e-ISSN: 2686-2573
DOI: 10.21512/emacsjournal.v8i1.13532

Web-Based Quality Control Dashboard Design for Data Validation and Monitoring: A Case Study of BMKG Instruments

Kartika Purwandari[1*], Brian Tirafi Aufauzan[2], Join Wan Chanlyn Sigalingging[3]

[1]Computer Science Department, School of Computer Science,
Bina Nusantara University,
Jakarta, Indonesia 11480

[2]Information Technology Department,
Gadjah Mada University,
Yogyakarta, Indonesia 55281

[3]Centre for Database,
The Agency for Meteorology, Climatology, and Geophysics,
Jakarta, Indonesia 10720
kartika.purwandari@binus.ac.id; briantirafiaufauzan@mail.ugm.ac.id;
join.wan.chanlyn@bmkg.go.id

*Correspondence: kartika.purwandari@binus.ac.id

Abstract – *Accurate meteorological data are vital for the operational activities of the Agency for Meteorology, Climatology, and Geophysics (BMKG), specifically for weather forecasting and disaster mitigation. However, Automatic Weather Station (AWS) instruments frequently encounter sensor degradation and technical malfunctions, which compromise data validity. Traditional manual validation is inefficient and prone to human error. This study addresses these gaps by designing a web-based Quality Control (QC) dashboard for real-time AWS data monitoring. Developed using the Laravel framework and PostgreSQL, the system integrates Leaflet.js and Chart.js for interactive spatial and analytical visualization. Using the Agile Scrum methodology, the development process was iteratively refined across eight sprints. Implementation results show a significant improvement in data validation accuracy and a reduction in potential human error. User Acceptance Testing (UAT) with fifteen BMKG specialists confirms high usability, with the system receiving "Strongly Agree" ratings for its efficiency in real-time monitoring and reporting. The practical implications include enhanced data credibility for national climate modeling. This paper concludes that while the dashboard streamlines workflows, future iterations should incorporate automated anomaly detection algorithms. Limitations include a current reliance on static validation thresholds, suggesting a need for machine learning integration in future research.*

Keywords: *BMKG; Automatic Weather Station (AWS); Quality Control (QC); Data Visualization; Agile Scrum*

I. INTRODUCTION

Meteorological data plays a crucial role in disaster mitigation and climate modeling. At Meteorology, Climatology, and Geophysics Agency (BMKG), observation instruments like the Automatic Weather Station (AWS) are the backbone of data collection (Faniriantsoa & Dinku, 2022). While prior studies have explored the reliability of specific AWS sensors, there remains a significant gap in the availability of integrated, real-time platforms for simultaneous monitoring and multi-parameter validation. Conventional methods often rely on periodic manual checks, which

fail to catch transient sensor failures (Purwandari, Siga, and Sigalingging, 2024). While previous studies by Purwandari et al. (2024) focused on QC in specific regions like DKI Jakarta, there is a lack of integrated platforms covering the seven types of AWS instruments (AWS, ARG, AAWS, ASRS, IKRO, AWSSHIP, and SOIL) simultaneously. This research fills the gap by providing a centralized, automated validation system that replaces time-consuming manual processes (Handiana, Novianti, and Sanusi, 2020). The performance of over 1,200 AWS instruments across Indonesia is critical for public safety. For instance, technical issues such as communication disruptions or extreme environmental conditions in remote regions often lead to missing or invalid data codes (codes 1 through 9), which can skew regional climate reports if not caught immediately (AlSalehy & Bailey, 2025).

However, maintaining the validity of observational data presents significant challenges. Technical issues such as sensor degradation, communication disruptions, extreme environmental conditions, and the absence of real-time monitoring systems often result in data invalidity or loss. Conventional manual validation processes are time-consuming and prone to human error, which may affect the timeliness and credibility of the information delivered to stakeholders (Afif and Meganendra, 2023).

In meteorology, Quality Control (QC) encompasses a range of techniques to ensure data integrity and detect anomalies or instrument malfunctions (Patro & Bartakke, 2025). Data validation on AWS typically involves methods such as range checks (ensuring values fall within reasonable thresholds), step checks (identifying sudden changes), and consistency checks (verifying inter-parameter relationships). Based on the combination of these methods, the QC system classifies data into nine categories, ranging from valid data (code 0) to missing data (code 9).

