Cloud-Based Data Pipeline for CSV Ingestion and Reporting using Azure and Databricks

by

Santiago Calvo Patiño

Tabla de Contenido

Introducción

This project addresses a technical challenge focused on database migration and analytical reporting. While the original requirement was to implement a local REST API for CSV file ingestion and batch insertion into a SQL database, this solution showcases a **cloud-native architecture** using **Azure services** and **Databricks**, demonstrating a modern, scalable approach to data engineering.

The goal was to:

- Ingest CSV files received via email
- Load the data into a SQL-like storage system
- Apply transformations
- Generate reports based on business logic
- Automate the entire workflow from input to delivery

Architecture Overview

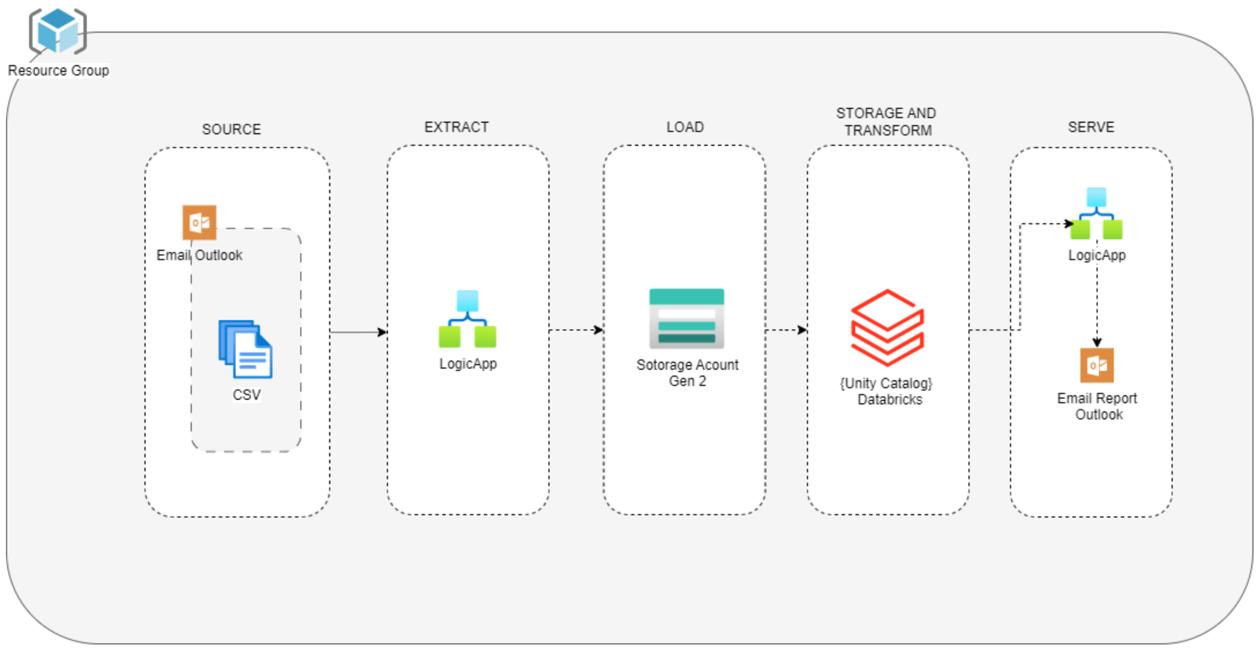


Imagen 1. Architecture Solution

The process is composed of five main stages:

a. Source

- CSV files are received via **Outlook email** as attachments.
- Each file represents one of the required entities: departments, jobs, employees.

b. Extract

- An **Azure Logic App** is triggered when a new email arrives.
- It extracts attachments and prepares them for storage.

c. Load

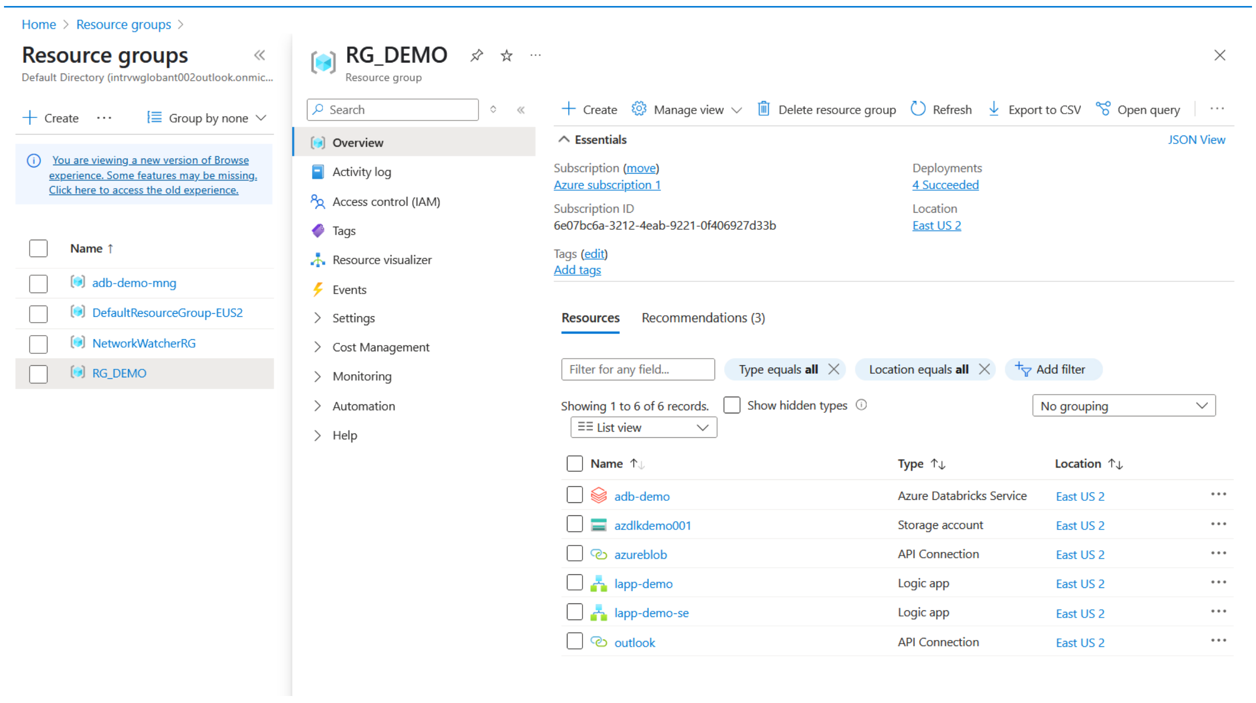
- The files are uploaded to **Azure Data Lake Storage Gen2**, organized into bronze-layer folders.

