
Business Requirement:

A company XYZ, operating in the **Cycling and Outdoor Sports Retail** sector, aims to gain deeper insights into customer demographics and their purchasing patterns. Specifically, stakeholders are interested in understanding the **gender distribution of customers** and its impact on sales across various product categories. The company's product portfolio includes a diverse range of cycling-related items, from **bikes and accessories** to **clothing and components**.

Key requirements include:

1. **KPI Dashboard:**
 - Insights into **sales by gender** and **product category**, highlighting:
 - Total products sold.
 - Total sales revenue.
 - Gender split of customers.
2. **Dynamic Filtering:**
 - Allow stakeholders to filter data by **product category**, **gender**, and **date range**.
3. **Real-time Access:**
 - Provide up-to-date data through an automated and scalable pipeline.

The company's existing data resides in an on-premises SQL database, requiring a robust and efficient system to process, analyze, and visualize the information.

Solution:

To address these business requirements, I proposed the following solution:

1. **Data Extraction and Ingestion:**
 - Extract sales and customer data from the on-premises SQL database using **Azure Data Factory (ADF)**.
 - Store the raw data securely in **Azure Data Lake Storage (ADLS)** for centralized processing.
2. **Data Transformation:**
 - Use **Azure Databricks** with **PySpark** to cleanse and transform the data.
 - Key transformations include:
 - Mapping product categories (e.g., **Bikes, Accessories, Clothing, Components**).
 - Aggregating KPIs such as:
 - Total products sold by gender and category.
 - Total revenue by gender and category.
 - Percentage breakdown of gender distribution.
3. **Data Modeling and Architecture:**
 - Implement the **Medallion Architecture**:

- **Bronze Layer:** Raw data directly extracted from the SQL database.
- **Silver Layer:** Cleansed and structured data with enriched product and customer details.
- **Gold Layer:** Aggregated KPI data, ready for reporting and dashboarding.

4. KPI Dashboard Development:

- Build a **Power BI dashboard** with:
 - Visualizations for total sales, products sold, and gender distribution.
 - Interactive slicers for filtering by **gender, product category, and date**.
 - Line charts and bar graphs showing trends over time.
 - Pie charts for gender distribution within product categories.

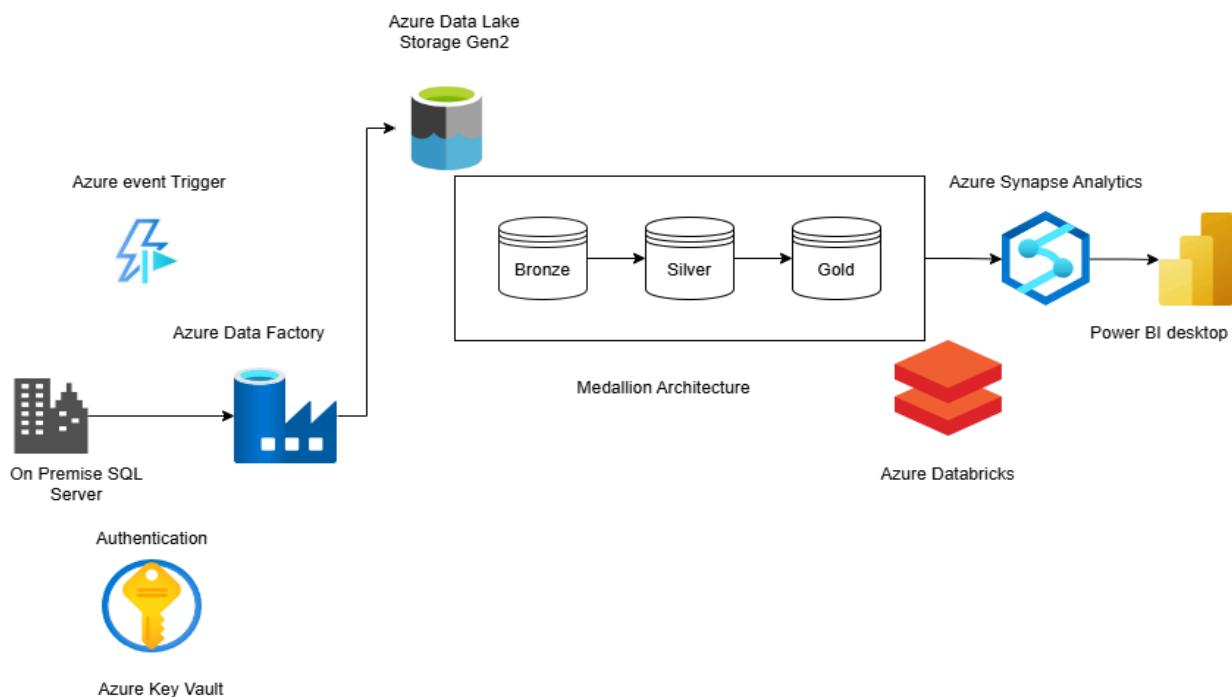
5. Pipeline Automation:

- Schedule daily pipeline runs using **ADF triggers** to ensure real-time data updates.

6. Governance and Security:

- Secure credentials using **Azure Key Vault**.
- Manage access with **Azure Active Directory** to ensure data confidentiality and integrity.

Architecture Diagram:



I provisioned the resources in Azure as per the above architecture in a resource group. Once the resources are provisioned, I can see the below image in resource groups in azure

The screenshot shows the Microsoft Azure portal at portal.azure.com/#browse/resourcegroups. The page title is "Resource groups". It displays a list of four resource groups: "databricks-rg-wsanalyticsspace-mnekoc4owaun2", "dev-rg", "NetworkWatcherRG", and "synapseworkspace-managedrg-55b5d0e9-ea23-4974-93df-ba2d0f03a7c4". Each group is associated with a "Free Trial" subscription and located in "UK South" or "South Central US". The interface includes a search bar, filter options (Subscription equals all, Location equals all), and a toolbar with "Create", "Manage view", "Refresh", "Export to CSV", "Open query", and "Assign tags". A message at the top indicates "You are viewing a new version of Browse experience. Some features may be missing. Click here to access the old experience." Below the table, it says "Showing 1 - 4 of 4. Display count: 10".

I created a user in the on-prem database and the user credentials will be saved in the key vault. The ADF will use the credentials in the key vault to access the on prem database.

Before creating a secret, I need to create a role that has the ability to create a secret. Then I assigned this role to myself.

Go to IAM, click ADD, add role assignment, click on :"Azure Key Vault Administrator", click "Next". Select "User, group or service principal". Select "Members". Then I selected my username and assigned it to myself (Review and assign)

Now, go to secret, now I have the ability to create a secret.click create and then give the details of username and password and click create

Go to Azure Data Factory, select the new pipeline, name the pipeline.
Select the copy data activity in ADF

In this case, our source is an on-prem SQL database. Go to the source section and click on "New". I was prompted to select a new data store. Select "SQL Server" from the list, Now, I was asked to select a linked service. Select "New". name the linked service

In the integration runtime section, select "New", select "Self Hosted Integration Runtime". Name the integration runtime, click "Create"

Option 1: Express setup

[Click here to launch the express setup for this computer](#)

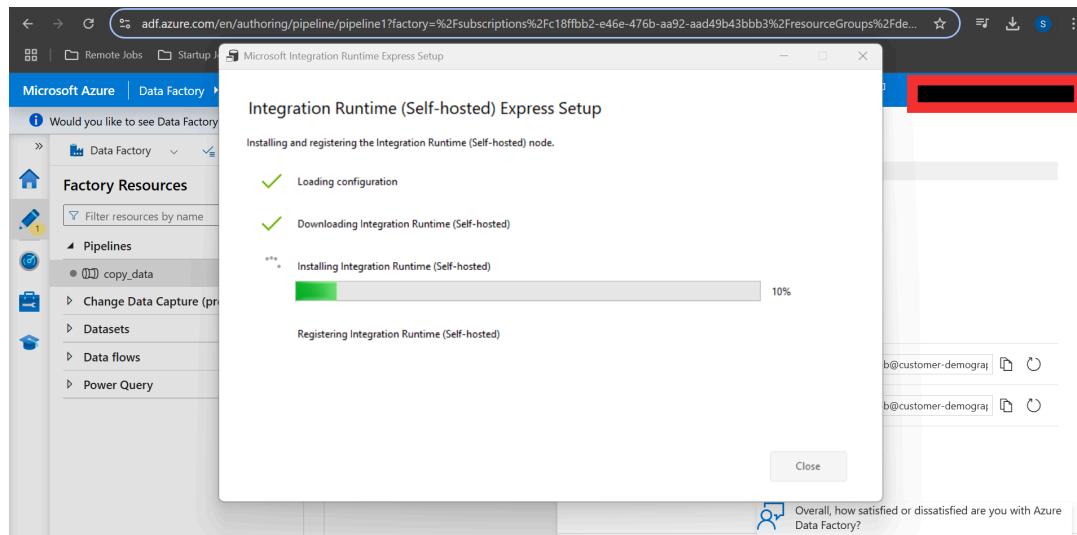
Option 2: Manual setup

Step 1: [Download and install integration runtime](#)

Step 2: Use this key to register your integration runtime

Name	Authentication key
Key1	IR@e30a6d05-799a-4c59-a34e-4154bc55405b@customer-demogra...
Key2	IR@e30a6d05-799a-4c59-a34e-4154bc55405b@customer-demogra...

From the above options, click “Express Setup”. A program will be downloaded. Run and execute the program



Once it is completed,I found the below screen

New linked service

 SQL server [Learn more](#)

Name *

SqlServeronpremlinkedservice

Description

Connect via integration runtime * ⓘ

 selfhostedintegrationRuntime1



Version

Recommended Legacy

 Import from connection string

Server name *

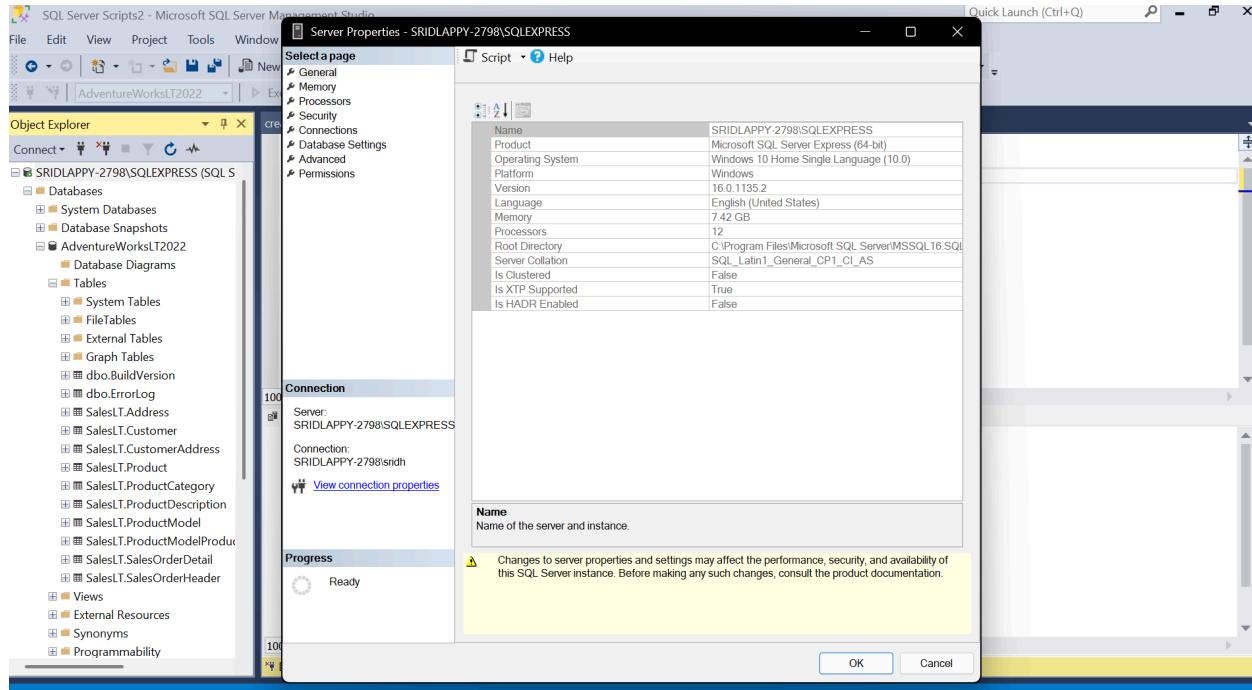
Database name *



Overall, how satisfied or dissatisfied are you with Azure?

Now, I need to populate the server name and database credentials of the on-premise SQL server. I can find the details from the SQL Server Management Studio.

If I click on the “properties” of the database in SQL Server Management Studio, I can find the below screen



Server Properties - SRIDLAPPY-2798\SQLEXPRESS

Select a page: General, Memory, Processors, Security, Connections, Database Settings, Advanced, Permissions

General

Name	SRIDLAPPY-2798\SQLEXPRESS
Product	Microsoft SQL Server Express (64-bit)
Operating System	Windows 10 Home Single Language (10.0)
Platform	Windows
Version	16.0.1135.2
Language	English (United States)
Memory	7.42 GB
Processors	12
Root Directory	C:\Program Files\Microsoft SQL Server\MSSQL16.SQL
Server Collation	SQL_Latin1_General_CI_AS
Is Clustered	False
Is XTP Supported	True
Is HADR Enabled	False

Connection

Server: SRIDLAPPY-2798\SQLEXPRESS
Connection: SRIDLAPPY-2798\sridh
[View connection properties](#)

Progress

Ready

Warning: Changes to server properties and settings may affect the performance, security, and availability of this SQL Server instance. Before making any such changes, consult the product documentation.

OK Cancel

Take the server name from the details and paste it in the “New Linked Service” as below

For the database name, take the name from sidebar and paste it in linked service section as below

For the username , I have to give the details of the username of the user which I have created earlier in SQL Server Studio.

For the password, I can leverage the Azure Key Vault in which I have created secrets

The screenshot shows the Microsoft Azure Data Factory interface. On the left, the 'Factory Resources' sidebar lists 'Pipelines' (copy_data), 'Datasets', 'Data flows', and 'Power Query'. The main area displays a pipeline named 'copy_data' with a single step named 'Copy data1'. The 'Source' tab is selected. On the right, a 'New linked service' dialog is open for an 'SQL server' type. The 'Authentication type' is set to 'SQL authentication'. The 'User name' field contains 'sreedhar'. The 'Password' tab is selected, while 'Azure Key Vault' is also present. Under 'AKV linked service', 'Select...' is shown. The 'Secret name' dropdown is set to 'No secret', and the 'Secret version' dropdown is set to 'Latest version'. A note at the bottom right says, 'Overall, how satisfied or dissatisfied are you with Azure Data Factory?'.

Click “New” for the AKV linked service. Name the linked service “Azurekeyvault1”. Choose “From Azure subscription”. Select free trial, select the azure key vault I have created earlier while I was setting up the infrastructure. Choose “system-assigned-managed-identity” in the Authentication Method section.

This screenshot is similar to the previous one but shows an error. The 'Secret name' dropdown is highlighted in red with the error message 'Loading failed'. Below it, a red box indicates 'Failed More'. The rest of the configuration is identical to the first screenshot.

Here I am not able to see any secret name even though I created a secret in the azure key vault earlier.

To resolve this issue,
Go to the key vaults, select the “kv customer demo” key vault I have created earlier.

