PROJECT REPORT

<u>SERVERLESS DATA PROCESSING</u>

Submitted by

<u>GROUP 4</u>

SAI ANISH

SAKSHI RATHI

SARA PRICILLA

SWETHA

Batch : Data Engineering Batch 4

Submitted as part of : Data Engineering Final Project

Date of Submission : 17/07/2025

Introduction

In today's data-driven world, organizations require scalable, efficient, and cost-effective solutions to process and analyze large volumes of data. Traditional data processing pipelines often involve managing complex infrastructure, which increases operational overhead and reduces agility. To overcome these challenges, cloud-native and serverless architectures offer a modern alternative by abstracting infrastructure management and enabling pay-as-you-go compute models.

This project focuses on implementing a Serverless Data Processing and Machine Learning Pipeline using various services from the Microsoft Azure ecosystem. The goal is to design and execute an end-to-end workflow that can:

- Ingest and explore data without provisioning dedicated servers
- Automate ETL (Extract, Transform, Load) jobs using event-driven triggers
- Train and monitor machine learning models using scalable, on-demand resources
- Integrate DevOps practices for infrastructure management and CI/CD workflows

Key components of the system include Azure Data Lake Storage Gen2, Azure Synapse Serverless SQL, Azure Databricks, Azure Functions, MLflow, and Azure DevOps. This architecture provides a robust framework for data handling, machine learning, and continuous integration and delivery — all without maintaining traditional infrastructure.

Problem Statement

This project addresses the need for scalable, serverless data processing and machine learning workflows using Azure cloud services. The focus is on building a fully automated, event-driven pipeline for data exploration, transformation, and model training without managing traditional infrastructure.

ETL Problem

- Enable **raw data exploration** directly on **Azure Data Lake Storage Gen2** using **Azure Synapse Serverless SQL**, removing the need for data movement or provisioning SQL servers.

- Design an **event-triggered ETL pipeline** by using **Azure Functions** to automatically invoke **Azure Databricks** jobs for large-scale data transformation.

- Manage all source code and infrastructure through **Azure DevOps Repositories** and **ARM templates**, ensuring version control and repeatable deployments.

Machine Learning Problem

- Build **serverless training pipelines** in **Azure Databricks** using **on-demand Spark clusters** to reduce resource costs and improve scalability.

- Develop a **customer segmentation model** using the **KMeans algorithm** from **Spark MLlib**, trained on processed data.

- Automate deployment of the trained model using **CI/CD pipelines** in **Azure DevOps**, and monitor performance and metrics using **MLflow** for reproducibility and tracking.

Tools and Technologies Used

This project integrates a range of cloud-native services and development tools from the Microsoft Azure ecosystem to build a fully serverless data processing and machine learning pipeline. Each component plays a specific role in the end-to-end workflow:

Azure Data Lake Storage Gen2 (ADLS Gen2)

ADLS Gen2 serves as the primary data storage layer for the project. It is used to store raw datasets in a hierarchical, scalable, and cost-effective manner. The service provides seamless integration with analytics tools like Synapse and Databricks, allowing direct access to data without requiring ingestion or duplication.

Azure Synapse Analytics (Serverless SQL Pool)

Azure Synapse Serverless SQL is used for querying and exploring raw data stored in ADLS Gen2. This eliminates the need for pre-loading data into relational tables and offers a serverless, pay-per-query model for efficient ad-hoc analysis.

Azure Databricks

Azure Databricks is the core compute engine for both ETL and machine learning processes in this project. It leverages Apache Spark to perform large-scale data transformations and supports on-demand cluster creation, which reduces resource costs. Databricks also facilitates the training of the customer segmentation model using the KMeans algorithm from Spark MLlib.

Azure Functions

Azure Functions act as event-driven triggers for the data pipeline. In this project, they are used to automatically initiate Databricks jobs for ETL and ML processes. Functions are lightweight, serverless, and can be deployed and managed directly from the Visual Studio Code environment.

MLflow

MLflow is used for tracking machine learning experiments. Integrated with Databricks, it captures key model metrics such as the silhouette score, logs hyperparameters, and manages model versioning. It ensures reproducibility and transparency in the training and deployment phases.

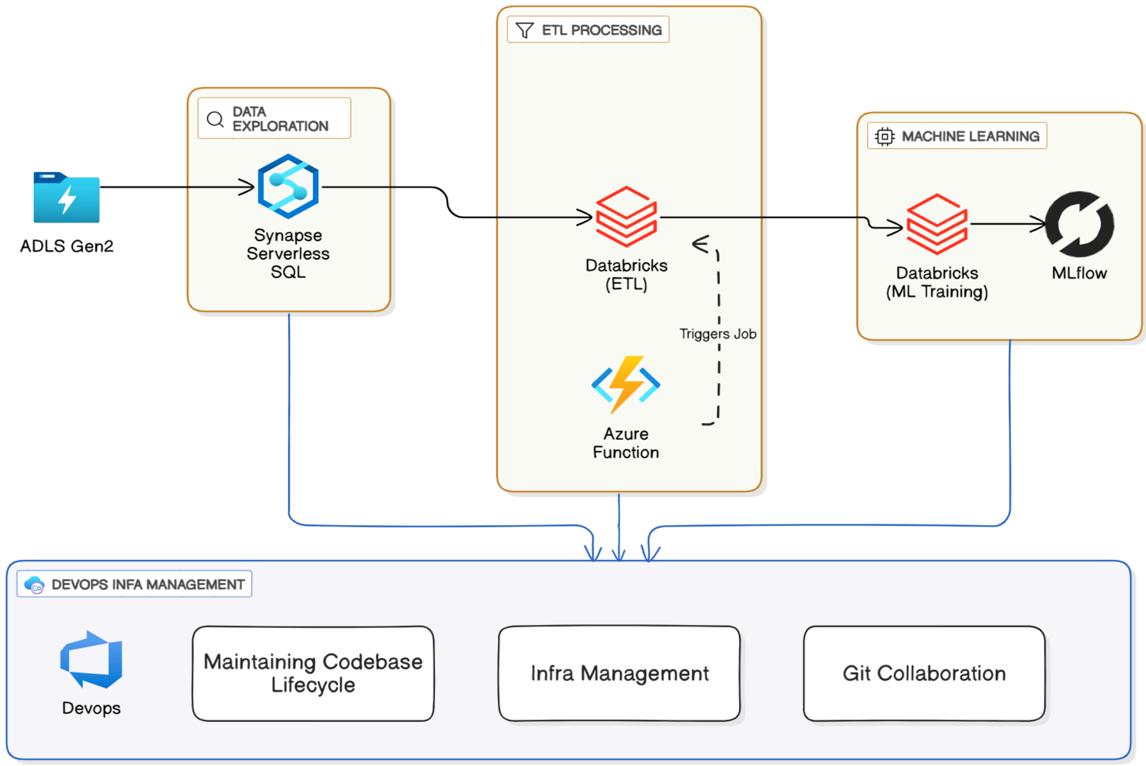
Azure DevOps

Azure DevOps plays a critical role in managing the entire project lifecycle. It is used for source code version control via Git repositories, managing infrastructure as code through ARM templates, and setting up CI/CD pipelines for automated deployment of ETL scripts, machine learning models, and Azure Functions.

Visual Studio Code (VS Code)

VS Code is the primary development environment used for building and deploying Azure Functions, managing Git repositories, and interacting with the Azure platform. Its integration with Azure and DevOps streamlines the development and deployment workflow.

Architecture Diagram



Methodology

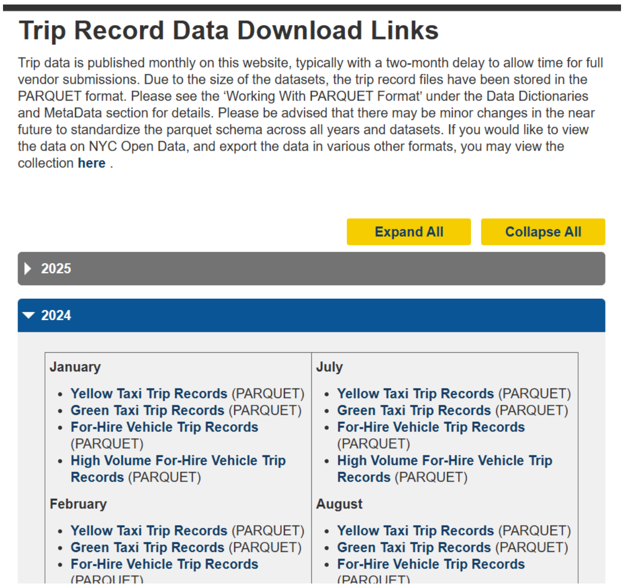
The methodology adopted in this project is based on a modular, serverless approach to data processing and machine learning using Microsoft Azure services. The workflow consists of sequential stages including data collection, exploration, transformation, model training, and deployment, all integrated via automated triggers and DevOps pipelines.