d. Storage and Transform

- **Databricks**, managed with **Unity Catalog**, processes the data:
  - Loads the CSVs into Delta tables
  - Performs data cleaning and transformations
  - Implements the required analytical SQL logic

e. Serve

- The final metrics are exported as a formatted **HTML report**.
- A Logic App then sends this report via **Outlook email** to stakeholders.


Home > Resource groups >

Resource groups
Default Directory (intrvwglobant002outlook.onmic...

+ Create ··· Group by none

You are viewing a new version of Browse experience. Some features may be missing. Click here to access the old experience.

Name ↑
adb-demo-mng
DefaultResourceGroup-EUS2
NetworkWatcherRG
RG_DEMO

RG_DEMO
Resource group

Search

Overview
Activity log
Access control (IAM)
Tags
Resource visualizer
Events
Settings
Cost Management
Monitoring
Automation
Help

+ Create Manage view Delete resource group Refresh Export to CSV Open query

Essentials JSON View

Subscription (move)
Azure subscription 1
Subscription ID
6e07bc6a-3212-4eab-9221-0f406927d33b
Tags (edit)
Add tags

Deployments
4 Succeeded
Location
East US 2

Resources Recommendations (3)

Filter for any field... Type equals **all** Location equals **all** Add filter

Showing 1 to 6 of 6 records. Show hidden types No grouping
List view

Name ↑↓	Type ↑↓	Location ↑↓
adb-demo	Azure Databricks Service	East US 2
azdlkdemo001	Storage account	East US 2
azureblob	API Connection	East US 2
lapp-demo	Logic app	East US 2
lapp-demo-se	Logic app	East US 2
outlook	API Connection	East US 2

General Deployment Flow

1. **Created Resource Group RG_DEMO** to logically organize all components.
2. **Deployed Azure Databricks Workspace** named adb-demo using the **Premium Free Trial** setup.
    - This triggered the automatic provisioning of a **management resource group** (adb-demo-mng), which includes:
        - A **Managed Identity**
        - A **VNet** with subnets (workers-vnet)
        - A **NAT Gateway** with Public IP
        - **Unity Catalog access connector**

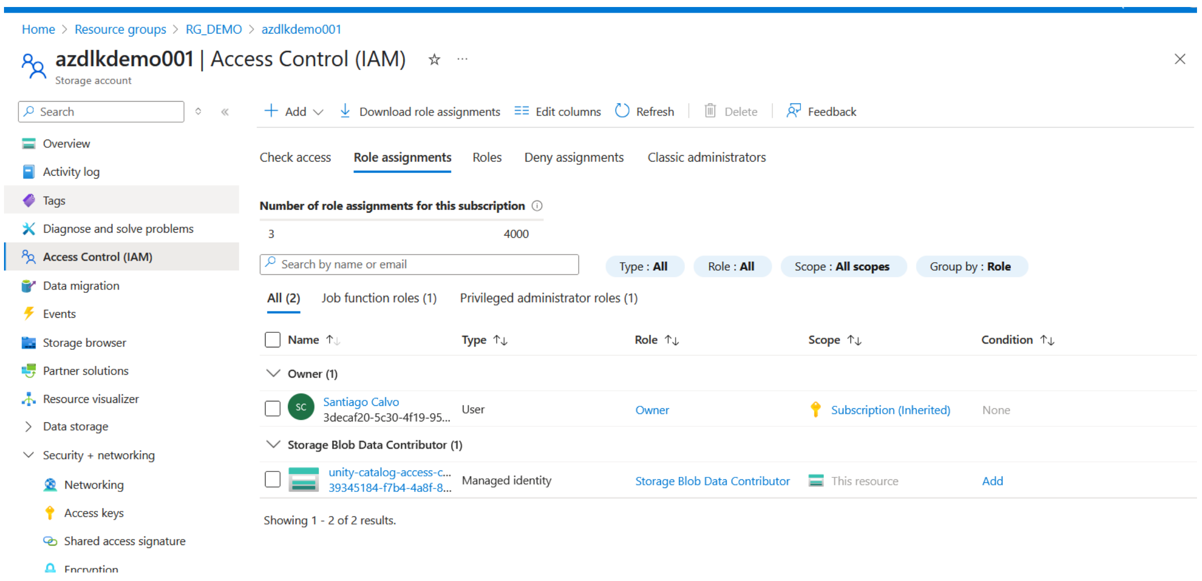
3. **Created a Storage Account (ADLS Gen2)** for Bronze layer ingestion.
4. **Configured Logic Apps**:
   - One for **email ingestion and storage upload**
   - Another for **email delivery with processed report**

Unity Catalog Setup

During the creation of the **Databricks workspace**, the Unity Catalog was automatically configured as part of the deployment. This included:

- Provisioning of the unity-catalog-access-connector
- Configuration of default **metastore** using the attached storage
- Integration with the **Managed Identity** to securely access the storage layer

No additional Unity Catalog manual setup was required. The system provisioned the minimum viable governance model automatically using Azure-native services.



Number of role assignments for this subscription

3 4000

Search by name or email | Type : All | Role : All | Scope : All scopes | Group by : Role

All (2) Job function roles (1) Privileged administrator roles (1)

Name	Type	Role	Scope	Condition
Owner (1)				
Santiago Calvo 3decaf20-5c30-4f19-95...	User	Owner	Subscription (Inherited)	None
Storage Blob Data Contributor (1)				
unity-catalog-access-c... 39345184-f7b4-4a8f-8...	Managed identity	Storage Blob Data Contributor	This resource	Add

Showing 1 - 2 of 2 results.