The screenshot shows the Microsoft Azure Key vaults interface. On the left, there's a sidebar with 'Key vaults' and a search bar. The main area has a title 'kvcustomerdemo | Access control (IAM)'. Below the title are tabs: 'Add role assignment', 'Check access', 'Role assignments', 'Roles', 'Deny assignments', and 'Classic administrators'. The 'Check access' tab is currently selected. It displays sections for 'My access' (with a 'View my access' button), 'Check access' (with a 'Check access' button), 'Grant access to this resource' (with a 'Learn more' link), and 'View access to this resource' (with a 'View access to this resource' button). A sidebar on the right lists various vault management options like Overview, Activity log, and Objects.

Click the “Add role assignment” and select “Azure Key Vault Secret User”

The screenshot shows the 'Add role assignment' page. At the top, there are tabs for 'Role', 'Members', 'Conditions', and 'Review + assign'. The 'Role' tab is selected. Below the tabs, it says 'A role definition is a collection of permissions. You can use the built-in roles or you can create your own custom roles. [Learn more](#)'. Under 'Job function roles', there are two categories: 'Privileged administrator roles' and 'Grant access to Azure resources based on job function, such as the ability to create virtual machines.' A search bar at the top right filters results by 'Type: All' and 'Category: All'. The table below lists roles with columns for Name, Description, Type, Category, and Details.

Name ↑↓	Description ↑↓	Type ↑↓	Category ↑↓	Details
Key Vault Administrator	Perform all data plane operations on a key vault and all objects in it, including certificates, k...	BuiltInRole	Security	View
Key Vault Data Access Administrator	Manage access to Azure Key Vault by adding or removing role assignments for the Key Vault...	BuiltInRole	None	View
Key Vault Reader	Read metadata of key vaults and its certificates, keys, and secrets. Cannot read sensitive val...	BuiltInRole	Security	View
Key Vault Secrets Officer	Perform any action on the secrets of a key vault, except manage permissions. Only works fo...	BuiltInRole	Security	View
Key Vault Secrets User	Read secret contents. Only works for key vaults that use the 'Azure role-based access contr...	BuiltInRole	Security	View

Select the “Managed Identity” section

Microsoft Azure Search resources, services, and docs (G+)

Home > Key vaults > kvcustomerdemo | Access control (IAM) >

Add role assignment

Selected role: Key Vault Secrets User

Assign access to: User, group, or service principal Managed identity

Members: + Select members

Description: Optional

Select managed identities

Subscription *: Free Trial

Managed identity: Select

Search by resource type

User-assigned managed identity (1)

System-assigned managed identity

All system-assigned managed identities (3)

Access Connector for Azure Databricks (1)

Data factory (V2) (1)

Synapse workspace (1)

Learn more about RBAC

Now, choose “Data Factory”

Microsoft Azure Search resources, services, and docs (G+)

Home > Key vaults > kvcustomerdemo | Access control (IAM) >

Add role assignment

Selected role: Key Vault Secrets User

Assign access to: User, group, or service principal Managed identity

Members: + Select members

Description: Optional

Select managed identities

Subscription *: Free Trial

Managed identity: Data factory (V2) (1)

Select:

Search by name

customer-demographics-df /subscriptions/c18ffbb2-e46e-476b-aa92-aad49b43bbb3/resourceGroups/dev-rg/...

Selected members:

No members selected. Search for and add one or more members you want to assign to the role for this resource.

Learn more about RBAC

Selected role Key Vault Secrets User

Assign access to User, group, or service principal Managed identity

Members + Select members

Name	Object ID	Type
customer-demographics-df	3b17fdf3-ba55-4174-b7c2-bbd6a6ebd8...	Data factory (V2)

Description Optional

Once I click “Review and assign”, I can see the following page in which the data factory will be a key vault user

Key vaults Default Directory

kvcustomerdemo | Access control (IAM) Key vault

Access control (IAM)

Name	Type	Role	Scope
customer-demographics-df /subscriptions/... Data Factory	Key Vault Secrets User	Key Vault Secrets User	This resource

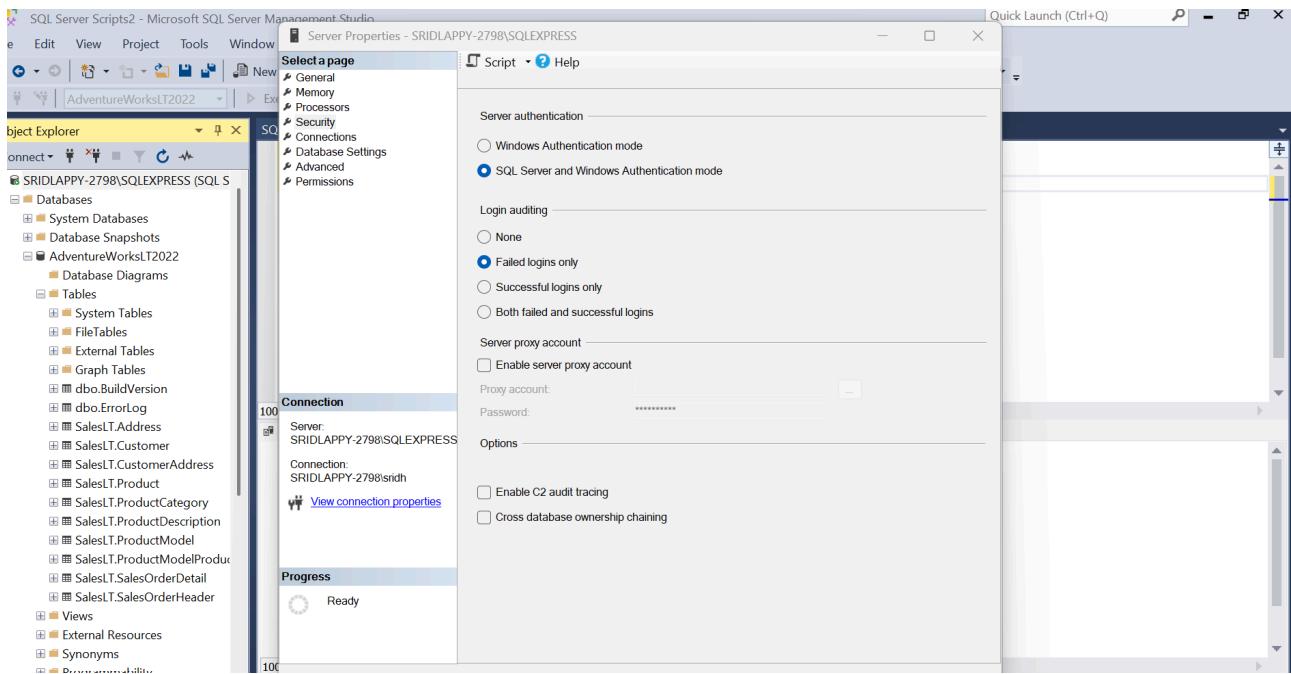
Now, if I go back to the azure data factory, I can see the below secret details that I have created earlier. I can see password2 in the drop down

The screenshot shows the Microsoft Azure Data Factory pipeline editor. On the left, the 'Factory Resources' sidebar lists 'Pipelines' (1), specifically 'copy_data'. The main workspace displays a 'Copy data' activity within a pipeline named 'copy_data'. A red circle highlights the 'Source dataset' field. To the right, a 'New linked service' configuration pane is open, titled 'SQL server'. It includes fields for 'User name' (set to 'sreedhar'), 'Password' (set to 'password2'), and 'AKV linked service' (set to 'AzureKeyVault1'). The 'Source' tab is selected. At the bottom right of the configuration pane, there is a survey question: 'Overall, how satisfied or dissatisfied are you with Azure Data Factory?'.

When I create the linked service error, the following error pops-up

This screenshot is similar to the one above, showing the pipeline editor with a 'copy_data' pipeline and a 'Copy data' activity. The 'Source dataset' field is highlighted with a red circle. The 'Set properties' pane on the right is open, showing a 'Failed' status with the message 'More' and an 'Enter manually' checkbox. The survey question at the bottom right is also present.

To resolve this issue, go to SQL server management studio, go to the “Properties” section of the database, click on “Security”, select “SQL Server and windows Authentication”



Also, edit the encryption settings in linked service to “Optional” and test the connection in the linked service section. Now I can see the connection is successful

But, if I check the image below, I am not able to find any tables. The catch here is, to grant the user (which I have created in SSMS earlier) the permissions to access the tables.

To grant permissions to the user, go to SSMS and execute the following as shown in the image below

```

--create login shree with password = 'Shree2728'
--create user shree for login shree

use AdventureWorksLT2022
GRANT SELECT ON SalesLT.Address to shree
  
```

Now, if I refresh the “Set Properties” tab in azure data factory, I can see that the table is available to access

The screenshot shows the Microsoft Azure Data Factory interface. On the left, there's a navigation pane with icons for Home, Pipelines, Datasets, Data flows, and Power Query. The main area shows a pipeline named 'copy_data' with one activity: 'Copy data'. A context menu is open over this activity, and a red box highlights the 'Set properties' option. The 'Set properties' dialog is displayed, titled 'Set properties'. It contains the following fields:

- Name: SqlServerTable1
- Linked service: SqlServerPrem
- Connect via integration runtime: selfhostedIntegrationRuntime1
- Table name: SalesLT.Address
- Import schema: None

At the bottom of the dialog are 'OK', 'Back', and 'Cancel' buttons.

Now, I can see that the source has been provisioned

The screenshot shows the Microsoft Azure Data Factory interface. On the left, there's a navigation pane with icons for Home, Pipelines, Datasets, Data flows, and Power Query. The main area shows a pipeline named 'copy_data' with one activity: 'Copy data'. The 'Properties' tab is selected in the ribbon. The 'Source' tab is currently active, showing the following settings:

- Source dataset: SqlServerTable1
- Use query: Table (selected)
- Query timeout (minutes): 120
- Isolation level: Select...
- Partition option: None (selected)

A note at the bottom says: 'Please preview data to validate the partition settings.'

The 'Properties' panel on the right shows:

- General tab: Name is 'copy_data'
- Description: (empty)
- Annotations: (empty)

To provision the sink, go to the sink and select "Sink", and select "New" and select "Azure Datalake storage Gen2"

The screenshot shows the Microsoft Azure Data Factory interface. On the left, there's a navigation pane with icons for Home, Copy Data, Data Flow, Synapse, Azure Data Explorer, Azure Function, Batch Service, Databricks, Data Lake Analytics, General, and HDInsight. The main area displays a pipeline named 'copy_data'. Under the 'Activities' section, 'Copy data' is selected. The 'Sink' tab is active, showing a 'Sink dataset *' field with a 'Select...' button. To the right, a 'New dataset' panel is open, titled 'In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store.' It includes a 'Select a data store' dropdown set to 'Azure Data' and a grid of data store options under 'All': Azure Data Explorer (Kusto), Azure Data Lake Storage Gen2, Azure Database for MySQL, Azure Databricks, and Azure Synapse Analytics.

Now I select 'Parquet' which is a columnar storage format

This screenshot continues from the previous one, showing the 'Select format' step. The 'Parquet' icon is highlighted in the bottom-left corner of the grid. The grid contains icons for various formats: Avro, Binary, DelimitedText, Iceberg, JSON, ORC, and Parquet. At the bottom of the screen, a taskbar shows icons for Continues, Back, Cancel, and a timestamp of 12:17 PM.

Now, I have to create one more linked service for azure data storage to link to the data factory. Here I choose “Autoresolve Integration Runtime” as the integration runtime because I am choosing another azure service to link to the data factory. Also, select the storage account name which I have created earlier while provisioning resources on azure.

The screenshot shows the Microsoft Azure Data Factory pipeline creation interface. A red box highlights the 'Sink dataset' dropdown in the 'copy_data' activity configuration. The 'Sink' tab is selected. To the right, a 'New linked service' configuration pane is open, showing settings for 'Azure Data Lake Storage Gen2'. It includes fields for 'Connect via integration runtime' (set to 'AutoResolveIntegrationRuntime'), 'Authentication type' (set to 'Account key'), 'Azure subscription' (set to 'Free Trial (c18ffbb2-e46e-476b-aa92-aad49b43bbb3)'), and 'Storage account name' (set to 'devstoragecustomer'). A 'Test connection' button is shown as successful. At the bottom right of the configuration pane are 'Create' and 'Cancel' buttons.

Now, I can select the destination folder as “bronze”.

The screenshot shows the Microsoft Azure Data Factory pipeline creation interface. A red box highlights the 'Sink dataset' dropdown in the 'copy_data' activity configuration. The 'Sink' tab is selected. To the right, a 'Browse' dialog is open, titled 'Select a file or folder'. It shows a 'Root folder' tree with four items: 'bronze', 'gold', 'silver', and 'synapse-fs'. Below the tree, it says 'Showing 1 - 4 of 4 items'.

The next step is to debug the pipeline I have created. I can do this by clicking the “debug” button. I can see that the debug was successful.

The screenshot shows the Microsoft Azure Data Factory interface. A pipeline named "copy_data" is currently running, indicated by the "In progress" status. The pipeline consists of a single "Copy data" activity. The "Output" tab is selected, showing details of the run. The pipeline run ID is 783992c1-8a33-43e8-8acc-0e0e679e281a. The activity name is "Copy data", status is "Succeeded", type is "Copy data", and the run start time is 11/16/2024, 12:28:11 P.

The screenshot shows the "Details" page for the "copy_data" activity. The activity run ID is 3613304f-5255-439c-b0ad-8f5ebf56ebce. The data flow is from "SQL server" to "Azure Data Lake Storage Gen2". The status is "Succeeded". Key performance metrics are listed:

Source	Target
SQL server	Azure Data Lake Storage Gen2
Data read:	60.192 KB
Rows read:	450
Peak connections:	1
Data written:	35.923 KB
Files written:	1
Rows written:	450
Peak connections:	1

Copy duration: 00:00:08. Throughput: 15.048 KB/s. Start time: 11/16/2024, 12:28:25 PM.

I can also see that the storage files were populated in the bronze container of azure data lake storage.

The screenshot shows the Azure Storage Container blade for the 'bronze' container. The left sidebar has 'Overview' selected. The main area displays a table with one item:

Name	Modified	Access tier	Archive status	Blob type	Size
SalesLT.Address.parquet	11/16/2024, 12:28:34...	Hot (Inferred)		Block blob	35.0

Now, I can create a new data pipeline to iterate through every table in the database and it brings all the tables into the bronze layer. To achieve this, I can create a new pipeline to copy all the tables (name it as “copy_all_tables”).

The first step is to get the list of all the tables automatically so that if more tables are populated in the database later, this pipeline needs to bring those newly populated tables too. For this, I have to use the “Lookup” activity in the “General” section of the “Activities” tab in Azure data factory. Select the source as “SQLServerTable1” and open the linked service.

The screenshot shows the Azure Data Factory pipeline editor. On the left, the 'Factory Resources' sidebar lists Pipelines, Datasets, and other components. The 'copy_all_tables' pipeline is selected. The main workspace shows the pipeline structure:

- A 'copy_all_tables' activity is connected to a 'SqlServerTable1' dataset.
- The 'SqlServerTable1' dataset is configured with a 'Linked service' set to 'SqlServerPrem' and an 'Integration runtime' set to 'selfhostedintegrationRuntime1'.
- The 'Properties' panel on the right shows the 'Name' field set to 'SqlServerTable1'.

