Azure End-To-End Data Engineering Project Report

This report outlines the steps I followed to complete the Azure End-to-End Data Engineering Project. The project focuses on building a data pipeline using Azure services, starting from data ingestion, followed by transformation, storage, orchestration, and visualization. The goal is to create a scalable and automated data processing pipeline.

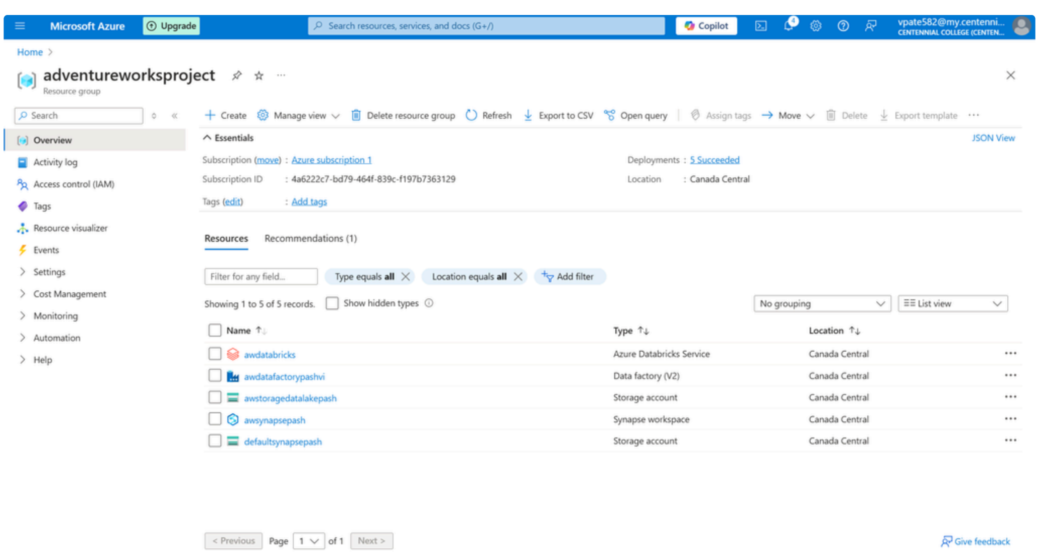
DATASET

- The dataset used in this project is the **AdventureWorks** dataset from Kaggle. It is a widely-used sample database that contains transactional data for a fictitious multinational manufacturing company. It includes tables for sales, products, customers, and employees.
- The dataset is structured with various relational tables, such as:
  - **SalesOrderHeader**
  - **SalesOrderDetail**
  - **Product**
  - **Customer**
  - **Employee**

PROJECT SETUP

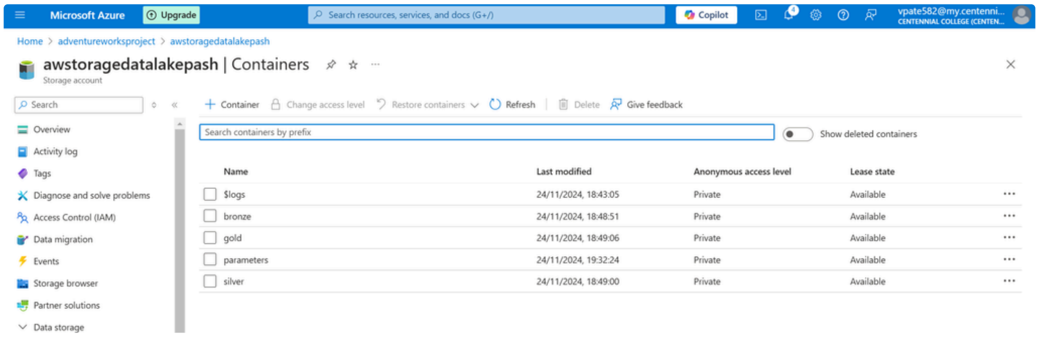
2.1 Environment Setup

- **Azure Subscription**: I started by using an existing Azure account to access the services required for the project.
- **Resource Group**: I created a new resource group on Azure to organize all the resources related to this project.



Blob Storage Setup

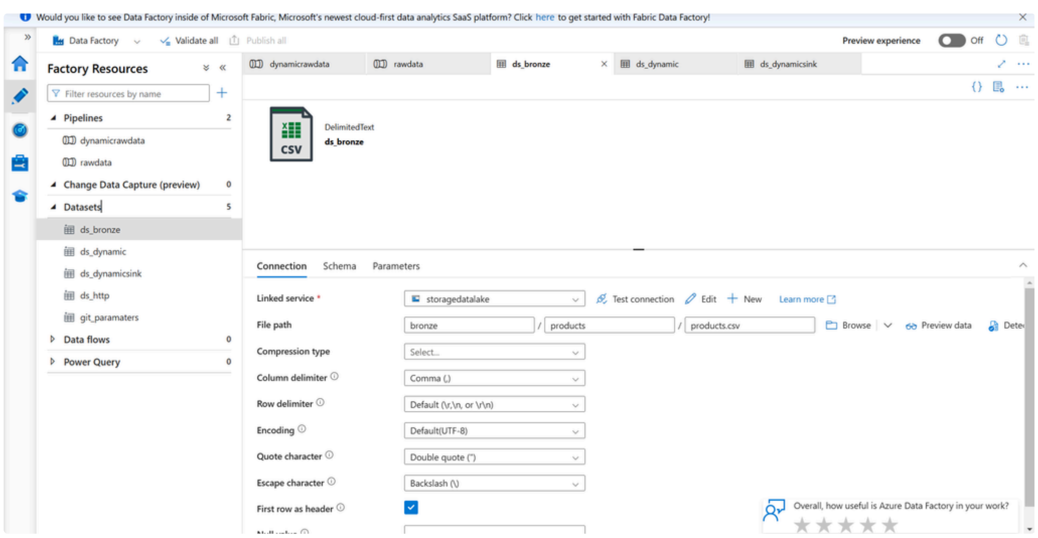
- I uploaded the **AdventureWorks** dataset to **Azure Blob Storage**, using CSV format for easy access and manipulation during the pipeline process.