To address these challenges, a system capable of real-time monitoring, automated data validation, and informative visual presentation is essential. One promising solution is the development of a web-based QC dashboard. By leveraging modern web technologies, such a dashboard can enhance data accessibility, streamline validation processes, and improve responsiveness to anomalies in instrument performance.

This study proposes the design and implementation of a web-based Quality Control dashboard specifically for AWS instruments operated by BMKG. The system is developed using the Laravel framework, known for its security features and ease of integration with database systems. To manage the complex and large-scale meteorological data, PostgreSQL is employed as the database management system. Data visualization is facilitated through the use of Leaflet.js for interactive geographic mapping and Chart.js for dynamic validation charts.

The state-of-the-art in QC involves automated range and consistency checks; however, these are often siloed within disparate database systems. The novelty of this study lies in the integration of the Extract-Transform-Load (ETL) pattern with a modern web-based dashboard. This approach provides a unified interface for nine distinct validation categories (ranging from valid to missing data) while offering interactive geographical mapping through Leaflet.js.

Through this integrated system, users can monitor AWS performance in real time, identify data inconsistencies, perform efficient validation, and generate comprehensive reports. The dashboard also includes filtering features based on instrument type, meteorological parameters, and time range, making it a practical tool for both operational and analytical purposes.

Furthermore, the system is developed using the Agile methodology, specifically the Scrum framework (Hron & Obwegeser, 2022), to ensure iterative development and continuous improvement based on user feedback and operational requirements (Bhavsar, Shah, & Gopalan, 2020; Al-Saqqa, Sawalha, & AbdelNabi, 2020). The main objective of this research is to enhance BMKG's internal workflow and contribute to the provision of high-quality data for weather and climate information services at both national and regional levels.

II. METHODS

The system architecture follows the Extract-Transform-Load (ETL) pattern to ensure structured data handling. The development adopts the Agile Scrum framework, characterized by iterative cycles called Sprints to maintain flexibility.

2.1. Development Method

This project adopts the Agile software development methodology using the Scrum framework. Scrum enables iterative and incremental progress through a series of time-boxed development cycles known as Sprints. Each Sprint aims to deliver a potentially shippable product increment, allowing for rapid feedback, adaptability, and continuous improvement. The use of Scrum ensures greater flexibility and responsiveness to changes in project requirements or stakeholder feedback.

The Scrum process in this project involves the following key ceremonies and components:

- **Product Backlog:** A set of tasks selected for completion during the Sprint.
- **Sprint Planning:** Defines the goals and tasks for the upcoming Sprint based on Product Backlog priorities.
- **Daily Scrum:** Short daily stand-up meetings to synchronize team progress and address any obstacles.
- **Sprint Review:** A presentation and evaluation of the work completed at the end of the Sprint.
- **Sprint Retrospective:** A reflection session to identify lessons learned and areas for improvement in the next Sprint cycle.

To ensure a structured data processing workflow, the system is designed based on the Extract-Transform-Load (ETL) architectural pattern (Mahmud & Ikbal, 2022). This pattern supports systematic handling of meteorological data from acquisition, transformation (including validation), to visualization. Modern web technologies are employed on both backend and frontend components to ensure that the resulting application is responsive, secure, and scalable (Ravi, 2025).

In the context of AWS instrument data processing, the system implements nine types of validation codes, which are stored in the central database. These validation codes are assigned based on the quality control checks applied to the incoming data:

0: Valid Data
1: Range Check
2: Step Check
3: Consistency Check
4: Range Check + Step Check
5: Range Check + Consistency Check
6: Step Check + Consistency Check
7: Range Check + Step Check + Consistency Check
9: Missing Data

These validation results serve as the basis for subsequent data classification, visualization, and reporting in the QC dashboard.

2.2. System Design

The system is designed as a web-based application that integrates a PostgreSQL (Hartono and Erfina, 2021, Schönig, 2020) database for storing AWS instrument data, a Laravel-based backend for processing (Bhagaskoro, Al Makky, & Huda, 2025), and a responsive frontend using Tailwind CSS (Kodali, 2024), Leaflet.js for mapping (Abdillah, Nawangnugraeni, & Yuniarto, 2021), and Chart.js for data visualization (Benbba, 2021).