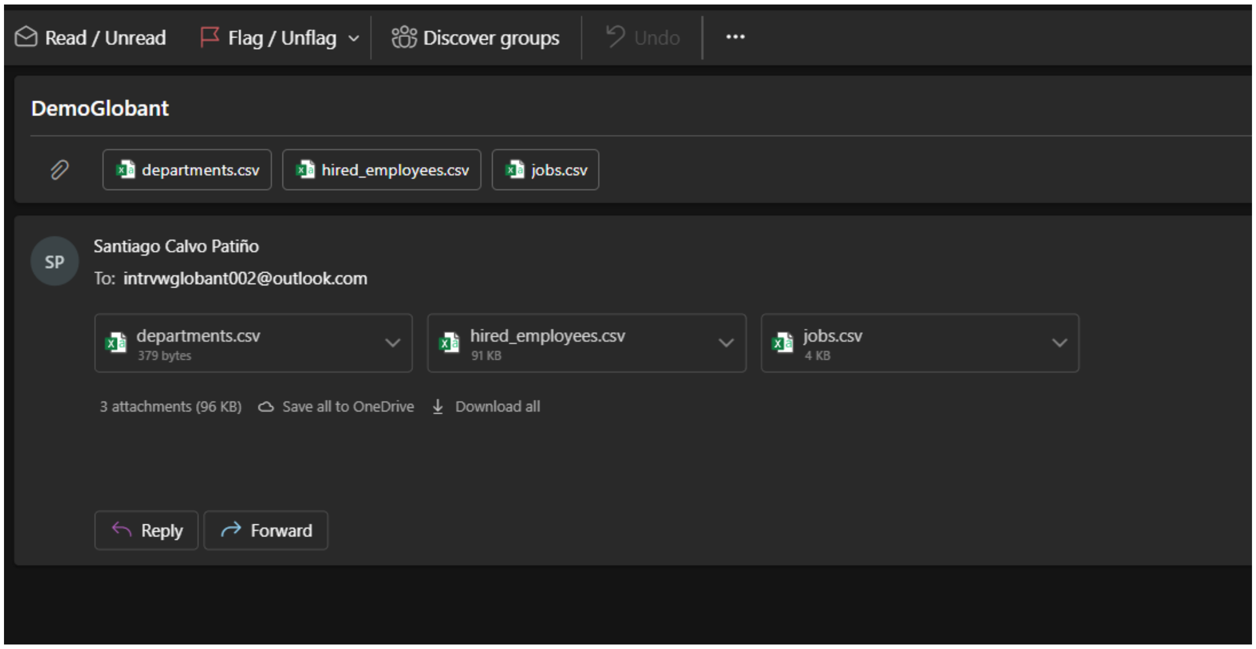
To enable **Databricks Unity Catalog** to interact securely with the Azure Data Lake Storage account azdlkdemo001, a **Managed Identity** named unity-catalog-access-connector was used.

This identity was **manually granted** the following role assignment:

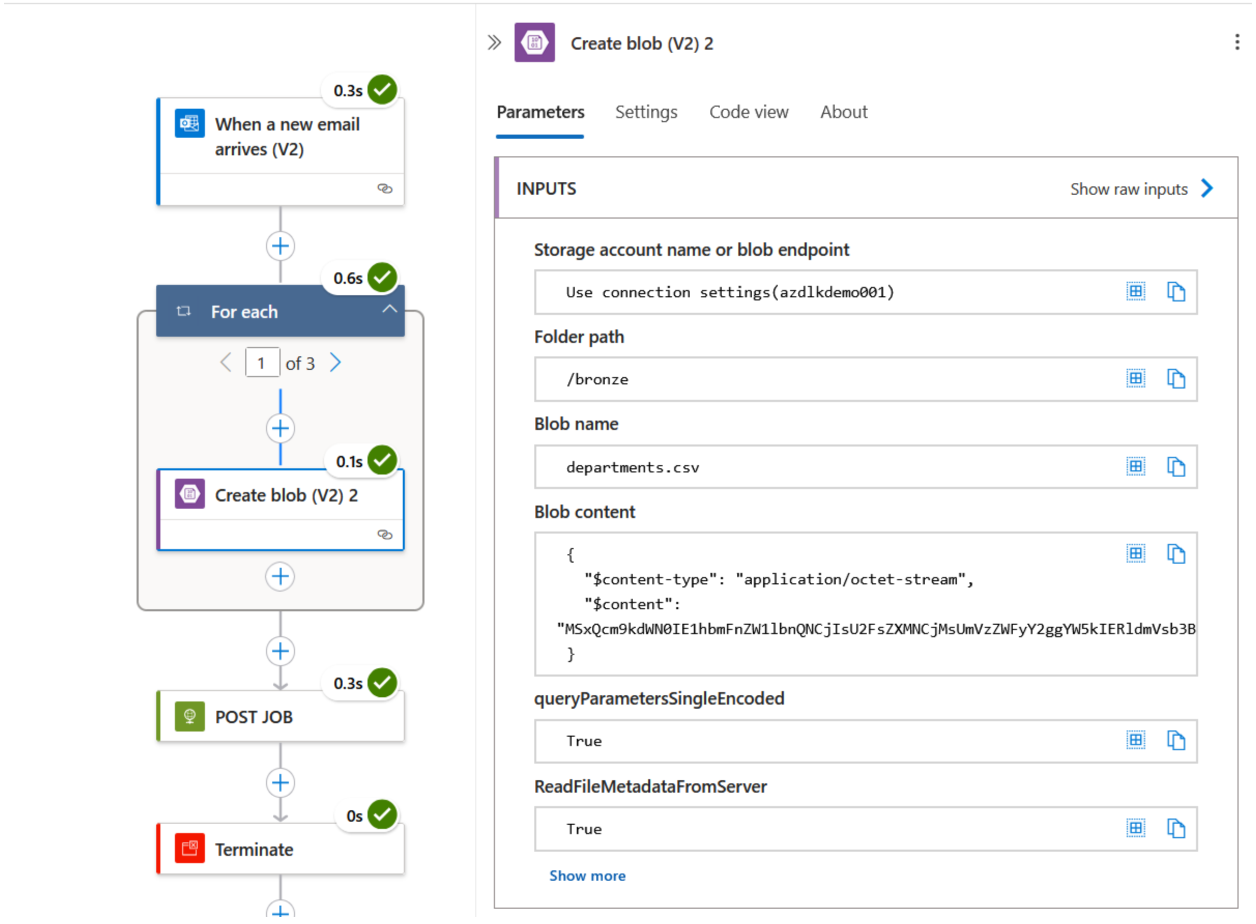
- **Role**: Storage Blob Data Contributor
- **Scope**: *This resource only* (specific to the azdlkdemo001 storage account)

