**✅ Hybrid Offline + GitHub Workflow for Running Local LLM Inference**

**This workflow enables autonomous daily data collection, LLM-powered cleaning using a quantized offline model (GGUF) on a local machine, and clean dataset updates committed back to GitHub. It balances free infrastructure, model size limitations, and local LLM power using a semi-automated hybrid approach.**

**🔧 Phase 0: Environment & Tooling Setup**

**📁 Project Structure**

**Assume a GitHub repository with the following layout:**

**graphql**

**CopyEdit**

**Car\_Clinic\_Project/**

**├── models/**

**│ └── TheDataCleaningModel/ # GGUF model and tokenizer**

**├── data/**

**│ ├── raw/ # Raw Reddit CSVs**

**│ └── cleaned/ # Cleaned outputs**

**├── python\_scripts/**

**│ └── reddit\_data\_cleaner/**

**│ └── llm\_cleaner.py # Your llama.cpp-powered LLM pipeline**

**├── .github/**

**│ └── workflows/**

**│ └── ingest\_data.yaml # GitHub workflow to collect & push raw CSVs**

**🧰 Tools Used**

* **GitHub + GitHub Actions: For data ingestion & versioning**
* **llama-cpp-python: For offline inference using a quantized GGUF model**
* **Local Scheduler (e.g., Task Scheduler or cron): For automating local processing**
* **Optional Notifier (email/Discord/IFTTT/Pushbullet): For device activation alerts**

**🔁 Phase 1: Raw Data Ingestion on GitHub**

**📦 What Happens**

* **A GitHub Action runs daily (or manually) to scrape and push new data from Reddit (or your source) into data/raw/YYYY-MM-DD.csv.**

**📤 Output**

* **New file in data/raw/ named after the ingestion date.**

**📍 Purpose**

* **Ensures data is collected consistently, even if your local machine is OFF.**

**💤 Phase 2: Waiting for Device Wake-Up**

**🕐 Situation**

* **Since the LLM cannot run on GitHub (due to model size), processing must wait until your local machine is turned ON.**

**🛎️ Solution: Notification Trigger**

* **At the end of the ingestion GitHub Action, send a notification (one or more of the following):**
  + **Push notification (e.g., Pushbullet)**
  + **Discord webhook**
  + **IFTTT automation**
  + **Gmail API or SMTP alert**
  + **Slack notification**

**Example Trigger (in GitHub Action):**

**yaml**

**CopyEdit**

**- name: Send Wake-up Notification**

**run: curl -X POST -H "Content-Type: application/json" \**

**-d '{"value1":"🚨 New Reddit data is ready for cleaning"}' \**

**https://maker.ifttt.com/trigger/new\_data\_ready/with/key/<<YOUR\_IFTTT\_KEY>>**

**⌛ Wait Logic**

* **LLM processing is deferred until the machine boots up and the scheduled job runs.**

**💻 Phase 3: Local Processing with LLM (Offline)**

**🏗️ Trigger**

* **Set up Task Scheduler (Windows) or cron (Linux/Mac) to run a local script (e.g., run\_cleaning.bat or run\_cleaning.sh) at boot or at a fixed time daily.**

**🔁 Flow**

1. **Pull latest Git repo.**
2. **Scan data/raw/ for unprocessed files (use a "done" list or match with data/cleaned/).**
3. **Use llama-cpp-python to:**
   * **Load GGUF model (TheDataCleaningModel/)**
   * **Read each CSV file row**
   * **Apply the structured prompt**
   * **Parse LLM output into structured JSON**
   * **Save cleaned output as CSV in data/cleaned/**

**Example Command:**

**bash**

**CopyEdit**

**python python\_scripts/reddit\_data\_cleaner/llm\_cleaner.py \**

**--model\_dir models/TheDataCleaningModel \**

**--input\_file data/raw/2025-07-31.csv \**

**--output\_file data/cleaned/2025-07-31.cleaned.csv**

**📤 Phase 4: Commit & Push Cleaned Data**

**💾 What Happens**

* **After successful processing:**
  + **Git add + commit cleaned CSV files**
  + **Push them to GitHub using git CLI**

**🧠 Tip:**

**Use a cleaned\_log.txt or check for file presence to avoid re-processing the same file.**

**Git Push Example:**

**bash**

**CopyEdit**

**git pull origin main**

**git add data/cleaned/2025-07-31.cleaned.csv**

**git commit -m "Cleaned data for 2025-07-31 via offline LLM"**

**git push origin main**

**📆 Phase 5: Automation & Monitoring**

**🛠️ Schedule Daily Job**

**Windows Task Scheduler:**

* **Trigger: Daily at 8:00 AM or at system boot**
* **Action: Run a .bat file that pulls, cleans, and pushes data**

