Smart Manufacturing Data Analytics Platform – End-to-End DevOps & DataOps

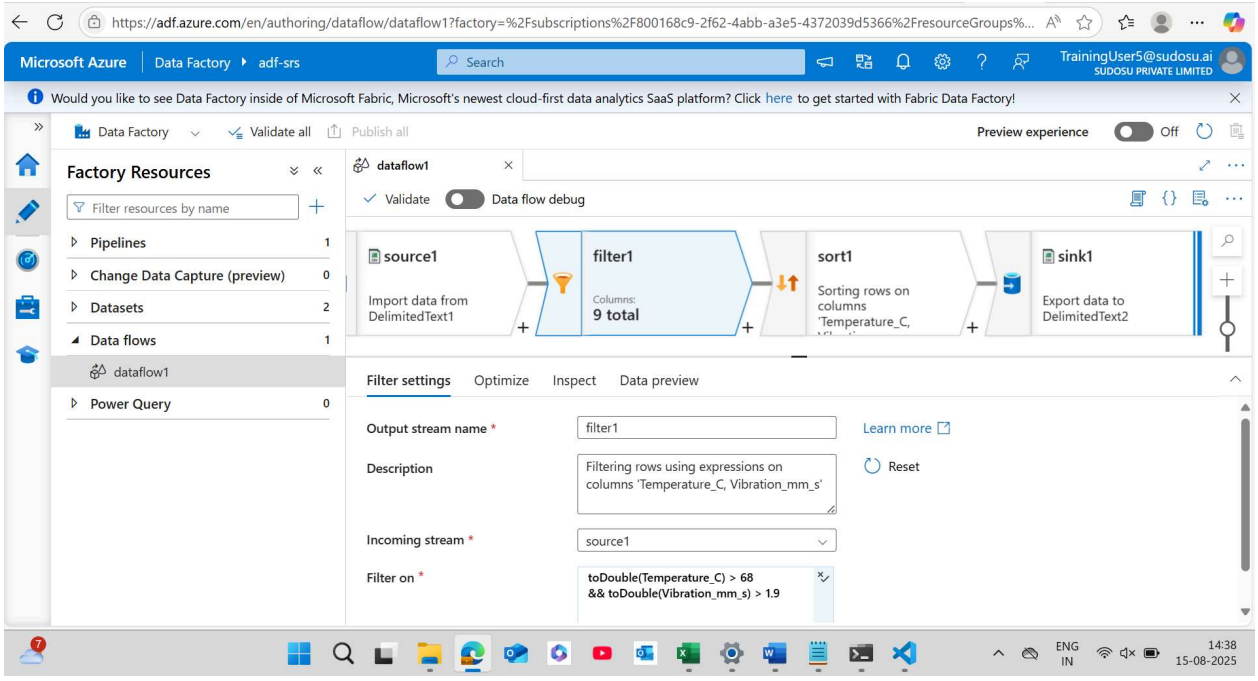
1. Problem Statement

AutoForge Ltd., a global automotive manufacturer, struggles with:

- Delayed analytics due to siloed legacy systems.
- Frequent deployment errors causing production data delays.
- Lack of real-time monitoring for IoT-based predictive maintenance.
- Inefficient model retraining due to absence of drift monitoring.
- Manual, error-prone deployment processes without rollback capability.

