**🚀 Data Engineering Interview Notes (Workflow Story Format)**

**1. Data Ingestion & Validation (Problem 3)**

* **Scenario:** Ingest CSV/Parquet/JSON into Bronze layer with schema validation & file checks.
* **Baseline approach:**
* **Advanced considerations:** (schema drift, bad records handling, retries, metadata logging)
* **Follow-ups:** Concurrency, handling corrupt files, schema evolution.

**2. Incremental Processing (Problem 4)**

* **Scenario:** Load only new files / partitions (Silver layer).
* **Baseline approach:**
* **Advanced considerations:** (watermarking, checkpointing, idempotency, streaming vs batch).
* **Follow-ups:** Late-arriving data, reprocessing old partitions, ensuring no duplicates.

**3. CDC Upserts with Audit (Problem 1)**

* **Scenario:** Daily CDC feed → Delta table upsert with audit.
* **Baseline approach:** Delta MERGE with conditional updates (only latest updated\_at).
* **Advanced considerations:** Idempotency, late events, schema evolution, audit log design.
* **Follow-ups:** Handling partial failures, scaling with large CDC, optimizing small files.

**4. Join Optimization & Skew Handling (Problem 2)**

* **Scenario:** Events (1B) join Users (1M), but skew on user\_id=0.
* **Baseline approach:** Broadcast smaller table + salting for skewed keys.
* **Advanced considerations:** AQE, partition tuning, hybrid join strategy.
* **Follow-ups:** Choosing salt size, AQE interaction, separating hot keys.

**5. Orchestration & Monitoring (Problem 5)**

* **Scenario:** Build Airflow DAG for ingestion → processing → load.
* **Baseline approach:** Tasks for each stage, retries, alerts.
* **Advanced considerations:** Dynamic DAGs, SLAs, sensors, external triggers, lineage tracking.
* **Follow-ups:** Backfilling, CI/CD for DAGs, handling dependent DAGs.

**6. Data Modeling & Warehousing (Problem 6)**

* **Scenario:** Design SCD, fact/dimension tables, schema evolution.
* **Baseline approach:** Normalize data, star/snowflake schema, SCD Type 2 with surrogate keys.
* **Advanced considerations:** CDC + SCD merge, schema-on-read vs schema-on-write.
* **Follow-ups:** Denormalization trade-offs, handling historical corrections.

**7. Data Quality & Testing (Problem 7)**

* **Scenario:** Ensure data correctness before promoting to Silver/Gold.
* **Baseline approach:** Null checks, schema checks, row counts.
* **Advanced considerations:** Great Expectations, unit tests, anomaly detection.
* **Follow-ups:** Automating quality gates, alerting on failures.

**8. SQL Query Optimization (Problem 8)**

* **Scenario:** BigQuery/Redshift query is slow.
* **Baseline approach:** Use partitions, clustering, limit data scanned.
* **Advanced considerations:** Indexing, materialized views, query plans.
* **Follow-ups:** When to denormalize, how to debug skew in SQL joins.

**9. Cloud Storage & File Validation (Problem 9)**

* **Scenario:** Upload files to S3/GCS with validation.
* **Baseline approach:** Validate schema, checksum, quarantine bad files.
* **Advanced considerations:** Automation with Lambda/Cloud Functions, metadata tables.
* **Follow-ups:** Secure uploads, governance, encryption.

**10. Streaming & Real-Time Processing (Problem 10)**

* **Scenario:** Kafka → Spark Structured Streaming → Delta.
* **Baseline approach:** Read from Kafka, write to Delta with checkpoint.
* **Advanced considerations:** Exactly-once semantics, watermarking, late data handling.
* **Follow-ups:** Scaling consumers, schema evolution in streaming.

**11. Security & Governance (Problem 11)**

* **Scenario:** Sensitive PII data in data lake.
* **Baseline approach:** Masking, access control, IAM roles.
* **Advanced considerations:** Tokenization, encryption at rest + in transit.
* **Follow-ups:** GDPR/CCPA compliance, column-level security, auditing access.

**12. Debugging & Production Issues (Problem 12)**

* **Scenario:** Spark job fails OOM or skew.
* **Baseline approach:** Check DAG, optimize partitions, avoid wide transformations.
* **Advanced considerations:** Adaptive Query Execution, caching strategies, job retries.
* **Follow-ups:** Debugging with Spark UI, handling lineage corruption, recovery strategy.

✅ This structure lets you:

1. See the **whole workflow** at once (like a story).
2. Drill into **baseline → advanced → follow-ups** for each problem.
3. Use it as **flashcards / quick revision** before interviews.