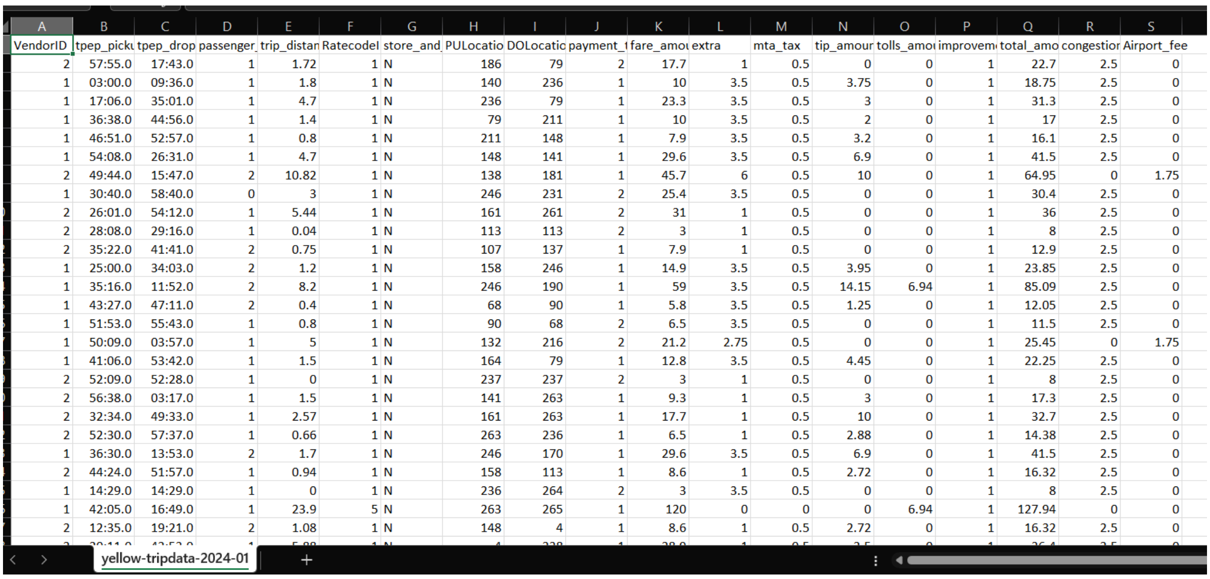
4.1 Data Collection and Storage

The data used in this project was sourced from the official **New York City Taxi & Limousine Commission (TLC)** data portal, accessible at:
https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page

The dataset selected for this project is:
yellow-tripdata-2024-01.csv, which contains detailed trip-level information about yellow taxi rides in New York City for the month of January 2024. This

includes fields such as pickup and drop-off timestamps, trip distance, fare amount, passenger count, payment type, and more.





VendorID	tpep_picku	tpep_drop	passenger_	trip_distan	RatecodeI	store_and_	PULocatio	DOLocatio	payment_t	fare_amou	extra	mta_tax	tip_amour	tolls_amou	improveme	total_amo	congestior	Airport_fee
2	57:55.0	17:43.0	1	1.72	1	N	186	79	2	17.7	1	0.5	0	0	1	22.7	2.5	0
1	03:00.0	09:36.0	1	1.8	1	N	140	236	1	10	3.5	0.5	3.75	0	1	18.75	2.5	0
1	17:06.0	35:01.0	1	4.7	1	N	236	79	1	23.3	3.5	0.5	3	0	1	31.3	2.5	0
1	36:38.0	44:56.0	1	1.4	1	N	79	211	1	10	3.5	0.5	2	0	1	17	2.5	0
1	46:51.0	52:57.0	1	0.8	1	N	211	148	1	7.9	3.5	0.5	3.2	0	1	16.1	2.5	0
1	54:08.0	26:31.0	1	4.7	1	N	148	141	1	29.6	3.5	0.5	6.9	0	1	41.5	2.5	0
2	49:44.0	15:47.0	2	10.82	1	N	138	181	1	45.7	6	0.5	10	0	1	64.95	0	1.75
1	30:40.0	58:40.0	0	3	1	N	246	231	2	25.4	3.5	0.5	0	0	1	30.4	2.5	0
2	26:01.0	54:12.0	1	5.44	1	N	161	261	2	31	1	0.5	0	0	1	36	2.5	0
2	28:08.0	29:16.0	1	0.04	1	N	113	113	2	3	1	0.5	0	0	1	8	2.5	0
2	35:22.0	41:41.0	2	0.75	1	N	107	137	1	7.9	1	0.5	0	0	1	12.9	2.5	0
1	25:00.0	34:03.0	2	1.2	1	N	158	246	1	14.9	3.5	0.5	3.95	0	1	23.85	2.5	0
1	35:16.0	11:52.0	2	8.2	1	N	246	190	1	59	3.5	0.5	14.15	6.94	1	85.09	2.5	0
1	43:27.0	47:11.0	2	0.4	1	N	68	90	1	5.8	3.5	0.5	1.25	0	1	12.05	2.5	0
1	51:53.0	55:43.0	1	0.8	1	N	90	68	2	6.5	3.5	0.5	0	0	1	11.5	2.5	0
1	50:09.0	03:57.0	1	5	1	N	132	216	2	21.2	2.75	0.5	0	0	1	25.45	0	1.75
1	41:06.0	53:42.0	1	1.5	1	N	164	79	1	12.8	3.5	0.5	4.45	0	1	22.25	2.5	0
2	52:09.0	52:28.0	1	0	1	N	237	237	2	3	1	0.5	0	0	1	8	2.5	0
2	56:38.0	03:17.0	1	1.5	1	N	141	263	1	9.3	1	0.5	3	0	1	17.3	2.5	0
2	32:34.0	49:33.0	1	2.57	1	N	161	263	1	17.7	1	0.5	10	0	1	32.7	2.5	0
2	52:30.0	57:37.0	1	0.66	1	N	263	236	1	6.5	1	0.5	2.88	0	1	14.38	2.5	0
1	36:30.0	13:53.0	2	1.7	1	N	246	170	1	29.6	3.5	0.5	6.9	0	1	41.5	2.5	0
2	44:24.0	51:57.0	1	0.94	1	N	158	113	1	8.6	1	0.5	2.72	0	1	16.32	2.5	0
1	14:29.0	14:29.0	1	0	1	N	236	264	2	3	3.5	0.5	0	0	1	8	2.5	0
1	42:05.0	16:49.0	1	23.9	5	N	263	265	1	120	0	0	0	6.94	1	127.94	0	0
2	12:35.0	19:21.0	2	1.08	1	N	148	4	1	8.6	1	0.5	2.72	0	1	16.32	2.5	0

yellow-tripdata-2024-01

Dataset Description:

Field Name	Description
VendorID	A code indicating the TPEP provider that provided the record. 1 = Creative Mobile Technologies, LLC 2 = Curb Mobility, LLC 6 = Myle Technologies Inc 7 = Helix
tpep_pickup_datetime	The date and time when the meter was engaged.
tpep_dropoff_datetime	The date and time when the meter was disengaged.
passenger_count	The number of passengers in the vehicle.
trip_distance	The elapsed trip distance in miles reported by the taximeter.
RatecodeID	The final rate code in effect at the end of the trip. 1 = Standard rate 2 = JFK 3 = Newark 4 = Nassau or Westchester 5 = Negotiated fare 6 = Group ride 99 = Null/unknown
store_and_fwd_flag	This flag indicates whether the trip record was held in vehicle memory before sending to the vendor, aka "store and forward," because the vehicle did not have a connection to the server. Y = store and forward trip N = not a store and forward trip
PULocationID	TLC Taxi Zone in which the taximeter was engaged.
DOLocationID	TLC Taxi Zone in which the taximeter was disengaged.
payment_type	A numeric code signifying how the passenger paid for the trip. 0 = Flex Fare trip 1 = Credit card 2 = Cash 3 = No charge 4 = Dispute 5 = Unknown 6 = Voided trip
fare_amount	The time-and-distance fare calculated by the meter. For additional information on the following columns, see https://www.nyc.gov/site/tlc/passengers/taxi-fare.page
extra	Miscellaneous extras and surcharges.
mta_tax	Tax that is automatically triggered based on the metered rate in use.
tip_amount	Tip amount – This field is automatically populated for credit card tips. Cash tips are not included.
tolls_amount	Total amount of all tolls paid in trip.
improvement_surcharge	Improvement surcharge assessed trips at the flag drop. The improvement surcharge began being levied in 2015.
total_amount	The total amount charged to passengers. Does not include cash tips.
congestion_surcharge	Total amount collected in trip for NYS congestion surcharge.
airport_fee	For pick up only at LaGuardia and John F. Kennedy Airports.
cbd_congestion_fee	Per-trip charge for MTA's Congestion Relief Zone starting Jan. 5, 2025.

