

## TASK 5

**Retrieving data. Wait a few seconds and try to cut or copy again:** Implement logic to handle data retrieval failures gracefully by waiting briefly (e.g., a few seconds) before retrying cut, copy, or extraction operations, improving pipeline resilience and reducing transient error impacts.

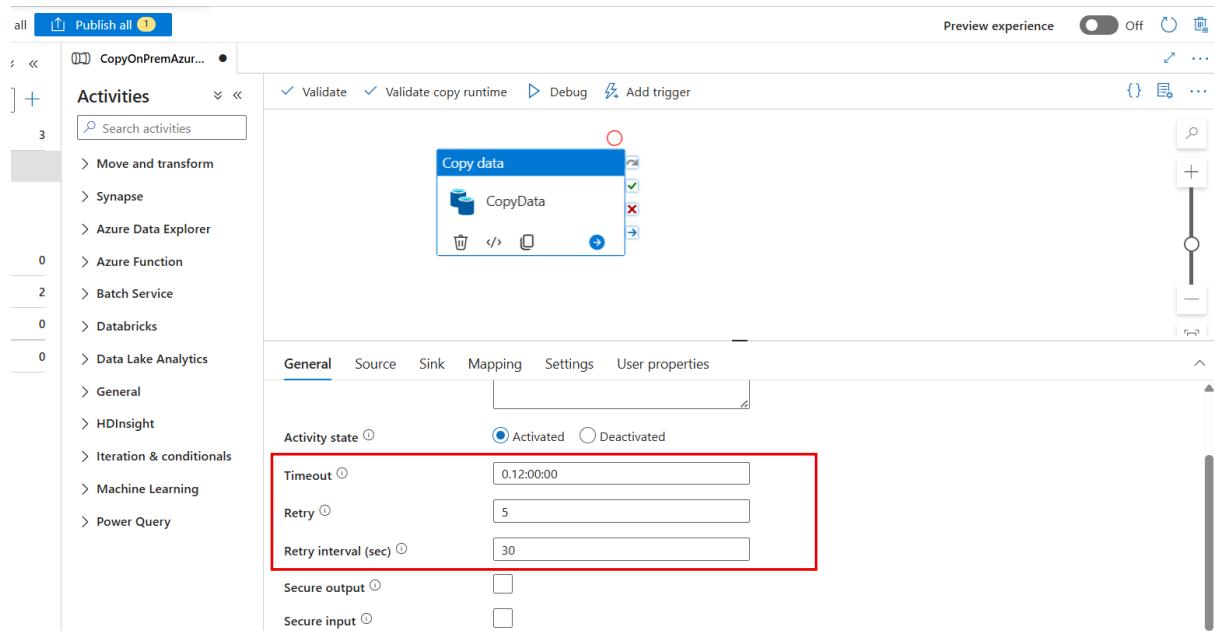
There are three methods to handle data retrieval failures gracefully:

### **Method 1: Use Built-in Retry on Activities**

**Steps:**

1. Select activity (e.g., Copy data, Lookup, Web).
2. In the **General tab**:
  - o Set Retry count (default is 0)
  - o Set Retry interval in **seconds**

This will automatically retry transient failures.



## Method 2: Add Custom Retry Logic (Wait + Until)

This method provides more control (e.g. retry only on certain errors), use a loop with **Wait** and **Until** activities.

### How it works:

- Loop up to N times
- Wait in between
- Stop when activity succeeds or max attempts reached

### Step-by-Step Guide:

#### 1. Add 2 Pipeline Variables:

- retryCount → type: **Int**, default: 0
- succeeded → type: **Bool**, default: false
- tempCount → type: **Int**, default: 0

The screenshot shows the Azure Data Factory pipeline editor interface. On the left, there's a sidebar with a tree view of activities: Move and transform, Synapse, Azure Data Explorer, Azure Function, Batch Service, Databricks, Data Lake Analytics, General, HDInsight, Iteration & conditionals, Machine Learning, and Power Query. The main area is titled 'Activities' and has a search bar. Below the search bar is a list of activities: Move and transform, Synapse, Azure Data Explorer, Azure Function, Batch Service, Databricks, Data Lake Analytics, General, HDInsight, Iteration & conditionals, Machine Learning, and Power Query. To the right of the sidebar is a large workspace. At the top of the workspace, there are tabs for 'Parameters', 'Variables', 'Settings', and 'Output'. The 'Variables' tab is selected. Below the tabs, there's a table with three rows. The first row has a red border and contains the following data:

Name	Type	Default value
retryCount	Integer	0
succeeded	Boolean	false
tempCount	Integer	0

## 2. Add a Set Variable Activity: ResetRetries

- Set retryCount = 0
- Set succeeded = false

The screenshot shows the Azure Logic Apps designer interface. It displays two parallel steps. The top step contains a 'Set variable' activity with '(x) retryCounts' and another 'Set variable' activity with '(x) Succeed'. Below these are their respective configuration panes.

**Left Step Configuration:**

- General tab selected.
- Variable type: Pipeline variable.
- Name: retryCount.
- Value: 0.

**Right Step Configuration:**

- General tab selected.
- Variable type: Pipeline variable.
- Name: succeeded.
- Value: false.

## 3. Add an Until Activity:

`@or(greaterOrEquals(variables('retryCount'), 3), variables('succeeded'))`

- This means: keep retrying **until** 3 attempts or success.

The screenshot shows the Logic Apps designer with an 'Until' activity. It has a 'Loop' condition and an 'Activities' section containing a placeholder 'No activities'.

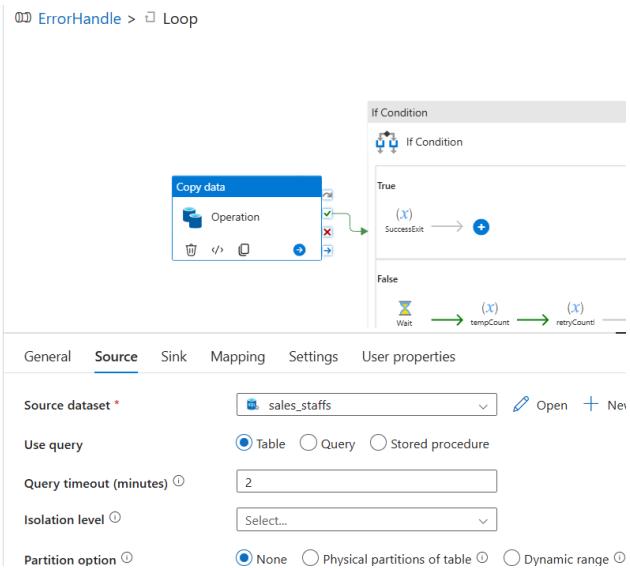
**Until Activity Settings:**

- General tab selected.
- Expression: `@or(greaterOrEquals(variables('retryCo...`
- Timeout: 0.12:00:00.

4. Inside the Until activity, add these (we are performing data transfer operation as Task 1):

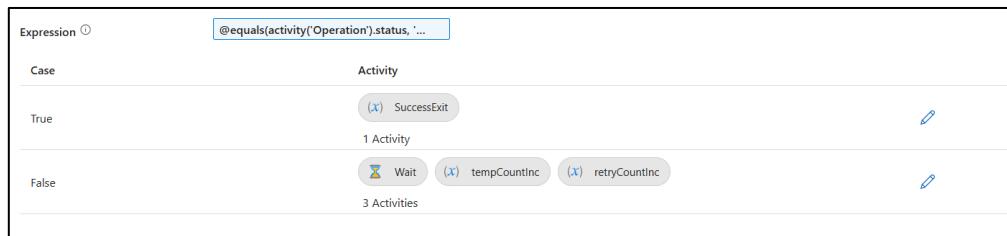
a. **Copy Data**

- Name: **Operation**
- Set up configuration



b. Add If Condition after Copy Data operation to check for success:

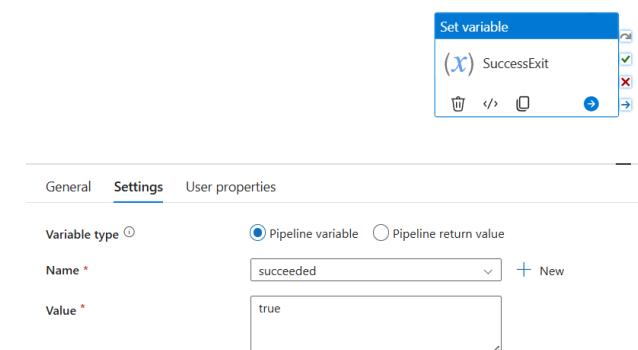
Value of If Condition: `@equals(activity('Operation').status, 'Succeeded')`



- On Success path:

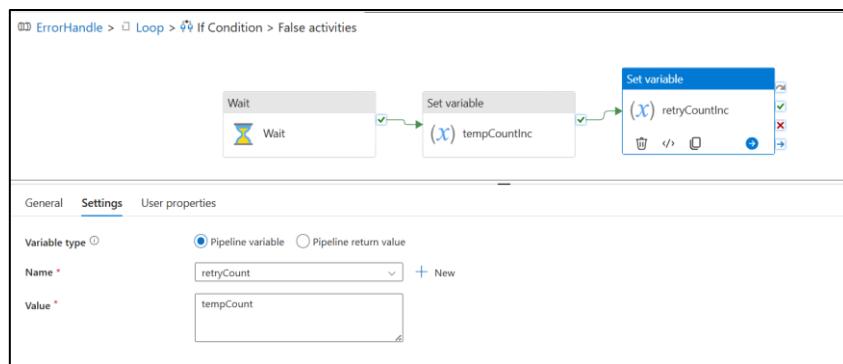
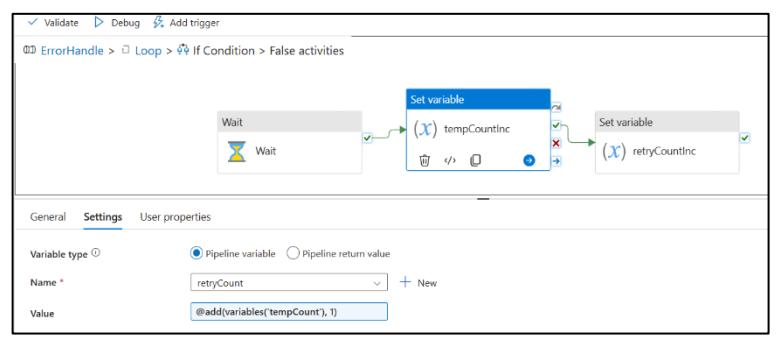
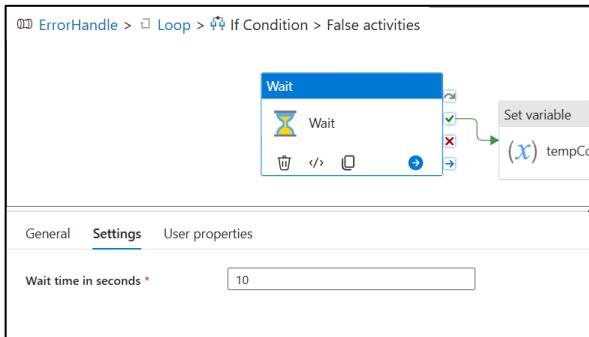
- Add Set Variable: succeeded = true