Creating Data Flow

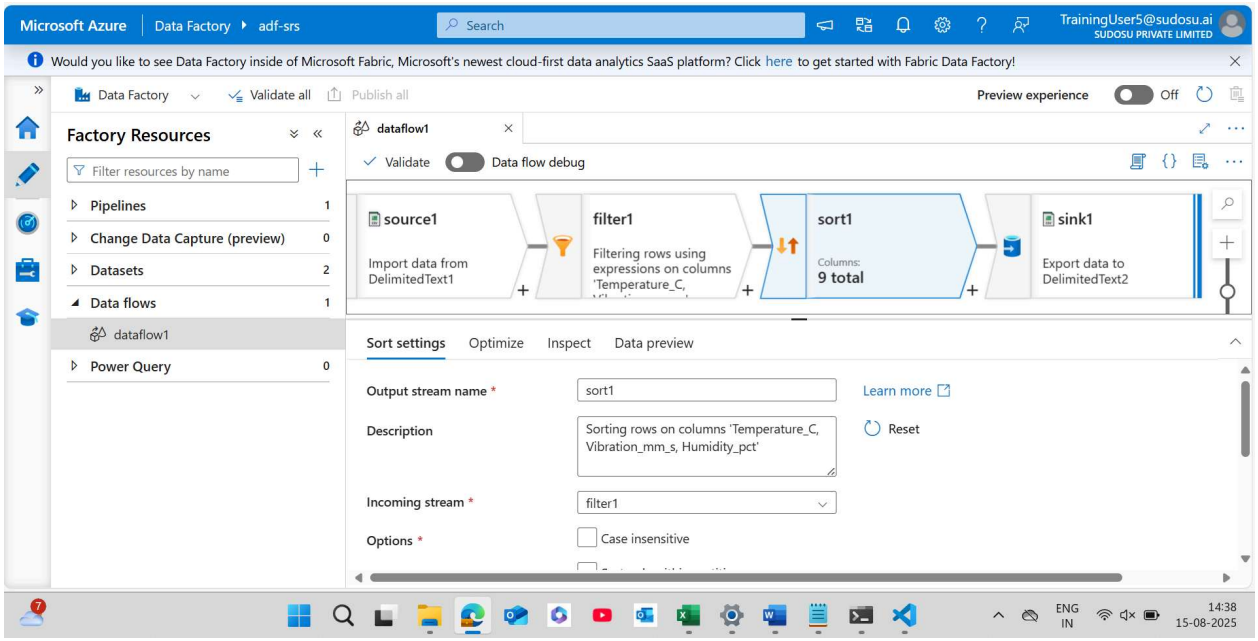
Source (CSV in Blob) → Sort → Filter → Sink (CSV/Parquet in Blob)



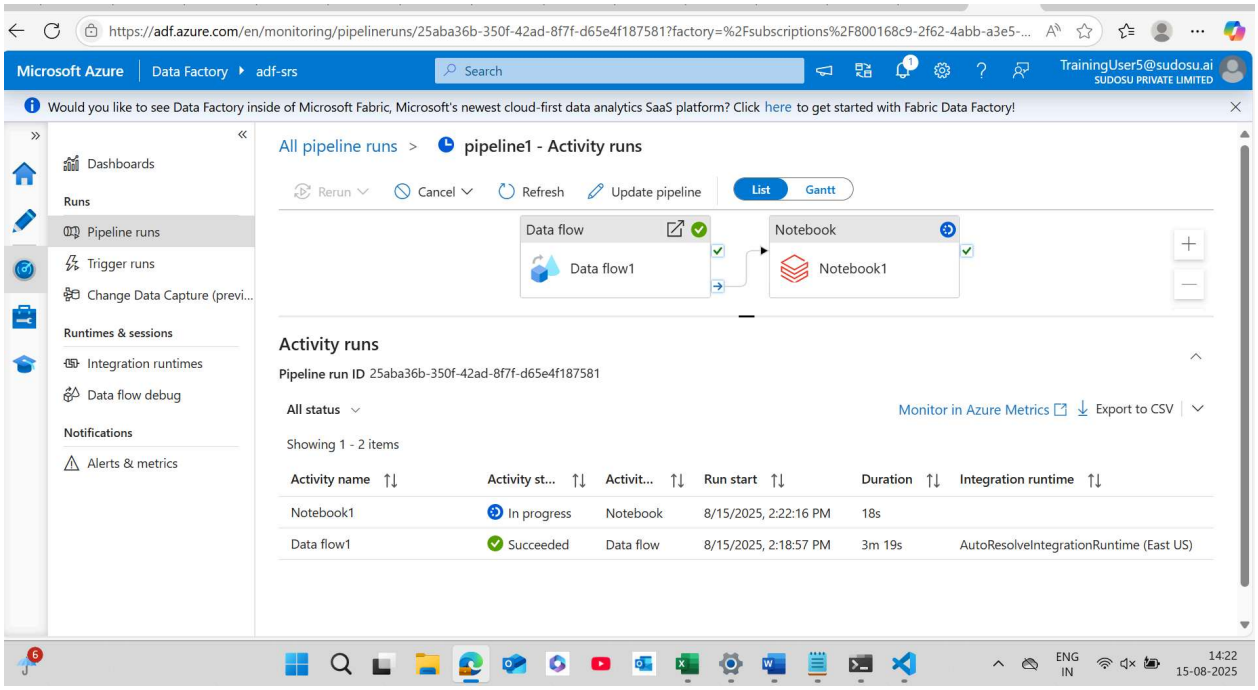
In theory, a data flow is the logical movement of data from a source system, through one or more transformation steps, to a target (sink) system.

