## 📘 Detailed Notes: Azure Data Factory Project – Copying Static & Dynamic Data from REST APIs to ADLS Gen2

### 🧭 Project Goal:

Build an end-to-end data pipeline using **Azure Data Factory (ADF)** to extract **static and dynamic data** from **REST APIs**, apply transformations and schema mapping, and load the data into **Azure Data Lake Storage Gen2 (ADLS Gen2)** in a structured format (CSV/Parquet/JSON).

### 🔗 1. Linked Services – First Step Always

*Linked Services act as the connection bridge between ADF and external data sources or sinks.*

* **Why?** They contain authentication details and connection strings required to access data sources (REST APIs) and destinations (ADLS Gen2).
* **Created for:**
  + ✅ Source: **REST API**
  + ✅ Sink: **ADLS Gen2**

**🔁 Best Practice:** Always create linked services **before creating datasets**.

### 📁 2. Data Types Handled

*Proper data classification helps in organizing workflows and processing strategies.*

* **Static Data**
  + One-time or rarely updated data
  + E.g.: Country codes, department master data, product catalog
* **Dynamic Data**
  + Changes frequently and fetched based on parameters (e.g., dates)
  + E.g.: Daily sales, logs, API data with time filters

### 💡 3. Key Best Practices

* 🟢 **If Source is CSV → Keep Destination as CSV**
  + Ensures **schema consistency**
  + Avoids **data type mismatches**
  + Reduces transformation complexity
* 🛠️ **Always Enable Schema Mapping**
  + When **source and destination have different column orders or names**
  + Use **Copy Activity → Mapping tab** to map fields correctly

### 🔁 4. Parameterization & Dynamic Pipelines

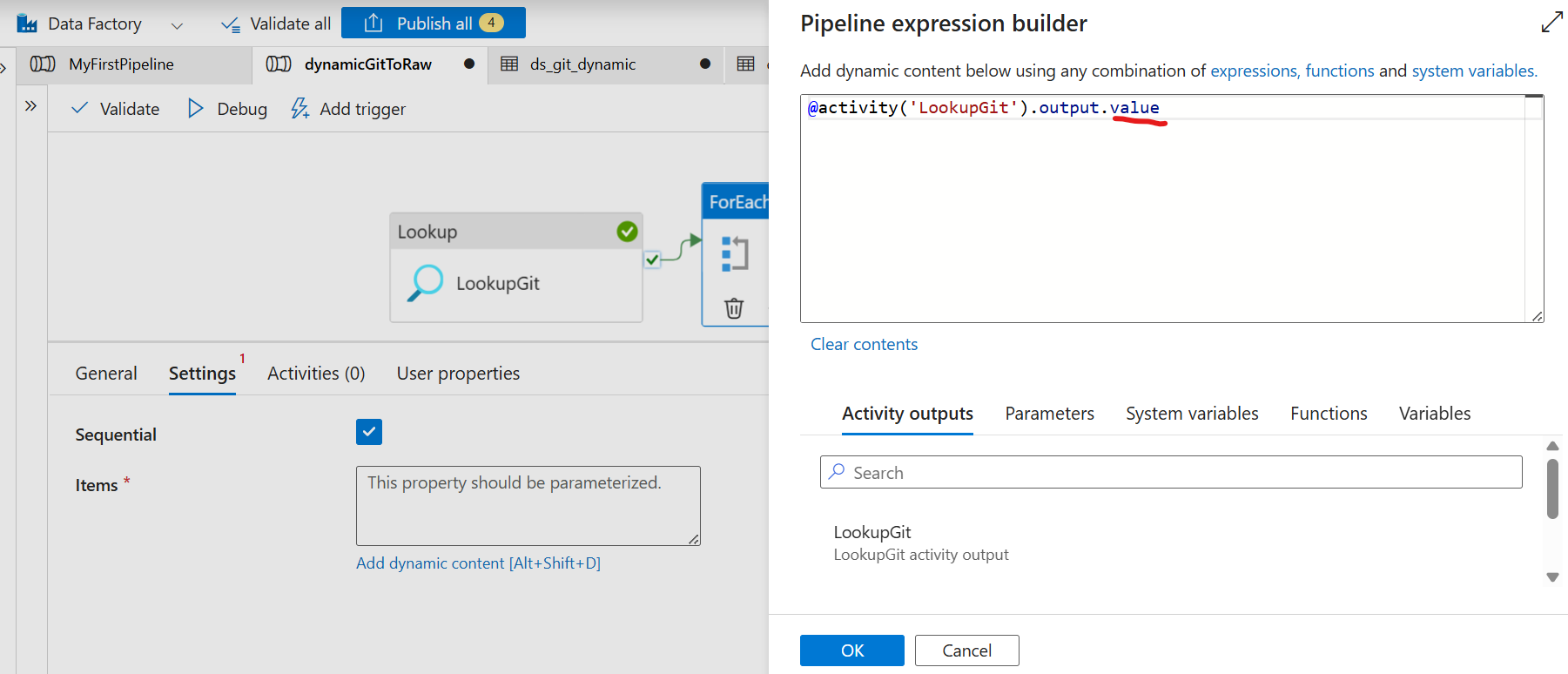
*Helps make pipelines reusable across different environments or datasets.*

