Case Study: End-to-End Data Pipeline for COVID-19 Test Data Management

1. Data Storage in Azure Blob Storage

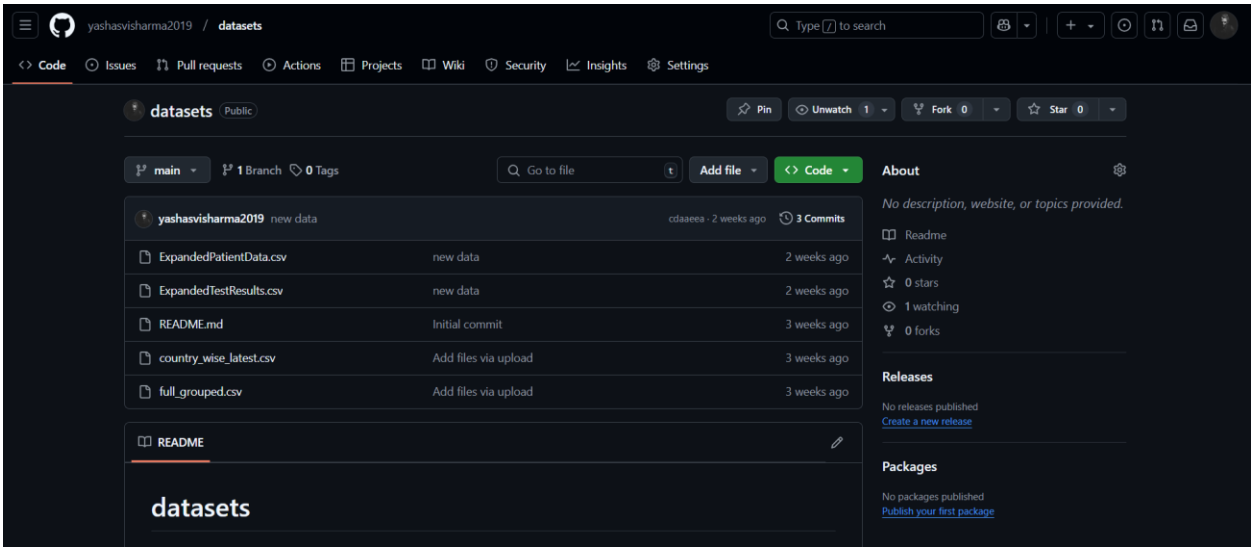
-**HTTP Locations:**

https://github.com/yashasvisharma2019/datasets

- **Source Data:**

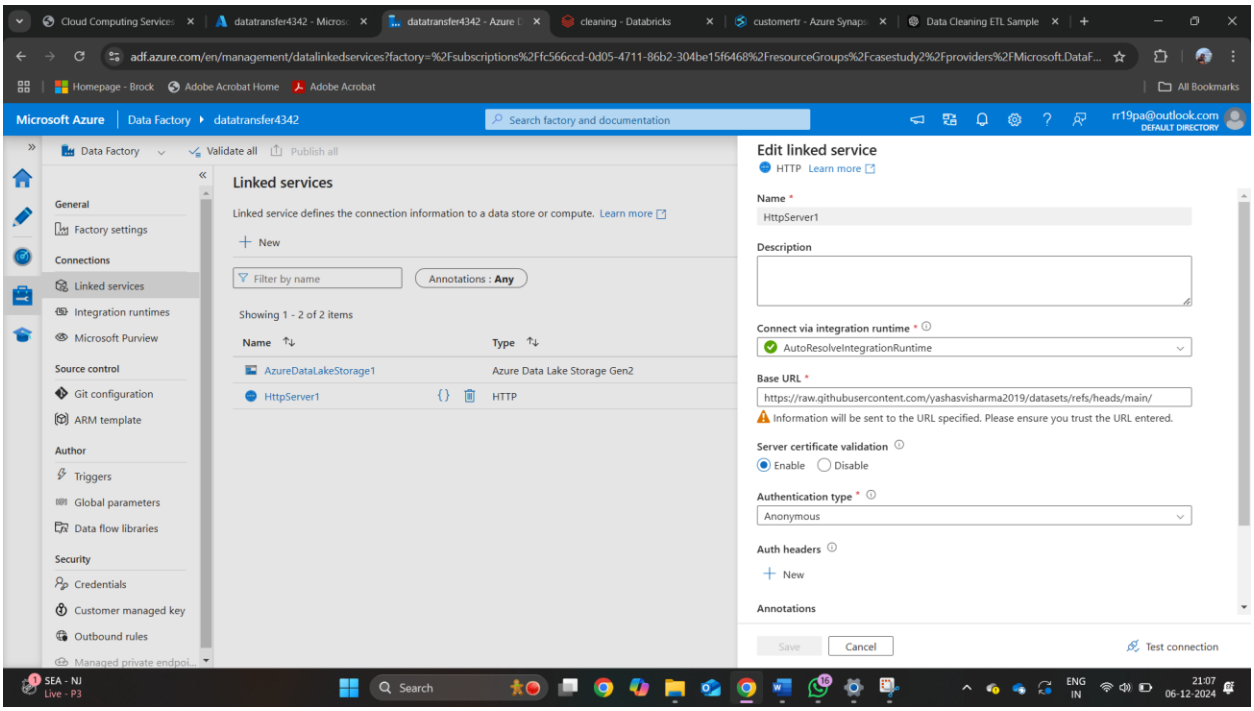
 - Patient data: `ExpandedPatientData.csv`