Now go to the settings of lookup activity to write a SQL query to get the list of table names in the database. Use the following SQL query to get the list of tables.

```

SELECT
    s.name as SchemaName,
    t.name as TableName
FROM
    sys.tables t
INNER JOIN
    sys.schemas s
ON
    t.schema_id = s.schema_id
WHERE
    s.name = 'SalesLT'

```

The screenshot shows the Microsoft Azure Data Factory Pipeline Editor. On the left, the 'Factory Resources' sidebar lists 'Pipelines' (copy_data, copy_all_tables), 'Datasets' (Parquetsink, SqlServerTable1), and 'Data flows'. The main workspace displays a pipeline named 'pipeline1' containing a 'Lookup' activity named 'Lookup1'. The 'Settings' tab is active, showing the query: 'select s.name as SchemaName, t.name as TableName from sys.tables t'. The 'Properties' pane on the right shows the pipeline's general settings, including its name 'pipeline1'.

Once I debug the lookup activity, I can find the below screen

The screenshot shows the Microsoft Azure Data Factory Pipeline Editor after debugging. The 'copy_all_tables' pipeline is selected. The 'Output' tab is active, displaying the pipeline run ID '9723ec6d-8f23-4776-9e43-d95d669db5bf' and the status 'Succeeded'. Below this, a table provides details for the single activity run:

Activity name	Activity status	Activity type	Run st
Lookup1	Succeeded	Lookup	11/17/

But if I take a closer look at the input and output of lookup activity, I can find that lookup activity only brought “Address” table. The output JSON of lookup activity looks like below

```
{  
    "firstRow": {  
        "SchemaName": "SalesLT",  
        "TableName": "Address"  
    },  
    "effectiveIntegrationRuntime": "selfhostedintegrationRuntime1",  
    "billingReference": {  
        "activityType": "PipelineActivity",  
        "billableDuration": [  
            {  
                "meterType": "SelfhostedIR",  
                "duration": 0.016666666666666666,  
                "unit": "Hours"  
            }  
        ],  
        "totalBillableDuration": [  
            {  
                "meterType": "AzureIR",  
                "duration": 0.06666666666666667,  
                "unit": "Hours"  
            }  
        ]  
    },  
    "durationInQueue": {  
        "integrationRuntimeQueue": 6  
    }  
}
```

This happened because, earlier when I granted permissions for the user in SSMS, I only granted the select permissions for the user on “Address” table. In order to pull all the tables, the user needs to have select permissions on all the tables. The permissions on all the tables can be granted using the following SQL query

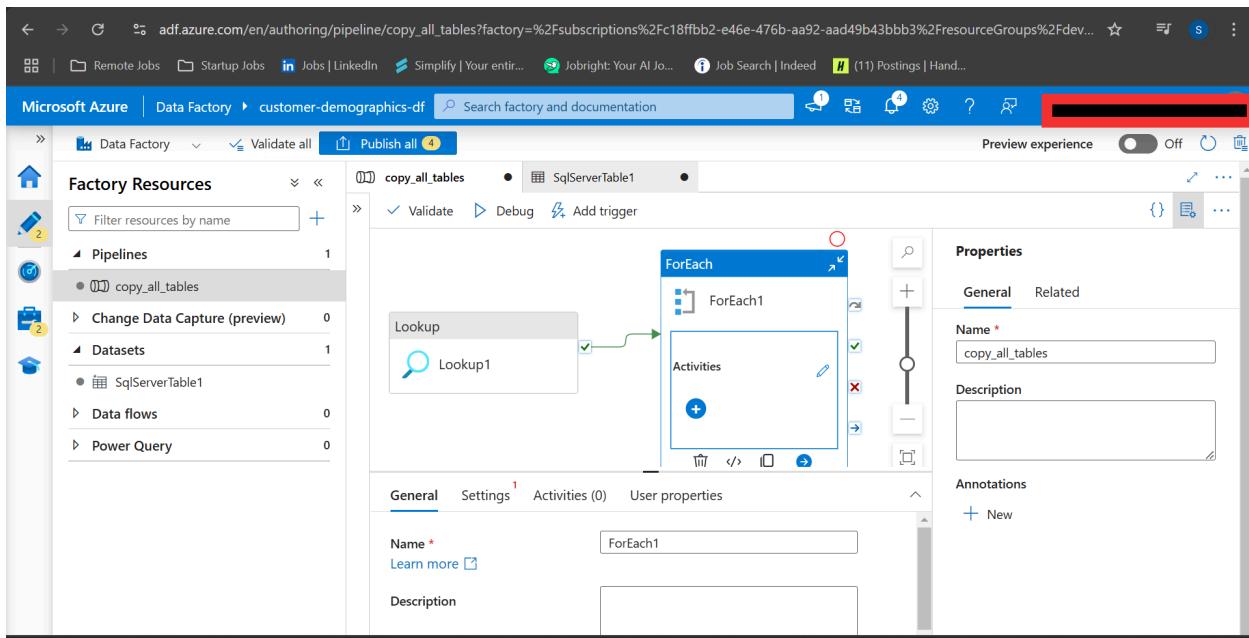
```
use AdventureWorksLT2022;  
GRANT SELECT ON SCHEMA::SalesLT TO shree
```

```

createlogin2.sql - SRIDLAPPY-2798\SQLEXPRESS.AdventureWorksLT2022 (SRIDLAPPY-2798\sridh (59)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query SQLQuery1.sql - SR..PY-2798\sridh (66)
AdventureWorksLT2022 Execute
Object Explorer Connect
SRIIDLAPPY-2798\SQLEXPRESS (SQL Server)
Databases
System Databases
Database Snapshots
AdventureWorksLT2022
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.BuildVersion
dbo.errorLog
SalesLT.Address
SalesLT.Customer
SalesLT.CustomerAddress
SalesLT.Product
SalesLT.ProductCategory
SalesLT.ProductDescription
SalesLT.ProductModel
SalesLT.ProductModelProduct
SalesLT.SalesOrderDetail
SalesLT.SalesOrderHeader
Views
External Resources
Synonyms
Programmability
100 %
Messages
Commands completed successfully.
Completion time: 2024-11-17T05:58:03.5742912-06:00
100 %
Query executed successfully.
SRIIDLAPPY-2798\SQLEXPRESS (...) SRIIDLAPPY-2798\sridh (59) AdventureWorksLT2022 00:00:00 0 rows

```

Now, the user has all the required permissions to access all the tables. Go to the azure data factory and attach a “ForEach” activity which happens on the success of lookup activity.



Now, I have to edit the activities in the “ForEach” activity. In the source section of the copy data activity, I have to select query and choose “Add Dynamic Content” option as I need to select the tables dynamically and pull them to the bronze layer.

The screenshot shows the Microsoft Azure Data Factory Pipeline expression builder. A pipeline named 'copy_all_tables' is selected. In the Source tab, a query is defined as follows:

```
@{concat('SELECT * FROM', item().SchemaName, '.', item().TableName)}
```

In the pipeline expression builder, use the following expression

```
@{concat('SELECT * FROM', item().SchemaName, '.', item().TableName)}
```

Add the sink dataset details.

The screenshot shows the Microsoft Azure Data Factory Pipeline configuration. The pipeline 'copy_all_tables' is selected. The Sink tab is active, showing 'Parquet1' selected as the sink dataset. The Properties panel on the right shows the dataset name as 'copy_all_tables'.

In the sink parameters, add the following parameters

The screenshot shows the Microsoft Azure Data Factory interface. On the left, the 'Factory Resources' sidebar is open, showing Pipelines, Datasets, and other resources. In the center, a pipeline named 'copy_all_tables' is selected. A dataset named 'Parquet1' is connected to it. The 'Properties' panel on the right shows the dataset's name is 'Parquet1' and its schema is 'Parquet'. The 'Parameters' tab in the center panel shows two parameters: 'schemaname' and 'tablename', both of type String.

Now, I add dynamic content value for each parameter I have created. For the schema name and tablename, add the dynamic content as follows:

The screenshot shows the Microsoft Azure Data Factory pipeline builder. The pipeline 'copy_all_tables' is selected. In the 'Sink' tab, the sink dataset is set to 'Parquet1'. Under 'Dataset properties', there are two entries: 'schemaname' with the value 'item().SchemaName' and 'tablename' with the value 'item().TableName'. The 'Properties' panel on the right shows the pipeline's name is 'copy_all_tables'.

In the connection settings of the sink, in the filepath, add the following expression in pipeline builder

The screenshot shows the Azure Data Factory pipeline editor. On the left, there's a pipeline named 'copy_all_tables' with three stages: 'SqlServerTable1', 'Parquet1', and another unnamed stage. Stage 'Parquet1' has a 'Parquet' icon. The 'Parameters' tab is selected in the 'Connection' section. In the 'File path' field, the expression `@{concat(dataset().schemaname,'/',dataset().tablename)}` is entered. To the right, the 'Pipeline expression builder' pane displays this expression with parameters 'schemaname' and 'tablename' listed below it.

In the File Name, add the below expression in pipeline builder

This screenshot is similar to the previous one, but the 'File path' field now contains the expression `@{concat(dataset().tablename,'.parquet')}`. The rest of the interface, including the pipeline stages and the expression builder pane, remains the same.

In the pipeline view, for the “ForEach” Activity, in the “Settings” option, add the following pipeline expression builder

Once the changes are done, publish the pipeline so that all the resources are deployed. Now I can add the trigger. Generally, triggers are used to trigger the pipeline on a daily basis or (to trigger when there is an update in the database). I can trigger this pipeline by using “Trigger Now” option.

I can see below that the pipeline run was successful

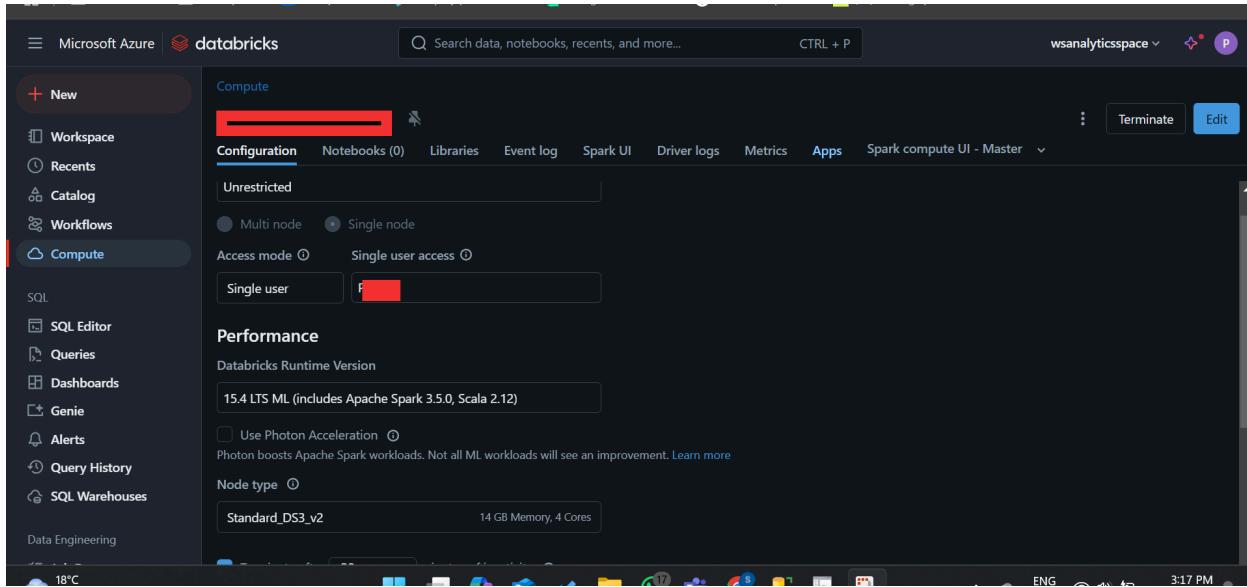
The screenshot shows the Microsoft Azure Data Factory interface. On the left, a sidebar menu includes options like Dashboards, Runs, Pipeline runs (which is selected), Trigger runs, Change Data Capture, Runtimes & sessions, Integration runtimes, Data flow debug, Notifications, and Alerts & metrics. The main area displays a table titled 'All pipeline runs > copy_all_tables - Activity runs'. The table has columns for Activity name, Activity status, Activity type, Run start, Duration, Integration runtime, and Us. The data shows 12 items, all of which have succeeded. The integration runtime for most activities is 'selfhostedintegrationR'. The table includes sorting and filtering options at the top.

I can also see that bronze layer now consists of all the tables and all the data in bronze layer is stored in parquet format.

The screenshot shows the Microsoft Azure Storage Explorer interface. The left sidebar shows a navigation tree with 'Home', 'bronze' (selected), 'Container', 'Overview', 'Diagnose and solve problems', 'Access Control (IAM)', and 'Settings'. The main area shows a table of files in the 'bronze' container. The columns are Name, Modified, Access tier, Archive status, Blob type, and Size. The table lists several files, all of which were modified on 11/17/2024, including 'Address', 'Customer', 'CustomerAddress', 'Product', 'ProductCategory', 'ProductDescription', 'ProductModel', 'ProductModelProductDescription', 'SalesOrderDetail', and 'SalesOrderHeader'. All files are of type 'Parquet'.

As I have the data in bronze layer, now I can apply the transformations on this data using azure databricks (which I have provisioned earlier) to create silver and gold layers. I need to create a compute which will be used by the databricks notebooks to apply transformations.

Create a compute with the below configuration.



As the cluster creation completes, I found another resource group in my azure account, with all the resources provisioned that can be used by databricks workspace

A screenshot of the Microsoft Azure portal showing the 'Resource groups' page. The URL is 'portal.azure.com/#browse/resourcegroups'. The page lists four resource groups: 'databricks-rg-wsanalyticsspace-mnekoc4owaun2', 'dev-rg', 'NetworkWatcherRG', and 'synapseworkspace-managedrg-55b5d0e9-ea23-4974-93df-ba2d0f03a7c4'. Each entry includes a checkbox, the name, subscription information ('Free Trial'), and location ('UK South', 'South Central US').

Name	Subscription	Location
databricks-rg-wsanalyticsspace-mnekoc4owaun2	Free Trial	UK South
dev-rg	Free Trial	South Central US
NetworkWatcherRG	Free Trial	South Central US
synapseworkspace-managedrg-55b5d0e9-ea23-4974-93df-ba2d0f03a7c4	Free Trial	UK South

The resource group with the name "[databricks-rg-wsanalyticsspace-mnekoc4owaun2](#)" is the resource group that will be used by the databricks.
It has the below resources provisioned

Once I provisioned the compute, I created a notebook with name “mountstorage”. Before I start using the data in Azure Data Lake, I have to mount it to a cluster so that I can have access point to the storage.