Figure 1. Visualization of the previous Quality Control (QC) project developed using Power BI

The system architecture follows the Extract-Transform-Load (ETL) pattern:

- Extract: Data from various AWS instruments is collected and forwarded to the backend.
- Transform: Raw data is validated, flagged, and transformed into user-friendly metrics.

- Load: Processed data is stored and made available through the frontend interface.

2.3. Scrum Backlog

In Scrum, a backlog is a list of work to be completed in system development and is dynamic according to the needs of the project. The backlog is divided into two, namely Product Backlog and Sprint Backlog.

Product Backlog contains all the desired features, changes, and improvements in the product, while Sprint Backlog is a part of the Product Backlog that is selected and broken down into a task to be done in one sprint.

The Product Backlog includes essential features such as:

- User Authentication
- Data Processing and Flagging
- Interactive Station Maps
- Data Visualization Charts
- Dynamic Filtering
- PDF Export
- UI and Mobile Responsiveness

Each Sprint is dedicated to specific feature development. For instance, Sprint 1 focuses on authentication, Sprint 2 on data processing, and so on. Each task is time-boxed and tracked to ensure timely delivery.

2.4. User Acceptance Test

User Acceptance Test where the results of the system that can be used are given to users to carry out the operation process in accordance with the planned business process. Users will be asked to provide feedback regarding the feasibility of the system with several questions (Priyatna, Hananto, & Nova, 2020). Users fill out a questionnaire with five answer indicators, namely: Strongly Disagree, Disagree, Disagree, Agree and Strongly Agree.

III. RESULTS AND DISCUSSION

The results of this paper will be a web-based quality control dashboard designed to monitor and validate data from BMKG instruments. To achieve this result, we will be using the scrum framework. Our scrum team consists of 3 people in total, in which we will follow the scrum framework steps, including making the product backlog, conducting daily sprint meetings, performing sprint reviews, and holding sprint retrospectives.

3.1. Research Flowchart

As shown in Figure 2, the research followed a structured pipeline starting with requirements gathering via the Product Backlog, followed by iterative development through eight Scrum Sprints, and concluding with User Acceptance Testing (UAT).

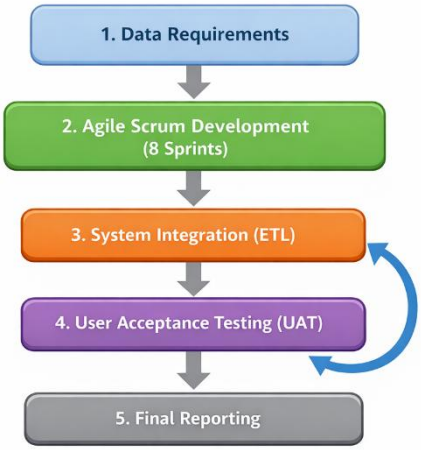

Figure 2. Research Flowchart

3.2. Product Backlog

In this step, we will communicate with the product owner to identify what features the web-based quality control dashboard should include to effectively monitor and validate data from BMKG instruments.

Table 1 lists the application development backlog categorized by priority and estimated time (ET - Estimated Time) in working days. Each feature is assigned with an ID for easy tracking during the development process.

Table 1. Product Backlog Table

ID	Backlog	Priority	ET (day)
F1	User Authentication (Login and Sign Up)	High	7
F2	Data Processing Flagging Percentage	High	5
F3	Interactive Map (Station Appearance)	High	8
F4	Data Visualization in Charts	High	6
F5	Filtering Data on the Website	Medium	7
F6	PDF Report Download	Medium	7
F7	Cohesive UI Development	Low	3
F8	Mobile Screen Responsiveness	Low	1

- High Priority Features

These features are key components of the system, such as user authentication, data processing, graphical visualisation, and interactive display of station maps. Due to the importance of these functionalities, these features are scheduled to be completed first.

- Medium Priority Features

These features support user interaction and system completeness, such as data filtering and PDF report download. Although not a core part, these features are still important to the user experience.

- Low Priority Features

These features improve aesthetics and user comfort, such as the development of a cohesive UI and responsiveness of displays on mobile devices. They are of lower priority than the main features.

The total estimated time to work on all features is 44 working days, which can be divided into several sprints based on Agile development methods.

