

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

### ### 1. Data Storage in Azure Blob Storage

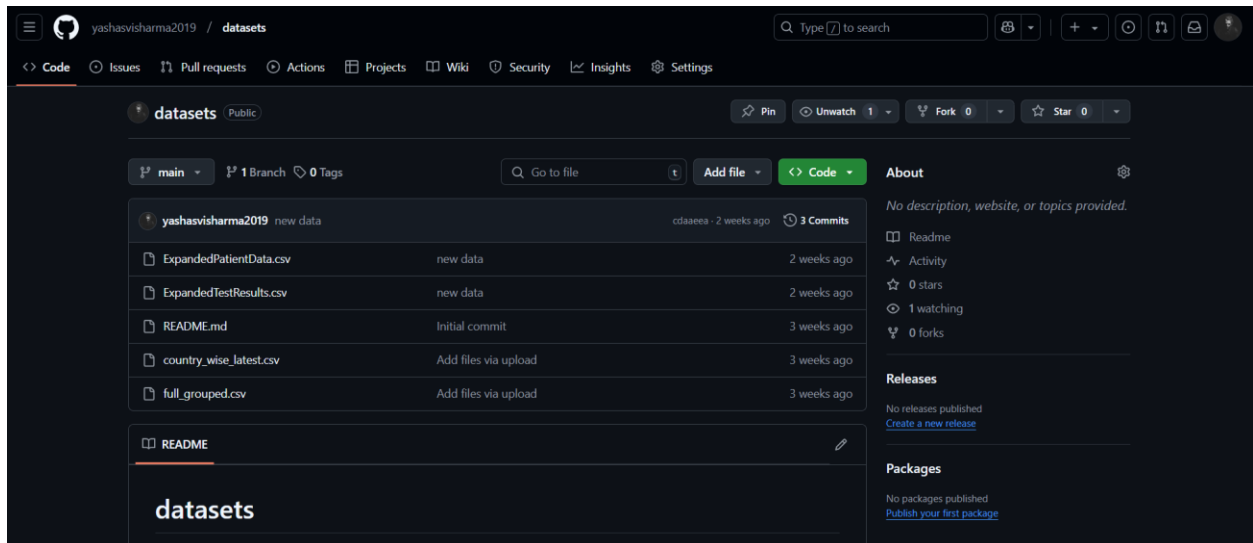
- \*\*HTTP Locations:\*\*

<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.

Cloud Computing Services | datatransfer4342 - Micro... | datatransfer4342 - Azure | cleaning - Databricks | customertr - Azure Synap... | Data Cleaning ETL Sample | +

adf.azure.com/en/management/datalinkedservices?factory=%2Fsubscriptions%2Ffc566ccd-0d05-4711-86b2-304be15f5468%2FresourceGroups%2Fcasestudy2%2Fproviders%2FMicrosoft.DataF...

Homepage - Brock | Adobe Acrobat Home | Adobe Acrobat

Microsoft Azure | Data Factory | datatransfer4342 | Search factory and documentation | rr19pe@outlook.com | DEFAULT DIRECTORY

» Data Factory | Validate all | Publish all

General | Factory settings | Connections | Linked services | Integration runtimes | Microsoft Purview | Source control | Git configuration | ARM template | Author | Triggers | Global parameters | Data flow libraries | Security | Credentials | Customer managed key | Outbound rules | Managed private endpoint...

Linked services

Linked service defines the connection information to a data store or compute. [Learn more](#)

+ New

Filter by name Annotations: Any

Showing 1 - 2 of 2 items

| Name                  | Type                         |
|-----------------------|------------------------------|
| AzureDataLakeStorage1 | Azure Data Lake Storage Gen2 |
| HttpServer1           | HTTP                         |

Edit linked service

HTTP [Learn more](#)

Name \* HttpServer1

Description

Connect via integration runtime \*   
 AutoResolveIntegrationRuntime

Base URL \*   
 https://raw.githubusercontent.com/yashavisharma2019/datasets/refs/heads/main/   
 Information will be sent to the URL specified. Please ensure you trust the URL entered.

Server certificate validation   
 Enable ☒ Disable ☐

Authentication type \*   
 Anonymous

Auth headers   
 + New

Annotations

Save Cancel Test connection



DelimitedText  
Patientdatahttp

Connection Schema Parameters

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



DelimitedText  
TestResultshttpdata

Connection Schema Parameters

Linked service \* HttpServer1 [Test connection](#) [Edit](#) [+ New](#) [Learn more](#)

Base URL https://raw.githubusercontent.com/yasha...

