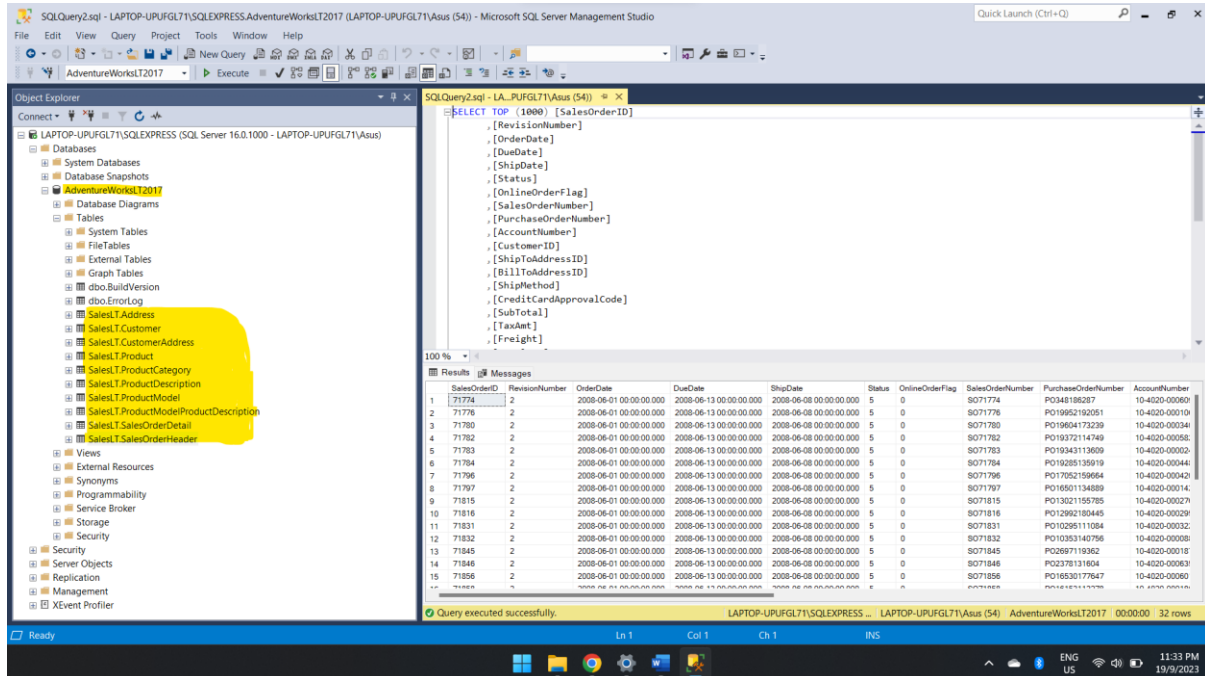
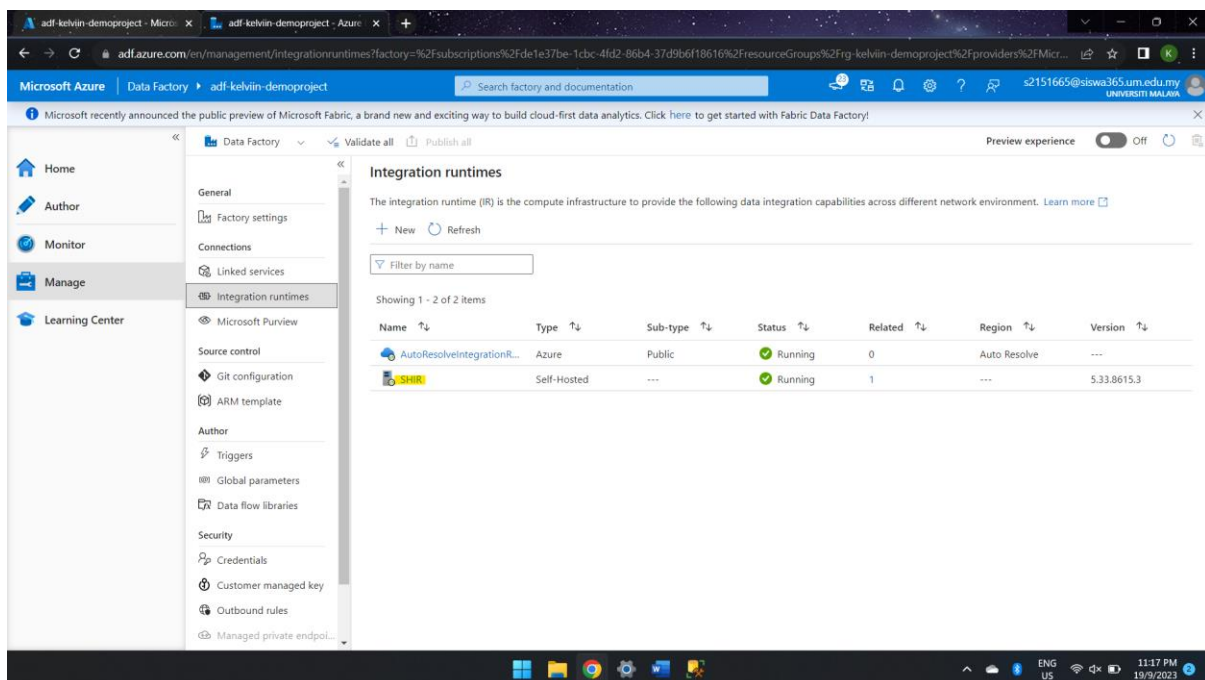


## PROJECT – END TO END DATA ENGINEERING PIPELINE IN MICROSOFT AZURE.

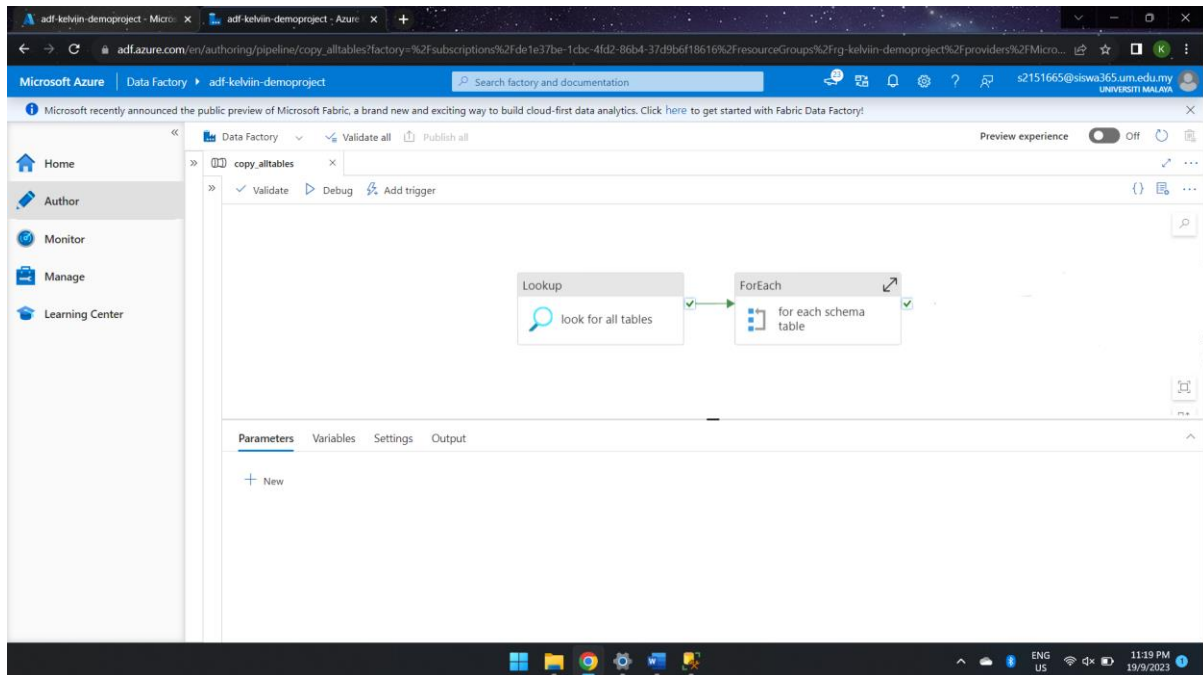
SQL Server and SSMS (SQL Server Management Studio) are installed in the local machine. **AdventureWorksLT2017** open-source database is restored into the SSMS and validated. This database will be migrated into Azure to perform data engineering tasks. Necessary **login credentials** to access this database are created here to be utilized in Azure.