4.2 Data Exploration

For data exploration, **Azure Synapse Analytics (Serverless SQL Pools)** was primarily used due to its ability to directly query files stored in **Azure Data**

Lake Storage Gen2 (ADLS Gen2) without requiring any data movement or pre-loading into databases. This serverless approach enabled rapid, cost-efficient querying of large datasets stored in the cloud.

The process began with the creation and deployment of an **Azure Synapse Analytics workspace**. Once the workspace was provisioned, a **linked service connection** was established to the ADLS Gen2 account where the dataset yellow-tripdata-2024-01.csv was stored. The dataset was then made accessible within the Synapse workspace.

Task 1) Data exploration using synapse analytics

Using **Synapse Serverless SQL**, an initial query was executed to **fetch the first 100 rows** from the CSV file in the data lake. This helped confirm connectivity and data visibility, as well as verify the file's schema. The raw dataset fields such as pickup and drop-off datetime, passenger count, trip distance, and fare amount were reviewed for validity.

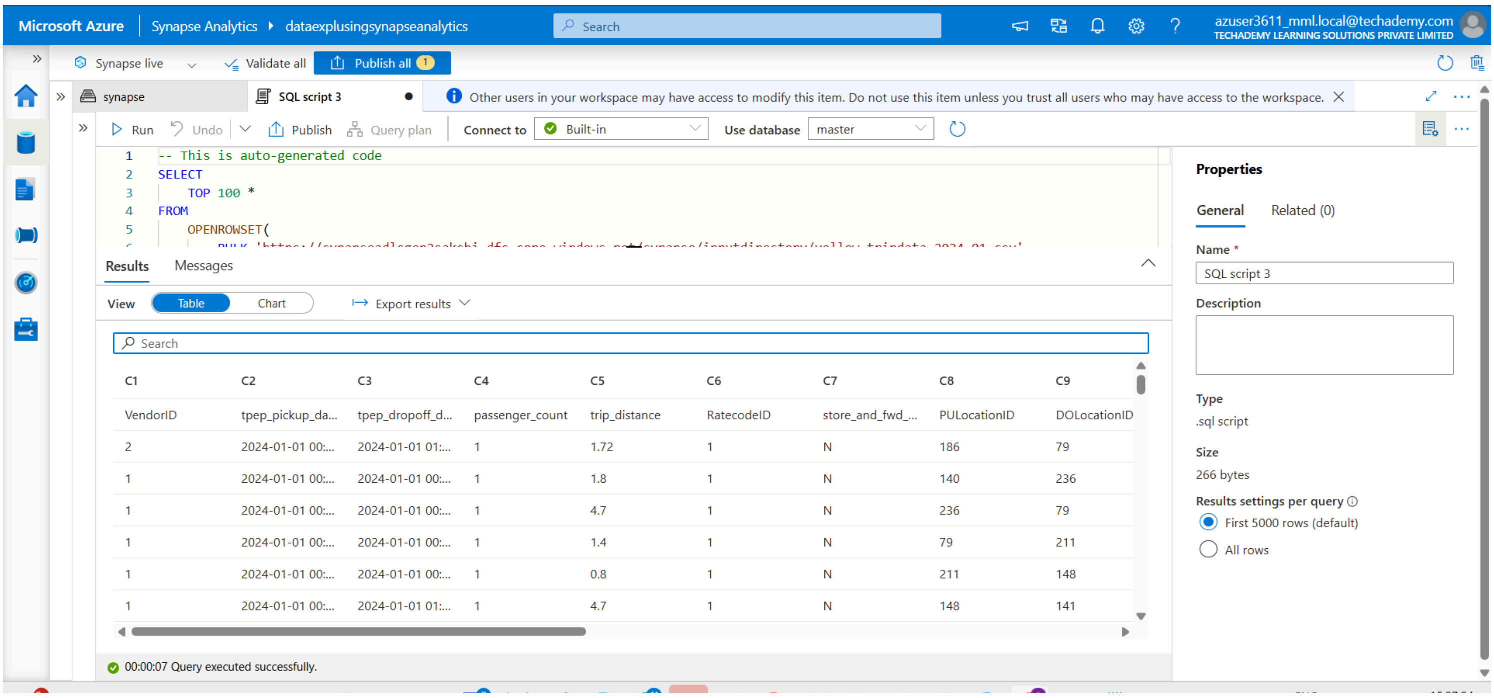
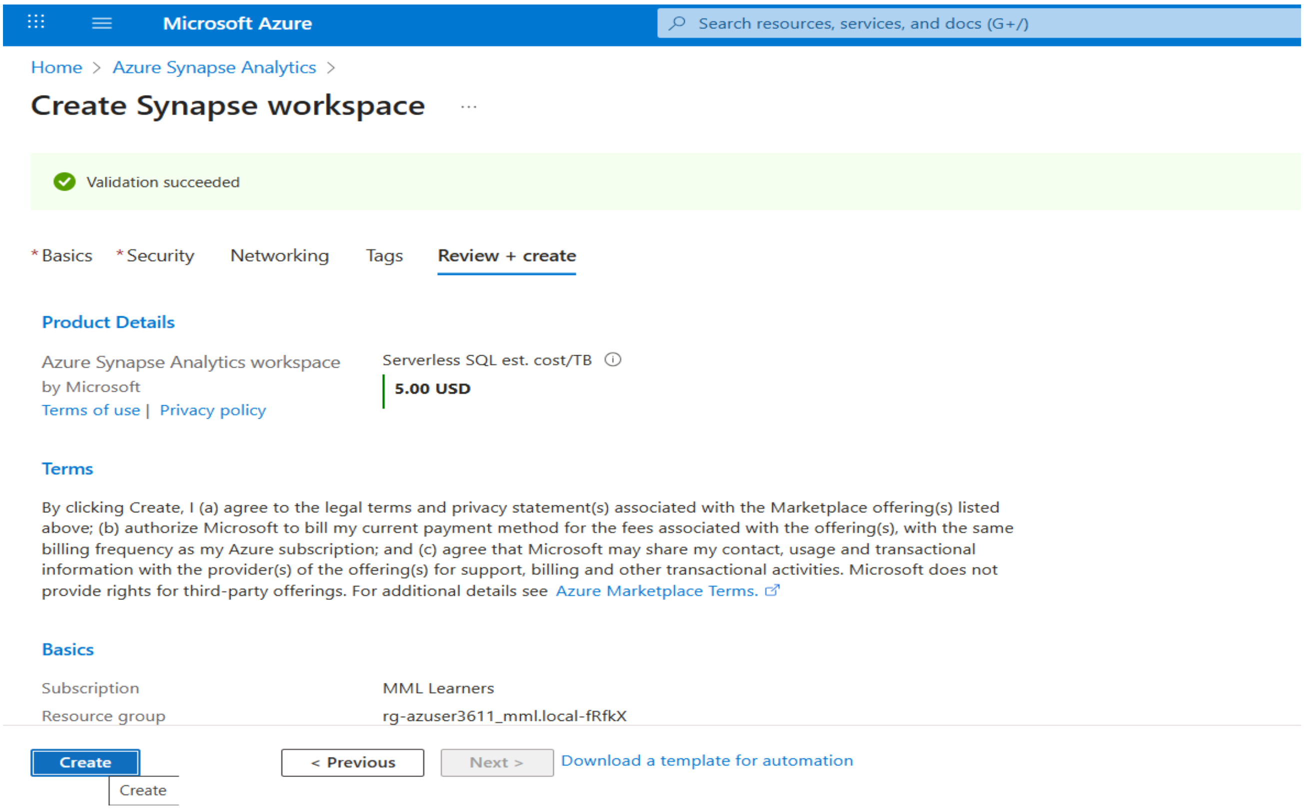
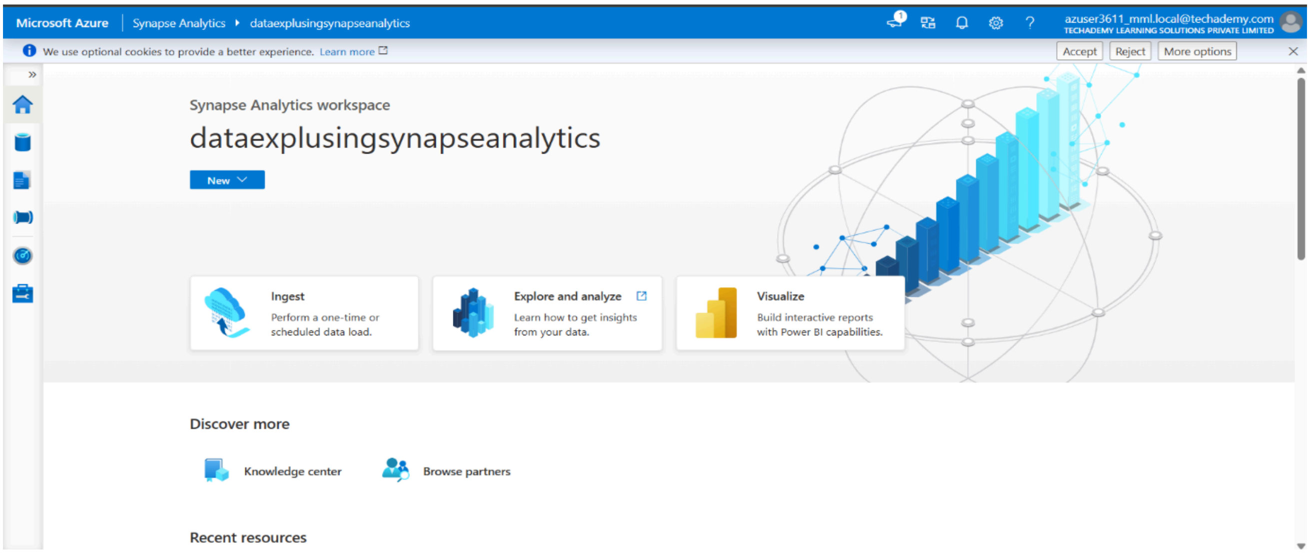
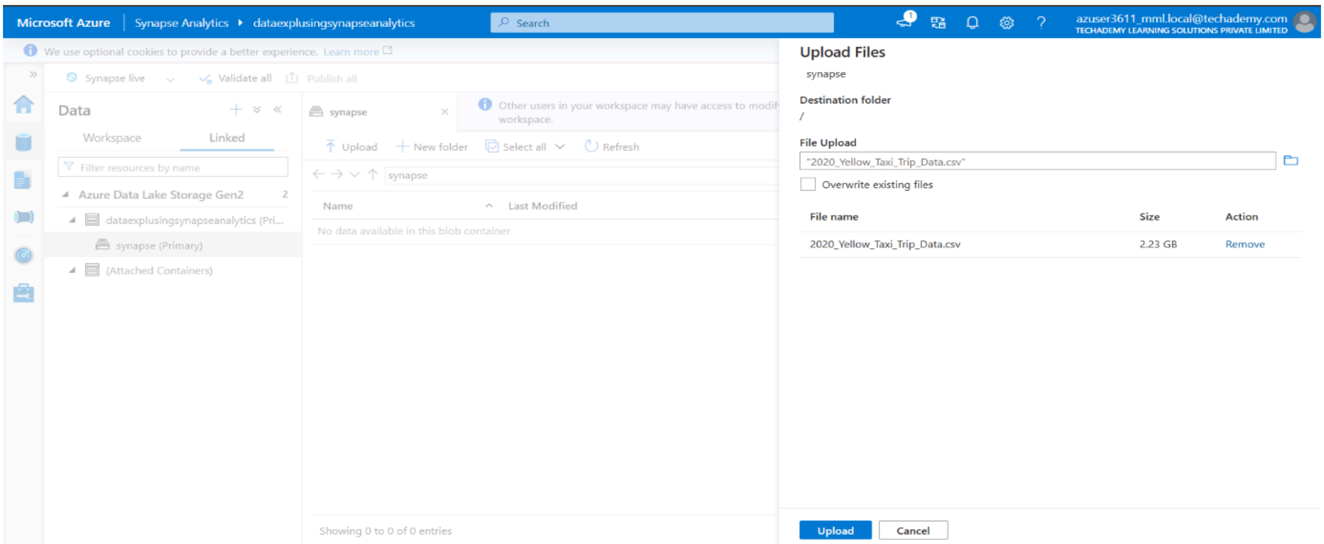
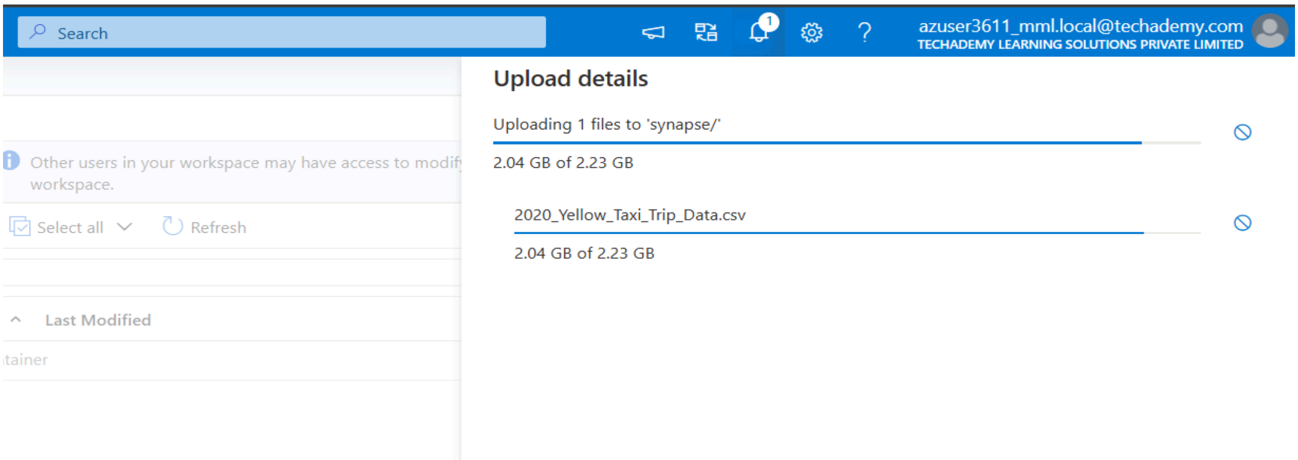


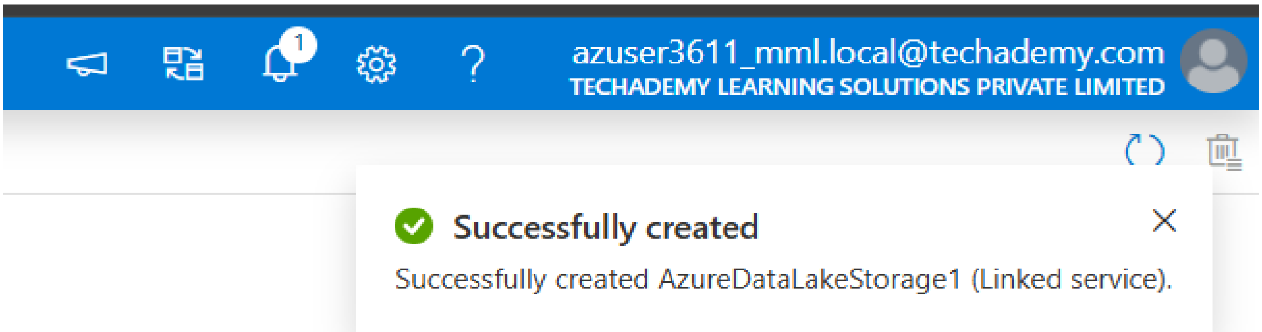
fig. a) **Fetching the first 100 rows using Synapse Serverless SQL from ADLS Gen2 dataset**