 - Test results: `ExpandedTestResults.csv`



- **Storage Location:**

 - Data was uploaded to an Azure Blob Storage container via HTTP.



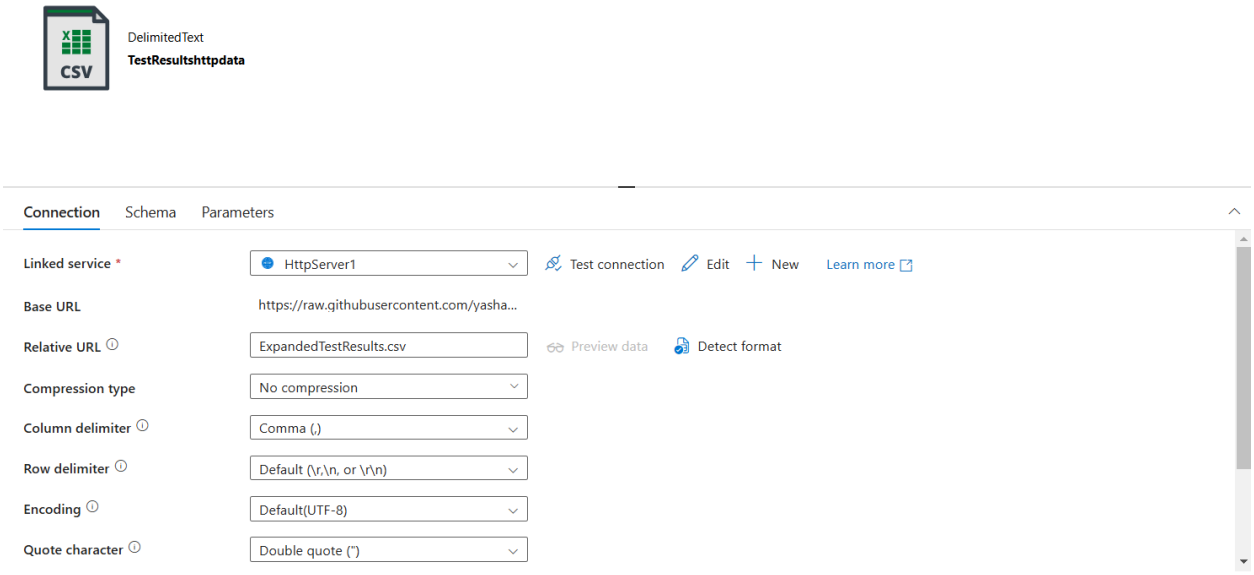
CSV

DelimitedText

Patientdatahttp

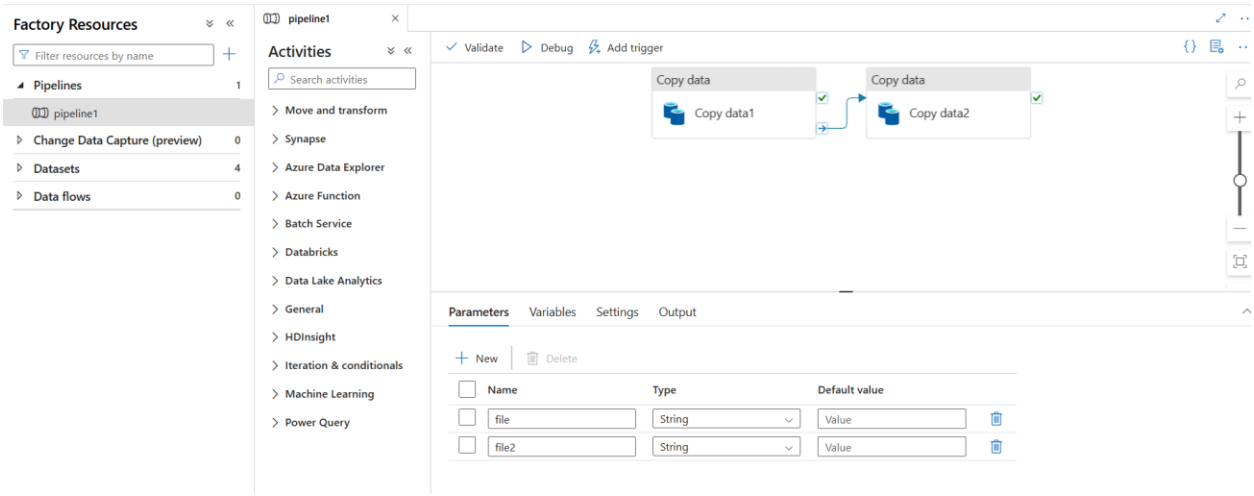
Connection | Schema | Parameters

Linked service *	HttpServer1	Test connection	Edit	New	Learn more
Base URL	https://raw.githubusercontent.com/yasha...				
Relative URL	ExpandedPatientData.csv	Preview data	Detect format		