Email Ingestion Process: Entry Point of the Pipeline

The data pipeline begins with the reception of an email sent to a dedicated address: **intrwglobant002@outlook.com.** This email contains three key CSV files: departments.csv, hired_employees.csv, and jobs.csv. These files form the core of the dataset required for the analysis and reporting process.



A Logic App is responsible for monitoring the inbox and is configured to trigger only when an email arrives **with a specific subject line**. In this case, the subject used is **“DemoGlobant”**. This condition ensures the workflow is only activated when a valid dataset submission is received, avoiding unnecessary executions due to unrelated messages.



Create blob (V2) 2

Parameters Settings Code view About

INPUTS Show raw inputs

Storage account name or blob endpoint

```
Use connection settings(azdlkdemo001)
```

Folder path

```
/bronze
```

Blob name

```
departments.csv
```

Blob content

```
{
  "$content-type": "application/octet-stream",
  "$content":
"MSxQcm9kdWN0IE1hbmFnZW1lbnQNCjIsU2FsZXMNCjMsUmVzZWFyY2ggYW5kIERldmVsb3B
}
```

queryParametersSingleEncoded

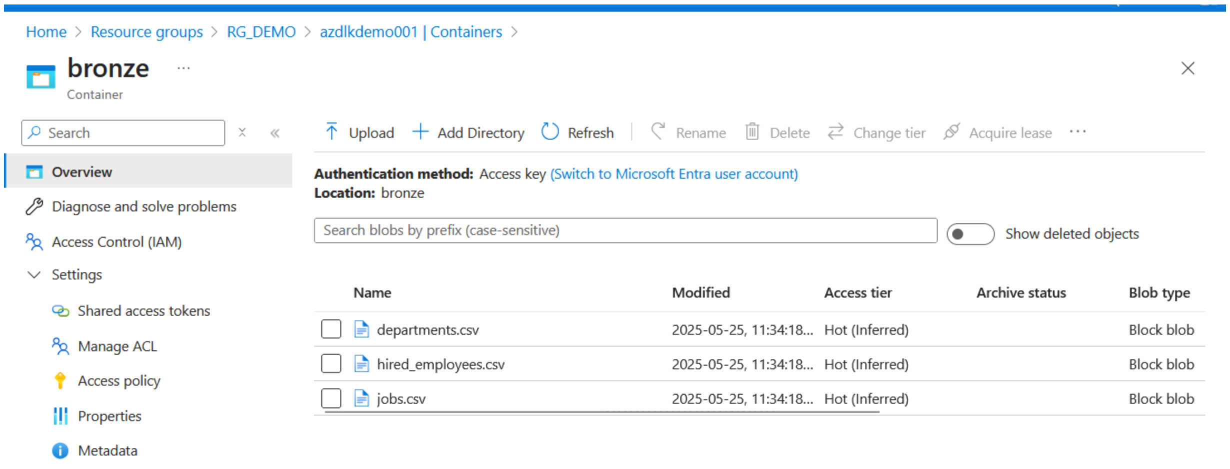
```
True
```

ReadFileMetadataFromServer

```
True
```

Show more

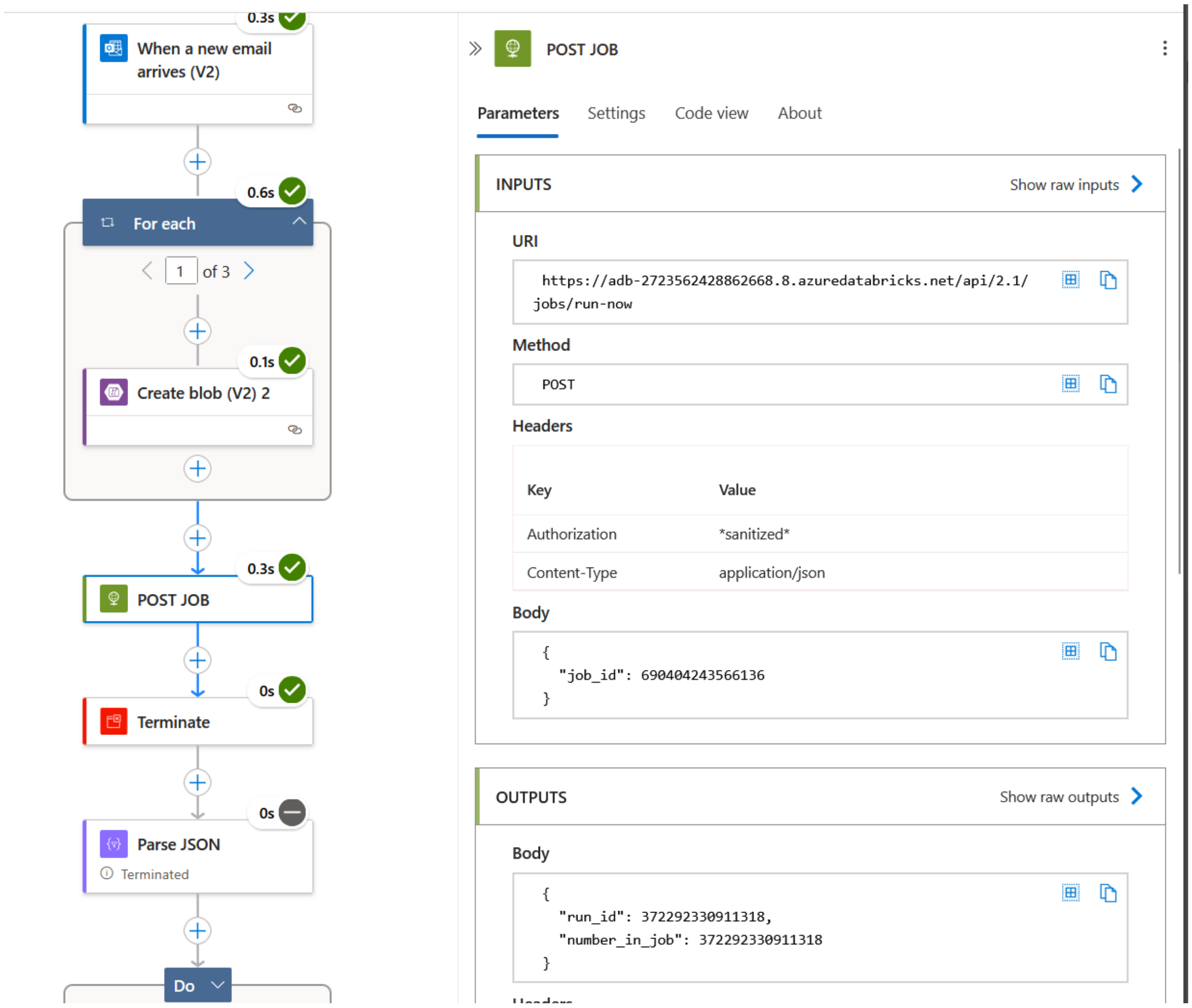
Immediately after extraction, the Logic App uploads the files to an **Azure Data Lake Gen2** storage account (**azdlkdemo001**). The files are stored in a predefined structure within the lake, typically in a **bronze** folder, maintaining a clean separation of ingestion layers.