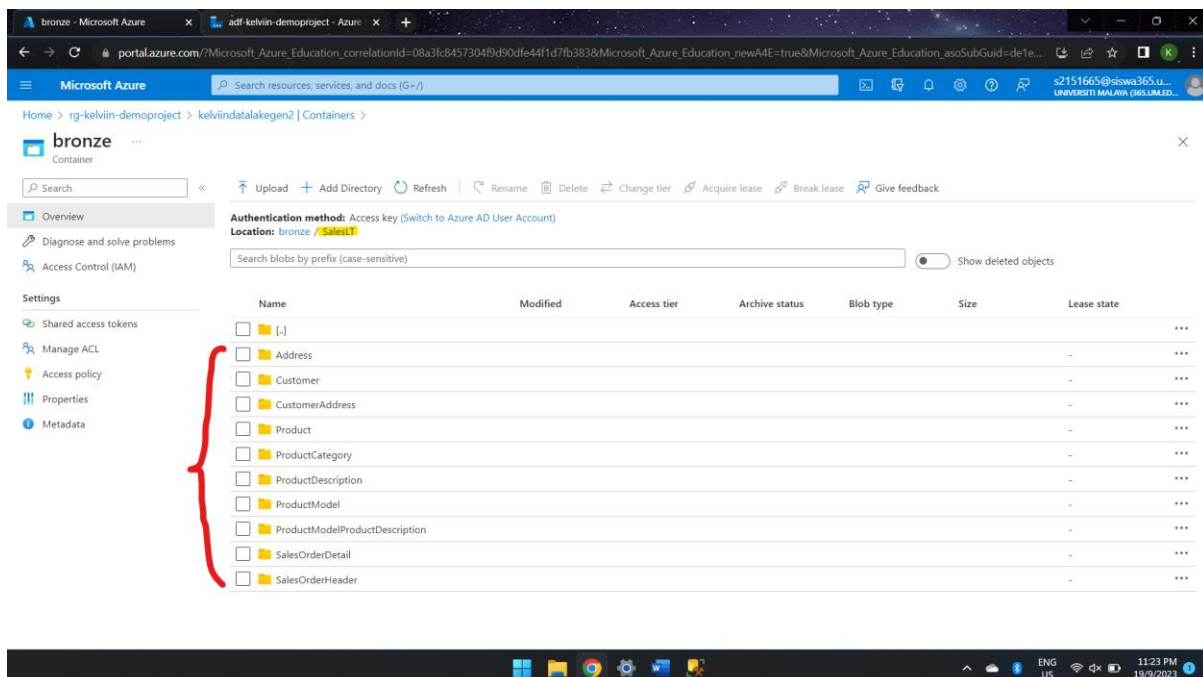
4 resources are created in the resource group (**Azure Data Factory, Storage Account, Azure DataBricks and Azure Synapse Analytics**). To establish connection between on-premise SQL server and Azure, we need to create a new **Self Hosted Integration Runtime (SHIR)** in the Azure Data Factory and install the SHIR in the local machine.



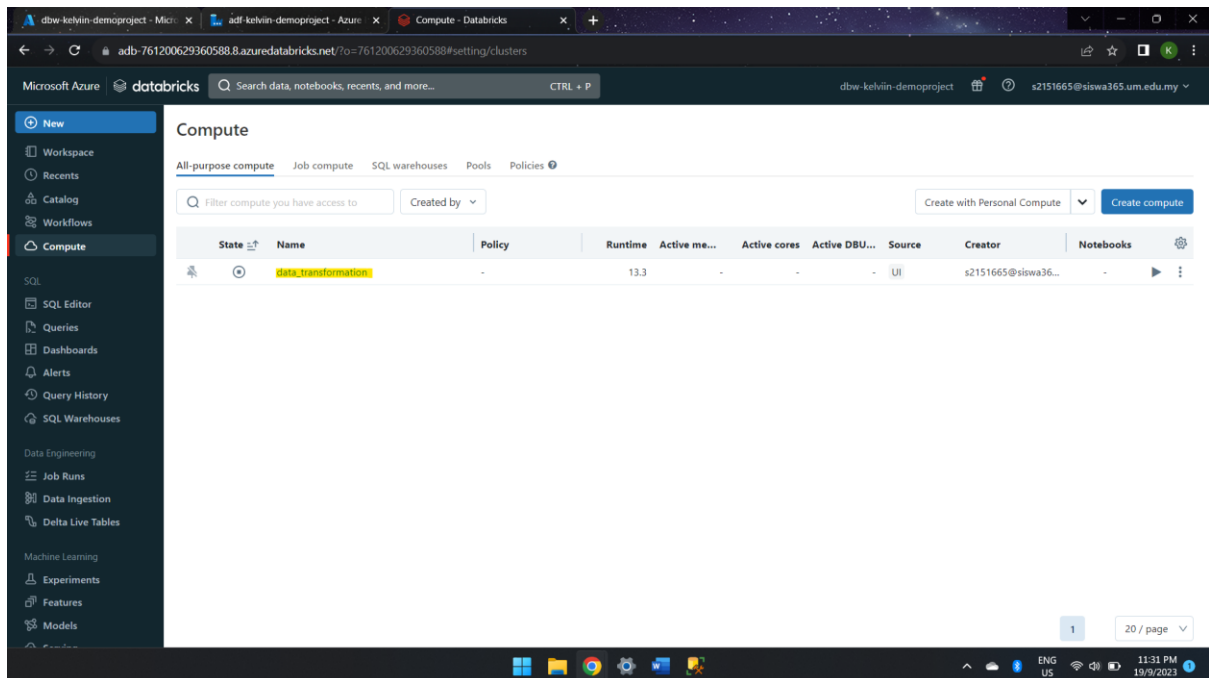
Create a new **pipeline** and drag the **lookup activity** and set the source referring to the **on-prem SQL server database**. Insert the **SQL script** into the query box in the setting to obtain information on the schema name and the table name. Next, drag the **foreach activity** and connect with the lookup activity on success. This foreach activity will **iterate through the output list from lookup**. This can be done by **updating the foreach activity settings (add dynamic content)**. Within the foreach activity, the **copy data activity** is added to copy all tables from the database. **Source (on-prem) and sink (data lake storage)** are inserted, settings are modified to **select all tables from the database and place these tables in a proper folder structure in the Data Lake** in bronze folder.