In your case, the data flows from the source dataset in Blob Storage, undergoes sorting and filtering transformations to arrange and refine the records, and is then written to the sink location for further use.

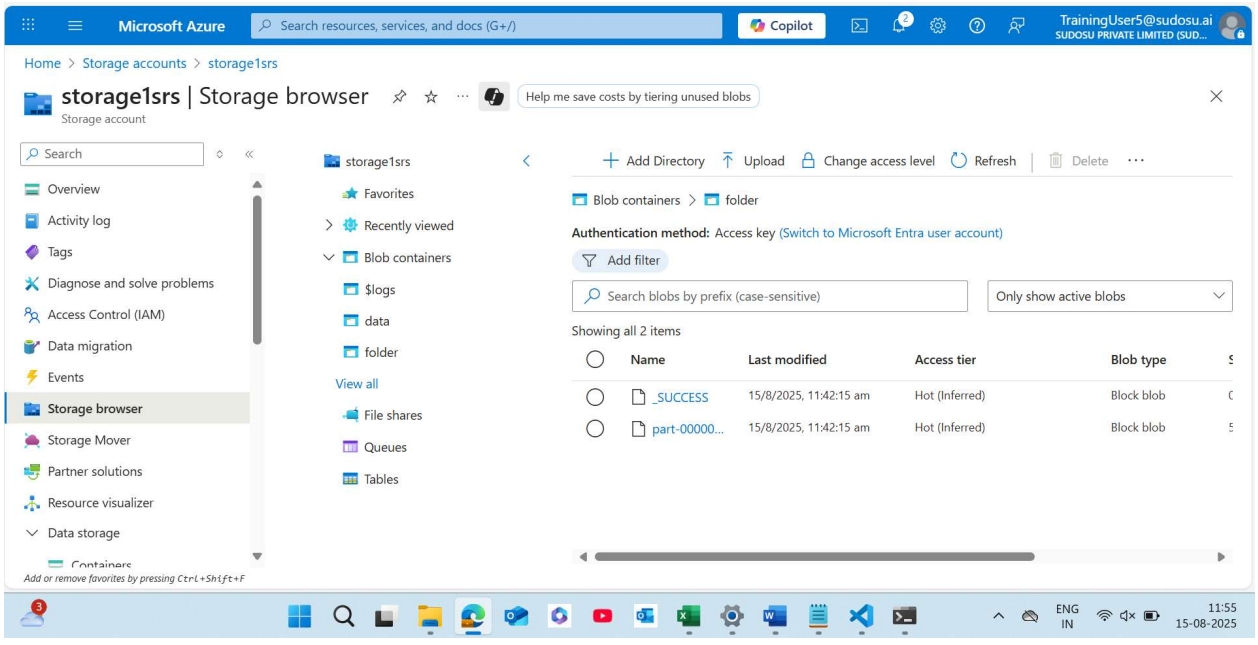
- Sort: You're arranging rows based on one or more columns.(Ascending)
- Filter: You're keeping only rows that match a condition (for example, `Temperature_C > 70 AND Vibration_mm_s > 2.0`).