Authentication method: Access key (Switch to Microsoft Entra user account)
Location: bronze

Name	Modified	Access tier	Archive status	Blob type
departments.csv	2025-05-25, 11:34:18...	Hot (Inferred)		Block blob
hired_employees.csv	2025-05-25, 11:34:18...	Hot (Inferred)		Block blob
jobs.csv	2025-05-25, 11:34:18...	Hot (Inferred)		Block blob

After the upload is successfully completed, the Logic App **calls a REST API to trigger a Databricks job**. This job is responsible for picking up the freshly ingested files, transforming the data, and generating the necessary metrics for reporting. The orchestration ensures that as soon as new data is available, processing begins automatically — making the pipeline event-driven and fully automated from ingestion to transformation.



URI

```
https://adb-2723562428862668.8.azuredatabricks.net/api/2.1/jobs/run-now
```

Method

```
POST
```

Headers

Key	Value
Authorization	*sanitized*
Content-Type	application/json

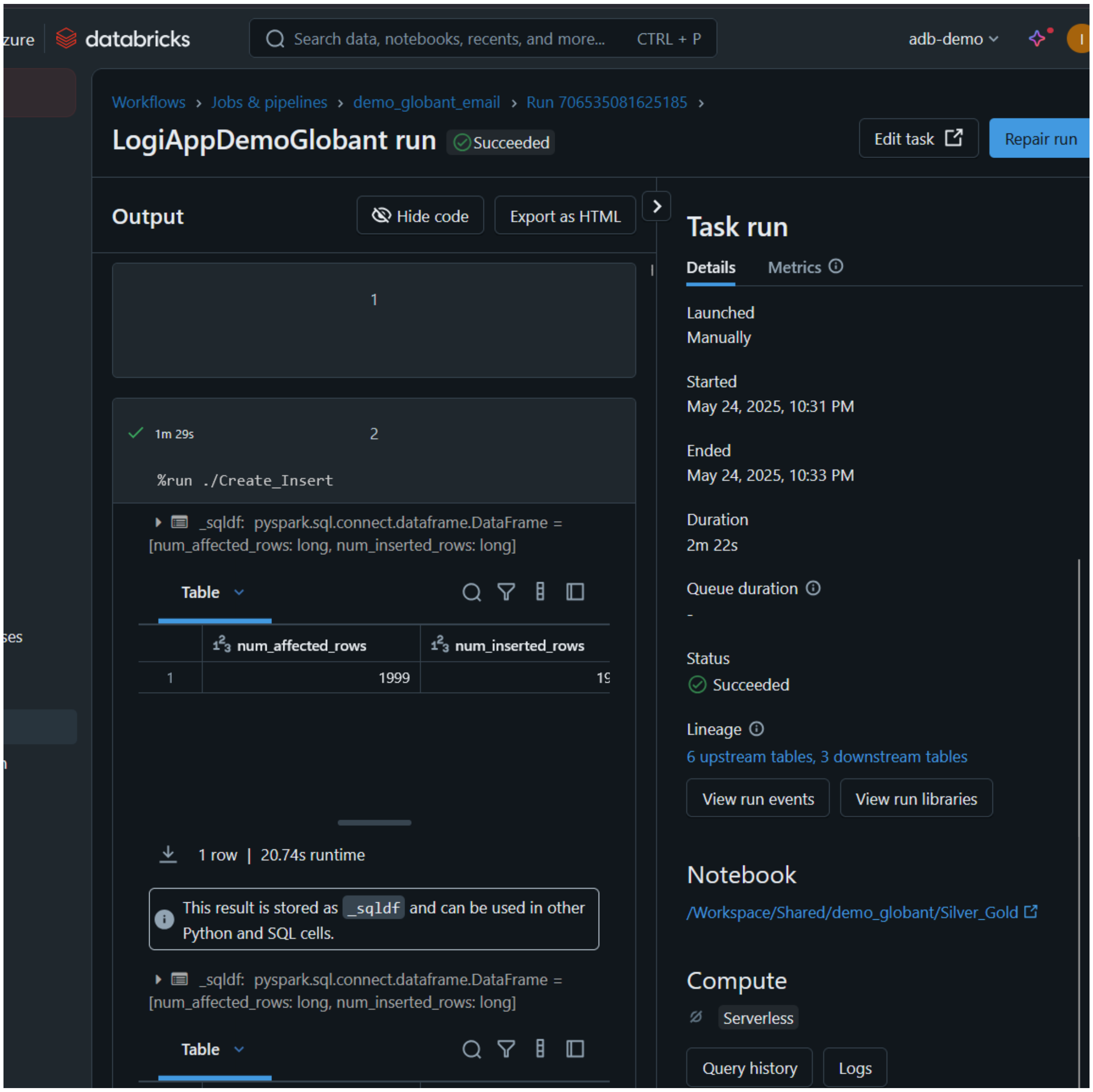
Body

```
{
  "job_id": 690404243566136
}
```