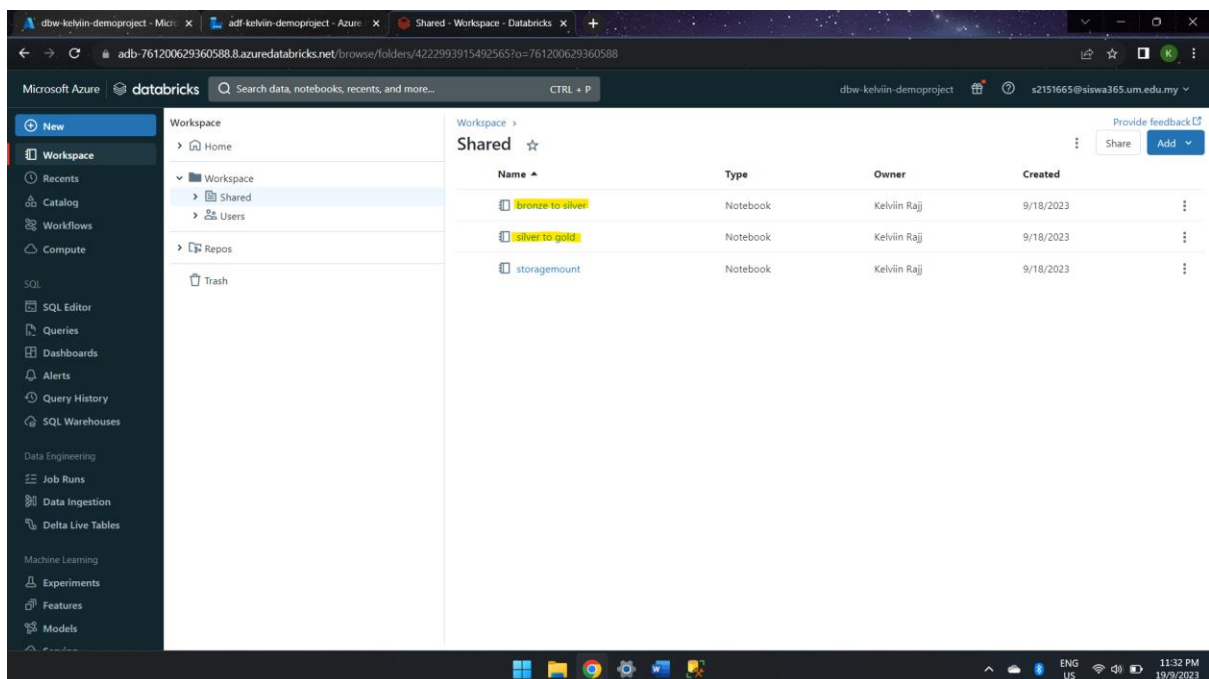
Once pipeline is executed, the tables will be **exported to the Data Lake** in the folder structure as below. (bronze/SchemaName/TableName/TableName.parquet)



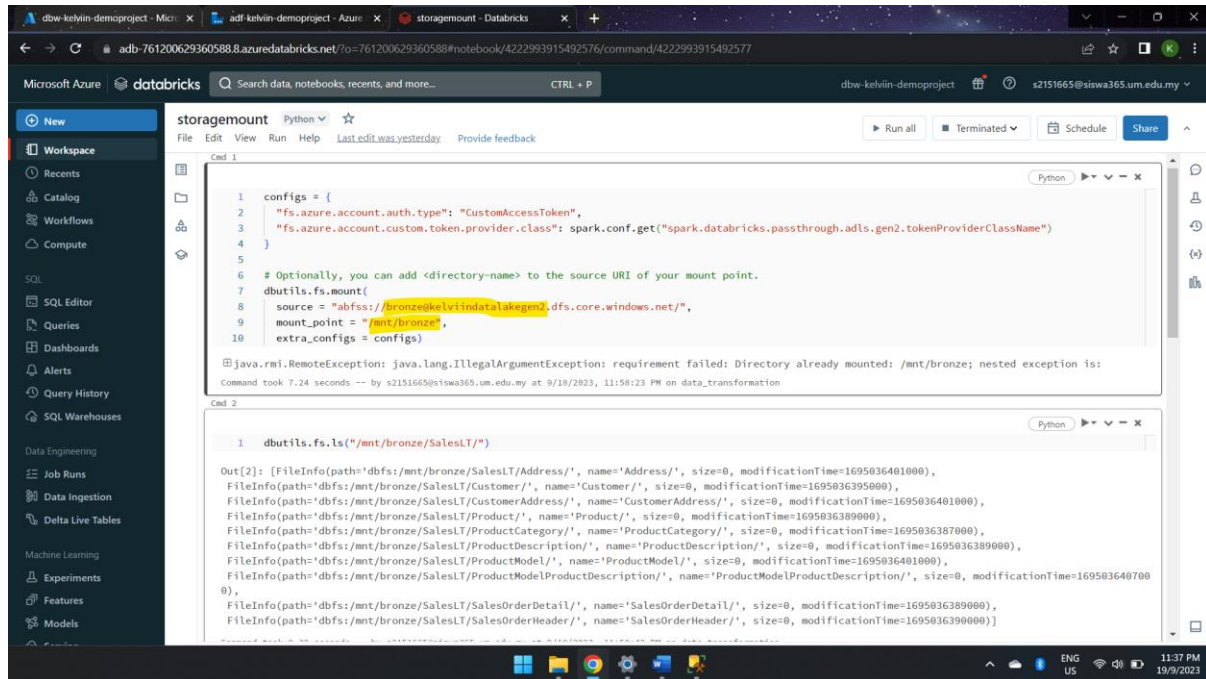
Login to Azure DataBricks and **create a new Spark compute cluster** to perform data transformation tasks on the SalesLT tables in the Data Lake. **Enable passthrough credentials** in the cluster to connect to Data Lake storage.



Create 3 new notebooks in the workspace. **Storage mount notebook** (to mount our Data Lake Storage to DataBricks), **bronze to silver notebook** (to perform level 1 data transformation) and **silver to gold notebook** (to perform level 2 data transformation).



The storage mount notebook will be used to **mount the storage account in Data Lake into DataBricks**. Code is inserted with **updated container name and storage account name**. Mount point name is updated with container path details. Dbutils script is executed with bronze container details inserted to verify the folder names in the Data Lake. Similar storage mount **scripts are executed for silver and gold containers**.