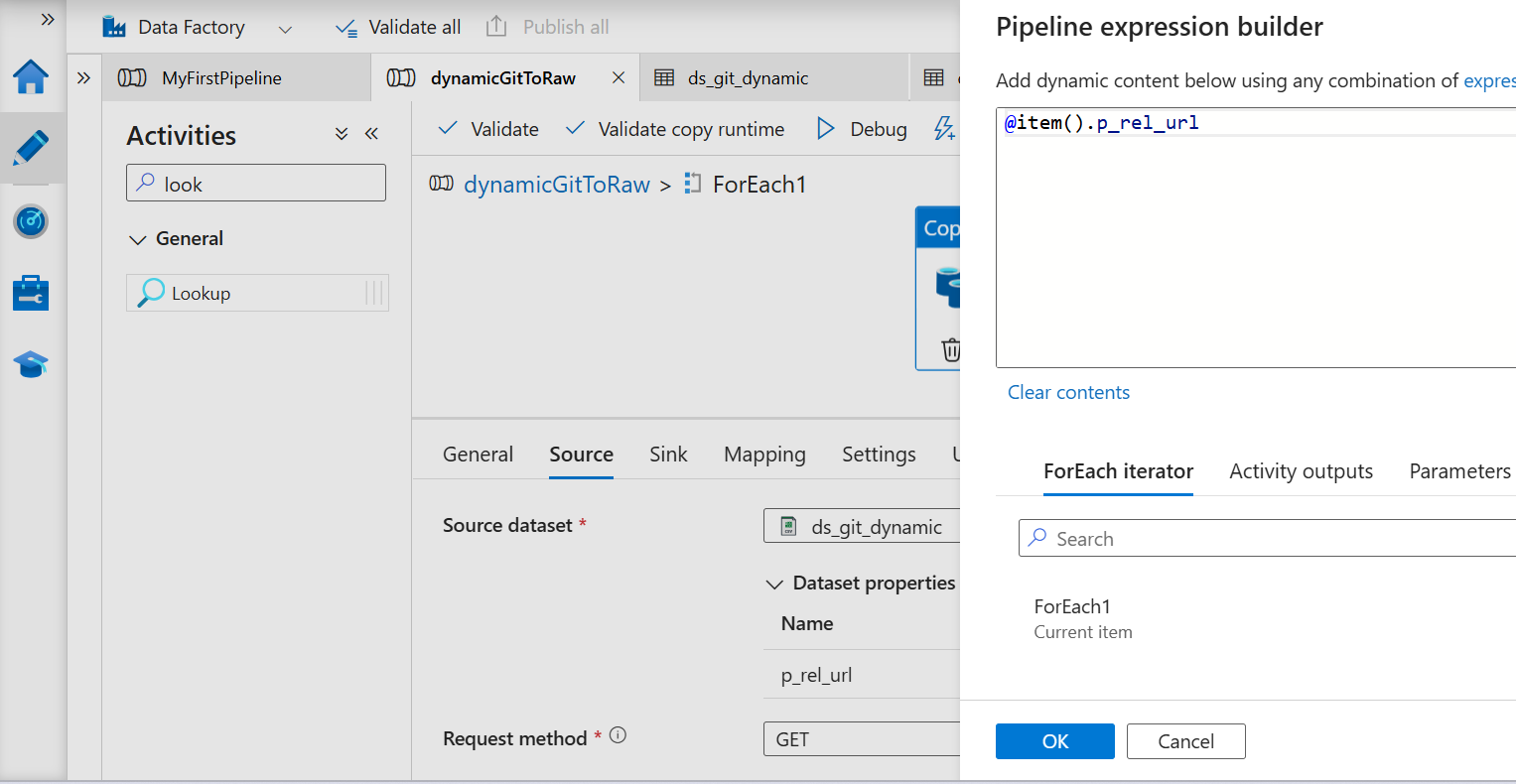
* **ADF Parameters Used For:**
  + Dynamic **API URLs**
  + Dynamic **file names** or **paths**
  + Start/End **date ranges**
  + Output folder structure (/year/month/day/)
* Used **expression language** (e.g., @concat(), @pipeline().parameters) in:  
  + Dataset paths
  + REST API query parameters
  + File naming conventions