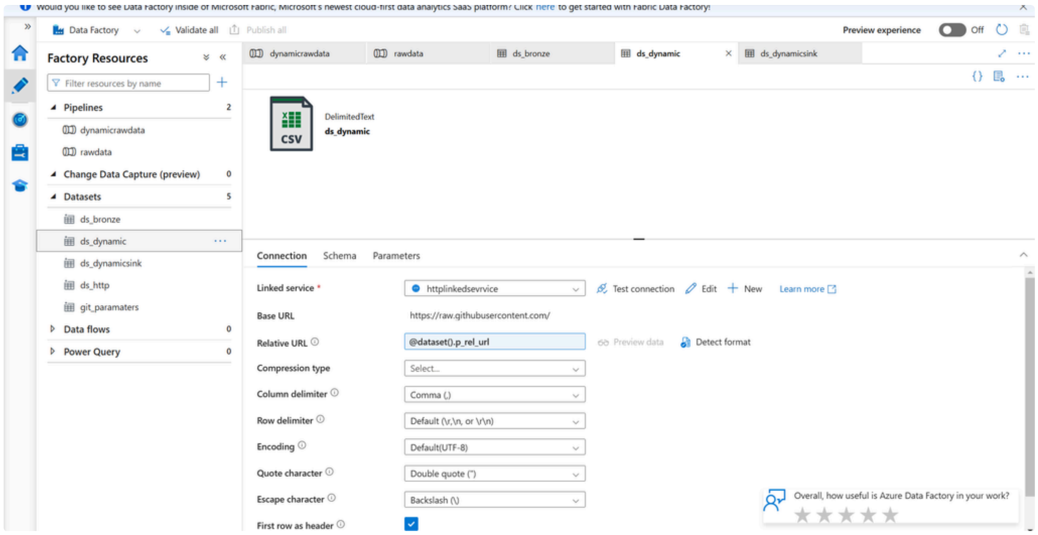


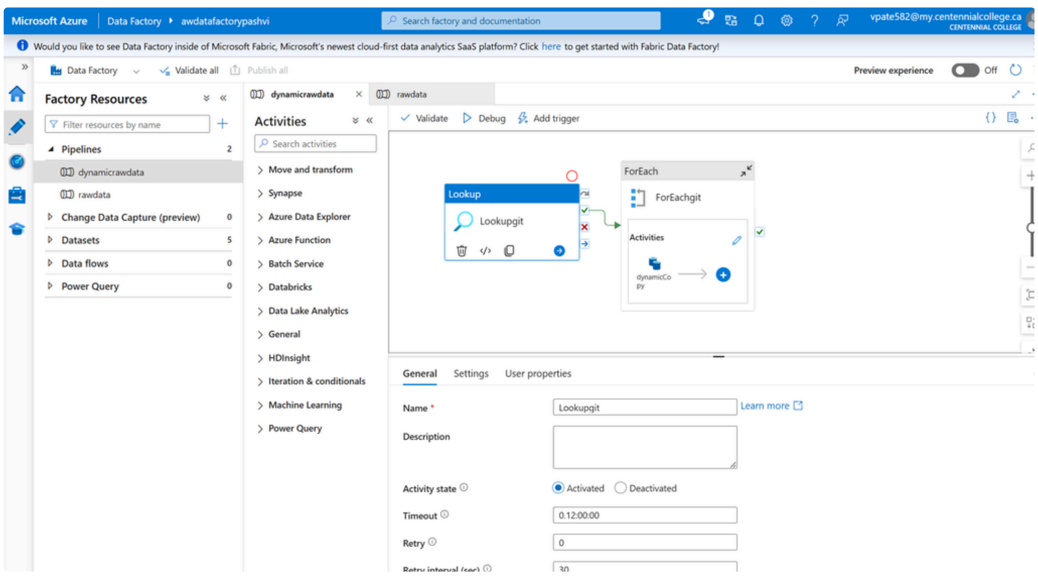
Creating a Dynamic Pipeline in Azure Data Factory

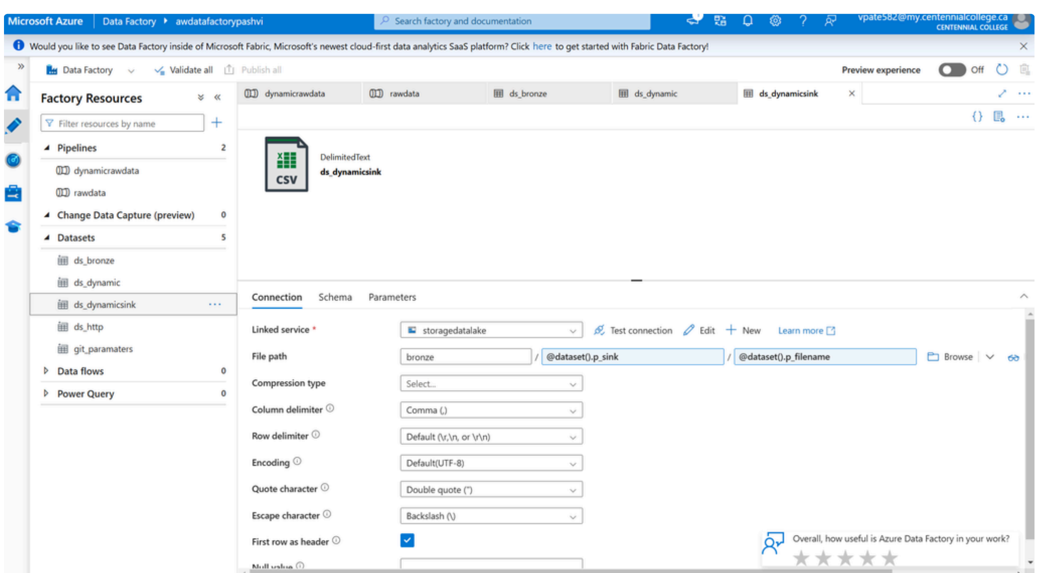
To streamline the data ingestion process, I created a **dynamic pipeline** in **Azure Data Factory (ADF)**. This pipeline was designed to automatically ingest all files from the **Blob Storage** container without the need for hardcoding individual file names. The steps to achieve this dynamic ingestion were as follows:

- **Get List of Files**: The first step in the pipeline was to use the **Get Metadata** activity to dynamically list all the files within the Blob Storage container. This allowed the pipeline to detect any new or updated files in the container automatically.
- **ForEach Activity**: After retrieving the list of files, I used the **ForEach** activity in ADF to iterate through each file. This loop processed each file dynamically, regardless of the number or name of the files. Inside the **ForEach** loop, I used the **Copy Data** activity to ingest each file into Azure Data Lake or Azure SQL Database.
- **Dynamic Parameters**: To handle different file names and paths, I used **parameters** within the pipeline to pass dynamic values, such as file names and paths, to the copy activity. This ensured that the pipeline was flexible and scalable, accommodating new files added to Blob Storage without needing to modify the pipeline.



