### **1. Why is data cleaning important in real-time data processing?**

* **Accuracy:** Raw data often contains duplicates, missing values, or errors. Cleaning ensures that only valid and accurate records are processed.
* **Consistency:** Standardizes formats (e.g., dates, currencies) so downstream systems can process data without mismatches.
* **Performance:** Reduces the volume of unnecessary or noisy data, improving processing speed and efficiency.
* **Reliability:** Cleaned data ensures dashboards, alerts, and machine learning models give trustworthy results.
* **Compliance:** Helps meet data governance, privacy, and quality requirements by filtering out invalid or sensitive information.

Without cleaning, real-time insights may be misleading, causing poor business decisions.

### **2. What are pipeline artifacts and how are they used in DevOps workflows?**

* **Definition:** Pipeline artifacts in Azure DevOps are files or packages produced during a pipeline run (build/test/release) that can be shared across stages or pipelines.
* **Usage in DevOps workflows:**
  + **Build Stage:** Application code is compiled into executables, libraries, or Docker images, saved as artifacts.
  + **Test Stage:** Test reports, logs, or coverage files are generated as artifacts for later review.
  + **Release Stage:** Artifacts (e.g., packaged apps) are deployed to environments such as Dev, QA, or Production.
  + **Traceability:** Artifacts ensure consistent deployment by preserving the exact version of the code built and tested.

In short, artifacts are **pipeline-generated outputs** that enable repeatability and consistency across DevOps workflows.

### **3. How would you modify the pipeline to store the cleaned data into Azure Blob storage?**

* **Step 1 – Add a Data Cleaning Step:** Ensure the pipeline processes raw input data and generates a cleaned dataset (CSV/JSON/Parquet).

* **Step 2 – Configure Azure Blob Storage Connection:**
  + Use an Azure Service Connection in DevOps to authenticate securely with Blob Storage.
  + Store connection strings/keys in Azure Key Vault or DevOps secrets.

* **Step 3 – Upload Task in Pipeline:**

* + Add an Azure CLI or Azure PowerShell task to the pipeline.

Command example (CLI):  
  
 az storage blob upload \

--account-name <storage\_account> \

--container-name <container> \

--file cleaned\_data.csv \

--name cleaned/cleaned\_data.csv

* **Step 4 – Error Handling:** Implement retry logic and log upload failures.
* **Step 5 – Verification:** Optionally, add a validation step to confirm that the file exists in Blob storage.

This modification ensures that **cleaned data is stored durably in Azure Blob Storage**, making it accessible for analytics, machine learning, or downstream pipelines.