

CASE STUDY · NO-CODE AUTOMATION · MAKE (INTEGROMAT)

# No-Code Lead Pipeline in Make:

## Automated Email Dispatch + Smart CRM Sync via Webhook

B2B / B2C · Lead Automation · January-February 2026

Two Make scenarios built, tested, and deployed with zero errors

2

Make (Integromat)

0

Google Sheets · Gmail

200

Webhook · Router

2s / 1s

Postman · API

# The business problem this solves

Small and mid-size businesses running email outreach face two operational bottlenecks that grow with scale: manually filtering which leads should receive an email based on their status, and keeping a CRM-style record current when new contact data arrives from external sources. Both tasks are repetitive, error-prone when done by hand, and consume admin time that should go toward revenue-generating work.

## The Data Source

A Google Sheet holds leads: email, subject, message, send\_status, send\_ts. New rows arrive manually or via external form submissions.

## The Gap

No system decides which rows should trigger an email. No deduplication when new data arrives. No automatic status logging after actions.

## The Goal

Build Make scenarios that handle filtering, dispatch, search, and sync automatically — with full audit trail and zero manual intervention.

**WHAT WAS BUILT:** Two independent Make scenarios that together cover the full lead processing cycle — from monitoring a data source to syncing records and dispatching emails. Each scenario includes a validation layer so only clean, complete records trigger actions.

# Three operational gaps that cost time and data quality

## Unfiltered Sends

100%

*of rows triggered email*

---

Emails were sent to all spreadsheet rows regardless of lead status — no condition check, no record of what was sent or when it happened.

## Duplicate Records



*risk of duplicates*

---

When a contact submitted data again through an external source, no system checked if they already existed — a new row was always added.

## No API Pre-validation

0

*endpoints tested upfront*

---

External data sources were connected to automations without prior testing, causing runtime errors when the response format didn't match.

**OBJECTIVE:** Replace all manual lead handling with two automated Make scenarios that filter, dispatch, validate, and sync data — reliably, without human intervention after initial setup.

# Two scenarios — one complete pipeline

## SCENARIO 1 — Conditional Email Dispatch with Status Logging



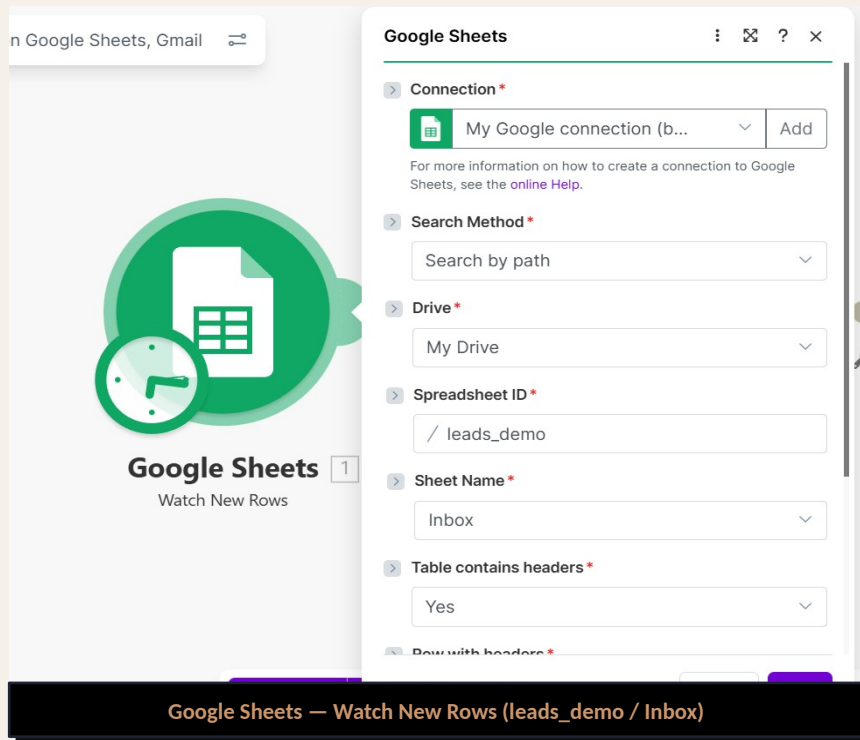
## SCENARIO 2 — Webhook-Driven Upsert: Update Existing or Create New Record