00 ErrorHandle > Loop > If Condition > True activities

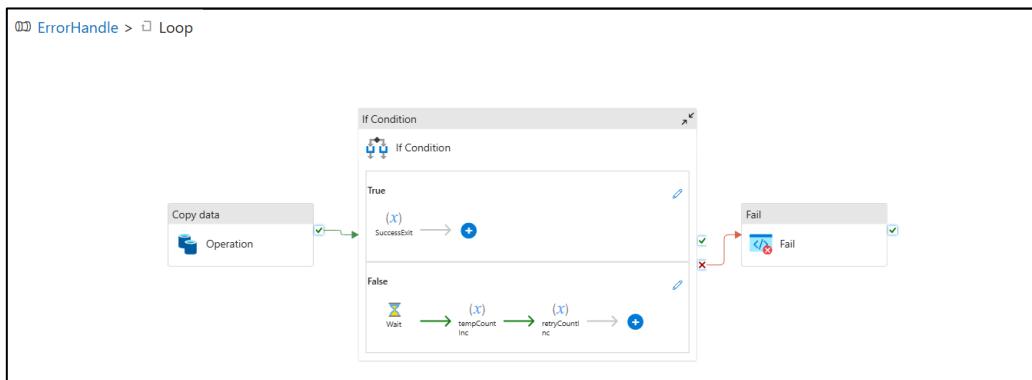


- Else,
  - Add a **Wait** activity: Duration: 10 (seconds)
  - Then add a **Set Variable** to increment tempCount  

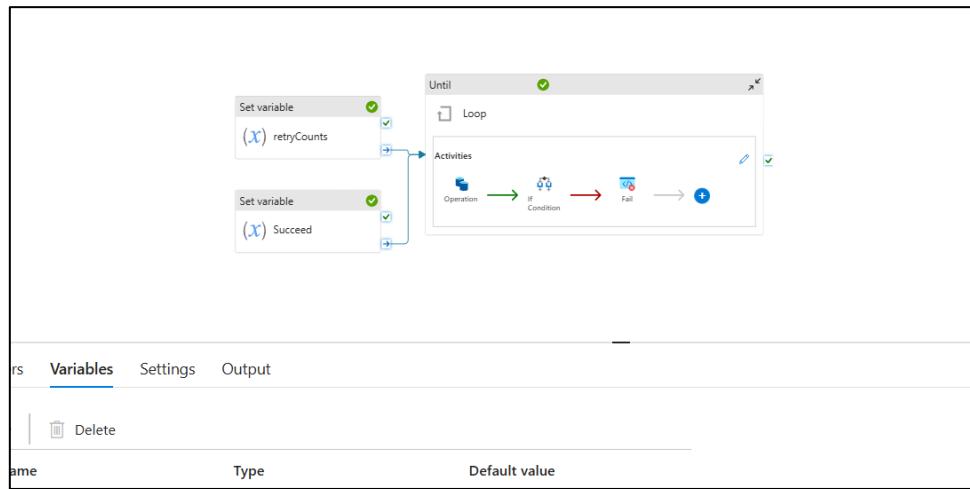
$$@add(variables('retryCount'), 1)$$
  - Set retryCount to tempCount



Final Until activity:



## Final pipeline:



## SUCCESSFUL AUTOMATION:

The screenshot shows the Azure Data Factory Pipeline Runs page. At the top, it displays the pipeline run details: 'ErrorHandler - Activity runs'. It indicates that this is a recent debug run. Below this, there's a summary table showing the status of various activities. Further down, a detailed table lists all activity runs from the pipeline run, including their names, statuses, start times, durations, and run IDs.

Activity name	Activity st...	Activit...	Run start	Duration	Integration runtime	User prop...	Activity run ID
SuccessExit	<span style="color: green;">✓</span> Succeeded	Set variable	7/14/2025, 1:13:27 PM	Less than 1s			63854ef8-b629-4
If Condition	<span style="color: green;">✓</span> Succeeded	If Condition	7/14/2025, 1:13:27 PM	2s			3c3c0470-9876-4
Operation	<span style="color: green;">✓</span> Succeeded	Copy data	7/14/2025, 1:13:06 PM	20s	SQLOnPrem		ff9b0ffd-e61b-47
Loop	<span style="color: green;">✓</span> Succeeded	Until	7/14/2025, 1:13:05 PM	28s			39088698-9496-4
retryCounts	<span style="color: green;">✓</span> Succeeded	Set variable	7/14/2025, 1:13:05 PM	Less than 1s			05fb2b4f-2c1c-4c
Succeed	<span style="color: green;">✓</span> Succeeded	Set variable	7/14/2025, 1:13:05 PM	Less than 1s			a3bcbb76-7b1d-4

### **Method 3: Graceful Failure Handling via OnFailure Path**

Use **activity error paths** to handle failure cleanly without failing the pipeline immediately.

This gives you the ability to:

- Log the error
- Wait/retry once or multiple times
- Exit cleanly or alert downstream systems