Then, I mount the bronze, silver and gold layers to databricks using the following code

```
#Configuration settings for authenticating and accessing ADLS Gen2 storage
adls_configs = {
    "fs.azure.account.auth.type": "CustomAccessToken",
    "fs.azure.account.custom.token.provider.class": spark.conf.get(
        "spark.databricks.passthrough.adls.gen2.tokenProviderClassName"
    )
}

# Mount the Bronze container from ADLS to Databricks file system
# This mount point allows access to raw data from the Bronze layer
dbutils.fs.mount(
    source="abfss://bronze@devstoragecustomer.dfs.core.windows.net/",
    mount_point="/mnt/bronze",
    extra_configs=adls_configs
)

# Mount the Silver container from ADLS to Databricks file system
# This mount point allows access to cleansed and enriched data
```

```

dbutils.fs.mount(
    source="abfss://silver@devstoragecustomer.dfs.core.windows.net/",
    mount_point="/mnt/silver",
    extra_configs=adls_configs
)

# Mount the Gold container from ADLS to Databricks file system
# This mount point allows access to business-level aggregated data
dbutils.fs.mount(
    source="abfss://gold@devstoragecustomer.dfs.core.windows.net/",
    mount_point="/mnt/gold",
    extra_configs=adls_configs
)
dbutils.fs.ls("/mnt/bronze")

```

Code Explanation:

1. Configuration Setup:

- The code uses a dictionary to store configuration settings required for authentication and accessing ADLS, such as authentication type and token provider class.

2. Mount ADLS Containers:

- The `dbutils.fs.mount` function is used to mount specific containers (e.g., bronze, silver, gold) from an ADLS Gen2 storage account to Databricks' file system. This allows seamless access to data in those containers via Databricks.

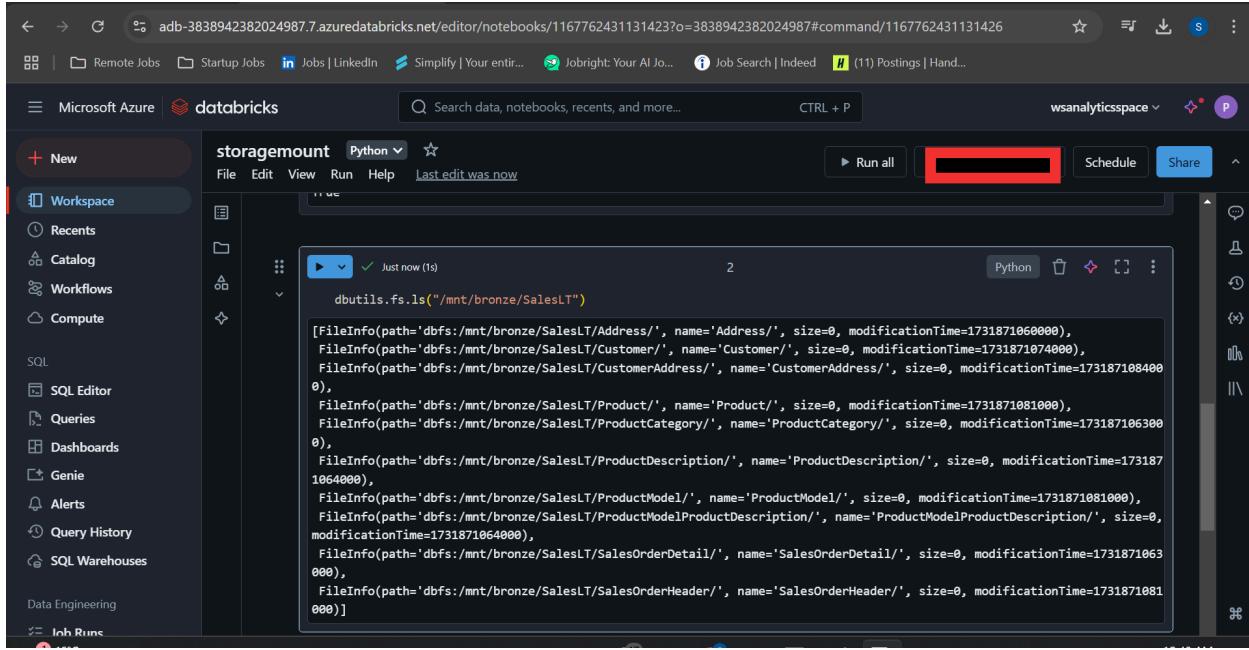
3. Error Handling:

- The code includes logic to check for existing mounts to avoid errors when attempting to remount directories.

4. Purpose:

- By mounting the storage containers, Databricks users can efficiently access and process data from ADLS without dealing directly with storage endpoints in each operation.

Once I run the notebook, I can see the following contents in the bronze layer



The next stage is to do transformations on the data which is in the bronze layer and promote the data to the silver layer.

In the transformations, I started with a single table column transformation and then I proceeded with multiple table transformations.

I used the below code

```

dbutils.fs.ls('mnt/bronze/SalesLT/')

dbutils.fs.ls('mnt/silver/') # check if the silver layer has any contents

# Load the Address data from the Bronze layer in Parquet format
address_df =
spark.read.format('parquet').load('/mnt/bronze/SalesLT/Address/Address.parquet')

# Display the contents of the Address DataFrame for a quick preview
display(address_df)

from pyspark.sql.functions import from_utc_timestamp, date_format
from pyspark.sql.types import TimestampType

# Convert the ModifiedDate column from UTC to a specific format
# (yyyy-MM-dd)
# 1. Cast the ModifiedDate column to TimestampType
# 2. Convert it to UTC timestamp using from_utc_timestamp

```

```
# 3. Format the timestamp to "yyyy-MM-dd" using date_format
address_df = address_df.withColumn(
    "ModifiedDate",
    date_format(
        from_utc_timestamp(address_df['ModifiedDate']).cast(TimestampType()), 
        "UTC"),
        "yyyy-MM-dd"
    )
)

# Display the contents of the Address DataFrame
display(address_df)

# Initialize an empty list to store extracted table identifiers
table_list = []

# List all files and directories in the specified Bronze layer path
for item in dbutils.fs.ls('mnt/bronze/SalesLT/'):
    # Extract the table name by splitting the directory/file name and
    # append to the list
    table_list.append(item.name.split('/')[0])

# Display the list of table identifiers
table_list

from pyspark.sql.functions import from_utc_timestamp, date_format
from pyspark.sql.types import TimestampType

# Iterate through the list of tables
for i in table_list:
    # Define the path to the source parquet file in the bronze layer
    path = '/mnt/bronze/SalesLT/' + i + '/' + i + '.parquet'

    # Load the parquet file into a DataFrame
    df = spark.read.format('parquet').load(path)

    # Get the column names of the DataFrame
    column = df.columns
```

```

# Check each column for "Date" or "date" and reformat it
for col in column:
    if "Date" in col or "date" in col:
        # Convert column to UTC timestamp and reformat to "yyyy-MM-dd"
        df = df.withColumn(col,
date_format(from_utc_timestamp(df[col].cast(TimestampType()), "UTC"),
"yyyy-MM-dd"))

# Define the output path for the silver layer in Delta format
output_path = '/mnt/silver/SalesLT/' + i + '/'

# Write the DataFrame to the silver layer in Delta format with
overwrite mode
df.write.format('delta').mode('overwrite').save(output_path)

# display the results
display(df)

```

Code Explanation:

1. Check Bronze and Silver Layers:

- The `dbutils.fs.ls` command lists the contents of the directories in the Bronze and Silver layers to verify their structure and contents.

2. Load and Preview Address Data:

- The `address_df` DataFrame loads the "Address" table from the Bronze layer (in Parquet format).
- The `display(address_df)` command previews the contents of the loaded table.

3. Date Column Transformation for `ModifiedDate`:

- The `ModifiedDate` column in `address_df` is:
 - Cast to `TimestampType`.
 - Converted to a UTC timestamp using `from_utc_timestamp`.
 - Reformatted to "yyyy-MM-dd" using `date_format`.

4. Extract Table Names from the Bronze Layer:

- The code lists all files/directories in the Bronze layer.
- Extracts table names by splitting the directory names and stores them in the `table_list` list.

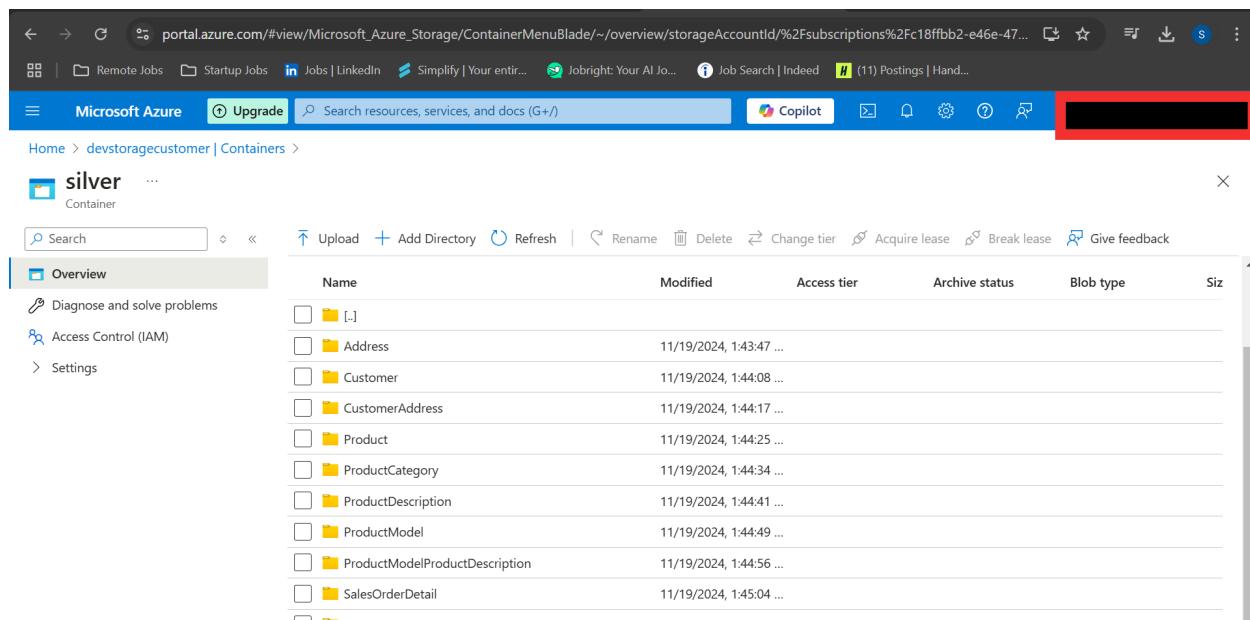
5. Process All Tables from the Bronze Layer:

- Iterates through the `table_list`.

- Loads each table as a DataFrame from the Bronze layer.
 - Checks for columns containing "Date" or "date" in their names.
 - Reformats these columns to "yyyy-MM-dd" in UTC.
- 6. Write Transformed Data to Silver Layer:**
- Writes the transformed DataFrame in Delta format to the Silver layer at the corresponding path.
 - Uses `mode('overwrite')` to replace existing files.
- 7. Preview Final DataFrame:**
- The final DataFrame (`df`) from the last iteration is displayed using the `display(df)` command.

This code processes and migrates data from the Bronze layer (raw Parquet files) to the Silver layer (cleaned and standardized Delta files) while ensuring date columns are properly formatted.

I can see the contents in the silver layer after the above code is executed in databricks



The screenshot shows the Microsoft Azure Storage Container Overview page for a container named 'silver'. The left sidebar includes links for 'Overview', 'Diagnose and solve problems', 'Access Control (IAM)', and 'Settings'. The main area displays a table of blob items with columns: Name, Modified, Access tier, Archive status, Blob type, and Size. The table lists several items, including 'Address', 'Customer', 'CustomerAddress', 'Product', 'ProductCategory', 'ProductDescription', 'ProductModel', 'ProductModelProductDescription', and 'SalesOrderDetail', all modified on 11/19/2024.

Name	Modified	Access tier	Archive status	Blob type	Size
[...]	11/19/2024, 1:43:47 ...				
Address	11/19/2024, 1:44:08 ...				
Customer	11/19/2024, 1:44:17 ...				
CustomerAddress	11/19/2024, 1:44:25 ...				
Product	11/19/2024, 1:44:34 ...				
ProductCategory	11/19/2024, 1:44:41 ...				
ProductDescription	11/19/2024, 1:44:49 ...				
ProductModel	11/19/2024, 1:44:56 ...				
ProductModelProductDescription	11/19/2024, 1:45:04 ...				
SalesOrderDetail	11/19/2024, 1:45:12 ...				

In the silver layer, the data is stored in delta format as shown below

The screenshot shows the Azure Storage Container blade for the 'silver' container. The container has three items:

- A folder named '[...]'
- A file named '_delta_log' (modified 11/19/2024, 1:44:34 ...)
- A file named 'part-00000-19a91e66-1082-4966-b7d9-595952e4067...' (modified 11/19/2024, 1:44:35 ...)

The next step is to create a gold layer from the silver layer.

I used the below code to generate the gold layer.

```
# List all files and directories in the specified path in the Databricks
File System (DBFS)
# 'mnt/silver/SalesLT/' refers to a mounted directory in DBFS, often
linked to external storage
dbutils.fs.ls('mnt/silver/SalesLT/')

dbutils.fs.ls('mnt/gold/') # list the files and directories in the gold
layer

# Reading a Delta table into a Spark DataFrame
# '/mnt/silver/SalesLT/Address/' specifies the path to the Delta table in
the Silver layer
df = spark.read.format('delta').load('/mnt/silver/SalesLT/Address/')

display(df)

def convert_column_names_to_snake_case(input_df):
    """
    Convert column names in a PySpark DataFrame from PascalCase or
    camelCase to snake_case.
    """

    Args:
```

Args:

```
    input_df (DataFrame): The input PySpark DataFrame with columns to
be converted.

    Returns:
        DataFrame: A new DataFrame with column names converted to
snake_case.

    """
    # Fetch the list of original column names
    original_column_names = input_df.columns

    # Dictionary to store the mapping of old to new column names
    column_name_mapping = {}

    # Iterate over the column names
    for original_name in original_column_names:
        # Generate the snake_case version of the column name
        snake_case_name = "".join([
            "_" + character.lower() if (
                character.isupper()                                # Check if the
                character is uppercase
            and index > 0                                         # Ensure it's not the
            first character
            and not original_name[index - 1].isupper()          # Ensure the
            previous character isn't uppercase
            ) else character.lower()   # Convert the character to lowercase
            otherwise
            for index, character in enumerate(original_name)
        ]).lstrip("_")  # Remove leading underscore, if any

        # Ensure the new column name is unique
        if snake_case_name in column_name_mapping.values():
            raise ValueError(f"Duplicate column name found after renaming:
'{snake_case_name}'")

        # Map the original column name to the new snake_case column name
        column_name_mapping[original_name] = snake_case_name

    # Apply the column name mapping to the DataFrame
    for old_name, new_name in column_name_mapping.items():
        input_df = input_df.withColumnRenamed(old_name, new_name)
```

```
return input_df

df = convert_column_names_to_snake_case(df)

display(df)

#transformations on multiple tables
# Initialize an empty list to store table names
table_names_list = []

# Iterate over the file system objects in the specified directory
for file_info in dbutils.fs.ls('mnt/silver/SalesLT'):
    # Append each file or folder name to the list
    table_names_list.append(file_info)

# Display the collected table names
table_names_list

# Initialize an empty list to store the table names
table_names = []

# Iterate over the file system objects in the specified directory
for file_object in dbutils.fs.ls('mnt/silver/SalesLT'):
    # Extract the name of the file or folder, removing any trailing
    slashes, and add it to the list
    table_names.append(file_object.name.split('/')[0])

# Display the list of table names
table_names

# Iterate over each table name in the list
for table_name in table_names:
    # Construct the input file path for the current table
    input_path = '/mnt/silver/SalesLT/' + table_name
    print(f"Reading data from: {input_path}")

    # Read the Delta table into a DataFrame
```

```

input_df = spark.read.format('delta').load(input_path)

# Convert column names to snake_case
df = convert_column_names_to_snake_case(input_df)

# Construct the output file path for the transformed table
output_path = '/mnt/gold/SalesLT/' + table_name + '/'
print(f"Writing transformed data to: {output_path}")

# Write the transformed DataFrame back to Delta format with overwrite mode
df.write.format('delta').mode('overwrite').save(output_path)

display(df)

```

List Files:

- Shows the files and folders in the Silver and Gold layers (`mnt/silver/SalesLT/` and `mnt/gold/`).