Body

```
{
  "run_id": 372292330911318,
  "number_in_job": 372292330911318
}
```

Once the CSV files are uploaded to Azure Data Lake Gen2 and the Databricks job is triggered by the Logic App, the transformation stage of the pipeline begins. This step is handled within **Azure Databricks**, where a notebook was developed to manage the catalog structure and load the data.


databricks
Search data, notebooks, recents, and more... CTRL + P
adb-demo
Workflows › Jobs & pipelines › demo_globant_email › Run 706535081625185 ›
Edit task
Repair run
Output
Hide code
Export as HTML
Table
View run events
View run libraries
Query history
Logs

LogiAppDemoGlobant run Succeeded

1

1m 29s 2

```
%run ./Create_Insert
```

_sqldf: pyspark.sql.connect.dataframe.DataFrame = [num_affected_rows: long, num_inserted_rows: long]

	num_affected_rows	num_inserted_rows
1	1999	19

1 row | 20.74s runtime

This result is stored as `_sqldf` and can be used in other Python and SQL cells.

_sqldf: pyspark.sql.connect.dataframe.DataFrame = [num_affected_rows: long, num_inserted_rows: long]

Task run

Details Metrics

Launched
Manually

Started
May 24, 2025, 10:31 PM

Ended
May 24, 2025, 10:33 PM

Duration
2m 22s

Queue duration
-

Status
Succeeded

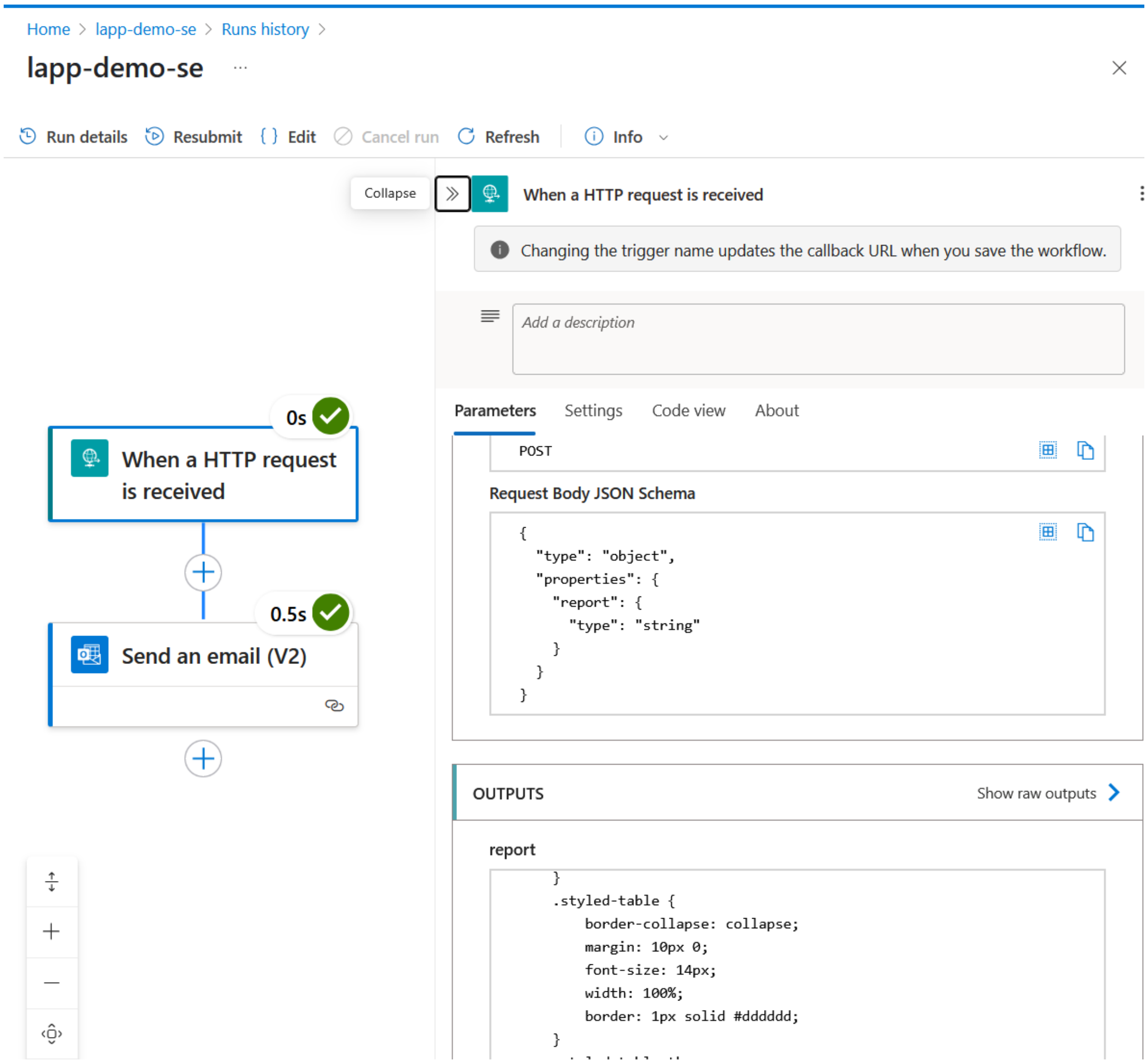
Lineage
6 upstream tables, 3 downstream tables

Notebook
/Workspace/Shared/demo_globant/Silver_Gold

Compute
Serverless

After the ingestion step, the second notebook in Databricks — called **Silver_Gold** — is automatically triggered. This notebook handles the full transformation of the raw data stored in the bronze layer. It calculates the key business metrics requested in the challenge, such as the number of employees hired per department and job, broken down by quarter, as well as identifying departments with above-average hiring.

