**Use Case: Dynamic Query: Automated Data Integration from Multiple Sources to a Data Warehouse**

**Background**

A retail company, "RetailCo," needs to integrate data from various operational databases (e.g., sales, inventory, and customer databases) into a centralized data warehouse for reporting and analytics. The company wants to automate this process due to the frequent changes in source tables and columns, as well as to streamline data loading operations.

**Requirements**

1. **Data Integration**: Load data from multiple source systems into the target data warehouse.
2. **Dynamic Schema Handling**: Source schemas may change frequently (new columns added, columns renamed, delete the column etc.).
3. **Automation**: Minimize manual intervention in the data loading process.

**Sample Data:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table\_ID | Source\_System | Table\_name | Column\_Name | Is\_Column\_active\_Flag | Is\_Table\_Active\_Flag |
| 1 | SQLServer | Sales\_Data | SaleID | 1 | 1 |
| 2 | SQLServer | Sales\_Data | SaleAmount | 1 | 1 |
| 3 | SQLServer | Sales\_Data | SaleDate | 1 | 1 |
| 4 | MySQL | Inventory | ProductID | 1 | 1 |
| 5 | MySQL | Inventory | Quantity | 0 | 1 |
| 6 | MySQL | Inventory | Location | 1 | 1 |
| 7 | Oracle | Customer | CustomerID | 1 | 1 |
| 8 | Oracle | Customer | Name | 0 | 1 |
| 9 | Oracle | Customer | Email | 1 | 1 |

**Solution Overview**

RetailCo can use Azure Data Factory (ADF) to create a dynamic ETL pipeline that:

1. **Extracts Metadata**: Retrieves the table and column information from a metadata repository.
2. **Builds Dynamic Queries**: Constructs SQL queries based on the retrieved metadata.
3. **Loads Data**: Executes these queries to transfer data from source systems to the target data warehouse.

**Hints : for Solution**

1. **Setup Metadata Repository**

Create a metadata table in a database (e.g., SQL Server) named Table\_List with the following structure:

1. **Create Datasets in Azure Data Factory**
   * **Source Dataset**: Define a dataset for each source system (SQL Server, MySQL, Oracle).
   * **Target Dataset**: Define a dataset for the target data warehouse.
2. **Create the Pipeline**
   * **Get Metadata Activity**: Retrieve metadata from the Table\_List table.
     + **Source Dataset**: Dataset pointing to the metadata table.
     + **Field List**: Include Table\_name, Column\_Name, etc.
   * **ForEach Activity**: Iterate over each table entry retrieved from the metadata.
     + **Items**: Output from the Get Metadata activity.
     + **Activity**: Inside the ForEach loop, use an activity to build and execute dynamic SQL queries.
   * **Dynamic SQL Query Construction**:

Within the ForEach loop, use a combination of activities or a Stored Procedure to construct and execute dynamic queries.

**For example, construct an SQL query dynamically in a Stored Procedure:**

CREATE PROCEDURE LoadData

@SourceSystem NVARCHAR(128),

@TableName NVARCHAR(128),

@Columns NVARCHAR(MAX)

AS

BEGIN

DECLARE @SQL NVARCHAR(MAX)

SET @SQL = 'INSERT INTO ' + @TableName + ' (' + @Columns + ') ' +

'SELECT ' + @Columns + ' FROM ' + @SourceSystem + '.' + @TableName

EXEC sp\_executesql @SQL

END

* + - **Parameters**: Pass parameters for SourceSystem, TableName, and Columns based on the current iteration.

1. **Testing and Monitoring**
   * **Debug**: Test the pipeline with a subset of data to ensure queries are constructed and executed correctly.
   * **Monitor**: Use Azure Data Factory monitoring tools to track the execution status and handle any errors.

**Benefits**

* **Flexibility**: Automatically adapts to schema changes in source systems without requiring manual updates to the pipeline.
* **Efficiency**: Streamlines data integration from multiple sources, reducing manual effort and potential errors.
* Handles large volumes of data and multiple source systems effectively.

**Conclusion**

By leveraging dynamic queries and Azure Data Factory, RetailCo can efficiently manage data integration from various sources into their data warehouse, adapting to changes in source schemas and automating the data loading process. This approach enhances operational efficiency and ensures that data is consistently available for reporting and analytics.