# Technical Documentation: Automated System for Scheduled Database Update and Notification

Document Version: 1.0

Date: September 12, 2025

Author: Team 21

Scenario ID: 2986420

Audience: Diverse bootcamp administrators and mentors, team 21 members responsible for maintenance.

## 1. Overview and Purpose

This document provides a comprehensive overview of the Scheduled Database Update and Notification automation system built on the Make.com platform. This system is designed to automate the process of querying Crypto data metrics(top 20 DEFI crypto by market cap) from coingecko api on a set schedule, updating the record Supabase and google sheet database,sending a notification via Gmail about the top trending DEFI crypto currencies among the top 20 Defi crypto my market cap, then finally create and interactive realtime dashboard on lovables using Lovables. The goal of this automation is to maintain up-to-date database records and ensure that key team members are notified of new or changed data without manual intervention.

## 2. Scenario Setup

### **2.1. Scenario Name and Location**

* **Name:** Scheduled Database Update and Notification
* **Folder:** Scenarios
* **URL:** [https://us2.make.com/1375106/scenarios/2986420/edit?showCheckRuns=true&showChangeLog=true]

### **2.2. Modules and Logic Flow**

The scenario is composed of the following modules, arranged in a sequential flow:

1. **Module 1: Cron-jobs**
   * **App:** cron-job.org
   * **Trigger/Action:** Schedule/instant
   * **Function:** This is the starting point of the workflow It triggers the entire workflow at a pre-defined interval (e.g., every day at 8:00 AM), or instant trigger during tests.
   * **Configuration:** Schedule is set to run once every day,configured with webhook url created and included into WEBHOOKS module in the scenario on [make.com](http://make.com)
2. **Module 2: Webhooks**
   * **App:** make.com (trigger Module)
   * **Trigger/Action:** Schedule/instant
   * **Function:** Recieves external trigger calls from [cron-jobs.org](http://cron-jobs.org) to trigger automation flow on make.com
   * **Configuration:** Webhook URL generated and incorporated into trigger app

**3. Module 2: Google Sheets**

* + **App:** make.com
  + **Trigger/Action:** Clear values from range
  + **Function:** This module retrieves specific rows of data from a designated Google Sheet based on a search criterion, then clears all data in range, this is to ensure only up to date data is present in database
  + **Configuration:** Spreadsheet ID: top 50 defi tokens, Worksheet Name: sheet 1, Range: A2:H50.

**4. Module 3: Http (via Webhook)**

* + **App:** make.com
  + **Trigger/Action:** Make a request
  + **Function:** This module request specific crypto data from coingecko api, data is retrieved as RAW JSON, the data from HTTP to the ITERATOR module, which then is processed into structured data and sent to Supabase and google sheet for storage, and Gmail module for notification.
  + **Configuration:** URL: [https://api.coingecko.com/api/v3/coins/markets?vs\_currency=usd&category=decentralized-finance-defi&order=market\_cap\_desc&per\_page=20&page=1&price\_change\_percentage=24h], Method: Get, Body type: JSON (raw), Content: {}", "Parse response": "Yes" }

**5. Module 4: Iterator**

**App:** make.com

**Trigger/Action:** Iterates Json data into structured data

**Function:** Maps each crypto data point from JSON file into each rows on on database

**Configuration:** Arrays: "{{data[]}}", mapping: Yes

**5. Module 5: Google Sheets**

* + **App:** make.com
  + **Trigger/Action:** Update a Row
  + **Function:** This module updates the record in the googlesheets database. The data from the iterator is mapped to the correct columns in the database.
  + **Configuration:** Spreadsheet ID: top 50 defi tokens, Worksheet Name: sheet 1, Range: A2:H50.

**6. Module 6: Supabase**

* + **App:** make.com
  + **Trigger/Action:** Update a Row
  + **Function:** This module updates the record in the Supabase database. The data from the iterator is mapped to the correct columns in the database.
  + **Configuration:** Spreadsheet ID: top 50 defi tokens, Worksheet Name: sheet 1, Range: A2:H50.

