Gas Monitor Documentation

By: Garret Gallo

Written on: July 10, 2025

Section 1: Introduction

H_2S (Hydrogen Sulfide) is a dangerous gas that can often be found in sanitary sewer manholes, with one of the primary causes being the breakdown of organic matter. OSHA defines 10ppm to be the threshold for when H_2S is considered hazardous, and as these levels rise, significant problems can arise. Once H_2S reaches these hazardous levels, it can corrode sewer infrastructure, disturb the surrounding environment with a horrible smell, and can even be deadly.

While I was at JMT, our client was getting several complaints from residents that there were horrible smells coming out of manholes, and that they wanted us to investigate to see if high levels of H_2S were the cause. JMT conducted these studies by purchasing monitors that would detect H_2S inside manholes, and were placed inside selected manholes. Then every week, I would go out to manually collect the data, to then finally observe the data and see if I could notice any trends.

The goal of this project was to create a data engineering pipeline that would extract data from the monitor's program, perform transformations on that raw data, and then be uploaded to a MySQL server where it could finally be pulled in Power BI for detailed visualization.

The reasoning behind the creation of this ETL pipeline was due to the severely inefficient manual method via Excel that was going to be used. While Excel is more than efficient for smaller workloads but as data grows, especially from multiple monitors, this methodology becomes almost impossible to work with manually. With a one-shot pipeline, this data could be aggregated and transformed in several more ways and at almost 10x the speeds without sacrificing quality, while being able to produce more appealing and detailed visualizations that couldn't be done on Excel.

Section 2: Process + Final Design

While I will describe my thought process in more detail, the image below serves as an overview of the final ETL pipeline that was used to create that final result.