3.3. Sprint

A sprint is a fixed-length event of 9 weeks. Each sprint will have a daily meeting, where the team will discuss the progress based on the sprint backlog. But before we advance to the sprint step, we will do the sprint planning first to decide what the team should accomplish in each sprint and determine how long each sprint will take based on the complexity of developing features for the quality control dashboard.

Sprint 1

1. Project Settings and Version Control (GitHub) (3 days)
2. Model, Controller and Database User Creation (2 days)
3. Authentication and Middleware Routing (2 days)

Sprint 2

4. Analysis Structural Table Data (2 days)
5. Data Processing with Table View (3 days)

Sprint 3

6. Integration of Leaflets into Project (3 days)
7. Pin Station Creation on Map (2 days)
8. Making Pop Up Pin (3 days)

The implementation of the QC dashboard provides immediate visual clarity regarding station health. As illustrated in Figure 3, the station distribution map allows operators to identify regional outages instantly. Unlike previous Power BI-based solutions shown in Figure 1, the new Laravel-based dashboard offers superior filtering capabilities.

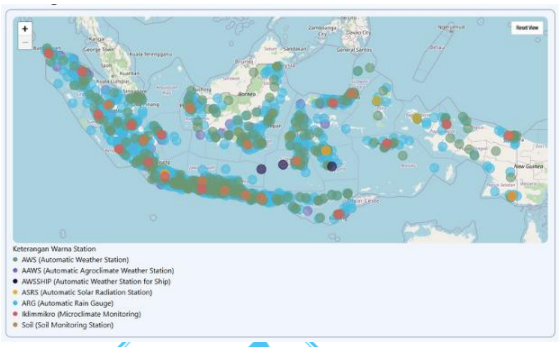

Figure 3. BMKG AWS station distribution map

Sprint 4

9. Chart.JS Integration to Project (2 days)
10. Processing Data into Chart-Appropriate Form (2 days)
11. Chart Implementation on Website Interface (2 days)

Sprint 5

12. Dynamic Filter Dropdown based on Columns in the Database (Reading Type, Instrument, and Province) (3 days)
13. Implementation of Filtering by Date (2 days)
14. Integration of Filters with Data Displayed on the Front-End (2 days)

Sprint 6

15. Generate PDF template creation result (2 days)
16. Integration of Frontend Data Capture and PDF Download (3 days)
17. Integrasi Button Download PDF Data Station pada Pop Up (2 days)

Sprint 7

18. Low Fidelity Design (1 day)
19. UI Implementation on Front-End Web (2 days)

Sprint 8

20. Updating UI to Responsive UI (1 day)

3.4. Sprint Review & Restrospective

Sprint Review will be a process of checking the work results of the completed Sprint. The Scrum Team will present the results of the work to policy makers and convey the achievement towards the Product

Goal. This process will be an opportunity to evaluate what work can be done next. Sprint Retrospective that aims to improve the effectiveness and quality of the Scrum Team.

Table 2. Sprint Review and Retrospective Table

Sprint Number	Review	Retrospective
Sprint 1	Project setup on GitHub and user authentication features was successfully completed.	The initial setup process went smoothly; the team was able to harmonize the workflow quickly.
Sprint 2	Analysis of the table structure and data display in tabular form was successfully carried out.	There was a little confusion regarding the table format, but it was quickly resolved.
Sprint 3	Leaflet maps were successfully integrated, including station pin placements and pop-ups.	There was a delay due to the adjustment of the marker display style on the map.
Sprint 4	The Chart component is successfully integrated and the chart can be displayed.	The chart display needs colour and scale adjustments to make it more informative.
Sprint 5	The data filtering feature based on columns and dates was successfully implemented and integrated into the interface.	There was a slight problem with the dropdown logic, but it was resolved through teamwork.
Sprint 6	The PDF report creation and download feature was successfully performed through a pop-up.	The compatibility issue with the PDF button on the front-end was resolved after testing.
Sprint 7	The low fidelity design was successfully implemented to the web interface.	The alignment of UI elements required several revisions.
Sprint 8	The user interface was successfully updated to be responsive.	Slight customisation is required for mobile viewing.

Table 2 shows the review and retrospective results of each sprint that was conducted during the dashboard development process. Sprint 1 focused on project initialisation, versioning on GitHub, and creating user authentication features, and went smoothly with aligned teamwork. Sprint 2 involved analysing the data structure and displaying the data in tables, which was completed well despite some confusion.