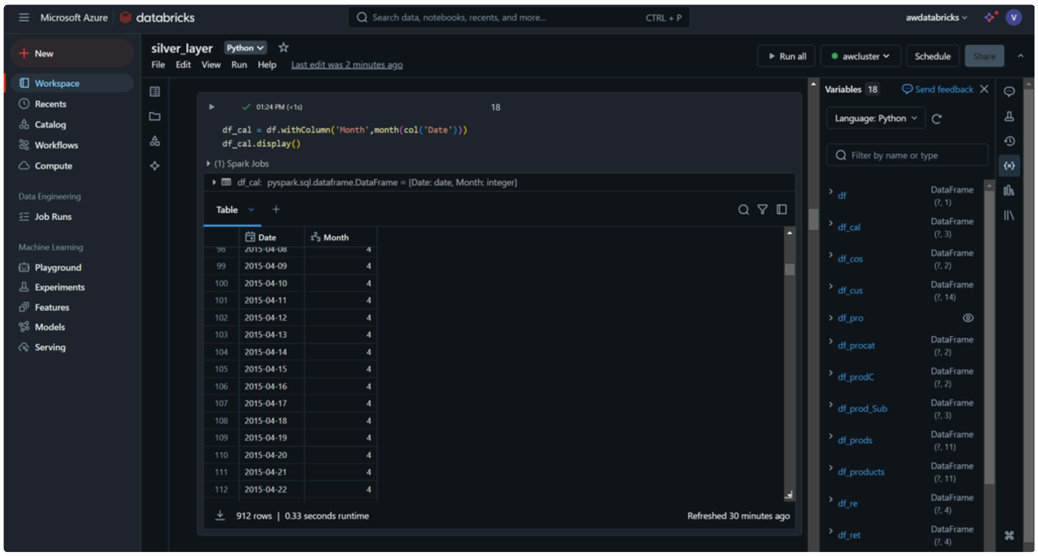

Would you like to see Data Factory inside of Microsoft Fabric, Microsoft's newest cloud-first data analytics SaaS platform? Click here to get started with Fabric Data Factory!
Data Factory
Validate all
Publish all
Preview experience
Off
Factory Resources
Filter resources by name
Pipelines 2
dynamicrawdata
rawdata
Change Data Capture (preview) 0
Datasets 5
ds_bronze
ds_dynamic
ds_dynamicsink
ds_http
git_paramaters
Data flows 0
Power Query 0
dynamicrawdata
rawdata
ds_bronze
ds_dynamic
ds_dynamicsink
DelimitedText
ds_dynamic
CSV
Connection
Schema
Parameters
Linked service *
httplinkedsevrvice
Test connection
Edit
New
Learn more
Base URL
https://raw.githubusercontent.com/
Relative URL
@dataset().p_rel_url
Preview data
Detect format
Compression type
Select...
Column delimiter
Comma (,)
Row delimiter
Default (\r,\n, or \r\n)
Encoding
Default(UTF-8)
Quote character
Double quote (")
Escape character
Backslash (\)
First row as header
Overall, how useful is Azure Data Factory in your work?

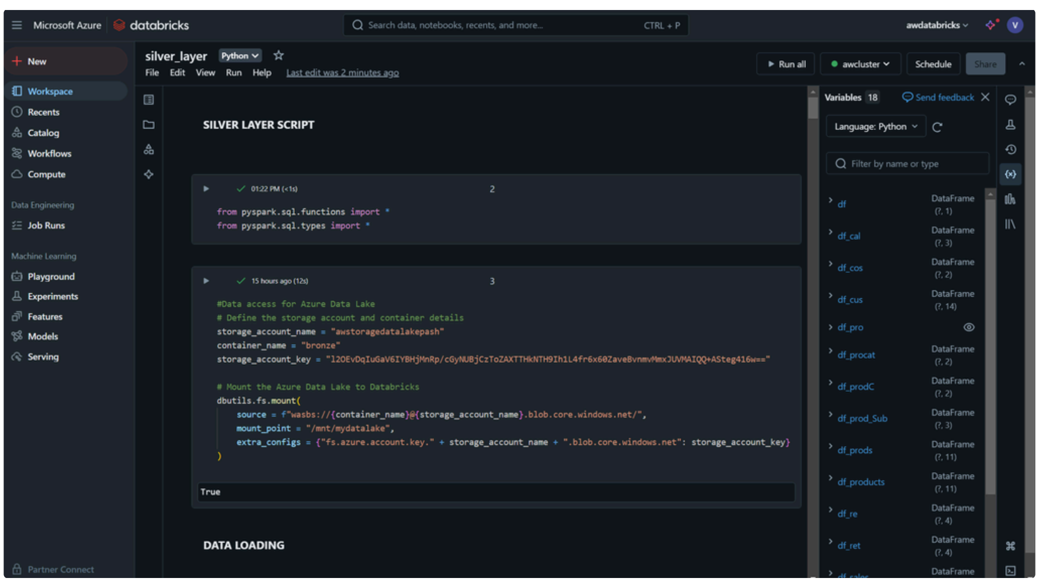

Microsoft Azure
Data Factory
awdatafactorypashvi
Search factory and documentation
vpate582@my.centennialcollege.ca
CENTENNIAL COLLEGE
Would you like to see Data Factory inside of Microsoft Fabric, Microsoft's newest cloud-first data analytics SaaS platform? Click here to get started with Fabric Data Factory!
Data Factory
Validate all
Publish all
Preview experience
Off
Factory Resources
Filter resources by name
Pipelines 2
dynamicrawdata
rawdata
Change Data Capture (preview) 0
Datasets 5
Data flows 0
Power Query 0
dynamicrawdata
rawdata
Activities
Search activities
Move and transform
Synapse
Azure Data Explorer
Azure Function
Batch Service
Databricks
Data Lake Analytics
General
HDInsight
Iteration & conditionals
Machine Learning
Power Query
Validate
Debug
Add trigger
Lookup
Lookupgit
ForEach
ForEachgit
Activities
General
Settings
User properties
Name *
Lookupgit
Learn more
Description
Activity state
Activated
Deactivated
Timeout
0.12:00:00
Retry
0