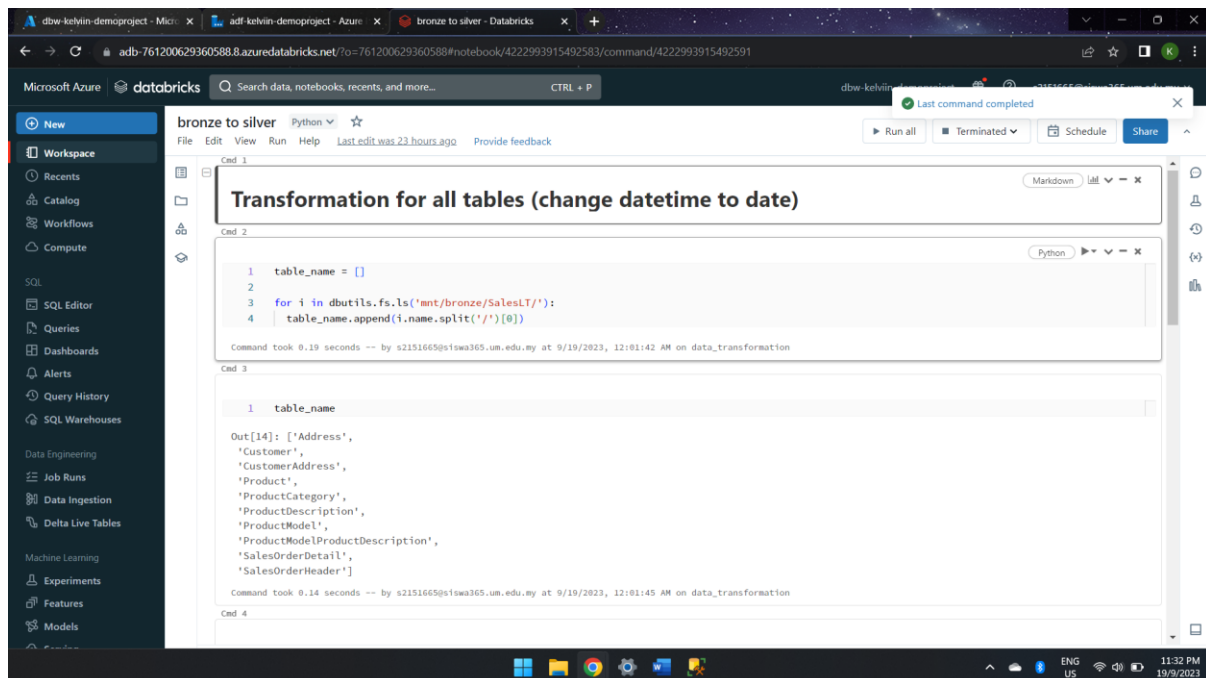


The screenshot shows a Databricks notebook interface with the title 'storage mount'. The code is written in Python and consists of two cells. Cell 1 defines a configuration dictionary and uses `dbutils.fs.mount` to mount the storage account. Cell 2 uses `dbutils.fs.ls` to list the contents of the mounted directory. The output of Cell 2 shows a list of files and directories in the `/mnt/bronze/SalesLT/` directory.

```
1 configs = {
2     "fs.azure.account.auth.type": "CustomAccessToken",
3     "fs.azure.account.custom.token.provider.class": spark.conf.get("spark.databricks.passthrough.adls.gen2.tokenProviderClassName")
4 }
5
6 # Optionally, you can add <directory-name> to the source URI of your mount point.
7 dbutils.fs.mount(
8     source = "abfss://bronze@kelvinadatalakegen2.dfs.core.windows.net/",
9     mount_point = "/mnt/bronze",
10    extra_configs = configs)
11
12 java.rmi.RemoteException: java.lang.IllegalArgumentException: requirement failed: Directory already mounted: /mnt/bronze; nested exception is:
13 Command took 7.24 seconds -- by s2151665@siwa365.um.edu.my at 9/18/2023, 11:58:23 PM on data_transformation
```

```
1 dbutils.fs.ls("/mnt/bronze/SalesLT/")
2
3 Out[2]: [FileInfo(path='/mnt/bronze/SalesLT/Address/', name='Address/', size=0, modificationTime=1695036401000),
4 FileInfo(path='/mnt/bronze/SalesLT/Customer/', name='Customer/', size=0, modificationTime=1695036395000),
5 FileInfo(path='/mnt/bronze/SalesLT/CustomerAddress/', name='CustomerAddress/', size=0, modificationTime=1695036401000),
6 FileInfo(path='/mnt/bronze/SalesLT/Product/', name='Product/', size=0, modificationTime=1695036389000),
7 FileInfo(path='/mnt/bronze/SalesLT/ProductCategory/', name='ProductCategory/', size=0, modificationTime=1695036387000),
8 FileInfo(path='/mnt/bronze/SalesLT/ProductDescription/', name='ProductDescription/', size=0, modificationTime=1695036389000),
9 FileInfo(path='/mnt/bronze/SalesLT/ProductModel/', name='ProductModel/', size=0, modificationTime=1695036401000),
10 FileInfo(path='/mnt/bronze/SalesLT/ProductModelProductDescription/', name='ProductModelProductDescription/', size=0, modificationTime=1695036407000),
11 FileInfo(path='/mnt/bronze/SalesLT/SalesOrderDetail/', name='SalesOrderDetail/', size=0, modificationTime=1695036390000),
12 FileInfo(path='/mnt/bronze/SalesLT/SalesOrderHeader/', name='SalesOrderHeader/', size=0, modificationTime=1695036390000)]
```

Level 1 data transformation: Python script with bronze container mount point is prepared to transform the date columns in all tables to **convert the format from datetime format to date-only format**.



The screenshot shows a Databricks notebook interface with the title 'bronze to silver'. The code is written in Python and consists of four cells. Cell 1 is a title. Cell 2 defines a list of table names. Cell 3 shows the output of the list. Cell 4 shows the output of the list.

```
1 table_name = []
2
3 for i in dbutils.fs.ls("/mnt/bronze/SalesLT/"):
4     table_name.append(i.name.split('/')[0])
5
6 Command took 0.19 seconds -- by s2151665@siwa365.um.edu.my at 9/19/2023, 12:01:42 AM on data_transformation
```

```
1 table_name
2
3 Out[14]: ['Address',
4 'Customer',
5 'CustomerAddress',
6 'Product',
7 'ProductCategory',
8 'ProductDescription',
9 'ProductModel',
10 'ProductModelProductDescription',
11 'SalesOrderDetail',
12 'SalesOrderHeader']
13
14 Command took 0.14 seconds -- by s2151665@siwa365.um.edu.my at 9/19/2023, 12:01:45 AM on data_transformation
```

As observed below, the datetime data in all columns with date values have been transformed to display date-only data. The **output data is exported into a delta format into the silver mount point**.

dbw-kelvin-demoproject - Micro x adf-kelvin-demoproject - Azure x bronze to silver - Databricks x +

adb-761200629360588.8.azure.databricks.net/?o=761200629360588#notebook/4222993915492583/command/4222993915492591

Microsoft Azure databricks Search data, notebooks, recents, and more... CTRL + P dbw-kelvin-demoproject s2151665@siswa365.um.edu.my

New Workspace Recents Catalog Workflows Compute SQL SQL Editor Queries Dashboards Alerts Query History SQL Warehouses Data Engineering Job Runs Data Ingestion Delta Live Tables Machine Learning Experiments Features Models

bronze to silver Python ☆ File Edit View Run Help Last edit was 23 hours ago Provide feedback Run all Terminated Schedule Share

Cnd 5

```
1 display(df)
```

(1) Spark Jobs

	SalesOrderID	RevisionNumber	OrderDate	DueDate	ShipDate	Status	OnlineOrderFlag	SalesOrderNumber	PurchaseOrderNumber
1	71774	2	2008-06-01	2008-06-13	2008-06-08	5	false	SO71774	PO348186287
2	71776	2	2008-06-01	2008-06-13	2008-06-08	5	false	SO71776	PO19952192051
3	71780	2	2008-06-01	2008-06-13	2008-06-08	5	false	SO71780	PO19604173239
4	71782	2	2008-06-01	2008-06-13	2008-06-08	5	false	SO71782	PO19372114749
5	71783	2	2008-06-01	2008-06-13	2008-06-08	5	false	SO71783	PO19343113609
6	71784	2	2008-06-01	2008-06-13	2008-06-08	5	false	SO71784	PO19285135919
7	71796	2	2008-06-01	2008-06-13	2008-06-08	5	false	SO71796	PO17052159664

32 rows | 0.42 seconds runtime Refreshed 23 hours ago

Command took 0.42 seconds -- by s2151665@siswa365.um.edu.my at 9/19/2023, 12:03:31 AM on data\_transformation