*KEY PRINCIPLE: Every action is conditional. No email is sent and no row is updated unless all filter conditions pass first.*

**SHARED FOUNDATION:** Google Sheets as lead database · Gmail for delivery · Postman for API validation before any Make connection

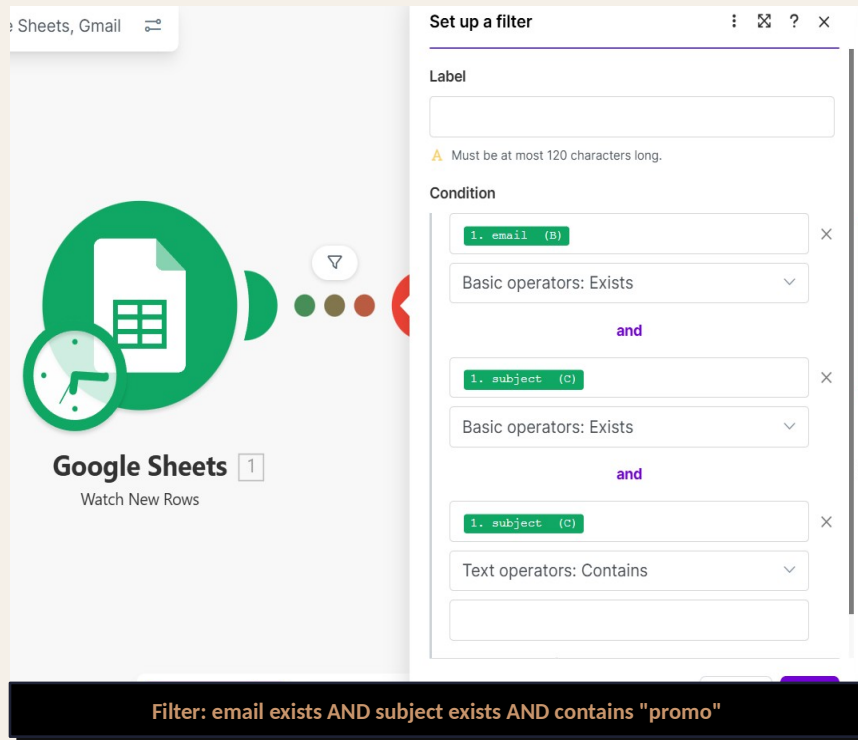
# Step 1: Watch new rows • Step 2: Apply 3-condition filter



The screenshot shows the 'Google Sheets' trigger configuration window. On the left is a large green icon of a document with a grid and a clock. The main configuration area includes:

- Connection \***: My Google connection (b...) [Add]
- Search Method \***: Search by path [v]
- Drive \***: My Drive [v]
- Spreadsheet ID \***: / leads\_demo
- Sheet Name \***: Inbox [v]
- Table contains headers \***: Yes [v]
- Row with headers \***: (partially visible)

At the bottom, a black bar contains the text: **Google Sheets — Watch New Rows (leads\_demo / Inbox)**



