**Incremental Data Loading and Notifications**

Bootcamp Project - 5

**Overview**

This project involves working with Azure Data Factory (ADF) to implement data ingestion and notification pipelines. The project will cover the following scenarios:

1. **Incremental Data Load and Process Tracking**: Load data incrementally into Azure SQL Database and update a tracking table.
2. **Time-Based Notification**: Send a notification if the pipeline succeeds or fails during a specific time window.
3. **Data Ingestion and Team Notification**: Ingest data from an SQL database into Azure Data Lake Storage and notify the data engineering team via Slack/Teams.

**Role:**

* **Data Pipeline Development:** Designed and implemented scalable data pipelines in Azure Data Factory for incremental data loading, ensuring efficient ingestion from external systems into Azure SQL Database and marking records as processed using stored procedures.
* **Automation and Notification:** Integrated webhook-based Slack/Teams notifications in ADF pipelines to enhance monitoring, ensuring proactive alerting for both success and failure scenarios in data ingestion processes.
* **Version Control and Deployment:** Followed structured development practices using GitHub for version control, working within feature branches for isolated changes, conducting peer reviews, and deploying ARM templates for streamlined release management.

**Scenario 1: Incrementally Load Data and Mark as Processed**

**Objective:**

Incrementally load data from an external system into Azure SQL Database. After each load, a stored procedure will execute to mark the records as processed or update the status in a tracking table.

**Pipeline Setup:**

1. **Lookup Activity**:
   * Retrieve the last processed timestamp from a control table to identify where to start the incremental load.
2. **Copy Data Activity**:
   * Ingest new records from the external system based on the timestamp.
3. **Stored Procedure Activity**:
   * Execute a stored procedure to update the control table and mark records as processed in the destination database.

**Resource groups:**

A screenshot of a computer

AI-generated content may be incorrect.

ADF is initially connected to Git Hub and later switched to its Feature branch Jirra-122.

A screenshot of a computer

AI-generated content may be incorrect.

SQL Database Schema and Sample Data:

1. **Create Source Table (Source Table)**

CREATE TABLE SourceTable (

    ID INT PRIMARY KEY,

    CustomerName NVARCHAR(100),

    Amount DECIMAL(10, 2),

    ModifiedDate DATETIME

);

-- Sample Data

INSERT INTO SourceTable (ID, CustomerName, Amount, ModifiedDate) VALUES

(101, 'Alice Johnson', 500.00, '2024-03-25 10:00:00'),

(102, 'Bob Smith', 1200.00, '2024-03-25 12:30:00'),

(103, 'Charlie Brown', 800.00, '2024-03-25 15:45:00'),

(104, 'David Williams', 2000.00, '2024-03-26 09:00:00'),

(105, 'Emma Watson', 3000.00, '2024-03-26 14:15:00');

1. **Create Destination Table (DestinationTable)**

CREATE TABLE DestinationTable (

    ID INT PRIMARY KEY,

    CustomerName NVARCHAR(100),

    Amount DECIMAL(10, 2),

    ModifiedDate DATETIME,

    ProcessedStatus BIT DEFAULT 0

);

-- Sample Data (for validation purposes)

INSERT INTO DestinationTable (ID, CustomerName, Amount, ModifiedDate, ProcessedStatus) VALUES

(100, 'Olivia Martinez', 700.00, '2024-03-24 08:00:00', 1);

1. **Create Control Table (ControlTable)**

--Control Table

CREATE TABLE ControlTable (

    ID INT PRIMARY KEY IDENTITY(1,1),

   LastProcessedTime DATETIME

);

INSERT INTO ControlTable (LastProcessedTime) VALUES

('2024-03-24 00:00:00');

1. **Stored Procedure to Update Control Table & Mark Records**

USE [sqldb-incremental-load]

GO

CREATE PROCEDURE UpdateControlTable

    @NewLastProcessedTime DATETIME

AS

BEGIN

    -- Step 1: Update Control Table

    UPDATE ControlTable

    SET LastProcessedTime = @NewLastProcessedTime;

    -- Step 2: Mark Records as Processed

    UPDATE DestinationTable

    SET ProcessedStatus = 1

    WHERE ModifiedDate <= @NewLastProcessedTime;

END;

GO

A screenshot of a computer

AI-generated content may be incorrect.