Relative URL ExpandedTestResults.csv [Preview data](#) [Detect format](#)

Compression type No compression

Column delimiter Comma (,)

Row delimiter Default (\r\n, or \n)

Encoding Default(UTF-8)

Quote character Double quote (")

## 1. \*\*Data Storage\*\*:

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

Factory Resources

Filter resources by name

Pipelines 1

pipeline1

Change Data Capture (preview) 0

Datasets 4

Data flows 0

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

pipeline1

Validate Debug Add trigger

Copy data

Copy data1

Copy data2

Parameters Variables Settings Output

+ New - Delete

| Name  | Type   | Default value |
|-------|--------|---------------|
| file  | String | Value         |
| file2 | String | Value         |

- File types included CSVs containing patient and test details.

raw

Container

×
«

↑ Upload
🔒 Change access level
🔄 Refresh
🗑 Delete
↔ Change tier
🔑 Acquire lease
🔑 Break lease
📷 View snapshots

Overview

🔧 Diagnose and solve problems
👤 Access Control (IAM)
⌵ Settings
🔑 Shared access tokens
🔑 Access policy
📊 Properties
📄 Metadata

Authentication method: Access key ([Switch to Microsoft Entra user account](#))

Location: raw

⌵

⚙ Add filter

| Name   | Modified                | Access tier    | Archive status | Blob type  |
|--|-------------------------|----------------|----------------|------------|
| <input type="checkbox"/> <a href="#">ExpandedPatientData202412061539.csv</a> | 12/6/2024, 10:40:05 ... | Hot (Inferred) |                | Block blob |
| <input type="checkbox"/> <a href="#">ExpandedTestResults202412061540.csv</a> | 12/6/2024, 10:40:17 ... | Hot (Inferred) |                | Block blob |

## 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'.

```

01:31 PM (22s) 2

mount_point = "/mnt/httpforstorage/raw"

if any(mount.mountPoint == mount_point for mount in dbutils.fs.mounts()):
    # Unmount the existing mount point
    dbutils.fs.unmount(mount_point)
    print(f"Unmounted existing mount at {mount_point}")

try:
    dbutils.fs.mount(
        source="abfss://raw@httpforstorage.dfs.core.windows.net/",
        mount_point=mount_point,
        extra_configs=configs
    )
    print(f"Mounted successfully at {mount_point}")
except Exception as e:
    print(f"Error mounting: {e}")

/mnt/httpforstorage/raw has been unmounted.
Unmounted existing mount at /mnt/httpforstorage/raw
Mounted successfully at /mnt/httpforstorage/raw

```

```
▶ 01:31 PM (10s) 6 Python ✨

# Databricks Data Cleaning for COVID-19 Data
# Import required libraries
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when, lit, countDistinct
from pyspark.sql.types import StringType, IntegerType, DateType

# Initialize Spark session
spark = SparkSession.builder.appName("COVIDDataCleaning").getOrCreate()

# Load Patient Data
patient_file_path = "/mnt/httpforstorage/raw/ExpandedPatientData202412061539.csv" # Update path if needed
patients_df = spark.read.format("csv").option("header", "true").load(patient_file_path)

# Load Test Data
test_file_path = "/mnt/httpforstorage/raw/ExpandedTestResults202412061540.csv" # Update path if needed
tests_df = spark.read.format("csv").option("header", "true").load(test_file_path)

# --- Data Cleaning on Patients Data ---
# Drop rows with missing PatientID
patients_df = patients_df.filter(col("PatientID").isNotNull())

# Cast PatientID and Age to Integer and TestDate to Date
def clean_patient_data(df):
    return (df.withColumn("PatientID", col("PatientID").cast(IntegerType()))
            .withColumn("Age", col("Age").cast(IntegerType()))
            .withColumn("TestDate", col("TestDate").cast(DateType())))

tests_df = clean_test_data(tests_df)

tests_df = tests_df.withColumn("LabName", when(col("LabName").isNull(), lit("Unknown"))
                                .otherwise(col("LabName")))

patients_df.select(countDistinct("PatientID").alias("Unique Patients")).show()
tests_df.select(countDistinct("TestID").alias("Unique Tests")).show()

# Save cleaned data as Parquet files and use Delta Lake for appending
cleaned_patient_file_path = "/mnt/httpforstorage/processed/CleanedPatientData"
cleaned_test_file_path = "/mnt/httpforstorage/processed/CleanedTestResults"
cleaned_patient_output_path = "abfss://processed@httpforstorage.dfs.core.windows.net/CleanedPatientData"
cleaned_test_output_path = "abfss://processed@httpforstorage.dfs.core.windows.net/CleanedTestResults"

patients_df.write.format("delta").mode("append").save(cleaned_patient_file_path)
tests_df.write.format("delta").mode("append").save(cleaned_test_file_path)
patients_df.write.format("delta").mode("append").save(cleaned_patient_output_path)
print(f"Cleaned patient data saved to {cleaned_patient_output_path}")

tests_df.write.format("delta").mode("append").save(cleaned_test_output_path)
print(f"Cleaned test data saved to {cleaned_test_output_path}")

print(f"Cleaned data saved to {cleaned_patient_file_path} and {cleaned_test_file_path} using Delta Lake")
tests_df.show()
patients_df.show()
```

- 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.

## (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     |

- 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.