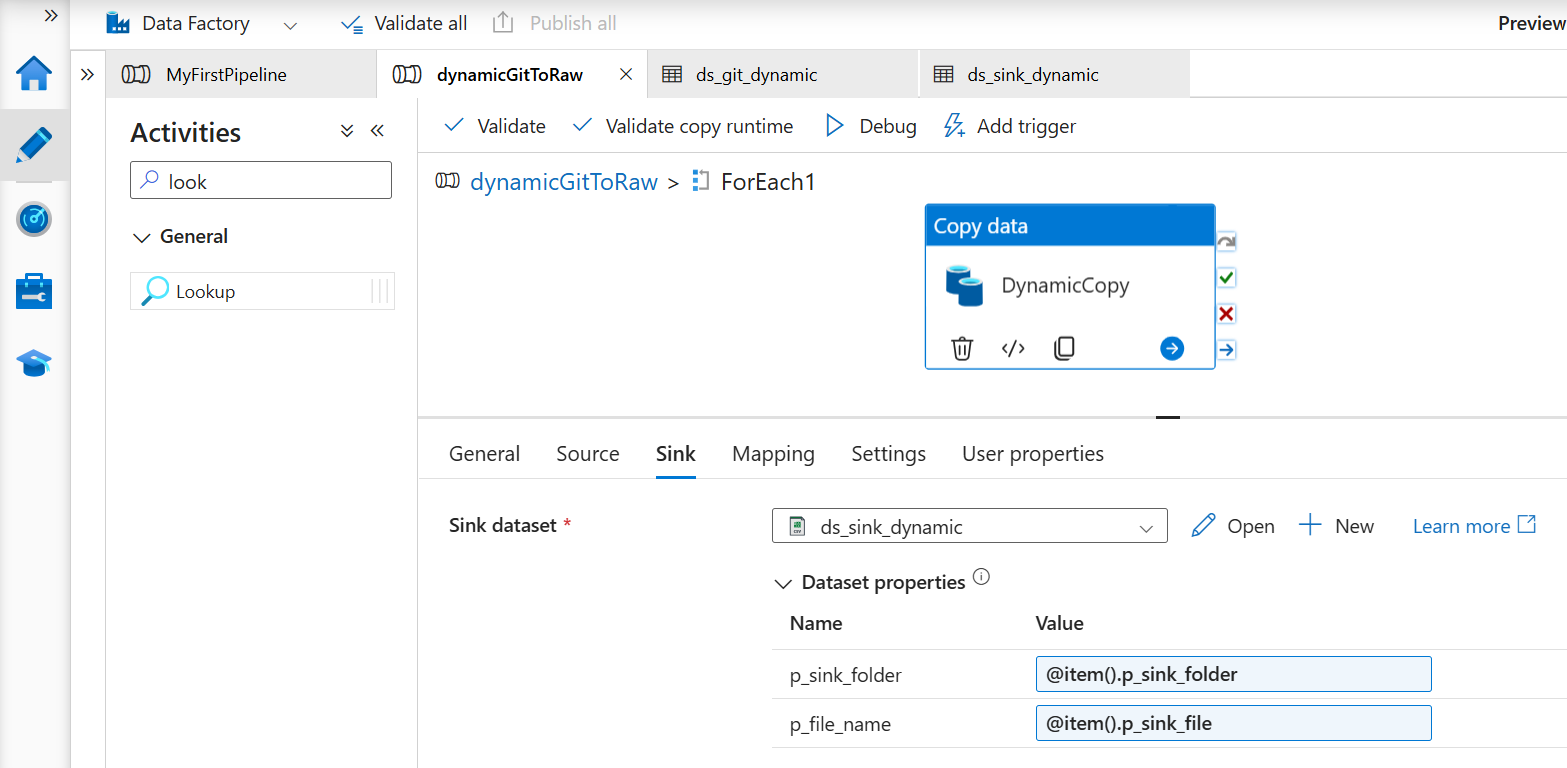
### 🔍 5. Lookup Activity

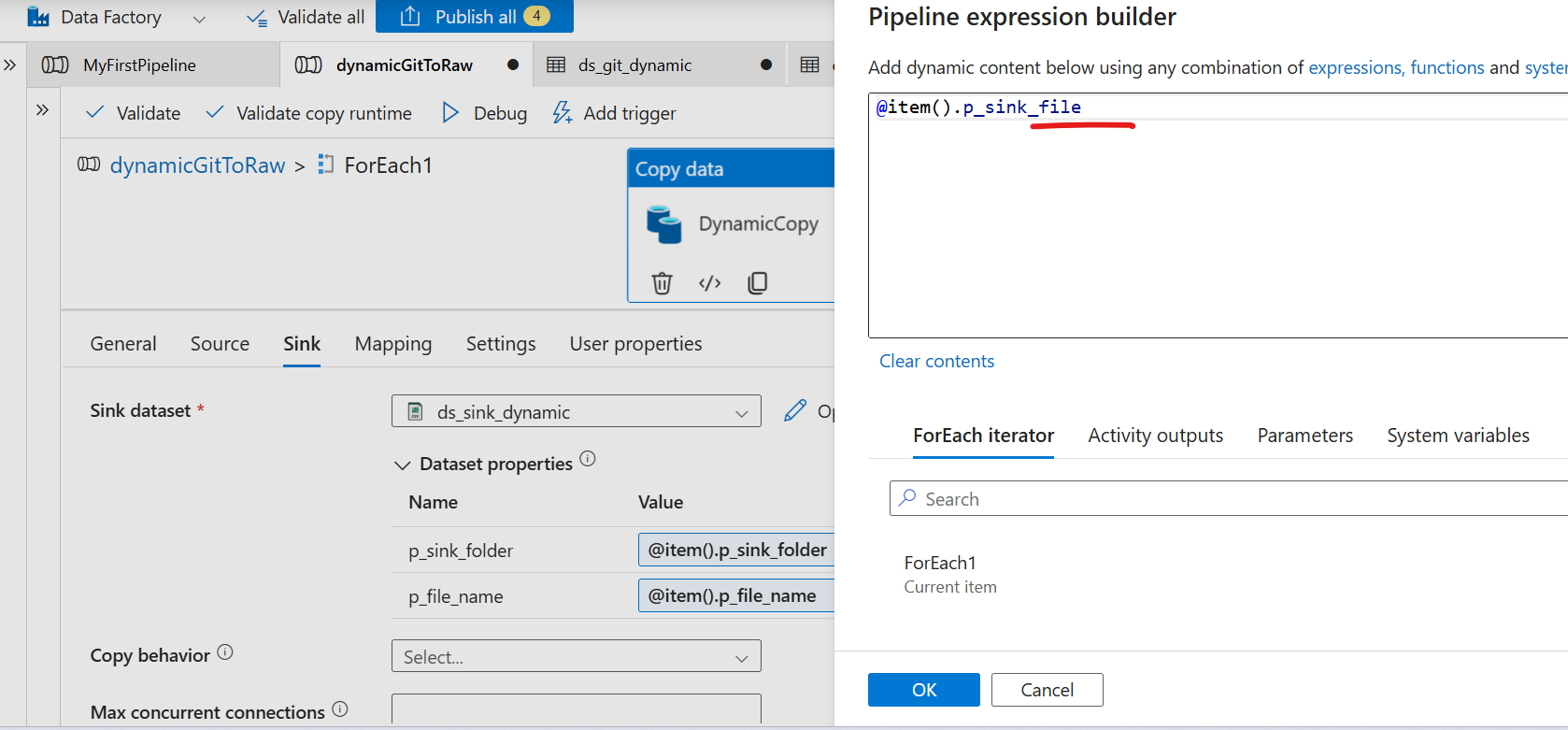
*Fetch control/config values to drive dynamic behavior in pipelines.*

