**Q Link Data Engineer Test**

1. Describe in detail how you will setup a pipeline (AWS, GCP, Azure, Python script or other) to ingest data into a cloud datawarehouse or lakehouse (on either AWS, GCP or Azure) from an external transactional system that will provide data as csv files. Please also motivate the you rationale for choosing the specific tools and components you chose.

**Answer** :

Platform: Microsoft Azure – Azure is one of the top cloud service providers with rich services for data and analytics

I will use Azure Data Factory (ADF), assuming that Azure Synapse is the destination (data warehouse), the transactional system will load the CSV file in Azure Blob Storage, and Azure Services have already been created i.e. ADF, Blob Storage, Synapse, etc.

In ADF on the Author tab in the pipeline section, I create my pipeline following the naming standard of the company e.g. Source\_Destination\_TableName (SAP\_Synapse\_FIN-Sales). Once the pipeline has been created we will use a copy activity ( copy CSV to stg), firstly we will create a linked service to connect our ADF to our Blob storage, this is done in the manage tab, select linked service section, configure our linked service properties, data store type (Azure Blob Storage), we give our linked service a name (BlobStorage) we specify configuration details such as Authentication, Azure Subscription, on the account selection method we will select Enter Manually so we can parametrize our linked service, on storage account name we will enter our parametrize string e.g. ` @{concat('https://',linkedService().StorageAccountName,'.blob.core.windows.net/')}` this will allow us to enter storage account through a dataset and we test the connection.

Once the connection to blob is successful we return to the Author tab in the dataset section to create the dataset ( dataset defines the structure, location, and schema within the data store) and configure our storage type(Blob Storage) choose CSV as our format, dataset name (ds\_csv\_blob) select the linked service created previously. We then parametrize our storage account name (@dataset().StorageAccountName ) and the file path (@dataset().Container/ @dataset().Directory/ @dataset().FileName) this will allow us to provide these details dynamically, tick first row as a header (assuming our file will have a header) then create our DS.

Returning to our pipeline in the copy activity in the source tab we select our DS providing the parameters that we specified. For the sink, we will need another linked service and the dataset. For the linked service we will link with our Synapse providing configurations such as linked service name, on the account selection method we will select Enter Manually we then provide a Fully qualified domain name, database name, authentication type, and login details (Username and password if authentication is set to SQL authentication). With the dataset, we select synapse as our datastore and the lined service created previously and parametrize the table name.

We will use this new DS for the sink specifying Auto Create on the table option (this will automatically create the table for us) and providing the parameters. We will then use stored procedure activity. We can name it (STG to DW) First we will need to create our destination table on Synapse using T-SQL (

*CREATE TABLE [SCHEMA].[TABLE NAME]*

*([ID] IDENTITY (1,1) ,*

*[column1] [nvarchar](100) NULL,*

*[LoadDate] [datetime] NOT NULL )*

*)*

We will then move on to create our stored procedure, making sure that we only load new or updated data to our table (

*CREATE PROCEDURE [SCHEMA].[LoadNewOrUpdatedData]*

*AS*

*BEGIN*

*SET NOCOUNT ON;*

*INSERT INTO [SCHEMA].[TABLE NAME] ([column1], [LoadDate])*

*SELECT*

*src.[column1],*

*GETDATE() AS [LoadDate]*

*FROM*

*[SOURCE\_SCHEMA].[SOURCE\_TABLE] src*

*LEFT JOIN*

*[SCHEMA].[TABLE NAME] tgt*

*ON*

*src.[column1] = tgt.[column1]*

*WHERE*

*tgt.[column1] IS NULL -- New data*

*OR src.[ModifiedDate] > tgt.[LoadDate]; -- Updated data*

*END;*

*GO )*

Back to ADF to configure our stored procedure activity in the setting tab we specify our linked service to synapse (DW) we then select our stored procedure. We add one last copy activity to move the file from the landing folder to the processed folder in the blob. In the copy activity for the source we will use the same ds for csv we used previously and specify delete files after the completion tick box. We will use the same ds for the sink but point to the processed file folder this time.

**Pipeline enhancements:**

1. Data Governance and management – we can ensure the data quality by implementing audit features. In the pipeline, we can add 2 more stored procedure activities, 2 stored procedures, and an audit table. The audit table will contain columns such as ID, pipeline\_name, Source\_count, Target\_Count, start\_detetime, end\_datetime, status, error\_capture, source\_Table, target\_table, batch\_no, and loaddate. We will then create our stored procedures first [SCHEMA].[Audit\_insert], this stored procedure will capture Pipeline name, source\_count, start\_datetime, source\_table, batch\_no, and load date as parameters, and the dates we will use the SQL date functions. And the second stored procedure [SCHEMA].[Audit\_updtae] will capture the remaining columns. We will configure our stored procedure activities: 1 at the beginning of the pipeline and the other at the end of the pipeline.

*(ID - auto creted, pipeline\_name – from system parameters in the pipeline, Source\_count – can obtained from a copy activity output rows copied, Target\_Count – we can change the STG activity from stored procedure to lookup activity this will require a new dataset then modify our [SCHEMA].[LoadNewOrUpdatedData] stored procedure to count of rows in the target table, start\_detetime, end\_datetime, status – obtained from the move file to process based on its outcome(we can connect on success and also on fail passing in pass or fail status based on the outcome), error\_capture – capture the error in case of failure, source\_Table we can use the file name as parameter to the pipeline, target\_table create a parameter in the pipeline and pass it in, batch\_no – sequence number can be created that assign a unique number for each run and loaddate)*

1. Handle multiple files – before the audit insert activity, we can add a Get Metadata activity to retrieve file names in the folder we can also add a filter condition to filter by specific names that we want and then wrap all the remaining activity inside for each activity to loop through the files. This will be possible by creating parameters and variables in our pipeline
2. Describe how you would do the above if you have an additional database as a data source but this time you have the option to use an ODBC connector.

Answer: for this, we can create a linked service for ODBC sources, specifying all the required properties such as authentication type, log-in details database, etc, then create a dataset for the source database table similar to the one above and point it to a linked service. We then use a copy activity at the beginning of the pipeline (before the Get Metadata activity) for the source we specify our newly created ds that will fetch the data from the external database on the specified table.

For the sink, we will use the previous DS to point to the same folder that the above external transactional system will provide the file. This will be possible since we parametrized our DS. Once the files have been loaded into the folder, the rest of the pipeline will not change, and it will process all the files. We must make sure that the file names align with the names from the external transactional system so that they can be filtered correctly.

1. Describe what measures you would take to ensure data quality of your data warehouse\lakehouse for a reporting platform.

Answer: to ensure data quality in the Data Warehouse/Lakehouse it is essential to implement data validation and governance processes, this can be done by implementing data profiling at the early stage of ETL (before loading) to identify data inconsistency, anomalies, and duplicates. Data integrity checks can also be enforced during the ETL process this may include applying various constraints to ensure consistency across all related datasets. An automated data quality pipeline can be put in place to monitor errors like missing values, invalid formats, or outliers in real time so that they can be detected as early as possible. We can also Audit the pipeline regularly to identify and rectify bottlenecks or processes contributing to poor data quality.

A framework for data stewardship can be implemented to assign ownership and accountability of the datasets to promote a culture of responsibility. It is essential to enforce data standardization across various sources to maintain a single source of truth for reporting accurately. Using version control to track data schemas and transformation changes is important, avoiding unintended impacts on reports. Implement metadata management to enhance transparency and usability by clearly defining data definitions, lineage, and relationships. Lastly, conduct regular data reconciliation with source systems and leverage data observability tools to ensure that any data drift or schema changes are flagged before