Read a Delta Table:

- Loads a specific table (`Address`) from the Silver layer into a Spark DataFrame.

Standardize Column Names:

- A function (`convert_column_names_to_snake_case`) is used to rename DataFrame columns to `snake_case` for consistency.

Get All Table Names:

- Collects all table names in the Silver layer directory (`mnt/silver/SalesLT/`).

Transform and Save Data:

- For each table in the Silver layer:
 - Reads the data into a DataFrame.
 - Converts column names to `snake_case`.
 - Saves the transformed data as a Delta table in the Gold layer (`mnt/gold/SalesLT/`).

Display Transformed Data:

- Displays the final transformed DataFrame to verify the output.

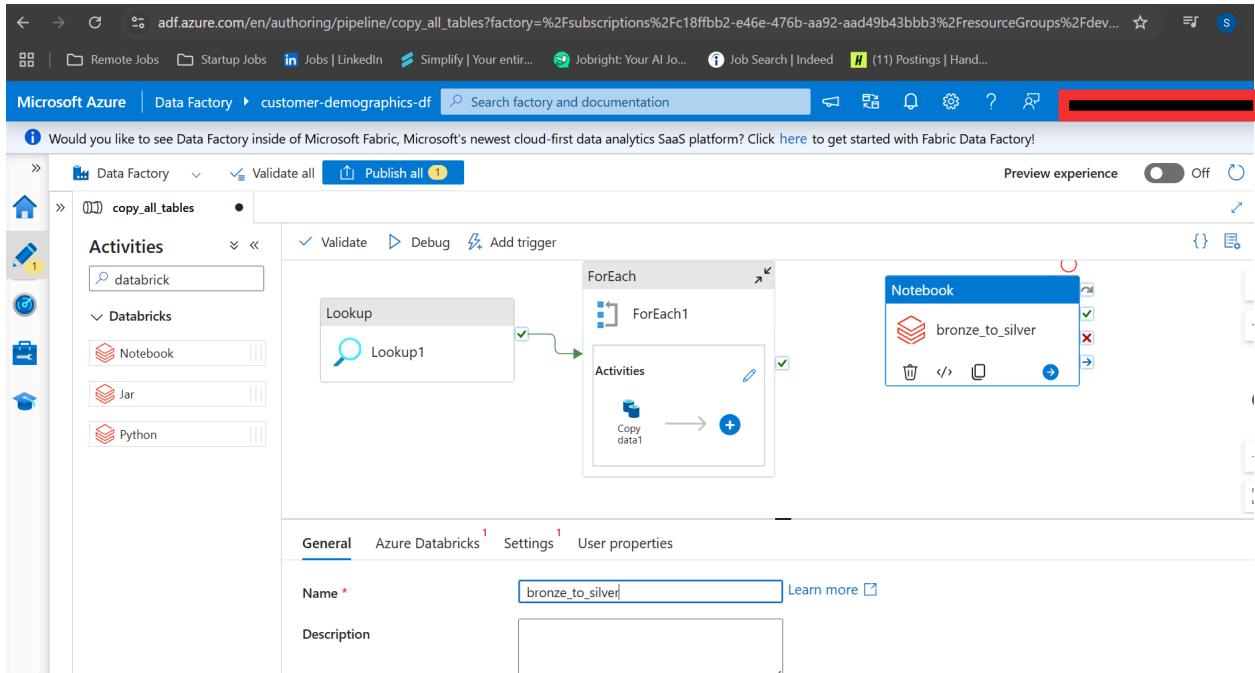
I can also find the files in the gold containers as below.

Name	Modified	Access tier	Archive status	Blob type	Size
[..]					
Address	11/20/2024, 10:53:40...				
Customer	11/20/2024, 10:54:00...				
CustomerAddress	11/20/2024, 10:54:11...				
Product	11/20/2024, 10:54:19...				
ProductCategory	11/20/2024, 10:54:28...				
ProductDescription	11/20/2024, 10:54:36...				
ProductModel	11/20/2024, 10:54:44...				
ProductModelProductDescription	11/20/2024, 10:54:51...				
SalesOrderDetail	11/20/2024, 10:54:59...				

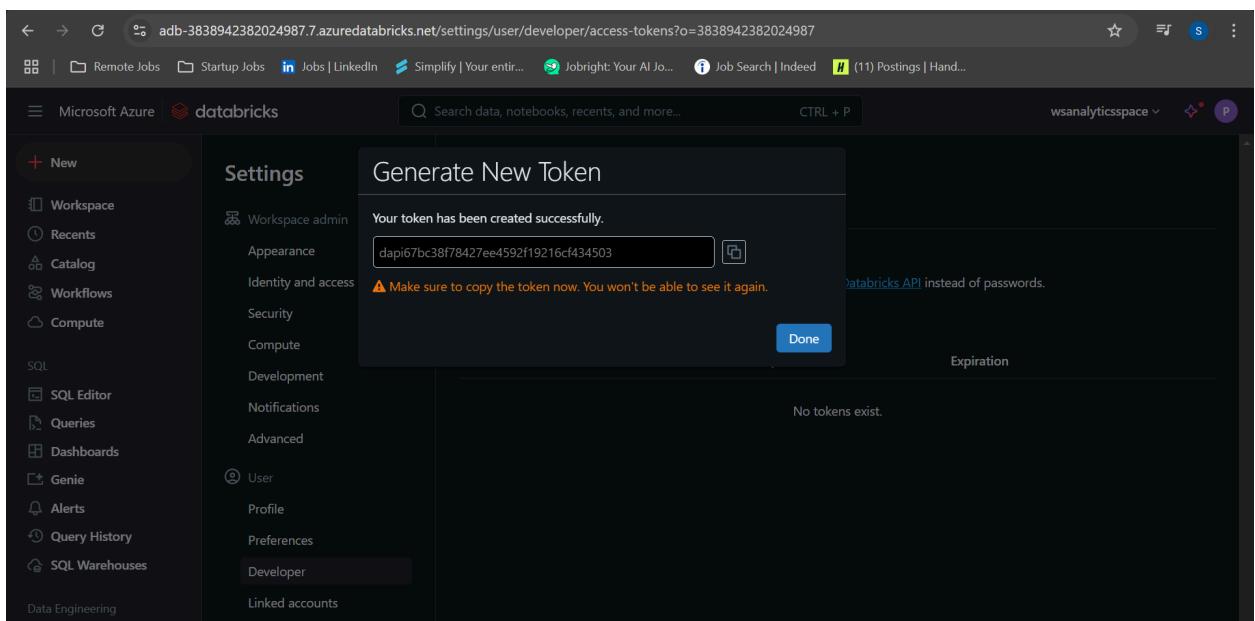
In the gold layer, the files are stored in delta format as shown below

Name	Modified	Access tier	Archive status	Blob type	Size
[..]					
_delta_log	11/20/2024, 10:53:40...				
part-00000-43e0907e-be53-4a71-b70d-bcb569fe7a16...	11/20/2024, 10:53:44...	Hot (Inferred)		Block blob	34.6

All the above transformations need to be performed as a part of the data pipeline I have created earlier in the data factory. So, the notebook needs to be a part of the pipeline. Attach the notebook to ForEach Activity in the Azure Data Factory.



Now, I created a Linked Service for Databricks. Used AutoResolveIntegrationRuntime as I connected to a service within Azure. For the cluster, I have chosen “Existing Interactive Cluster”. For the Authentication Type, I have chosen “Azure Key Vault”. Access tokens can be generated from Developer settings in databricks. These tokens are stored as secrets in the azure key vault. Azure Key vault can be leveraged for authentication purposes.



As the token can't be seen again, I created a secret in the Azure Key Vault as shown below

The screenshot shows the 'Create a secret' page in the Microsoft Azure portal. The 'Name' field is set to 'adb-key'. The 'Secret value' field contains a redacted password. The 'Enabled' switch is set to 'Yes'. At the bottom, there are 'Create' and 'Cancel' buttons.

Upload options: Manual
Name *: adb-key
Secret value *: *****
Content type (optional)
Set activation date:
Set expiration date:
Enabled: Yes
Tags: 0 tags
Create Cancel

Now, in the linked service authentication, I have selected the Azure Key vault which I have created earlier and also the secret I generated with the databricks access token stored in it. I have chosen the existing cluster in the cluster options as shown below

The screenshot shows the Microsoft Azure Data Factory pipeline editor. A 'copy_all_tables' pipeline is displayed. On the right, a 'New linked service' dialog for 'Azure Databricks' is open. It shows the 'Secret name' as 'adb-key', 'Secret version' as '1f5e9073e7f5425ca7fb3d360f9a41c2b (Current version)', and 'Choose from existing clusters' set to 'Priya R's Cluster'. The pipeline itself consists of a 'Lookup' activity followed by a 'ForEach' loop containing a 'Copy data1' activity.

New linked service
Azure Databricks Learn more
Azure Databricks
Secret name *: adb-key
Secret version *: 1f5e9073e7f5425ca7fb3d360f9a41c2b (Current version)
Choose from existing clusters *: Priya R's Cluster
Annotations
Parameters
Advanced

In the Settings, I have chosen the notebook path which points to the “bronze to silver” notebook in the databricks workspace as shown below

The screenshot shows the Microsoft Azure Data Factory interface. A pipeline named 'copy_all_tables' is selected. On the left, the 'Activities' pane lists 'databrick' under 'Databricks', with 'Notebook' selected. The main workspace displays a flow starting with a 'Lookup' activity (Lookup1) connected to a 'ForEach' loop. Inside the loop, there is an 'Activities' section containing a 'Copy data1' activity. The 'Settings' tab is active, showing the 'Notebook path' as '/Users/priyaravuri160'. To the right, a 'Browse' window shows a folder structure under 'Root folder > Users > priyaravuri1602@gmail.com', listing 'Sample Dashboards', 'bronze to silver', 'silver to gold', and 'storagemount'. A message at the top asks if the user wants to see Data Factory inside Microsoft Fabric.

Repeat the above steps in order to add one more notebook “silver to gold” to the data pipeline. After adding the “silver to gold” notebook to the pipeline, the ADF looks like below. Once all the pipeline settings are configured, I published the pipeline.

The screenshot shows the Microsoft Azure Data Factory interface after adding the 'silver_to_gold' notebook. The pipeline 'copy_all_tables' now includes an additional 'ForEach' loop iteration for the 'silver_to_gold' notebook. The 'Activities' pane on the left shows 'Move and transform' options like 'Copy data' and 'Data flow', along with 'Synapse', 'Azure Data Explorer', 'Azure Function', 'Batch Service', 'Databricks', 'Data Lake Analytics', and 'General' sections. The pipeline flow now includes a second 'ForEach' loop iteration for the 'silver_to_gold' notebook. The 'Settings' tab shows the 'Notebook path' as '/Users/priyaravuri1602@gmail.com/silver...'. The 'Preview experience' toggle is off.

I triggered the pipeline using an on-demand trigger and I can see that the pipeline activities are queued.

The screenshot shows the Microsoft Azure Data Factory pipeline editor. On the left, a sidebar lists navigation options: Dashboards, Runs (selected), Pipeline runs, Trigger runs, Change Data Capture (previous), Runtimes & sessions, Integration runtimes, Data flow debug, Notifications, and Alerts & metrics. The main area displays a pipeline diagram and its execution history.

Pipeline Diagram:

```
graph LR; L1[Lookup] --> F1[ForEach]; F1 --> A1[Activities]; A1 --> N1[Notebook: bronze_to_silver]; N1 --> N2[Notebook: silver_to_gold]
```

Execution History:

Activity	Status	Start Time	Duration	Runtimes
Copy data1	Succeeded	11/21/2024, 12:58:29 A	12s	selfhostedIntegrationR
Copy data1	Succeeded	11/21/2024, 12:58:29 A	38s	selfhostedIntegrationR
Copy data1	Succeeded	11/21/2024, 12:58:29 A	12s	selfhostedIntegrationR
Copy data1	Succeeded	11/21/2024, 12:58:29 A	39s	selfhostedIntegrationR
Copy data1	Succeeded	11/21/2024, 12:58:29 A	21s	selfhostedIntegrationR
Copy data1	Succeeded	11/21/2024, 12:58:29 A	26s	selfhostedIntegrationR
Copy data1	Succeeded	11/21/2024, 12:58:29 A	32s	selfhostedIntegrationR
Copy data1	Succeeded	11/21/2024, 12:58:29 A	12s	selfhostedIntegrationR
Copy data1	Succeeded	11/21/2024, 12:58:29 A	21s	selfhostedIntegrationR
bronze_to_silver	In progress	11/21/2024, 12:59:10 A	1m 2s	

Note that the time taken might be long if it takes longer time for the cluster to start

I can monitor the execution as shown below

The screenshot shows the Databricks interface for a job run. On the left, the sidebar includes options like Workspace, Recents, Catalog, Workflows, Compute, SQL, SQL Editor, Queries, Dashboards, Genie, Alerts, Query History, SQL Warehouses, Data Engineering, Job Runs (which is selected), and Data Ingestion. The main area displays a job run titled "ADF_customer-demographics-df_copy_all_tables_bronze_to_silver_3950c867-df0b-4ae6-a1c2-d7bc197dfa1 run". It shows an "Output" section with a code snippet and a "Task run details" panel on the right. The task run details include:

Job ID	991030004087056
Task run ID	560194410876761
Run as	Priya R
Started	11/21/2024, 01:03:48 AM
Ended	-
Duration	51s
Queue duration	-
Status	Running - Cancel
Lineage	No lineage information for this job. Learn more

The same progress can be tracked using the job runs in databricks. Also, the timeline of the activities can be monitored from Gantt Charts in Azure Data Factory

The screenshot shows the Azure Data Factory Gantt chart for a pipeline run. The left sidebar lists options: Dashboards, Runs (selected), Pipeline runs, Trigger runs, Change Data Capture (previous), Runtimes & sessions, Integration runtimes, Data flow debug, Notifications, and Alerts & metrics. The main area shows a Gantt chart for a pipeline run ID. The chart displays tasks: "Copy data1" (Nov 21 12:59:00 to Nov 21 01:01:00), "bronze_to_silver" (Nov 21 01:03:00 to Nov 21 01:05:00), and "silver_to_gold" (Nov 21 01:08:00 to Nov 21 01:08:00). The chart includes buttons for Rerun, Cancel, Refresh, Update pipeline, List, and Gantt.

The next step is to embed Azure Synapse Analytics into the data pipeline. Azure Synapse Analytics can ingest, store, and analyze data from various sources. It can also be used to create machine learning models. In the Azure Synapse resource which I have created earlier, I created a SQL database to organize the workload. Here, I have two options to create a SQL database. Serverless and Dedicated. Serverless is a compute option (Azure manages this SQL pool relative to the workload). Dedicated is both compute and storage option. I have chosen the serverless option and created a database name with "gold_layer_db".