**ADF Configuration**

* Created Azure SQL Database Linked Service
* Azure Blob Storage Linked Service

A screenshot of a computer

AI-generated content may be incorrect.

**Dataset Creation**

* Dataset for Source Data
* Dataset for Destination Data
* Dataset for Control Table

**A screenshot of a computer

AI-generated content may be incorrect.**

**Pipeline Design**

**Lookup Activity:**

* Retrieves the most recent timestamp from Control Table.
* **SQL Query:**

SELECT MAX(LastProcessedTime) AS LastProcessedTime FROM ControlTable;

**Copy Data Activity:** IncrementalDataLoad

* Copies new data from SourceTable to DestinationTable.
* **Source Query (with dynamic content):**
* SQL Query:

SELECT \* FROM SourceTable

WHERE ModifiedDate > '@{activity('GetLastProcessedTime').output.firstRow.LastProcessedTime}'

**Stored Procedure Activity:** MarkAsProcessed

* Calls the UpdateLastProcessedTimestamp procedure.

A computer screen shot of a computer

AI-generated content may be incorrect.

Validation:

* Destination Table has the newly ingested data with Processed Status = 1 (True)

A screenshot of a computer

AI-generated content may be incorrect.

* Control Table has the updated timestamp.

A screenshot of a computer

AI-generated content may be incorrect.

**Create a Pull Request (PR) for Peer Validation**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Merge PR into Main Branch**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Switching back ADF to Main branch**

A screenshot of a computer

AI-generated content may be incorrect.

Everything is saved to the Main Branch

A screenshot of a social media account

AI-generated content may be incorrect.

**Scenario 3: Data Ingestion with Slack/Teams Notification**

**Objective:**

Ingest data from an SQL database into Azure Data Lake Storage, and upon success or failure of the job, send a notification to the data engineering team via Slack/Teams.

**Pipeline Setup:**

1. **Copy Data Activity**:
   * Ingest data from the SQL database into Azure Data Lake Storage (ADLS).
2. **Web Activity (Success)**:
   * Send a webhook to Slack or Teams with a success message when the pipeline succeeds.
3. **Web Activity (Failure)**:
   * Send a webhook to Slack or Teams with a failure message when the pipeline fails.

**Resource Creation:**

* **ADLS** with a container
* **SQL Db**

In SQL Db, required tables are created,

Query used:   
CREATE TABLE SalesData (

    SaleID INT PRIMARY KEY,

    CustomerName NVARCHAR(100),

    Product NVARCHAR(100),

    SaleAmount FLOAT,

    SaleDate DATETIME

);

INSERT INTO SalesData VALUES

(1, 'Alice', 'Laptop', 1200.00, '2025-03-25'),

(2, 'Bob', 'Smartphone', 800.00, '2025-03-26');

A screenshot of a computer

AI-generated content may be incorrect.

**Azure Data Factory Setup**

* Created 2 linked services for SQLDB and ADLS

A screenshot of a computer

AI-generated content may be incorrect.

* Created data sets as required

A close up of a text

AI-generated content may be incorrect.

**Pipeline Creation:**

* Copy activity for SQL-ADLS
* Web Activity (Success)
* Web Activity (Failure)

A screenshot of a computer

AI-generated content may be incorrect.

**Create a Pull Request (PR) for Peer Validation**

A screenshot of a computer

AI-generated content may be incorrect.

**Merge PR into Main Branch**

A screenshot of a chat

AI-generated content may be incorrect.

**Switching back ADF to Main branch**

A screenshot of a computer

AI-generated content may be incorrect.