Bonus Question

---

1. Why is storing cleaned data in Azure Blob Storage important for real-time pipelines?

- Raw data is usually messy (duplicates, missing values, inconsistent formats).
- Real-time systems need **clean, reliable, and consistent input**; otherwise, dashboards, ML models, or reports will be wrong.
- By storing the **cleaned version** in Blob Storage, downstream systems (Databricks, Synapse, Power BI, Azure ML, etc.) can access **ready-to-use data**.
- Blob Storage acts as a **centralized data lake** that is:
    - **Scalable** (handles huge volumes of data)
    - **Durable & cost-effective** (safe long-term storage)
    - **Accessible in real-time** for batch jobs and streaming pipelines.

☞ In short: **Blob Storage ensures a single source of truth for real-time analytics.**

---

2. What's the difference between pipeline artifacts and Blob Storage uploads?

- **Pipeline Artifacts**
    - Temporary files generated by the Azure DevOps pipeline.
    - Used to **pass outputs between jobs/stages** inside the pipeline.
    - Typically short-lived (deleted when pipeline history expires).
    - Scope = DevOps environment only.
- **Blob Storage Uploads**
    - Permanent storage in the cloud.
    - Accessible to **external systems, teams, or applications** beyond DevOps.
    - Used for **long-term retention, data sharing, and integration**.
    - Scope = global (any authorized app can fetch it).

☞ Think of artifacts as **"pipeline-internal handoff"**, and Blob Storage as **"external long-term storage."**

---

3. How would you handle failures in file uploads in a production setup?

In production, uploads might fail due to **network issues, wrong credentials, quota limits, or transient Azure errors**. Best practices:

1. **Retry with exponential backoff** (e.g., retry after 2s, 4s, 8s…).
2. **Graceful error handling** → log the error, don't just crash.
3. **Monitoring & Alerts** → send logs to Azure Monitor / Application Insights, trigger alerts if upload fails.

4. **Fallback storage** → move failed files to a **quarantine container** for manual inspection.
5. **Security best practices** → use SAS tokens or Managed Identity instead of storing raw keys.
6. **Idempotency** → allow overwriting (`overwrite=True`) so retries don't create duplicate blobs.