Once the data is transformed, I generate a dynamic **HTML report** inside the notebook, styled using inline CSS to ensure it's readable and visually structured. The notebook then **sends this report to a Logic App using an HTTP POST request**. That Logic App is responsible for sending out the final email to stakeholders with the full results embedded in the message body.



Changing the trigger name updates the callback URL when you save the workflow.

Add a description

POST

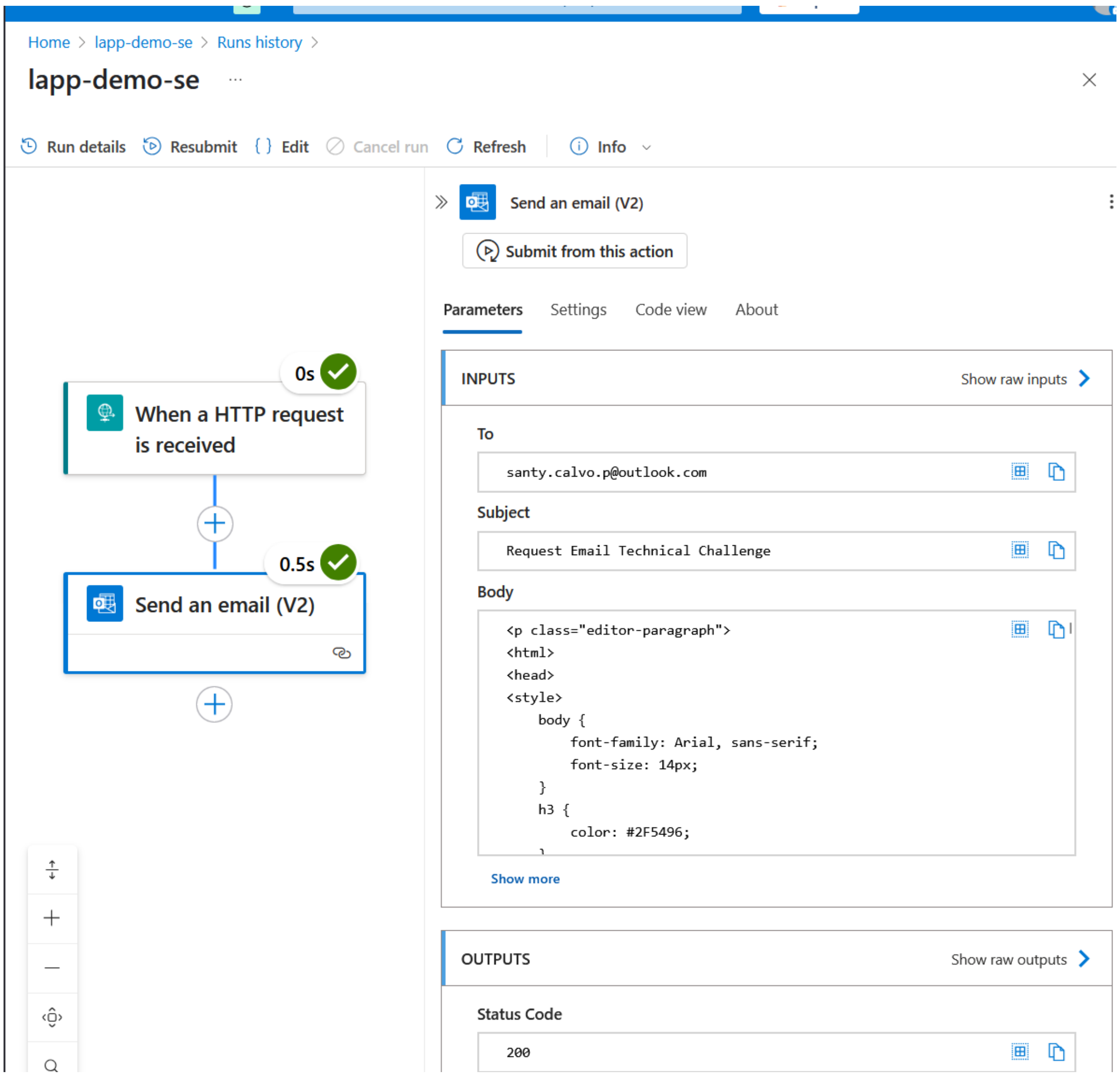
Request Body JSON Schema

```
{
  "type": "object",
  "properties": {
    "report": {
      "type": "string"
    }
  }
}
```

report

```
        }
        .styled-table {
            border-collapse: collapse;
            margin: 10px 0;
            font-size: 14px;
            width: 100%;
            border: 1px solid #dddddd;
        }
```