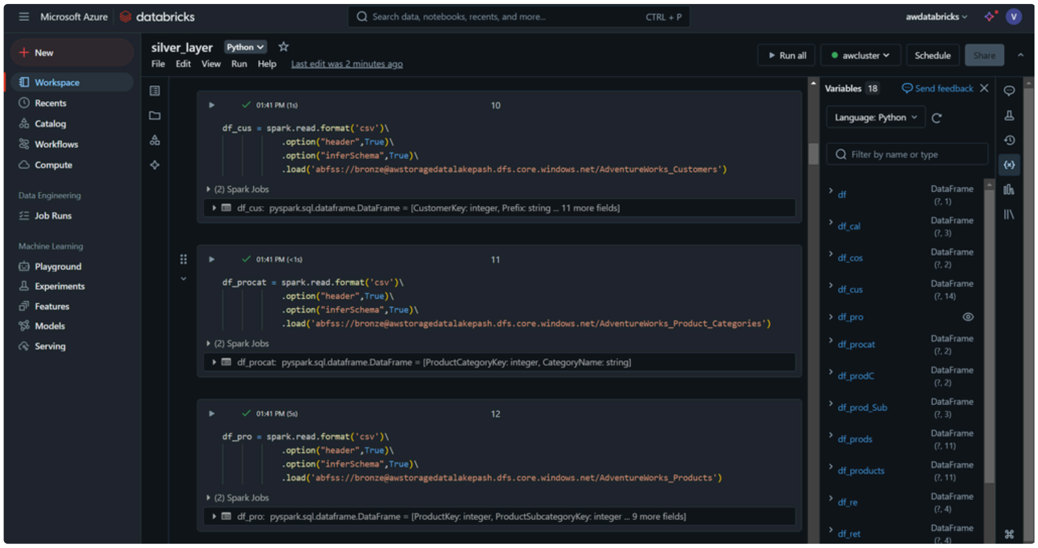

Microsoft Azure
Data Factory
awdatafactorypashvi
Search factory and documentation
Would you like to see Data Factory inside of Microsoft Fabric, Microsoft's newest cloud-first data analytics SaaS platform? Click here to get started with Fabric Data Factory!
Data Factory
Validate all
Publish all
Preview experience
Off
Factory Resources
Filter resources by name
Pipelines 2
dynamicrawdata
rawdata
Change Data Capture (preview) 0
Datasets 5
ds_bronze
ds_dynamic
ds_dynamicsink
ds_http
git_paramaters
Data flows 0
Power Query 0
dynamicrawdata
rawdata
ds_bronze
ds_dynamic
ds_dynamicsink
DelimitedText
ds_dynamicsink
CSV
Connection
Schema
Parameters
Linked service *
storagedatalake
Test connection
Edit
New
Learn more
File path
bronze
@dataset().p_sink
@dataset().p_filename
Browse
Compression type
Select...
Column delimiter
Comma (,)
Row delimiter
Default (\r,\n, or \r\n)
Encoding
Default(UTF-8)
Quote character
Double quote (")
Escape character
Backslash (\)
First row as header
Overall, how useful is Azure Data Factory in your work?

The **AdventureWorks** dataset was analyzed for its key tables and columns that would be relevant for transformation. The main transformation tasks involved:

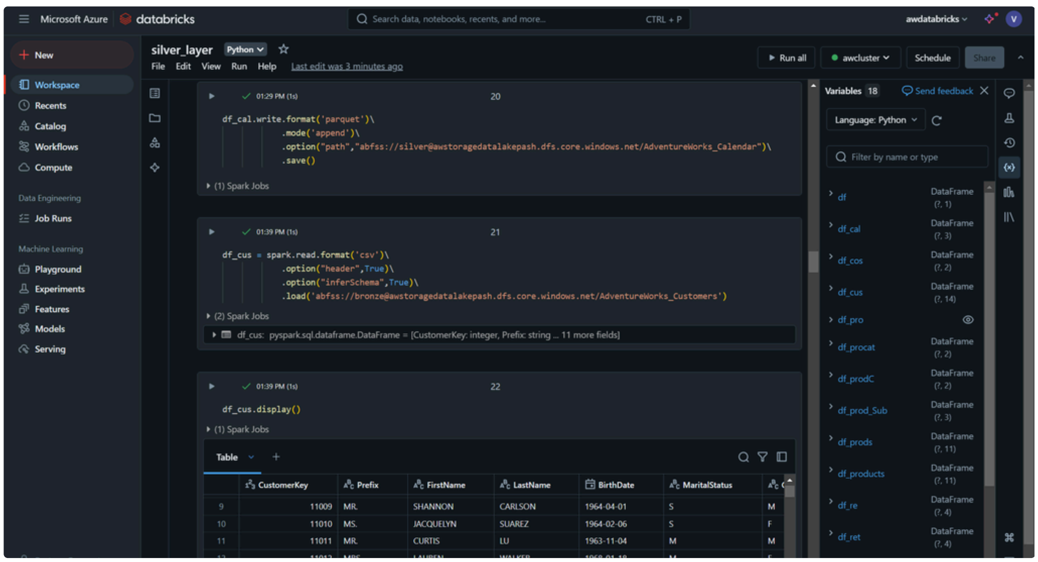
- **Data Cleansing**: Removing any records with missing values or duplicates, particularly in critical fields like order IDs or customer details.
- **Data Aggregation**: Combining multiple related tables to get meaningful insights such as total sales by customer or product category.
- **Feature Engineering**: Creating new columns, by combining **SalesOrderDetail** line items and grouping by order.