**Linux/macOS cron job:**

**bash**

**CopyEdit**

**0 8 \* \* \* /home/user/projects/Car\_Clinic\_Project/run\_cleaning.sh**

**🔔 Optional: Post-cleaning Notification**

* **Send notification back that cleaning succeeded.**
* **Use Pushbullet, Discord, or IFTTT.**

**🚫 Edge Case Handling**

| **Scenario** | **Strategy** |
| --- | --- |
| **Device is off for multiple days** | **Backlog is stored in data/raw/, all files are processed once device turns on** |
| **LLM crashes** | **Log errors, skip bad rows, don’t push incomplete CSVs** |
| **No new raw files** | **Skip processing for the day, sleep** |
| **Device storage full** | **Add disk space alerts or prune logs/files** |

**📌 Summary of All Phases**

| **Phase** | **Component** | **Trigger** | **Outcome** |
| --- | --- | --- | --- |
| **0** | **Setup** | **One-time** | **Tools, folders, model downloaded** |
| **1** | **Data Ingestion** | **GitHub Action (daily)** | **New CSV pushed to data/raw/** |
| **2** | **Waiting** | **Passive** | **Notification sent when new data is ready** |
| **3** | **Cleaning** | **Scheduled on boot/daily** | **Run llama-cpp offline, generate cleaned data** |
| **4** | **Push** | **Post-processing** | **Cleaned data pushed to GitHub** |
| **5** | **Automation** | **One-time setup** | **Ensures daily autonomous operation** |
| **X** | **Error handling** | **N/A** | **Detect skips, duplicates, crashes** |

**Would you like me to generate:**

* **A full llm\_cleaner.py template?**
* **A .bat or .sh script for local automation?**
* **A GitHub Action template that sends the notification?**

**Second more comprehensive roadmap that include more error handling:  
✅ Final Version: Hybrid Offline + GitHub Workflow (With Integrated Error Handling)**

**This version is production-grade, modular, and fully comprehensible to any LLM, engineer, or orchestrator. All edge cases are not only handled but also tracked and logged.**

**📂 Folder Structure**

**bash**

**CopyEdit**

**Car\_Clinic\_Project/**

**├── models/**

**│ └── TheDataCleaningModel/ # Quantized GGUF model**

**├── data/**

**│ ├── raw/ # Raw daily Reddit CSVs**

**│ └── cleaned/ # Cleaned results**

**├── python\_scripts/**

**│ └── reddit\_data\_cleaner/**

**│ └── llm\_cleaner.py # Core offline inference pipeline**

**├── logs/**

**│ └── cleaner\_log.txt # Logs + error traces**

**├── .github/**

**│ └── workflows/**

**│ └── ingest\_data.yaml # GitHub ingestion workflow**

**├── run\_cleaning.bat or run\_cleaning.sh # Local runner script**

**🧩 Phase-by-Phase Breakdown**

**🔧 Phase 0: Environment & Tooling Setup**

**Purpose: Bootstrap the environment**

* **✅ Install llama-cpp-python**
* **✅ Install Git CLI**
* **✅ Install notification system (e.g., Pushbullet, Discord webhook, Gmail SMTP, or IFTTT)**
* **✅ Set up GitHub repo locally**
* **✅ Place GGUF model in models/TheDataCleaningModel/**

**🔁 Phase 1: Raw Data Ingestion on GitHub**

**🔄 What Happens**

* **GitHub Action runs daily to collect and push raw Reddit data into data/raw/YYYY-MM-DD.csv.**

**✅ Added Error Handling**

| **Scenario** | **Strategy** |
| --- | --- |
| **❌ No new data fetched** | **Log the day and exit cleanly** |
| **✅ Notification still triggered** | **So local system can still check if backlog exists** |

**🕐 Phase 2: Wake-up + Notification System**

**⏰ Added Behavior**

* **GitHub Action sends a notification to your local device (via Pushbullet, Discord, IFTTT, etc.) saying:**

**🚨 “New Reddit data is available at data/raw/ - Ready for LLM Cleaning.”**

* **Notification is always sent, even if no new data, to allow the local runner to check for backlog.**

**🖥️ Phase 3: Local LLM Cleaning (Core Pipeline)**

**This phase is now split into structured sub-phases with integrated error-handling hooks.**

**🧩 Phase 3.1: Local Boot Scheduler**

**🔁 What Happens**

* **On system boot or scheduled time, Task Scheduler / cron:**
  + **Pulls latest GitHub repo.**
  + **Logs device boot timestamp.**
  + **Checks for disk space.**
  + **Checks for new raw files (or backlog).**

**✅ Added Error Handling**

| **Scenario** | **Strategy** |
| --- | --- |
| **📴 Device was off for days** | **Iterate through all unprocessed files in data/raw/** |
| **💾 Disk nearly full** | **Halt execution and log "Disk space low. Skipping cleaning."  Optional: Send alert/notification** |