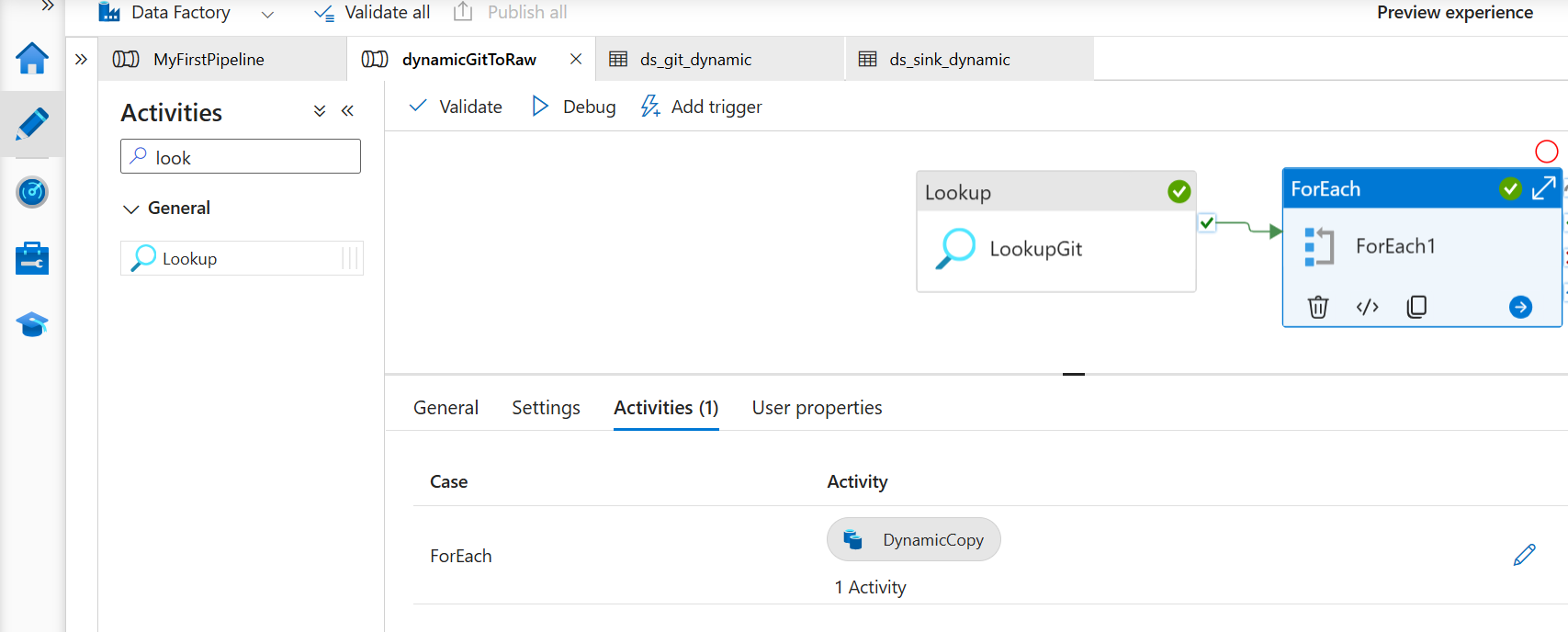
* **Use Cases:**
  + Fetch **last run timestamp** from a control table or config file
  + Read **authorization tokens** or API keys from Key Vault/config file
  + Retrieve **list of countries/dates** for iteration using ForEach
* **Connected To:**
  + SQL table / Azure Blob / ADLS file with lookup logic