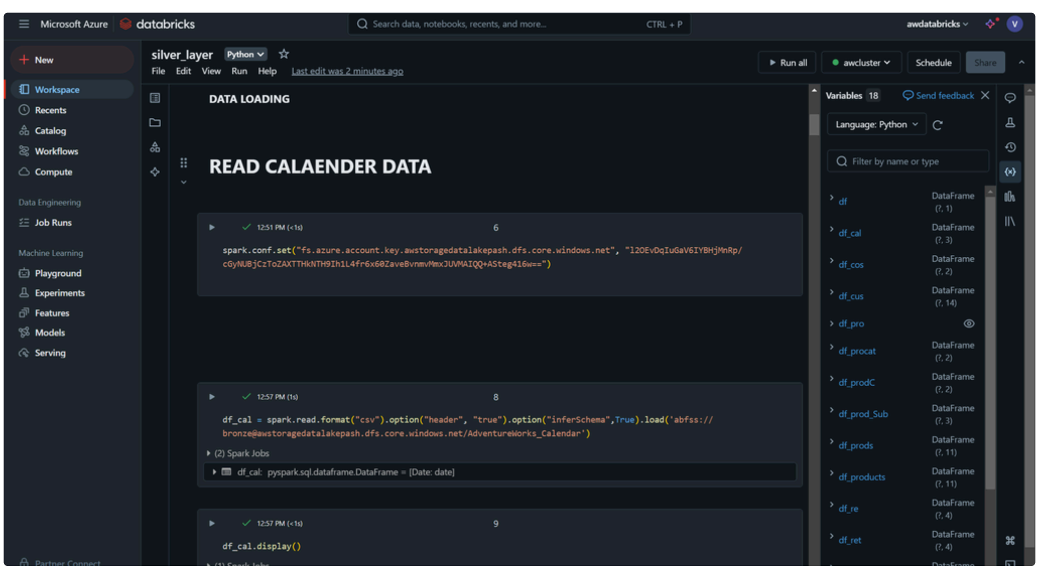




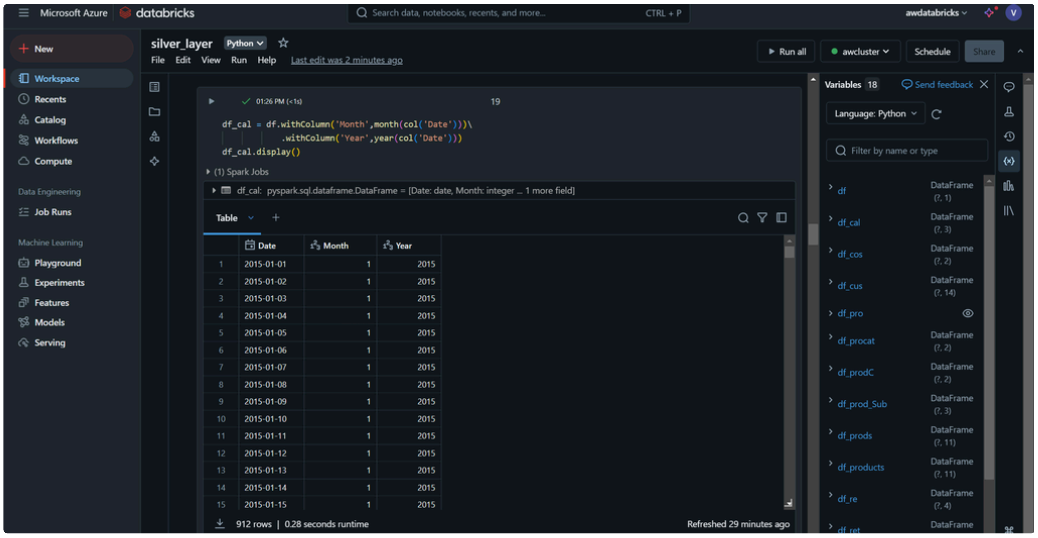
Microsoft Azure
databricks
silver_layer
Python
File Edit View Run Help
Last edit was 2 minutes ago
Run all
awcluster
Schedule
Share
01:41 PM (1s)
10
df_cus = spark.read.format('csv')\
.option("header",True)\
.option("inferSchema",True)\
.load('abfss://bronze@awstoragedatalakepash.dfs.core.windows.net/AdventureWorks_Customers')
(2) Spark Jobs
df_cus: pyspark.sql.dataframe.DataFrame = [CustomerKey: integer, Prefix: string ... 11 more fields]
01:41 PM (<1s)
11
df_procat = spark.read.format('csv')\
.option("header",True)\
.option("inferSchema",True)\
.load('abfss://bronze@awstoragedatalakepash.dfs.core.windows.net/AdventureWorks_Product_Categories')
(2) Spark Jobs
df_procat: pyspark.sql.dataframe.DataFrame = [ProductCategoryKey: integer, CategoryName: string]
01:41 PM (5s)
12
df_pro = spark.read.format('csv')\
.option("header",True)\
.option("inferSchema",True)\
.load('abfss://bronze@awstoragedatalakepash.dfs.core.windows.net/AdventureWorks_Products')
(2) Spark Jobs
df_pro: pyspark.sql.dataframe.DataFrame = [ProductKey: integer, ProductSubcategoryKey: integer ... 9 more fields]

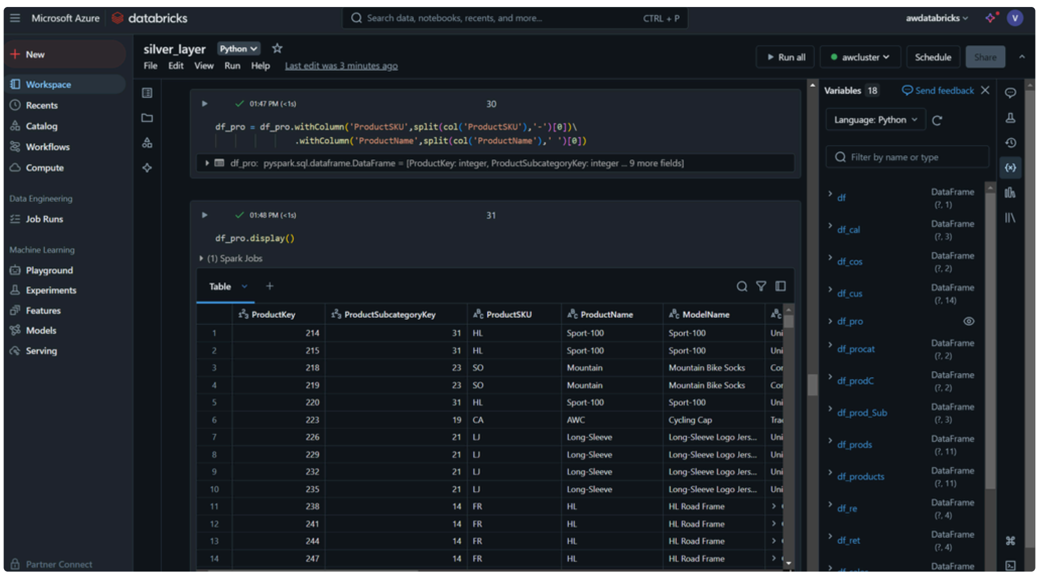


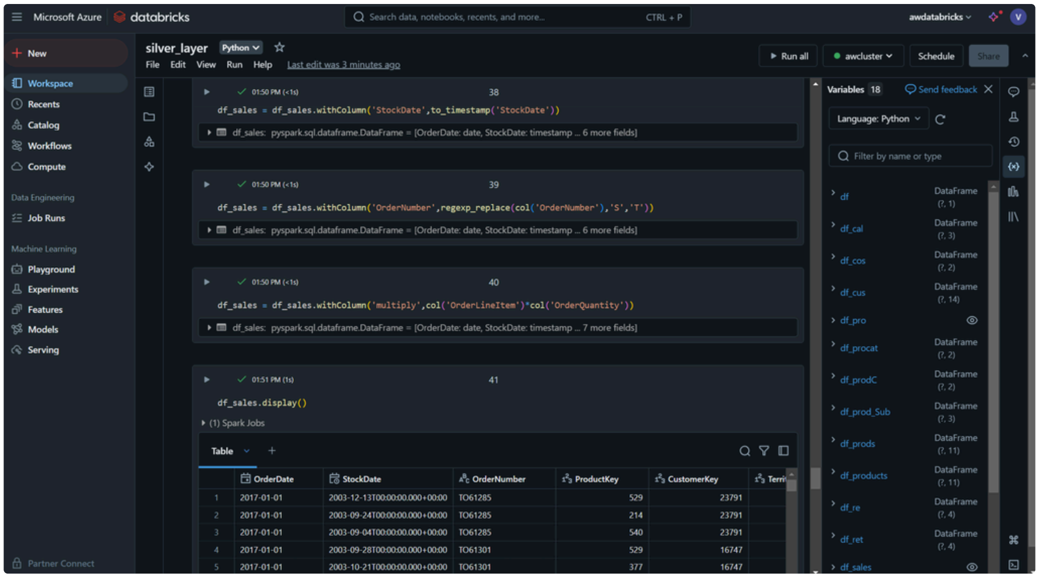
Microsoft Azure
databricks
silver_layer
Python
File Edit View Run Help
Last edit was 3 minutes ago
01:29 PM (1s)
20
df_cal.write.format('parquet')\
.mode('append')\
.option("path","abfss://silver@awstoragedatalakepash.dfs.core.windows.net/AdventureWorks_Calendar")\
.save()
(1) Spark Jobs
01:39 PM (1s)
21
df_cus = spark.read.format('csv')\
.option("header",True)\
.option("inferSchema",True)\
.load('abfss://bronze@awstoragedatalakepash.dfs.core.windows.net/AdventureWorks_Customers')
(2) Spark Jobs
df_cus: pyspark.sql.dataframe.DataFrame = [CustomerKey: integer, Prefix: string ... 11 more fields]
01:39 PM (1s)
22
df_cus.display()
(1) Spark Jobs
Table
CustomerKey Prefix FirstName LastName BirthDate MaritalStatus
9 11009 MR. SHANNON CARLSON 1964-04-01 S M
10 11010 MS. JACQUELYN SUAREZ 1964-02-06 S F
11 11011 MR. CURTIS LU 1963-11-04 M M