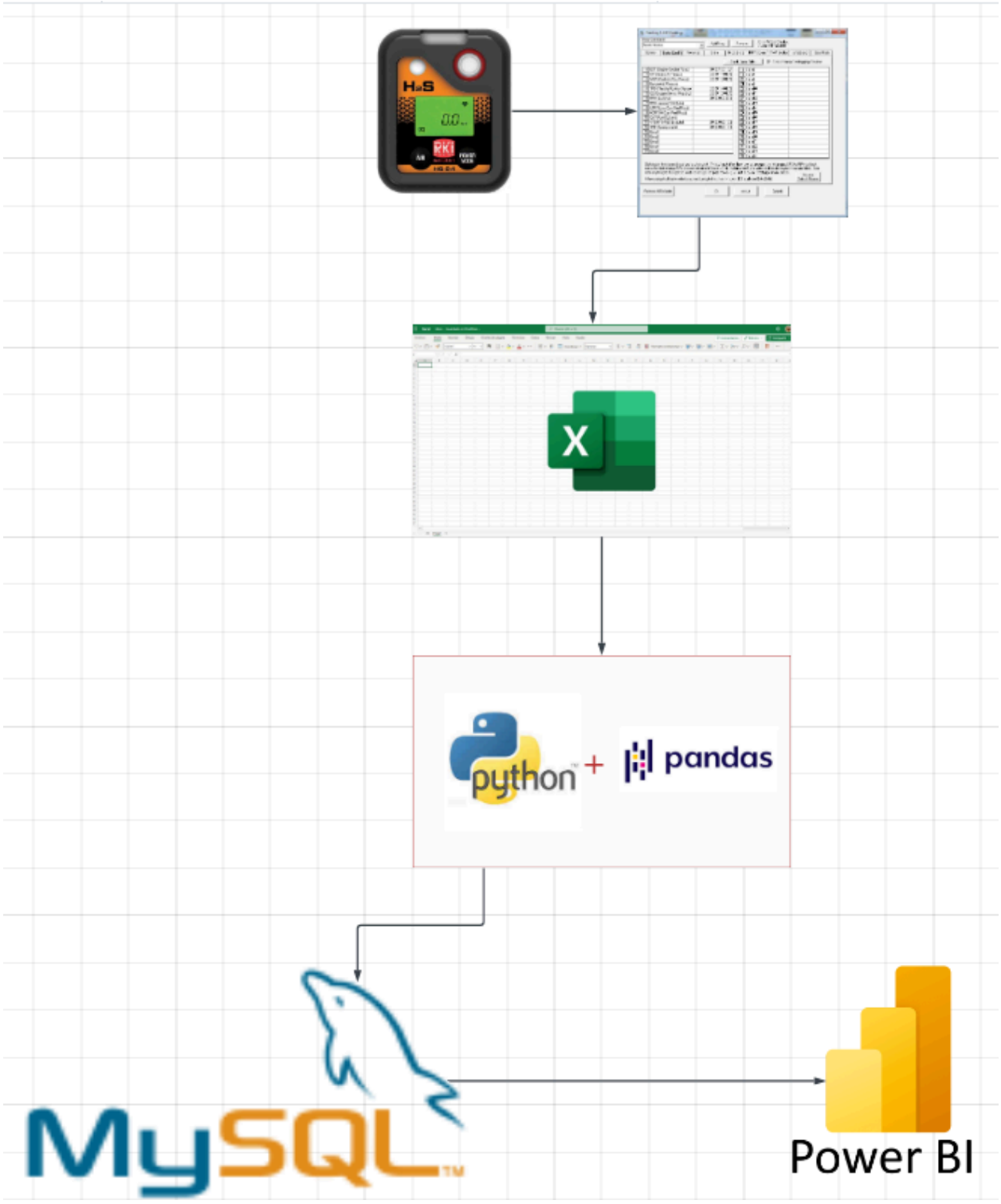

Figure 1: ETL Pipeline

Section 2.1: Extraction

In terms of extraction, it was a relatively straightforward process. Every week, I would connect the monitor to my laptop via an IrDA sensor so that the data could be uploaded to the 03-Datalogging-Series program. Once that was completed for all monitors, I would download that week's data in a CSV file (each monitor separately) and then transform all the data into a master raw Excel file.

Unfortunately, there was no way to remotely connect to the monitor or pull from an API, which added steps to the extraction process. Additionally, when extracting data to and from the datalogging program, each monitor had to be done separately, which made the process more cumbersome.

Section 2.2: Transformation

For the purposes of this project, I initially saved all extracted data from all monitors into a raw master Excel file. From there, I could simply use pandas to perform data transformations.

For this project, I elected to calculate an average, 75% and max H_2S on a daily, weekly and monthly basis. My reasoning was the daily breakdown would be used for the main charts down the line, while the weekly and monthly basis breakdowns could paint a more concise picture of trends over a long stretch of time. I also calculated the percentage of all measurements that fell into three categories—safe (ppm < 4), detectable (4.1–9.9 ppm), and hazardous (ppm > 10). These figures provide a clear, actionable view of how the data is distributed across key safety thresholds. Finally, using .describe() to create an overview of all data for each monitor.

With all of these transformations, all useful and insightful data were calculated, and could be used down the pipeline for visualization.

Section 2.3: Load and Visualization

Finally, once all my aggregations were complete, I uploaded the final data to MySQL. The following schematic is the outline for the relational database.