The screenshot shows the Microsoft Azure Synapse Analytics workspace interface. On the left, there's a sidebar with icons for Home, Data, Workspace, and Linked. The 'Data' section is currently selected. In the main area, there's a 'Create SQL database' dialog box. It includes a title 'Create SQL database', a description 'Create database to organize your workload into databases and database objects.', a 'Select SQL pool type' section with 'Serverless' selected, a 'Database' input field, and 'Create' and 'Cancel' buttons.

In the “Linked” section, I can see that the Azure Data Lake storage is already linked to synapse studio.

The screenshot shows the Microsoft Azure Synapse Analytics workspace interface. The 'Data' section is selected in the sidebar. The main area displays the 'Linked' section of the Data Explorer. It shows a list of linked storage accounts under 'Azure Data Lake Storage Gen2': 'analyticsspace (Primary - dev...)' which contains 'synapse-fs (Primary)', 'bronze', 'gold', and 'silver'; and '(Attached Containers)'. There are also icons for 'Select an item' and 'Use the resource explorer to select or create a new item'.

For an example query, I previewed the top 100 rows from the “Address” Table.

The screenshot shows the Microsoft Azure Synapse Analytics workspace interface. A red box highlights the top navigation bar. The main area displays a SQL script named "SQL script 1" in the "gold" database. The script uses OPENROWSET to bulk load data from a CSV file in Azure Storage into a temporary table named "[result]". The results pane shows two rows of address data from the "SalesLT.Address" table. The properties pane on the right shows the script is a ".sql script" of size 203 bytes, with the first 5000 rows selected by default.

```
-- This is auto-generated code
SELECT
    TOP 100 *
FROM
    OPENROWSET(
        BULK 'https://devstoragecustomer.dfs.core.windows.net/gold/SalesLT/Address/',
        FORMAT = 'DELTA'
    ) AS [result]
```

address_id	address_line1	address_line2	city	state_province	country_region	postal_code
9	8713 Yosemite ...	(NULL)	Bothell	Washington	United States	98011
11	1318 Lasalle Str...	(NULL)	Bothell	Washington	United States	98011

00:00:22 Query executed successfully.

Now, I created a view on top of this result to register a view. I can find this view in the workspace as below

The screenshot shows the Microsoft Azure Synapse Analytics workspace interface. A red box highlights the left sidebar under "Data". The "Views" section is expanded, showing a single view named "dbo.SalesLT_AddressView". The main area displays the same SQL script as the previous screenshot, but the results pane now shows "No results to show" because the view is registered but not yet populated with data. The properties pane remains the same.

```
CREATE VIEW SalesLT_AddressView AS
SELECT
    TOP 100 *
FROM
    OPENROWSET(
        BULK 'https://devstoragecustomer.dfs.core.windows.net/gold/SalesLT/Address/',
        FORMAT = 'DELTA'
    ) AS [result]
```

No results to show
Your query yielded no displayable results

00:00:17 Query executed successfully.

To create the views for each table dynamically, I used the following query in Azure Synapse

```

1 use gold_layer_db
2 GO
3
4 CREATE OR ALTER PROC CreateSQLServerlessView_gold @ViewName nvarchar(100)
5 AS
6 BEGIN
7     DECLARE @statement VARCHAR(MAX)
8     SET @statement = N'CREATE OR ALTER VIEW ' + @ViewName + ' AS
9         SELECT *
10        FROM
11            OPENROWSET(
12                BULK "https://devstoragecustomer.dfs.core.windows.net/gold/SalesLT/" + @ViewName + '/',
13                FORMAT = ''DELTA''
14            ) AS [result]
15    EXEC (@statement)
16 END
17 GO

```

Properties

- General Related (0)
- Name *
- Description
- Type .sql script
- Size 0 bytes
- Results settings per query
 - First 5000 rows (default)
 - All rows

Results Messages

00:00:03 Query executed successfully.

After publishing the script, I created a pipeline that uses the above stored procedure. Before that, I created a linked service to connect this serverless SQL DB.

New linked service

Azure SQL

All Azure Compute Database File General

Azure SQL Database Azure SQL Database Managed Instance

Continue Cancel

I have chosen “Autoresolve Integration Runtime” as I am connecting with another service in Azure. In the Account selection method, I chose to give the details manually. For the fully qualified domain name, I copied the details from the “serverless SQL endpoint” from “settings” of workspace properties. For the database name, I have given “gold_layer_db” as I am currently using this database. For the authentication, I have selected “System Assigned Managed Identity”. This uses my email address to connect to the database (gold_layer_db)

The screenshot shows the Microsoft Azure Synapse Analytics interface. On the left, there's a navigation sidebar with options like 'Synapse live', 'Analytics pools', 'SQL pools', 'Apache Spark pools', 'Data Explorer pools (pre...)', 'External connections', 'Linked services', 'Microsoft Purview', 'Integration' (which is expanded to show 'Triggers' and 'Integration runtimes'), 'Security', 'Access control', 'Credentials', and 'Managed private endpoints'. The main area is titled 'Linked services' and shows a list of existing linked services: 'analyticsspace-WorkspaceDefault...' (Azure Synapse Analytics) and 'analyticsspace-WorkspaceDefault...' (Azure Data Lake Storage). A red box highlights the top right corner of the page, which contains the URL 'web.azure-synapse.net/en/management/datalinkedservices?workspace=%2Fsubscriptions%2Fc18ffbb2-e46e-476b-aa92-aad49b43bbb3%2Fresourcegroup...', a search bar, and several small icons.

New linked service

Azure SQL Database [Learn more](#)

AutoResolveIntegrationRuntime

Version

Recommended Legacy

[Import from connection string](#)

Account selection method

From Azure subscription Enter manually

Fully qualified domain name *

analyticsspace-ondemand.sql.azuresynapse.net

Database name *

gold_layer_db

Authentication type *

System-assigned managed identity

Managed identity name: analyticsspace
Managed identity object ID: 91a30257-b9d9-41a5-b71e-a54694efb46f

Create Back Test connection Cancel

I can use this linked service to access the stored procedure from the pipeline.

Then, I created a pipeline using the “integrate” option in synapse studio. Here I had to add a new dataset. As there was no existing dataset, I created a new dataset by selecting “Azure Data Lake storage”. The format of the file would be “binary”. The path would be pointing to the gold layer in the container.

The screenshot shows the Microsoft Azure Synapse Analytics interface in 'Integrate' mode. The left sidebar shows 'Pipelines' with one item named 'Pipeline 1'. The main area is titled 'New integration dataset' and shows a list of activities: 'Get m' (selected), 'Get Metadata', 'General', and 'Dataset * Select...'. Below this, there's a 'Validate' section with a 'Get Metadata' button. To the right, there's a 'Data store' section with a 'Search' bar and a grid of data store icons categorized under 'All', 'Azure', 'Database', and 'File'. A red box highlights the top right corner of the page, which contains the URL 'web.azure-synapse.net/en/authoring/orchestrate/pipeline/Pipeline%201?workspace=%2Fsubscriptions%2Fc18ffbb2-e46e-476b-aa92-aad49b43bbb3%2Fresourcegroup...', a search bar, and several small icons.

New integration dataset

In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. [Learn more](#)

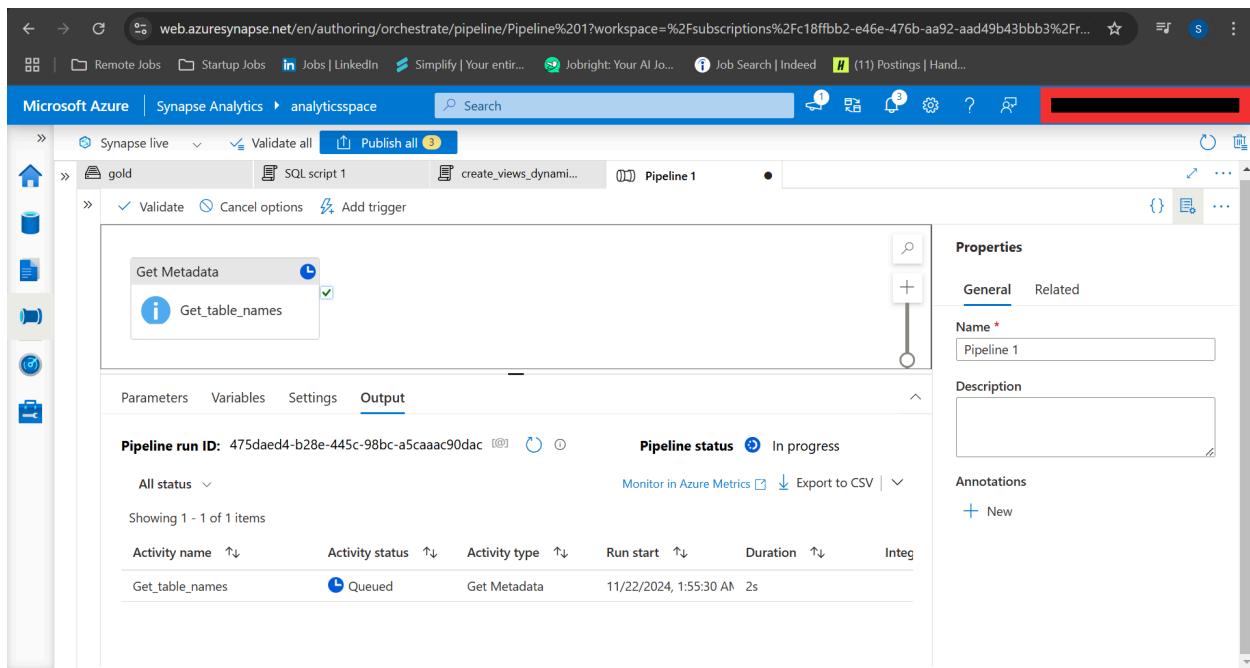
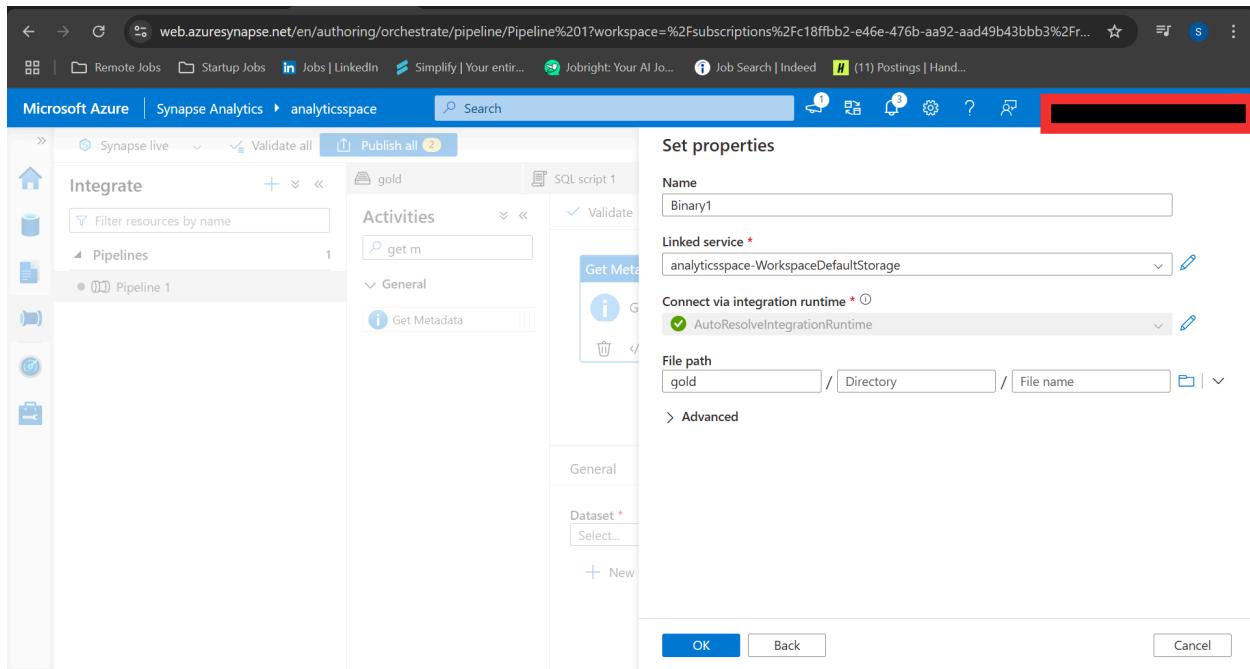
Select a data store

Search

All Azure Database File

Amazon RDS for SQL Server Amazon S3 Amazon S3 Compatible

Dataset * Select... New



The output JSON after debugging will look like below

The screenshot shows the Microsoft Azure Synapse Analytics studio interface. At the top, it displays the workspace path: Microsoft Azure | Synapse Analytics > analyticsspace. Below the header, there are buttons for "Synapse live", "Validate all", and "Publish all". The main area shows a pipeline named "gold". The pipeline has three stages: "Validate", "Debug", and "Add trigger". The "Output" stage is currently selected, displaying the following JSON object:

```
"childItems": [ { "name": "Address", "type": "Folder" }, { "name": "Customer", "type": "Folder" } ]
```

Below the JSON, there are two activities listed: "Get_table_names" and "Get Metadata". The "Get_table_names" activity is marked as "Succeeded".

Now, I have the list of the table names that I can use. Now, I added a for each activity as I need to create a view for each table which was in the list. I need the “For Each” activity to run on the success of the “Get Metadata” activity. In the settings of the “For Each” activity, choose “Add Dynamic Content” and select “Get_table_names childItems”. The pipeline expression builder contains the following expression

```
@activity('Get_table_names').output.childItems
```

Pipeline expression builder

Add dynamic content below using any combination of **expressions**, **functions** and **system variables**.

```
@activity('Get_table_names').output.childItems
```

Clear contents

Activity outputs Parameters System variables Functions Variables

Search

Get_table_names
Get_table_names activity output

Get_table_names childItems
List of subfolders and files in the given folder

OK Cancel

Now, I created a new activity within the foreach activity to call the stored procedure for each table. so,I embedded a stored procedure in the “foreach” activity.