Compression type	No compression				
Column delimiter	Comma (,)				
Row delimiter	Default (\r,\n, or \r\n)				
Encoding	Default(UTF-8)				
Quote character	Double quote (")				

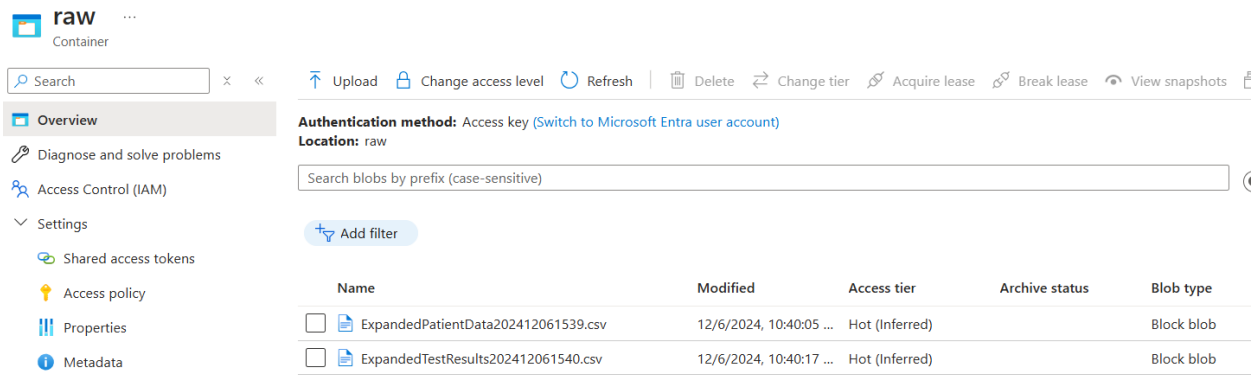


1. **Data Storage**:

   - Raw data was stored in Azure Blob Storage (HTTP endpoint).



   - File types included CSVs containing patient and test details.



2. **Data Cleaning and ETL**:

- Databricks was used to clean and transform the data using PySpark.