**🧠 Phase 3.2: Data Selection & Deduplication**

* **Loop through data/raw/ and check for cleaned counterpart in data/cleaned/.**

**python**

**CopyEdit**

**for raw\_file in raw\_dir.glob("\*.csv"):**

**if not (cleaned\_dir / f"{raw\_file.stem}.cleaned.csv").exists():**

**process\_file(raw\_file)**

**✅ Added Error Handling**

| **Scenario** | **Strategy** |
| --- | --- |
| **✅ Already processed** | **Skip automatically** |
| **🔁 Multiple unprocessed files** | **Process them one by one in loop** |
| **📂 data/raw/ empty or all files processed** | **Log and exit gracefully with message: "No unprocessed files found."** |

**🧼 Phase 3.3: LLM Cleaning Execution**

* **Load model using llama-cpp-python + GGUF.**
* **Read row-by-row and feed into model.**
* **Save to temp\_cleaned.csv.**

**python**

**CopyEdit**

**try:**

**result = run\_llm(prompt)**

**except Exception as e:**

**logger.error(f"Failed to process row {i}: {e}")**

**continue**

**✅ Added Error Handling**

| **Scenario** | **Strategy** |
| --- | --- |
| **💥 LLM fails on a row** | **Log the error and skip that row** |
| **💣 LLM model load fails** | **Exit immediately and log error to error\_log.txt** |
| **🧪 CSV has malformed data** | **Skip row, log issue, continue safely** |
| **⚠️ Model crashes entirely** | **Retry max N times, log, exit cleanly** |

**✅ Phase 3.4: Atomic File Writing & Validation**

* **After successful run, write output as:  
  data/cleaned/YYYY-MM-DD.cleaned.csv**
* **Delete temp file (temp\_cleaned.csv) only after successful commit.**

**💾 Phase 4: Git Commit + Push**

**🔁 Flow**

**bash**

**CopyEdit**

**git pull origin main**

**git add data/cleaned/YYYY-MM-DD.cleaned.csv**

**git commit -m "Cleaned data for YYYY-MM-DD"**

**git push origin main**

**✅ Added Error Handling**

| **Scenario** | **Strategy** |
| --- | --- |
| **⛔ Git push fails** | **Retry up to 3 times, else save cleaned file locally for future push** |
| **📶 No internet** | **Save push command to a pending\_pushes.log to retry on next run** |

**📆 Phase 5: Automation Setup**

***Same as before — Windows Task Scheduler or cron to trigger at boot/daily.***

**📣 Phase 6: Post-Cleaning Notification (Optional)**

* **Use Pushbullet / Email / IFTTT to notify:**

**✅ "Cleaning for YYYY-MM-DD complete and pushed."**

**✅ Added Error Handling**

| **Scenario** | **Strategy** |
| --- | --- |
| **📡 Notification fails** | **Save message to local retry log** |
| **❌ Notification service rate-limited** | **Back off and retry next cycle** |

**📋 Phase 7: Error & Health Logging System**

**🧾 Files to Maintain**

* **logs/error\_log.txt: for LLM failures, disk issues, malformed rows**
* **logs/device\_health.log: for memory/disk checks, boot time, scheduler runs**
* **logs/pending\_pushes.log: in case of offline Git push**
* **logs/processed\_dates.log: optional, just to track historical runs**

**🔚 Final Flow Overview (Updated)**

| **Phase** | **Description** | **Trigger** | **Error Handling Summary** |
| --- | --- | --- | --- |
| **0** | **Setup** | **One-time** | **Ensure models, folders, scripts exist** |
| **1** | **Data Ingestion** | **GitHub Action** | **Logs "no data" case, still triggers notification** |
| **2** | **Notification** | **GitHub Action** | **Works even if data missing, avoids deadlocks** |
| **3.1** | **Device Wakeup** | **System Boot** | **Checks disk, logs health, loops backlog** |
| **3.2** | **File Selection** | **Auto-scan data/raw/** | **Skips if already cleaned** |
| **3.3** | **LLM Cleaning** | **llama-cpp-python** | **Catches row, model, or file errors** |
| **3.4** | **Output Write** | **Local disk** | **Atomic save & fallback if push fails** |
| **4** | **Git Commit & Push** | **Local** | **Handles offline pushes with queueing** |
| **5** | **Automation Setup** | **One-time** | **Resilient to reboots, idempotent** |
| **6** | **Post-Cleaning Notify** | **Optional** | **Notification fallback logging** |
| **7** | **Logging** | **Throughout** | **Track errors, health, and recovery states** |

**🧩 Optional: Visual Diagram Available**

**Let me know if you'd like a diagram (flowchart or architecture) for this pipeline.**

**Also, would you like me to generate:**

* **run\_cleaning.sh or .bat file template?**
* **Full llm\_cleaner.py with logger + safe row skipping?**