		
Laboratory 8: Smart Alert Logging with MariaDB and History Dashboard	School of Applied Digital Technology	
Name: Name:	ID: ID:	Section: Section:
Date:	Due date:	

Objectives

- Log all smart home events (Gas, Motion, Temperature, Light) into MariaDB.
- Enable Home Assistant Recorder and History visualization.
- Create a new Dashboard tab to visualize sensor trends and alert history.
- Verify data storage directly from phpMyAdmin.

1. Database Configuration (MariaDB)

- Your setup already includes the recorder integration in configuration.yaml ✅
- Make sure this section looks like this:

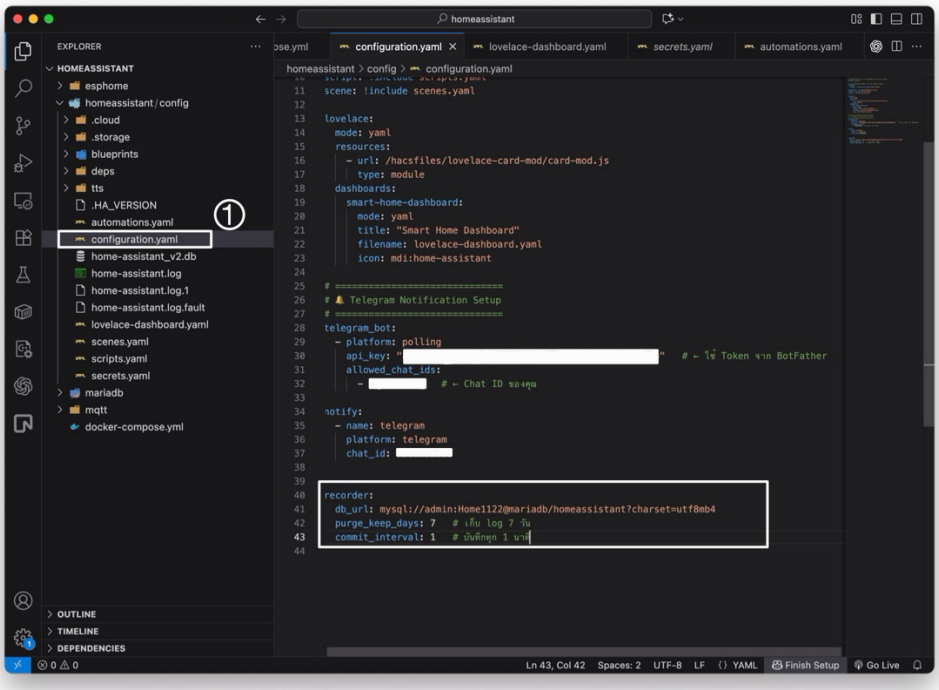
  Yaml:

```
recorder:
    db_url: mysql://admin:Home1122@mariadb/homeassistant?charset=utf8mb4
    purge_keep_days: 7
    commit_interval: 1
```

✅ Verify MariaDB Connection

1. Run docker ps to confirm the container mariadb is running.
2. Open phpMyAdmin in your browser ⟶ http://localhost:8080
    - Server: mariadb
    - Username: admin
    - Password: Home1122
    - Database: homeassistant
3. Check if you can see tables like states, events, recorder_runs.

If yes — Home Assistant is successfully logging data to MariaDB 🎉



2. Enable History and Logbook

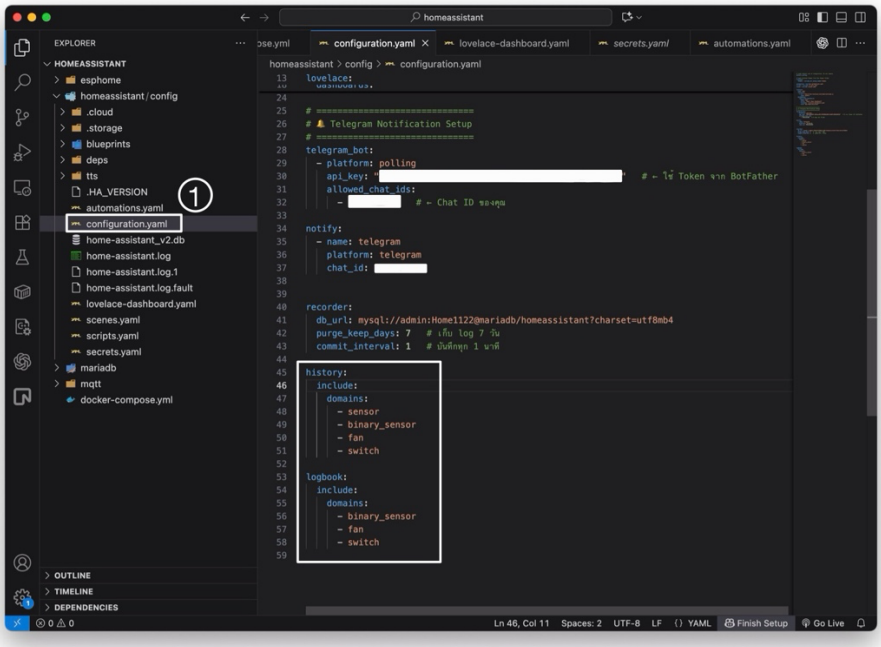
- Add or update the following lines in your configuration.yaml:

Yaml:

```
history:
  include:
    domains:
      - sensor
      - binary_sensor
      - fan
      - switch

logbook:
  include:
    domains:
      - binary_sensor
      - fan
      - switch
```

This ensures all device state changes and events are stored and displayed in the “History” and “Logbook” sections.



- Open: /config/lovelace-dashboard.yaml

  Then add this **new view** at the bottom (after your Lab 7 dashboard views):

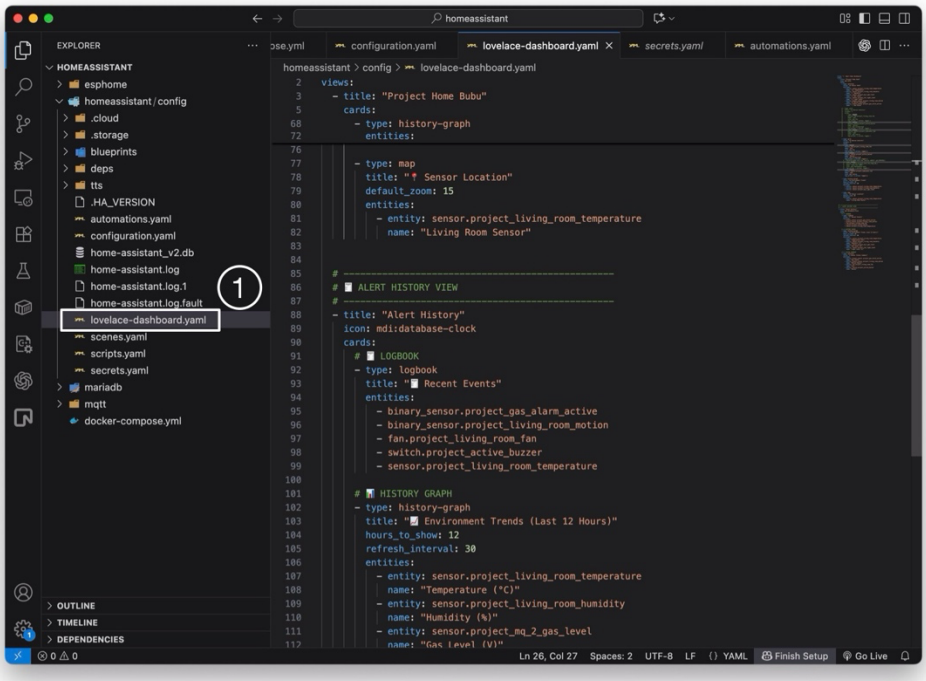
  Yaml:

```
- title: "Alert History"
  icon: mdi:database-clock
  cards:
    #  Logbook View
    - type: logbook
      title: "  Recent Events"
      entities:
        - binary_sensor.project_gas_alarm_active
        - binary_sensor.project_living_room_motion
        - fan.project_living_room_fan
        - switch.project_active_buzzer
        - sensor.project_living_room_temperature

    #  History Graphs
    - type: history-graph
      title: "  Environment Trends (Last 12 Hours)"
      hours_to_show: 12
      refresh_interval: 30
      entities:
        - entity: sensor.project_living_room_temperature
          name: "Temperature (°C)"
        - entity: sensor.project_living_room_humidity
          name: "Humidity (%)"
        - entity: sensor.project_mq_2_gas_level
```

```
      name: "Gas Level (V)"
    - entity: sensor.project_ldr_light_level
      name: "Light Level (V)"

# ⚙️ Sensor Status Summary
- type: entities
  title: "⚡ Sensor States Summary"
  entities:
    - entity: binary_sensor.project_gas_alarm_active
      name: "Gas Alarm"
    - entity: binary_sensor.project_living_room_motion
      name: "Motion Sensor"
    - entity: fan.project_living_room_fan
      name: "Fan"
    - entity: switch.project_active_buzzer
      name: "Buzzer"
```