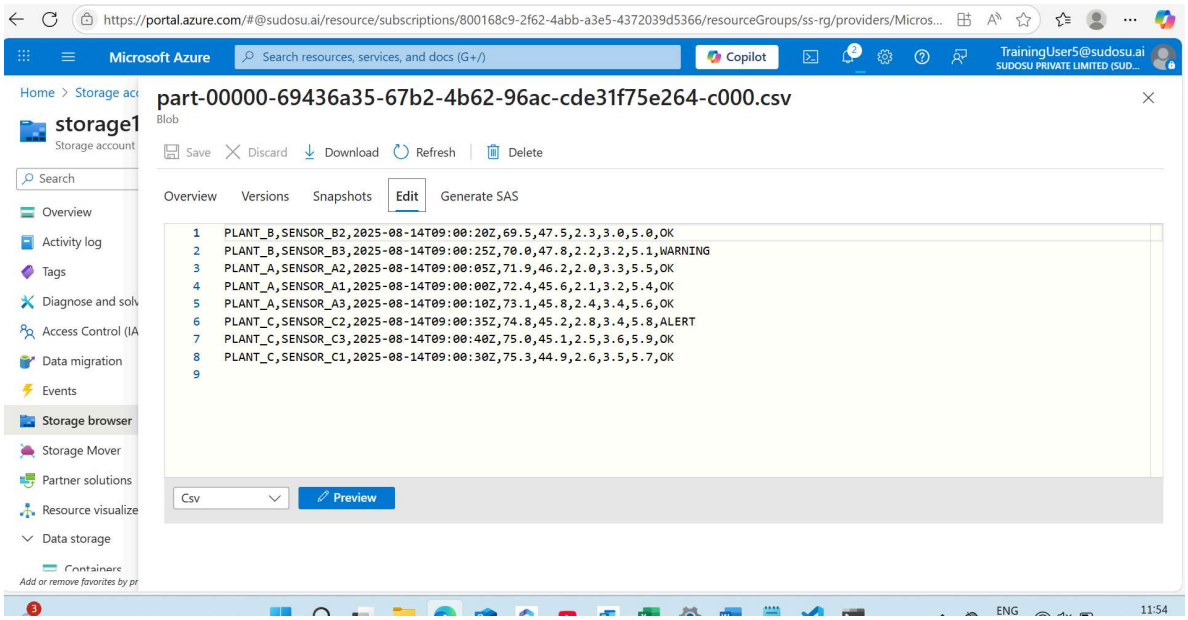


Running a pipeline from a data flow after applying transformations to the notebook means that the pipeline orchestrates the process of fetching raw data from the source, executing the defined data flow logic (such as sorting and filtering), and then triggering the notebook to perform additional processing or analytics.The data flow handles structured ETL tasks, while the notebook adds custom code-based operations, creating a seamless end-to-end data processing workflow.