Properties

General Related

Name * Pipeline 1

Description

Annotations + New

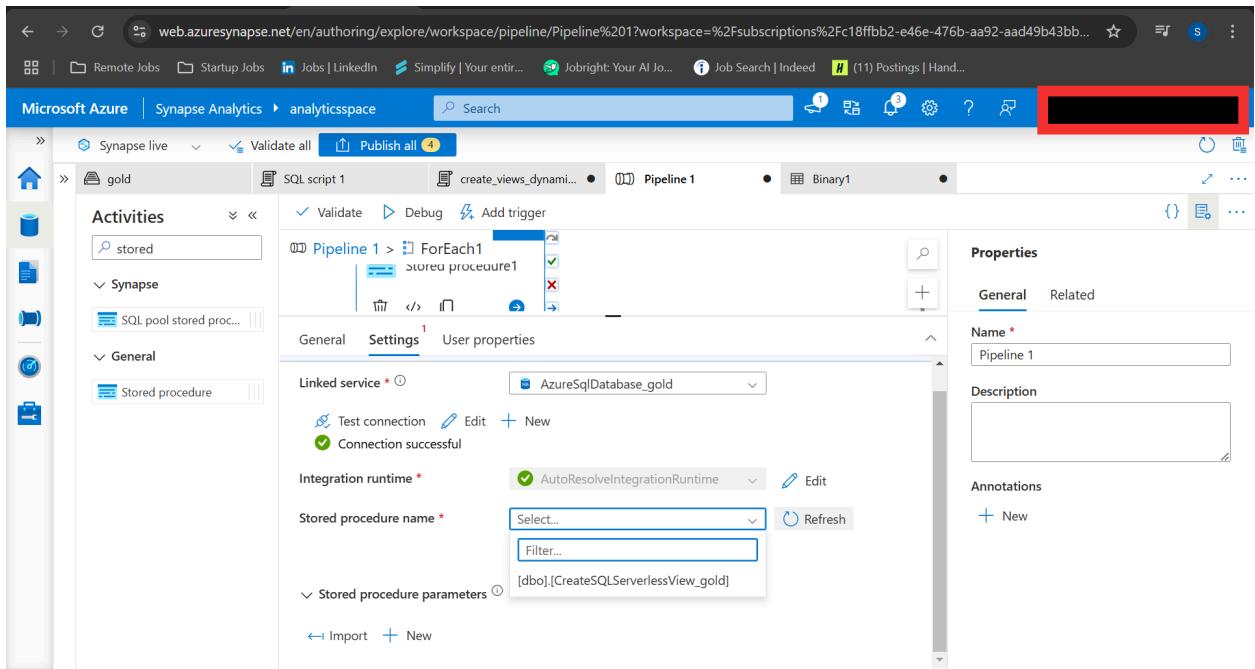
General Settings User properties

Name * Stored procedure1

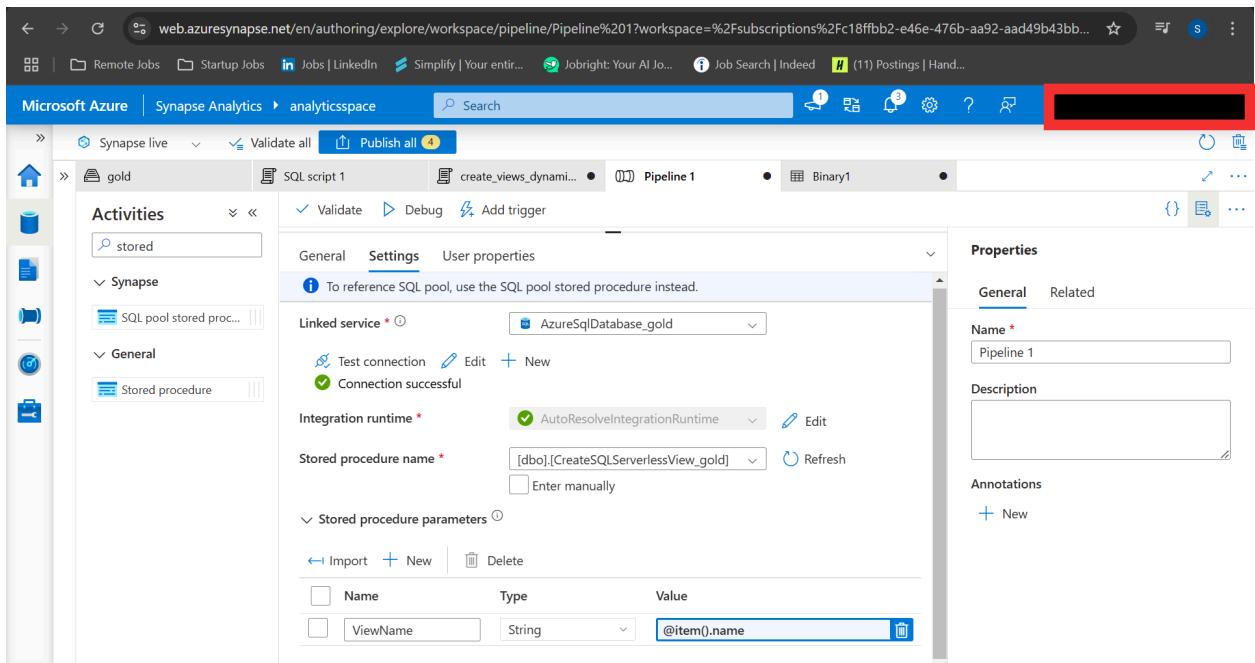
Description

Activity state Activated Deactivated

Choose “AzureSqlDatabase_gold” in the linked services available. In the stored procedure name, I selected the stored procedure which I had created earlier.



I created a new parameter for the stored procedure as shown below



The screenshot shows the Microsoft Azure Synapse Analytics pipeline publishing interface. On the left, there's a sidebar with icons for Home, Databases, Tables, Functions, Pipelines, Datasets, and Storage. The main area shows a pipeline named 'gold' under 'Activities'. It includes sections for General, Settings, and User properties. Under General, there's a 'Linked service' dropdown set to 'AzureSQLDatabase', a 'Test connection' button which shows 'Connection successful', and a 'Stored procedure name' input field set to '[dbo].[CreateSQLServer]'. Below these are sections for 'Integration runtime', 'Stored procedure parameters', and 'Import' options. On the right, a 'Pending changes (4)' summary is shown:

NAME	CHANGE	EXISTING
Pipelines		
Pipeline 1	(New)	-
Datasets		
Binary1	(New)	-
Linked services		
AzureSQLDatabase.gold	(New)	-
SQL script		
create_views_dynamically	(Edited)	create_views_dynamically

At the bottom right are 'Publish' and 'Cancel' buttons.

I triggered the pipeline using the “Trigger Now” option

The screenshot shows the Microsoft Azure Synapse Analytics pipeline run history. The left sidebar shows 'Data' with 'Workspace' selected, displaying a tree view of 'gold' database resources like 'gold_layer_db (SQL)', 'Views', 'Tables', 'Schemas', and 'Security'. The main area shows a pipeline named 'Pipeline 1' with its details: 'Activities' (including 'Synapse', 'Move and transform', 'Azure Data Explorer', etc.), 'Parameters' (with 12 items), 'Variables' (with 1 item), 'Settings' (with 1 item), and 'Output' (with 12 items). The 'Output' table lists the runs:

Activity name	Activity status	Activity type	Run start	Duration
Stored procedure1	In progress	Stored procedure	11/22/2024, 3:03:00 AM	29s
Stored procedure1	In progress	Stored procedure	11/22/2024, 3:03:00 AM	29s
Stored procedure1	In progress	Stored procedure	11/22/2024, 3:03:00 AM	29s
Stored procedure1	In progress	Stored procedure	11/22/2024, 3:03:00 AM	29s
Stored procedure1	In progress	Stored procedure	11/22/2024, 3:03:00 AM	29s
Stored procedure1	In progress	Stored procedure	11/22/2024, 3:03:00 AM	29s
Stored procedure1	In progress	Stored procedure	11/22/2024, 3:03:00 AM	29s
Stored procedure1	In progress	Stored procedure	11/22/2024, 3:03:00 AM	29s
Stored procedure1	In progress	Stored procedure	11/22/2024, 3:03:00 AM	29s
ForEach1	In progress	ForEach	11/22/2024, 3:02:59 AM	29s

The screenshot shows the Microsoft Azure Synapse Analytics interface. On the left, there's a sidebar with options like Analytics pools, Activities, and Integration. The main area is titled "Pipeline 1 - Activity runs" and displays a table of pipeline runs. The columns include Activity name, Activity status, Activity type, Run start, Duration, Integration runtime, and User. All activities listed are "Succeeded".

Activity name	Activity status	Activity type	Run start	Duration	Integration runtime	User
Get_table_names	Succeeded	Get Metadata	11/22/2024, 3:04:08 AM	7s	AutoResolveIntegration	
ForEach1	Succeeded	ForEach	11/22/2024, 3:04:15 AM	16s	AutoResolveIntegration	
Stored procedure1	Succeeded	Stored procedure	11/22/2024, 3:04:16 AM	12s	AutoResolveIntegration	
Stored procedure1	Succeeded	Stored procedure	11/22/2024, 3:04:16 AM	13s	AutoResolveIntegration	
Stored procedure1	Succeeded	Stored procedure	11/22/2024, 3:04:16 AM	10s	AutoResolveIntegration	
Stored procedure1	Succeeded	Stored procedure	11/22/2024, 3:04:16 AM	11s	AutoResolveIntegration	
Stored procedure1	Succeeded	Stored procedure	11/22/2024, 3:04:16 AM	11s	AutoResolveIntegration	
Stored procedure1	Succeeded	Stored procedure	11/22/2024, 3:04:16 AM	12s	AutoResolveIntegration	
Stored procedure1	Succeeded	Stored procedure	11/22/2024, 3:04:16 AM	11s	AutoResolveIntegration	
Stored procedure1	Succeeded	Stored procedure	11/22/2024, 3:04:16 AM	12s	AutoResolveIntegration	

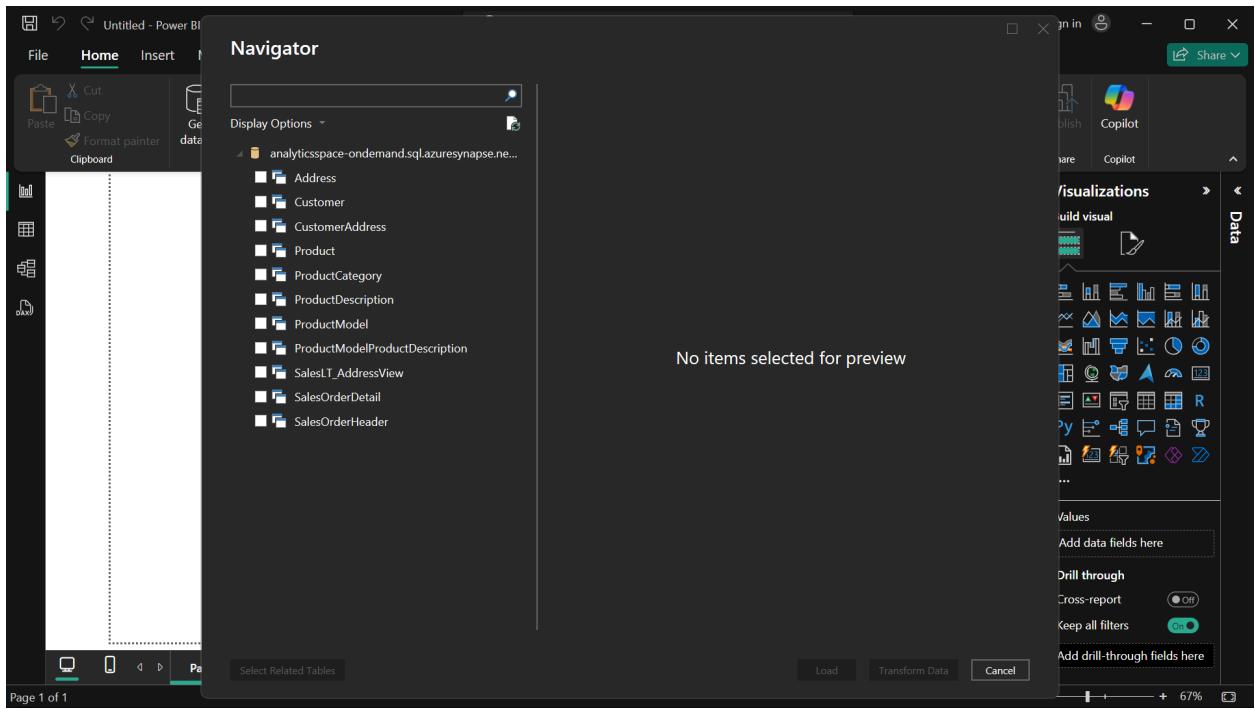
I can also see that the views are created dynamically as shown below

The screenshot shows the Microsoft Azure Synapse Analytics workspace. The left sidebar has options like Home, Data, Develop, Integrate, Monitor, and Manage. The main area shows a pipeline named "Pipeline 1" with its activities. The activities include a "Get Metadata" step followed by a "ForEach" loop containing a "Stored procedure..." step. Below the pipeline, there's a table of pipeline runs with columns for Activity name, Activity status, and Activity type.

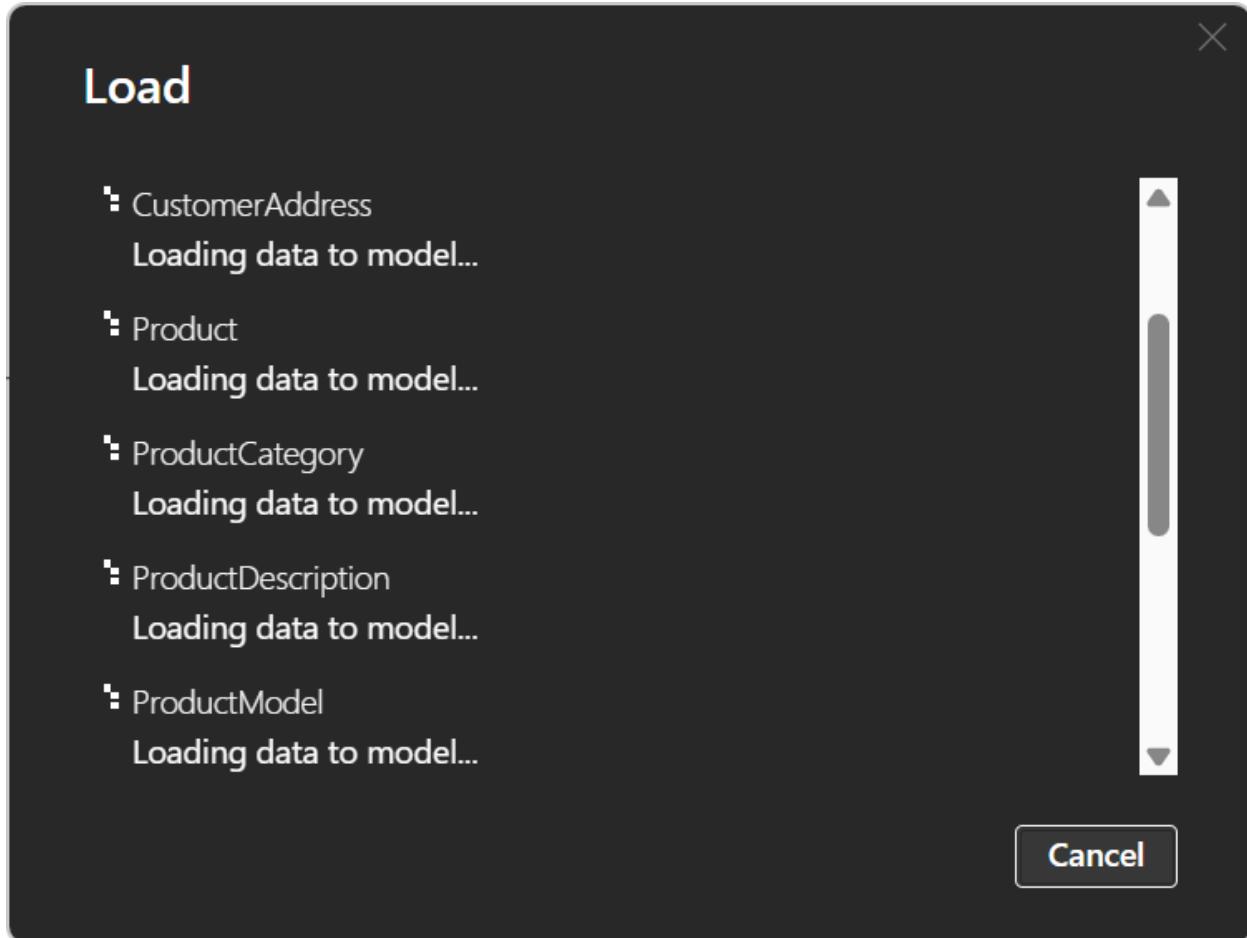
Activity name	Activity status	Activity type
Get_Metadata	Succeeded	Get Metadata
ForEach1	Succeeded	ForEach
Stored procedure1	Succeeded	Stored procedure