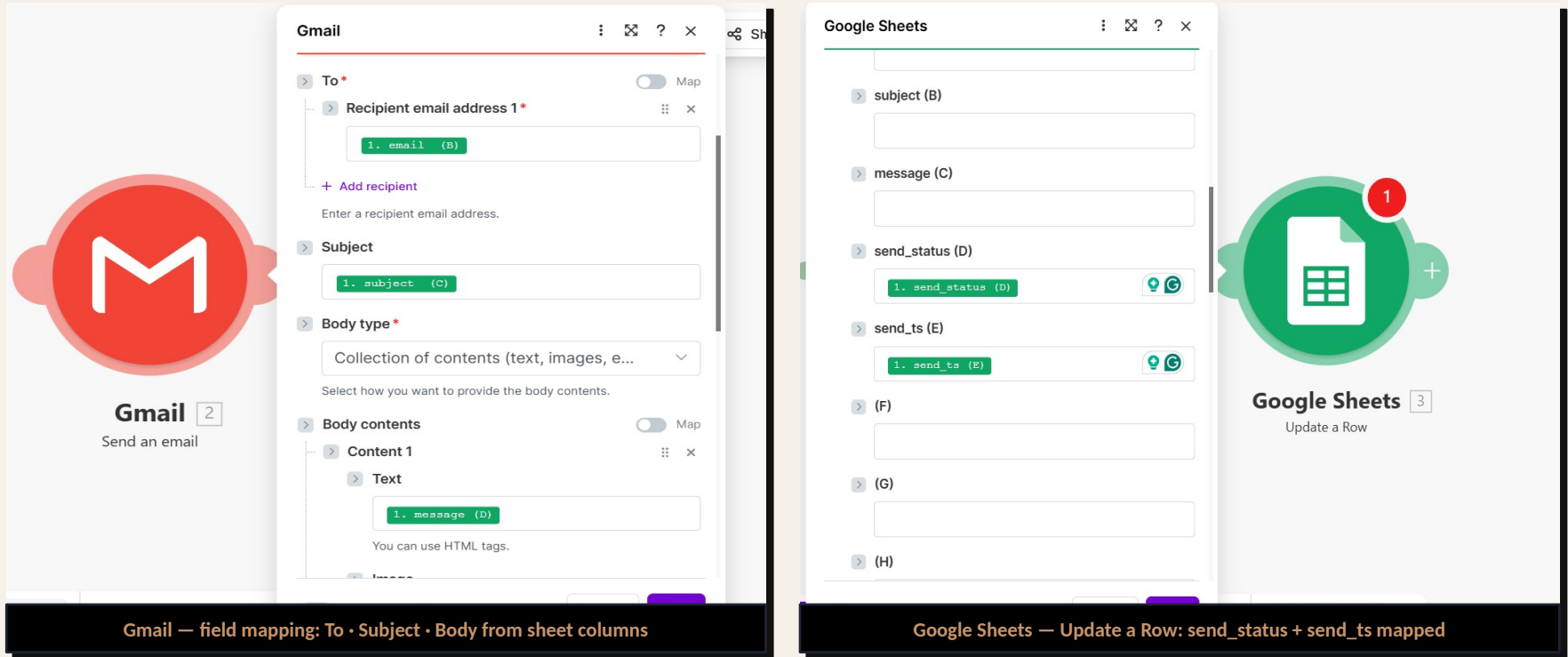
The screenshot shows the 'Set up a filter' configuration window. On the left is a large green icon of a document with a grid and a clock. The main configuration area includes:

- Label**: (empty text box)
- Condition**:
  - 1. email (B) [x] Basic operators: Exists [v]
  - and
  - 1. subject (C) [x] Basic operators: Exists [v]
  - and
  - 1. subject (C) [x] Text operators: Contains [v]

At the bottom, a black bar contains the text: **Filter: email exists AND subject exists AND contains "promo"**

Trigger monitors Inbox sheet for new rows. Filter passes only rows where email is filled, subject exists, AND subject contains the word "promo" (case insensitive).

## Step 3: Send email · Step 4: Write send\_status + timestamp



The image displays two side-by-side screenshots of web interfaces. On the left is the Gmail 'Compose' screen, and on the right is the Google Sheets 'Update a Row' dialog. The Gmail interface shows the 'To' field mapped to '1. email (B)', the 'Subject' field to '1. subject (C)', and the 'Body contents' field to '1. message (D)'. The Google Sheets interface shows the 'send\_status (D)' field mapped to '1. send\_status (D)' and the 'send\_ts (E)' field mapped to '1. send\_ts (E)'. A red 'M' logo is on the far left, and a green 'G' logo with a red '1' is on the far right. Below each screenshot is a black bar with white text explaining the field mappings.

