Note: *As a first step we should review and re-write our requirements into a logical format we can use as a guide. This may also surface additional questions we need to know*

Legend:

- Instructional
- Project Requirement
- Technical Requirement
- DDL Object
- DML Logic
- Data Quality Check
- Challenges

Overview:

- This exercise evaluates your ability to design and implement a file-based ingestion + ELT pipeline in Snowflake with strong data quality checks, idempotency, and production-ready thinking.
- Time Expectation: 2–3 hours maximum.
- Submission: Please submit a single zip file or a Git repository containing:
  - SQL scripts (required)
  - A short README (required)
  - Optional: diagram or notes
- Constraints
  - Use Snowflake SQL only (stored procedures may be SQL-based or JavaScript-based; choose what you prefer)
  - Assume files arrive in AWS S3 and are accessed via a Snowflake external stage
  - You do not need to connect to a real S3 bucket; treat the stage as already available

Scenario

- A monthly dataset arrives as a pipe-delimited file (|), GZIP compressed, with a header row.
- Each month includes:
  - New records
  - Updated records (existing business keys may change field values)
- We want:
  - A RAW landing table (bronze)
  - A transformation layer (silver)

    - A TARGET table (gold)
    - A load history table for auditing
    - QC checks (row count, duplicates, null checks)
  - Data Model: Assume the incoming file has these columns:
    - reporting_month (YYYY-MM, string)
    - loan_id (string)
    - servicer_name (string)
    - balance (number)
    - interest_rate (number)
    - state (string)
    - updated_at (timestamp string)
    - Business Key
    - loan_id + reporting_month

Assignment Tasks

- Task 1: DDL: Provide Snowflake DDL scripts for:
  - RAW_LOAN_MONTHLY (bronze)
    - Include metadata columns: file_name, load_time, run_id
  - VW_LOAN_MONTHLY_CLEAN (silver)
    - Apply type casting and basic cleansing rules
  - TARGET_LOAN_MONTHLY (gold)
    - Designed for analytics and incremental updates
  - LOAD_HISTORY (audit table)
    - Capture run_id, start/end time, file_name, rows_parsed, rows_loaded, errors_seen, status, error_message

- Task 2: Load Script (Historical / Monthly Ingest) Write a SQL script that demonstrates how you would:
  - COPY INTO RAW_LOAN_MONTHLY from an external stage
  - Use an explicit FILE FORMAT (pipe-delimited, skip header)
  - Capture load metadata into LOAD_HISTORY
  - You may use placeholders like: @MY_EXT_STAGE/path/LOAN_MONTHLY_202601.gz

- Task 3: Incremental Merge into TARGET
  - Write a MERGE statement (or equivalent approach) that loads data from VW_LOAN_MONTHLY_CLEAN into TARGET_LOAN_MONTHLY using the business key loan_id + reporting_month.
  - Requirements:
    - Insert new records
    - Update existing records when values change
    - Ensure idempotency (running twice should not duplicate)

- Task 4: Data Quality Checks (QC) Provide SQL queries (or stored procedure logic) that perform the following checks per run_id:
  - Row count check
    - rows_loaded in RAW vs rows_processed into TARGET
  - Duplicate check
    - duplicates in RAW by business key
  - Null check
    - loan_id and reporting_month must not be null
  - Basic anomaly check (simple)
    - balance < 0 OR interest_rate < 0 OR interest_rate > 25
  - Write QC results to a table (you may create QC_RESULTS table) with: run_id, check_name, expected_value, actual_value, status, severity, sample_query

- Task 5: Operational Behavior (Short Answers). In README, answer these:
  - How do you prevent duplicate loading if the same file arrives twice?
  - How do you handle schema evolution if the file adds new columns?
  - What would you do if COPY INTO partially loads a file and then fails?
  - What warehouse sizing / file sizing guidance would you give for performance?
  -

Evaluation Criteria

- We will evaluate:
  - Correctness and clarity of SQL
  - Production readiness (idempotency, auditing, error handling)

    - Data quality mindset (QC coverage and structure)
    - Maintainability (clean naming, modular scripts, readable README)
    - Practicality (realistic and operable approach)

Bonus (Optional)

- Provide a simple runbook section in README: “How to run end-to-end”
- Provide a lightweight object dependency diagram