Shift+Enter to run  
Shift+Ctrl+Enter to run selected text

Level 2 data transformation: Python script with silver container mount point is prepared to **transform the column headers in all tables into an appropriate format** for convenient SQL processing in Synapse Analytics. (with **'\_'** inserted in column headers) (**ListPrice > List\_Price**)

dbw-kelvin-demoproject - Micro x adf-kelvin-demoproject - Azure x silver to gold - Databricks x +

adb-761200629360588.8.azure.databricks.net/?o=761200629360588#notebook/4222993915492595/command/4222993915492603

Microsoft Azure databricks Search data, notebooks, recents, and more... CTRL + P dbw-kelvin-demoproject s2151665@siswa365.um.edu.my

New Workspace Recents Catalog Workflows Compute SQL SQL Editor Queries Dashboards Alerts Query History SQL Warehouses Data Engineering Job Runs Data Ingestion Delta Live Tables Machine Learning Experiments Features Models

silver to gold Python ☆ File Edit View Run Help Last edit was 23 hours ago Provide feedback Run all Terminated Schedule Share

Cnd 1

### Transformation for all tables (change all column names)

Cnd 2

```
1 table_name = []
2
3 for i in dbutils.fs.ls("mnt/silver/SalesLT/"):
4     table_name.append(i.name.split('/')[0])
```

Command took 0.17 seconds -- by s2151665@siswa365.um.edu.my at 9/19/2023, 12:19:44 AM on data\_transformation

Cnd 3

```
1 table_name
```

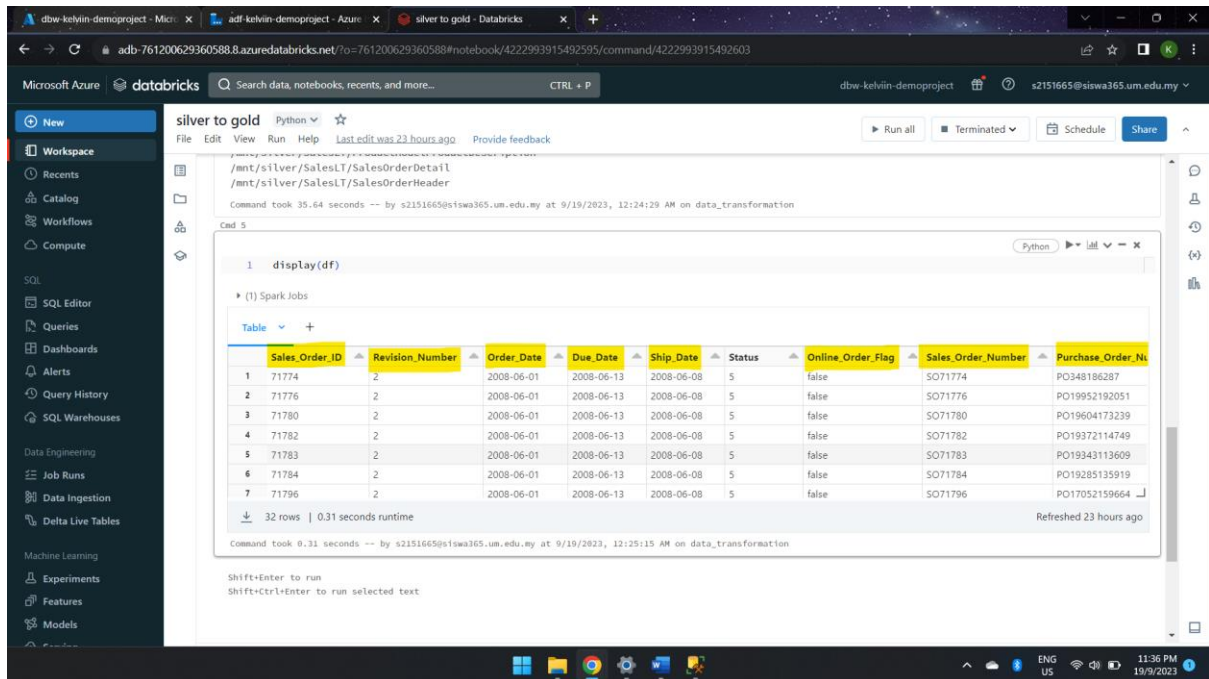
Out[11]: ['Address', 'Customer', 'CustomerAddress', 'Product', 'ProductCategory', 'ProductDescription', 'ProductModel', 'ProductModelProductDescription', 'SalesOrderDetail', 'SalesOrderHeader']

Command took 0.08 seconds -- by s2151665@siswa365.um.edu.my at 9/19/2023, 12:19:47 AM on data\_transformation

Cnd 4



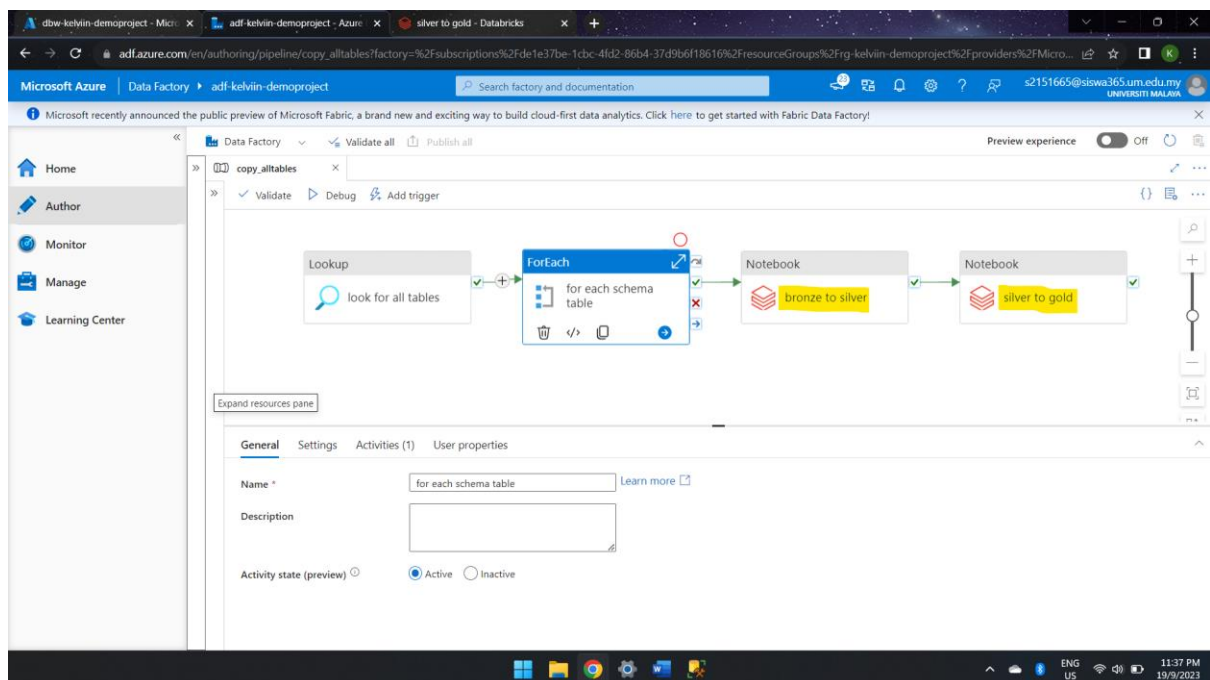
As observed below, the column names in all tables have been transformed as intended with ‘\_’ inserted. The output of the data is exported in delta format into the gold container.



