**Incremental Data Loading and Notifications**

This project focuses on implementing incremental data loading, process tracking, and notifications using Azure Data Factory (ADF).

**Project Overview**

The project involves three main scenarios:

1. **Incremental Data Load and Process Tracking**: Load data incrementally into Azure SQL Database and track progress.
2. **Time-Based Notifications**: Send notifications based on pipeline success or failure within a specific time window.
3. **Data Ingestion and Team Notification**: Ingest data from SQL database into Azure Data Lake Storage (ADLS) and send Slack/Teams notifications.

**Architecture Diagram:**

A diagram of data processing

AI-generated content may be incorrect.

**My Role in the Project:**

**Objective:**

Ensure that the data engineering team receives real-time notifications on **Slack** regarding the success or failure of data ingestion and processing pipelines.

**Key Responsibilities:**

1. **Understanding Notification Requirements:**
   * Collaborated with the team to determine when notifications should be triggered (success/failure scenarios).
   * Identified the Slack Webhook URL to send messages to the correct Slack channel.
2. **Implementing Web Activity in ADF:**
   * Used **Web Activity** in **Scenario 2** to send messages to Slack.
   * Configured the **POST request** to the Slack webhook with a JSON payload containing the status message.
3. **Adding Conditional Logic:**
   * Used **If Condition Activity** to check if the pipeline ran within the required time window (for Scenario 2).
   * Configured **Failure Paths** to send failure notifications when the pipeline encountered errors.

**Resource Group:**

A screenshot of a computer

AI-generated content may be incorrect.

**Scenario 1: Incremental Data Load and Process Tracking**

**Objective:**

Incrementally load data from an external system into Azure SQL Database. After each load, update a control table to mark records as processed.

**Pipeline Setup:**

* **Lookup Activity**: Retrieve the last processed timestamp from a control table to determine where to start the incremental load.
* **Copy Data Activity**: Ingest new records based on the retrieved timestamp.
* **Stored Procedure Activity**: Execute a stored procedure to update the control table and mark records as processed.

**Phase 1 - Changes for Scenario 1**

1. **GitHub Integration**:
   * Connect Azure Data Factory with your GitHub repository for version control.
2. **Workspace Setup**:
   * Switch to or create the feature/jira-122 branch for development.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Peer Validation**:
   * Conduct a peer review within the feature/jira-122 branch to ensure proper functionality.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Merge and Publish**:
   * Merge changes into the main branch after successful testing and publish the ARM templates for deployment.

A screenshot of a computer

AI-generated content may be incorrect.

**Scenario 2: Time-Based Notification**

**Objective:**

Configure a pipeline to send notifications if it succeeds or fails during a specific time window (e.g., 7 AM to 9 AM).

**Pipeline Setup:**

* **Web Activity (Success/Failure Notification)**: Trigger a notification webhook (Slack/Teams) based on pipeline success or failure.
* **Conditional Logic**: Apply conditions to check the current time before sending notifications, ensuring they are only sent during the time window.

**Phase 1 - Changes for Scenario 2**

1. **GitHub Integration**:
   * Set up GitHub repository integration in Azure Data Factory.
2. **Workspace Setup**:
   * Switch to or create the feature/jira-123 workspace for development.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Peer Validation**:
   * Peer review within the QA workspace.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Merge and Publish**:
   * After successful testing, merge changes into the main branch and publish ARM templates.

A screenshot of a computer

AI-generated content may be incorrect.

**Version Control and Collaboration**

* **GitHub Integration**: Each scenario’s development is version-controlled through GitHub. Feature branches (e.g., feature/jira-122, feature/jira-123) were created for each task.
* **Peer Review**: Each feature branch underwent peer validation before merging into the main branch.
* **ARM Templates**: After merging the branches, the ARM templates were published to the GitHub publish branch for deployment.

**GitHub Link:**

https://github.com/Arunkumar-Senthilkumar/Incremental-Data-laoding