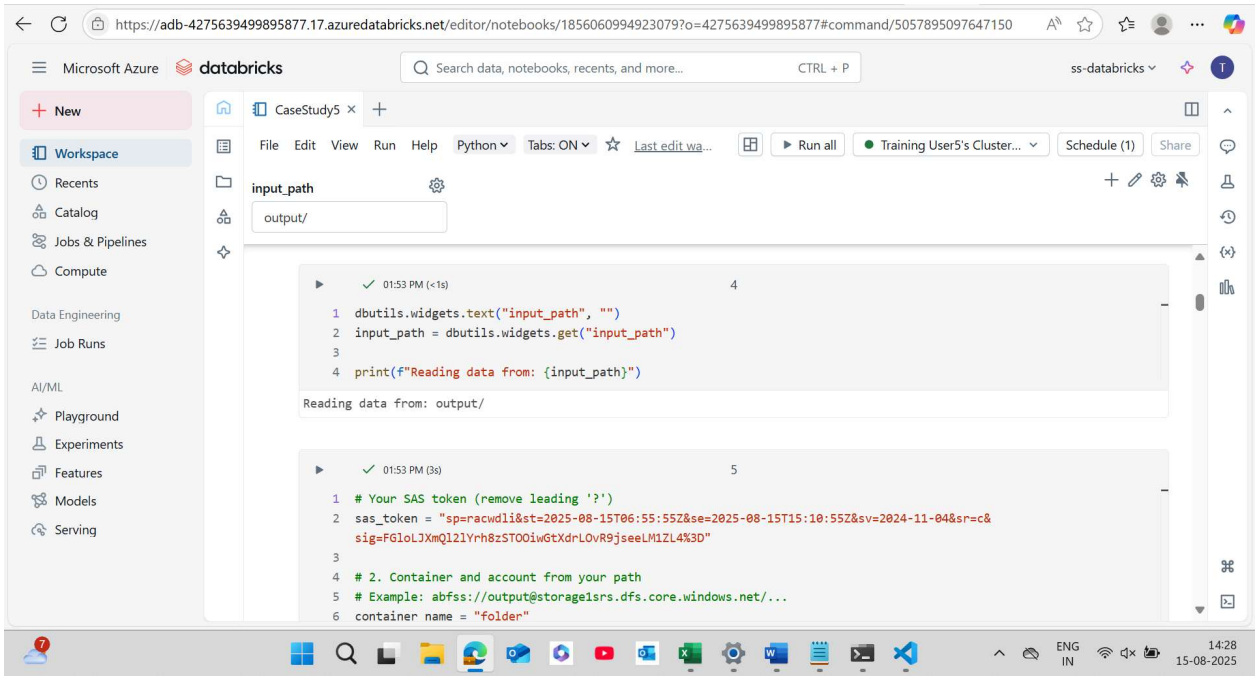


The csv file in a Sink blob container

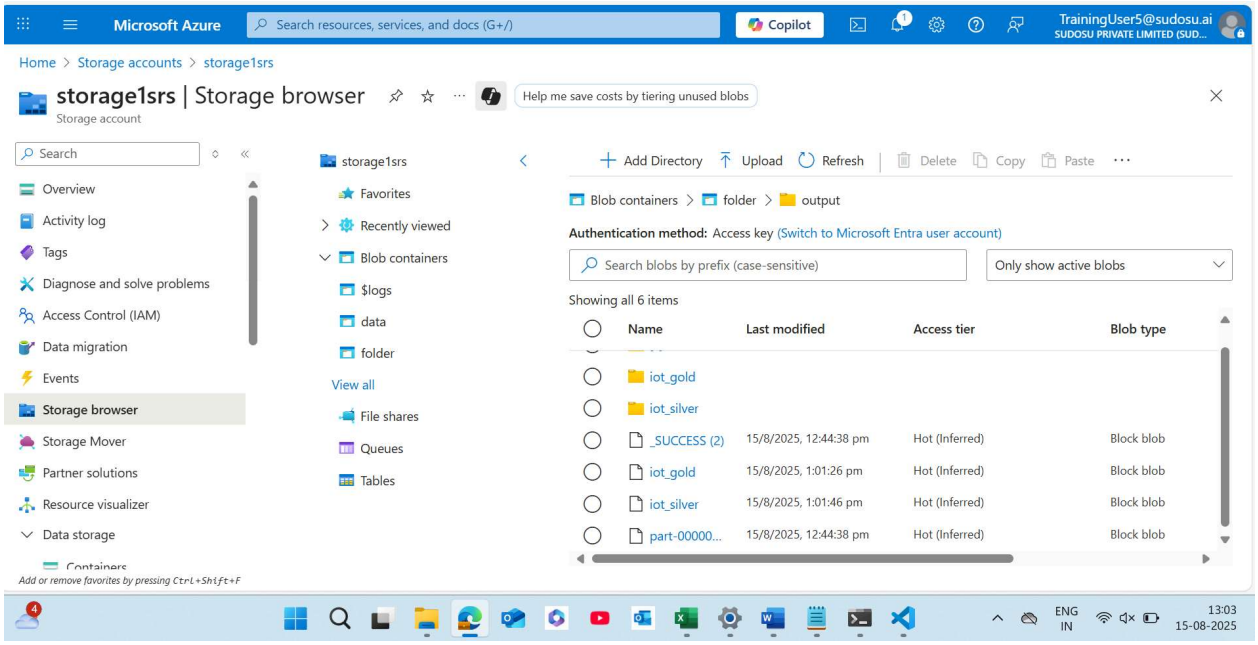




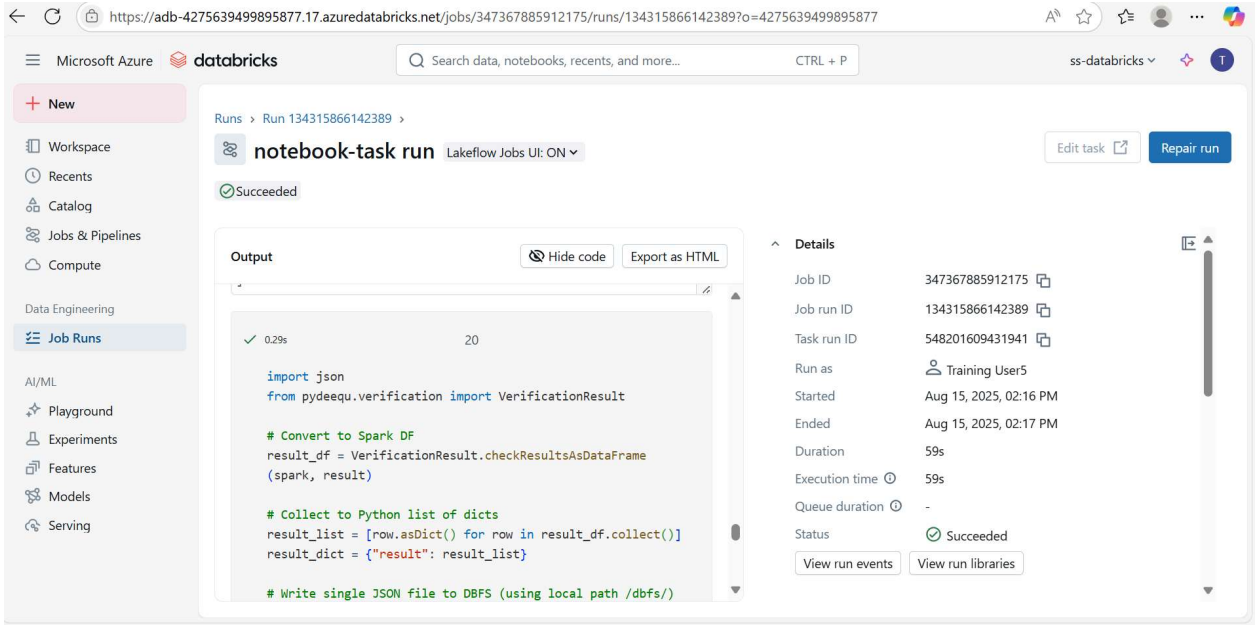
Notebook- the input_path from the output files after running the dataflow