Microsoft Azure
databricks
silver_layer
Python
File Edit View Run Help
Last edit was 2 minutes ago
DATA LOADING
READ CALAENDER DATA
12:51 PM (<1s)
6
spark.conf.set("fs.azure.account.key.awstoragedatalakepash.dfs.core.windows.net", "l2OEvDqIuGaV6IYBHjMnRp/cGyNUBjCzToZAXTTHkNTH9Ih1L4fr6x60ZaveBvnmvMmxJUVMAIQQ+ASteg416w==")
12:57 PM (1s)
8
df_cal = spark.read.format("csv").option("header", "true").option("inferSchema",True).load('abfss://bronze@awstoragedatalakepash.dfs.core.windows.net/AdventureWorks_Calendar')
(2) Spark Jobs
df_cal: pyspark.sql.dataframe.DataFrame = [Date: date]
12:57 PM (<1s)
9
df_cal.display()

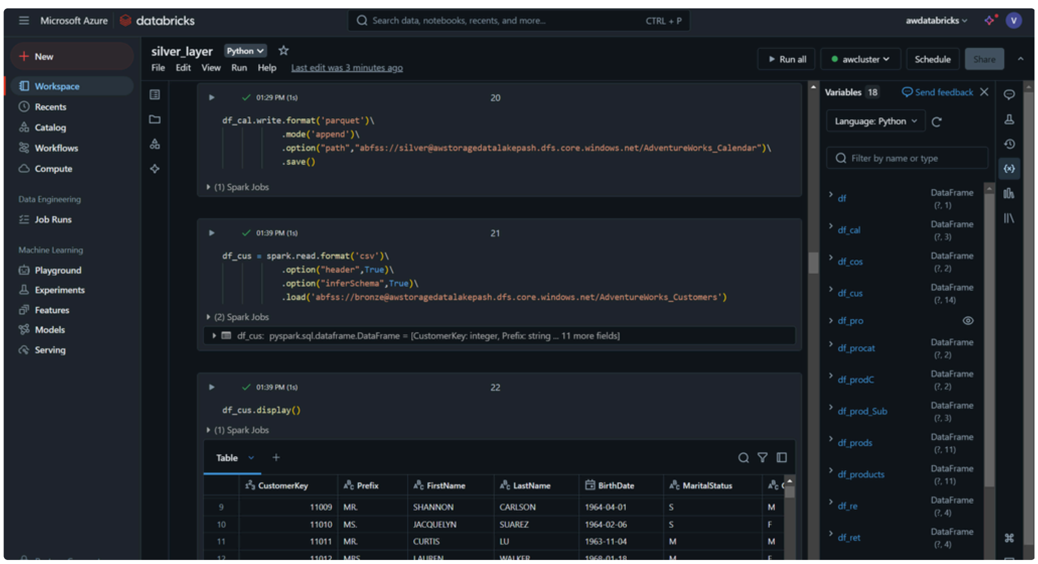

Microsoft Azure
databricks
silver_layer
Python
File Edit View Run Help
Last edit was 2 minutes ago
Run all
awcluster
Schedule
Share
df_cal = df.withColumn('Month',month(col('Date')))\
.withColumn('Year',year(col('Date')))
df_cal.display()
(1) Spark Jobs
df_cal: pyspark.sql.dataframe.DataFrame = [Date: date, Month: integer ... 1 more field]
Table
Date
Month
Year
912 rows | 0.28 seconds runtime
Refreshed 29 minutes ago
Variables 18
Send feedback
Language: Python
Filter by name or type

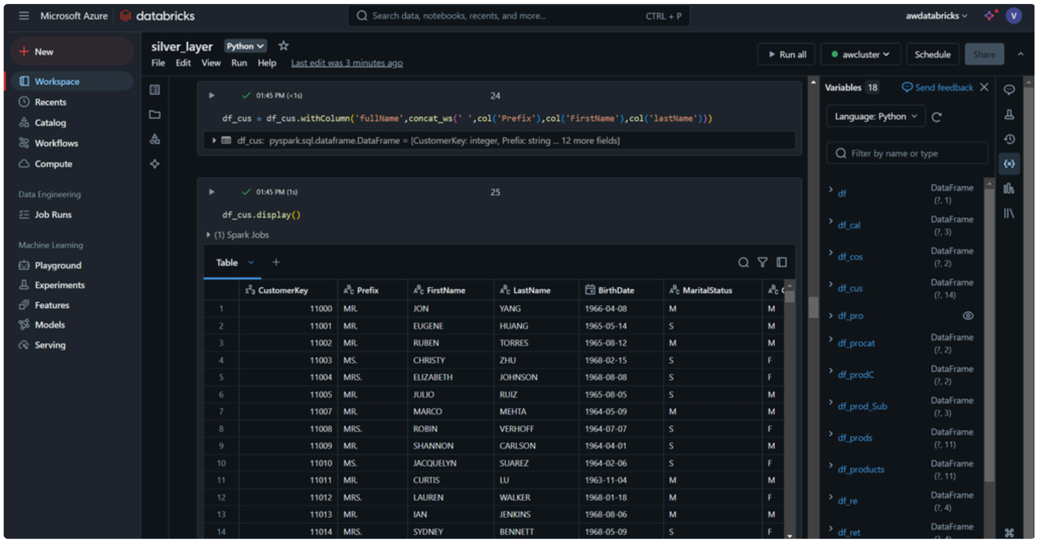

Microsoft Azure
databricks
silver_layer
Python
File Edit View Run Help
Last edit was 3 minutes ago
Run all
awcluster
Schedule
Share
df_pro = df_pro.withColumn('ProductSKU',split(col('ProductSKU'),'-')[0])\
.withColumn('ProductName',split(col('ProductName'),' ')[0])
df_pro: pyspark.sql.dataframe.DataFrame = [ProductKey: integer, ProductSubcategoryKey: integer ... 9 more fields]
df_pro.display()
(1) Spark Jobs
Table
ProductKey
ProductSubcategoryKey
ProductSKU
ProductName
ModelName
Variables 18
Send feedback
Language: Python
Filter by name or type