4. Test and Verify

- Simulate Sensor Triggers
- Go to Developer Tools ⟶ States and manually change values:

Test Case	Entity	New State	Expected Result
Gas alert	binary_sensor.project_gas_alarm_active	on	Telegram alert sent
High temperature	sensor.project_living_room_temperature	35	Telegram alert sent
Motion alert	binary_sensor.project_living_room_motion	on	Telegram alert sent
Light alert	sensor.project_ldr_light_level	1.0	LED ON (if configured)

Then check:

1. Dashboard ⟶ Alert History shows your recent events.
2. History Graph updates every 30 seconds with real-time data.
3. phpMyAdmin ⟶ states table updates with new entity states.

5 (Optional). Export Data from MariaDB

- If you'd like to export logs into a .csv report:

Step 1: Open phpMyAdmin

- Go to **http://localhost:8080**
- Select the database ⟶ **homeassistant**

Step 2: Create a "View" that joins both tables

- In the **SQL** tab, run this command:

SQL:

```
CREATE OR REPLACE VIEW v_states_full AS
SELECT
    sm.entity_id AS entity_id,
    s.state AS state,
    s.last_changed AS last_changed
FROM states AS s
JOIN states_meta AS sm
    ON s.metadata_id = sm.metadata_id;
```

✅ Once executed, you'll see a new view called v_states_full in the sidebar under the "Views" section.

Step 3: Query the sensors you want to export

- Run this SQL command:

SQL:

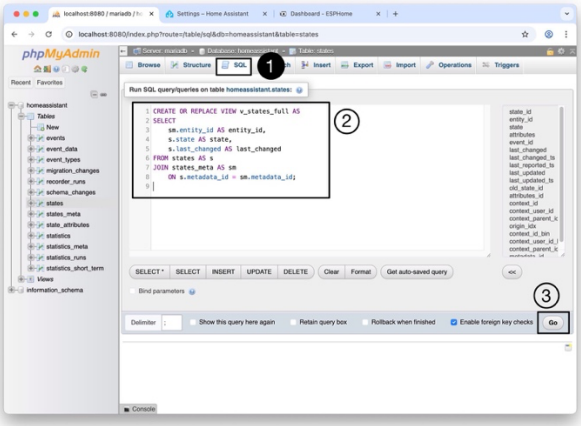
```
SELECT
    entity_id,
    state,
```

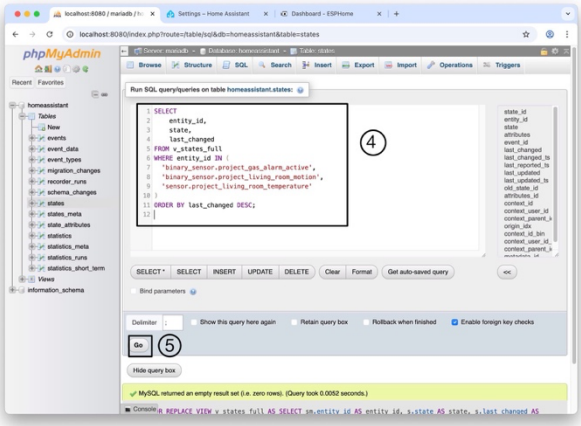
```
    last_changed
FROM v_states_full
WHERE entity_id IN (
  'binary_sensor.project_gas_alarm_active',
  'binary_sensor.project_living_room_motion',
  'sensor.project_living_room_temperature'
)
ORDER BY last_changed DESC;
```

