Medallion Procedure

1. Create a Resource Group with the Nearest Region.
2. Create a Storage Account within the same resource group of the hierarchial namespace; Enable Public Access from all networks.
3. Create 3 storage containers → bronze, silver, gold; Create Data Factory within the same resource group and region.
4. Create Key Vault within the same resource group and region along with Vault Access Policy Model. Set the required permissions accordingly to use the Access Policy.
5. Launch Data Factory Studio and create a SQL database and SQL server with a suitable authentication username and password. Set a public endpoint and under Data Source, use existing data (AdventureWorksLT). Check the database for the data loaded.
6. Resolve the errors related to IP Permissions by configuring and allowing Azure Services to access this server.
7. Create Linked Service in Data Factory to connect to the database and one for Data Lake Storage Gen2. For the latter, test connection to file path (/bronze) and create the service after test connection.
8. Under author, create a new pipeline and a new dataset, under activities, search for lookup where the source dataset will be the dataset created before. Preview the data and save it. Later on, search for ForEach component, connect it with the lookup.
9. Create new dataset and under parameters add 2 cols (SchemaName, TableName), search for copy data and add dynamic data for each column
10. Now for storing the transformed format, Create new dataset (Data Lake Storage Gen2), Parquet format and a linked service. Under parameters add 2

columns (FolderName, FileName) and data is added dynamically. Now add this to Sink section in ForEach component.

11. Validate the pipeline for errors and then debug the pipeline.
12. Create an Azure Databricks instance with a standard pricing tier and the default region and launch the workspace as well as create the namespace and a notebook.
13. Create a secret scope for this notebook by using the url upto the numbers after '?o=' and adding #secrets/createScope.
14. Navigate to Storage Accounts → Access Keys → Copy 1 key after rotating it → Go to key vaults → Secrets → Paste the key here along with the name and create it.
15. Now move to the properties section, copy the vault URI, Resource ID to DNS Name, Resource ID of the scope, set a name and check for all users and create it.
16. Validate the code by connecting it to compute (choose single node cluster, Runtime 14.1 (Spark 3.5.0), Node type (Standard_DS3_v2) and create compute)
17. Execute the code written in databricks_base_notebook.py to validate the mounting.
18. Now we need to create a link between Data Factory and Databricks → Add Databricks Notebook to copy data and create a new linked service.
19. Account Selection Method → Enter Manually, Workspace URL from Notebook url upto (.net/)
20. For Access Token, go to Databricks settings → Developer → Manage → Generate Token → choose existing Interactive cluster → Test Connection → Create
21. Navigate to Base Notebook for Notebook path settings and add 3 base parameters → table_schema , table_name, fileName.
22. Under databricks console, go to Job Runs to check whether the pipeline is working; Navigate to Catalog, there will be a SalesLT database with the corresponding data

23. Databricks commands: Go to VS Code → Within folder → pip install dbt-databricks databricks-cli;(databricks configure --token) the url will be the notebook url upto .net; (databricks secrets list-scopes) List of Secret Scopes; (databricks fs ls) Check for catalog data connection
24. (dbt init) Initialize DBT to local/cloud, use databricks as database; Host → notebook url upto .net, For HTTP path → Go to compute → Cluster → Advanced Options → JDBC/ODBC; Use Access Token option, schema → saleslt
25. Create snapshots so as to retain the original data; create 2 Folders in models folder → Staging & Marts; Add the bronze.yml (staging) file with the logic to read & validate data into dbt
26. dbt debug and then dbt snapshot to see if the data is pushed into the silver layer
27. For the gold layer, create 3 folders → Customers, Products and Sales and their respective SQL, YML files (YML files are for test and schema structure and SQL is for transformation)
28. Commands for dbt → (dbt debug) for Fixing errors; (dbt run) for creating the transformed data and pushing it to gold layer; (dbt test) for testing the dbt pipeline; (dbt docs generate) to create the documentation; (dbt docs serve) to read/view the documentation at localhost:8080