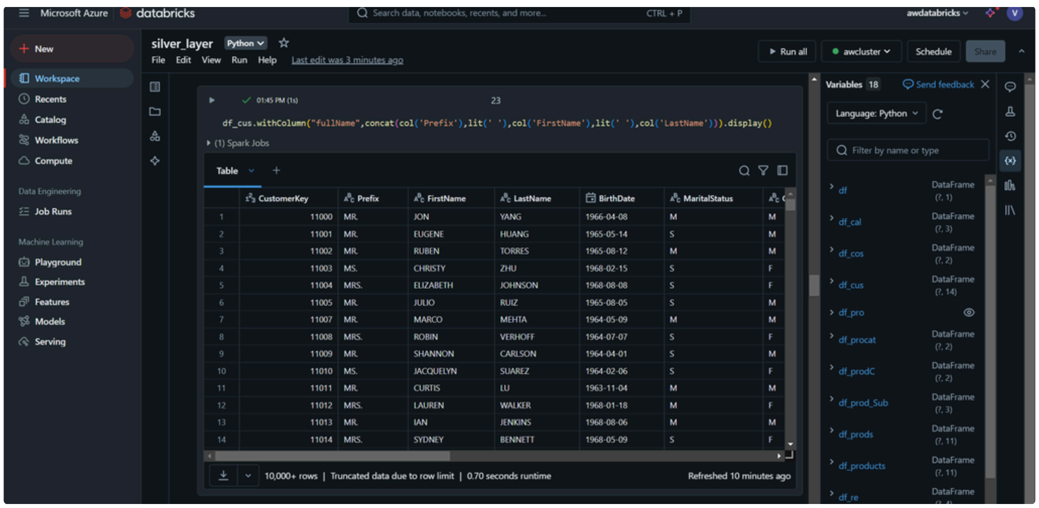


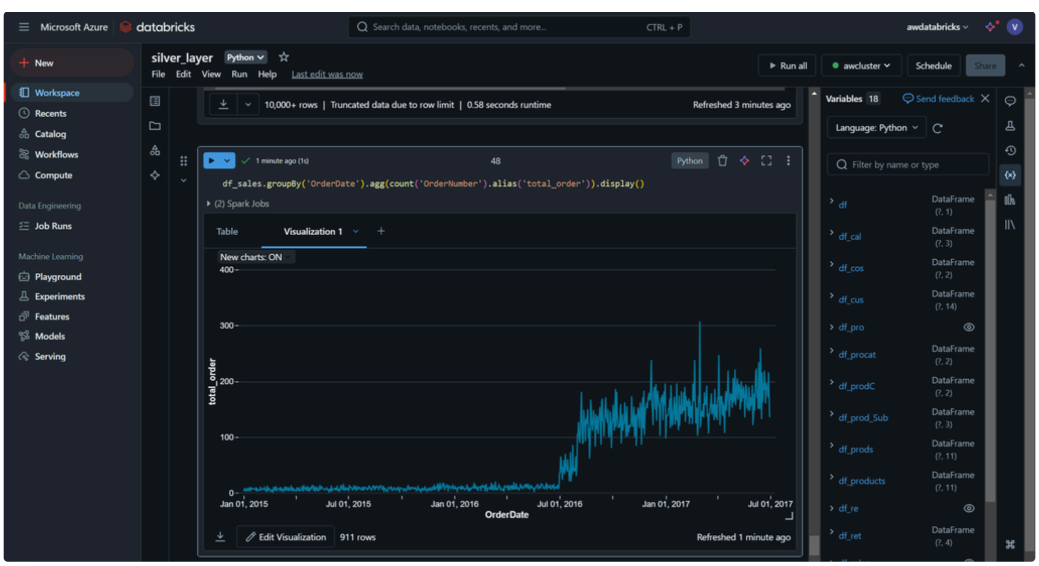
Using **Azure Databricks**, I applied **PySpark** to perform data transformations, including:

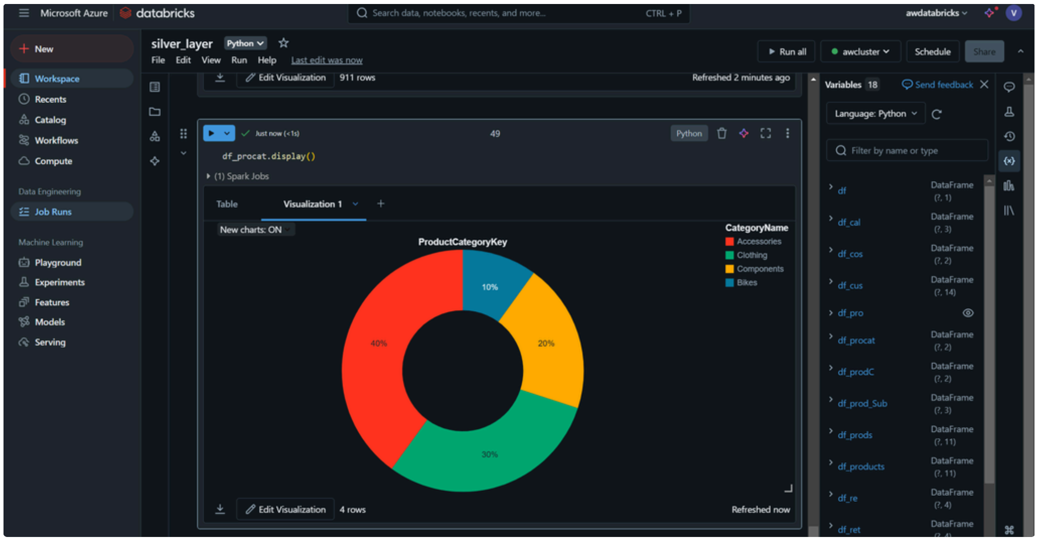
- **Join operations** to combine related tables
- **Data filtering** to clean out incomplete or incorrect records.
- **GroupBy operations** to summarize data by customer, region, or product.