**Gmail** 2  
Send an email

**Google Sheets** 3  
Update a Row

**Gmail — field mapping: To · Subject · Body from sheet columns**

**Google Sheets — Update a Row: send\_status + send\_ts mapped**

Gmail maps To → email (B), Subject → subject (C), Body → message (D). After send, Update a Row writes send\_status = "sent" and send\_ts = now() to the same row automatically.

# Google Sheets data · Make execution confirmed green

leads\_demo ☆ 📄

Файл Змінити Вигляд Вставити Формат Дані Інструменти Розширення Довідка

100% | грн. % 0.00 123 | Monts... | - 12 + | B I ↺ A 🗑️ 📄 📑

	A	B	C	D	E	F	G
		<b>email</b>	<b>subject</b>	<b>message</b>	<b>send_status</b>	<b>send_ts</b>	
1	1	borat8907@gmail.com	spring promo-10%	Hello!			
2	2	borat8907+1@gmail.com	VIP promo	Limited offer			
3	3	borat8907+2@gmail.com	Any subject	No email case			
4	4	borat8907+3@gmail.com		No subject case			
5	5	borat8907+4@gmail.com	Regular update	Just info			
6			promo test				
7			hello test				
8							
9							
10							
11							

leads\_demo spreadsheet — test rows with promo and non-promo subjects

Integration Google Sheets, Gmail

3 лютого 2026 P. 18:11:19

DIAGRAM HISTORY > 3 лютого 2026 P. 18:11:19 INCOMPLETE EXECUTIONS Cycle 1/1

Replay run

3 лютого 2026 P. 18:11:19

Run ID: 5d10b18899084f29b5fa0e91feb1518...

Run name: -

Trigger: Manual

Duration: 2 seconds

Operations: 5

Credits: 5

Data size: 1.3 KB

Source run: -

Simple log Advanced log

- Google Sheets - Watch New Rows 1 +0.4s  
The operation was completed.
- Gmail - Send an email 2 +1.1s  
The operation was completed.
- Google Sheets - Update a Row 3 +0.2s  
The operation was completed.
- Gmail - Send an email 2 +1.0s  
The operation was completed.
- Google Sheets - Update a Row 3 +0.2s  
The operation was completed.

Make History — 3 Feb 2026 · 2s · 5 ops · 0 errors

Test confirmed: rows 1 and 2 (contain "promo") passed filter → Gmail sent → row updated. Rows 3-5 (no promo or no email) were blocked. History shows all operations green, 2 seconds total.

# Testing external endpoints before connecting to Make

The screenshot shows a Postman interface for a POST request to `https://hook.eu1.make.com/9ab1ngxls0gcvkp93nuxggi3jji41qp`. The request body is a JSON object: `{ "email": "student+demo@example.com", "subject": "Test", "message": "Hello from webhook" }`. The response is `200 OK` with a status of `Accepted` and a response time of `750 ms`. A black banner at the bottom of the screenshot reads: **POST to Make Webhook — 200 OK · "Accepted" · 750ms**

The screenshot shows a Postman interface for a GET request to `https://api.open-meteo.com/v1/forecast?latitude=50.004&longitude=36.231...`. The response is `200 OK` with a status of `valid JSON` and a response time of `4.29 KB`. A black banner at the bottom of the screenshot reads: **GET Open-Meteo API — 200 OK · valid JSON · 4.29 KB**

API-first approach: all endpoints validated in Postman before any Make module is connected. Left: webhook POST returns 200 OK + "Accepted". Right: Open-Meteo GET returns structured JSON with hourly temperature data.