Microsoft Azure
Search resources, services, and docs (G+/)
Home > Azure Synapse Analytics >
Create Synapse workspace
Validation succeeded
*Basics *Security Networking Tags Review + create
Product Details
Azure Synapse Analytics workspace
by Microsoft
Terms of use | Privacy policy
Serverless SQL est. cost/TB
5.00 USD
Terms
By clicking Create, I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with the offering(s), with the same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. For additional details see Azure Marketplace Terms.
Basics
Subscription
MML Learners
Resource group
rg-azuser3611_mml.local-fRfkX
Create
Create
< Previous
Next >
Download a template for automation


Microsoft Azure
Synapse Analytics
dataexplusingsynapseanalytics
azuser3611_mml.local@techademy.com
TECHADEMY LEARNING SOLUTIONS PRIVATE LIMITED
We use optional cookies to provide a better experience. Learn more
Accept
Reject
More options
Synapse Analytics workspace
dataexplusingsynapseanalytics
New
Ingest
Perform a one-time or scheduled data load.
Explore and analyze
Learn how to get insights from your data.
Visualize
Build interactive reports with Power BI capabilities.
Discover more
Knowledge center
Browse partners
Recent resources


Microsoft Azure
Synapse Analytics
dataexplusingsynapseanalytics
Search
azuser3611_mml.local@techademy.com
TECHADEMY LEARNING SOLUTIONS PRIVATE LIMITED
Synapse live
Validate all
Publish all
Data
Workspace
Linked
Filter resources by name
Azure Data Lake Storage Gen2
dataexplusingsynapseanalytics (Pri...
synapse (Primary)
(Attached Containers)
synapse
Upload
New folder
Select all
Refresh
Name
Last Modified
No data available in this blob container
Showing 0 to 0 of 0 entries
Upload Files
synapse
Destination folder
/
File Upload
"2020_Yellow_Taxi_Trip_Data.csv"
Overwrite existing files
File name
Size
Action
2020_Yellow_Taxi_Trip_Data.csv
2.23 GB
Remove
Upload
Cancel





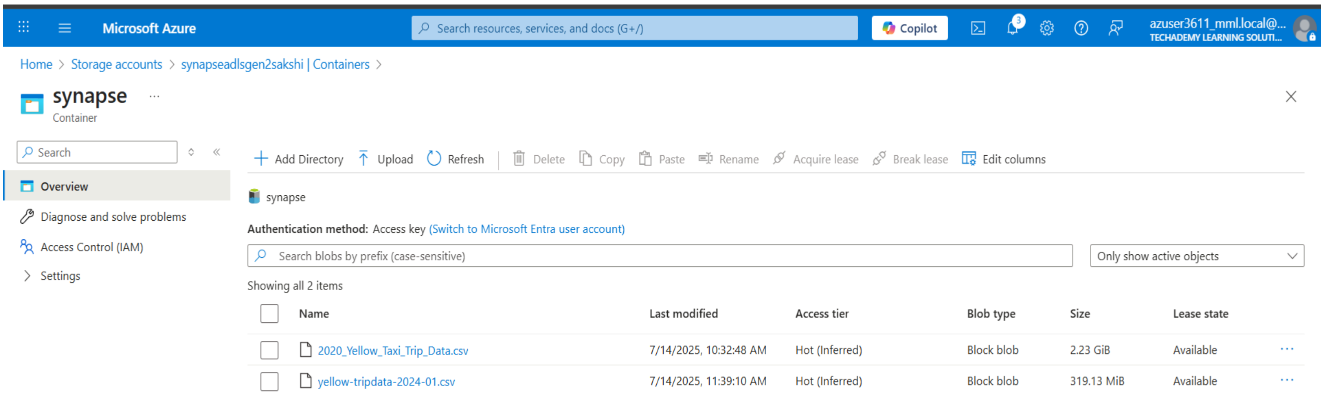


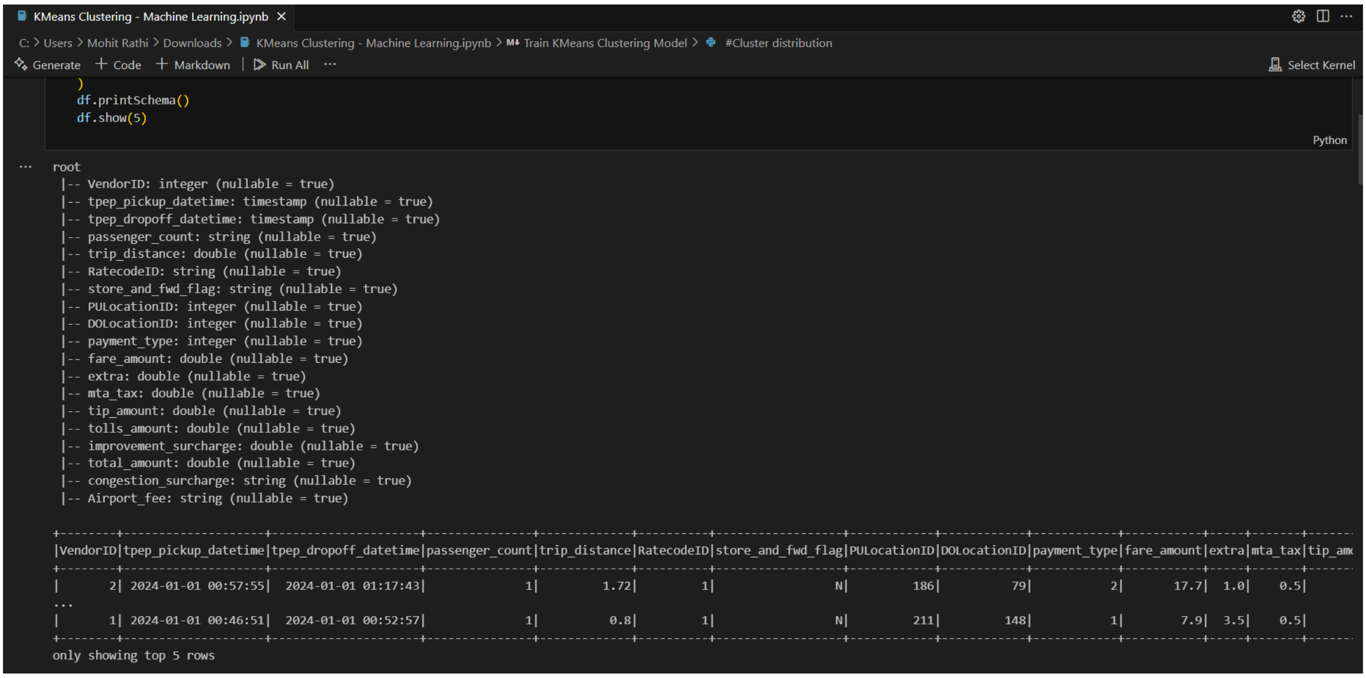
fig. b) ADLS Gen2 CSV file is accessible from Synapse Serverless

4.3 ETL Processing

The **ETL (Extract, Transform, Load)** process in this project was designed using a **serverless, automated architecture** that ensures modularity, scalability, and reusability. The core of the ETL workflow is built around the **Medallion Architecture** (also known as Bronze-Silver-Gold layers), which provides a systematic framework for incrementally refining raw data into structured and analytics-ready formats.

Task 2) PySpark ETL Operations

To achieve automation and scalability, a **Python-based Azure Function** was developed using **Visual Studio Code**. This function served as the serverless trigger to initiate **Azure Databricks Jobs**. Once executed, these jobs performed various Spark-based transformation operations on the raw data stored in **Azure Data Lake Storage Gen2 (ADLS Gen2)**.

