A diagram of a silver and silver and silver and silver and silver and silver and silver and silver

AI-generated content may be incorrect.

**🧠 Mock Interview Q&A: CVS Pipeline (Databricks + GitLab + Unity Catalog)**

**🔹 General / High-Level Questions**

**❓ Q1: Can you walk us through the overall pipeline design?**

✅ *Answer Tip*: Use your draw.io diagram to describe source → bronze → silver → gold, highlighting tools used at each stage (Autoloader, PySpark, Delta, Unity, GitLab CLI).

**❓ Q2: What problem does this pipeline solve?**

✅ *Answer Tip*: “We needed to unify multiple budget, ERP, and treasury datasets into trusted Delta tables for financial analysis in QLIK. Each source had different formats, rules, and update frequencies, so the pipeline needed to be modular, automatable, and secure.”

**🔸 Ingestion & Autoloader**

**❓ Q3: Why did you choose Autoloader? What options did you configure?**

✅ *Answer Tip*:

* Autoloader handles new file detection, schema evolution, and scaling.
* Key options used:
  + cloudFiles.format = csv
  + cloudFiles.schemaLocation = /mnt/schemas/fastbook
  + badRecordsPath = /mnt/errors
  + checkpointLocation = /mnt/checkpoints/...
  + includeExistingFiles = true (for historical backfill)

**❓ Q4: How do you handle Excel files in Autoloader?**

✅ *Answer Tip*: "Autoloader doesn’t natively support .xlsx, so we automate the conversion to CSV using a Python process before ingesting them. FAST Book and R/M/P/O files follow this path."

**❓ Q5: What triggers the ingestion?**

✅ *Answer Tip*: "Some sources like IQUERY are manually triggered. FAST Book is automated via a scheduled download to Azure Blob. For GitLab-based logic files, we plan to implement file notification using Autoloader’s cloud events in the future."

**🔸 Transformation (Bronze → Silver)**

**❓ Q6: What kind of transformations do you perform in the silver layer?**

✅ *Answer Tip*:

* Deduplication (e.g., using row hash or surrogate keys)
* Schema standardization (aligning TAS, fund codes)
* Business logic application via SME-uploaded file
* Refactored legacy pandas code to scalable PySpark functions

**❓ Q7: Why did you refactor from pandas to PySpark?**

✅ *Answer Tip*: “Pandas couldn’t scale well in our distributed environment. We needed parallel processing for large budget files, and also wanted native Delta Lake support and integration with Unity Catalog.”

**🔸 Delta Lake & Gold Layer**

**❓ Q8: How do you ensure quality before writing to the gold layer?**

✅ *Answer Tip*:

* Used **Delta Expectations** (e.g., expect(column IS NOT NULL))
* Added audit columns (ingest\_time, source\_file)
* Implemented schema checks and versioned logic

**❓ Q9: What format is used in gold and how is it accessed?**

✅ *Answer Tip*: “Gold layer is stored as partitioned Delta tables, Z-Ordered where applicable, and served to QLIK via Databricks SQL endpoints.”

**🔸 Governance: Unity Catalog**

**❓ Q10: How does Unity Catalog fit into your pipeline?**

✅ *Answer Tip*: “All tables are registered in Unity Catalog with appropriate permissions set at schema/table level. It helps with lineage tracking, access control by persona, and makes it easy for downstream tools like DBSQL or QLIK to consume clean data.”

**🔸 CI/CD with GitLab**

**❓ Q11: How do you deploy your pipeline?**

✅ *Answer Tip*:

* Stored .py scripts and .job.json specs in GitLab repo
* Used GitLab CI/CD runner + Databricks CLI:

databricks jobs create --json-file jobs/bronze\_job.json

databricks jobs run-now --job-id 123

**❓ Q12: How do you handle testing and promotion between environments?**

✅ *Answer Tip*: “We validate code via linting and schema checks in GitLab. Future enhancements could include unit tests with pytest and use of environment-specific job JSON files or Workspace Paths.”

**🔸 Scenario-Based / Curveballs**

**❓ Q13: What if a file has a new column not present in the schema?**

✅ *Answer Tip*: “We store schemas using cloudFiles.schemaLocation, so Autoloader detects changes. For controlled evolution, we set mergeSchema = true and alert when breaking changes occur.”

**❓ Q14: What challenges did you face?**

✅ *Answer Tip*: Mention:

* Excel parsing limitations
* Memory issues with pandas
* Manual file coordination at early stages
* Refactoring logic to fit Spark’s distributed model

**❓ Q15: How do you monitor or debug failures?**

✅ *Answer Tip*:

* Autoloader bad records path
* Logging file names and row counts
* Checkpointing + Delta time travel for replay/debug