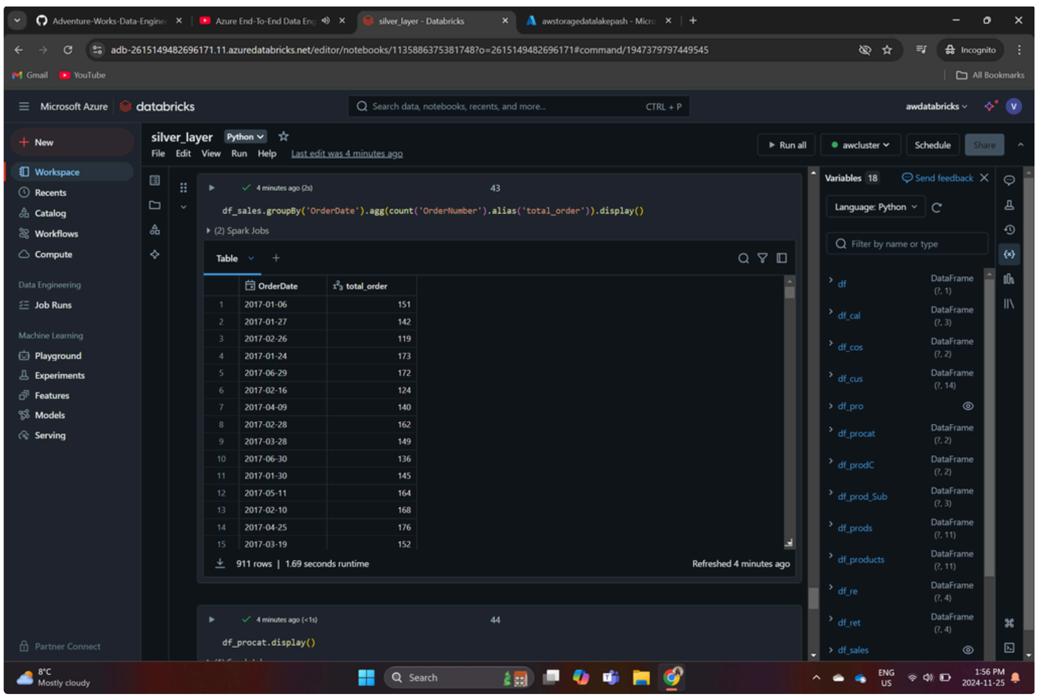



silver_layer
df_cus = df_cus.withColumn('fullName',concat_ws(' ',col('Prefix'),col('FirstName'),col('lastName')))
df_cus: pyspark.sql.dataframe.DataFrame = [CustomerKey: integer, Prefix: string ... 12 more fields]
df_cus.display()
(1) Spark Jobs
Table
CustomerKey
Prefix
FirstName
LastName
BirthDate
MaritalStatus
Variables 18
Language: Python

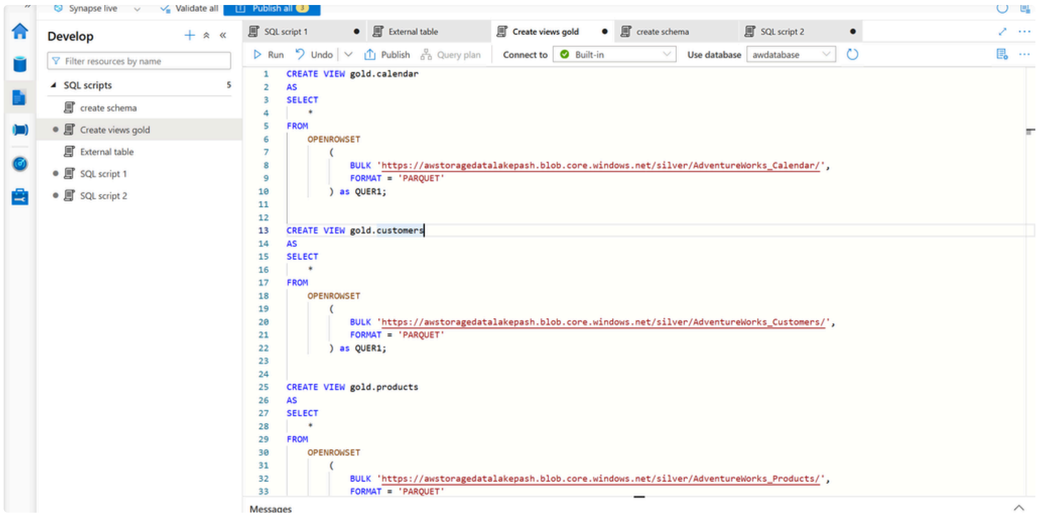

silver_layer
df_cus.withColumn("fullName",concat(col('Prefix'),lit(' '),col('FirstName'),lit(' '),col('LastName'))).display()
(1) Spark Jobs
Table
CustomerKey
Prefix
FirstName
LastName
BirthDate
MaritalStatus
10,000+ rows | Truncated data due to row limit | 0.70 seconds runtime
Refreshed 10 minutes ago

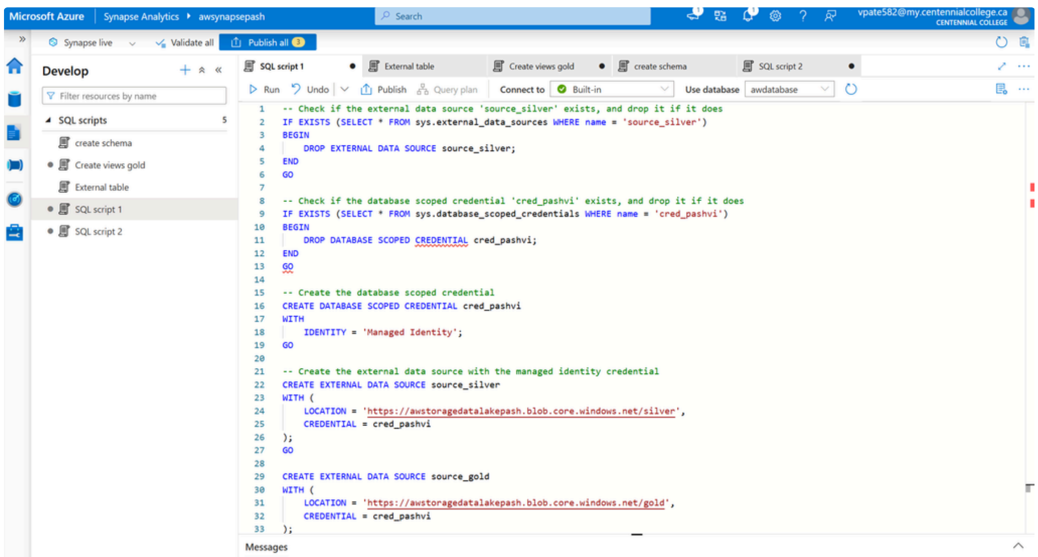

silver_layer
10,000+ rows | Truncated data due to row limit | 0.58 seconds runtime
Refreshed 3 minutes ago
df_sales.groupBy('OrderDate').agg(count('OrderNumber').alias('total_order')).display()
(2) Spark Jobs
Table
Visualization 1
New charts: ON
total_order
OrderDate
Jan 01, 2015
Jul 01, 2015
Jan 01, 2016
Jul 01, 2016
Jan 01, 2017
Jul 01, 2017
Edit Visualization
911 rows
Refreshed 1 minute ago





For data analytics, I used **Azure Synapse Analytics** to run SQL-based queries and derive business insights from the transformed data. Synapse integrated with both Data Lake Storage and Azure SQL Database for seamless querying.





Azure Logic Apps for Error Handling

- I configured **Azure Logic Apps** to send automated notifications in case of failures, ensuring that issues are quickly addressed.
- I set up **Azure Monitor** to track the performance of the entire data pipeline, including resource utilization and error tracking.

Challenges & Learnings

Throughout this project, I faced the following challenges:

- **Data Quality Issues**: Encountered missing or inconsistent data, which required careful handling during the transformation phase.
- **Pipeline Orchestration**: Ensuring that the data ingestion, transformation, and loading steps were coordinated effectively was a key challenge, especially when dealing with large datasets.

Learnings:

- Gained hands-on experience with core Azure services, such as Azure Data Factory, Databricks, and Synapse.
- Improved my skills in automation and monitoring within Azure.

The Azure End-to-End Data Engineering project successfully implemented an automated data pipeline using a range of Azure services. The project helped me build expertise in data ingestion, transformation, orchestration, and reporting, and I am confident that these skills will be valuable for future data engineering tasks.