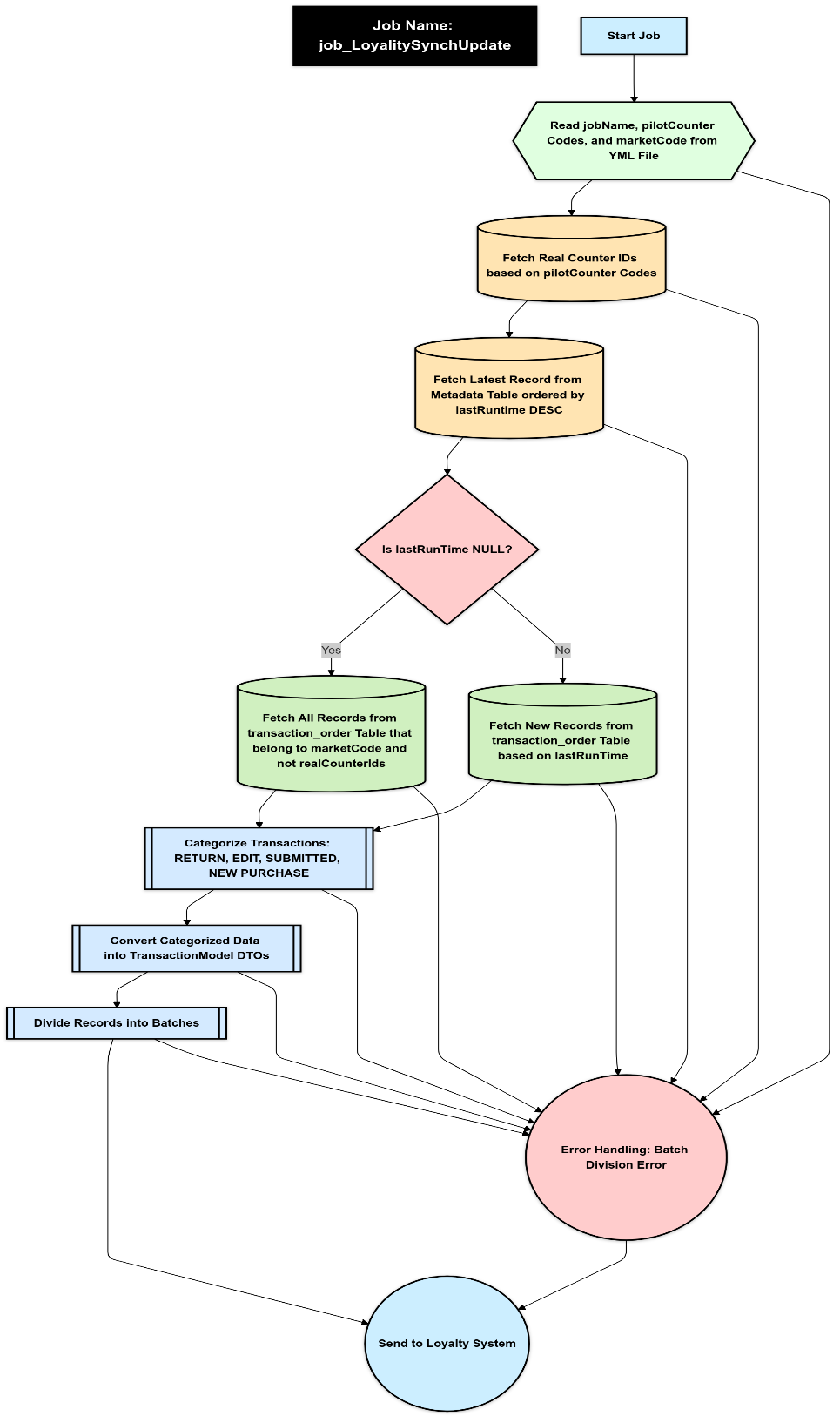
**Detailed Flow of the DDPLoyalitySynchUpdateSrviceImpl Implementation**

**1. Overview of the Service**

The DDPLoyalitySynchUpdateSrviceImpl class is responsible for processing loyalty synchronization updates. It interacts with the database, categorizes transactions(EDIT, RETURN,SUBMITTED), processes various types of transactions, and prepares data to send to loyalty systems.



**2. Primary Method: loyalitySynchUpdateProcess**

This is the entry point for the loyalty synchronization job.