Converted the data to silver and gold **(used Pydeequ)**



The Job Run- Succeded



The **Deequ checks passed**

Deequ in the GitHub Actions workflow is being used as an automated data quality testing framework.

Here's the flow in your case:

1. Databricks Notebook Execution – The GitHub Actions workflow triggers a Databricks notebook that runs PyDeequ checks (e.g., .isNonNegative(), .hasSize(), etc.) on IoT data. These checks validate that the dataset meets business and technical quality rules.

2. Result Generation – PyDeequ outputs the verification results (pass/fail for each check) in a structured JSON format and saves it to DBFS (/FileStore/.../deequ_result.json).

3. Report Actions – The GitHub Actions job downloads this JSON file from DBFS, prints it to the log, and uses jq to parse the check_status field.

    - If check_status is "Success", the workflow continues.

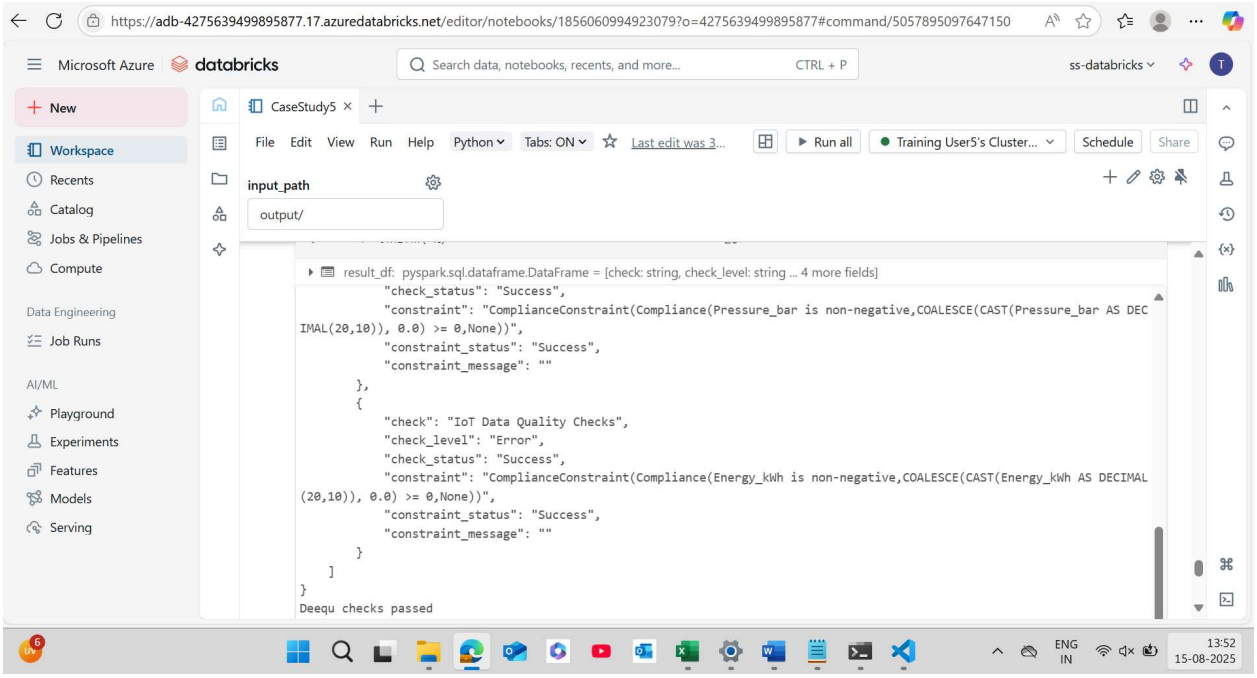
    - If check_status is "Error" or any failure, the workflow exits with a non-zero status, marking the Action as failed.

