Google Ads ETL Pipeline Setup

Step 1: Secure Staging Directory Setup for Google Ads

Purpose:

To store API-fetched reports temporarily before loading into SQL in a secure, encrypted location.

Actions Taken:

• Created sub-folder: C:\Secure_ETL_Staging\google_ads_data.

• Confirmed folder inherits locked-down permissions and encryption from parent directory.

Step 2: Environment Variables Setup

Purpose:

To store Google Ads API credentials securely without hardcoding.

Actions Taken:

• Updated .env file to include:

- GOOGLE_CLIENT_ID
- GOOGLE_CLIENT_SECRET
- GOOGLE_REFRESH_TOKEN (acknowledging it doesn't expire unless revoked)
- GOOGLE_DEVELOPER_TOKEN
- GOOGLE_CUSTOMER_ID
  • Validated that sensitive data is never stored directly in scripts.

Step 3: Logging Setup

Purpose:

To maintain traceability of each API call and ETL activity.

Actions Taken:

• Planned centralized logging in C:\Secure_ETL_Staging\logs.

• Will auto-generate google_ads_etl.log using existing logging_config.py.

Step 4: Data Validation & Structure Handling

Purpose:

To ensure clean, schema-consistent data from Google Ads API before SQL load.

Actions Taken:

• Inline JSON structure validation in-memory for each API response.

• Used fallback defaults for missing fields.

• Typecasting of spend, conversions, and cost fields to ensure schema consistency.
• Inline fallback defaults for missing metrics.
• Strict API response validation with logging of row counts and metric totals.

Step 5: Table Creation in Secure Database
Purpose:
Store cleaned weekly metrics in MySQL for reporting and analysis.
Actions Taken:
• Created table google_ads_campaigns_fact inside shopify_etl.
• Schema defined with appropriate datatypes, primary key on campaign_id + start_date + end_date.
• Set etl_user permissions for SELECT, INSERT, UPDATE only (delete restricted for safety).

Step 6: Secure ETL Script Development
Purpose:
Automate end-to-end API call, validation, transformation, and SQL load.
Actions Taken:
• Developed googleads_etl_pipeline.py mirroring Shopify ETL structure.
• Used secure credential loading from .env.
• Dynamic client selection under MCC.
• Inline data correction for start_date, end_date if API output differs.
• Added structured logging.

Step 7: Automated Scheduling
Purpose:
Ensure hands-free weekly execution.
Actions Taken:
• Registered Windows Task using schtasks.
• Set up to run every Monday at 1:00 PM securely.
• Verified execution and logs.

Step 8: Automated Logs Cleanup
Purpose:
To automatically manage log files and avoid storage bloat.
Actions Taken:
• Created googleads_log_cleanup.py under utils.
• Configured to delete logs older than 60 days.

• Scheduled monthly execution on the 1st of every month at 3:00 PM via Windows Task Scheduler.
• Verified execution and logs.