**Steps in the Method:**

1. **Logging Job Start:**
   * Logs the beginning of the synchronization process.
2. **Input Extraction:**
   * Extracts the jobName, pilotCounter, and marketCode from the DataLoadBaseConfiguation object.
3. **Split Pilot Counter:**
   * Calls splitPilotCounterToList to convert the pilotCounter string into a list of counter codes.
4. **Fetch Real Counter IDs:**
   * Calls getRealCounterIds to retrieve actual counter IDs corresponding to the counter codes.
5. **Fetch New Records:**
   * Retrieves new transactions by invoking fetchNewRecords using the jobName, marketCode, and real counter IDs.
6. **Categorize Transactions:**
   * Calls categorizeTransactions to group transactions by their types (PerfTransactionType—EDIT, RETURN, SUBMITTED).
7. **Process Categorized Transactions:**
   * Invokes processCategorizedTransactions to process transactions based on their category and prepares them for sending to the loyalty system.
8. **Error Handling:**
   * Logs and throws exceptions if any error occurs during the process.

**3. Supporting Methods**

**3.1 splitPilotCounterToList**

* **Input:** pilotCounter (comma-separated string)
* **Output:** List of non-empty trimmed strings.
* **Flow:**
  + Splits the string using ,.
  + Removes empty entries and trims whitespace.

**3.2 getRealCounterIds**

* **Input:** List of counter codes.
* **Output:** List of real counter IDs fetched from the database.
* **Flow:**
  + Checks if the input list is null or empty, and throws an exception if true.
  + Queries the CounterRepository to fetch IDs for the provided counter codes.

**3.3 fetchNewRecords**

* **Input:** jobName, marketCode, and listOfRealCounterIds.
* **Output:** List of new TransactionEntity records.
* **Flow:**
  + Retrieves the last run time of the job using LoyalitySynchUpdateMetaDataRepository.
  + Queries the TransactionRepository to fetch transactions created after the last run time.
  + Logs the fetched records.

**3.4 categorizeTransactions**

* **Input:** List of new transaction entities.
* **Output:** Map of PerfTransactionType to lists of TransactionEntity.
* **Flow:**
  + Initializes a map with empty lists for all transaction types.
  + Categorizes each transaction based on its PerfTransactionType.

**3.5 processCategorizedTransactions**

* **Input:** Categorized transactions and marketCode.
* **Output:** Map of PerfTransactionType to lists of OrderTransactionDto.
* **Flow:**
  + Iterates through each category.
  + Calls specific methods for each transaction type:
    - processOriginalTransactions
    - processReturnTransactions
    - processEditTransactions
    - processNewPurchaseTransactions

**4. Processing Methods for Transaction Categories**

**4.1 processOriginalTransactions**

* **Purpose:** Processes original transactions and prepares OrderTransactionDto objects.
* **Flow:**
  + Converts TransactionEntity objects to TransactionModel.
  + Prepares loyalty DTOs using prepareSendCompletedOrderList.
  + Logs success or warnings for each transaction.

**4.2 processReturnTransactions**

* **Purpose:** Processes return transactions.
* **Flow:**
  + Converts TransactionEntity to TransactionModel.
  + Prepares loyalty DTOs using sendRefundedOrder.
  + Logs results and errors.

**4.3 processEditTransactions**

* **Purpose:** Processes edited transactions.
* **Flow:**
  + Similar to processReturnTransactions, but uses the sendEditedOrder method for DTO preparation.

**4.4 processNewPurchaseTransactions**

* **Purpose:** Processes new purchase transactions.
* **Flow:**
  + Converts entities to models.
  + Uses relevant logic for processing (currently incomplete in the code).

**5. DTO Preparation Methods**

**5.1 prepareSendCompletedOrderList**

* **Purpose:** Converts a transaction into an OrderTransactionDto.
* **Flow:**
  + Maps transaction and counter entity to DTO.
  + Fetches campaign details and populates the DTO with campaign names and codes.
  + Sets additional fields like event type and update date.

**5.2 sendRefundedOrder**

* **Purpose:** Prepares OrderTransactionDto for refunded orders.
* **Flow:**
  + Retrieves the original and purchase transactions.
  + Validates consumer status.
  + Maps the return transaction to a DTO using OrderRefundedMapper.
  + Populates campaign details.

**5.3 sendEditedOrder**

* **Purpose:** Prepares OrderTransactionDto for edited orders.
* **Flow:**
  + Maps the transaction and counter to a DTO.
  + Fetches and populates campaign details.

\*Note :- Document will be updated (logic, flow) once Completed Job Implementation Will Done