In Sprint 3, the integration of the Leaflet map with station markers and pop-ups was carried out, but there was a delay due to pin design adjustments. Sprint 4 completed the integration of Chart.js with the result that the chart could be displayed, although visual adjustments were needed.

Sprint 5 focused on the dynamic filtering feature based on several parameters and was effective after minor logic improvements. Sprint 6 successfully implemented PDF creation and download on station pop-ups. Sprints 7 and 8 focused on user interface development, from low fidelity design to improved display responsiveness for mobile devices, with satisfactory results.

3.5. Testing

The User Acceptance Test (UAT) was conducted involving end-users from BMKG responsible for AWS data monitoring and validation. The indicators used in the UAT are as follows:

- Q1. The system is designed in alignment with the operational workflow of AWS data monitoring and validation.
- Q2. The system meets user expectations and functional requirements in ensuring data quality.
- Q3. Users find the QC dashboard interface easy to operate.
- Q4. The system assists staff in efficiently performing validation and reporting of AWS data.
- Q5. The information system improves time efficiency in QC processes and meteorological data reporting.
- Q6. The data displayed by the system is consistent with field observations and actual measurement results.
- Q7. The system facilitates real-time monitoring and evaluation of AWS instrument performance.

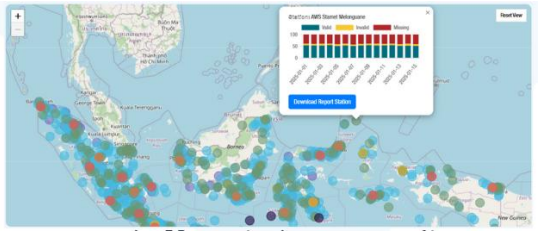

Figure 4. Appearance of specific station data

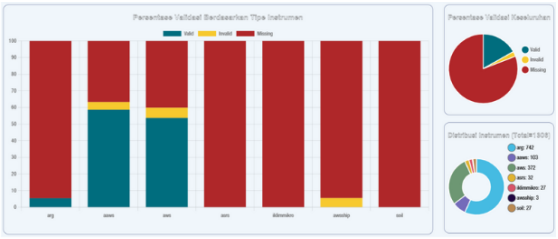

Figure 5. AWS data validation graphs and diagrams

Based on user responses to the User Acceptance Test, the data is summarized as follows:

Table 3. Recap of User Acceptance Test Results (Individual Level)

Question	0	1	2	3	4
Q1	0	0	0	8	7
Q2	0	0	2	8	5
Q3	0	0	0	6	9
Q4	0	0	0	4	11
Q5	0	0	0	5	10
Q6	0	0	1	9	5
Q7	0	0	0	3	12
Total	**0**	**0**	**3**	**43**	**59**

Notes : 0 = Strongly Disagree, 1 = disaggre, 2=somewhat disagree, 3 = agree, 4 = strongly agree

Table 3 indicates that the User Acceptance Test was conducted with fifteen participants from BMKG's relevant divisions. Based on their respective roles, the system received positive feedback, as evidenced by responses marked “Agree” and “Strongly Agree” across all indicators.

As shown in Figure 5, the system provides a clear breakdown of "Valid," "Invalid," and "Missing" data across all station types. For example, the high percentage of "Missing" data (Red) in certain categories indicates a need for immediate field maintenance. Table 2 provides a retrospective of the development, highlighting that while initial UI alignments required revisions, the logic for data filtering (Sprint 5) and PDF generation (Sprint 6) was successfully integrated to meet BMKG's operational standards. The UAT results in Table 3 demonstrate that the system’s greatest impact is on "Real-time monitoring" (Q7), which received 12 "Strongly Agree" responses. This confirms that moving from static Power BI reports (Figure 1) to a dynamic Laravel environment provides the responsiveness required for meteorological services.

IV. CONCLUSION

The development of a web-based Quality Control (QC) dashboard system for BMKG’s Automatic Weather Station (AWS) instruments significantly enhances users' ability to interpret and validate observational data in real time. By integrating automated validation techniques and interactive visualizations, the system supports the maintenance and improvement of data reliability for meteorological and climatological, observations. This solution aligns with operational workflows and is designed using an Agile methodology with the Scrum framework, ensuring iterative refinement and responsiveness to user feedback. The development of the web-based QC dashboard significantly enhances BMKG’s ability to validate observational data in real-time using automated techniques. By leveraging the Scrum framework, the project successfully delivered features ranging from interactive maps to dynamic PDF reporting.