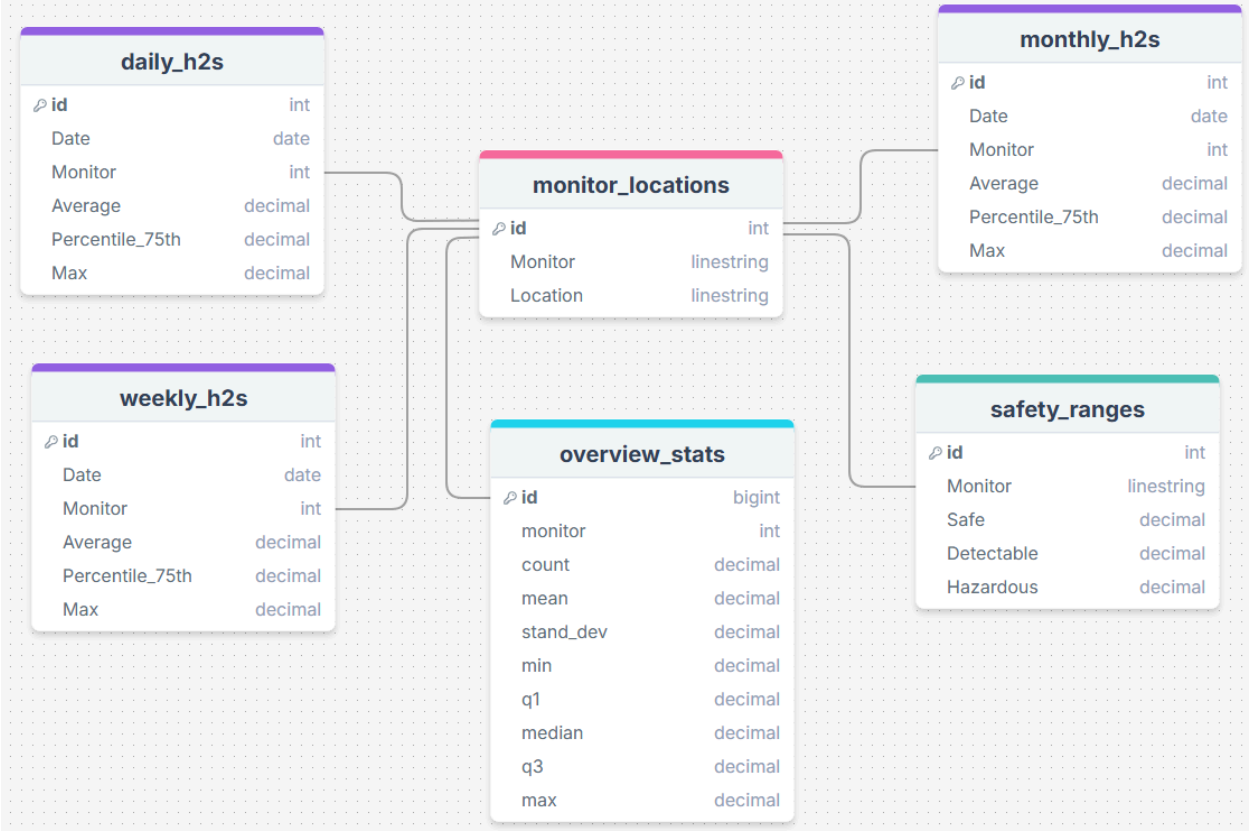

Figure 2: MySQL database schema

First, I decided to use monitor locations as the central table that all other tables would connect to. Monitor is ID of the monitor, while location is the manhole that the monitor is located in (ie 325-123). Next, I created daily, weekly, and monthly H_2S tables that contained data from ALL monitors. Next, I created tables for overview_stats and safety_ranges, which once again contained data from ALL monitors. All of these tables had a monitor column as a foreignkey used to connect to the monitor_locations primarykey column ID.

Section 3: Future Improvements

While the goal of this project was successfully completed - there are definitely a handful of improvements that could be made to improve efficiency, durability, and the visualization of the final product.

1. Using a NoSQL Database for the raw data
   While there might not be anything inherently wrong with storing the extracted data in an Excel file, using a NoSQL database like MongoDB would be a more preferable option.

   The upside to using MongoDB would be that it can handle high volumes of data, and Excel can get extremely lagging or even corrupted if the file becomes too large. MongoDB also includes better querying and indexing, especially as the database grows to considerable sizes. Finally, MongoDB also has seamless integration with Python, making it an extremely reliable choice.

2. Better extraction methodologies
   Previously, I mentioned that unfortunately the only way to extract the data was to do so manually every week. While still functional, ideally, there would be some remote way to connect to the monitors for easier extraction. For this to be a reality, a more expensive monitor would have to be purchased, but it would be needed if the project were to be expanded to hundreds of monitors.

3. Automation of the Pipeline
   Once again, data extraction of this pipeline was done on a weekly basis. Using an orchestration service like Apache Airflow could greatly improve the efficiency of this process, as it could be scheduled to run on a weekly basis.

   Additionally, if running locally is too intensive for the machine, especially if the project was expanded to hundreds of monitors, using a CSP like AWS in conjunction with MWAA could be an alternative to a larger-scale version of this project.