Solution #1: Salesforce Org Case object Sync to Snowflake using batch processing with scheduler.

Assumptions:

- Able to use Snowflake Connector
- Able to use Json Logger
- All processes in one API, not following API led design so that everything is within one project. Realistically would want to split into system, process, and experience layer APIs for reusability.
- Not grabbing all fields from Case object, only ones needed are listed in data mapping below.
- Three different case tables for snowflake - case_usa, case_can, case_aus
- We will be using the CaseNumber as the unique Id for both Snowflake and Salesforce
- Deploying to Cloudhub, and will have ability to scale the vcores/workers to handle to amount of data.

Limitations:

- No access to Affirm Salesforce Org or Snowflake DB - Using made up placeholders for configuration

Process Overview:

1. Scheduler kicks off process (one for each region - USA, CAN, AUS)
2. Use ObjectStore to capture the last time process ran (if first time, sets default to now())
3. Reference batch process flow
4. Retrieve the last time process ran from object store
5. Query Salesforce for all changed Case object records since last run
6. Use Batch to process all records from previous step:
    a. Batch step 1 - Query Snowflake to check if current record exists and output the result to a variable “recordExists”, then transform the Salesforce Data into Snowflake format.
    b. Batch step 2 - use AcceptExpression configuration to only run this batch step if the recordExists variable is null (doesnt exist). Then we will <u>insert</u> the record into Snowflake.
    c. Batch Step 3 - use AcceptExpression configuration to only run this batch step if recordExists is not null (the record is already in Snowflake). Then we will <u>update</u> the existing record in Snowflake with up to date Salesforce data.

Solution #2: Salesforce Org Case object sync to Snowflake using synchronous Salesforce Topic Listener. (near real time)

Assumptions:

- A Streaming Push Topic is created in Salesforce Org that is listening for all changes on Case object (both updated and new), as well as a subscription to that push topic that we can use in Mulesoft with “Subscribe Topic Listener” component.
- Able to use Snowflake Connector
- Able to use Json Logger
- All processes in one API, not following API led design so that everything is within one project. Realistically would want to split into system, process, and experience layer APIs for reusability.
- Only fields for case object needed in Snowflake are given below in data mapping.
- Three different case tables for snowflake - case_usa, case_can, case_aus
- We will be using the CaseNumber as the unique Id for both Snowflake and Salesforce
- Deploying to Cloudhub, and will have ability to scale the vcores/workers to handle to amount of data.

Limitations:

- No access to Affirm Salesforce Org or Snowflake DB - Using made up placeholders for configuration

Process Overview:

1. Use “Subscribe Topic Listener” component as the source, to be subscribed to the push topic listener on Salesforce end. This way anytime a Case record is updated or created, the data from the record will be received in Mulesoft.
2. Make query to Snowflake to check if the record coming in from Salesforce exists in Snowflake DB. Set the output of this query to variable “recordExists”.
3. Transform he Salesforce Data into Snowflake Format.
4. Use choice router to check the country of the record, route to subflow based on country (USA, CAN, AUS). This will determine which table in Snowflake we update/insert in the subflows.
5. Use choice router in subflow:
    a. If recordExists var is not null (record exists in Snowflake), then we update Snowflake case record with most recent incoming Salesforce data.
    b. Else, we need to create new Case record in Snowflake.

Data Mapping: Case

Salesforce Object	**Salesforce Field**	**Snowflake Table**	**Snowflake Field**
Case	CaseNumber	case_<usa/can/aus>	case_number

Case	Origin	case_<usa/can/aus>	origin
Case	Reason	case_<usa/can/aus>	reason
Case	ContactEmail	case_<usa/can/aus>	contact_email
Case	Country__c	case_<usa/can/aus>	country
Case	CreatedDate	case_<usa/can/aus>	created_date
Case	ClosedDate	case_<usa/can/aus>	closed_date
Case	LastModifiedDate	case_<usa/can/aus>	last_modified