In the next step, I connected the azure synapse to the Power BI desktop. While connecting, I used the following credentials in the authentication

In the server name , I have added the details of SQL serverless endpoint. In the database details, I gave the name “gold_layer_db” (the database which I have created in synapse earlier).once the synapse is connected to power BI, I can see all the tables in power BI as below



Also, I have chosen the “import” mode in Power BI, which means the data is essentially in the internal memory of Power BI. I proceeded to load all the tables



Once the data is loaded, I can see all the data in table view section of Power BI

Untitled - Power BI Desktop

File Home Help Table tools

Structure Calendars Relationships Calculations

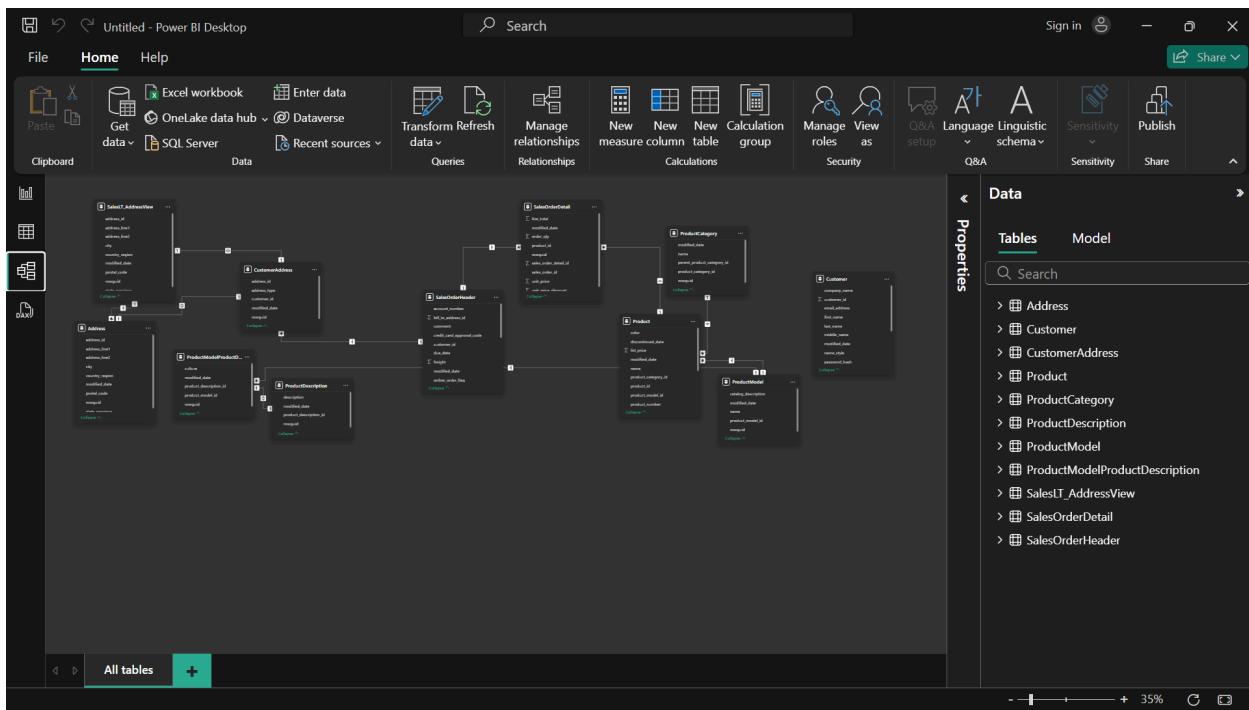
Data

Search

Address Customer CustomerAddress Product ProductCategory ProductDescription ProductModel ProductModelProductDescription SalesLT_AddressView SalesOrderDetail SalesOrderHeader

address_id	address_line1	address_line2	city	state_province	country_region	postal_code	rowguid
988	482505 Warm Springs Blvd.		Fremont	California	United States	94536	cd6d22b0-d941-4928-b6f7-1508c
989	39933 Mission Oaks Blvd		Camarillo	California	United States	93010	8b1241ca-1db6-4f9d-961e-1d6cc
991	60025 Bollinger Canyon Road		San Ramon	California	United States	94583	962a4d00-c5aa-41cc-b7bc-2afbe;
992	9992 Whipple Rd		Union City	California	United States	94587	88258a1b-b50f-440f-93b9-5d7af
993	Corporate Office		El Segundo	California	United States	90245	6cccc74f-6389-4d08-8ed1-9079b
994	25001 Montague Expressway		Milpitas	California	United States	95035	89901b2e-fda8-4925-b22c-dd5d5
995	4460 Newport Center Drive		Newport Beach	California	United States	92625	d6b9e9e9-f288-4590-be12-8ddbf
997	70259 West Sunnyview Ave		Visalia	California	United States	93291	7c2e0511-de61-4761-80ea-6a7b2
998	60750 San Clemente		Hayward	California	United States	94541	822dc2be-aac2-4283-93a6-5027c
999	Receiving		Fullerton	California	United States	92831	93d3c32f-cc26-4f27-b91f-35a804
1000	22555 Paseo De Las Americas		San Diego	California	United States	92102	982bcd22-6429-4725-a6df-4fa03
1001	Incom Sports Center		Ontario	California	United States	91764	079e5de8-169f-4eb6-be75-813b4
1003	5967 W Las Positas Blvd		Pleasanton	California	United States	94566	6b26292f-1ebe-41ca-93a2-f42d3
1004	25600 E St Andrews Pl		Santa Ana	California	United States	92701	489390e-048c-4bb4-a5dc-38c4t
1005	6756 Mowry		Newark	California	United States	94560	cfb5eda2-43fe-4102-8413-73c74c
1006	25472 Marlay Ave		Fontana	California	United States	92335	90c5f983-b302-4f6c-9d8a-ffb9e1
1011	9700 Sisk Road		Modesto	California	United States	95354	f8563347-c157-43b3-a7ed-a8514
1013	54254 Pacific Ave.		Stockton	California	United States	95202	7b82a39f-d175-4543-8f6f-ae6fe9
1014	25136 Jefferson Blvd.		Culver City	California	United States	90232	05196d3f-f726-4ab9-a66b-a8f75c
1015	99000 S. Avalon Blvd. Suite 750		Carson	California	United States	90746	20616d9-0324-476d-909e-607fe
1016	72502 Eastern Ave.		Bell Gardens	California	United States	90201	06431d74-4caf-442c-870e-efc20f
1018	630 N. Capitol Ave.		San Jose	California	United States	95112	aceb7a03-c8e6-43f6-94f2-266e57

Table: Address (450 rows)



I can find the relationships between the tables using “Manage” relationships table as below

Manage relationships

<input type="checkbox"/> From: table (column) ↑	Relationship	To: table (column)	Status	⋮
<input type="checkbox"/> Address (address_line1)	* ← → 1	SalesLT_AddressView (address...)	Inactive	⋮
<input type="checkbox"/> CustomerAddress (address_id)	1 ← → 1	Address (address_id)	Active	⋮
<input type="checkbox"/> CustomerAddress (address_id)	1 ← → 1	SalesLT_AddressView (address...)	Active	⋮
<input type="checkbox"/> CustomerAddress (customer_id)	* ← → 1	SalesOrderHeader (customer_id)	Active	⋮
<input type="checkbox"/> Product (product_category_id)	* ← → 1	ProductCategory (product_cat...)	Active	⋮
<input type="checkbox"/> Product (product_model_id)	* ← → 1	ProductModel (product_mode...)	Active	⋮
<input type="checkbox"/> ProductModelProductDescript...	1 ← → 1	ProductDescription (product_d...)	Active	⋮
<input type="checkbox"/> ProductModelProductDescript...	* ← → 1	ProductModel (product_mode...)	Active	⋮
<input type="checkbox"/> SalesOrderDetail (product_id)	* ← → 1	Product (product_id)	Active	⋮
<input type="checkbox"/> SalesOrderDetail (sales_order_i...	* ← → 1	SalesOrderHeader (sales_order...)	Active	⋮

Close

Before proceeding to design any dashboard, I thoroughly read the business requirement and tried to narrow down the questions that the business is interested in.

From the requirement, following questions can be more relevant

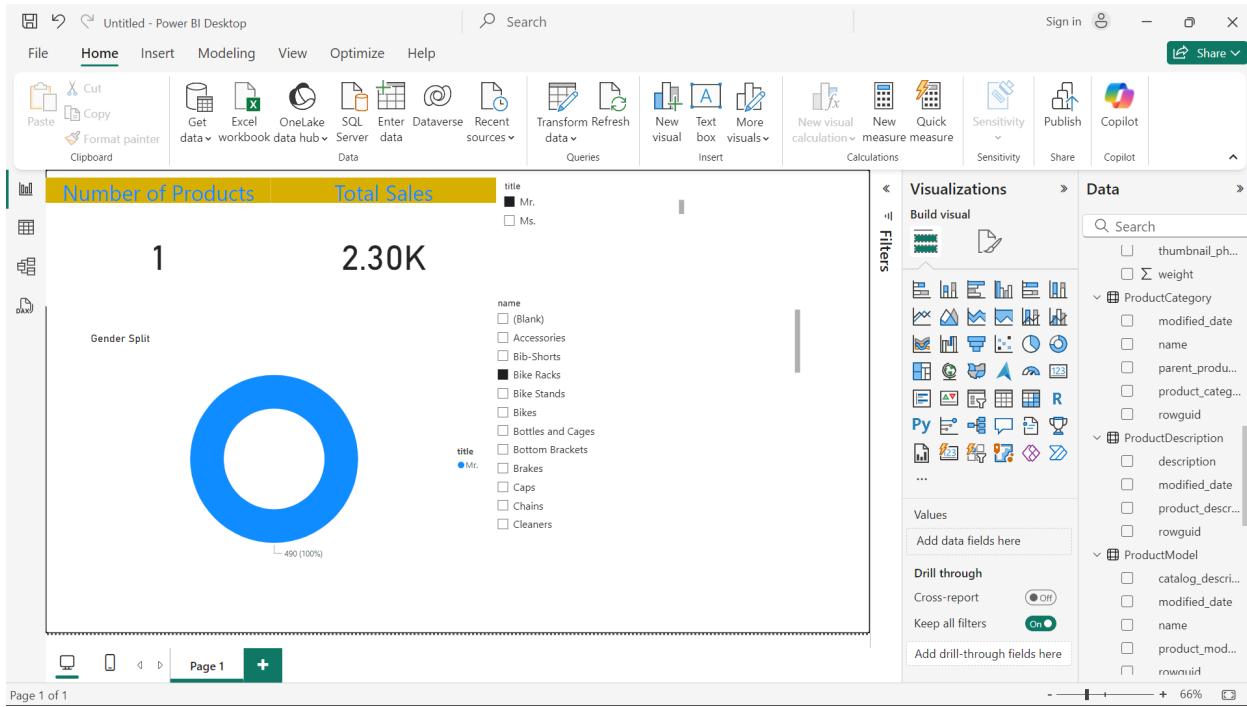
What are the distinct products that are available?

What are the total sales?

For a given gender, what are the sales for a particular product?

To answer the above, I created the following dashboard. I created the dashboard in such a way that the user can apply filters on both product category and gender to show the sum of sales.

Below is the snapshot of the dashboard



Resume bullet points

Designed and automated a scalable ETL pipeline using Azure Data Factory to extract customer and sales data from an **on-premises SQL Server**, loading it into **Azure Data Lake Storage** in the **Bronze layer** of the **Medallion Architecture**, ensuring secure integration through **Azure Key Vault**.

Transformed and modeled datasets using Azure Databricks, leveraging **PySpark** and **SQL** to process data from **Bronze to Silver and Gold layers**, enabling dynamic **data warehousing** and enriched analytics through structured **data modeling**.

Delivered actionable insights by developing an interactive **Power BI dashboard** that visualized KPIs such as **sales revenue**, **products sold**, and **gender distribution**, utilizing **Azure Synapse Analytics** for efficient querying and implementing **data governance** to ensure compliance and security.

Future Scope:

Future Scope of the Project

1. **Real-Time Data Integration and Analysis:**
 - Enhance the pipeline by incorporating **real-time data streaming** capabilities using tools like **Azure Stream Analytics** or **Apache Kafka**. This will allow stakeholders to access up-to-the-minute sales metrics and demographic insights, providing greater agility in responding to emerging trends and customer behavior.
2. **Advanced Predictive Analytics and Personalization:**
 - Implement **machine learning models** using **Azure Machine Learning** and **Databricks** to forecast sales trends, identify potential product preferences by demographic segments, and recommend personalized product offerings. This will elevate the utility of the KPI dashboard from reporting to decision support.
3. **Scalability and Multi-Region Support:**
 - Extend the pipeline to handle data ingestion and reporting for multiple regions or stores, enabling a unified view of customer demographics and sales trends across geographies. Utilize **Azure Synapse Analytics**' dedicated pools for optimized performance on high-volume data processing.
4. **Enhanced Dashboard Features:**
 - Integrate **geographic analysis** and drill-down capabilities into the **Power BI dashboard** to offer richer visualizations, such as region-specific gender distribution and product sales insights. Add predictive visualizations, like projected sales by category and demographic trends over time.
5. **Broader Data Source Integration:**
 - Expand data sources to include **social media analytics**, **CRM systems**, and **web traffic logs** to enrich customer demographic data, providing a more comprehensive understanding of customer preferences and engagement.
6. **Data Governance and Compliance:**
 - Implement advanced data governance tools such as **Azure Purview** to manage data lineage, ensure compliance with regulations like GDPR, and maintain data quality across the pipeline. Introduce **fine-grained access controls** to secure sensitive customer data effectively.
7. **Incremental and Historical Data Processing:**
 - Transition the pipeline from a daily overwrite model to an **incremental load approach** using **Delta Lake**'s time-travel features. This will enable efficient updates and maintain a history of changes for longitudinal analysis.
8. **Automation and CI/CD Pipelines:**
 - Introduce **CI/CD pipelines** for the data engineering workflows using tools like **Azure DevOps** or **GitHub Actions**, ensuring seamless deployment, testing, and version control for all components of the data pipeline.
9. **Self-Service Analytics for Stakeholders:**
 - Develop self-service capabilities within **Power BI** or as APIs, allowing business users to perform ad hoc queries, customize reports, and explore data insights independently.
10. **Integration with Business Processes:**

- Leverage insights from the pipeline and dashboard to trigger automated workflows using **Azure Logic Apps** or **Power Automate**, such as sending inventory alerts or initiating targeted marketing campaigns based on sales trends and gender-based preferences.