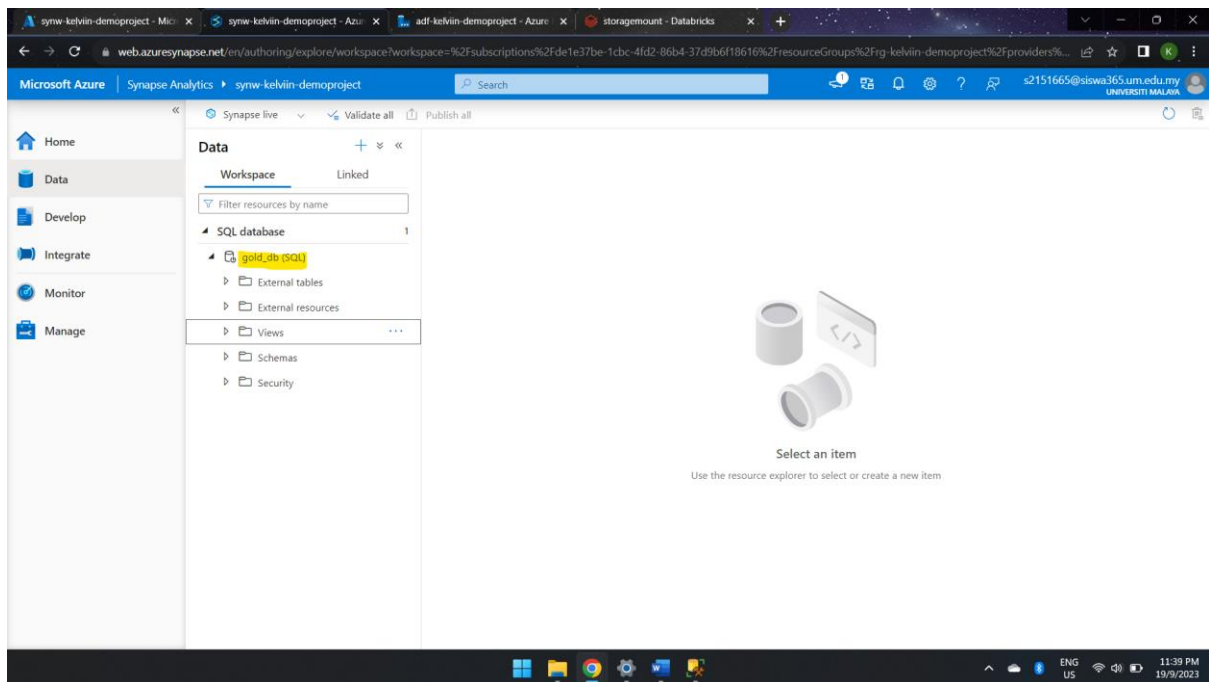
The screenshot shows a Databricks notebook interface. The notebook is titled 'silver to gold' and is running a Python command. The output is a table with 10 columns: Sales\_Order\_ID, Revision\_Number, Order\_Date, Due\_Date, Ship\_Date, Status, Online\_Order\_Flag, Sales\_Order\_Number, and Purchase\_Order\_N. The table contains 7 rows of data. The command took 0.31 seconds to run.

	Sales_Order_ID	Revision_Number	Order_Date	Due_Date	Ship_Date	Status	Online_Order_Flag	Sales_Order_Number	Purchase_Order_N
1	71774	2	2008-06-01	2008-06-13	2008-06-08	5	false	SO71774	PO348186287
2	71776	2	2008-06-01	2008-06-13	2008-06-08	5	false	SO71776	PO19952192051
3	71780	2	2008-06-01	2008-06-13	2008-06-08	5	false	SO71780	PO19604173239
4	71782	2	2008-06-01	2008-06-13	2008-06-08	5	false	SO71782	PO19372114749
5	71783	2	2008-06-01	2008-06-13	2008-06-08	5	false	SO71783	PO19343113609
6	71784	2	2008-06-01	2008-06-13	2008-06-08	5	false	SO71784	PO19285135919
7	71796	2	2008-06-01	2008-06-13	2008-06-08	5	false	SO71796	PO17052159664

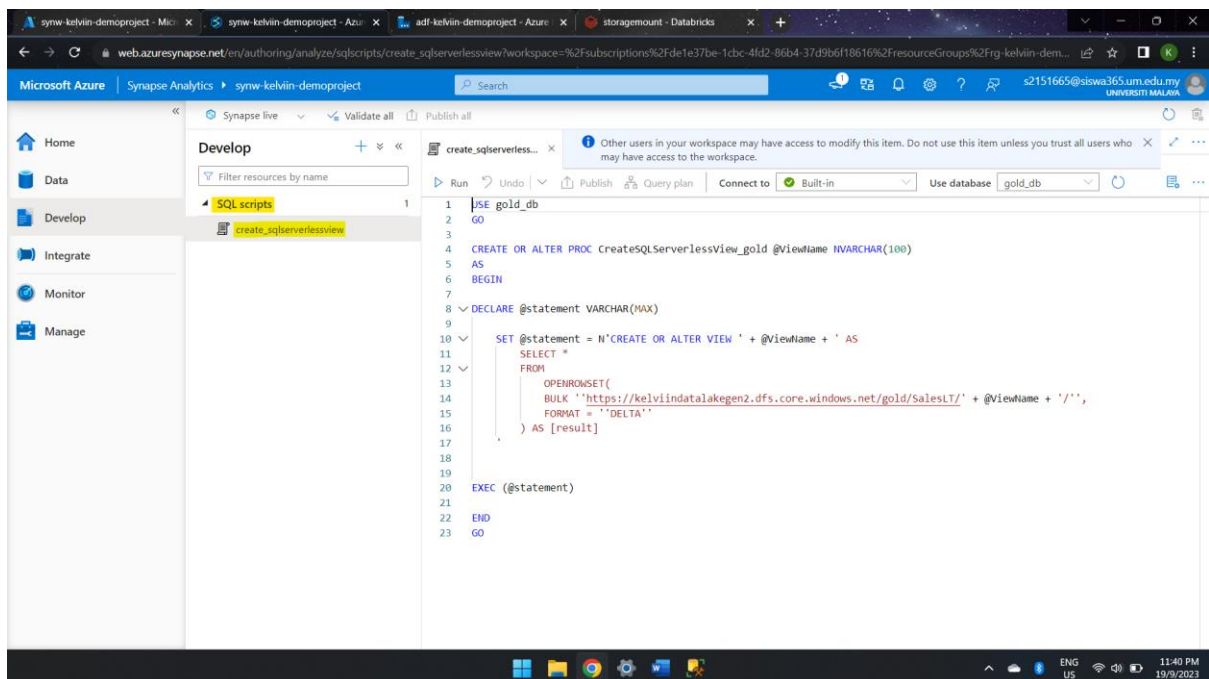
Coming back to Data Factory, **DataBricks** is added into a **new linked service**. Both Databricks notebooks (bronze to silver & silver to gold) are **connected into the existing pipeline**. Once pipeline is executed, all folders with transformed tables will be visible in the **silver container (datetime transformed)** and **gold container (datetime and column headers transformed)** in the Data Lake storage. The tables in the gold container will be utilized for subsequent processing tasks.



