the customer file. Include a timestamp column indicating when the record was processed.

Knowledge Topics

- In Talend Studio (Trial version)
- lookup joins, data deduplication, and custom logic and error handling
- Advanced components and its usage
- tMap with lookup functionality
- tUniqRow for deduplication
- tLogCatcher and tDie for error handling
- tFileOutputExcel for exporting data
- tFlowToIterate and tIterateToFlow to demonstrate iteration
- tJavaRow to apply custom Java logic

Scoring Rubric

Assessment Aspect	Marks
Handled CSV files in Talend?	20 marks
Was the join between customers and orders based on customer_id implemented effectively (e.g., via tMap)?	20 marks
Did the job use tMap to handle logic such as deduplication, loyalty score calculation, and setting "INACTIVE" st	20 marks
Did the output go to a structured Excel file using components like tFileOutputExcel with expected columns and	20 marks
Was there logic to detect and log issues such as missing or invalid customer IDs in the orders data using comp	20 marks

Knowledge Pills

DETAILED SOLUTION

1. Handled CSV files in Talend? [20 marks]

Step 1: Reading and Parsing CSV Files In this Talend job, we begin by using the tFileInputDelimited component for each CSV file: one for customers.csv and one for orders.csv. Be sure to set the field separator (typically a comma) and the header row option to correctly parse the files. Implementation Details: • Configure tFileInputDelimited for customers.csv with the schema including customer_id and other customer fields. • Configure another tFileInputDelimited for orders.csv, ensuring the schema includes order_id, customer_id, and order_value. • Ensure the components are set to catch header rows, and delimiters are correctly specified. Code Example: // Example pseudo-configuration snippet // For customers.csv Component: tFileInputDelimited File Name: "customers.csv" Field Separator: "," Header: 1 row Schema: {customer_id:String, name:String, ...} // For orders.csv Component: tFileInputDelimited File Name: "orders.csv" Field Separator: "," Header: 1 row Schema: {order_id:String, customer_id:String, order_value:Double, ...} This setup demonstrates a problem-specific solution by ensuring proper CSV parsing which is foundational to the subsequent data transformation tasks.

2. Was the join between customers and orders based on customer_id implemented effectively (e.g., via tMap)? [20 marks]

Step 2: Joining Datasets via tMap The next step involves merging customers and orders based on the customer_id field using the tMap component. In the tMap, we set the main flow from the customers component (after deduplication handling) and a lookup flow from the orders component. Implementation Details: • Connect the customer input as the main flow to the tMap. • Connect the orders input as the lookup, matching on customer_id. • Use the joining condition customer_main.customer_id = orders_lookup.customer_id. • If the lookup does not find a match, additional logic will later mark the customer as "INACTIVE". This method addresses the problem-specific requirement of linking data across the two CSV files. // Pseudo configuration within tMap: // Main Flow: Customers // Lookup Flow: Orders // Join condition: main.customer_id == lookup.customer_id

3. Did the job use tMap to handle logic such as deduplication, loyalty score calculation, and setting "INACTIVE" status? [20 marks]

Step 3: Business Logic in tMap This section explains how to embed complex logic using tMap along with supporting components: Implementation Details: • Deduplication: Use the tUniqRow to remove duplicate customer records based on unique customer_id before the join. This prevents multiple rows for the same customer from affecting the loyalty calculation. • Loyalty Score Calculation: Incorporate a tJavaRow component after the join to aggregate the total order_value and to apply tier-based loyalty scoring. For instance, if a customer's total spending exceeds a threshold, assign a higher loyalty score. • Flagging INACTIVE Customers: Within tMap, add a condition so that if no order details are found (i.e., lookup is null), the customer status is set to "INACTIVE". Code Example: // tJavaRow pseudo-code for calculating loyalty score if (row1.order_value != null) { double total = row1.order_value; // aggregated from multiple rows, normally by a prior aggregation if (total > 1000) { row1.loyalty_score = "Gold"; } else if (total > 500) { row1.loyalty_score = "Silver"; } else { row1.loyalty_score = "Bronze"; } } else { row1.loyalty_score = "Bronze"; } // Set inactive status if no orders attached if (row1.customer_id == null || row1.order_value == null) { row1.status = "INACTIVE"; } else { row1.status = "ACTIVE"; } This detailed implementation shows how the tMap component is customized for problem-specific business logic.

4. Did the output go to a structured Excel file using components like tFileOutputExcel with expected columns and formatting? [20 marks]

Step 4: Exporting to an Excel File After processing, the final enriched and cleaned dataset is written to an Excel file using the tFileOutputExcel component. This ensures that the output is well structured and formatted. Implementation Details: • Connect the output from the previous transformation components (including deduplication and loyalty score calculations) to the tFileOutputExcel component. • Specify the output path and ensure the component uses appropriate Excel formatting (e.g., column names, headers, data types). • Include all required columns: customer_id, name, total_order_value, loyalty_score, status, and a timestamp. Code Example: // Pseudo configuration for tFileOutputExcel Component: tFileOutputExcel Output File: "output/enriched_customers.xlsx" Schema: {customer_id, name, total_order_value, loyalty_score, status, processed_timestamp} Options: Create header row, define column formats if needed This solution component directly fulfills the requirement for a structured Excel output with proper columns and formatting.

5. Was there logic to detect and log issues such as missing or invalid customer IDs in the orders data using components like tLogCatcher, tWarn, or error handling branches? [20 marks]

Step 5: Error Handling and Logging To address potential data integrity issues, such as missing or invalid customer IDs in the orders data, the Talend job incorporates custom error handling components. Implementation Details: • Route rows with missing or invalid customer_id from the orders file to an error handling branch using the tLogCatcher and tWarn components. • When a discrepancy is detected (e.g., an order with a customer_id not found in the customers dataset), generate a warning message and log the issue. • Utilize the tDie component where necessary to halt the job if critical errors occur, ensuring prompt attention. • Also include a timestamp column via a tJavaRow to record when each record was processed. Code Example: // Example pseudo-code within a tJavaRow for logging timestamp and error handling if (orderRow.customer_id == null || orderRow.customer_id.isEmpty()) { globalMap.put("ERROR_MESSAGE", "Missing customer_id in order: " + orderRow.order_id); // Redirect row to an error flow using tLogCatcher } else if (!customerLookup.containsKey(orderRow.customer_id)) { globalMap.put("WARNING_MESSAGE", "Order " + orderRow.order_id + " refers to non-existent customer_id: " + orderRow.customer_id); // Issue a warning via tWarn } row.processed_timestamp = new java.util.Date(); This part of the solution demonstrates explicit error detection and logging tailored to the problem's specifications.