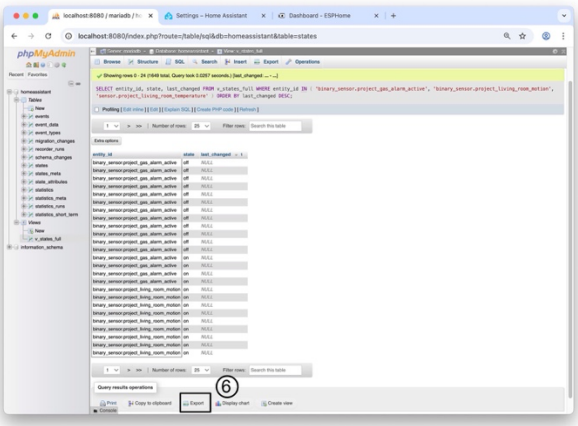
Step 4: Export to CSV

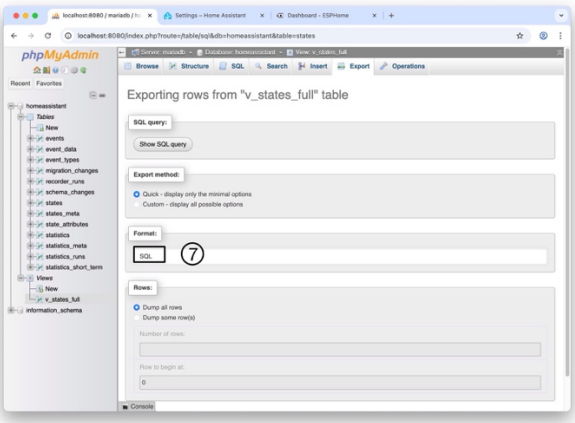
- Click Export at the top of phpMyAdmin
- Choose CSV as the format
- Set a file name (e.g. sensor_log_report.csv)
- Click Go ✅

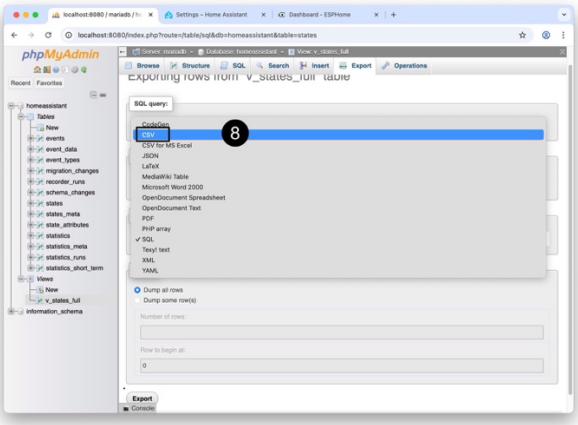
You'll get a downloadable CSV file that can be opened in Excel or Google Sheets.

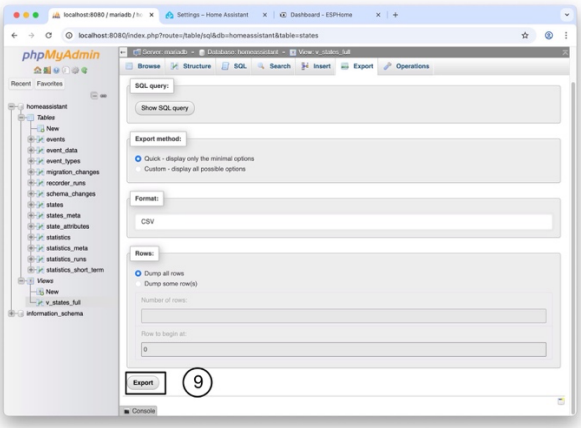












Continued on the next page.

Take 1: Correlate Events and States

Objective

To understand how events and states in Home Assistant are related and to analyze how system activities (events) trigger changes in sensor states.

You will use SQL to find correlations between event records and state changes within a defined time window.

Requirements

1. Use the MariaDB database connected to Home Assistant (homeassistant schema).

2. Focus on these key tables:

• events — stores all system-level actions (automation triggered, service called, etc.)

• states — stores entity states and sensor readings.

• states_meta — maps metadata_id to the entity_id name.

3. Write a query to:

• Detect event-state pairs that happened within ±60 seconds of each other.

• Filter only for binary_sensor.project_gas_alarm_active entity.

• Show the event_type, event timestamp, state change timestamp, and entity_id.

4. Analyze the output — explain what kind of events usually happen when the gas alarm is triggered.

Database Relationships Overview

• states.metadata_id ↔️ states_meta.metadata_id ⟶ provides the entity_id name.

• events.time_fired_ts ⟶ timestamp (double precision, Unix time format).

• states.last_changed_ts ⟶ timestamp (double precision).

• You can use ABS(events.time_fired_ts - states.last_changed_ts) to measure time difference in seconds.

Code:

----- Have a good day -----

Answer:

```
SELECT
    sm.entity_id,
    s.state,
    FROM_UNIXTIME(s.last_changed_ts) AS state_time,
    e.event_type,
    FROM_UNIXTIME(e.time_fired_ts) AS event_time,
    ROUND(ABS(e.time_fired_ts - s.last_changed_ts), 1) AS time_diff_sec
FROM homeassistant.events AS e
JOIN homeassistant.states AS s
    ON ABS(e.time_fired_ts - s.last_changed_ts) <= 60
JOIN homeassistant.states_meta AS sm
    ON s.metadata_id = sm.metadata_id
WHERE sm.entity_id = 'binary_sensor.project_gas_alarm_active'
ORDER BY s.last_changed_ts DESC
LIMIT 15;
```