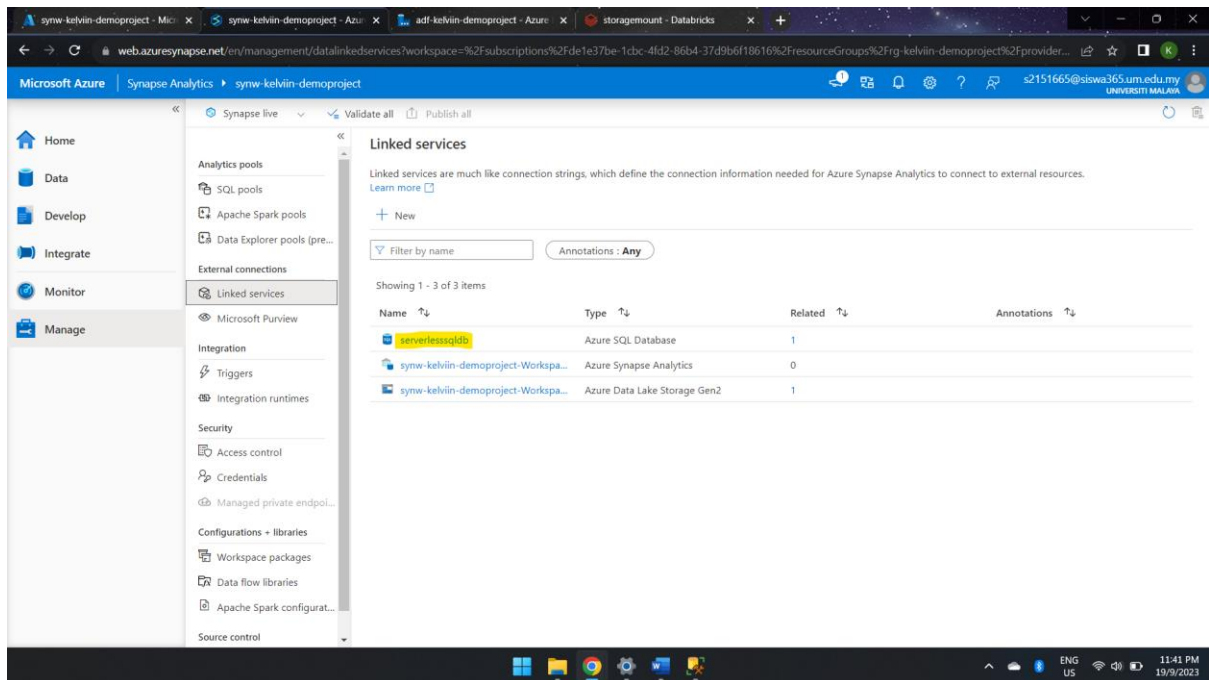
Open Synapse Analytics workspace and **add a new serverless SQL database named gold\_db**.



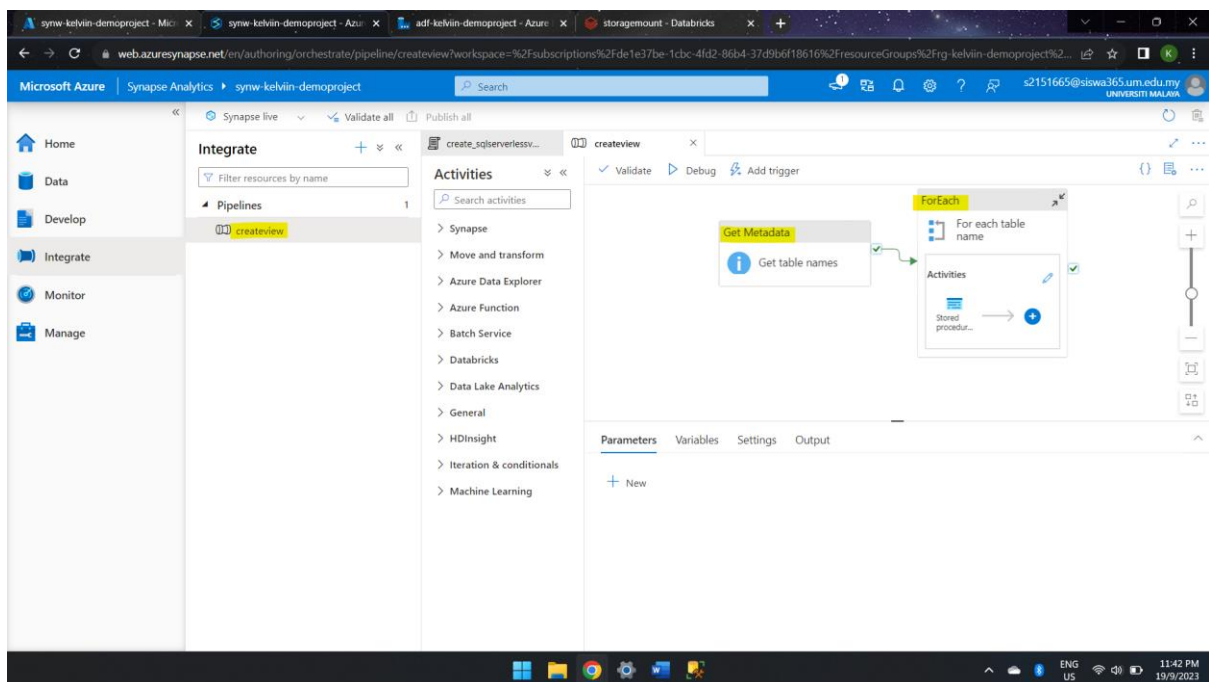
Create a **SQL script** to generate a **stored procedure** with parameters that can **dynamically create views** for all tables in the gold container and publish the changes to save the script.



Next, a **new pipeline** will be created in Synapse that will **use the stored procedure to create views** in the serverless SQL database. A **new linked service connection is added (Azure SQL Database)** with correct information inserted and changes are published to save the new linked service.