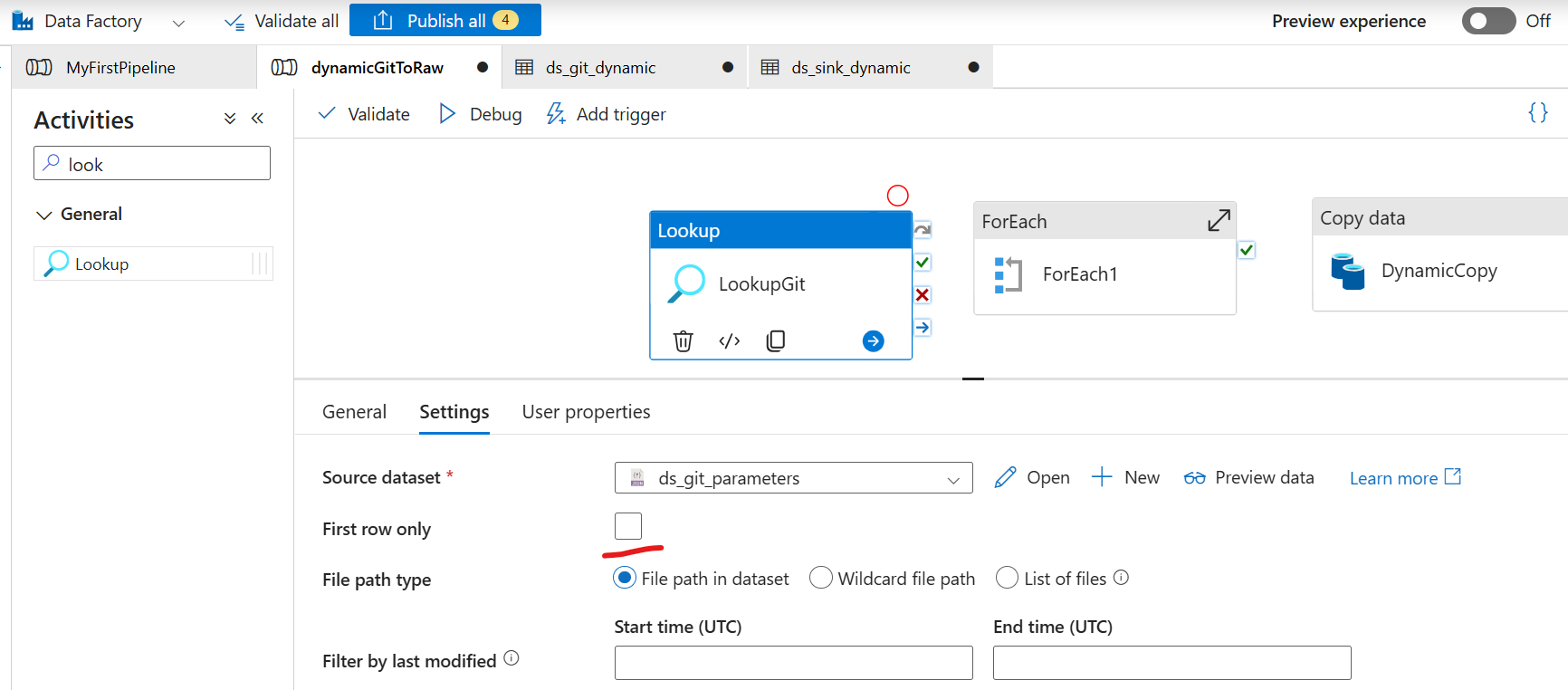






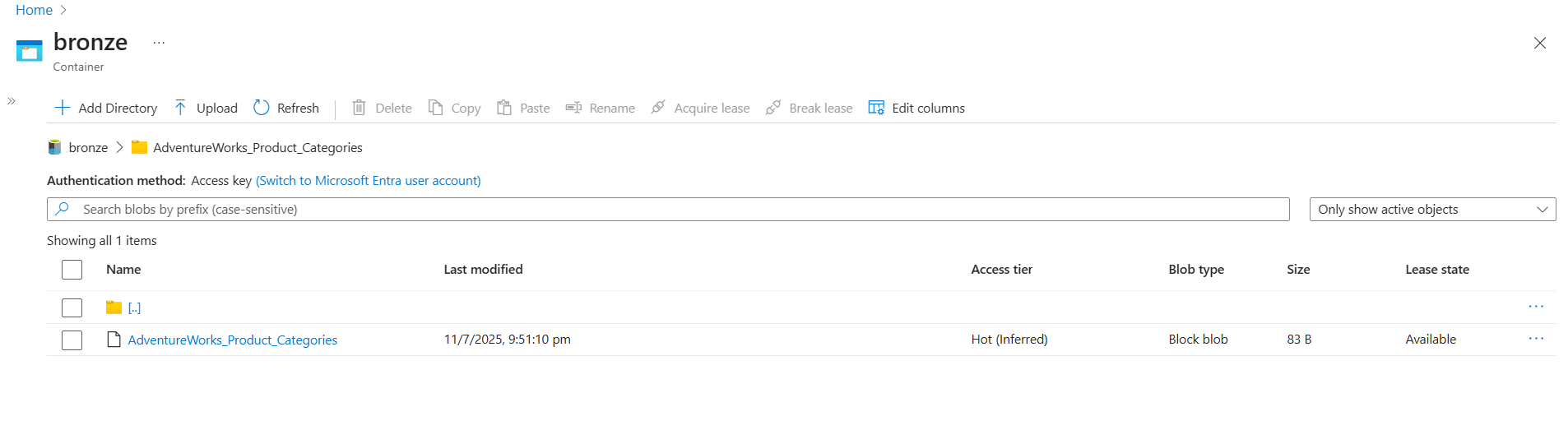