**7. Module 6: Text aggregator**

* + **App:** make.com
  + **Trigger/Action:** Aggregate text
  + **Function:** This module aggregates text file from iterator and converts to a suitable HTML format usable in gmail.
  + **Configuration:** Text: <tr>
  + <td><img src="{{17.image}}" height="30" width="30" alt=" {{17.name}} icon"></td>
  + <td> {{17.name}} </td>
  + <td>${{17.current\_price}} </td>
  + <td> {{17.price\_change\_percentage\_24h\_in\_currency}}%</td>
  + <td> {{17.market\_cap\_rank}} </td>
  + <td>${{17.market\_cap}} </td>
  + <td>${{17.total\_volume}} </td>
  + <td> {{17.last\_updated}} </td>
  + </tr>

**8. Module 8: Gmail**

* + **App:** Gmail
  + **Trigger/Action:** Send an email
  + **Function:** This module sends a summary email to a specified recipient, confirming that the data processing and database update were successful.
  + **Configuration:** To: "team\_leader@example.com", Subject: "Automation Success: New Data Processed", Content: "The record with ID {{2.ID}} has been successfully processed and updated in the Supabase database. The updated status is: {{4.status}}."

## 3. Logic Applied

The core logic of this scenario is as follows:

* **Data Ingestion:** The scenario is initiated by the **Cron-jobs** module, which runs on a schedule. This module triggers the rest of the workflow.
* **Data Retrieval:** The **HTTP** module then queries data from coingecko api and is retrieved in json format
* **Data Processing:** The scenario sends the data to the supabase and google sheet(update row )module via an **iterator** module. It then updates the leads table in **Supabase** and Google sheet with the data from the http
* .**Conditional Routing:** There is a condition whereby only trending tokens are sent via gmail notifications(24hrs price change greater than 2.5%
* **Error Handling:** The scenario includes a basic error-handling path. If the any module fails, the workflow will stop and mark the execution as incomplete for manual review.

## 4. Maintenance and Troubleshooting

Maintaining this automation system is crucial for its continued reliability. Follow these guidelines for ongoing management:

### **4.1. Best Practices**

* **Consistent Naming:** Do not change the name of the scenario or the modules. This ensures the documentation remains accurate.
* **Monitor Execution History:** Regularly check the Make.com History log for this scenario. Pay close attention to any failed or incomplete executions.
* **Check Connections:** Periodically verify that the connections for all apps (Google Sheets, Supabase, Gmail) are active and have the necessary permissions. Reconnect them if a token has expired.
* **Regular Testing:** If a major change is made to any connected app or to the scenario itself, perform a manual test run to ensure it functions as expected.

### **4.2. Common Issues and Solutions**

* **Scenario Fails with a Red Exclamation Mark:**
  + **Cause:** A module encountered an an error (e.g., API limit reached, missing required field, or an expired connection).
  + **Solution:** Click the red exclamation mark in the History log to view the error details. The error message will provide guidance on what went wrong. Address the issue (e.g., add a missing field, re-authenticate the connection, or wait for API limits to reset) and re-run the incomplete execution.
* **Data is not being processed as expected:**
  + **Cause:** This is often a filter or mapping issue.
  + **Solution:** Check the filters to ensure they are configured correctly. Verify that the dynamic values ({{...}}) are correctly mapped from the source module to the destination module. You can use the History log to inspect the data bundle at each step of the execution.
* **Unexpected Behavior after a change in a connected app:**
  + **Cause:** An API change in a connected app might have broken the scenario.
  + **Solution:** Check the app's documentation or status page for recent updates. You may need to update the module in Make.com to use a new API endpoint or a different field.

**Revision History**

| **Version** | **Date** | **Author** | **Description** |
| --- | --- | --- | --- |
| 1.0 | 2025-09-12 | Team 21 | Initial Document Creation |
|  |  |  |  |