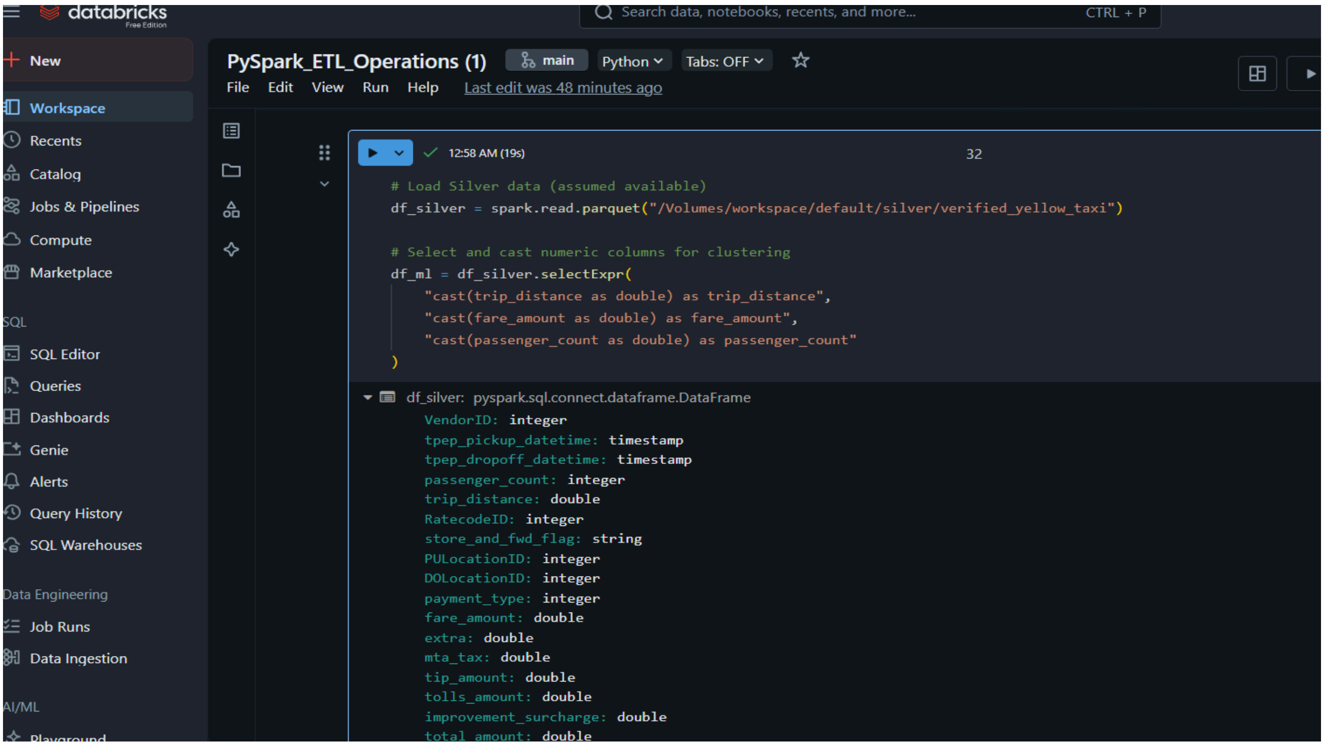


The ETL process followed the **Medallion Architecture** as outlined below:

- **Bronze Layer (Raw Ingestion)**:
  The raw CSV file (yellow-tripdata-2024-01.csv) stored in ADLS Gen2 was accessed and **mounted into Azure Databricks**. This allowed

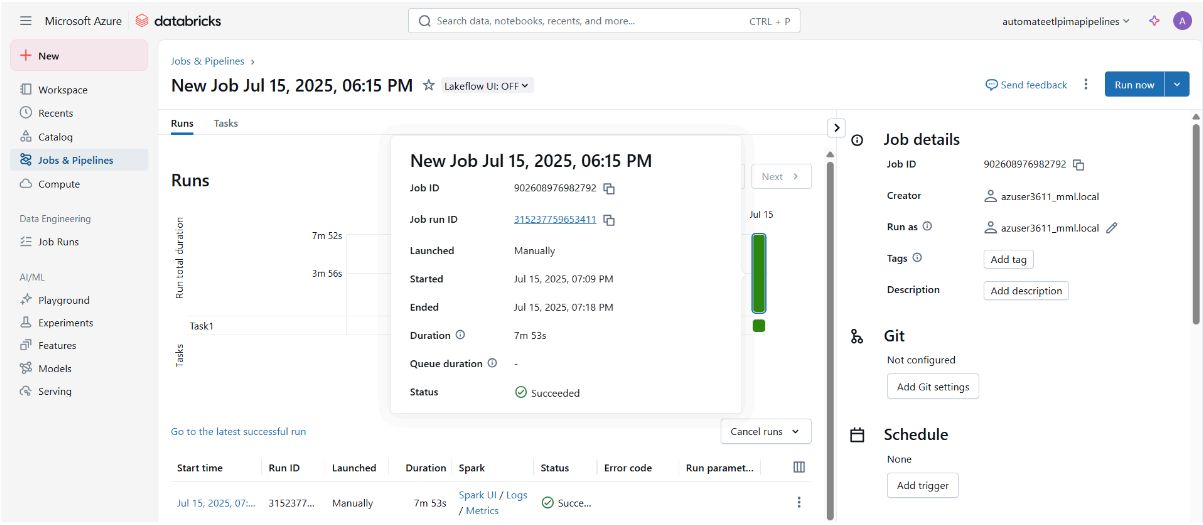
seamless interaction with the file system. The data was read in its original form, preserving schema and content, and loaded into a Delta Lake table as-is for auditability.

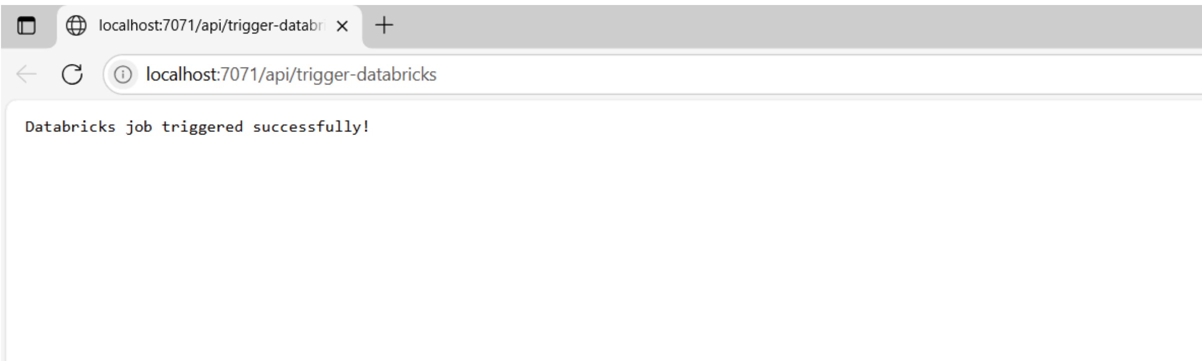
- **Silver Layer (Cleaned Data)**:
  Data cleaning and transformation tasks were performed in this layer. These included:
  - Handling missing or null values
  - Filtering out invalid records (e.g., zero fare with trip distance)
  - Converting data types and standardizing formats (e.g., timestamp parsing)
  - Generating derived fields such as trip duration or average speed
- **Gold Layer (Analytics-Ready)**:
  The final layer produced **curated, structured datasets** ready for downstream analytics and machine learning. Feature selection and formatting were completed, and the output was written back to ADLS Gen2 in **Parquet or Delta format** for optimized querying and storage.