- Key cleaning steps:

  - Removing rows with missing identifiers (e.g., `PatientID`, `TestID`).

  - Standardizing data types (e.g., casting `Age` to integer, `TestDate` to date).

  - Handling null values by replacing them with defaults like 'Unknown'.

- ETL

- Joined patient and test data on `PatientID`.

- Added a derived column `ResultStatus` based on test results ('Positive' or 'Negative').

- Selected and renamed columns for analytics.

```python
patients_df = spark.read.format("delta").load(cleaned_patient_output_path)
tests_df = spark.read.format("delta").load(cleaned_test_output_path)

# Join the cleaned patient and test data on PatientID
joined_df = patients_df.join(tests_df, on="PatientID", how="inner")

# Add a new column to indicate if a patient's test result is 'Positive' or 'Negative'
joined_df = joined_df.withColumn("ResultStatus", when(col("COVIDStatus") == "Positive", lit("Positive"))
                                                 .otherwise(lit("Negative")))

# Select and transform relevant columns for reporting
etl_output_df = joined_df.select(
    col("PatientID"),
    col("Age"),
    col("COVIDStatus"),
    col("TestID"),
    col("LabName"),
    col("ResultDate"),
    col("ResultStatus")
)

# Show a preview of the transformed data
etl_output_df.show()
```

```python
# Save the transformed data to Delta format in Azure Storage
etl_output_path = "abfss://gold@httpforstorage.dfs.core.windows.net/ETLResultData"

etl_output_df.write.format("delta").mode("append").save(etl_output_path)
print(f"ETL output data saved to {etl_output_path}")
```

```
(7) Spark Jobs
etl_output_df: pyspark.sql.dataframe.DataFrame = [PatientID: integer, Age: integer ... 5 more fields]
joined_df: pyspark.sql.dataframe.DataFrame = [PatientID: integer, Name: string ... 11 more fields]
patients_df: pyspark.sql.dataframe.DataFrame = [PatientID: integer, Name: string ... 6 more fields]
tests_df: pyspark.sql.dataframe.DataFrame = [TestID: integer, PatientID: integer ... 3 more fields]
```

PatientID	Age	COVIDStatus	TestID	LabName	ResultDate	ResultStatus
1	20	Negative	1035	LabCorp	2024-06-11	Negative
2	33	Recovered	1049	PathCare	2024-11-07	Negative
3	42	Negative	1023	Quest Diagnostics	2024-10-08	Negative
4	61	Negative	1075	MedLife Labs	2024-08-15	Negative
5	34	Recovered	1078	Quest Diagnostics	2024-09-08	Negative
6	44	Negative	1063	PathCare	2024-08-04	Negative
7	65	Recovered	1066	LabCorp	2024-09-08	Negative
9	59	Positive	1095	LabCorp	2024-01-27	Positive
10	46	Positive	1058	Quest Diagnostics	2024-05-29	Positive
12	26	Negative	1094	MedLife Labs	2024-05-15	Negative
13	35	Recovered	1046	LabCorp	2024-11-17	Negative

3. **Data Loading**:

- Transformed data was saved in Delta format.

- Data was ingested into an Azure Synapse Analytics dedicated SQL pool for reporting and analytics

Error Handling

- Missing columns or unresolved variables were handled with validation steps and exceptions.

- Write operations to storage were encapsulated in try-except blocks to capture and log errors.

3. Data Loading to Synapse Analytics

- **Output Storage**: Transformed data was saved in Delta format in Azure Data Lake.

- **Synapse Pipeline**:

  - Data from Delta tables was ingested into an Azure Synapse Analytics dedicated SQL pool.

Results

- Improved data quality by resolving inconsistencies and handling missing values.

- Achieved a centralized, optimized SQL-based data store for reporting and analytics.

- Enabled real-time insights for healthcare decision-making.

Conclusion

This end-to-end pipeline effectively addressed the challenges of managing raw COVID-19 test data. The integration of Azure Blob Storage, Databricks, and Synapse Analytics streamlined the process from data ingestion to reporting. The scalable architecture supports future enhancements for additional datasets or analytics requirements.