Step 1: Creating a Secure Staging Directory
Purpose: To temporarily store sensitive ETL staging files in a secure, access-controlled, and encrypted location.

Actions Taken:

- Created a locked-down folder: C:\Secure_ETL_Staging
- Removed inherited permissions and ensured that only:
    - SYSTEM
    - Administrators (you)
    - Your specific user account
      can access and modify contents.
- Applied folder encryption via Windows EFS (Encrypting File System).
- Confirmed that all future files added will automatically inherit encryption.

Step 2: **Setting Up Environment Variables**
Purpose: To securely store database credentials and key directory paths without hardcoding them into scripts, ensuring maintainability and security.

Actions Taken:
• Created a .env file inside the secured directory: C:\Secure_ETL_Staging
• Added placeholders for:

- MYSQL_HOST
- MYSQL_PORT
- MYSQL_USER
- MYSQL_PASS
- MYSQL_DB
  • Defined directory paths for:
- STAGING_DIR (staging area)
- PROCESSED_DIR (processed data area)
- LOG_DIR (log storage)
  • Ensured sensitive credentials are not hardcoded in scripts — they will be accessed programmatically via environment variables.
  • Verified file save and presence in the correct locked-down location.
  • Compliant with security principles for secure credential handling and directory referencing.

Step 3: Logging System Setup
Purpose: To ensure every ETL activity, error, or event is recorded for traceability, debugging, and security audits.

Actions Taken:
• Created logging_config.py inside the config folder.
• The logger is configured to:
— Write logs to etl_pipeline.log inside C:\Secure_ETL_Staging\logs.
— Rotate logs daily at midnight.
— Keep the last 7 days of logs for reference.
— Capture timestamps, log levels, and messages for complete context.
• Successfully tested the logger using test_logger.py.
• Verified automatic log creation and rollover settings.

Step 4: Input Validation Setup

Purpose:
To ensure that every incoming CSV file follows the expected schema before processing, preventing garbage data from being loaded into MySQL and maintaining data integrity.

Actions Taken:
• Created input_validator.py inside the utils folder.
• The validator script:
— Verifies that the incoming file exists.
— Checks the header structure against predefined column names.
— Raises descriptive errors if headers do not match or file is missing.
— Provides console feedback upon successful structure validation.
• Integrated a testing block at the end of the file to enable quick validation of the validator itself.
• Compliant with security and best practices by not allowing incorrect or malformed data to pass through.

Step 5: Data Sanitization Before Loading into SQL
Purpose:
To prevent SQL injection attacks, remove special characters, and ensure only clean, valid data is inserted into MySQL tables.
Actions Taken:

- Used Python's replace () and strip() methods to clean CSV values of unwanted commas, percentage signs, and extraneous spaces.
- Ensured all numeric columns are cast to float or int types before database insertion.
- Validated all strings to ensure no unexpected characters or SQL statements are present.
- Verified through log messages and tests that no malformed data reaches the SQL layer.
- Aligned this step with best practices in secure data engineering.

Step 6: Automated Scheduling
Purpose:

To automate the ETL process, ensuring it runs on the 3rd of every month at a set time without manual triggers, maximizing reliability and consistency.

Actions Taken:

• Opened PowerShell as Administrator.

• Navigated to the secure ETL staging directory:

```
cd C:\Secure_ETL_Staging
```

• Used Windows Task Scheduler via schtasks to register an automated job:

```
schtasks /Create /SC MONTHLY /D 3 /TN "Secure_Shopify_ETL_Monthly" /TR "python -m scripts. shopify_etl_pipeline" /ST 13:00 /F /RU SYSTEM
```

• Verified the successful creation of the scheduled task:

```
schtasks /Query /TN "Secure_Shopify_ETL_Monthly" /V
```

• Confirmed the task is set to run as SYSTEM, with next run date on the 3rd of every month at 1:00 PM.

• Locked the task to SYSTEM mode to avoid accidental edits or unauthorized triggers.

Step 7: Data Sanitization and Maintenance Cleanup

Purpose:

To ensure only clean, validated data is loaded into the SQL database, while maintaining ongoing file and log hygiene through automated archival and cleanup.

Actions Taken:

• Implemented Python-based sanitization logic in all ETL scripts using replace(), strip(), and typecasting to convert values into safe, valid formats.

• Stripped unwanted characters (commas, special symbols, percentage signs) and ensured no direct SQL string injections.

• Applied strict typecasting for numeric values (float/int).

• Built cleanup_archival.py to automatically archive processed CSVs older than 7 days into an archives folder.

• Created log_cleanup.py placeholder for future log file cleanup automation.

• Scheduled both cleanup scripts using Windows Task Scheduler:

– Archive cleanup runs weekly.

– Log cleanup scheduled monthly on the 15th at 3:00 AM.

• Tested ETL execution after sanitization and verified stable data insertions without SQL errors or malformed entries.

• Verified automation reliability by reviewing log outputs and archival movement.