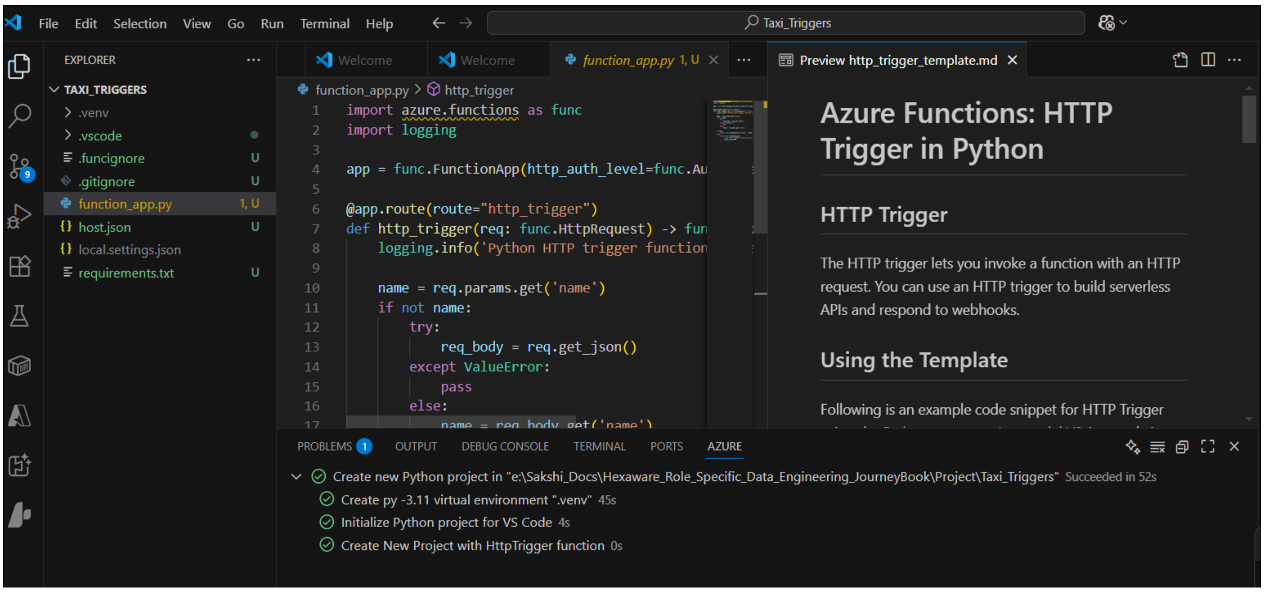


Task 3) Trigger Jobs using azure functions

The use of **Azure Databricks** with **Spark** allowed high-performance transformation at scale, while **Azure Functions** enabled event-driven execution without manual intervention. This serverless orchestration, combined with the Medallion Architecture, ensured a clear separation of raw, cleaned, and enriched data, making the pipeline robust, maintainable, and production-ready.







Task 4) Git Configuration - Azure DevOps & Synapse Analytics

To streamline version control and enable collaborative development, Git integration was configured using Azure DevOps Repos. This allowed us to maintain all Synapse SQL scripts, Spark notebooks, pipeline definitions, and ARM templates in a centralized and trackable Git repository. Developers could clone, commit, push, and pull changes directly from Synapse Studio using Git-connected workspaces.

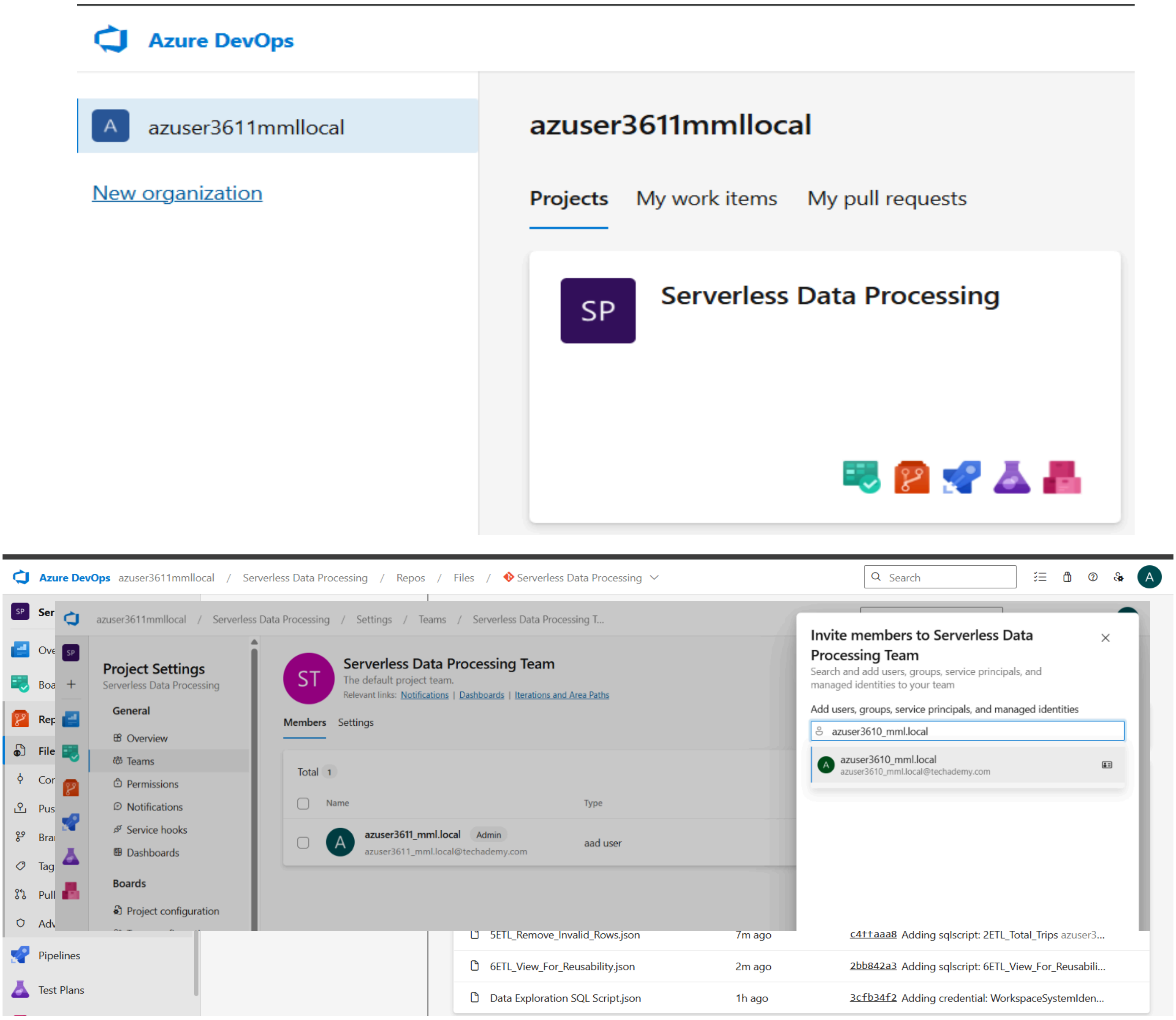


fig. c) Git collaboration with team

🔗**Complete project codebase at Azure DevOps:**

[DevOps Repository – Serverless Data Processing]

4.4 Machine Learning Pipeline

K-Means is a machine learning algorithm used to find groups (called clusters) in data. Each cluster groups similar rows (data points) together.

In this project, K-Means was applied to NYC Taxi Trip data using trip-related features like trip distance, fare amount, and passenger count. The goal was to segment taxi rides into meaningful clusters to identify travel patterns, high-value trips, or unusual behaviors. The final model was trained using scikit-learn inside mapInPandas() to ensure compatibility with Databricks Community Edition, and the results were tracked and evaluated using MLflow for transparency and reproducibility.

Example:

In NYC Taxi data, we might want to group trips like:

- Short trips with low fare
- Long trips with high fare
- Short trips with big tips
  These are called clusters.

Column	Meaning
trip_distance	**Distance of the taxi ride (in miles)**
fare_amount	**Fare for the ride (in $)**
passengercount	**No. of passengers during the trip**
prediction	**Which cluster the trip belongs to (0–3)**

Interpreting The Results

Cluster	Approx % of Total	Likely Pattern
0	~82%	Regular trips — maybe 1–2 miles, 1 passenger, normal fare. Most common.
1	~10%	Possibly longer trips or group rides.
3	~7%	Could be airport trips or higher fare rides.

KMeans Clustering - Machine Learning.ipynb ×

C: > Users > Mohit Rathi > Downloads > KMeans Clustering - Machine Learning.ipynb > M↓ Train KMeans Clustering Model > #Cluster distribution

Generate + Code + Markdown | Run All ···

```python
#Cluster distribution
from pyspark.sql.functions import col

cluster_counts = predictions.groupBy("prediction").count().orderBy("prediction")
display(cluster_counts)
```

prediction	count
0	451364
1	56147
3	36224

Databricks visualization. Run in Databricks to view.