4. Purpose – This setup turns Deequ into an automated quality gate:

    - Ensures data is validated before merging code or deploying pipelines.

    - Detects anomalies or rule violations early in the development cycle.

It's basically unit testing for data—but running inside Databricks and controlled from GitHub Actions.

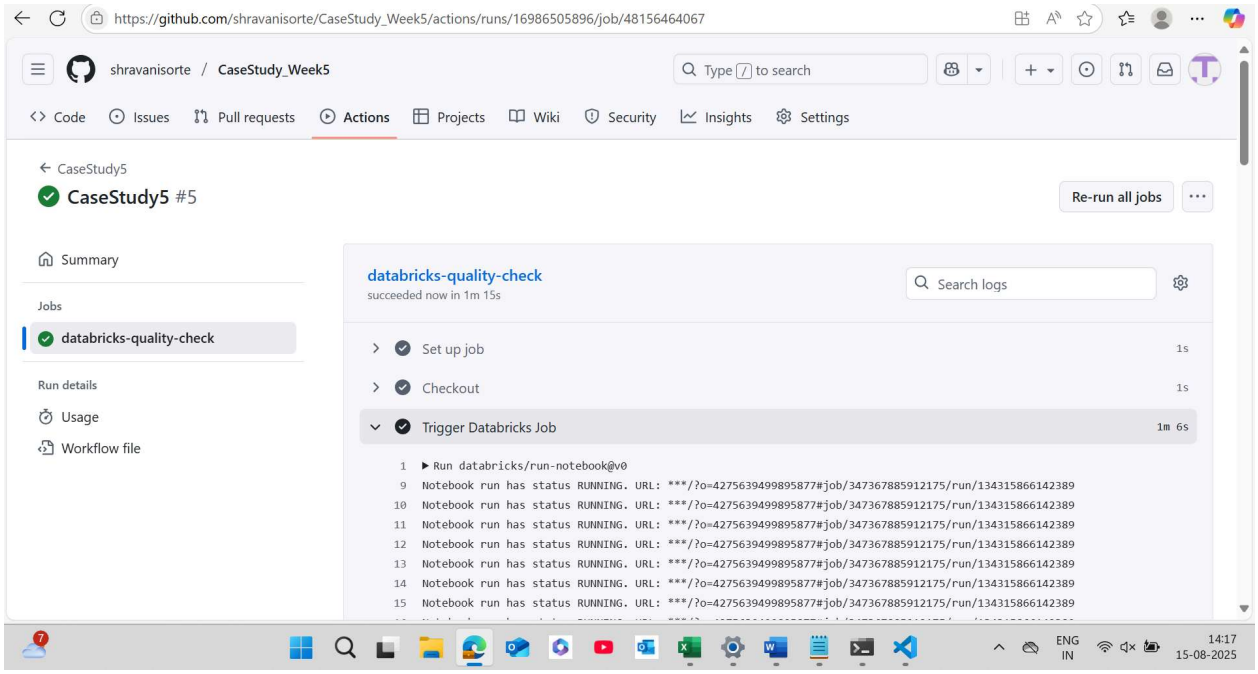


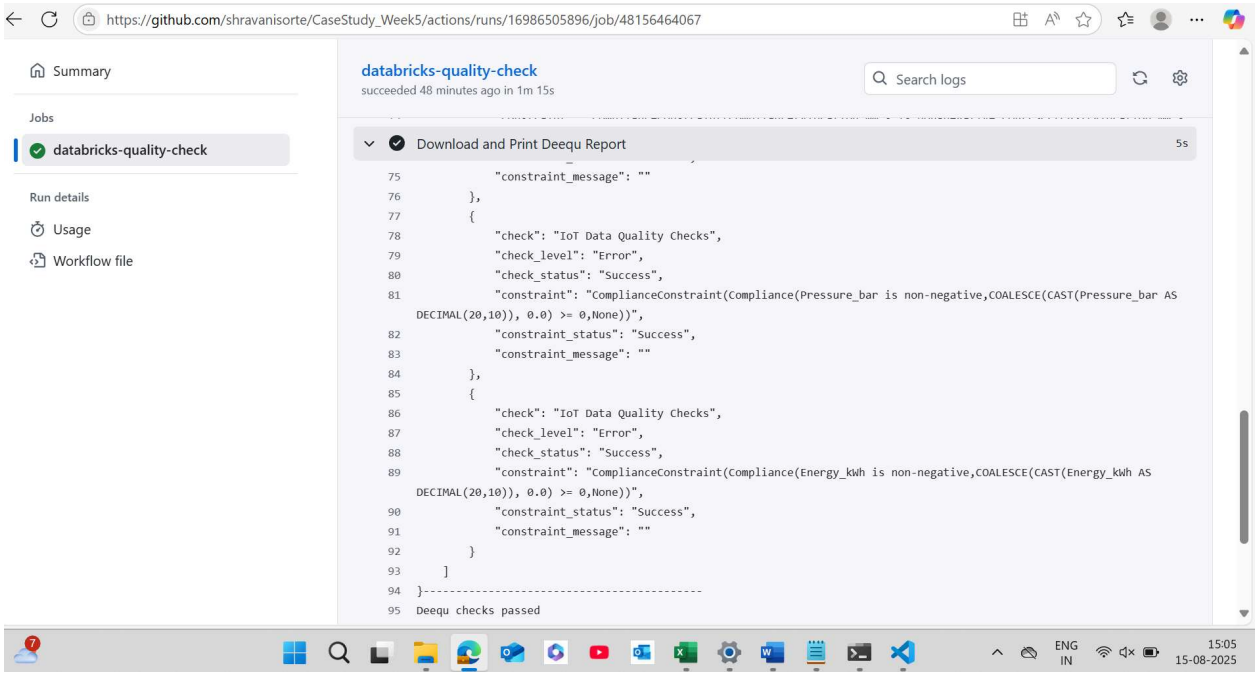
Github/Workflow

The report actions section of the GitHub workflow is responsible for retrieving and validating the results of the Databricks job run.

After the notebook finishes running on Databricks (where data quality checks are executed), this step downloads the generated JSON report from DBFS (Databricks File System) to the GitHub Actions runner. It then prints the report contents for visibility in the Actions log and uses tools like jq to parse the JSON, checking the check_status field.

If the status indicates failure, the workflow exits with an error code, ensuring that any pull request or manual run fails when data quality issues are detected. This enforces automated data validation and acts as a gatekeeper before merging or deploying changes.


https://github.com/shravanisorte/CaseStudy_Week5/actions/runs/16986505896/job/48156464067
shravanisorte / CaseStudy_Week5
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
CaseStudy5
CaseStudy5 #5
Re-run all jobs
Summary
Jobs
databricks-quality-check
Run details
Usage
Workflow file
databricks-quality-check
succeeded now in 1m 15s
Search logs
Set up job 1s
Checkout 1s
Trigger Databricks Job 1m 6s
1 ▶ Run databricks/run-notebook@v0