E-mail settings



To

```
santy.calvo.p@outlook.com
```

Subject

```
Request Email Technical Challenge
```

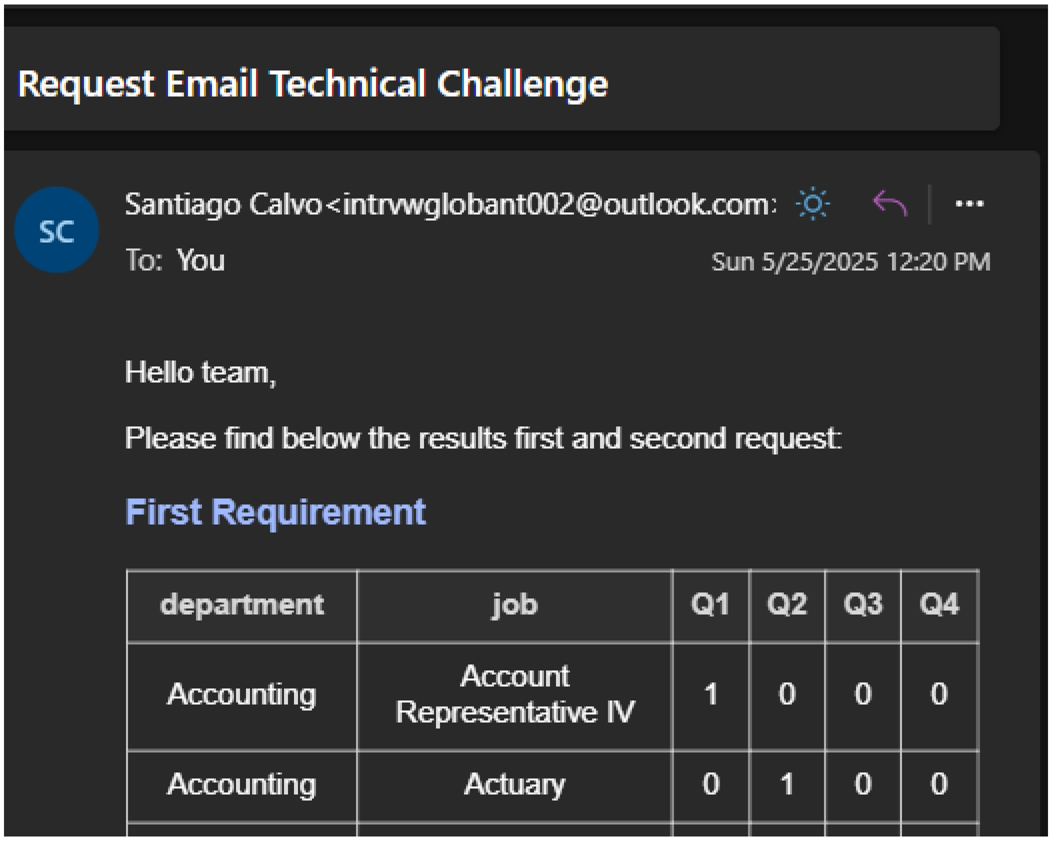
Body

```
<p class="editor-paragraph">
<html>
<head>
<style>
    body {
        font-family: Arial, sans-serif;
        font-size: 14px;
    }
    h3 {
        color: #2F5496;
```

Status Code

```
200
```

As the final step of the pipeline, a dynamic HTML report is sent by email once the data has been successfully transformed and analyzed. This report contains the output of both requirements specified in the technical challenge.



Request Email Technical Challenge

Santiago Calvo<intrvwglobant002@outlook.com>

To: You

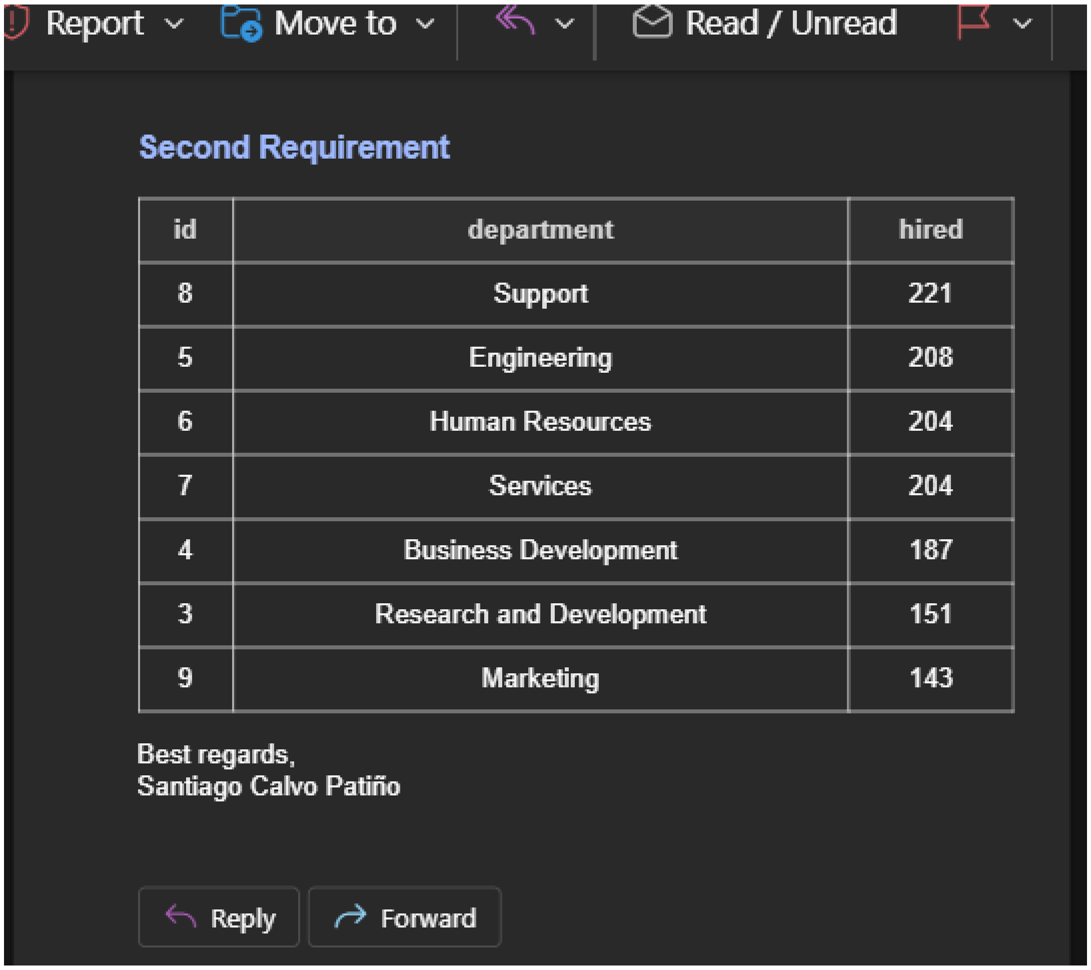
Sun 5/25/2025 12:20 PM

Hello team,

Please find below the results first and second request:

First Requirement

department	job	Q1	Q2	Q3	Q4
Accounting	Account Representative IV	1	0	0	0
Accounting	Actuary	0	1	0	0



Second Requirement

id	department	hired
8	Support	221
5	Engineering	208
6	Human Resources	204
7	Services	204
4	Business Development	187
3	Research and Development	151
9	Marketing	143

Best regards,
Santiago Calvo Patiño