Despite its success, the study is limited by its reliance on historical threshold data for range checks. Future research should explore the integration of AI-driven anomaly detection to identify more subtle sensor drifts. Additionally, expanding the dashboard to include mobile-native notifications for urgent instrument failures would further improve response times for BMKG field technicians.

REFERENCES

Abdillah, M. Z., Nawangnugraeni, D. A., & Yuniarto, A. H. P. (2021). Geographic information system (GIS) for mapping greenpark using leaflet JS. JTIK (Jurnal Teknik Informatika Kaputama), 5(2), 259-266.

Afif, M., & Meganendra, D. A. (2023). Predictive Maintenance for Automatic Weather Station (AWS) Based on Anomaly Detection Using Autoencoder: A Literature Review. Journal of Computation Physics and Earth Science (JoCPES), 3(2).

AlSalehy, A. S., & Bailey, M. (2025). Improving Time Series Data Quality: Identifying Outliers and Handling Missing Values in a Multilocation Gas

and Weather Dataset. *Smart Cities*, *8*(3), 82.

Al-Saqqa, S., Sawalha, S., & AbdelNabi, H. (2020). Agile software development: Methodologies and trends. International Journal of Interactive Mobile Technologies, 14(11).

Benbba, S. (2021). Comparison of D3. js and Chart. js as visualisation tools. Science and Engineering.

Bhagaskoro, A., Al Makky, M., & Huda, H. (2025, February). Development of a Laravel-Based Backend Application Server to Support Farm Animal Information Management at Sein Farm Bandung City. In 2025 International Conference on Advancement in Data Science, E-learning and Information System (ICADEIS) (pp. 1-6). IEEE.

Bhavsar, K., Shah, V., & Gopalan, S. (2020). Scrum: An agile process reengineering in software engineering. International Journal of Innovative Technology and Exploring Engineering, 9(3), 840-848.

Faniriantsoa, R., & Dinku, T. (2022). ADT: The automatic weather station data tool. *Frontiers in Climate*, *4*, 933543.

Handiana, D., Novianti, R., & Sanusi, H. (2020). PENGUJIAN KUALITAS DATA AUTOMATIC WEATHER STATION (AWS) MENGGUNAKAN RELIABILITY DAN CONSISTENCY TEST PADA SISTEM AWS CENTER. Jurnal Widya Climago, 2(2).

Hartono, N., & Erfina, E. (2021). Comparison of stored procedures on relational database management system. Tech-E, 4(2), 8-15.

Hron, M., & Obwegeser, N. (2022). Why and how is Scrum being adapted in practice: A systematic review. Journal of Systems and Software, 183, 111110.

Kodali, N. (2024). Tailwind CSS Integration in Angular: A Technical Overview. International Journal of Innovative Research in Science Engineering and Technology, 13(16652), 10-15680.

Mahmud, D., & Ikbal, M. Z. (2022). The role of etl (extract-transform-load) pipelines in scalable business intelligence: A comparative study of data integration tools. *ASRC Procedia: Global Perspectives in Science and Scholarship*, *2*(1), 89-121.

Patro, B. S., & Bartakke, P. P. (2025, March). Quality Control (QC) and Quality Assurance (QA) Procedures for Meteorological Data from Automatic Weather Stations. In *2025 4th International Conference on Range Technology (ICORT)* (pp. 1-6). IEEE.

Priyatna, B., Hananto, A. L., & Nova, M. (2020). Application of UAT (User Acceptance Test) Evaluation Model in Minggon E-Meeting Software Development. Systematics, 2(3), 110-117.

Purwandari, K., Siga, F. A., & Sigalingging, J. W. (2024, July). Quality Control of Automatic Weather Station Data: Study Case in DKI Jakarta. In 2024 IEEE 10th International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA) (pp. 98-102). IEEE.

Ravi, C. (2025). ETL (Extract, Transform & Load) Automation. *International Journal of Emerging Trends in Computer Science and Information Technology*, *6*(1), 52-55.

Schönig, H. J. (2020). Mastering PostgreSQL 13: Build, administer, and maintain database applications efficiently with PostgreSQL 13. Packt Publishing Ltd.