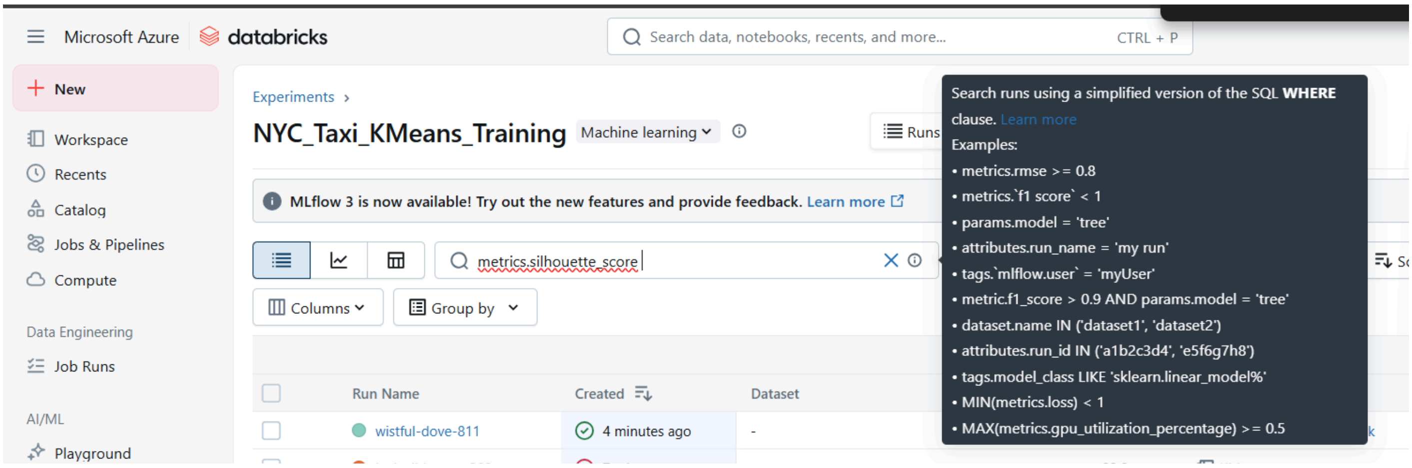
```python
#Cluster-wise Averages
cluster_avg = predictions.groupBy("prediction").avg("trip_distance", "fare_amount", "passenger_count").orderBy("prediction")
display(cluster_avg)
```

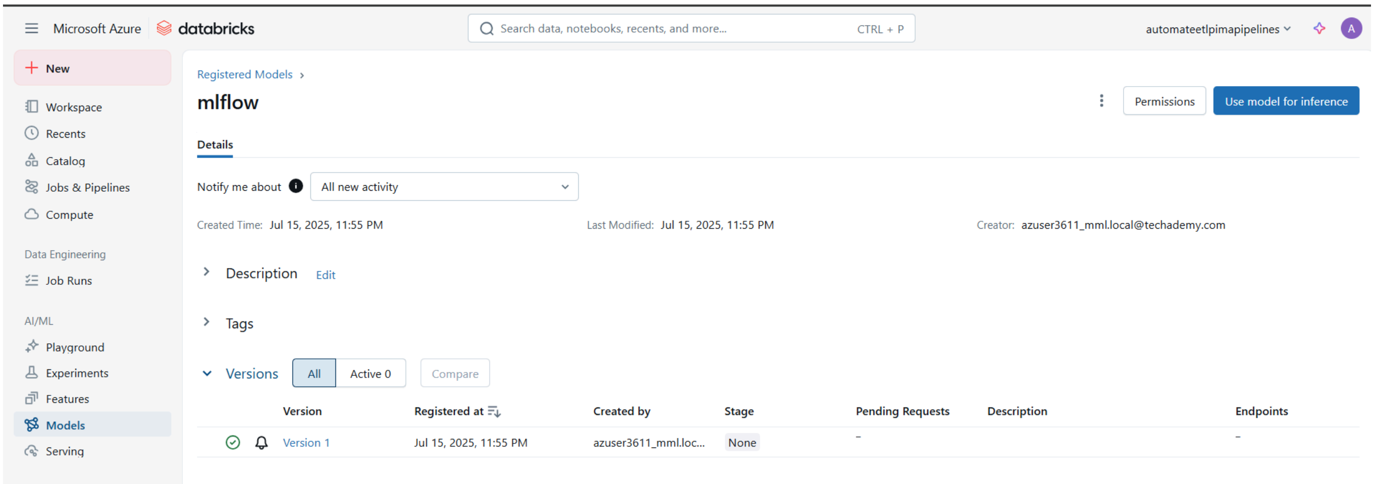
prediction	avg(trip_distance)	avg(fare_amount)	avg(passenger_count)
0	1.9954654558183649	13.466840775959088	1.1497106548151825
1	14.291601866528948	60.30749425614899	1.3167221757173135
3	2.5960435070671393	15.994524624558299	3.9974602473498235

Interrupt 00:17 21

```python
import mlflow
import mlflow.spark

with mlflow.start_run():
    k = 4
    kmeans = KMeans(featuresCol="features", predictionCol="prediction", k=k, seed=42)
    model = kmeans.fit(train_data)

    predictions = model.transform(test_data)
    silhouette = evaluator.evaluate(predictions)

    mlflow.log_param("k", k)
    mlflow.log_metric("silhouette_score", silhouette)

    # Log the trained model
    mlflow.spark.log_model(model, "kmeans_model")
```

(24) Spark Jobs
- Job 69 View (1 stage)
- Job 70 View (2 stages)
- Job 71 View (1 stage)
- Job 72 View (1 stage)
- Job 73 View (1 stage)
- Job 74 View (1 stage)
- Job 75 View (1 stage)
- Job 76 View (1 stage)
- Job 77 View (2 stages)
- Job 78 View (2 stages)


Microsoft Azure
databricks
Search data, notebooks, recents, and more...
CTRL + P
New
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Data Engineering
Job Runs
AI/ML
Playground
Experiments
NYC_Taxi_KMeans_Training
Machine learning
Runs
MLflow 3 is now available! Try out the new features and provide feedback. Learn more
metrics.silhouette_score
Columns
Group by
Run Name
Created
Dataset
wistful-dove-811
4 minutes ago
-
Search runs using a simplified version of the SQL WHERE clause. Learn more
Examples:
• metrics.rmse >= 0.8
• metrics.`f1 score` < 1
• params.model = 'tree'
• attributes.run_name = 'my run'
• tags.`mlflow.user` = 'myUser'
• metric.f1_score > 0.9 AND params.model = 'tree'
• dataset.name IN ('dataset1', 'dataset2')
• attributes.run_id IN ('a1b2c3d4', 'e5f6g7h8')
• tags.model_class LIKE 'sklearn.linear_model%'
• MIN(metrics.loss) < 1
• MAX(metrics.gpu_utilization_percentage) >= 0.5


Microsoft Azure
databricks
Search data, notebooks, recents, and more...
CTRL + P
automateetlpimapipelines
New
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Data Engineering
Job Runs
AI/ML
Playground
Experiments
Features
Models
Serving
Registered Models
mlflow
Permissions
Use model for inference
Details
Notify me about
All new activity
Created Time: Jul 15, 2025, 11:55 PM
Last Modified: Jul 15, 2025, 11:55 PM
Creator: azuser3611_mml.local@techademy.com
Description
Edit
Tags
Versions
All
Active 0
Compare
Version
Registered at
Created by
Stage
Pending Requests
Description
Endpoints
Version 1
Jul 15, 2025, 11:55 PM
azuser3611_mml.loc...
None

Conclusion

This project successfully demonstrates the implementation of a **serverless, automated data processing and machine learning pipeline** using Microsoft Azure services. By leveraging Azure Synapse Serverless SQL, Databricks, Azure Functions, and DevOps tools, the entire data journey—from raw ingestion to machine learning deployment—was streamlined without managing any physical servers or long-running infrastructure.

The use of **Azure Functions** to trigger **Databricks jobs** enabled an event-driven ETL process that scaled efficiently with workload demands. **Azure DevOps** played a critical role in managing the codebase, automating infrastructure deployments, and ensuring reproducibility through CI/CD pipelines.

A **KMeans clustering model** was trained using **Spark MLlib** to segment customers based on patterns in the transformed dataset. The integration of **MLflow** allowed comprehensive tracking of model parameters, metrics, and versions, ensuring transparency and consistency.

Overall, the solution is scalable, modular, and cost-efficient, demonstrating the power of serverless technologies in modern data engineering and machine learning workflows. The project also lays a strong foundation for integrating real-time data streams, additional machine learning models, and advanced monitoring in the future.