9 Notebook run has status RUNNING. URL: ***/?o=4275639499895877#job/347367885912175/run/134315866142389
10 Notebook run has status RUNNING. URL: ***/?o=4275639499895877#job/347367885912175/run/134315866142389
11 Notebook run has status RUNNING. URL: ***/?o=4275639499895877#job/347367885912175/run/134315866142389
12 Notebook run has status RUNNING. URL: ***/?o=4275639499895877#job/347367885912175/run/134315866142389
13 Notebook run has status RUNNING. URL: ***/?o=4275639499895877#job/347367885912175/run/134315866142389
14 Notebook run has status RUNNING. URL: ***/?o=4275639499895877#job/347367885912175/run/134315866142389
15 Notebook run has status RUNNING. URL: ***/?o=4275639499895877#job/347367885912175/run/134315866142389
ENG
IN
14:17
15-08-2025


https://github.com/shravanisorte/CaseStudy_Week5/actions/runs/16986505896/job/48156464067
Summary
Jobs
databricks-quality-check
Run details
Usage
Workflow file
databricks-quality-check
succeeded 48 minutes ago in 1m 15s
Search logs
Download and Print Deequ Report 5s
"constraint_message": ""
},
{
"check": "IoT Data Quality Checks",
"check_level": "Error",
"check_status": "Success",
"constraint": "ComplianceConstraint(Compliance(Pressure_bar is non-negative,COALESCE(CAST(Pressure_bar AS DECIMAL(20,10)), 0.0) >= 0,None))",
"constraint_status": "Success",
"constraint_message": ""
},
{
"check": "IoT Data Quality Checks",
"check_level": "Error",
"check_status": "Success",
"constraint": "ComplianceConstraint(Compliance(Energy_kWh is non-negative,COALESCE(CAST(Energy_kWh AS DECIMAL(20,10)), 0.0) >= 0,None))",
"constraint_status": "Success",
"constraint_message": ""
}
]
}----------
Deequ checks passed
ENG
IN
15:05
15-08-2025