# Webhook → Search → Router: Update existing or Add new record

**Integration Webhooks** Replay run

DIAGRAM HISTORY > 10 ЛЮТОГО 2026 П. 16:46:39 INCOMPLETE EXECUTIONS Cycle 1/1

**10 ЛЮТОГО 2026 П. 16:46:39**

**Run ID:** 4d0910d37953435c982c82cb1c6c1...  
**Run name:** -  
**Trigger:** Manual  
**Duration:** 1 second  
**Operations:** 4  
**Credits:** 4  
**Data size:** 725.0 B  
**Source run:** -

**Simple log** **Advanced log**

- ✓ **Webhooks - Custom webhook** [1] +0.1s  
The operation was completed.
- ✓ **Google Sheets - Search Rows** [2] +0.5s  
The operation was completed.
- ✓ **Google Sheets - Update a Row** [4] +0.4s  
The operation was completed.
- ✓ **Google Sheets - Add a Row** [5] +0.8s  
The operation was completed.

**Make History — 10 Feb 2026 · 1s · 4 operations · both Router branches confirmed**

History confirms: Webhook (1) received POST → Search Rows (2) found contact → Router (3) chose correct branch → Update a Row (4) updated, Add a Row (5) added new. Both branches active. Zero errors.

# Google Sheets before and after webhook execution

leads\_demo 2 ☆ Збережено на Диску  
Файл Змінити Вигляд Вставити Формат Дані Інструменти Розширення Довідка

100% грн. % .00 123 За ум... - 10 + B I A

	A	B	C	D	E	F	G
1		email	subject	message	send_status	send_ts	
2	1						
3	2						
4	3						
5	4						
6	5						
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							

**BEFORE — empty table, no records yet**

leads\_demo 2 ☆ Збережено на Диску  
Файл Змінити Вигляд Вставити Формат Дані Інструменти Розширення Довідка

100% грн. % .00 123 За ум... - 10 + B I A

	A	B	C	D	E	F	G
1					stored	2026-02-10T14:59:46.220Z	
2		email	subject	message	send_status	send_ts	
3	1		Hello from webhook		stored	2026-02-10T14:59:45.960Z	
4	2						
5	3						
6	4						
7	5						
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							

**AFTER — row written: send\_status="stored" + timestamp**

Left: leads\_demo 2 before any webhook call — table empty. Right: after POST via Postman — row 1 appears with subject "Hello from webhook", send\_status = "stored", send\_ts = 2026-02-10T14:59:45.960Z.

# What was delivered and verified

2

Make scenarios  
built & confirmed

2s

Scenario 1  
execution time

1s

Scenario 2  
execution time

200

HTTP status  
all Postman tests

0

Errors in  
Make History

✓ **Zero unnecessary sends**

Filter blocks unqualified rows at the entry point — no emails go out to leads that don't match conditions

✓ **No duplicate records**

Router identifies returning contacts and routes to Update instead of creating a second entry

✓ **Full audit trail**

Every processed row gets send\_status and send\_ts automatically — no manual logging ever needed

✓ **Validated before connected**

All external endpoints tested in Postman prior to Make integration — eliminates the most common source of automation failures

✓ **Production-ready architecture**

Both scenarios are modular — Sheets → any CRM, Gmail → any email provider, webhook source → any form or platform

# Everything used to build and validate the pipeline

## Make (Integromat)

Main automation platform — both scenarios built and executed here

## Google Sheets

Lead database — stores email, subject, message, send\_status, send\_ts

## Gmail

Email delivery channel — mapped fields from spreadsheet columns

## Postman

API testing — validated webhook POST and Open-Meteo GET before Make connection

## Custom Webhook

Entry point for Scenario 2 — receives POST JSON from external source

## Router (Make)

Branching logic — routes to Update or Add based on Search Rows result

## Filter (Make)

Condition gate — ensures only complete, qualified records trigger actions

## Open-Meteo API

External REST API used for GET validation practice (200 OK confirmed)

*No paid tools, no custom code. The entire pipeline is built with free-tier services and no-code modules — demonstrating that production-grade automation is accessible without engineering resources.*