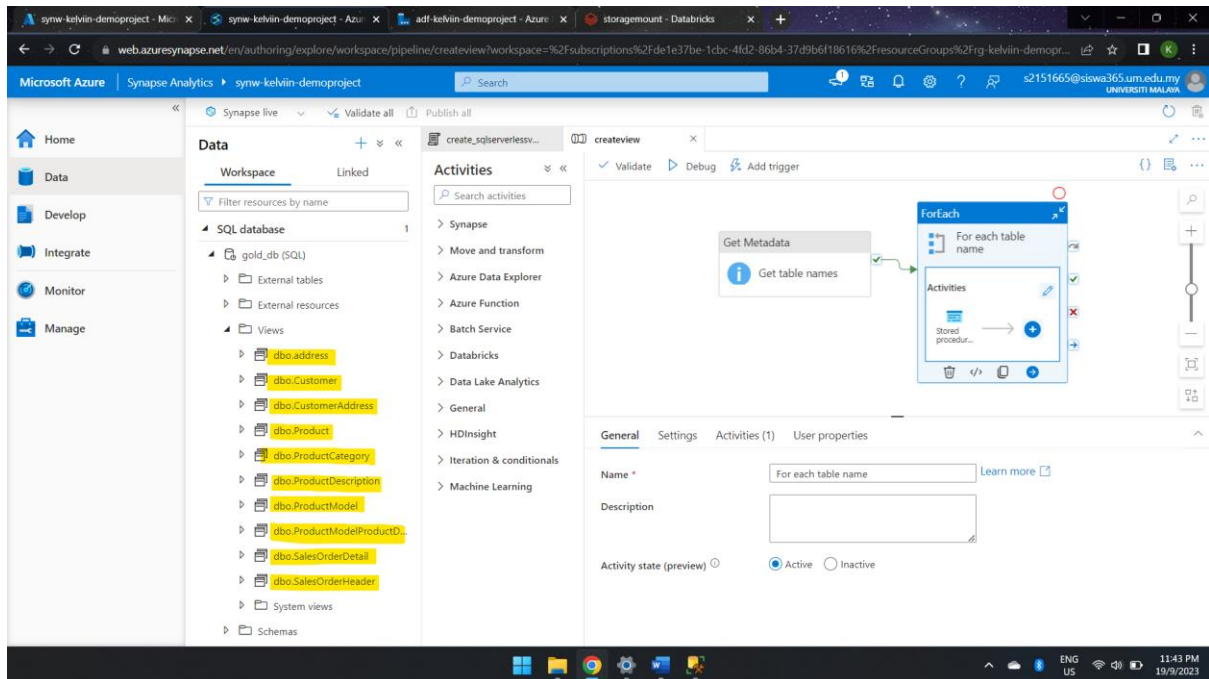


Get **metadata activity** to list table names in the Data Lake. Select Data Lake storage in setting (**binary format**) with **linked service connection** selected and **specify gold container** to get information from and select **childitem** in field list. Next **foreach activity** is added to **iterate table names to create the view** dynamically from the output of the get metadata activity. Inside the foreach activity, **add a stored procedure activity**. Here the **linked service SQL DB** and **stored procedure name** (SQL script to create serverless view) is selected. Add view name in parameter with `item.name()` inserted in dynamic content.

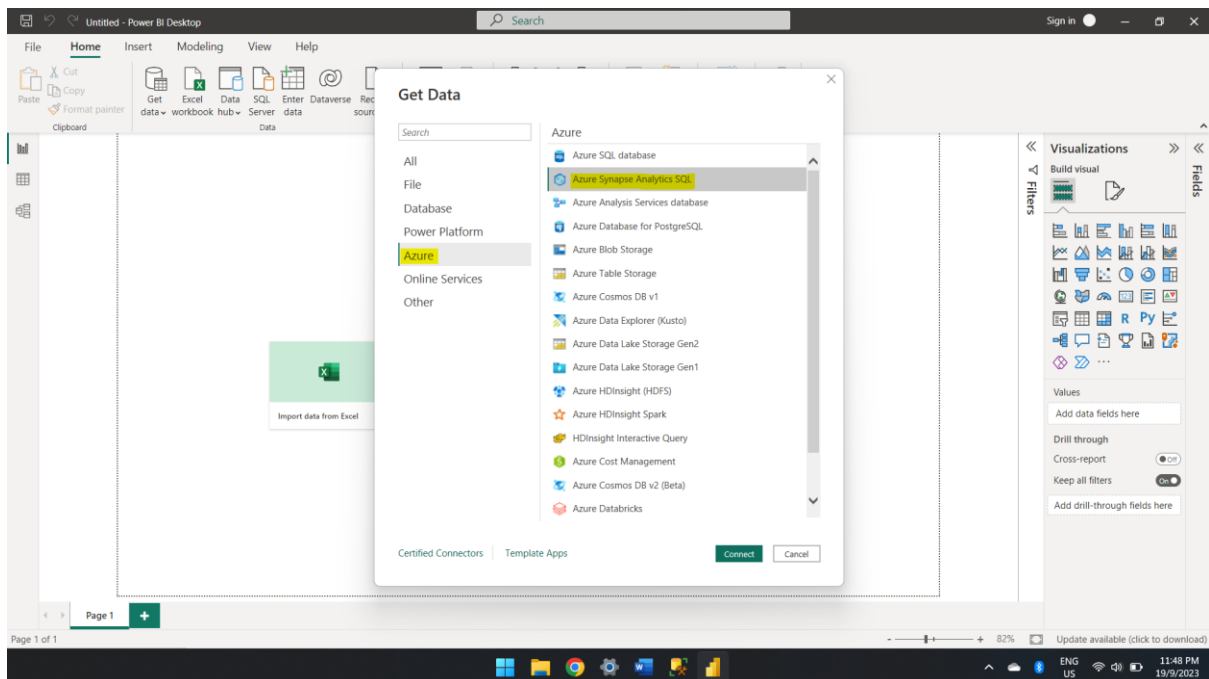




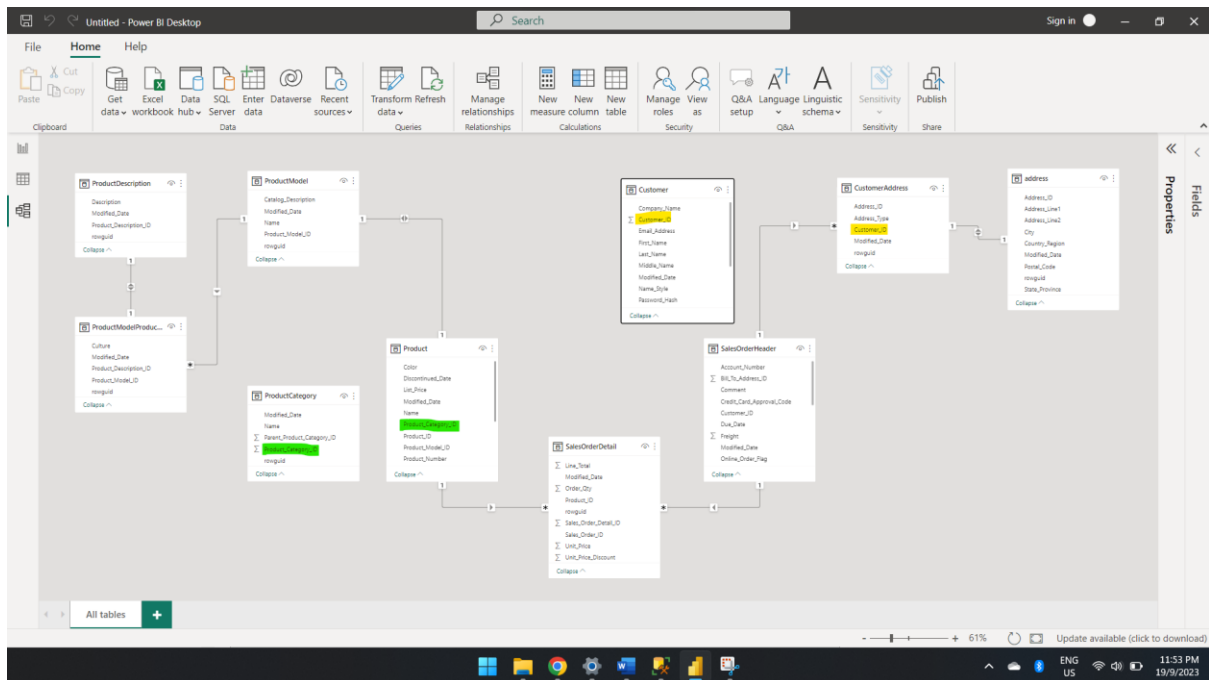
As a result of the pipeline triggered, the tables in the gold container will be **visible in the new serverless SQL database** as observed below.



As a last step in the data engineering pipeline, **Microsoft PowerBI** will be linked to the Azure Synapse to create insightful dashboards based on the information in the database. Specify **server endpoint** and **sign in with Microsoft credentials** and import the data into PowerBI.



Microsoft PowerBI by default is capable to establish relationships between tables that have matching column headers. For tables with relationships not created, **manual establishment of relationship** between tables is performed to connect the tables to create a matured data model.



As observed **all tables in the database have been imported** into the PowerBI workspace and we can **generate intuitive visualizations** (graphs, charts, data cards, etc.) by dragging columns from the tables into the respective data fields for the use of **business intelligence operations**.