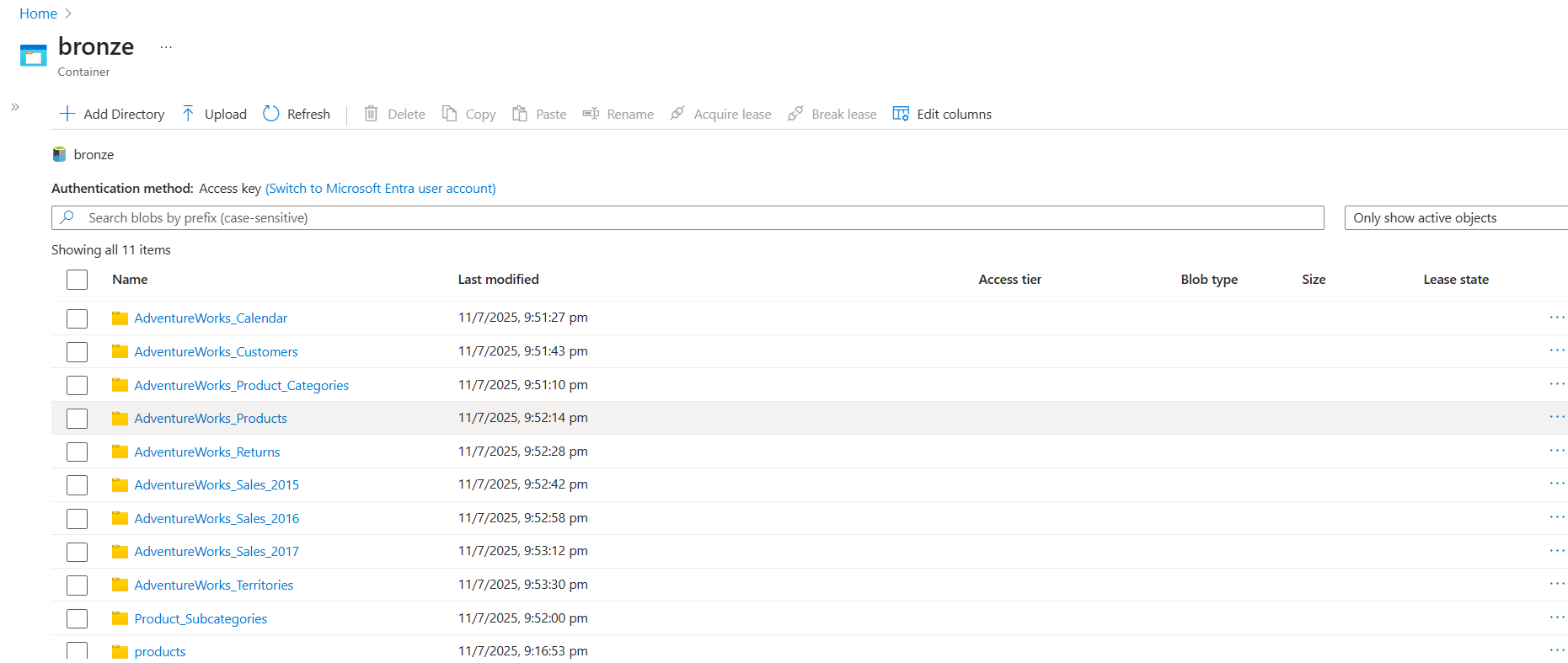




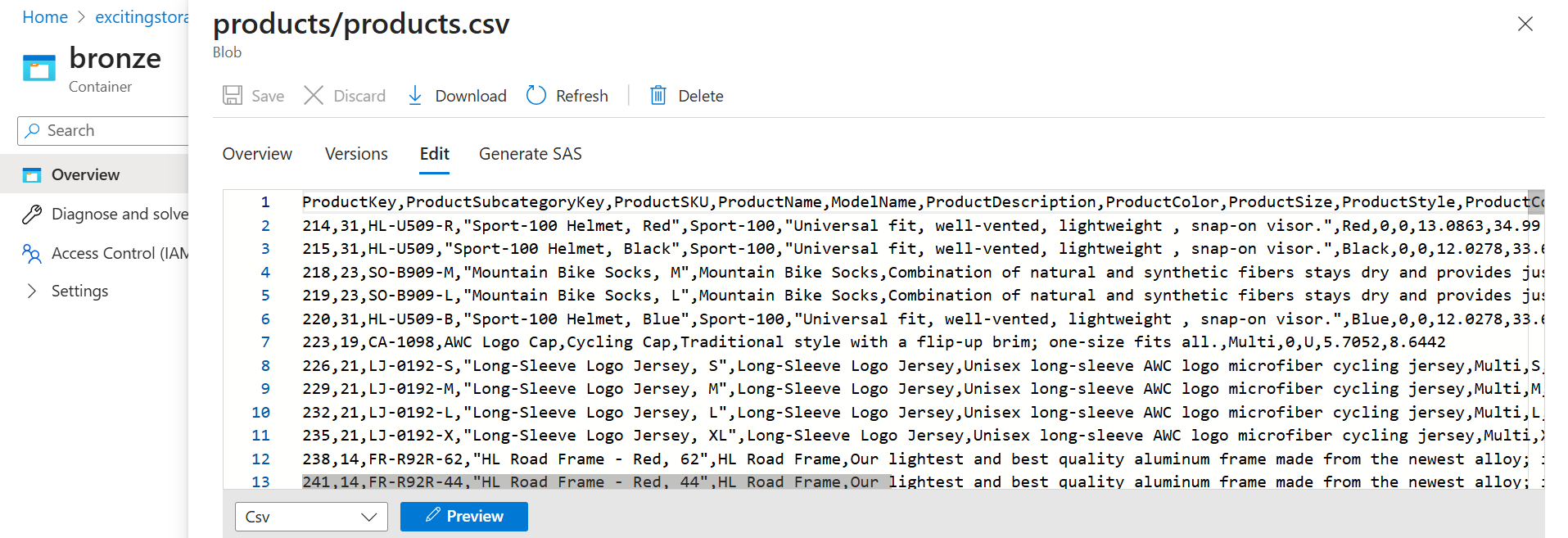
### Output:

Statically file placing:





### 



### 🧩 6. Schema Mapping & File Versions

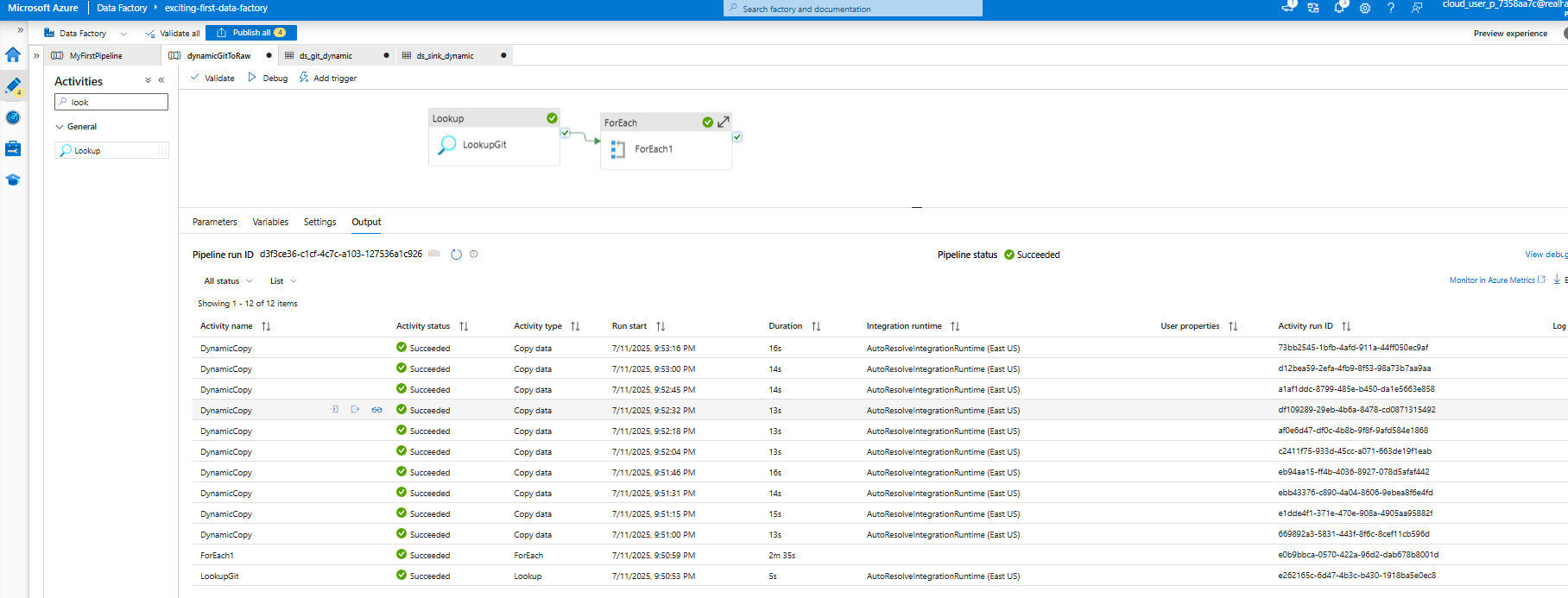
*Maintaining structure and separating raw vs processed data helps traceability and audit.*

* **Mapping Logic:**
  + Set mappings manually or allow auto-mapping based on field names
  + Handle **nulls**, **type casting**, and **field order mismatches**
* **File Versions:**
  + **Raw file:** Direct dump from API
  + **Processed file:** Cleaned/validated and schema-aligned file

### 🕒 7. Triggers & Scheduling

*Automate the pipeline to run at specific times or events.*

* Used **Time-based triggers**:  
  + Daily at specific time
  + Hourly (for frequently changing data)
  + Tumbling Window (if you need retry & dependency handling)
* Configured **timezone**, **recurrence**, and **start/end time**

****

### 🛡 8. Monitoring, Logging & Error Handling

*Ensure robustness and visibility into pipeline health.*

* **Retry Policies:**
  + Configured in Copy/Lookup activity for transient failures
* **Monitoring:**
  + Enabled **Activity Run Logs** and **Pipeline Run Logs**
  + Set up alerts using **Azure Monitor** (email/webhook notifications)
* **Failure Handling:**
  + Used **IF Condition / Set Variable / Email notification** on failure
  + Logged errors and pipeline status to a dedicated ADLS folder or SQL control table

### 🧰 Tech Stack Used

| **Component** | **Purpose** |
| --- | --- |
| Azure Data Factory | Orchestration, pipeline management |
| REST API | Source system for static/dynamic data |
| Azure Data Lake Gen2 | Data sink for storing raw/processed files |
| CSV / JSON / Parquet | File formats used for input/output |
| Copy Activity | Core data transfer logic |
| Lookup Activity | Read control/config values dynamically |
| Parameters & Expressions | Dynamic pipeline design |
| Azure Monitor | Alerts, pipeline monitoring |
| Triggers | Automated pipeline execution |

### 📌 Summary

This project reinforced key ADF skills such as:

* Dynamic pipeline development using parameters
* REST API integration with data mapping
* Best practices in schema management
* Monitoring and automation using triggers & alerts